UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dipartimento di Ingegneria "Enzo Ferrari"

Master's Degree in Artificial Intelligence Engineering

A Practical Study of Ensemble Learning for Multi-Horizon Financial Forecasting

Supervisor: Prof. Carlo Augusto Grazia

External Supervisor: Alessandro Lambertini - Axyon AI

Candidate: Federico Perico

A.Y. 2024/25

Abstract

This thesis presents a practical study of ensemble learning techniques applied to the field of financial forecasting, conducted at Axyon AI, a fintech company which offers AI-based insights, asset signals and investment strategies.

A central focus is on multi-horizon forecasting: integrating predictions from weak learners trained on different investment horizons in order to improve overall accuracy.

The experiments were carried out on the Japan Target Market dataset, across 20-day and 60-day horizons. The dataset is based on the Morningstar Japan Target Market Exposure index, which measures the performance of large-cap and mid-cap stocks in Japan, and covers the top 85% of the market by capitalization.

The tested Horizon Union methods demonstrated superior performance when compared to the 20-day single-horizon baseline. These results highlight the practical value of multi-horizon predictions in the context of AI-driven investment strategies.

"what would life be like if we only did what is necessary?"

- Niki Lauda

Contents

List of Figures

List of Tables

1. Introduction

1.1 Axyon AI: overview and technological framework

1.1.1 Axyon AI: overview

This is an experimental thesis fully developed in a company setting at Axyon AI.

Axyon AI is a fintech company founded in Modena in 2016 by Daniele Grassi, Jacopo Credi and Giacomo Barigazzi, which provides professional investment managers with AI-based insights, asset signals and investment strategies.

The company is particularly focused on providing reliable, high-quality products to its clients, while also making sure that every single step of the signal and strategy generation process is fully explainable.

1.1.2 Asset signals

Within the context of Axyon AI, an asset signal is defined as a daily prediction generated by an AI system, which reflects the expected financial performance of the asset over a specified period of time, also referred to as investment horizon. The concept of investment horizon will be discussed in 1.4.2.

These asset signals can then serve as the basis to generate fully or partially AI-based investment strategies.

From here on, for brevity, asset signals will be referred to simply as signals.

1.1.3 Axyon AI: technological framework

Axyon AI can generate signals following a few different generation processes. The one examined in this study is based on rankings: for each day, every

stock - or other financial instrument - is virtually compared to its peers within a predefined investable universe, by ranking it according to a score generated by an heterogeneous ensemble of machine learning models.

There are two important sources of heterogeneity for the ensembles created by Axyon AI: the implementation of different kinds of supervised learning labels and the heterogeneity of the model architectures used. Both these elements will be further discussed in following sections. The feature bagging approach employed is a further source of heterogeneity. Feature bagging refers to performing the training of models on random subsets of features, in order to reduce overfitting and improve robustness.

Let's present a toy example, for a given day t we might have this situation:

t			$t+h$	
Asset	Signal	Axyon AI rank	Return at day h	Realized rank
Stock A	0.457	6	-3.0%	6
Stock B	0.407	7	-2.0%	5
Stock C	0.529	4	-6.0%	8
Stock D	0.494	5	2.0%	2
Stock E	0.622	1	5.0%	1
Stock F	0.581	2	-5.0%	7
Stock G	0.541	3	1.0%	3
Stock H	0.367	8	-1.0%	4

Table 1.1: Axyon AI signals and ranks example

where h is the investment horizon at which the returns are evaluated.

Once asset signals are available, many different investment strategies can be developed using the signals as a basis. The simplest example of an investment strategy would be to invest the entire capital on the best ranked stock. Of course this is not desirable due to the lack of diversification, and real strategies are much more sophisticated.

Whatever the investment strategy is, though, the cumulative return, so the performance of the strategy, will be strongly dependent on the quality of underlying signals. In other words, the more accurate our predictions are and the better any strategy will perform, regardless of how simple or sophisticated it may be.

1.2 Supervised machine learning for ranking

1.2.1 Supervised machine learning

Supervision in machine learning refers to a paradigm in which the objective is to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that maps input features $x \in \mathcal{X}$ to output labels $y \in \mathcal{Y}$, in a way such that f generalizes well to unseen data drawn from the same distribution $\mathcal{X}$. [1]

The fundamental assumption of independence and identical distribution (i.i.d.), which states that, for learning to be possible, each training sample and inference sample must come from the same underlying distribution $\mathcal{X}$ and be independently drawn, is violated when working with financial time-series datasets, in ways that will be detailed in 1.4, but the i.i.d. framework remains the theoretical foundation for this study and its application.

1.2.2 Ranking

Ranking is the problem of ordering a set of items according to a given metric. In the context of financial forecasting, this metric usually corresponds to stock prices or returns. Ranking stock performance is often preferred to pure regression - which would mean directly predicting a value for each stock - due to the inherently noisy nature of the ground truth data. For the same reasons, nonlinear models such as neural networks and random forests are widely used. [2]

There are several possible approaches in order to apply supervised machine learning to a ranking problem, categorized by Liu (2009) [3] as:

1. The pointwise approach
2. The pairwise approach
3. The listwise approach

1.2.3 Ranking: the pointwise approach

As opposed to the listwise approach, which considers all the elements to rank at the same time, and the pairwise approach, which, as the name suggests, considers a pair of items at a time, the pointwise approach consists in scoring each item separately while still considering its position in the ranking. This is achieved either by directly using the ranking positions as the supervised

learning labels, or by using a discrete ordinal framework, such as dividing stocks into categorical groups, for example "stocks expected to perform well" and "stocks expected to perform poorly".

Axyon AI implements both of these approaches in the training of its weak learners, which are hence divided into two main categories, regardless of the underlying model architecture: regression weak learners and classification - or ordinal - weak learners.

1.3 Ensemble learning

1.3.1 Introduction to ensemble learning

Ensemble learning is an established technique in machine learning which consists in exploiting the predictive capabilities of multiple models instead of a single one, in order to obtain a more accurate prediction.

This method has proven to be effective in multiple machine learning scenarios, and in particular in financial forecasting. [2, 4]

Ensemble learning is based on the concept of weak learner, a term used to indicate a single model, of limited capabilities, composing the ensemble. Again, it has been proven that, when applied to financial forecasting, "shallow" ensemble learning often outperforms deep learning. [4]

There are multiple subcategories of ensemble learning, that define exactly how weak learners are trained and aggregated: the most popular are bagging, boosting and stacking.

1.3.2 Homogeneous and heterogeneous ensembles

An ensemble is referred to as homogeneous when all the models that constitute it are derived from the same base architecture. In such ensembles, the individual learners can be trained using different subsets of the data or varying initialization parameters, resulting in distinct trained models, but the underlying model type remains the same.

On the contrary, when an ensemble is made up of weak learners coming from a variety of base architectures, it's called heterogeneous.

1.3.3 Bagging

Bagging is an ensemble learning technique formalized by Breiman in 1996. [5]

The core idea behind bagging is to generate multiple versions of a base model by training each one on a different subset of the original dataset, and then aggregating their predictions. This aggregation is typically done by averaging - sometimes with weights - for regression tasks, and by majority voting for classification tasks.

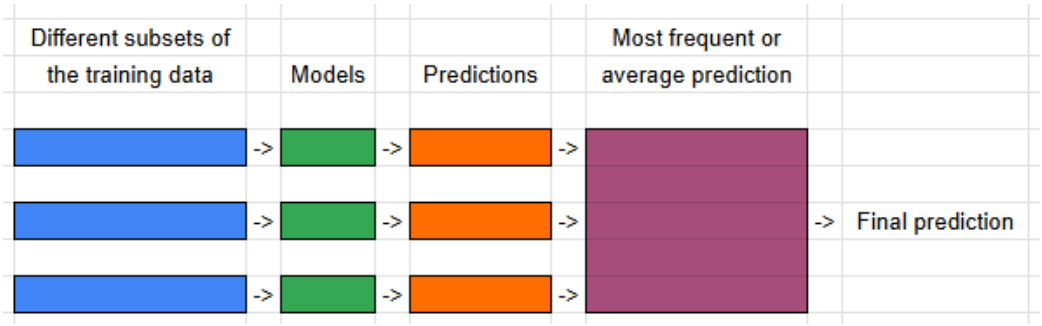

Figure 1.1: Bagging

1.3.4 Boosting

Boosting is an ensemble learning paradigm in which models are trained sequentially, with each successive model focusing on the errors made by its predecessor. As in bagging, the final prediction is usually a weighted sum of predictions for regression, and the result of a majority vote for classification.

