Predicting Stock Market States by Machine Learning Methods

***Abstract* – Traditional financial time series prediction methods commonly operate binary labeling by exploring price returns. In this paper, we develop eight approaches utilizing technical indicators and unsupervised models to construct classification into three market states - bull, bear, or static. We introduce an extended multi-dimensional time series analysis. To mitigate the limitations of manual labeling, we implement several supervised learning models and also explore unsupervised learning, including Hidden Markov Model and Gaussian Mixture Model. As a result, labels derived from Hidden Markov Model combined with market state prediction of XGBoost algorithm shows the superior performance of our trading strategies compared to naive buy-and-hold strategy.**

***Keywords* – Stock Market State, Time Series, Machine Learning, Deep Learning, Trading Strategy**

I. INTRODUCTION

Nowadays, technology has become an asset in finance and machine learning gains a competitive advantage in high-frequency trading when dealing with large data volumes. Investors are always searching for the best strategy to grab market trends and excess profits. Decades ago researchers focused on applying statistical models, such as regression and time series models (ARIMA, GARCH), to study stock prices. Considering that the stock market essentially holds a dynamic, nonlinear, and noisy nature, it fails to meet the assumptions of traditional models. Thus, investors turn to predict the "hidden" states (bear, bull, or static) of the market to instruct the trading process.

In our paper, starting with elaborating all eight methods defined to label market states, we present the process of feature engineering with a list of independent variables. Furthermore, covering supervised, unsupervised, and deep learning models, we select six algorithms to predict hidden market states. Finally, by trading with our strategy and evaluating through both computational and financial measures, we figure out the best strategy model which could optimize profits and beat the benchmark buy-and-hold strategy. The trading technique proposed in our research could contribute a lot to machine learning applications in the trading field.

II. DATA

A. Russell 3000 Index

Russell 3000 Index, a benchmark of the US stock market, has been holding an upward trend since 2003, growing almost five times its value. It experienced two major decreases in 2008 and 2020, due to the financial crisis and Covid-19. As figure 1 below here shows, the trend is in general upwards. Our analysis takes into account the daily Open, High, Low, Close price of Russell 3000 from 2003 to 2010 and calculates returns to study trends.

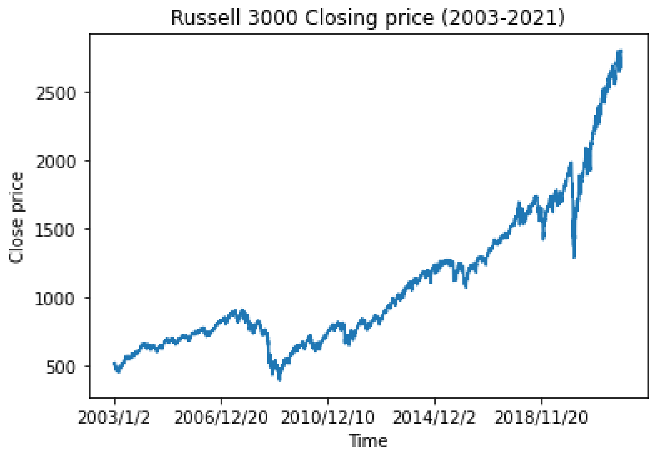


Fig. 1. Close price of Russell 3000 (2003-2021)

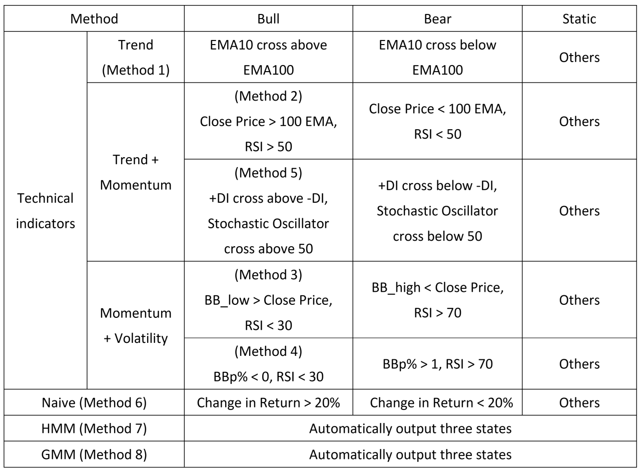
B. Data Labeling

1) Definition: Supervised learning deeply depends on the classification label in the training dataset, which in this case is the classification of bull, bear, or static market state. After considering an enormous amount of existing approaches to determine market states, we try eight methods under three major categories to achieve a good prediction. The first is a naive approach, simply looking at stock returns. The second approach uses combinations of technical indicators. The last approach is based on the classification results of unsupervised models Hidden Markov Model (HMM) and Gaussian mixture model (GMM).

For the naive approach, we intuitively use the daily returns with different thresholds to determine the market state. The optimal threshold value is 20% annualized return through research [1].

Secondly, five definitions using combinations of technical indicators are constructed. These combinations are derived from three technical indicator categories, i.e. trend, momentum, and volatility. As Russell 3000 is traded periodically, we forgo the volume indicators. Trend and volatility indicators are lagging indicators while momentum indicators are leading indicators. Therefore, it is beneficial to consider both when making trading decisions, with lagging indicators to exploit the trend and leading indicators to predict the future.

