COURANT INSTITUTE, NEW YORK UNIVERSITY

# Tactical Asset Allocation with Ensemble Learning Using Walk-Forward Optimization[*]

Renjie Pan, Liang Zou, Tianyu Zhang

December 20, 2019

[*]The capstone project of BNP Paribas Asset Management and Mathematics in Finance program in the Courant Institute, New York University. Special thanks to our advisors, Dan Singleman, BNP Paribas Asset Management, and Petter Kolm, Courant Institute, New York University.

# Contents

## 1 Introduction

Tactical asset allocation is a dynamic investment strategy that actively adjusts a portfolio's asset allocation to improve the risk-adjusted returns of passive management investing[2]. One of the most traditional ways to implement tactical asset allocation is to apply mean variance optimization to expected return and covariance matrix and get the optimal weight for each asset. One can also use equal weights and risk parity to construct optimal portfolios.

Among these different methods, one of the common steps is to estimate the expected return and covariance matrix. Many previous works have been done to predict expected returns based on different factors, and in recent years, machine learning is widely used to make predictions. However, as mentioned by Kolm, Tutuncu, and Fabozzi, [9], mean variance optimization is sensitive to estimation errors of expected returns and covariance matrix, which leads to the question to find a robust estimation of both the expected returns and covariance matrix.

The Black-Litterman model incorporates investor views to get a more robust estimation. Intuitively, the Black-Litterman model uses the market equilibrium return as the prior mean return and uses historical covariance matrix as the prior covariance, with the investors' views considered as added information, the Black-Litterman model can calculate the posterior mean and covariance, which will then be used in portfolio optimization.

Ensemble learning algorithms combine multiple weak learners to obtain better prediction performance, which are used in both classification and regression tasks. The overall performance of an ensemble learner is much better than a single weak learner. When we want to estimate the expected returns of different assets, we can consider it as a regression problem and predict the future returns directly. Alternatively, one can apply classification algorithms to predict the future directions of asset prices. Despite the less information provided by the direction, classification algorithms are commonly used in the financial markets since it is still important to express a bullish view or a bearish view when we don't have adequate information.

In this project, we want to explore the viability of combining ensemble learning with Black Litterman to construct optimal portfolios. We will use monthly price data of different asset classes as dependent variables and use monthly factor data as independent variables. We formulate a binary classification problem by predicting the future directions of each asset. We create a pool of various ensemble learning classifiers including random forest, XGBoost, etc. We will apply walk-forward optimization instead of cross-validation to tuning the hyper-parameters, which retains the information in the time series data. With the directional predictions given by our models, we will construct active portfolios and compare their performance with passive portfolios. We will incorporate qualitative views into the Black-Litterman model to get a more robust estimation of expected returns and covariance matrix, and we will focus on the comparison between Black Litterman portfolio and mean variance optimization portfolio.

# 2 Data

## 2.1 Data Source

The data is provided by BNP Paribas Asset Management.

## 2.2 Data Description

We use monthly index data of 15 assets from December 1987 to August 2019: MSCI EM, SNP 500, Eurostoxx 600, MSCI Japan, MSCI World, Russell 2000, Crude Oil, CRB Metals, Gold Price, SNP GSCI, UST 7-10yr, German 7-10yr, US IG Corps, US HY Corp and US Mortgages.

We use monthly data of different factors from February 1982 to August 2019, including 35 macro factors, 22 sentiment factors and 11 market factors, they are:

Macro Data: CESIGL Index, JPEASIEM Index, CECICUSD Index, CECICBRI Index, CECICLTA Index, CECIDEUR Index, CECIDEM Index, CGUSECOG Index, LALBLBIM Index, CNIFSVCT Index, GSDECAI Index, GSGBCAI Index, GSEACAII Index, CIICDUSD Index, CIICDG10 Index, CIICDGBP Index, CSIIUSD Index, CSIIG10 Index, CSIIGBP Index, UIGDPRIC Index, CRBIX Index, CIISCIEP Index, EPUCGLOB Index, EPUCJNTP Index, EPUCJNMP Index, EPUCNUSD Index, EPUCCUSM Index, EPUCMONE Index, FEPUUSE Index, EPUCSP index, CONSGOVR Index, CLEVCPIA Index, CGNOXAY% Index, JPM Global Manufacturing Export orders, EPUCSP Index

Sentiment Data: EUESDE Index, EUESIT Index, CONCCONF Index, COMFCOMF Index, CONSEXFB Index, CONSUEXU Index, FRCCCARS Index, GRCCI Index, ITPSCURR Index, ITPSNAT Index, ITPSECOF Index, ITPSPERS Index, ITPSFFIE Index, ITPSSAVC Index, ITPSDUR Index, CHCSEXPC Index, CHCSCONF Index, JCOMHCF Index, UKCCI Index UKCCFINA Index, EUCBLIYY Index, TWCILIY Index

Market Data: Number of States with Positive Leading Index, US High Yield Percentage of CCC Issuers Accessing Primary Market (3-month), US High Yield Percentage of CCC Issuers 3-month Default Rate, US High Yield Percentage of CCC Issuers 12-month Default Rate, SLDETIGT Index, US High Yield 12-month Default Rate, Global DM High Yield Distressed Ratio, US High Yield BBs, US High Yield Bs, US High Yield CCCs, US High Yield CCCs / US High Yield BBs

## 2.3 Data Transformation

### 2.3.1 Dependent Variables

We transform the monthly index data of 15 assets to one-month and three-month returns, which will be used as dependent variables in our ensemble learning algorithms. We consider the prediction problem as a binary classification, i.e., if one-month or three-month return is positive, the label will be set to 1, otherwise it will be set to 0.

### 2.3.2 Independent Variables

The monthly data of 68 factors are transformed to be used as independent variables. We apply two groups of transformations based on the format of the original data:

| Transformation 1 | Transformation 2 |
|---|---|
| 1-month absolute delta | 1-month absolute delta |
| 2-month absolute delta | 2-month absolute delta |
| 3-month absolute delta | 3-month absolute delta |
| 12-month absolute delta | 12-month absolute delta |
| 1-month % change | 2-month average |
| 2-month % change | 3-month average |
| 3-month % change | 6-month average |
| 12-month % change | 6-month average - 3-month average |
| 3-year Z-score | |
| 5-year Z-score | |
| 2-month average | |
| 3-month average | |
| 6-month average | |
| 12-month average | |
| 6-month average - 3-month average | |

The transformation rules are as follows:

1. If the factor of time series contains negative numbers, Transformation 2 is used.

2. If the factor is already a percentage, Transformation 2 is used.

3. In all other cases, Transformation 1 is used.

After transformation, we expand the original 68 factors to 871 factors.

# 3 Methodology

## 3.1 Feature Selection

In the field of Machine Learning, the quality of data is of vital importance. Usually, for a good machine learning algorithm, tuning hyper-parameter won't affect much on the performance. That's why we need feature engineering. Basically speaking, feature engineering, also named feature transformation and feature crosses, is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Implementing feature engineering is challenging and difficult. People have to face both the pros and cons of it. The disadvantage lies in the fact that the newly constructed features may not have a positive effect on the model or even have a negative effect on the model, thus worsening the performance of the model. Unfortunately, because of the regulation, we cannot know the exact definition of our features which leads to the restriction of the way to perform feature engineering by economic or financial intuitions. Thus, we tried the vanilla transformation as the potential way of feature transformation and then filter the features. Thus, where we can improve from the vanilla feature transformation lies in the feature selection. We need to rule out the features that behave bad to reduce the curse of dimensionality and avoid over-fitting.

Since our data record is roughly 400 and we want to control the variance of the estimators of the models, heuristically, we should choose roughly ten features from the feature pool we have mentioned in Chapter 2.

The methods of feature engineering include Filter methods, Wrapper methods and Embedded methods.

### 3.1.1 Filter Methods

The filter methods basically measure how much each feature devotes to the target variable based on a metric of the information dependency. Among them I would like to list correlation, absolute co-moment (developed by ourselves) and mutual information.