Adaboost is one of the first and most popular implementations of a boosting algorithm, which puts boosting into practice by iterating over the training process and adjusting sample weights at each iteration, increasing the weights for misclassified samples. [6]

One drawback of boosting when compared to bagging is that the training process is sequential instead of parallel, leading to increased computational time.

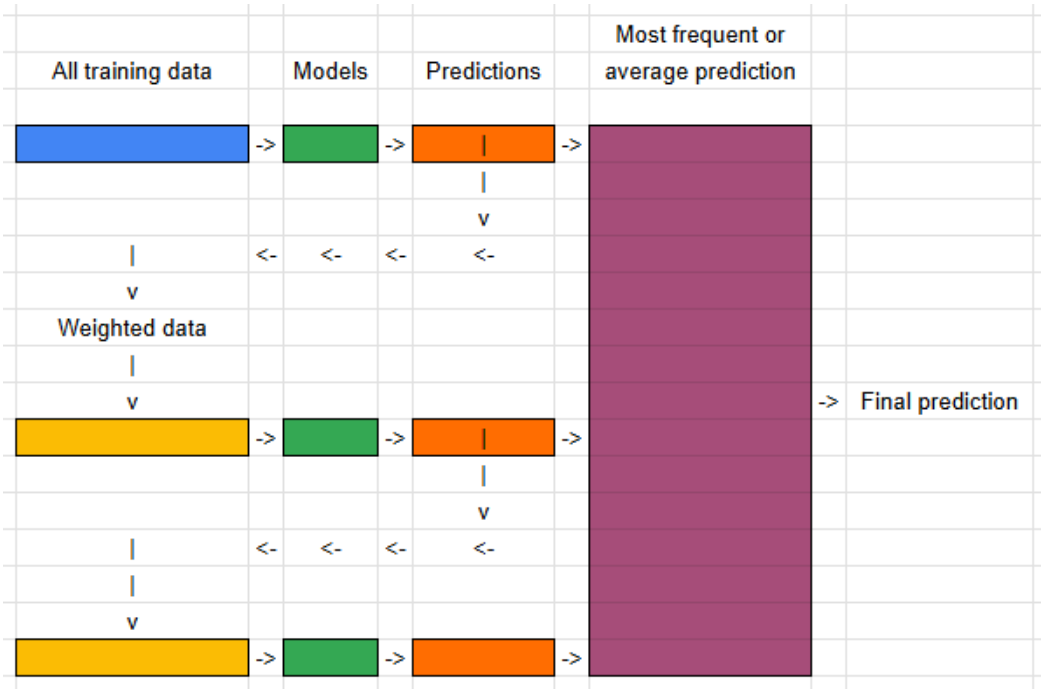

Figure 1.2: Boosting

1.3.5 Stacking

Stacking, also known as meta-learning, refers to an ensemble learning technique which consists in training second-level models - meta-learners - in order to learn how to optimally combine the predictions of first-level models - base learners. [7]

In a typical stacking setup, the base learners are trained on the original dataset, while the meta-learners have as inputs the predictions of the base learners, as labels the true target values from the original dataset, and learn how to dynamically weigh base-learner predictions in order to maximize accuracy. This way, ideally, the meta-learner can exploit complementary strengths and mitigate weaknesses of different models.

Meta-learners are often chosen to be simple models, such as linear regression, to reduce the risk of overfitting.

1.3.6 Weak learner de-correlation

It's important to underline the role of de-correlation among different weak learners composing an ensemble.

To determine the performance of an ensemble, while the individual accuracy of weak learners is of course relevant, de-correlation between them is just as important, because ultimately that's what determines whether ensembling yields performance improvements or not.

To illustrate this principle, let's consider three binary classification weak learners ensembled with majority voting - each with an accuracy of 0.7 - that compose an ensemble. If they are all perfectly correlated with each other, as in, they all generate exactly the same predictions, and therefore make the same mistakes, ensembling them actually produces no improvement at all: the resulting ensemble behaves exactly as any of the single model would - yielding a final accuracy of 0.7, exactly as before ensembling. Instead, if the three models make errors all completely de-correlated with each other, ensembling them brings the accuracy up to 1.0.

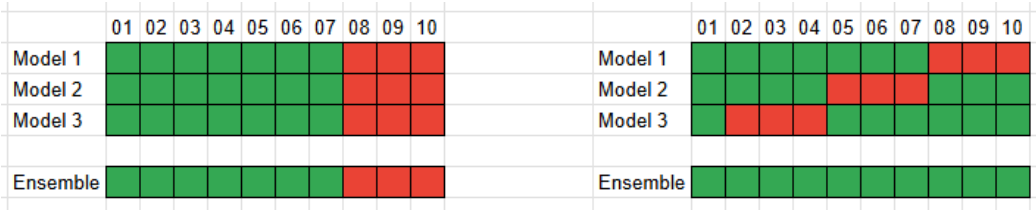

Figure 1.3: Weak learner de-correlation

Therefore, it becomes clear why one major challenge in ensemble selection is to manage the trade-off between raw weak learner performance and their de-correlation. [8]

1.4 Financial time-series data and investment horizon

1.4.1 Financial time-series datasets

Financial time-series datasets are characterized by unique properties, especially in the context of machine learning, in which they pose distinctive challenges, and therefore need to be handled with adequate care. [9]

Firstly, they exhibit a strong time-dependency, also referred to as autocorrelation: for example, today's stock price is strongly influenced by its value from yesterday. This is the main reason why the fundamental assumption of independence and identical distribution (i.i.d.), presented in 1.2.1, is violated when working with financial time-series datasets.

Then, when we examine financial time-series at any granularity level, they suffer of noise. This means that, rather than following smooth patterns, these series display abrupt spikes and random fluctuations, making it difficult for machine learning models to extract reliable patterns.

What's particularly subtle is that, depending on what scale we observe the time-series at, the "signal" and the "noise" become different things. In other words, something that is noise if we consider a whole year can become a very relevant signal if we consider only 5 days. This is one of the reasons why the concept of investment horizon is so important when working with financial time-series datasets.

1.4.2 Investment horizon

The investment horizon is the point in time in which the returns on different investments are evaluated and compared between one another. Especially in a ranking setting, but also in general, the concept of investment horizon is key in evaluating how good an investment is.

Different investment horizons can of course generate different rankings, but most importantly, they generate rankings with different properties. For example, by analyzing real-world data it's clear that longer horizon rankings have significantly less inherent variability from one day to the next one. This inherent stability might make it easier for a machine learning model to predict more accurately rankings with a longer horizon.

Some of the experiments of this study have been performed on two different horizons - 20 days and 60 days - and they have sometimes yielded different results between horizons. This fact further proves how the investment horizon is a key factor to take into consideration when working with financial time-series datasets.

1.4.3 Noise in datasets and the SNR

Noise in datasets is defined as real world data containing irrelevant or misleading, often random, components, which can significantly affect machine learning

tasks. [10]

Noisy datasets appear in various settings such as finance, computer vision, natural language processing, speech recognition and many more.

The Signal to Noise Ratio (SNR) is a measure used to quantify how much noise there is in a certain dataset:

$$\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} \tag{1.1}$$

where P_{signal} is the power of the signal and P_{noise} is the power of the noise. Therefore, a low SNR implies a high level of noise.