Thirdly, aside from manually establishing market state definitions, we later discovered that the states defined by unsupervised machine learning models could produce satisfying trading results. Thus, HMM and GMM predictions are also adopted as training datasets for other supervised learning models. Figure 2 is a general description of the eight labeling methods we used.



Method		Bull	Bear	Static
Technical indicators	Trend (Method 1)	EMA10 cross above EMA100	EMA10 cross below EMA100	Others
	Trend + Momentum	(Method 2) Close Price > 100 EMA, RSI > 50	Close Price < 100 EMA, RSI < 50	Others
		(Method 5) +DI cross above -DI, Stochastic Oscillator cross above 50	+DI cross below -DI, Stochastic Oscillator cross below 50	Others
	Momentum + Volatility	(Method 3) BB_low > Close Price, RSI < 30	BB_high < Close Price, RSI > 70	Others
		(Method 4) BBp% < 0, RSI < 30	BBp% > 1, RSI > 70	Others
Naive (Method 6)		Change in Return > 20%	Change in Return < 20%	Others
HMM (Method 7)		Automatically output three states		
GMM (Method 8)		Automatically output three states		

Fig. 2. Labelling methods

2) Definition Filtering: To obtain an overall understanding and preliminary filter for the above eight methods, we evaluate them from the views of both trading results and label balance. The trading results come from our strategies, which will be explained in detail in the following section. If the PnL for Y using label definition already performs poorly, even the 100% accuracy of a supervised model is hard to make much profit. So, we need to select the classifying definition which generates the higher PnL as our labels for supervised models. The results of the market state trading strategy show that methods 1, 5, 7, 8 own a great PnL, while 2, 3, 4 did not. Particularly, methods 7 and 8 generate higher PnL than the buy-and-hold strategy on an average of 111.80%.

Imbalance classification issues occur commonly in financial time series prediction. We find that the number of static states is much higher than bull and bear states, especially in technical indicator labeling methods. To avoid fake high accuracy with just one most-frequent category during training, we also prefer more balanced labels to ensure model sensitivity for supervised machine learning. Among all eight labeling methods, method 2 has the best balance with 19.1% bear, 17.6% static, and 63.3% bull market states. Thus, although the plain PnL of method 2 is low, we still reserve it in our final selection to see how it will perform in models.

In conclusion, we choose methods 1, 2, 5, 7, 8 to construct the model labels.

C. Features Engineering and Selection

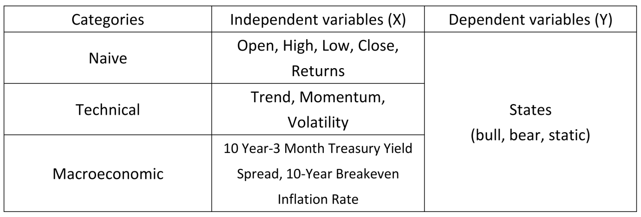
1) Feature Engineering: After labeling states, to find independent variables containing effective information for predicting, we evolved our research from univariate to multivariate time series analysis. Therefore, we firstly engineer a feature list with 85 features which can be classified into three types, detailed as follows:

The first is the historical price data of Russell 3000, which includes OHLC price (open, high, low, close) and adjusted close price. We have also calculated daily returns.

The second type of feature is macroeconomic economic indicators. Macroeconomic indicators drive broader market sentiment, which in turn affects individual stock prices to varying degrees. We follow the results obtained by Chen [2], who discovered that term spreads and inflation rates were the most useful predictors for recessions in the stock market. 10 Year-3 Month Treasury Yield Spread (T10Y3M) and 10-Year Breakeven Inflation Rate (T10YIE) are selected as macroeconomic indicators daily.

The third type of independent variable is technical indicators. Technical indicators are effective in spotting price patterns and predicting future trends. All 78 technical indicators are computed daily with data previous to that day to prevent information leakage.

2) Feature Selection: Given that numerous independent variables are considered, it is effective to extract some of the most important features as representatives. We firstly delete features whose values are mostly zero. During operating, for data using specific technical indicator labels in section 1,)we will exclude those indicator features that have been used in determining states. Also, given that many technical indicators are from the same category, they have similar measures. We apply Random Forests to generate several sub-features lists based on feature importance, which prepares for later tests in our models. The five most important features are Relative Strength Index (RSI), Percentage Price Oscillator (PPO), Bollinger Band Percent (BBp%), Donchian Channel Percentage Band (DCP), and Positive Directional Indicator (+DI). Different models will adopt different sub-feature lists based on their respective characteristics. Here in figure 3, we show the structure of our final datasets.



Categories	Independent variables (X)	Dependent variables (Y)
Naive	Open, High, Low, Close, Returns	States (bull, bear, static)
Technical	Trend, Momentum, Volatility	
Macroeconomic	10 Year-3 Month Treasury Yield Spread, 10-Year Breakeven Inflation Rate	

Fig. 3. Labelling methods

III. MODELING

As we have defined different methods to generate our labels and different combinations of features, we then proceed to put our data into models. With the advent of big data and the rapid development of artificial intelligence techniques, recent studies have turned to machine learning and deep learning techniques, which showed outstanding performance in processing time-series data as these models have no prerequisite Ruof linearity, stationarity, homoscedasticity, or normality. Multivariate time series and classification requirements have increased the complexity of our research problem. Thus, our group is going to use unsupervised, supervised, and deep learning models while exploring the effect of different datasets, training periods, and model hyperparameters.