#### Correlation

As we all known, the correlation of a pair of variables measures the linear dependency of them. It is always a baseline of feature selection. However, it has a fatal problem. It cannot detect any nonlinear dependency. The information given by a correlation coefficient is not enough to define the dependence structure between random variables. The correlation coefficient completely defines the dependence structure only in very particular cases, for example when the distribution is a multivariate normal distribution. In the case of elliptical distributions it characterizes the (hyper-)ellipses of equal density; however, it does not completely characterize the dependence structure (for example, a multivariate t-distribution's degrees of freedom determine the level of tail dependence). Assume, we got a Standard Gaussian Distribution $X$ subjects to $N(0,1)$, $Corr(X, 0.9X) = 0.9$ but $Corr(X, X^2) = 0$. Thus, based on the high order co-moment, we developed nonlinear dependency metric.

### Absolute Co-moment

We developed the Absolute Co-moment to extend the correlation as the nonlinear dependency metric of 2 random variables. Take the coskewness as an example,the definition of the coskewness is:

$$Coskewness(X,Y,Z) = \frac{E[(X-E[X])(Y-E[Y])(Z-E[Z])]}{\sigma_X\sigma_Y\sigma_Z} \tag{3.1}$$

Let X = Y, we got

$$Coskewness(X,Z) = \frac{E[(X-E[X])^2(Z-E[Z])]}{\sigma_X^2\sigma_Z} \tag{3.2}$$

Now, let's see the counter example of correlation that I mentioned above. $Coskewness(X,X^2) = Corr(X^2,X^2) = 1$ which is make sense. We can extend the definition to other order of co-moment.

$$Comoment_n(X,Z) = \frac{E[(X-E[X])^{n-1}(Z-E[Z])]}{\sigma_X^{n-1}\sigma_Z} \tag{3.3}$$

Also, we want a bounded metrics for us to filter the metrics, so that we can know how much value we can reach when the non-linear dependency is really high. A naive way to do that is using positive random variables.

$$absComoment_n(X,Z) = \frac{E[(|X|-E[|X|])^{n-1}(|Z|-E[|Z|])]}{\sigma_{|X|}^{n-1}\sigma_{|Z|}} \tag{3.4}$$

Proof of bounded:[12][1]
Theorem: Assume r $\in (0,+\infty]$ and $p_1$,...,$p_n \in (0,+\infty]$ s.t.

$$\sum_{k=1}^{n}\frac{1}{p_k} = \frac{1}{r}$$

Then, for all measurable functions $f_1,\ldots,f_n$, we have

$$\left\|\prod_{k=1}^{n} f_k\right\|_r \leq \prod_{k=1}^{n}\|f_k\|_{p_k} \tag{3.5}$$

Note that

$$f_k \in L^{p_k(\mu)}\forall k \in \{1,\ldots,n\} \Longrightarrow \prod_{k=1}^{n} f_k \in L^r(\mu) \tag{3.6}$$

This theorem is an extension of Holder's inequality and can be proved by mathematical induction. It's trivial to see the $n=1$ case. Let's consider proving $n=k$ case by assuming

$n = k-1$ case is right. Since we can always sort the $p_1, ...p_n$ and redefine the subscript, assume that $p_1 \le ... \le p_n$.

Case 1 ($p_n = \infty$):

$$\sum_{k=1}^{n-1} \frac{1}{p_k} = \frac{1}{r} \tag{3.7}$$

Thus, by the induction hypothesis, we have

$$\begin{aligned} \|f_1 \cdots f_n\|_r &\le \|f_1 \cdots f_{n-1}\|_r \|f_n\|_\infty \\ &\le \|f_1\|_{p_1} \cdots \|f_{n-1}\|_{p_{n-1}} \|f_n\|_\infty \end{aligned} \tag{3.8}$$

Case 2 ($p_n < \infty$):

$$p := \frac{p_n}{p_n - r}, \quad q := \frac{p_n}{r} \tag{3.9}$$

$r < \infty$ is also required. Applied Holder inequality, we got

$$\| f_1 \cdots f_{n-1}|^r |f_n|^r \|_1 \le \| |f_1 \cdots f_{n-1}|^r \|_p \| |f_n|^r \|_q \tag{3.10}$$

Thus, we have

$$\|f_1 \cdots f_n\|_r \le \|f_1 \cdots f_{n-1}\|_{pr} \|f_n\|_{qr} = \|f_1 \cdots f_{n-1}\|_{pr} \|f_n\|_{p_n} \tag{3.11}$$

According to case 1 and case 2. The proof is done by the mathematical induction.

### Mutual Information

The mutual information of two random variables aims to measure the mutual dependence (including the nonlinear dependency) of two variables. To be specific, it quantifies the "amount of information" obtained by one random variable through observing the other random variable. The definition of mutual information is similar to the entropy of a random variable, which is a fundamental notion that quantifies the expected "amount of information" held in a signal random variable.[5]

Just like correlation, mutual information provides a more general way to measure dependency. It determines how different the joint distribution of the pair $(X, Y)$ is to the product of the marginal distributions of $X$ and $Y$. We denote $\mathrm{I}(X; Y)$ as mutual information between $X$

and $Y$, $\mathrm{H}(Y|X)$ is the entropy of $Y$ conditional on $X$.

$$\begin{aligned}
\mathrm{I}(X;Y) &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{(X,Y)}(x,y)\log\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)} \\
&= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{(X,Y)}(x,y)\log\frac{p_{(X,Y)}(x,y)}{p_X(x)} - \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{(X,Y)}(x,y)\log p_Y(y) \\
&= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_X(x)p_{Y|X=x}(y)\log p_{Y|X=x}(y) - \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p_{(X,Y)}(x,y)\log p_Y(y) \\
&= \sum_{x\in\mathcal{X}} p_X(x)\left(\sum_{y\in\mathcal{Y}} p_{Y|X=x}(y)\log p_{Y|X=x}(y)\right) - \sum_{y\in\mathcal{Y}} p_{(X,Y)}(x,y)\Bigg)\log p_Y(y) \\
&= -\operatorname{Im}(X)\mathrm{H}(Y|X=x) - \sum_{y\in\mathcal{Y}} p_Y(y)\log p_Y(y) \\
&= -\mathrm{H}(Y|X)+\mathrm{H}(Y) \\
&= \mathrm{H}(Y)-\mathrm{H}(Y|X)
\end{aligned} \tag{3.12}$$

### 3.1.2 Wrapper Methods

In wrapper methods, the feature selection process is based on a specific machine learning algorithm that we are trying to fit on a given dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion.

#### Recursive Feature Elimination [11]

The recursive feature elimination (RFE, Guyon et al. (2002)) is basically a backward selection of the predictors. This technique begins by building a model on the entire set of predictors and computing an importance score for each predictor. The least important predictor(s) are then removed, the model is re-built, and importance scores are computed again. In practice, the analyst specifies the number of predictor subsets to evaluate as well as each subset's size. Therefore, the subset size is a tuning parameter for RFE. The subset size that optimizes the performance criteria is used to select the predictors based on the importance rankings. The optimal subset is then used to train the final model.

### 3.1.3 Embedded methods

Embedded methods perform feature selection as part of the model construction. For example, one can build LASSO or Decision Tree to filter the features first and then use the filtered features to train the another model as the target model.

#### Regularization

First and foremost, the regularization here refers to L1 regularization (LASSO) and L0 regularization, not includes L2 regularization (Ridge).

The optimization problem for LASSO is

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \left|\beta\right|_1 \leq s \tag{3.13}$$

The optimization problem for L0 regularization is

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \left|\beta\right|_0 \leq s \tag{3.14}$$

where $\left|\beta\right|_0$ denotes the length of the vector $\beta$

These two regularization techniques can be used in feature selection because the optimized regions are not strictly convex. Thus the $\beta_i$ is easy to be punished to zero. However, the optimize region of Ridge Regression is strictly convex which means the $\beta_i$ will never be punished to zero.