Now, in our context, it's very difficult to define precisely what P_{signal} and P_{noise} are, numerically speaking, because as briefly discussed upon in 1.4, the definition of noise and consequently the SNR depend on the investment horizon we're looking at.

But even if we were to fix an investment horizon, it would still be impossible to know a priori if a spike in the data has to be considered as signal or noise. The presence itself of the spike in the data may suggest that it is a genuine signal, reflecting a real-world event that may recur in the future, and as such it would be a desirable piece of information to learn. But at the same time, it could steer the model away from learning more significant and impactful "true" underlying patterns. Hence, there is a trade-off between trying to predict real-world precise data, and how well a model - or an ensemble - is able to learn truly meaningful patterns from the data itself. This trade-off impacts directly on the generalization capabilities of our models and ensembles.

1.5 Motivation and goals

This section aims to explain, mostly from a business point of view, the motivation and goals for this thesis.

For Axyon AI, it is key to provide its clients with the most high-quality signals and strategies possible. Hence, two separate development options for the signal generation process were theorized, aimed at improving signal quality. Both these options will be briefly presented now, and further explained respectively in Chapters 4 and 6.

The first goal was the exploration of new aggregation strategies for signals. In this case, the hypothesis was that the simple daily averaging aggregation

process in use could be optimized.

The second goal was to create a setting in which "collaborating" multi-horizon predictions are possible, and to evaluate if uniting horizons would improve performance. In this case, the hypothesis was that, because, as said in 1.4.2, longer-term rankings are more stable and easier to predict for a machine learning model, trickling down some of this information to shorter-term ensembles could prove to be beneficial.

These goals led me to study and analyze the whole signal generation process, both from a theoretical perspective and from an experimental perspective.

2. Performance Evaluation Methods

2.1 Introduction

This chapter introduces and defines appropriate performance evaluation methods in the context of assessing the quality of daily rankings produced by our ensembles.

When the employed dataset is fixed, like in the experiments performed and presented in this study, evaluating the quality of daily rankings means that we're actually evaluating the quality of the procedures and techniques used for weak learners selection and aggregation.

2.2 Evaluation metrics

2.2.1 Rank IC - RIC

Spearman's Rank Correlation Coefficient or Spearman's Rho is a measure that evaluates rank correlation between two sets.

Spearman's Rank Correlation Coefficient between two sets A and B is:

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} (rank(a_i) - rank(b_i))^2}{n(n^2 - 1)} \tag{2.1}$$

where a_i and b_i are the i-th observations in the two sets, and n is the number of observations.

Formula 2.1 is exact when there are no tied ranks. When ties occur, and average, equal, ranks are therefore assigned, Spearman's Rank Correlation is instead computed as the Pearson correlation between the rank-transformed variables:

$$\rho = \frac{\text{cov}(\, \textit{rank}(A),\ \textit{rank}(B)\,)}{\sigma(\textit{rank}(A))\,\sigma(\textit{rank}(B))} \tag{2.2}$$

Rank Information Coefficient (Rank IC) is defined as Spearman's Rank Correlation Coefficient computed between a predicted ranking and an actual - also called "realized" - ranking. In symbols:

$$\text{Rank IC} = \rho(\textit{rank}(\text{predicted}), \textit{rank}(\text{actual})) \tag{2.3}$$

Rank IC can range between a maximum value of *+1*, meaning that the two sets have a perfect monotonic relationship - or in other words, that they have the same ranking order - and a minimum value of *-1*, meaning that the two rankings are exactly opposite.

Considering that we produce forecasts in the form of rankings, and that we need to compare them with realized rankings, Rank IC may seem like an ideal performance measure. In reality, it suffers from a few drawbacks. The main one is that Rank IC is not sensitive to the actual magnitudes of underlying returns.

To better highlight this limitation, let's examine a practical example:

	Realized returns	Realized ranking
Stock A	10.0%	1
Stock B	8.0%	2
Stock C	0.0%	3
Stock D	-10.0%	4

Table 2.1: Rank IC example - Realized returns and ranking

Let's imagine that our investment strategy is simply to go long on the best ranked stock, and short on the worst ranked stock.

Predicted rankings	Rank IC	Return
1: B, 2: A, 3: C, 4: D	0.8	18.0%
1: A, 2: B, 3: D, 4: C	0.8	10.0%

Table 2.2: Rank IC example - Predicted rankings

From this toy example we can see how predicted rankings with the same Rank IC may yield very different results once the underlying returns are taken into consideration.

Despite this drawback, Rank IC is still a good metric to evaluate the overall quality of a predicted ranking compared to a realized ranking, and when used as a machine learning metric to be maximized, it actually produces fairly similar results when compared to more precise metrics such as BDCG.

2.2.2 BDCG

Discounted Cumulative Gain (DCG) is a measure of ranking quality introduced in 2002 by Järvelin and Kekäläinen in their paper *Cumulated Gain-Based Evaluation of IR Techniques*. [11]

DCG accumulated at ranking position p is defined as:

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{\mathrm{rel}_i}{\log_2(i+1)} \tag{2.4}$$

where rel_i is the relevance score of the result at position i.

The main historical use of DCG is to evaluate the performance of search engines, in the field of information retrieval. For this reason, it progressively discounts the relevance factors in one direction.

A variant of DCG is Bi-directional DCG (BDCG).

The complete formulation of this metric is non-disclosable, but it is clear enough to say that BDCG, instead of focusing only on the top k results, focuses on the top $k/2$ and bottom $k/2$ results.

BDCG is an ideal performance metric because we want to be creating a ranking that's generally as accurate as possible, while giving more relevance to the top $k/2$ and bottom $k/2$ results, which are the most important ones for our use-case.

2.2.3 Risk-adjusted metrics: Sharpe ratio

The Sharpe ratio is a widely used risk-adjusted financial metric. It measures the return of an investment relative to its risk. In the original definition, it is given by:

$$\text{Sharpe} = \frac{R - R_{rf}}{\sigma_R} \quad [12] \tag{2.5}$$

where R is the return of the investment, R_{rf} is the return of a risk-free investment - such as government bonds -, and σ_R is the standard deviation of R, calculated over a specific frequency, for example: the standard deviation of daily returns.

In the setting of this study, we have already defined performance metrics that are better suited to our specific goals, so the Sharpe ratio can be simply expressed as:

$$\text{Sharpe} = \frac{\mu(\text{metric})}{\sigma(\text{metric})} \tag{2.6}$$

where μ is the mean of the time-series of daily performance metric values, and σ is its standard deviation.

Maximizing the Sharpe ratio of one of our performance metrics means that we're not only favouring high performance, as simply maximizing the mean or overall cumulative performance would do, but we're also requesting that said performance is consistent, stable, and as a consequence, as risk-free as possible.

As mentioned above, the Sharpe ratio is one of the most widely used tools in the financial world, even though some studies have highlighted a few potential limitations. [13, 14, 15] These works discuss especially the impact - if any - that relying on the Sharpe ratio - rather than alternative tools - has on performance evaluation.

2.3 Backtesting

In the context of investing, a backtest is the process of simulating an investment strategy using historical data, to evaluate how it would have performed if it had been deployed in the past.

For the purposes of this thesis, a single, basic kind of backtest was performed for each tested pool of signals.

Backtest results are the most reliable measure of performance, because a backtest is not merely a performance indicator, but a full simulation that takes into consideration real-world trading conditions, such as transaction costs. So, for example, a backtest simulation can capture the advantage of an investment strategy that has lower turnover, which is something that a simple metric cannot do.

Standard metrics presented in 2.2 are still useful and necessary to evaluate performance during the training and ensemble selection phases, and generally "good" metrics will still show similar results when compared to backtest results.

3. Problem framework

3.1 Datasets

In order to generate its signals, Axyon AI uses datasets made up of various kinds of features, which can range from purely financial, economic features, to hand-crafted market sentiment features and more.

