Market State Prediction with Machine Learning

1 Introduction

RECOGNIZING market transition has been a prolonged topic for more than a century. Scholars and investment professionals have regularly used it to predict market trends. An empirical study Münnix et al.(2012)[1] shows the industries' correlations increase dramatically during the financial crisis. Using Russell 3000 daily returns across industries from 01/02/2002 to 01/25/2022, we display similar results: industries exhibit relatively positive correlation in bear markets, whereas general negative correlation in bull markets (Figure 1 in page 2). For that reason, we believe that it is reasonable to model market data into regimes. There are two common methods to define market states among researchers. The first method is Set of Rules (see e.g. Lunde and Timmermann(2004)[2], Pegan and Sossounov(2003)[3]), which defines bull(bear) market based on price dynamics within a period. The second method utilizes a statistical model to classify states endogenously, for example the Regime-Switching Markov Models which characterize markets by different mean returns and volatilities. Dias, Vermunt, and Ramos(2015)[4] identify market into three regimes through Regime-Switching Markov Models to determine the so-called bull, bear, and a static regime. Although Set of Rules is more intuitive in economic sense, the Regime-Switching Markov Models allow a higher degree of dynamic complexity exhibited in the real financial markets. In addition, our results show that it is more difficult for supervised learning models to learn states defined by Set of Rules, the fact largely ignored by many projects focusing mainly on supervised learning. Even though the Markov Model / Set of rules can accurately recognize different market states, one cannot apply it to do trading due to the lag effect. In order to do so, another prediction model is required. Machine Learning has grown rapidly in the recent decade. It helps to identify intrinsic relationships robustly. In this study, two classical machine learning models and two deep learning models are used for the market state prediction.

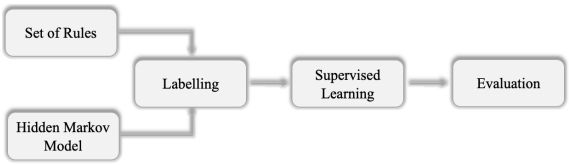
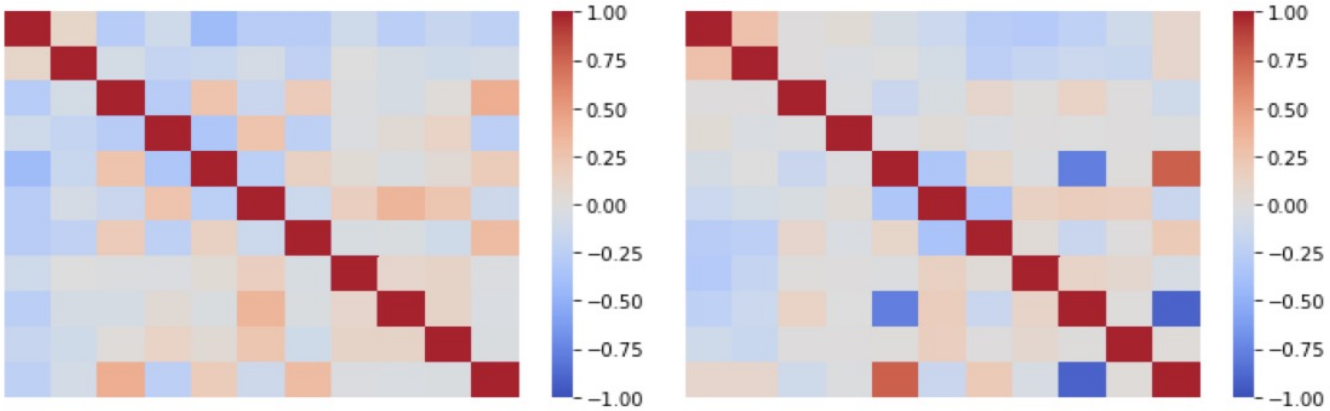


Figure 2: Flow Chart

Illustrated by Figure 2, our paper proceeds as follows. In Section 2, we discuss data and features suggested by academic or practical research. Section 3 introduces labels from Set of Rules and Regime-Switching Markov Models. In section 4, we conduct our machine learning research, including data preprocessing, classical machine learning methods, deep learning methods, backtest results based on the canonical trading strategy. Finally, we conclude our findings and discuss further improvements in section 5 and 6.



(a) Bull

(b) Bear

Figure 1: Correlation Heat Map of Bull and Bear Market

2 Data Selection

This study uses the daily close price and volume of Bloomberg's Russell 3000 Total Return Index. In contrast to a price index, the total return index better reflects the actual returns that an investor holding the index components would receive. Dates range from 01/02/2002 to 01/24/2022.

In the following unsupervised and supervised learning models, we split the entire dataset into training(01/02/2002 - 12/31/2013), validation(01/01/2014 - 12/31/2017) and testing sets(01/01/2018 - 01/24/2022).