### Decision Tree: Information Gain and Gini index

Let's imagine that we want to build a decision tree algorithm for these features. To find the best feature which serves as a root node in terms of information gain, we first use each descriptive feature and split the dataset along the values of these descriptive features and then calculate the entropy of the dataset. This gives us the remaining entropy once we have split the dataset along the feature values. Then, we subtract this value from the originally calculated entropy of the dataset to see how much this feature splitting reduces the original entropy which gives the information gain of a feature and is calculated as:

$$\begin{aligned} InfoGain_{x_i} &= Entropy_X - Entropy_{x_i} \\ Entropy_x &= -\sum_k (Pr(x=k) log_2(Pr(x=k))) \end{aligned} \tag{3.15}$$

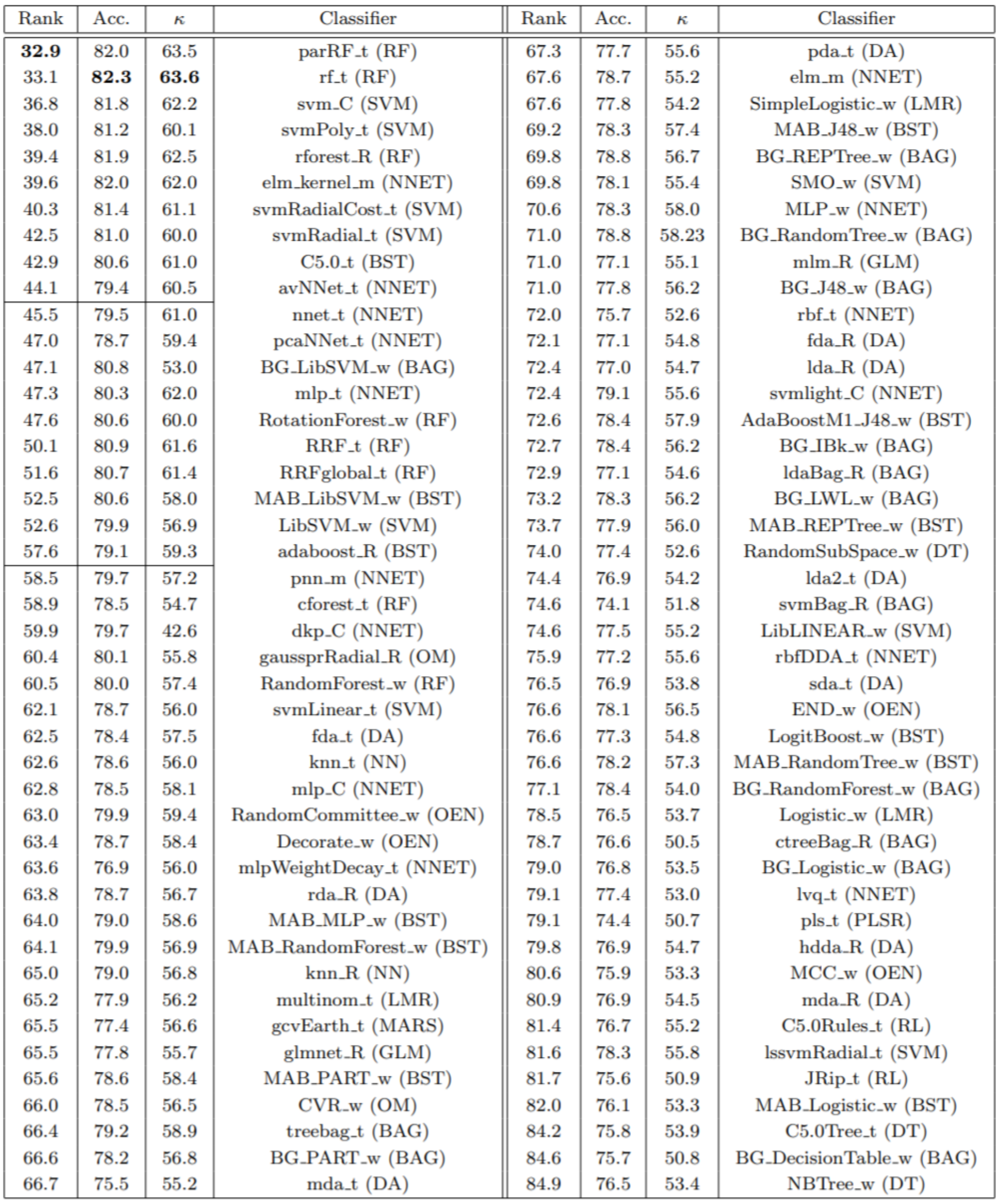
Gini index is a similarity metric to Information Gain. It is used by CART version of decision tree algorithm and is calculated by subtracting the sum of squared probabilities of each class from one. It favors larger partitions and easy to implement whereas information gain favors smaller partitions with distinct values.

## 3.1.4 Our Choice

In conclusion, we did experiments of feature selection by using LASSO, absolute co-moment and recursive feature elimination. We choose different $\alpha$ in LASSO, reasonable linear combination of co-moments in absolute co-moment and different path of elimination in recursive feature elimination method. The result of absolute co-moment is the robustest. Thus, we choose the absolute co-moment as our feature selection method.

## 3.2 Model Training and Validation

In the paper of named *Do we need hundreds of classifiers to solve real world classification problems.*[4]. The authors researched on hundreds of machine learning algorithms on various dataset.



| Rank | Acc. | $\kappa$ | Classifier | Rank | Acc. | $\kappa$ | Classifier |
|---|---|---|---|---|---|---|---|
| **32.9** | 82.0 | 63.5 | parRF_t (RF) | 67.3 | 77.7 | 55.6 | pda_t (DA) |
| 33.1 | **82.3** | **63.6** | rf_t (RF) | 67.6 | 78.7 | 55.2 | elm_m (NNET) |
| 36.8 | 81.8 | 62.2 | svm_C (SVM) | 67.6 | 77.8 | 54.2 | SimpleLogistic_w (LMR) |
| 38.0 | 81.2 | 60.1 | svmPoly_t (SVM) | 69.2 | 78.3 | 57.4 | MAB_J48_w (BST) |
| 39.4 | 81.9 | 62.5 | rforest_R (RF) | 69.8 | 78.8 | 56.7 | BG_REPTree_w (BAG) |
| 39.6 | 82.0 | 62.0 | elm_kernel_m (NNET) | 69.8 | 78.1 | 55.4 | SMO_w (SVM) |
| 40.3 | 81.4 | 61.1 | svmRadialCost_t (SVM) | 70.6 | 78.3 | 58.0 | MLP_w (NNET) |
| 42.5 | 81.0 | 60.0 | svmRadial_t (SVM) | 71.0 | 78.8 | 58.23 | BG_RandomTree_w (BAG) |
| 42.9 | 80.6 | 61.0 | C5.0_t (BST) | 71.0 | 77.1 | 55.1 | mlm_R (GLM) |
| 44.1 | 79.4 | 60.5 | avNNet_t (NNET) | 71.0 | 77.8 | 56.2 | BG_J48_w (BAG) |
| 45.5 | 79.5 | 61.0 | nnet_t (NNET) | 72.0 | 75.7 | 52.6 | rbf_t (NNET) |
| 47.0 | 78.7 | 59.4 | pcaNNet_t (NNET) | 72.1 | 77.1 | 54.8 | fda_R (DA) |
| 47.1 | 80.8 | 53.0 | BG_LibSVM_w (BAG) | 72.4 | 77.0 | 54.7 | lda_R (DA) |
| 47.3 | 80.3 | 62.0 | mlp_t (NNET) | 72.4 | 79.1 | 55.6 | svmlight_C (NNET) |
| 47.6 | 80.6 | 60.0 | RotationForest_w (RF) | 72.6 | 78.4 | 57.9 | AdaBoostM1_J48_w (BST) |
| 50.1 | 80.9 | 61.6 | RRF_t (RF) | 72.7 | 78.4 | 56.2 | BG_IBk_w (BAG) |
| 51.6 | 80.7 | 61.4 | RRFglobal_t (RF) | 72.9 | 77.1 | 54.6 | ldaBag_R (BAG) |
| 52.5 | 80.6 | 58.0 | MAB_LibSVM_w (BST) | 73.2 | 78.3 | 56.2 | BG_LWL_w (BAG) |
| 52.6 | 79.9 | 56.9 | LibSVM_w (SVM) | 73.7 | 77.9 | 56.0 | MAB_REPTree_w (BST) |
| 57.6 | 79.1 | 59.3 | adaboost_R (BST) | 74.0 | 77.4 | 52.6 | RandomSubSpace_w (DT) |
| 58.5 | 79.7 | 57.2 | pnn_m (NNET) | 74.4 | 76.9 | 54.2 | lda2_t (DA) |
| 58.9 | 78.5 | 54.7 | cforest_t (RF) | 74.6 | 74.1 | 51.8 | svmBag_R (BAG) |
| 59.9 | 79.7 | 42.6 | dkp_C (NNET) | 74.6 | 77.5 | 55.2 | LibLINEAR_w (SVM) |
| 60.4 | 80.1 | 55.8 | gaussprRadial_R (OM) | 75.9 | 77.2 | 55.6 | rbfDDA_t (NNET) |
| 60.5 | 80.0 | 57.4 | RandomForest_w (RF) | 76.5 | 76.9 | 53.8 | sda_t (DA) |
| 62.1 | 78.7 | 56.0 | svmLinear_t (SVM) | 76.6 | 78.1 | 56.5 | END_w (OEN) |
| 62.5 | 78.4 | 57.5 | fda_t (DA) | 76.6 | 77.3 | 54.8 | LogitBoost_w (BST) |
| 62.6 | 78.6 | 56.0 | knn_t (NN) | 76.6 | 78.2 | 57.3 | MAB_RandomTree_w (BST) |
| 62.8 | 78.5 | 58.1 | mlp_C (NNET) | 77.1 | 78.4 | 54.0 | BG_RandomForest_w (BAG) |
| 63.0 | 79.9 | 59.4 | RandomCommittee_w (OEN) | 78.5 | 76.5 | 53.7 | Logistic_w (LMR) |
| 63.4 | 78.7 | 58.4 | Decorate_w (OEN) | 78.7 | 76.6 | 50.5 | ctreeBag_R (BAG) |
| 63.6 | 76.9 | 56.0 | mlpWeightDecay_t (NNET) | 79.0 | 76.8 | 53.5 | BG_Logistic_w (BAG) |
| 63.8 | 78.7 | 56.7 | rda_R (DA) | 79.1 | 77.4 | 53.0 | lvq_t (NNET) |
| 64.0 | 79.0 | 58.6 | MAB_MLP_w (BST) | 79.1 | 74.4 | 50.7 | pls_t (PLSR) |
| 64.1 | 79.9 | 56.9 | MAB_RandomForest_w (BST) | 79.8 | 76.9 | 54.7 | hdda_R (DA) |
| 65.0 | 79.0 | 56.8 | knn_R (NN) | 80.6 | 75.9 | 53.3 | MCC_w (OEN) |
| 65.2 | 77.9 | 56.2 | multinom_t (LMR) | 80.9 | 76.9 | 54.5 | mda_R (DA) |
| 65.5 | 77.4 | 56.6 | gcvEarth_t (MARS) | 81.4 | 76.7 | 55.2 | C5.0Rules_t (RL) |
| 65.5 | 77.8 | 55.7 | glmnet_R (GLM) | 81.6 | 78.3 | 55.8 | lssvmRadial_t (SVM) |
| 65.6 | 78.6 | 58.4 | MAB_PART_w (BST) | 81.7 | 75.6 | 50.9 | JRip_t (RL) |
| 66.0 | 78.5 | 56.5 | CVR_w (OM) | 82.0 | 76.1 | 53.3 | MAB_Logistic_w (BST) |
| 66.4 | 79.2 | 58.9 | treebag_t (BAG) | 84.2 | 75.8 | 53.9 | C5.0Tree_t (DT) |
| 66.6 | 78.2 | 56.8 | BG_PART_w (BAG) | 84.6 | 75.7 | 50.8 | BG_DecisionTable_w (BAG) |
| 66.7 | 75.5 | 55.2 | mda_t (DA) | 84.9 | 76.5 | 53.4 | NBTree_w (DT) |