A. Model Selection

Among popular models, we specifically chose six models which all have their own superiority in solving multivariate financial time series classification problems:

1. Unsupervised: Hidden Markov Model (HMM), Gaussian Mixture Model (GMM)
2. Supervised: Random Forest (RF), XGBoost
3. Deep Learning: Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP)

1) Unsupervised machine learning:

- Hidden Markov Model (HMM)

Hidden Markov Model is a black box model that automatically classifies label and calculate the transition matrix between states. In the below graph 4, we assume:

X_t	Market states.
O_t	Observable (prices and indicators).

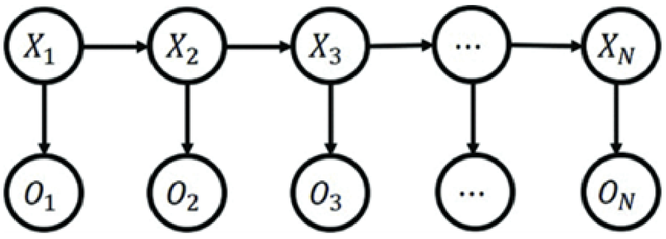


Fig. 4. HMM demonstration

HMM provides the following advantages over other selected models. First, we do not need to pre-label our features since it is unsupervised learning. Due to simplicity, we do not need data pre-processing or grid search, which saves time. Also, it performs well on time series data which we have. We also find HMM to prevent over or under-fitting. For example, HMMs have been used to solve problems in earthquake forecasting, machine-based speech translation, and even in financial markets with some positing that hierarchical HMMs are responsible for some of Renaissance Tech's successful trading algorithms [3].

- Gaussian Mixture Models (GMM)

GMM is homogeneous to HMM while GMM is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the co-variance structure of the data as well as the centers of the latent Gaussian's [4]. Without creating initialized transition and emission probabilities, it could help to find the most appropriate transition probabilities and emission probabilities.

2) Supervised machine learning:

- Random forest (RF)

Random Forest is a classifier composed of multiple decision trees, whose structure makes it an expert to reduce bias and produce robust results. Random forest uses the most voted prediction of individual decision trees as a modeling result. The randomness, as implied in its name, refers to the Bagging (bootstrap and aggregating) of samples and random selection of features in each tree. This model allows for the elimination of bias and works well with a large number of features, which is suitable for our case.

- XGBoost

XGBoost is also a good choice of model for our problem while it is similar to Random Forest. XGBoost stands for extreme gradient boosting, which is more advanced than Random Forest [5]. The first advantage XGBoost has is that it saves run-time since it measures the tree with similarity score before entering the model. Also, XGBoost performs better for unbalanced datasets because it improves the weight of labels during execution. Thus, XGBoost is coherent to our imbalanced dataset.

3) Deep learning: Deep Learning methods like Neural Networks have been widely used to predict the performance of financial time series. Long Short-Term Memory (LSTM), a specific Recurrent Neural Network designed to solve gradient disappearance issues in the training period, stands out for its efficiency and accuracy in long-term dependent time series data [6]. This is because LSTM networks have gates and different tensor operations that can learn what information to add or remove to the hidden state. Besides, Multi-layer Perception (MLP) is also a sensitive model to deal with sequential data and financial classification issues and is used to compare with LSTM.

B. Strategy

We have implemented a benchmark buy-and-hold strategy and a market state strategy to evaluate the performance of our predicted labels in the testing period 2018-2021. The buy-and-hold strategy buys Russell 3000 index at the beginning of the trading period and holds until the end of the trading period. In contrast, the market state strategy trades based on the daily prices and market state prediction of that day. We trade with the assumption that there is no transaction cost. On each day, the trading strategy is as follows.

When state prediction is 1, which refers to a bull market, we close the previous order if we currently hold a sell order, or create a buy order of 1 unit if not. If we already hold a buy order, we check if it has been exactly 5 days since the execution of this buy order, which would be a sign for us to add another unit to pursue more profit. And if it has been 15 days since the buy order execution, we would close the order to prevent loss. Vice versa for state prediction -1, a bear market. If market state prediction is 0 (static), we hold our current position and make no additional movement.

In our market state strategy, we rarely have a period where we are out of the market as that would require the previous position to be 0 and state to be 0 at the same time, which only happens at the beginning of our trading period or after the loss prevention 15 days has passed.

C. Model Evaluation

For model evaluation, we examined our models by accuracy and profitability. We trained our model in 3 different time periods: 2003-2017, 2010-2017, 2015-2017 to investigate the best length of the training period. After fitting, we applied our model to the testing (trading) period of 2018-2021 to predict the market state and then actively trade Russell 3000. For supervised models, we applied accuracy, weighted f1 score, and confusion matrix to rank model efficiency.

For unsupervised models HMM and GMM, we

visualize market state predictions along with actual stock prices to evaluate. We choose the most frequent predicted market state from February to March 2020 to be the bear state, based on the stock market crash after growing instability caused by the COVID-19 pandemic, then we specify the most frequent predicted market state after April 2020 to be the bull state, which is validated by growing market price afterward. The rest predicted market states is given to the static state.