The capital market is affected by many factors such as political upheaval, economic situation, market events, investors' behavior, market liquidity, and much more. Here, we consider a large set of 46 variables (see Table 1) that have proved to be empirically relevant or recommended from a practical point of view to predict market regimes. Features include bond yields, term spreads, macroeconomic variables, technical indicators, options market information, commodity prices, and Fama-French factors. First, the bond market reflects inflation growth and the economic condition which directly affects the stock prices. Accordingly, we include government treasury yield of all maturities[1], LIBOR interbanking rate, daily interest rates, and yield spread over selected different maturities[2]. Second, literature has widely documented the effects of macroeconomic news announcements on the financial market. Jareño and Negrut (2015)[5] found that GDP and the unemployment rate have statistically significant relationships with the stock market. Correspondingly, we create four dummy variables to show the announcement dates of CPI, Unemployment rate, total non-farm, and GDP. "1" is set for announcement dates while "0" for other dates.

Features	Frequency	Source
RU30InTR Index	Daily	Bloomberg
SPXT Index	Daily	Bloomberg
VIX Index Price	Daily	Bloomberg
Treasury Rate Changes (1M to 30Y Daily)	Daily	FRED
Yield Spread Change (10Y-3M, 10Y-2Y, 3M LIBOR)	Daily	FRED
CPI Change	Monthly	US BLS
Unemployment Rate Change	Monthly	US BLS
Total Non-Farm Change	Monthly	US BLS
GDP Change	Quarterly	US BLS
WTI Oil Futures	Daily	Bloomberg
WTI Spot	Daily	Bloomberg
Fama-French Three factor	Daily	Kenneth R.French
CCI	Daily	Bloomberg
Williams % R	Daily	Bloomberg
Parabolic Studies	Daily	Bloomberg
DMI	Daily	Bloomberg
Fear/Greed Indicator	Daily	Bloomberg
EMA5	Daily	Python Package
EMA10	Daily	Python Package
EMA5_VPT	Daily	Python Package
EMA5_FI	Daily	Python Package
EMA5_KST	Daily	Python Package
EMA5_BollingerBands	Daily	Python Package
EMA5_RSI	Daily	Python Package
EMA5_MACD	Daily	Python Package

Table 1: Description of 46 Features

In addition, we utilize thirteen technical indicators involving momentum and moving averages as features to incorporate historical information. Many fund managers and investors generally accept and use these technical indicators to predict future price trends. Furthermore, we use VIX Index to gauge the level of risk and fear in the market. Banerjee et al. (2007)[6] found a positive relationship between S&P 500 future performance and the VIX index. Given that the S&P 500 is a subset of Russell 3000, we expect VIX to be a robust predictive feature for future Russell 3000 returns.

Finally, we also use close price and trading volume of the S&P 500 total return index and additional variables (i.e. WTI oil price and Fama-French factors) that indicate changes in the level of risk.

[1] We use treasury yield from short to long maturities that include 1-month, 3-month, 6-month, 1-year, 3-year, 5-year, 10-year, and 30-year.

[2] In this study, we use 10 year - 3-month treasury yield to reflect difference between long-term and short-term rate, 10 year -2 year yield spread to represent long-term and medium-term interest difference, and TED rate to indicate difference between the interbank loans and T-bills.

3 Labeling

Our goals in this section consist of three: Firstly, the label should facilitate the supervised learning task. Secondly, the trading strategy determined by the label should generate a higher return and higher Sharpe ratio than does the buy-and-hold strategy. Lastly, the label should be more stable than the sign of Russell 3000 log return. Many researchers comment that noise in financial data makes market prediction extremely challenging. Labeling is, in fact, a way to screen out the noise in the raw data.

Following Garcia-Almanza and Tsang (2006)[7], we create the label RET, which is a future signal that calculated by looking ahead in a future horizon. Many supervised learning projects tackling the same task define their labels in a similar fashion.

$$\text{RET} = \begin{cases} -1, & \text{if 10-day simple return < -0.02} \\ 0, & \text{if } |\text{10-day simple return}| < 0.001 \\ 1, & \text{if 10-day simple return > 0.02} \\ \text{forward fill}, & \text{otherwise} \end{cases}$$

RET succeeds in the last two goals but fails the first one: RET is extremely difficult to predict. For example, all the supervised learning models we tried have categorical accuracy of less than 0.5. We thus look to a more structural way to classify the data: the Markov-Switching model.

The Markov-Switching model introduced by Hamilton (1989) allows for a time-varying conditional mean and variance. In contrast, the unconditional distribution can have skewness and fat tails, present in Russell 3000 data. This approach sorts the data endogenously into finite regimes.