The labels of these datasets, instead, are precise manipulations of the time-series of the stocks' prices.

Each dataset focuses on a specific set of financial instruments. The one that was employed for this study, Japan Target Market, as the name suggests, focuses on the Japanese equity market. The dataset is based on the Morningstar Japan Target Market Exposure index, which measures the performance of large-cap and mid-cap stocks in Japan, and covers the top 85% of the market by capitalization.

Some preliminary steps were also conducted on a synthetic dataset, containing fake but realistic data, created so that the properties of each feature could be controlled artificially. For example there could be perfectly informative features - perfect correlation with the label - or entirely uninformative features.

The employed dataset, Japan Target Market, contains data for 459 Japanese stocks, from 2010 to the present day, with 73 features per stock per day. There are two versions of this dataset that have been employed in this study, differing only in the labels: one with 20-day horizon labels and one with 60-day horizon labels.

NOTE: It's important to note that datasets don't contain entries for days such as weekends or holidays, when markets are closed. This means that, when we talk about a 20-day horizon, this is equivalent to one month. Similarly, a 60-day horizon refers to three months.

3.2 Sliding fold data splitting

3.2.1 Sliding fold data splitting

For the reasons explored in 1.4, the datasets, which have financial time-series as labels, shouldn't be used directly to train models.

Axyon AI's process includes a very specific way of dividing - splitting - and using the data, called sliding fold data splitting.

The main idea behind sliding fold data splitting is to perform model training operations - called runs - on a progressively expanding window of data. In this study, one run corresponds to three months of additional training data.

Axyon AI's process is made up of two main parts:

1. Training and ensemble selection

2. Out-Of-Sample (OOS) evaluation

For each run, the data is divided into training, validation and test data. The models are trained on the training data, while the validation data and the test data are used to perform ensemble selection, as will be detailed in 3.5.

At this point, for each year - 4 runs -, an ensemble is selected. This is done because it simulates exactly what would happen if we were to create an ensemble right now, but at multiple time points in the past - once per year.

Then, in order to test the real performance of the selected ensembles through time, the OOS evaluation is carried out: each ensemble is employed to generate signals for the subsequent year, and those signals are evaluated using the appropriate metrics, described in 2.2. The final ensemble selection is not carried out as it's reserved for deployment purposes.

In the following figure you can see a representation of the two steps of the sliding fold data splitting process, in the case of Japan Target Market with horizon 20 days - hence, one month gaps between training and validation and between validation and test.

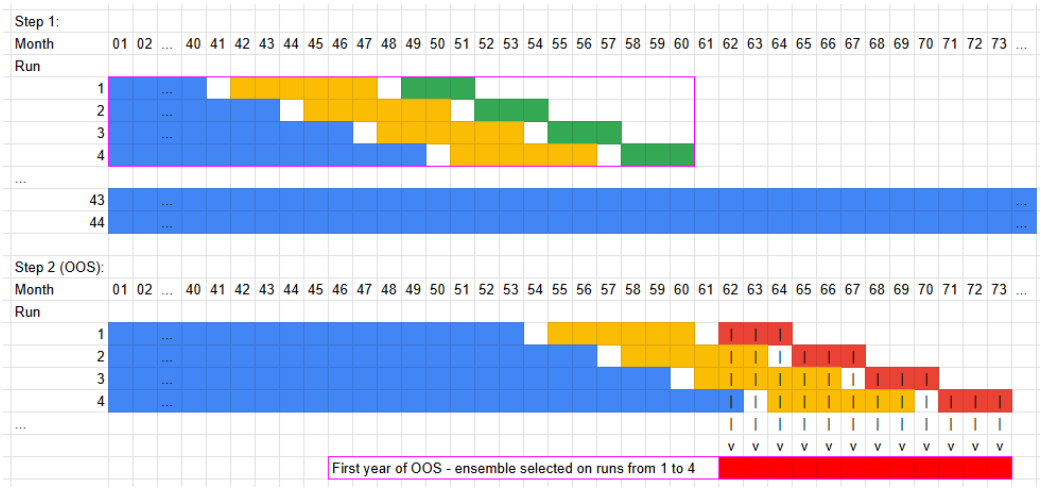

Figure 3.1: Sliding fold data splitting

3.2.2 Split settings

In order to instruct the part of the codebase which is tasked with performing the training of single weak learners to split the dataset in a certain way, respecting the sliding fold data splitting approach, a JSON parameter is passed, called the *json_split_settings*.

This JSON file specifies exactly the number of days to be used in the first run for each split - training, validation and test -, and it also specifies the lengths of the gaps between each split, which are fundamental to avoid look-ahead bias. The splits for subsequent runs are automatically calculated starting from the information contained in the JSON.

The parameters contained in json_split_settings are:

1. Number of training days
2. Training - validation gap length
3. Number of validation days
4. Validation - test gap length
5. Number of test days

3.3 Baseline models

Through a long analysis process and continuous adjustments, Axyon AI has selected a set of baseline models that have proved to provide good and reliable predictions, generally speaking, on all of its datasets.

The full details of the architectures are non-disclosable, but it's possible to present them generally. The baseline models relevant for this study are:

	Name	Type
1	nn cls small	Classification
2	nn cls medium - no regularization	Classification
3	nn cls medium - with regularization	Classification
4	logistic regression cls	Classification
5	nn cls large	Classification
6	nn reg small	Regression
7	gradient boosting reg medium	Regression
8	lasso reg	Regression
9	random forest reg small	Regression
10	linear reg base	Regression
11	nn reg medium - no regularization	Regression
12	nn reg large - with regularization	Regression

Table 3.1: Baseline models

Starting from these baseline models, random variations are generated in a process called random search. These baseline models and their variations compose the pool of models that can then be selected to be part of an ensemble. The full ensemble selection process will be detailed in 3.5.

3.4 Generation and training of candidate models

Before exploring the ensemble selection process in detail, it is useful to briefly describe the preliminary step: the generation and training of candidate models. While this step is a significant part of Axyon AI's overall signal generation process, it was not the main focus of my analysis, as it's not directly related to this study's goals. Nonetheless, understanding and executing it was essential in order to carry out the experimental work.

Starting from the baseline models described in 3.3, through a process called random search, a large number of models - thousands in commercial applications, hundreds in R&D studies such as this work - are automatically generated, by altering hyperparameters and masking input features in multiple ways - performing feature bagging. This generates a fairly diverse set of models, which ideally should be able to learn different patterns from the dataset.

Then, each one of these models has to be trained, respecting the sliding fold data splitting approach detailed in 3.2, so training run by run on the training data, in order to generate test predictions that will be the input of the next step of the process: ensemble selection.

Training hundreds of models is very resource intensive: for this reason, most of the training computational load of this thesis was executed through the European supercomputer Leonardo, hosted in Bologna and managed by Cineca. This was possible thanks to the partnership between Cineca and Axyon AI.

3.5 Ensemble selection process

3.5.1 Ensemble selection

The ensemble selection process has the goal of finding the best possible combination between the models explored - trained - in the random search step, in order to generate an ensemble with high performance and generalization capabilities.

The inputs of the process are the predictions on the validation and test sets of each run that's being taken into consideration, for all of the n explored candidates, which typically are hundreds or even thousands of models.

The output of the process is a list of w models selected, with:

$$\text{min\_ensemble\_size} < w < \text{max\_ensemble\_size} \tag{3.1}$$

3.5.2 Ensemble selection - Greedy algorithm

A single ensemble selection is performed by executing a greedy algorithm, which selects the best performing ensemble at each step, iteratively adding one model at a time. This algorithm is not guaranteed to find the overall optimum, but it is guaranteed to find a local optimum.

The greedy ensemble selection algorithm can early stop when the overall metric computed on the validation data decreases for *early_stopping_patience* consecutive iterations, in order to significantly reduce its computational time.