Later we compared trading results of different models. We calculate PnL, total compound return (rtot), Sharpe ratio, Calmar ratio, variability-weighted return (vwr), and maximum drawdowns of each trading. In general, we have adapted both model measurements and trading measurements to optimize our models.

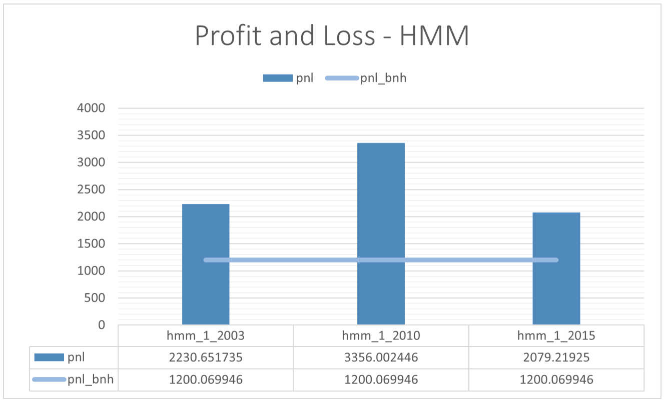
D. Model Results

1) Unsupervised machine learning: In this section, we tuned 2 unsupervised models with 8 different selections of features and 3 training period lengths. We now present our comprehensive results. It turns out that HMM and GMM with only Russell 3000 daily returns as input, which is labeling definition 1, achieved the best results. As the table 5 shows out 8 feature selection methods. The model with the most profits is HMM with only

#	Features
1	Returns
2	Returns + Open/High/Low/Close(OHLC)
3	Returns + OHLC +Technical
4	Returns + OHLC +Technical
5	Returns + OHLC +Technical + Economic
6	Returns + OHLC +Technical + Economic
7	Returns + Economic
8	Returns +Technical + Economic

Fig. 5. HMM features selection list

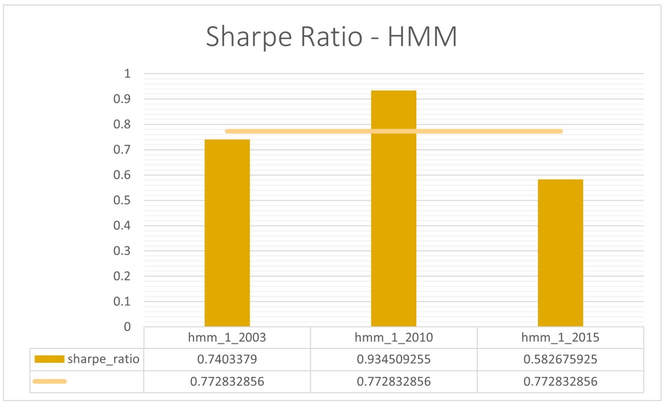
Russell 3000 daily returns as input and training period 2010-2017. It achieved the best PnL and 0.93 Sharpe ratio in the final strategy, which earns 35.14% more than the second profitable model 6 and 7 significantly more than the buy-and-hold strategy. As the figure and below shows, all HMM models with labeling definition 1 achieved better results in terms of profitability than the buy-and-hold strategy, regardless of the length of the training period. HMM models only achieved



	hmm_1_2003	hmm_1_2010	hmm_1_2015
pnl	2230.651735	3356.002446	2079.21925
pnl_bnh	1200.069946	1200.069946	1200.069946

Fig. 6. Profit and Loss of HMM model.

high Sharpe ratio in training period of 2010-2017, which it results above the benchmark, as we can see in figure 7. In general, the trading result of HMM gives high profitability but appears to be more risky.

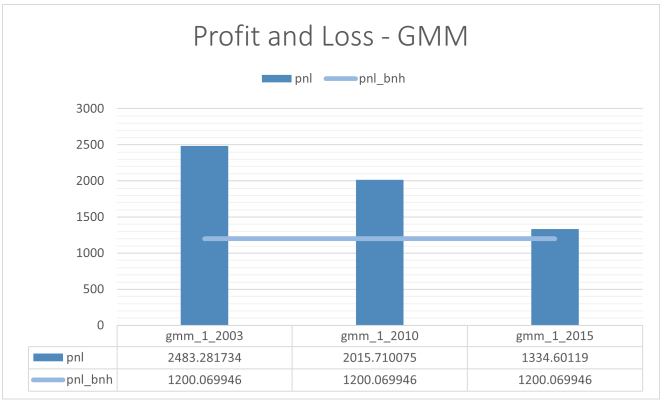


	hmm_1_2003	hmm_1_2010	hmm_1_2015
sharpe_ratio	0.7403379	0.934509255	0.582675925
	0.772832856	0.772832856	0.772832856

Fig. 7. Sharpe Ratio of HMM model.