The Augmented Dickey-Fuller test shows that the Russell 3000 adjusted close price contains a unit root, whereas the p-value is less than 10^{-27} for the log return of the price. On the other hand, Khaidem and Dey (2016)[8] have suggested using smoothing techniques before classification, such as moving average (MA) and exponentially weighted moving average (EWMA).

The basic Markov-Switching model is a pair (S_t, Y_t) such that unobservable S_t is a Markov process whose transition probability matrix is a constant matrix and the observable $Y_t \sim N(\mu_{S_t}, \sigma^2_{S_t})$. We compute maximum likelihood estimators, showing that the basic Markov-Switching model can recognize two regimes rather than three or more regimes. The model with more regimes fails to converge and suffers from unstable states, i.e., the transition matrix is not diagonally dominant. The parameters estimated in the basic two-regime model (HMM1) are statistically significant and consistent with other relevant findings: low-mean with high variance and high-mean with low variance regimes respectively. The three issues left to discuss in the HMM1 are measurability, mapping, and stableness.

Measurability: All the features and parameters should be adapted to the time t information in the testing set. Otherwise, the trading strategy constructed from the model cannot be put into practice. Although the labeling task is free from the measurability constraint, we can take the lag-1 label as a feature in supervised learning models if we adapt them to the testing set. Note that the main challenge in this study is supervised learning, shown by the label RET. Ideally, we want to create adapted labels with high autocorrelation. Thus, by inputting the lag-1 label as a feature, the supervised learning models can attain much better accuracy. In the Markov-Switching model, the filtered probability of state i at time t, executed through the forward algorithm, is defined as $Pr(S_t = i \mid Y_1, Y_2 \dots Y_t)$. Suppose we estimate the model parameters using only the training set and label the entire data through a measurable map from the filter probability. In that case, the label is adapted in the validation and testing sets and thus can be used as a feature in the supervised learning task. Note that the smoothing probability, executed through the forward-backward algorithm, involved future information, i.e., not measurable, so we construct our labels through the filtered probability.

Mapping: We set two thresholds h_1 and h_2, where $h_1 \geq h_2 \in [0,1]$ to map from the filtered probability of the low-mean regime to the three states. When the probability is higher than h_1, it will map to bear; the probability in between, it will map to static; otherwise, it will map to bull. We can then define a canonical trading strategy on the three states: bear $\mapsto$ short, static $\mapsto$ cash, bull $\mapsto$ long. The h_1 and h_2

are estimated such that the two maximize the trading return in the training set. Note that the Sharpe ratio maximization would map to two states rather than three to minimize the variation of the trading return, so we consider the Sharpe ratio maximization inappropriate here.

Stableness: To capture the stableness, we introduce the notion of state duration, defined as the length of a run of the consecutive states. Most labels created by the Markov-Switching models have a much lower average of bear and static duration than the bull. In some cases, the average can be as close as one. In other words, changes from static and bear states occur so often that the label fails our last criteria – stable states. To impose a stickier state structure, we can smooth the log return time-series data through a measurable function[3] or add duration constraints when maximizing the trading return.[4] There exists a trade-off between a high trading return and a sticky structure in our setting, and the constraints are chosen to reflect a reasonable trade-off. On the other hand, Lunde and Timmermann (2004)[2] show the necessity to incorporate state duration in predicting market states, so state duration constructed from the labels will also be our feature for supervised learning models. The state duration is also adapted in the validation and testing sets.

Due to the robust structure of the Markov-Switching model, the supervised learning models are more likely to predict the label with high accuracy. The latter section shows that both classical and deep learning models can achieve around 70% accuracy in the validation set.

The success of the Markov-Switching model encourages us to take a further step. The common use of the same set of features might further facilitate the supervised learning task, so we run an extension of the basic Markov-switching model (HMM2). Rather than a constant matrix, the transition probability is now parameterized using the logistic function: $Pr(S_t = i \mid S_{t-1} = i, X_{t-1}) = \frac{e^{\beta_i X_{t-1}}}{1+e^{\beta_i X_{t-1}}}$ where $i \in \{1,2\}$. All the features, HMM1 labels[5], and state duration constructed from the labels are chosen as dependent variables. Due to the multicollinearity, we regress on the first three principal components derived from the dependent variables.

With limited space, we report only the results of the labels used in the latter section. The labels are chosen based on the metrics to reflect trade-off between duration and trading performance. The tables 2 and 3 summarize the metrics of the labels.