Figure 3.1: The average performance of Classifiers
http://jmlr.org/papers/volume15/delgado14a/delgado14a.pdf

In this figure, the *RF* means the algorithms based on random forest, *BST* means the algorithms based on boosting and *SVM* means the algorithms based on supported vector machine. These algorithm class usually outperform a lot than other algorithms. Combining our experience and knowledge, we decided to pick the CART based Random Forest in the *RF* class, XGBoost in the *BST* class and SVM with RBF kernel. Also, in order to get a expressive result and help us get good intuitions, we pick Logistic Regression.

### 3.2.1 Pool of Ensemble Learning Classifiers

#### Logistic Regression []

Logistic regression is estimating the parameters of a logistic model which is a form of generalized linear regression.The core of regression method is to find the most suitable parameters for the Logistic function, so that the value of the function and the value of the sample are the closest.

For example,

$$\ell = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \tag{3.16}$$

Logistic regression is also the basic component of many classification algorithms. Its advantage is that the output value naturally falls between 0 and 1, and has probability significance. Because it is essentially a linear classifier, it can not deal with the dependency between features. Although we performance is not strong, it is very expressive. The fitted parameters represent the influence of each feature on the result which may give some intuitive for data analysis.

#### Support Vector Machine

Support vector machines (SVM) are supervised learning algorithms which can be used for both classification and regression problem. SVM is very good at capture the nonlinear relationship of the dataset. Using the kernel trick, we can map the low dimensional non-linear feature space to a high dimensional linear feature space like what the picture shown.

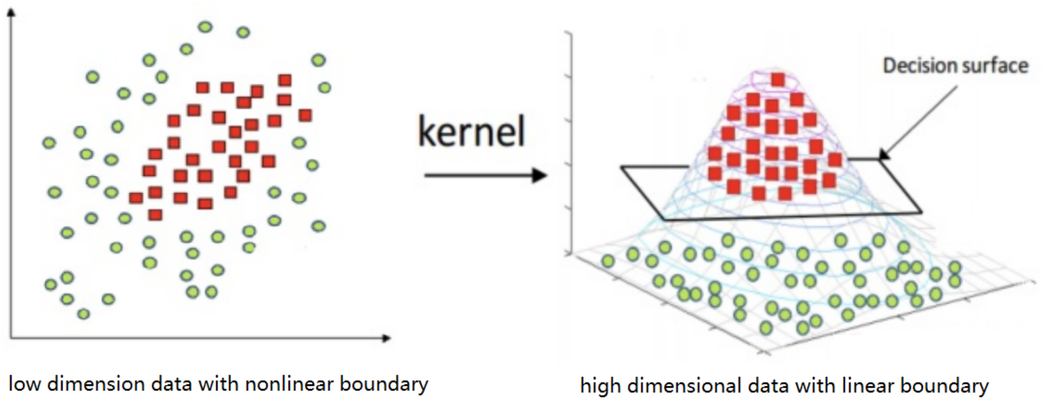

kernel
Decision surface
low dimension data with nonlinear boundary
high dimensional data with linear boundary

Figure 3.2: The kernel trick of SVM

There are some examples for the kernels:

Polynomial (homogeneous):

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j})^d$$

Polynomial (inhomogeneous):

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j} + 1)^d$$

Gaussian radial basis function:

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \exp\left(-\gamma \|\overrightarrow{x_i} - \overrightarrow{x_j}\|^2\right) \text{ for } \gamma > 0$$

Hyperbolic tangent:

$$k(\overrightarrow{x_i}, \overrightarrow{x_j}) = \tanh(\kappa \overrightarrow{x_i} \cdot \overrightarrow{x_j} + c)$$

### Random Forest

Random forests are an ensemble learning algorithm for both classification and regression. It’s ensemble hundreds of decision tree so that the Random decision forests correct for decision trees’ habit of over-fitting to their training set. Random forest uses an improved tree learning algorithm, which selects a random feature subset on each candidate segmentation point in the learning process. [10] This process is sometimes referred to as "feature bagging". The reason for this is the tree correlation in a normal bootstrap sample: if one or more features are very strong predictors of response variables (target output), then these features will be selected in many B trees, so that they become dependent with each other.

The advantage of random forest is that it does not need to deal with the multicolinearity problem. If features are highly dependent, the algorithm will choose the most suitable one by some information metric like Gini index and information gain.

### XGBoost

Extreme gradient boosting is a machine learning algorithm for both classification and regression. In contrast to random forest based on the thought of Bagging which decrease the variance. The gradient boosting is based on thought of boosting which decrease bias from weak machine learning algorithms (usually decision trees). It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The advantage lies in that it has its own feature selection and only uses the features found in the training set. Thus, the number of features need to be calculated decreases which relieve the high dimension problems in some extent. In the Kaggle competition, the winner usually construct a XGBoost algorithm or a ensemble model with XGBoost as a part of it.

### 3.2.2 Walk-Forward Optimization

After model training, the next step is to apply our classifiers to out-of-sample data to evaluate their performances. One common way in machine learning is cross-validation [8] which partitions the entire data set into multiple disjoint subsets (denote the total number of subsets as k). Every time we use k - 1 subsets to train our classifiers and leave one subset for validation. After k times, we can obtain the overall performance for the entire data set to evaluate the generality of machine learning algorithms.