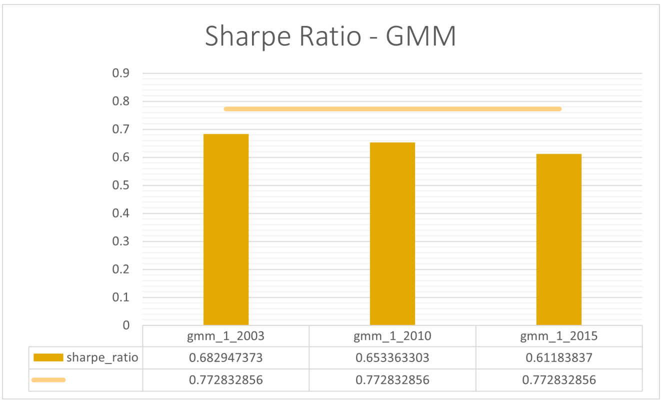
GMM with only returns as input achieved its best PnL using training sets of 2003-2017 and 2010-2017, twice the PnL of 2015-2017, which indicates that GMM requires a longer training period to learn the state transition pattern and to generate more precise state predictions. However, the Sharpe ratio of GMM models are not over bench mark. As we have demonstrated in figure 8 and figure 9, the PnL and Sharpe ratio of GMM model trading results. Grid Search for parameters in the two models hardly affects the overall performance, because the most important parameter is the n components - the number of states we want to predict.

The outstanding performance of HMM derives



	gmm_1_2003	gmm_1_2010	gmm_1_2015
pnl	2483.281734	2015.710075	1334.60119
pnl_bnh	1200.069946	1200.069946	1200.069946

Fig. 8. Profit and Loss of GMM model.



	gmm_1_2003	gmm_1_2010	gmm_1_2015
sharpe_ratio	0.682947373	0.653363303	0.61183837
	0.772832856	0.772832856	0.772832856

Fig. 9. Sharpe Ratio of GMM model.

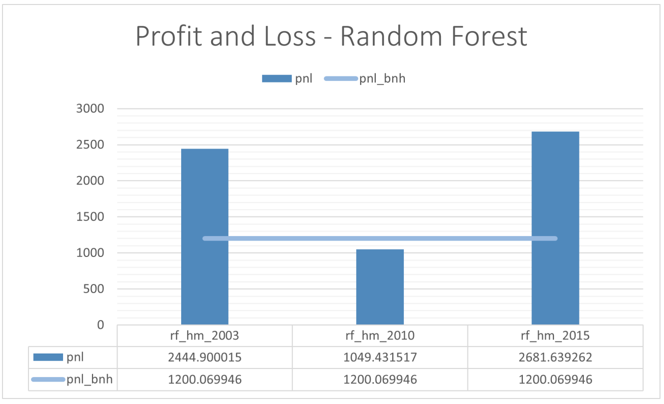
from its learning mechanism. Stock market state prediction highly corresponds to HMM in a way that they both use observable data to determine unobserved states and the sequential transitions in between. Moreover, regardless of the prevailing technical or economic approaches to define bull and bear market, its essence lies on whether stock prices are going up or down, basically whether stock returns are significantly positive or negative. This connection validates our intuitive choice of HMM and GMM. We expect the models to classify states into three categories with one of positive returns, one of negative returns, and one of returns around zero. We also expect the momentum character to be captured by the transition matrix, which would be higher possibility to stay within the same market state. Statistical metrics agreed to these expectations and proved the efficiency of HMM and GMM in our case.

Given the extraordinary performance achieved by HMM and GMM with solely returns input, we actively apply these market state predictions as "true" labels to study in supervised models, which will be discussed in the following section.

2) Supervised machine learning:

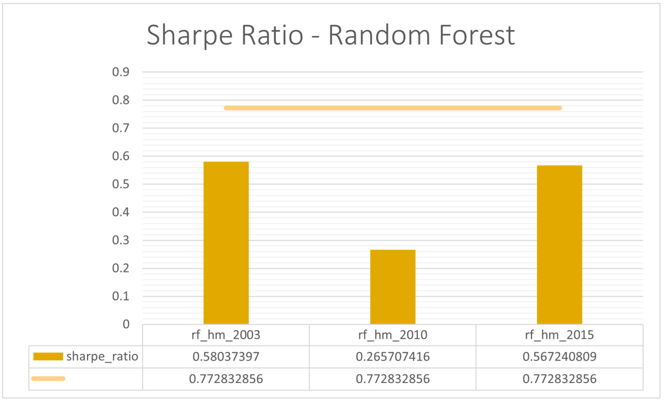
- Random forest

In this analysis, Random Forests illustrated a strong ability in finding the dominant factors of the Russell 3000 state and studying the movement pattern. All data with various state definitions (y label) could be well studied, mostly with an accuracy score of at least 90%. Changes in length of the training period (2003, 2010, 2015) make barely any difference on random forest results. Grid search, varying in the number of estimators and max depth of trees, could improve the results to a certain degree, but not much. RF generates PnL highly in correspondence with the underlying data (y label) as figure 10 shows. This proves that RF could distinguish the possible components of each market state definition and mimic its behavior to define the market state in the test period. However, RF Sharpe ratio results fall under the plain results of the original dataset, as shown in figure 11. This might be due to the extreme timing sensitivity of the original datasets and RF's defect in dealing with time-series data and producing smooth continuous results.



	rf_hm_2003	rf_hm_2010	rf_hm_2015
pnl	2444.900015	1049.431517	2681.639262
pnl_bnh	1200.069946	1200.069946	1200.069946

Fig. 10. Profit and Loss of Random Forest model.



	rf_hm_2003	rf_hm_2010	rf_hm_2015
sharpe_ratio	0.58037397	0.265707416	0.567240809
	0.772832856	0.772832856	0.772832856

Fig. 11. Sharpe Ratio of Random Forest model.