Training set	Mean bull duration	Mean static duration	Mean bear duration	Annual return	Sharpe ratio
Buy-and -Hold	N/A	N/A	N/A	4.30	0.19
RET	18.07	4.86	14.82	97.20	2.53
HMM1 (log return)	13.57	2.12	17.42	12.21	0.66
HMM1 (EWMA10)	12.36	2.56	4.12	44.59	1.92
HMM2 (MA10)	11.69	2.01	4.07	77.91	2.63

Table 2: the metrics of the labels in the training set

We close this section with some remarks. First, our baseline model HMM1 (log return) is likely to overfit the training set, and the anomalies are colored in red. Our experiment shows the metrics of the label is highly dependent on the time frame of the training set. Smoothing preprocessing is necessary to prevent overfitting. Second, HMM1 works better with EWMA, whereas HMM2 does so with MA. This patterns appear in all the labels we have constructed. With fewer features, HMM1 has to put enough weight on the latest data point to attain high

[3]MA5, MA10, MA15, EWMA5, EWMA10 and EWMA15 applying on the log return of the Russell 3000

[4]h_1 and h_2 maximize the trading return in the training set such that the mean bear duration is larger than 4 and the mean static duration is larger than 2

[5]The labels are constructed by HMM1 with inputs: log return, MA5, MA10, MA15, EWMA5, EWMA10, and EWMA15. We include all of them to capture the information across horizons. Spectral analysis has recently been a hot topic in the finance literature, for example, Bandi, Chaudhuri, Lo, Andrew W. and Andrea (2019)

trading return and Sharpe ratio. Relying on other features, HMM2 can afford to put less weight on the latest to smooth out the noise. Third, HMM2 is generally better than HMM1 in terms of labeling partially shown by the metrics in the tables. Fourth, the validation set has much higher proportion of bull state than does the training set, making the mean bear duration drop among labels we have constructed.

Validation set	Mean bull duration	Mean static duration	Mean bear duration	Annual return	Sharpe ratio
Buy-and -Hold	N/A	N/A	N/A	11.27	0.80
RET	19.29	13.19	10.05	42.67	2.75
HMM1 (log return)	11.42	2.07	5.71	9.25	0.80
HMM1 (EWMA10)	20.65	2.78	2.96	31.18	2.52
HMM2 (MA10)	18.84	1.43	2.32	43.50	2.84

Table 3: the metrics of the labels in the validation set

4 Machine Learning

4.1 Data preprocessing

4.1.1 Features Engineering

To prevent "Garbage in, garbage out", we need to conduct meaningful and reasonable feature transformations before putting raw data into the model. The following additional features are included into the input dataset. (1) Return of price data as price is non-stationary; (2) Differencing on interest rate data; (3) EWMA smoothing before applying to technical indicators to reduce noise[8]; (4) Weekday and Month indicators to explain seasonal effects, such as January effect; (5) Using HMM information, such as state and current state duration, as features. Since the HMM only uses training dataset for training, it won't include any future information in validation/testing dataset.

4.1.2 Imbalanced Classification Problem

Imbalanced sample set across different classes would cause the model to be more sensitive to the majority class since the majority class has a larger weight in the loss function. In order to deal with this problem, class weight adjustment and oversampling are used in Classical Machine Learning and Deep Learning respectively.

4.1.3 Standard Scaler

If the orders of magnitude vary across features, one with higher orders dominates in the objective function, weakening the effectiveness of other features. Feature standardization transforms each feature independently by computing the relevant statistics on the samples in the training set, to make the values look like Gaussian distributed data. As a result, all features are centered and scaled by subtracting the means and then dividing by standard deviations.

4.2 Classical Machine Learning

The general idea of ensemble methods is to combine multiple models to produce a stronger learner. Under the circumstances, ensemble models tend to get better results than single ones. In the meanwhile, there exist problems such as computational complexity and overfitting.

There are two main types of ensemble learning, namely bagging and boosting. Bagging combines Bootstrapping and Aggregation to form one ensemble model. Specifically, independent base models are performed on different subsets of the training data in a parallel way. Boosting, however, trains several weak models sequentially by focusing on the mistakes of prior iterations. Specifically, two ensemble models are chosen to train our labels. The features for this part do not contain lagging terms.

4.2.1 Model - Random Forest Classifier

Random forest classifier can be considered as a bagging of decision trees. The only difference is that random forest models make split decisions based on different randomly selected features. By doing so, we achieve a decrease in correlation which leads to a lower variance. In other words, this model gen-

erates relatively robust results. The main problem is that random forest is a black box type of model and does not allow much participation in estimation aside from randomization.

4.2.2 Model - Adaboost Classifier

Adaboost classifier is the first successful boosting algorithm developed for classification. More weight is put on previously misclassified residuals, while less on those already handled well. Intuitively, boosting can result in better performance than bagging. Whereas they tend to be harder to tune and be more likely to overfit.

4.3 Deep Learning

Deep neural networks have achieved accuracies far beyond that of classical ML in many areas including NLP, speech and image recognition. Therefore, Deep Learning is promising in solving our problem.