In time series or other sequential data, we need to consider the order of data in a logical way. For example, we can only use data in the past to predict the future instead of the other way around. Therefore, we introduce a modified version of cross-validation, namely walk-forward optimization, which is frequently used in finance.

Instead of randomly splitting the data set, walk-forward optimization partitions the data into different time periods. We start with the first period as our training samples and validate our results in the next period. After that, we move to the next period based on the increment of step size as training samples and validate our results using the period after the next. Following this procedure, we can evaluate the performances of models in every window in a sequential order.

For every asset class, we set a window size of 10 years period (where we use 8 years for training and the rest of 2 years for validation). In each window size, we choose features from 871 independent variables using feature selection method as described in (3.1). After feature selection, we train every classifier in training set and validate their performances in testing set. We set the step size to be 18 months. As a result, in a total period of 380 months, we obtain 15 different classifiers for every asset class.

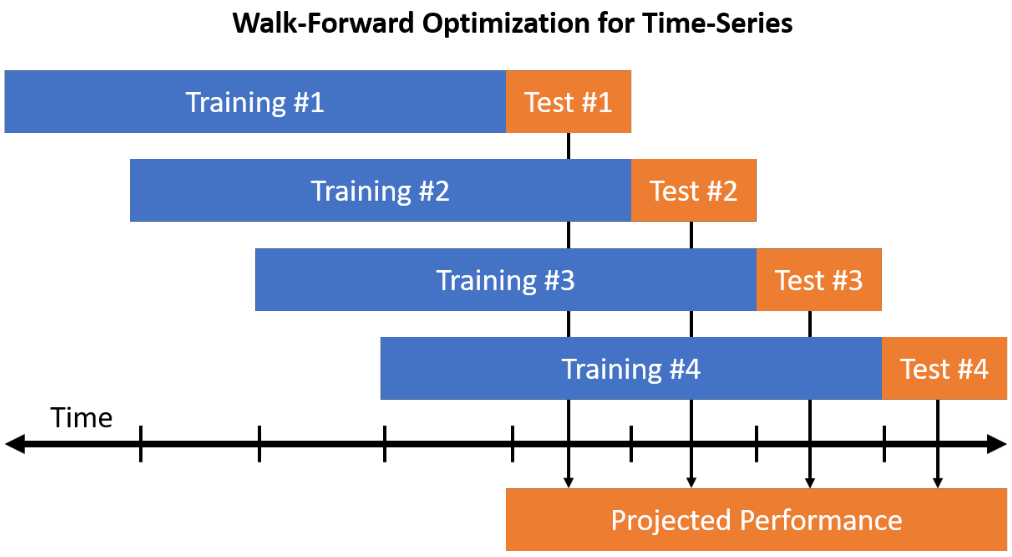


Figure 3.3: Walk-Forward Optimization
https://conlanscientific.com/posts/category/blog/post/avoiding-data-leakage-machine-learning/

## 3.3 Backtest

Since there are a large amount of missing data in CRB Metals Index, this asset will not be included in any of our portfolios.

We use the first 60 months from December 1987 to November 1992 to calculate the initial expected returns and covariance matrix of the returns of 14 assets.

The starting capital is set to 1 million dollars. In the following sections, we will construct 7 portfolios, 3 of them are passive portfolios and 4 of them are active portfolios. These portfolios will be rebalanced every month or every three months based on the changing expected returns and classification signals.

Note that in all the following portfolios, we do not short any asset, which means we will have a non-negative weight for each asset.

### 3.3.1 Passive Portfolios

#### Equal weights (EW)

The equal weights portfolio allocates equal capital to each asset, and therefore the portfolio return is the average return of each asset.

### Mean-Variance Optimization (MVO)

The mean-variance optimization [15] tries to find a weight of portfolio which maximizes the expected return and minimizes the volatility.

$$\max_{w} w^T \mu - w^T \Sigma w$$

$$s.t.\ w^T \mathbf{1} = 1, \quad w_i \geq 0$$

For every period, we solve the quadratic optimization for portfolio weights using the expected returns and covariance matrix of all asset classes as described above.

### Risk Parity (RP)

The risk parity portfolio [14] is another portfolio strategy which allocates the weight based on asset class volatility. One of the commonly used method in risk parity is equally-weighted risk construction portfolio method. We start with a portfolio of equally-weighted asset classes, and compute the overall volatility of this portfolio.

$$\sigma(w) = \sqrt{w^T \Sigma w}$$

Next, we compute the risk contribution for every asset class.

$$\sigma_i(w) = \frac{w_i (\sum w_i)}{\sqrt{w^T \Sigma w}}$$

We can find the risk-parity portfolio weight by solving the quadratic optimization problem:

$$\min_{w} \sum_{i} \left( w_i - \frac{\sigma^2(w)}{(\sum w_i) N} \right)^2$$

where N is the total number of asset classes.

### 3.3.2 Active Portfolios

To compare with passive portfolios, we also want to construct the active portfolios which incorporate machine learning forecast. We only select assets with positive predictions for the next time period and construct portfolio strategies based on Equal Weights (EW), Mean-Variance Optimization (MVO), and Risk Parity (RP).

There is one exception: If our machine learning predicts less than 2 asset classes with positive returns, we will not construct portfolios in this situation. Instead, we will keep our current portfolios until the next period when classifiers predict more than 1 positive returns.

In addition to these three portfolios, we also apply the framework of Black Litterman, which will use all the predictions whether they are positive or negative.

### EW with Machine Learning

The equal weights with machine learning portfolio combines classification results given by our ensemble learning algorithms with equal weights. The portfolio return is the average return of the assets with positive predictions.

### MVO with Machine Learning

We use our machine learning forecasts with positive returns to compute expect returns and covariance matrix, which computes the optimal portfolio similar to mean-variance optimization in passive models.

### RP with Machine Learning

Similarly, we calculate the equally-weighted risk parity weight only using asset classes with positive returns based on our forecast.

### Black Litterman

To introduce the Black-Litterman model, we follow the notations in [7] and [13]. The posterior mean and covariance matrix given by Black Litterman are:

$$\begin{aligned} E_{BL}(R) &= [(\tau\Sigma)^{-1} + P^T\Omega^{-1}P]^{-1}[(\tau\Sigma)^{-1}\pi + P^T\Omega^{-1}q] \\ \Sigma_{BL}(R) &= \Sigma + [(\tau\Sigma)^{-1} + P^T\Omega^{-1}P]^{-1} \end{aligned}$$

where

$\tau$ is the uncertainty in equilibrium return
$\pi$ is the equilibrium prior mean return
$\Sigma$ is the prior covariance matrix
$P$ is the descriptions matrix of views
$q$ is the vector of views
$\Omega$ is the uncertainty in views

As discussed in [13], a smaller $\tau$ can be interpreted as more confidence in equilibrium return, given the expected return $E(R)$ follows a normal distribution $N(\pi, \tau\Sigma)$. Typically $\tau$ takes value between 0.01 and 0.05. In our backtest, we set $\tau$ to be 0.025.

Since we will compare the performance of Black Litterman with mean-variance optimization and other benchmark portfolios, we set the equilibrium return as the historical mean return, which is what we will have for the portfolio optimization if we don't have any view.

$\Sigma$ is set as the historical covariance matrix computed by using a window of 60 months' data, which is updated every one month or every three months.

The $P$ matrix describes the views on assets. In our case, classification predictions tell us whether the asset is bullish or bearish in the next time period. Intuitively, we will long the bullish one and short the bearish one. To incorporate these qualitative views in the Black-Litterman model, we follow [13] to set $P$ as a diagonal matrix, with diagonal value equal to 1 if it's a bullish view and -1 if it's a bearish view.

To illustrate this idea, suppose we only have 3 assets: oil, equities and gold, and our models predict that oil and equities prices will go up, while gold price will go down, then we will construct the $P$ matrix as:

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

In our case, there are 14 assets and our ensemble learning classifiers predict the direction for each of them, and therefore we have a $14 \times 14$ diagonal matrix as $P$.

The relationship between the $P$ matrix and the view vector $q$ is given by:

$$P \cdot E(R) \sim N(q, \Omega)$$

In [13], $q$ is set as:

$$q_i = \sigma_{ai} \cdot IC_i \cdot \sqrt{BR_i}$$

where

$\sigma_{ai} = \sqrt{diag(P \Sigma P^T)_i}$ is the tracking error of the $i$-th view.