Algorithm 3.1 Greedy Ensemble Selection Algorithm

Require: Pool C made up of n candidate models, a given evaluation metric (such as Rank IC sharpe, BDCG sharpe...), min_ens_size, max_ens_size, $early\_stopping\_patience$

Ensure: Selected ensemble E_{sel}

$k \leftarrow 0$
Initialize empty ensemble $E_k = E_0 \leftarrow \{\}$
Remove from C candidates with worse-than-random performance
repeat
    for each model m in C **do**
        $E_{tmp} \leftarrow E_k \cup \{m\}$
        Compute and save the metric on the validation data for E_{tmp}
    end for
    Pick the ensemble E^*_{tmp}, and consequently the model m^*, that maximizes the metric on the validation data
    $E_k \leftarrow E_k \cup \{m^*\}$
    $C \leftarrow C \setminus \{m^*\}$
    $k \leftarrow k + 1$
until $k = max\_ens\_size + 1$ **or** $C = \{\}$ **or** early stopping criteria met
for each $k >= min\_ens\_size$ and consequently ensemble E_k **do**
    Compute and save the metric on the test data for E_k
end for
Pick the ensemble E^*_k that maximizes the metric on the test data
$E_{sel} \leftarrow E^*_k$

NOTE: Please note how computing the given metric means performing an evaluation of the quality of each day's rankings.

As we can see from the pseudo-code, the algorithm maximizes the metric on the validation data, but performs the actual selection on the test data. This is done in order to mitigate overfitting.

The greedy algorithm is employed because exploring all of the possible combinations between hundreds of candidates is computationally unfeasible, as it's a problem

with exponentially growing complexity.

For n candidates, the possible ensembles are $2^n - 1$. The greedy algorithm explores a much lower number of configurations - in the order of $n \times k$, where k is the maximum tested ensemble size. Despite this, when n is large, the process is still computationally heavy, and that's the reason why it can be early stopped.

3.5.3 Ensemble selection for Japan Target Market

Let's examine the case of our dataset Japan Target Market for a 20-day horizon: because one ensemble per year has to be selected, this means that 10 distinct ensemble selections have to be carried out in order to get a full Out-Of-Sample evaluation of the results:

		Year
1	From run 1 to run 4	2015
2	From run 1 to run 8	2016
3	From run 1 to run 12	2017
4	From run 1 to run 16	2018
5	From run 1 to run 20	2019
6	From run 1 to run 24	2020
7	From run 1 to run 28	2021
8	From run 1 to run 32	2022
9	From run 1 to run 36	2023
10	From run 1 to run 40	2024

Table 3.2: Ensemble selections to be performed for Japan Target Market

Each one of these selections, in the standard case, is performed on n=500 candidates, and takes into consideration a growing test time period, as can be seen from the following illustration:

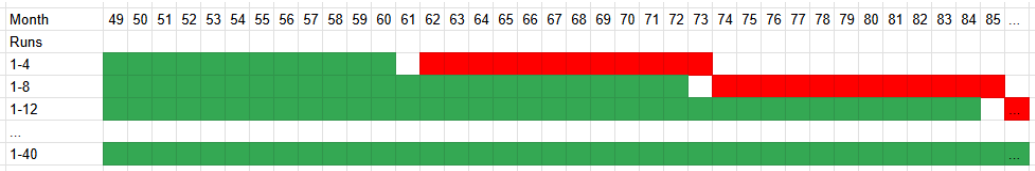

Figure 3.2: Ensemble selection

For the 10 selections, the naive approach would require evaluating a total of

$$10 \times (2^n - 1) = 10 \times \left(2^{500} - 1\right) \tag{3.2}$$

ensembles, which is a number with 152 digits.

Instead, if we fix $max\_ensemble\_size = k = 30$, and execute the greedy algorithm, the total evaluated ensembles are in the order of

$$10 \times n \times k = 10 \times 500 \times 30 = 150{,}000 \tag{3.3}$$

3.6 Standard aggregation process

The standard aggregation of weak learners is performed by simply averaging the most recent prediction of each weak learner. In this setting, each model contributes equally to the ensemble prediction, meaning the average is non-weighted.

There could be a number of possible sophistications - as a start, some form of weighted average - that might yield good results if applied, but these are not employed for a few reasons: firstly, all of those that were tested did not show significant performance improvement. Another possible issue is that, as pointed out in 1.3.6, weak learner de-correlation is key to have a good ensemble, and weighting - especially if dynamic - could heavily affect it.

The standard aggregation process, also called Last aggregation, is represented in figure 4.1.

4. Single-horizon selection and aggregation

4.1 Introduction

The first of the two main goals of this thesis was to test alternative selection and aggregation strategies.

In order to conduct a more complete evaluation of the effects of the alternative selection and aggregation strategies that will be presented in section 4.3, they were tested on two different horizons: 20 days and 60 days.

For each considered horizon, the baseline was one of Axyon AI's ensembles that is trained and produces predictions for the set of Japanese stocks contained in the dataset Japan Target Market, already presented in section 3.1.

In the baseline, the ensemble selection process is performed normally, so considering for every day and for every model only the last available signal, and there is no particular aggregation method, so again, only the last available signal for each model is used. The baseline aggregation method, also called Last aggregation, is represented in figure 4.1.

4.2 Hypothesis

The hypothesis that led to performing the experimental work presented in this chapter, was that creating and implementing more sophisticated selection and aggregation methods could have improved the Out-Of-Sample performance.

4.3 Experiments

This section presents the experiments carried out to obtain the results shown in this chapter.

Using BDCG Sharpe as the metric to maximize, which is the standard in Axyon AI's process, I tested two alternative approaches:

1. Selection on BDCG Sharpe Last - Average aggregation (LA AVG)
2. Selection on BDCG Sharpe Average - Average aggregation (AVG AVG)

The first approach leaves the selection process unchanged and focuses on aggregating differently the available signals. In particular, instead of only considering the last available signal for each model in the ensemble, it averages the last h available signals, with h being the horizon. So, for example, if the horizon is 20, today's signal will be the mean of the last 20 days' signals. These 20 signals will be referred to as "active signals". To rephrase, a signal is considered active when it still has at least some theoretical predictive power, meaning that its horizon period is not fully in the past yet.

Let's consider this toy example where the selected ensemble is composed of 3 models:

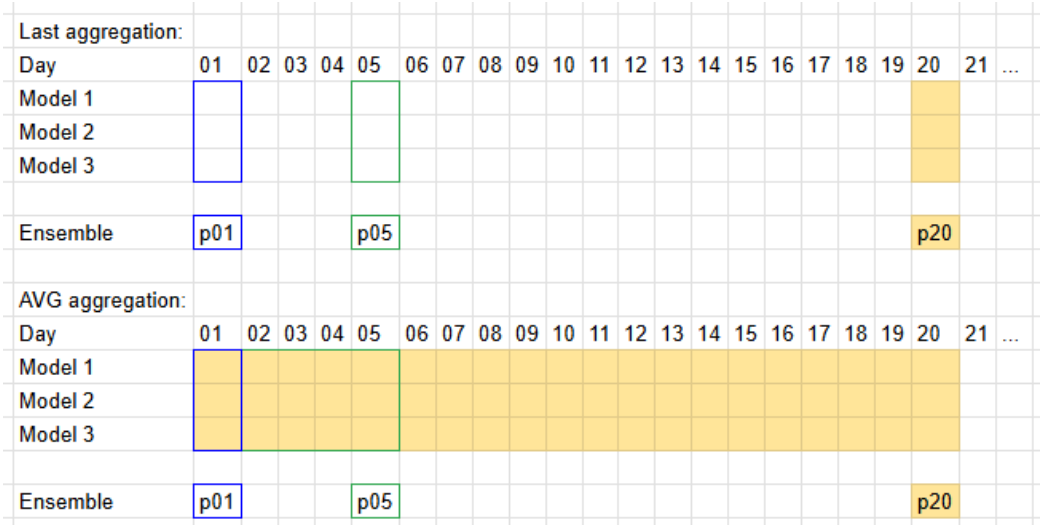

Figure 4.1: Aggregation methods

As we can see, in the baseline case of Last aggregation, the final prediction of the day p_{ii} always considers - by averaging them - a total of 3 weak learner predictions. The Average aggregation method, instead, considers a variable number of weak learner predictions: from 3 on the first day to $20 \times 3 = 60$ for each day from day 20 onwards. The additional predictions considered on day $t = 20$ are temporally misaligned with respect to the horizon day $t + h = 20 + 20 = 40$. Despite this, they are active signals - for example,

p_{01} is trying to predict day $t + h = 1 + 20 = 21$, which is still in the future when we're located in day $t = 20$ - and as such they are assumed to still hold a somewhat relevant predictive power. This is the reason why a maximum number of daily signals equal to h is considered and averaged.