4.3.1 Input Transformation and Oversampling

The structure of the deep learning algorithm in TensorFlow is different from that in sklearn. TensorFlow requires further preprocessed data before fitting into the model.

In order to use a time series model in Deep Learning, the $2D$ input matrix has to be transformed into $3D$ with shape = (nSample, nTimesteps, nFeatures)[6]. The 3-class label vector has to be transformed into a 3-column matrix through One Hot Encoding.

Data augmentation for time series data is a challenging task. It is well established in image data, like cropping, mirroring, and color augmentation, but it is not established as a standard procedure in time series data. The challenge of using time series is that time series follows structural patterns that are dependent on element order. There are a few categories of data augmentation techniques, e.g. Random Transformation, Pattern Mixing, Generative Models and Decomposition[9]. However, not all of the data augmentation techniques are applicable or beneficial to our model. Some advanced techniques like Synthetic Minority Oversampling Technique (SMOTE) cannot be directly applied to time series data. Here we use Jittering (adding noise) under Random Transformation. Independent normal random variables with a small volatility are added into each feature to create some noise.

4.3.2 Model - GRU

The Gated Recurrent Unit (GRU) neutral network is one of the recurrent neural networks, which is similar to LSTM. Although both were introduced to solve the vanishing gradient problem[10], GRU is less complex so it is more suitable for a smaller dataset. GRU uses two gates: update and reset gate, while LSTM uses three gates: input, output and forget gates. Since our dataset is not large enough compared to the number of parameters trained in the model and considering the computational efficiency, GRU is a better choice compared to LSTM.

Architecture: A GRU block, which consists of 3 layers of GRU, is connected to the input layer. The GRU block returns an output at the last timestep instead of that from the whole sequence. It is then connected to a dense block, which consists of 5 dense layers with ReLU activation. Lastly, it is connected to a head layer consisting of 3 nodes using Softmax activation predicting the probability of the 3 hidden states.

4.3.3 Model - ESN

Reservoir computing (RC) models have taken a significantly different approach to model recurrent neural networks[11]. RNN models are trained by backpropagation, while in RC models, the input signal is connected to a fixed (non-trainable) and random dynamical system (the reservoir), a higher dimension representation (embedding) is then created and connected to the desired output via trainable units.

The Echo State Networks (ESN)[12] is the most common one among all RC models. The non-trainable weights between the reservoirs are determined by linear regression and do not rely on back-

[6]nSample: number of samples; nTimesteps: number of days in the lookback period, here we use 180 days; nFeatures: number of features used

propagation. Because of its characteristics, ESN is a highly efficient neural network. The recurrent structure of an ESN within the reservoir makes ESNs well suited for time series modeling.

Architecture: The ESN model used has a similar structure to the GRU model. The ESN block consists of 1 layer of ESN. Although some research[12] suggest a deep echo state network performs better than a single layer ESN model, the DeepESN mode doesn't perform well in our case.

4.3.4 Avoiding overfitting

Training a neural network with a small dataset can cause the network to memorize all training examples, in turn leading to overfitting and poor performance on a holdout dataset. Thus, some techniques are used to avoid overfitting.

Early Stopping: The Early Stopping callback monitors the validation loss and stop further training when it reaches the optimal stage. It helps to tackle overfitting.

Dropout Layer: Deep learning neural networks are likely to quickly overfit a training dataset with few examples. Dropout layers are inserted between blocks to randomly drop some number of layer outputs. This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer.

4.4 Result

4.4.1 ML Result interpretation

For each combination of labels and models, we utilize cross-validated grid-search to tune specified hyperparameters for our estimators. Once our model is set with the best optimized parameters, we interpret the result from the aspects of ROC score and confusion matrix.

Take HMM1(EWMA10) and random forest classifier for example, the best parameter $\{n\_estimators = 420, max\_depth = 3\}$ remains robust and insensitive to change. Fitting the model with the whole training set, the average CV score (balanced accuracy) reaches 77.16%. As a result, balanced accuracy score reaches 82.60% during the outsample period.

As is shown in the ROC curve, the classifier behaves well at all classification thresholds, with all AUC scores greater than 0.85. It is also obvious that bull and bear states are easier to recognize, while static state 0 tends to oscillate between them, aligning with the HMM labeling characteristics.