- XGBoost

XGBoost conducted the same process on different y labels, training period start time, and hyperparameters. The figure 13 shows that the shorter the training period before trading, the better PnL performance after learning. In addition, the Sharpe ratio was not affected by the length of the training period. In the below graphs, we present the PnL and Sharpe ratio comparison with each length of the training period and before learning. XGBoost outperforms in active trading than buy and hold in PnL for each training period but cannot achieve the high Sharpe ratio as the buy and hold strategy. For XGBoost, adding grid search can improve the accuracy by about 0.1.

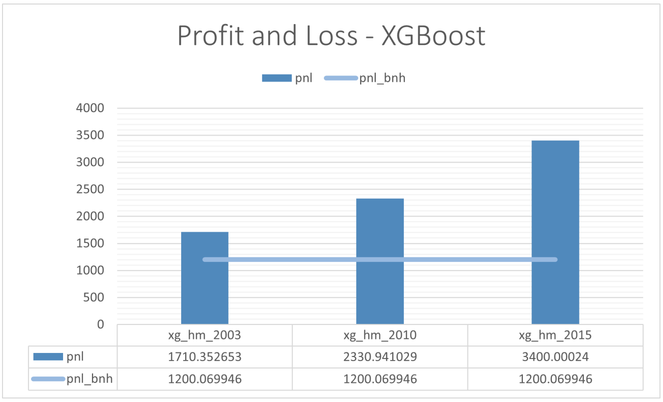


Fig. 12. Profit and Loss of XGBoost model.

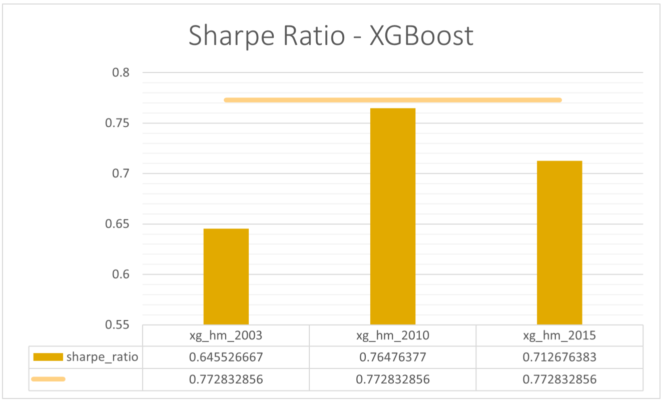


Fig. 13. Sharpe Ratio of XGBoost model.

Both Random Forest and XGBoost belong to Ensemble Leaning, as they use the majority predictions of multiple individual classifiers. This case highly suits advanced supervised learning models, considering the wide range of indicators as independent variables and their close connection with the dependent variable - market state. With a strong link from market state to stock returns to indicators, either directly derived from or closely related to, it is not surprising that both methods successfully captures the components of market state and makes accurate predictions.

To take a further step, the essential difference of the two models is that Random Forest uses Bagging while XGBoost uses Boosting. Boosting allows for examination of residuals and continuous improvement by building trees accordingly. As a result, it is natural that XGBoost realizes such high performance.

3) Deep learning: To establish and tune deep learning models, we firstly center and scale all features based on percentiles, therefore the models are not influenced by marginal outliers. Three different lengths of training periods are implemented. A rolling window training method is applied in LSTM by trying window length from 1 day to 30 days to cut time series into sub-sequences. For example, the LSTM model will predict the 11th day's state by using previous 10-day features during training. We also carry out a One-Hot Encoder for labels and implement various sets of features and labels into the LSTM and MLP models. Furthermore, hyper-parameter tuning always plays a crucial role in deep learning. [7] We explore different values of the number of layers, dropout rate, learning rate and epochs numbers, and different types of the activation function.

We plot loss functions for training and validation results to prevent over-fitting problems [8]. We evaluate our classification LSTM and MPL models by both computational methods of confusion matrix and F1-score, and financial performance of PnL and Sharpe ratio. As a result, firstly, for parameters, the time of 2015-2017 and a rolling window of three days were settled because of their higher PnL in most cases. Epoch value of 50 can balance the optimization of accuracy and not occurrences of over-fitting. Secondly, for features and labels, LSTM prefers more features than less features since it has a dropout rate to control information. Labeling methods 2 and 7 perform well. Although the data using the labels created by GMM show the best PnL of 804.25, this model suffers imbalance problems and all labels are predicted as one state bull. The tuning of giving more weights to minority class does not help much on PnL.