NOTE: Please note how p_{01} in the case of Last aggregation and p_{01} in the case of Average aggregation, coincide.

Considering that the first approach - LA AVG - introduces a different usage of the signals than what the original ensemble selection process tries to optimize, the second approach was created to minimize the mismatch between the optimization objective and the real usage of the signals. For this reason, the same kind of averaging was introduced in the ranking evaluation step of the ensemble selection process.

NOTE: Regarding this chapter, once the Average selections processes were performed, 3 clones of each selected model were created and trained in order to obtain the Out-Of-Sample results presented. The baselines, instead, had 5 clones each. A clone is a repetition of the same model, with the same parameters, dataset, settings, etc. Clones are generated in order to reduce the variability introduced by random factors in models that exploit randomness - for example, random forests. The effects of creating 3 clones instead of 5 are considered negligible.

4.4 Results - 20d

4.4.1 Performance - RIC

	Mean RIC	Sharpe RIC
Baseline 20d	0.03346	0.23053
Selection on Last - Average aggregation	0.03442	0.24047
Selection on Average - Average aggregation	0.03634	0.24926

Table 4.1: 20d - Aggregation methods - Mean and Sharpe RIC

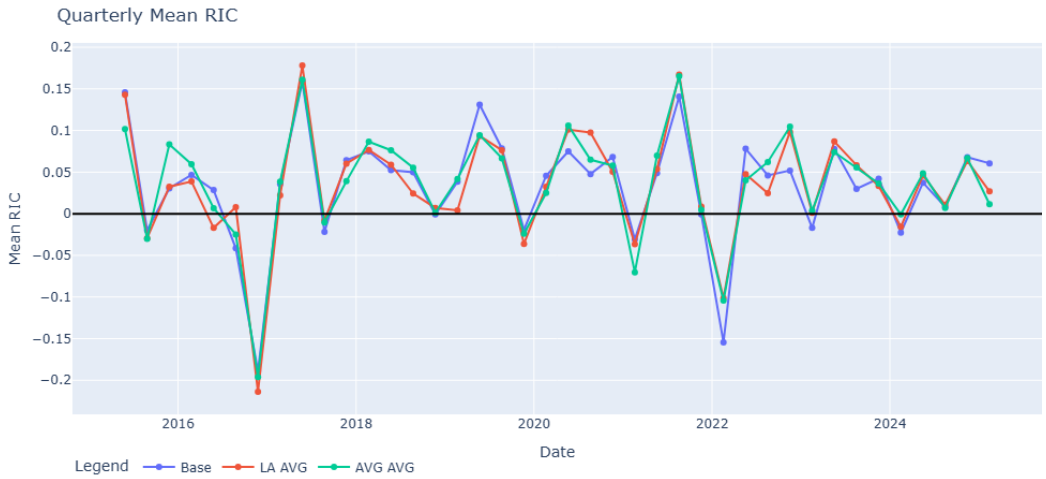

Figure 4.2: 20d - Aggregation methods - Quarterly Mean RIC

4.4.2 Performance - BDCG

	Mean BDCG	Sharpe BDCG
Baseline 20d	0.05391	0.16889
Selection on Last - Average aggregation	0.05632	0.17845
Selection on Average - Average aggregation	0.06147	0.19069

Table 4.2: 20d - Aggregation methods - Mean and Sharpe BDCG

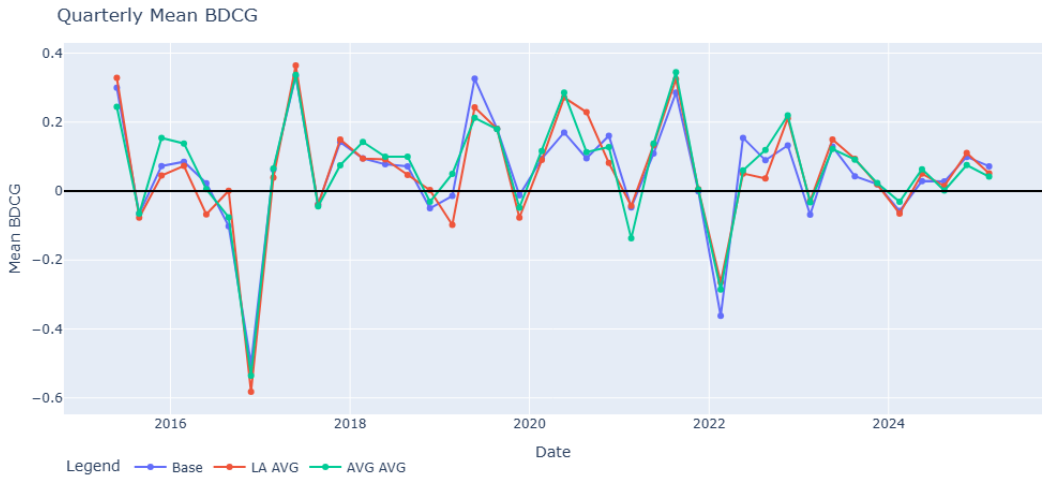

Figure 4.3: 20d - Aggregation methods - Quarterly Mean BDCG

4.4.3 Performance - Backtest

The "Sharpe" column in the table refers to the Sharpe ratio of daily returns.

	Cumulative return	Sharpe
Baseline 20d	40.54%	0.61
Selection on Last - Average aggregation	41.73%	0.68
Selection on Average - Average aggregation	40.56%	0.66

Table 4.3: 20d - Aggregation methods - Backtest

4.4.4 Correlation

	Mean	Maximum	Minimum
Base vs Last - Average	0.88850	1.0	0.54886
Base vs Average - Average	0.84023	0.95453	0.42155

Table 4.4: 20d - Aggregation methods - Correlation

Figure 4.4: 20d - Aggregation methods - Correlation heatmap

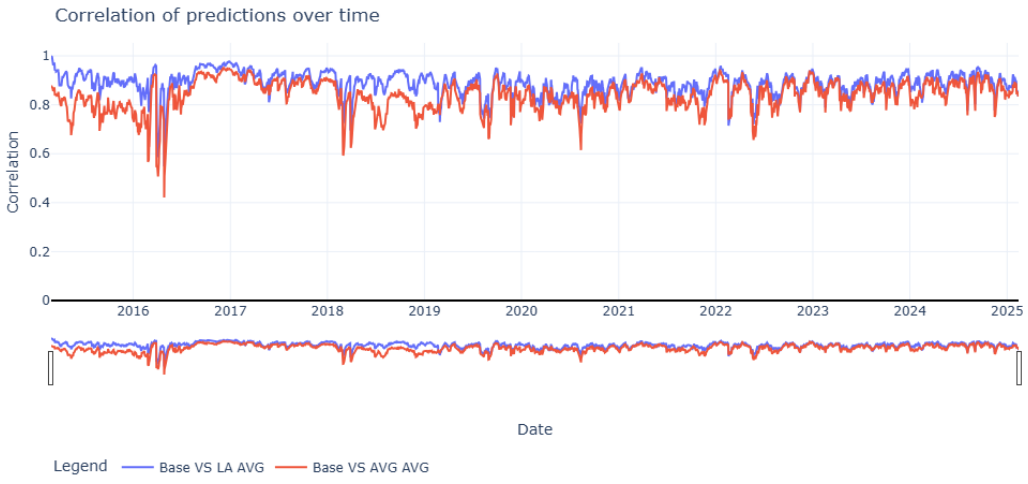

Figure 4.5: 20d - Aggregation methods - Correlation

4.4.5 Selection analysis

Two selections were executed to obtain the results in this section: one standard selection - Base - and one selection on BDCG Sharpe Average - 20 days.