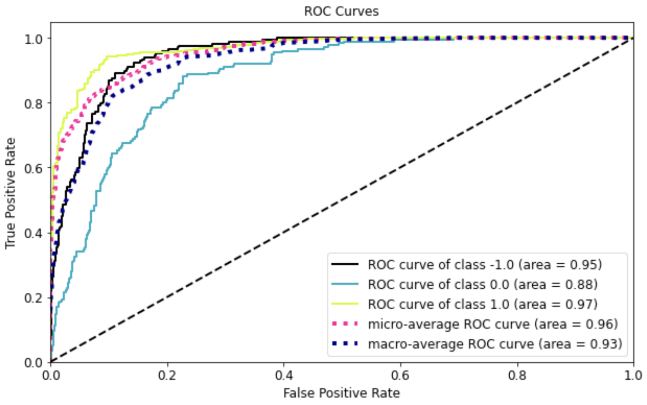


Figure 3: ROC Curves

The confusion matrix shows the same pattern. Luckily, the worst cases where bear is classified as bull or otherwise rarely appear. The main problem is that our model is not good at telling uncertain state 0. Intuitively, static state is classified at discretion by setting thresholds, so there is no inherit structure for our models to make precise classifications.

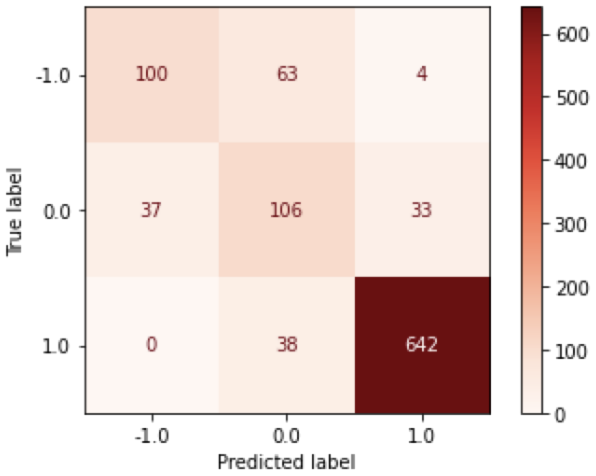


Figure 4: Outsample Confusion Matrix

Feature importances are calculated as the mean of accumulation of the impurity decreases within each tree. Top 20 important features mainly consist of HMM information, technical indicators and market sentiment indexes.

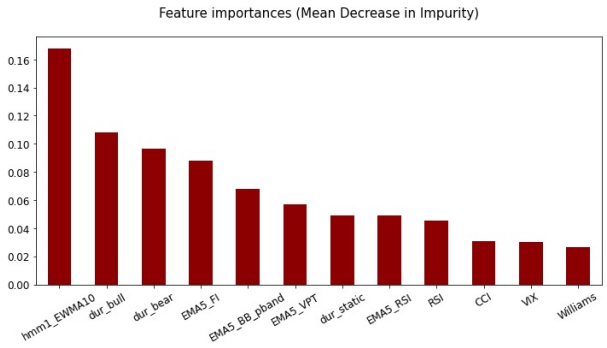


Figure 5: Feature Importances

4.4.2 Threshold Selection

From a trading perspective, entering into the market is risky. Hence, predicting a static state incorrectly is more acceptable than predicting an inaccurate bull/bear state. In other words, one only enters the market at a high-confidence level. Thus, it is classified as a bull/bear state only when the corresponding probability is larger than a certain threshold. The threshold is decided by considering the trading result and robustness[7] from the training and validation datasets.

The threshold tuning results are listed in Table 4.

First of all, label RET created by Set of Rules at discretion produces least accuracy scores among all labels. By contrast, labels created by HMM models are more structured and have a much better performance. Secondly, there exhibits significant differences between the results of ensemble methods and neural networks. Ensemble methods tend to fit our data very well, with balanced accuracy scores up to more than 60%. Thirdly, there is a decoupling phenomenon between accuracy and returns - higher accuracy scores do not guarantee higher returns. Predictions based on label RET lead to relatively high Sharpe Ratios, but such results lack stability and adaptability. By checking the confusion matrix, almost all states are predicted to be bull, acting like an untrained model. Fourthly, exposure time describes the percentage of time that our portfolio has long/short positions during the whole testing period. Trades with little exposure time are considered to be inactive.

Taking robustness, prediction accuracy and trading performance in both training and validation datasets into consideration, GRU model with label HMM2 (MA10) shows the best performance. It also gives a similar result in the testing dataset. Our final result has a better Sharpe Ratio (0.8) and outperform the market (0.5) in terms of return and risk profile.