Finally, after deleting all these kinds of ineffective models, our best model realizes 310.63 PnL, which is larger than the best MLP model PnL of 209.66. Its loss of training set was decreasing as well as validation loss function also decreasing, which demonstrates this best model is not over-fitting. Confusion matrix shows that there is an imbalance in predicted dependent values, but it is much more related to the original framework of data itself. The best model's confusion matrix figure 14 and loss function figure 15 are as follows:

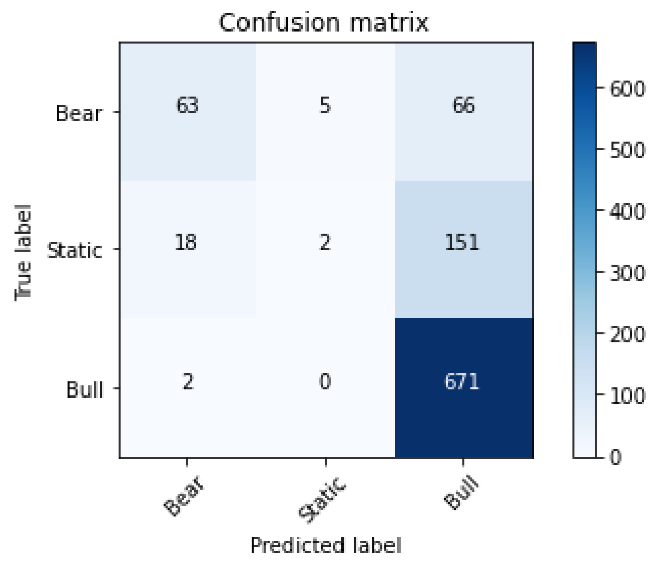


Fig. 14. Confusion Matrix of the Best LSTM Model

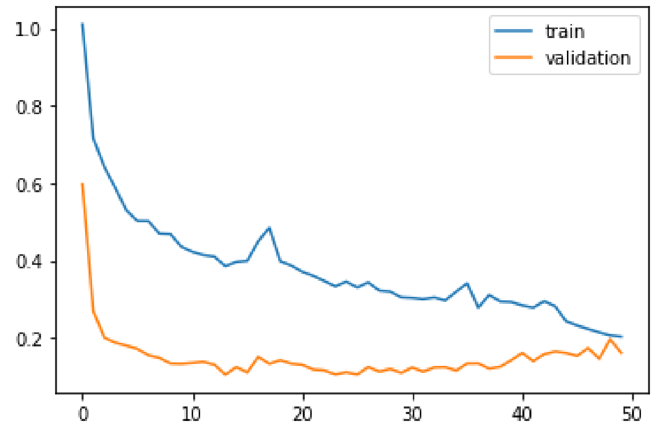


Fig. 15. Loss Function of the Best LSTM Model

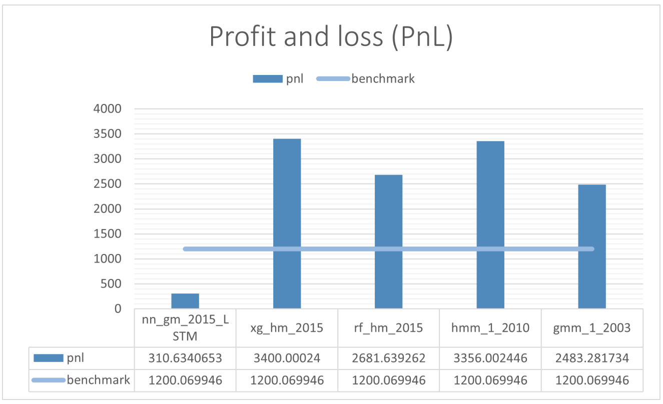
IV. OPTIMAL MODEL COMPARISON

In this section, we choose the top performance model of each type. Next, we compare our five best models in terms of model accuracy and profitability to select one optimal model as our final decision. The following figure 16, figure 17, and figure 18 describes PnL, Sharpe ratio, and maximum drawdown among the 5 best models.

For deep learning LSTM model, it performs with an 75% accuracy in the test period 2018-2021. From the charts we could see that LSTM's PnL is 310.63 and is the only model that not outperforms naive strategy. It deserves mentioning that its maximum drawdown is 1.35, which is the lowest value and much less than benchmark's maximum drawdown. This result shows that LSTM model's investment is risk-less. However, even the best LSTM model has a negative -34.22% Sharpe ratio and under-performs the buy-and-hold strategy. Suffering severely from imbalanced classification issues, LSTM networks cannot learn the minor class in label well and perform poorly in predicting hidden market state. Thus, we will not choose the non-profitable LSTM model.

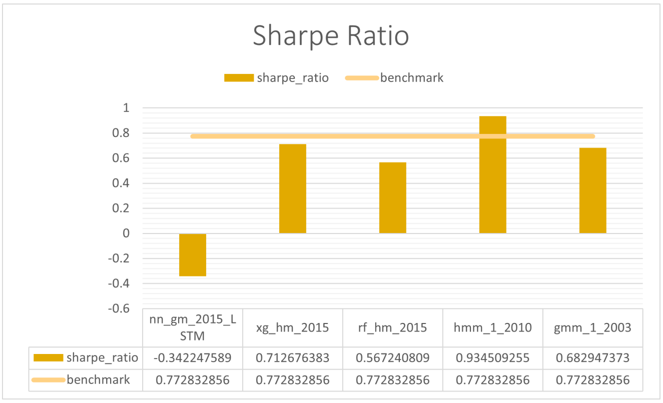
All four other top performance models achieved PnL higher than benchmark, although volatility increases along. The best PnL result comes from the XGBoost model learning with HMM defined labels, resulting in 183.32% over the benchmark. As for the Sharpe ratio, we find that HMM has the largest result of 0.93 and exceeds benchmark. However, XGBoost, random Forest and GMM exhibits maximum drawdown higher than benchmark. This is acceptable considering that risk and volatility rises and falls along. Although XGBoost produced the best profitability together with relatively higher risk, we primarily focus on profitability so we overlook the risk and brings XGBoost to the trading process.