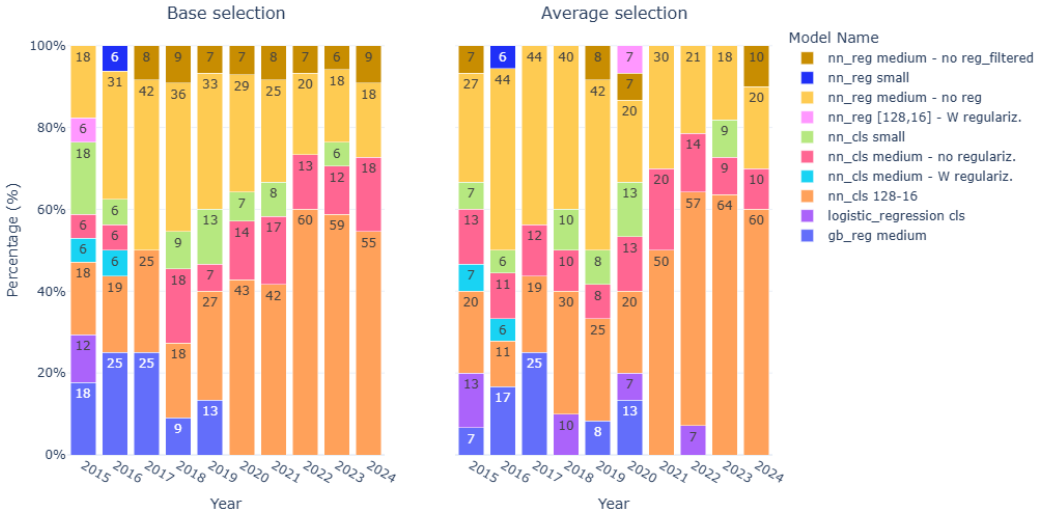

Figure 4.6: 20d - Aggregation methods - Models selected

4.5 Results - 60d

4.5.1 Performance - RIC

	Mean RIC	Sharpe RIC
Baseline 60d	0.04922	0.33758
Selection on Last - Average aggregation	0.04436	0.31697
Selection on Average - Average aggregation	0.04179	0.29376

Table 4.5: 60d - Aggregation methods - Mean and Sharpe RIC

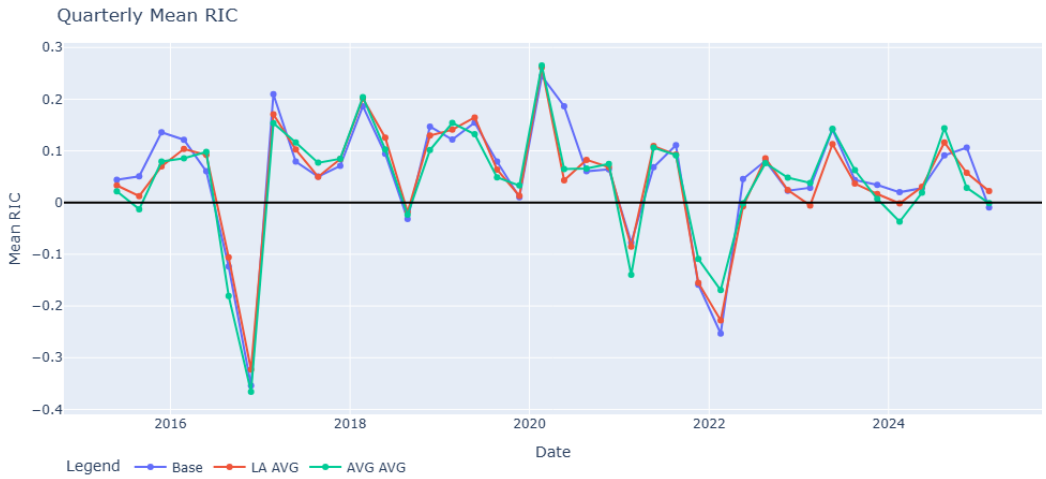

Figure 4.7: 60d - Aggregation methods - Quarterly Mean RIC

4.5.2 Performance - BDCG

	Mean BDCG	Sharpe BDCG
Baseline 60d	0.16855	0.30276
Selection on Last - Average aggregation	0.14101	0.26508
Selection on Average - Average aggregation	0.12155	0.22016

Table 4.6: 60d - Aggregation methods - Mean and Sharpe BDCG

Figure 4.8: 60d - Aggregation methods - Quarterly Mean BDCG

4.5.3 Performance - Backtest

The "Sharpe" column in the table refers to the Sharpe ratio of daily returns.

	Cumulative return	Sharpe
Baseline 60d	56.46%	0.81
Selection on Last - Average aggregation	52.67%	0.80
Selection on Average - Average aggregation	31.64%	0.50

Table 4.7: 60d - Aggregation methods - Backtest

4.5.4 Correlation

	Mean	Maximum	Minimum
Base vs Last - Average	0.87451	1.0	0.50362
Base vs Average - Average	0.84507	0.95298	0.52579

Table 4.8: 60d - Aggregation methods - Correlation

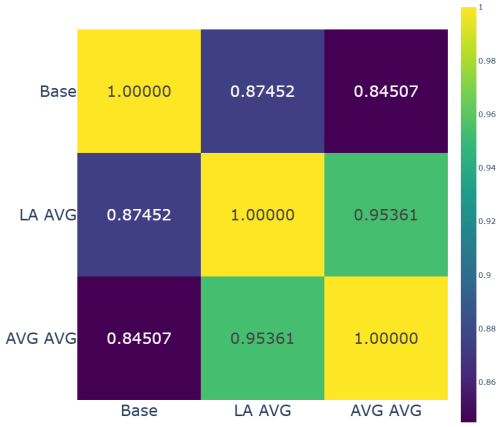

Figure 4.9: 60d - Aggregation methods - Correlation heatmap

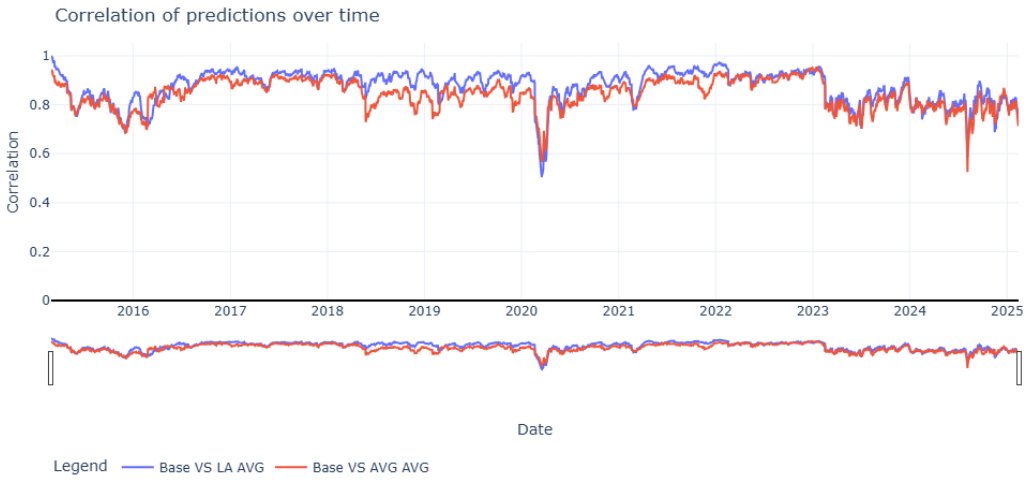

Figure 4.10: 60d - Aggregation methods - Correlation

4.5.5 Selection analysis

Two selections were executed to obtain the results in this section: one standard selection - Base - and one selection on BDCG Sharpe Average - 60 days.