			Validation	Testing Dataset			
Label	**Model**	**Threshold**	**Accuracy**	**Accuracy**	**Return**	**Sharpe Ratio**	**Exposure Time**
Buy-and-Hold	N/A	N/A	N/A	N/A	69.62%	0.565	100%
RET	Random Forest Classifier	0.40	47.62%	39.41%	-5.03%	-	59%
	AdaBoost Classifier	0.33	39.00%	45.40%	39.27%	0.367	97%
	GRU	0.50	40.00%	31.92%	53.06%	0.597	93%
	ESN	0.33	44.77%	31.24%	44.06%	0.434	87%
HMM1 (log return)	Random Forest Classifier	0.50	75.64%	73.96%	-16.94%	-	74%
	AdaBoost Classifier	0.33	61.23%	63.23%	-8.26%	-	79%
	GRU	0.50	45.27%	42.76%	0.61%	0.010	73%
	ESN	0.50	51.07%	50.60%	11.64%	0.132	87%
HMM1 (EWMA10)	Random Forest Classifier	0.90	64.54%	63.49%	21.07%	0.849	42%
	AdaBoost Classifier	0.40	75.40%	74.12%	41.86%	0.498	83%
	GRU	0.60	70.13%	54.22%	-6.53%	-	44%
	ESN	0.80	59.06%	56.69%	2.12%	0.047	61%
HMM2 (MA10)	Random Forest Classifier	0.60	63.70%	63.58%	17.35%	0.244	79%
	AdaBoost Classifier	0.33	68.51%	62.18%	0.43%	0.006	97%
	GRU	0.80	62.64%	62.56%	42.84%	0.804	73%
	ESN	0.33	74.20%	65.22%	30.61%	0.362	89.05%

Table 4: Supervised Learning with trading results

[7]There exists a large enough neighborhood of the chosen parameter that all elements in the neighborhood produce similar trading results

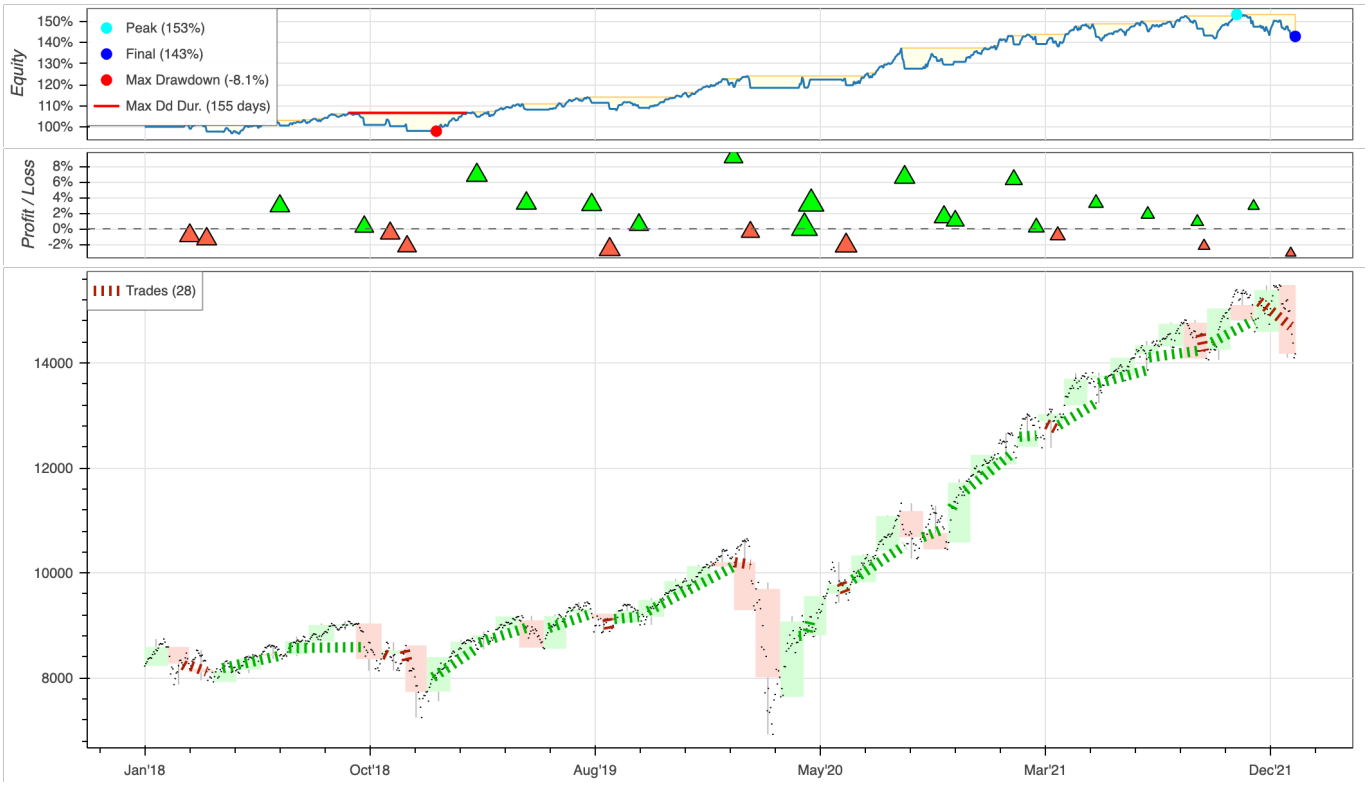


Figure 6: Backtesting Result

5 Conclusion

Based on the performance in training and validation datasets, GRU model with label HMM2(MA10) gives a return of 43% and Sharpe Ratio of 0.8 from 2018 to 2021, beating the buy-and-hold strategy(0.56). The outperformance validates our approach in determining and predicting the "hidden" state of the market (bear, bull, or static).