In general, the best model among supervised learning models is XGBoost because it maintains high accuracy while also producing the highest trading result. The best model among unsupervised models is HMM considering the high PnL. Although deep learning models have not outperformed the benchmark, it provides insights into the application of deep learning models on highly imbalanced time series. By applying predicted states of best HMM model as training labels into XGBoost model, we achieve a higher PnL than solely HMM and XGBoost model and a fairly Sharpe ratio as well.



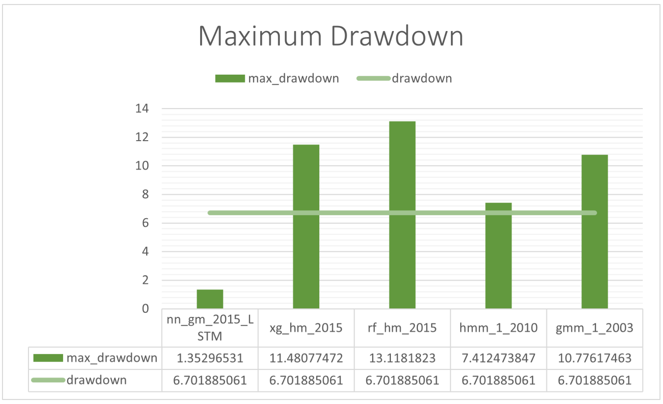
	nn_gm_2015_LSTM	xg_hm_2015	rf_hm_2015	hmm_1_2010	gmm_1_2003
pnl	310.6340653	3400.00024	2681.639262	3356.002446	2483.281734
benchmark	1200.069946	1200.069946	1200.069946	1200.069946	1200.069946

Fig. 16. PnL for 5 Best Models



	nn_gm_2015_LSTM	xg_hm_2015	rf_hm_2015	hmm_1_2010	gmm_1_2003
sharpe_ratio	-0.342247589	0.712676383	0.567240809	0.934509255	0.682947373
benchmark	0.772832856	0.772832856	0.772832856	0.772832856	0.772832856

Fig. 17. Sharpe Ratio for 5 Best Models



	nn_gm_2015_LSTM	xg_hm_2015	rf_hm_2015	hmm_1_2010	gmm_1_2003
max_drawdown	1.35296531	11.48077472	13.1181823	7.412473847	10.77617463
drawdown	6.701885061	6.701885061	6.701885061	6.701885061	6.701885061

Fig. 18. Maximum Drawdown for 5 Best Models

V. DISCUSSION

- **Data Imbalance Issue.** Among our eight definitions to classify market states, several resulted in an imbalanced distribution of market states, either with rarely any static state or with very few bear and bull. This imbalance seriously affects the performance of some models and attaching weights for different states could not save the situation. In future research, we recommend to seek a more balanced and reasonable method to classify market states.
- **Overfitting and Underfitting.** The inner stochastic nature of stocks make it very hard to catch their trends and easy to over-fitting. Thus, it is quite necessary to check if our prediction is over-fitting, but this issue is hard to adjust in real operation. What makes our multivariate time series classification more complex is that the label imbalance can cause our results actually under-fitting, but have fake high accuracy. Here, our "under-fitting" means that the results with only one category actually can't be used for our strategy.

VI. CONCLUSION

In conclusion, by conducting unsupervised, supervised, and deep learning models, we have successfully established a system of market state trading, including definition, prediction, back-testing, and evaluation. Our best definition comes from Hidden Markov Model with indifference in training period length while our best prediction derives from XGBoost where it improves PnL by a maximum of 183.32%. We have also built our back-testing and evaluation framework to evaluate our results not only from a modeling perspective but also from a trading perspective. In the end, our final model is the combination of Hidden Markov Model labels and XGBoost predicting algorithm. Beating the buy-and-hold strategy, our final model possesses superiority in predicting market states and obtaining profits from our active trading strategy.

REFERENCES

[1] G. Gallagher, "Worried About a Recession? Heed the Lessons from the Bulls and Bears", .

[2] Shiu-ShengChen, "Predicting the bear stock market: Macroeconomic variables as leading indicators", *Journal of Banking Finance*, vol. 33, no. 2, pp. 211–223, July 2009.

[3] M. R. Hassan, B. Nath, "Stock market forecasting using hidden Markov model: a new approach", *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, pp. 192–196, 2005.

[4] "Gaussian mixture models", .

[5] A. Gupta, "XGBoost versus Random Forest", , April 2021.

[6] J. Min, “Financial Market Trend Forecasting and Performance Analysis Using LSTM”, *Journal of Banking Finance*, 2020.

[7] M. AJia, J. Huang, L. Pang, Q. Zhao, “Analysis and Research on Stock Price of LSTM and Bidirectional LSTM Neural Network”, , January 2020.

[8] E. Kole, D. van Dijk, “How to Identify and Predict Bull and Bear Markets?”, , September 2010.