$IC_i$ is the Information Coefficient of the $i$-th view, which can be interpreted as the probability of that particular view to happen. We use different information coefficients in our one-month and three-month portfolio. For the one-month portfolio, $IC_i = 0.2$ for every asset; for the three-month portfolio, $IC_i$ is set as $2 \cdot Accuracy_i - 1$, where $Accuracy_i$ is the average out-of-sample accuracy of our ensemble learning classifiers on the historical data.

$BR_i$ is the strategy's breadth of the $i$-th view, which is defined as the number of independent forecasts made per year. For the one-month portfolio, $BR_i = 12$; for the three-month portfolio, $BR_i = 4$.

Finally, as suggested by [6], we set $\Omega$ as:

$$\Omega = diag(P\tau\Sigma P^T)$$

With all these inputs, we are able to use Black Litterman to calculate posterior mean return and covariance matrix, which will then be plugged into mean-variance optimization and give us the optimal weights for each asset in our Black Litterman portfolio.

# 4 Results

## 4.1 Classification Evaluations

For each asset class, we use the following 3 metrics to evaluate our 1-month and 3-month classifications.

### 4.1.1 Accuracy

Accuracy is most commonly used and straight-forward metric in classification. It directly reflects how accurate our prediction is by computing the ratio of all correct predictions over all the samples as the following:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

### 4.1.2 Precision

Precision [16] is another metric for classifier evaluation. It measures how precise our model prediction can detect the positive samples, which is very important especially when the data set is imbalanced.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

In this project, we need to minimize our false positive rate (our model predicts the asset is increasing but it is actually decreasing) since it will result in the loss of capital when we construct our active portfolios.

### 4.1.3 AUC-ROC

An ROC curve (receiver operating characteristic curve) [3] is a graph demonstrating the relationship of classification model between true positive rate (TPR) and false positive rate (FPR) given different thresholds.

The Area Under the Curve (AUC-ROC) is defined as

$$A = \int_{x=0}^{1} TPR[FPR^{-1}(x)]dx$$

If AUC is larger, our model will be better since it produces a higher true positive rate or a lower false rate given the same threshold. Thus, we expect our AUC-ROC can approach 1 as best as possible.

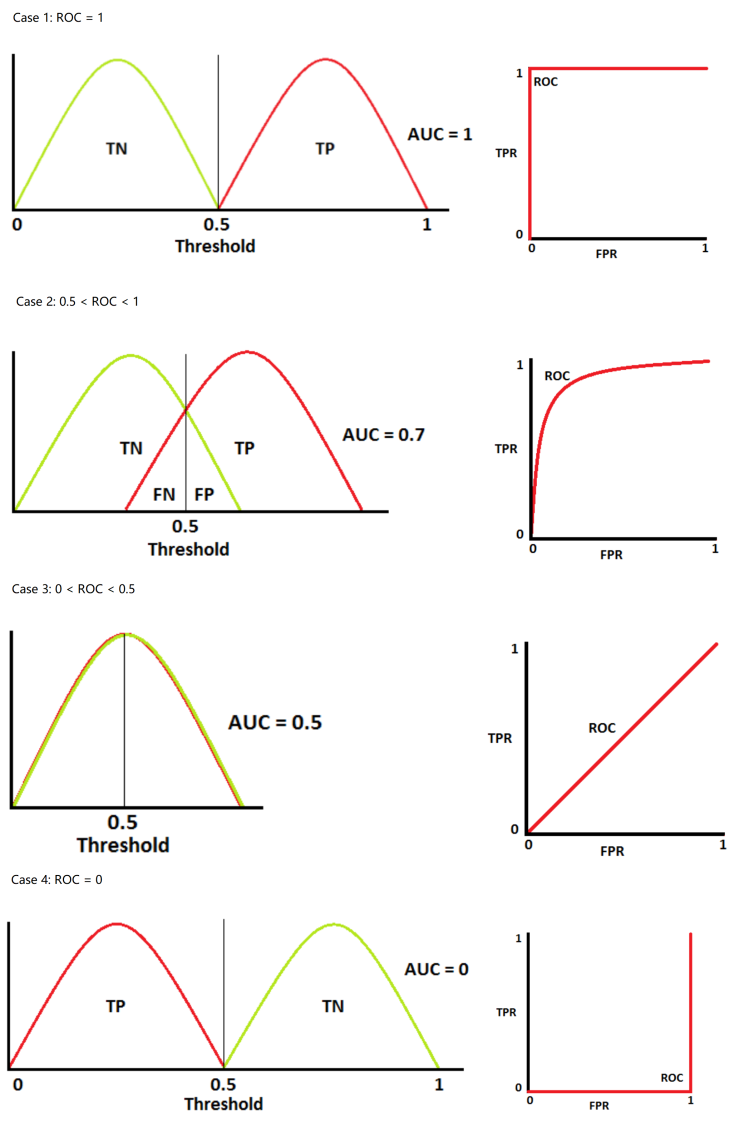
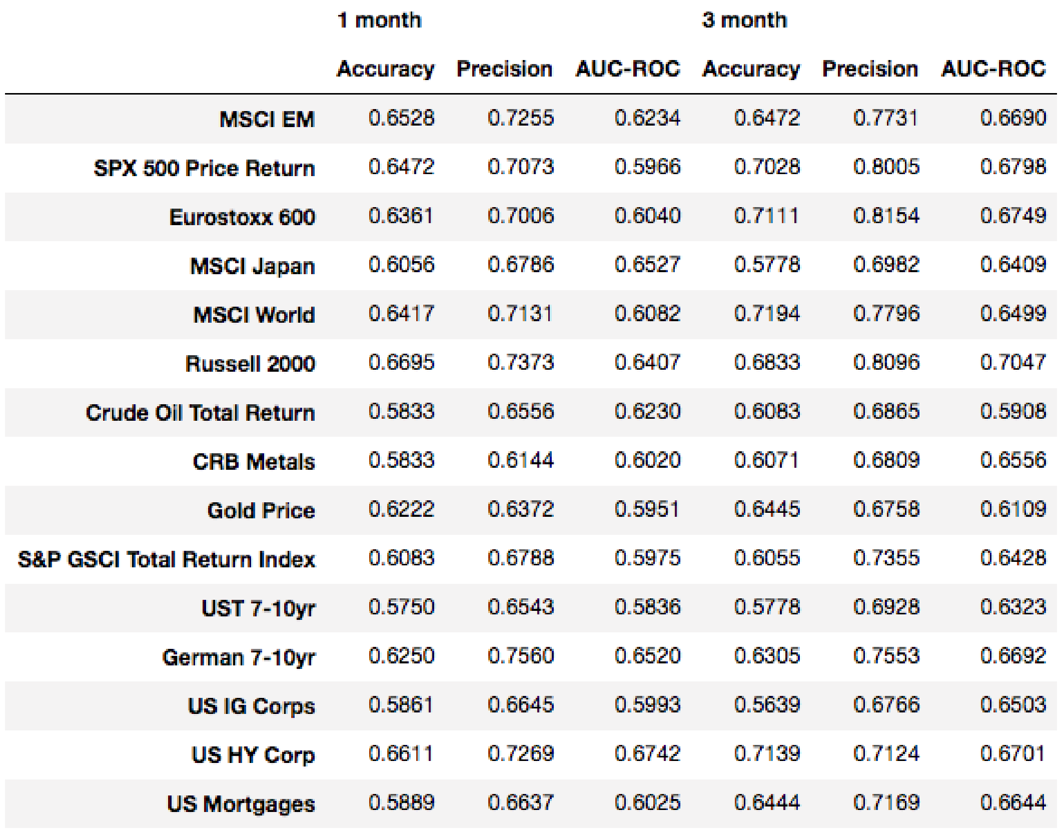