5.1. Noisy label: RET

Label RET itself shows a strong trading performance since this label is created directly from the future returns. RET, however, is not an ideal label. Shown by the low validation accuracy from table 4, all the supervised learning models in this project fail to effectively learn RET, possibly because of unclear structure or extreme randomness nature in the label. In contrast to RET, labels generated from a model with structure, like HMM, facilitate the supervised learning task. One can refer to the relatively high validation accuracy from table 4.

5.2 Better Labeling - HMM2 (MA10)

From the table 2 and 3, the HMM2 label is better than the HMM1 label in terms of trading and state stableness. Relying on other features, HMM2 classifies market states with higher granularity. In the Machine Learning section, supervised learning models also confirm that the HMM2 label produces both higher accuracy and better trading performance, since the transition matrix of HMM2 has incorporated information from other features. Also, the labels generated using a smoothed price data with less noise give a better trading performance. Even though RET itself performs much better in terms of trading, RET has too much noise that no model can predict it accurately and robustly.

5.3 Decoupling: Accuracy vs. Trading result

Having a decent prediction result from the supervised learning model does not mean that it also performs well in the trading. One incorrect prediction has slight influence on our accuracy score, while the penalty of making a wrong trading decision can worsen our P&L to a large extent, resulting in the decoupling between accuracy and trading result. To mitigate the decoupling effect, one can use a label that represents the market return better. Shown by Table 4, HMM2 suffers less from the decoupling issue. The accuracy and trading performance are

more aligned with supervised models using HMM2 label than using RET and HMM1. Moreover, the loss function in the ML models is not directly related to trading. Accordingly, the model does not learn a trading-oriented result.

6. Further Improvements

Here we have three further improvement for this study. Firstly, the loss function can be improved in the supervised learning model. It is easy to observe that the severity of predicting a bull market as a bear market is much more serious than as a static market. The misclassification issue appears in some ML models. Thus, more penalty can be added into the loss function for exactly opposite predictions.

The second one concerning the decoupling of accuracy and trading strategy. In order to align the prediction model with the trading result, assigning different weights on the samples depending on the corresponding return and risk in the upcoming period could help the model to focus more on those critical days in terms of trading.

Last but not least, more sophisticated trading strategies can be built to enhance the trading performance. Nonetheless, to show the validity of our market state prediction, we pick a straightforward one to avoid blurring out main theme. Furthermore, many systematic traders now seek the premia in combining and overlapping multiple signals. Accordingly, our market state prediction can be combined with other signals to enhance the trading performance.

References

[1] Münnix, M. C., Shimada, T., Schäfer, R., Leyvraz, F., Seligman, T. H., Guhr, T., & Stanley, H. E. (2012). Identifying states of a financial market. Scientific reports, 2(1), 1-6.

[2] Lunde, A., & Timmermann, A. (2004). Duration dependence in stock prices: An analysis of bull and bear markets. *Journal of Business & Economic Statistics*, 22(3), 253-273.

[3] Pagan, A. R. and Sossounov, K. A. (2003). A simple framework for analysing bull and bear markets. Journal of Applied Econometrics, 18(1), 23–46.

[4] Dias, J. G., Vermunt, J. K., and Ramos, S. (2015). "Clustering Financial Time Series: New Insights From an Extended Hidden Markov Model", European Journal of Operational Research, 243(3), 852–864.

[5] Jareño, F., & Negrut, L. (2015). US stock market and macroeconomic factors. *Journal of Applied Business Research (JABR)*, 32(1), 325-340.

[6] Banerjee PS, Doran JS, Peterson DR. Implied volatility and future portfolio returns. *Journal of Banking Finance*, 32(10), 3183-3199.

[7] Garcia-Almanza, A. L., & Tsang, E. P. (2006, July). Forecasting stock prices using genetic programming and chance discovery. In 12th International Conference On Computing In Economics And Finance (No. 489).

[8] Khaidem, L., Saha, S., & Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. arXiv preprint arXiv:1605.00003.

[9] Iwana, B. K., & Uchida, S. (2021). An empirical survey of data augmentation for time series classification with neural networks. Plos one, 16(7), e0254841.

[10] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[11] Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. Computer Science Review, 3(3), 127-149.

[12] Kim, T., & King, B. R. (2020). Time series prediction using deep echo state networks. Neural Computing and Applications, 32(23), 17769-17787.