Figure 4.1: AUR-ROC Curves



| | 1 month | | | 3 month | | |
|---|---|---|---|---|---|---|
| | Accuracy | Precision | AUC-ROC | Accuracy | Precision | AUC-ROC |
| **MSCI EM** | 0.6528 | 0.7255 | 0.6234 | 0.6472 | 0.7731 | 0.6690 |
| **SPX 500 Price Return** | 0.6472 | 0.7073 | 0.5966 | 0.7028 | 0.8005 | 0.6798 |
| **Eurostoxx 600** | 0.6361 | 0.7006 | 0.6040 | 0.7111 | 0.8154 | 0.6749 |
| **MSCI Japan** | 0.6056 | 0.6786 | 0.6527 | 0.5778 | 0.6982 | 0.6409 |
| **MSCI World** | 0.6417 | 0.7131 | 0.6082 | 0.7194 | 0.7796 | 0.6499 |
| **Russell 2000** | 0.6695 | 0.7373 | 0.6407 | 0.6833 | 0.8096 | 0.7047 |
| **Crude Oil Total Return** | 0.5833 | 0.6556 | 0.6230 | 0.6083 | 0.6865 | 0.5908 |
| **CRB Metals** | 0.5833 | 0.6144 | 0.6020 | 0.6071 | 0.6809 | 0.6556 |
| **Gold Price** | 0.6222 | 0.6372 | 0.5951 | 0.6445 | 0.6758 | 0.6109 |
| **S&P GSCI Total Return Index** | 0.6083 | 0.6788 | 0.5975 | 0.6055 | 0.7355 | 0.6428 |
| **UST 7-10yr** | 0.5750 | 0.6543 | 0.5836 | 0.5778 | 0.6928 | 0.6323 |
| **German 7-10yr** | 0.6250 | 0.7560 | 0.6520 | 0.6305 | 0.7553 | 0.6692 |
| **US IG Corps** | 0.5861 | 0.6645 | 0.5993 | 0.5639 | 0.6766 | 0.6503 |
| **US HY Corp** | 0.6611 | 0.7269 | 0.6742 | 0.7139 | 0.7124 | 0.6701 |
| **US Mortgages** | 0.5889 | 0.6637 | 0.6025 | 0.6444 | 0.7169 | 0.6644 |

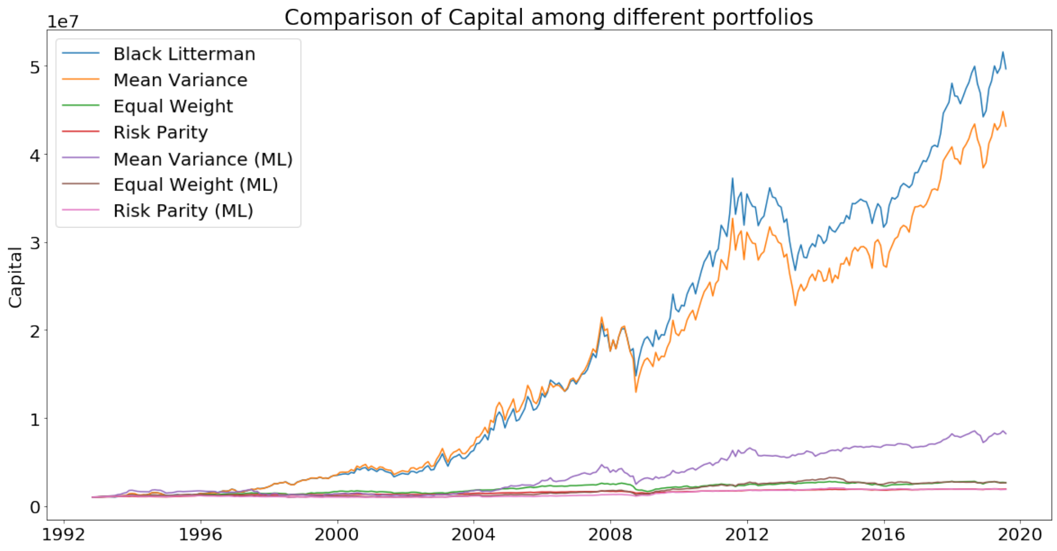
Figure 4.2: Model Classification Summary

Our table above shows the majority of scores of three metrics has a range of between 60 and 70 percents. For each metric, our 3 month forecast produces roughly higher scores than 1 month prediction across various asset classes.

Among all the asset classes in 1-month predictions, Russell 2000 has the highest accuracy (66.95%) while US Treasury of 7-10 years has the lowest accuracy (57.50%). In terms of precision, German of 7 - 10 years has the highest score (75.60%) and CRB Metals Index has the lowest precision (61.44%). For AUC-ROC, US High Yield Corp has the highest score (67.42%) while US Treasury of 7-10 years is the lowest (58.36%).

For 3-month forecasts, the performances of asset classes are quite different than 1-month predictions. MSCI World Index produces the highest accuracy rate (71.94%) while the lowest accuracy comes from MSCI Japan Index and US Treasury of 7-1o years (57.78%). In terms of precision rates, 3 asset classes are above 80 percent (SPX 500, Eurostoxx 600, and Russell 2000), among which Eurostoxx 600 Index is the highest (81.54%). Regarding AUC-ROC scores, Russell 2000 is the only asset class with over 70% and the lowest one is Crude Oil with less than 60% (59.08%).

## 4.2 1-month Backtest



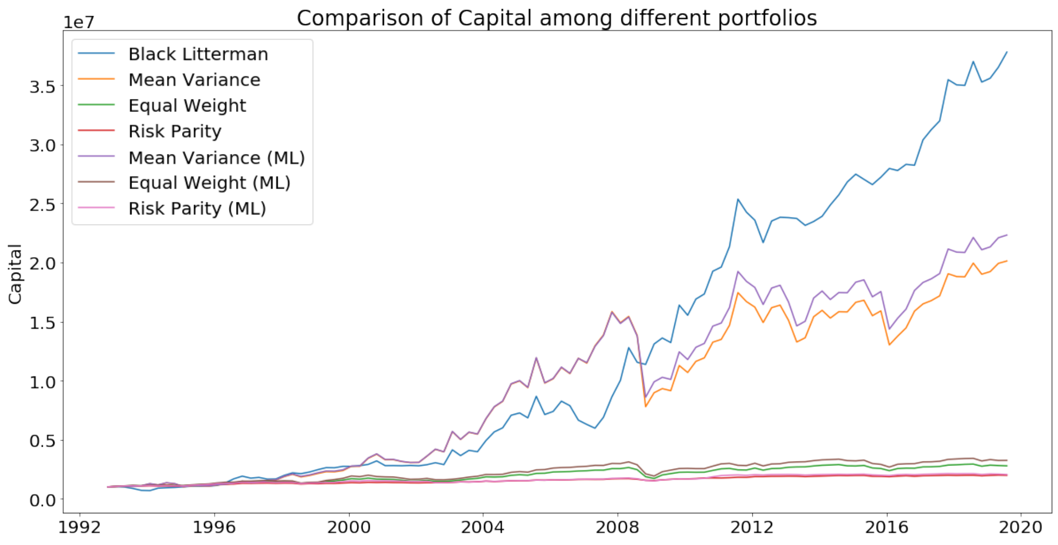
According to our one-month backtest results, before the year of 2009, the performance of Black Litterman and passive mean-variance optimization. One of the potential reasons is that there exists a lot of missing values in our feature pools. This could restrict the choice of features and affect machine learning forecast. After the year of 2009, the performance of Black Litterman portfolio is better than any other portfolios.

If we compare the performances of mean-variance optimization between passive and active models, we can see the passive one is much better than the active one as time goes by. When we evaluate the machine learning forecast, it usually only predicts 4 - 5 asset classes with positive returns. As a result, the portfolio considers fewer assets than the passive one. In terms of other portfolios such as equal weight and risk parity, their performances are very similar to each other.

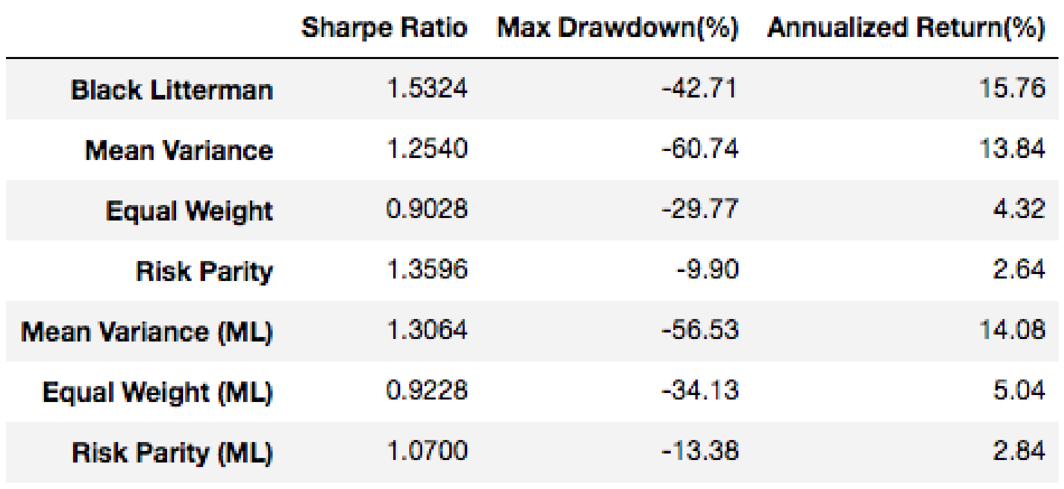
| | Sharpe Ratio | Max Drawdown(%) | Annualized Return(%) |
|---|---|---|---|
| **Black Litterman** | 3.0156 | -29.95 | 16.44 |
| **Mean Variance** | 2.8068 | -34.48 | 16.08 |
| **Equal Weight** | 1.6224 | -21.21 | 4.08 |
| **Risk Parity** | 2.1168 | -9.66 | 2.52 |
| **Mean Variance (ML)** | 1.9464 | -37.74 | 9.24 |
| **Equal Weight (ML)** | 1.5636 | -19.54 | 4.08 |
| **Risk Parity (ML)** | 1.5396 | -12.82 | 2.52 |

Based on the summary statistics, the Black Litterman model produces the highest Sharpe Ratio as well as largest Annualized Returns. In addition, its max drawdown (29.95%) is lower than mean-variance of both passive (34.48%) and active models (37.74%) but it's higher than equal weight and risk parity portfolios.

## 4.3 3-month Backtest



In 3-month backtest, the performances of passive and active mean variance optimization overlap before the year of 2009, and they are better than Black Litterman portfolio. After 2009, the Black Litterman outperforms other portfolios, and active MVO is better than the passive one. Compared with 1-month forecast, our 3-month machine learning predicts a range of between 10 and 14 positive assets. In terms of equal weight, the active model is slightly better than the passive one.



| | Sharpe Ratio | Max Drawdown(%) | Annualized Return(%) |
|---|---|---|---|
| **Black Litterman** | 1.5324 | -42.71 | 15.76 |
| **Mean Variance** | 1.2540 | -60.74 | 13.84 |
| **Equal Weight** | 0.9028 | -29.77 | 4.32 |
| **Risk Parity** | 1.3596 | -9.90 | 2.64 |
| **Mean Variance (ML)** | 1.3064 | -56.53 | 14.08 |
| **Equal Weight (ML)** | 0.9228 | -34.13 | 5.04 |
| **Risk Parity (ML)** | 1.0700 | -13.38 | 2.84 |

To summarize the portfolio performances, again, the Black-Litterman produces the highest Sharpe Ratio and annualized return. In addition, its max drawdown is lower than mean-variance portfolios, but much higher than any other portfolios. In 3-month backtest, all the active models demonstrate higher Sharpe Ratios and annualized returns than their corresponding passive models.

## 5 Conclusion

In this project, we analyze the effectiveness of ensemble learning in tactical asset allocation. We use monthly price data of 15 asset classes and monthly data of 68 factors. We transform the original data to get more features. We then use feature selection techniques to choose factors that have large contributions to our prediction. After feature selection, we retain the top 10 features for each asset during each time period. We formulate a binary classification problem by predicting the future directions of asset prices. We create a pool of ensemble learning classifiers and use walk-forward optimization to tune the hyper-parameters. We use the classification results given by our optimal models to generate qualitative views and incorporate them into the Black-Litterman model to get the posterior expected returns and covariance matrix. For backtest, we construct 7 portfolios including both passive portfolios and active portfolios. We compare the performance of Black Litterman portfolio and other benchmark portfolios and conclude that our Black Litterman portfolio based on ensemble learning achieves a better result.

We have the following conclusions:

1. Feature quality is very important in classification tasks. We initially transform 68 factors to 871. If we use all the factors, our models will capture part of the noise from the dataset, which will cause over-fitting. Thus, feature selection is essential before training the classifiers. Our absolute co-moment metric performs an efficient way of measuring both linear and non-linear dependency between features and responses.

2. Ensemble learning is very powerful in combining various weak learners into strong learners. Due to different designs or optimization algorithms, various models may not be robust to different inputs and outputs. Hence, making the majority vote or computing weighted average of different model outputs can effectively reduce the variance of predictions, i.e., reduce the risk of over-fitting.

3. Incorporating our machine learning forecasts into Black Litterman improves our portfolio performance compared with other portfolios, since it produces higher Sharpe Ratio and annualized return in both 1-month and 3-month backtests. This means machine learning models actually provide some guidance on investments that can help asset managers make better decisions.

We also have the following limitations:

1. The max drawdowns of our Black Litterman portfolios in both 1-month and 3-month backtests are very large, which is partially due to the 2008 financial crisis. We can observe that almost all portfolios have a large drawdown during that period of time. Another reason that causes our portfolios to have a large drawdown is that we constrain the weights of each asset to be non-negative, which is a requirement of tactical asset allocation. During the time when all asset prices decline, our PL will suffer from a large drawdown.

2. In our backtest, we do not include transaction cost, which is also an important part of portfolio construction in real world. When transaction costs are considered, we cannot make big changes to our portfolio weights too frequently. Also, if different asset classes have

different impact models, we need to add more terms to the objective function that will be optimized. It is possible that when we include transaction cost, we will get totally different optimal weights for each asset during the same time period.

3. Our machine learning algorithms may suffer from the numerous missing values of features in early period of time, since our backtest shows the Black Litterman portfolio does not perform better than mean-variance optimization. We may infer one of the reasons is due to the missing values of many indices before the year of 2009, which highly restrict our choice of factors for training classifiers.

To further improve our project, we have the following suggestions:

1. Analyze the time when our portfolios suffer from large drawdowns. As we have mentioned above, our optimal Black Litterman and mean variance optimization with ML portfolios have large drawdowns during the 2008 financial crisis. Since in our case we cannot short assets, we may add constraints to prevent our portfolios from taking positions when most of the markets go down.

2. Consider transaction cost in our backtest. When transaction cost is considered, the backtest results will be more useful for asset managers to make decisions. To incorporate transaction cost to our portfolio construction, we need an impact model for each asset. Instead of optimizing portfolio weights, we need to optimize the difference between previous weights and potential weights.

3. There are multiple ways to handle missing value problems. Based on this situation, we can build predictive models from other existing features to infer the missing values. Also, we can create a new feature which records number of missing values for every observation since those missing values may contain some useful information.

## References

[1] Holder's inequality, https://en.wikipedia.org/wiki/h

[2] Tactical asset allocation, https://en.m.wikipedia.org/wiki/tactical_asset_allocation.

[3] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[4] S. B. Fernandez-Delgado, Eva Cernadas. Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research*, 2014.

[5] S. Guiasu. *Information Theory with Applications*, volume 1. McGraw-Hill, 1977.

[6] G. He and R. Litterman. The intuition behind black-litterman model portfolios. *Goldman Sachs Asset Management Working Paper*, 1999.

[7] T. Idzorek. A step-by-step guide to the black litterman model: Incorporating user-specified confidence levels. *Technical report, Duke University*, 2002.

[8] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[9] P. Kolm, R. Tütüncü, and F. Fabozzi. 60 years of portfolio optimization: Practical challenges and current trends. *European Journal of Operational Research*, 234(2):356–371, 2014.

[10] I. G. Krzysztof Gajowniczek and T. Zabkowski. Reducing false arrhythmia alarms using different methods of probability and class assignment in random forest learning methods. *Sensors (Basel)*, 2019.

[11] M. Kuhn and K. Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*, volume 1. Chapman and Hall/CRC, 2019.

[12] L. P. Kuptsov. Holder inequality. *Springer Science+Business Media B.V. / Kluwer Academic Publishers*, 2001.

[13] A. Maggiar. Active fixed-income portfolio management using the black-litterman model. *Master's thesis, Imperial College of Science, Technology and Medicine*, 2009.

[14] S. Maillard, T. Roncalli, and J. Teiletche. Equally-weighted risk contributions: a new method to build risk balanced diversified portfolios, 2008.

[15] S. Mannor and J. Tsitsiklis. Mean-variance optimization in markov decision processes. *arXiv preprint arXiv:1104.5601*, 2011.

[16] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.