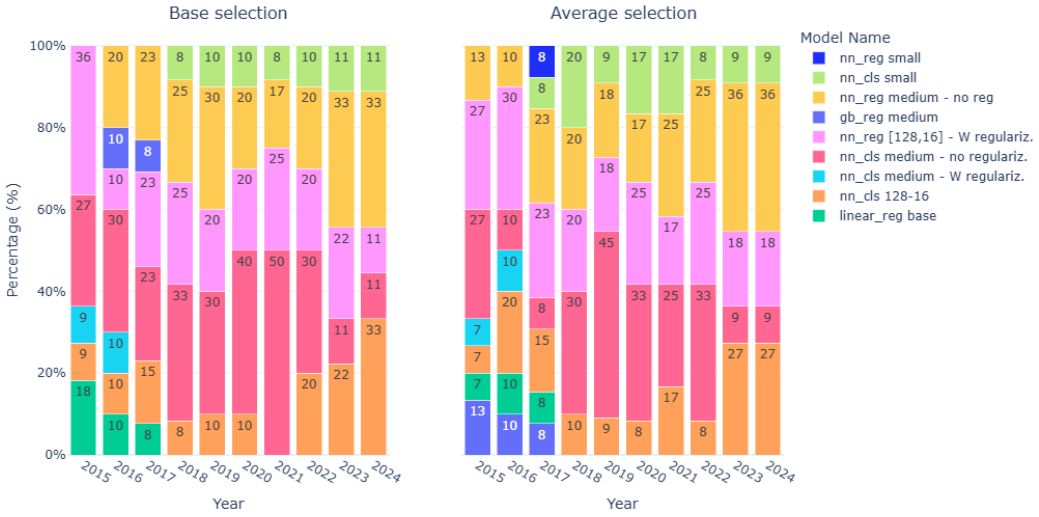

Figure 4.11: 60d - Aggregation methods - Models selected

4.6 Experiments in numbers

The following table details the number of individual models trained, in order to offer an idea of the scale of the experiments carried out and presented in this chapter:

	Baselines	Average selections
Random search models trained	1,000	0
OOS models trained	1,285	759

Table 4.9: Aggregation methods - Models trained

The random search models were trained for 44 runs each. The OOS models, instead, were trained for 4 runs each. The Average selections did not require additional random search models to be trained, as they were performed on the original 1,000 random search models.

The 759 OOS models trained for Average selections are 3 clones each per 253 individual models.

The following table, instead, presents the number of individual ensemble selections performed in order to complete the experiments carried out and presented in this chapter:

		n
Baselines	20	500
Selections on Last - Average aggregation	0	
Selections on Average - Average aggregation	20	500

Table 4.10: Aggregation methods - Ensemble selections performed

where n is the number of candidates for each selection.

No additional ensemble selections were required for Selections on Last - Average aggregations, as the needed selections coincide with those of the baselines.

5. KDE Area

5.1 Introduction

The study of the general theoretical framework of the problem and of the results of the experiments on aggregation methods, led me to theorize a few possible improvements in various parts of the process, additionally to those strictly connected to the thesis motivation and goals.

One of these possible improvements, called KDE Area, was deemed theoretically sound by me and my company supervisors, and yielded good results in preliminary small-scale testing. For these reasons, it was decided to add a large scale test of KDE Area to this study's work.

5.2 Hypothesis and KDE Area definition

The idea for KDE Area is to be a metric that can substitute Sharpe, which has a few documented drawbacks, as discussed in 2.2.3.

The Sharpe ratio, as defined for the purposes of this study, uses two parameters of a discrete distribution, mean μ and standard deviation σ, to model the underlying "true" distribution. Essentially, the Sharpe ratio is modeling the distribution of the daily series of a performance metric - whether that's Rank IC or BDCG - as a single Gaussian. Then, the Sharpe ratio is maximized through the greedy ensemble selection algorithm presented in 3.5.2.

Maximizing the Sharpe ratio is equivalent to maximizing the discrete integral of the Gaussian distribution:

$$\int_0^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx \tag{5.1}$$

which is the area under the Gaussian curve from 0 to $+\infty$.

Because our performance metrics - Rank IC and BDCG - are centered in 0, and because the Gaussian is a probability distribution function, and therefore:

$$\int_{-\infty}^{+\infty} \mathcal{N}(x)\, dx = 1 \tag{5.2}$$

maximizing the 5.1 definite integral means that we're maximizing the approximated cumulative positive returns.

KDE Area, instead, models that distribution using a Kernel Density Estimation - KDE - and in particular, it's modeling it as a sum of n Gaussians, where n is the number of data points - or, in our case, also days - taken into consideration.

In practice, the estimation is performed using SciPy's *gaussian_kde*() function [16], so the kernel size or bandwidth h, which determines how "wide" each Gaussian is, is automatically determined using Scott's Rule, which sets h as:

$$h = n^{-1/(d+4)} \tag{5.3}$$

where n is the number of data points and d is the dimensionality of the data (here $d = 1$, because we're considering a series).

Then, as in the case of Sharpe, the discrete integral of the distribution is computed and maximized:

$$\text{KDE Area} = \int_{0}^{+\infty} \text{gaussian\_kde}(x)\, dx \tag{5.4}$$

which is the area under the estimated curve from 0 to $+\infty$.

Finally, just like before, because the curve generated by *gaussian_kde*() is still a probability distribution function, we have:

$$\int_{-\infty}^{+\infty} \text{gaussian\_kde}(x)\, dx = 1 \tag{5.5}$$

and this means that by maximizing the 5.4 definite integral, we're also maximizing the approximated cumulative positive returns, but in this case the approximation should be more accurate.

5.3 KDE Area example

To illustrate the idea behind KDE Area, let's consider an example.

I generated two random but realistic series that represent daily metric value series for a period of 169 days. The two series were built in a way as to have the same mean μ, the same standard deviation σ and consequently also the same Sharpe ratio, but different KDE Area.

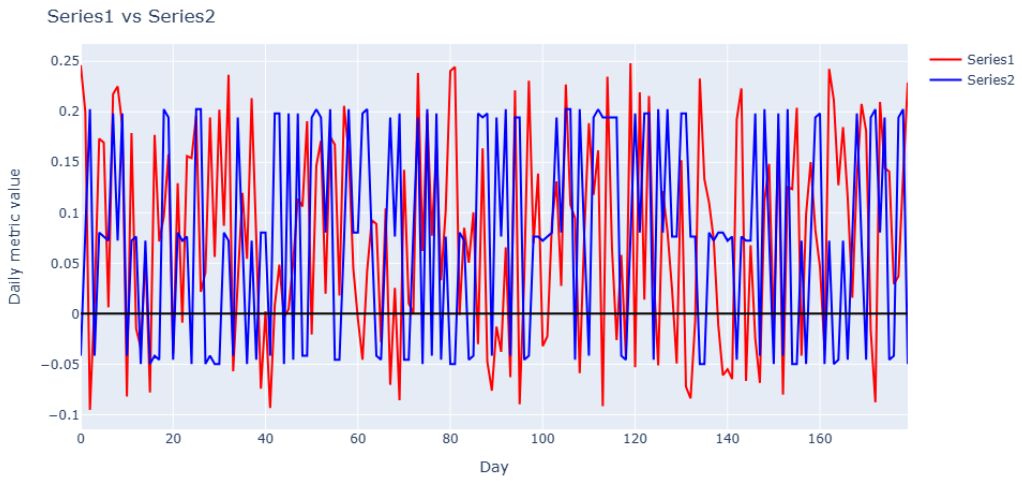

Figure 5.1: KDE Area example

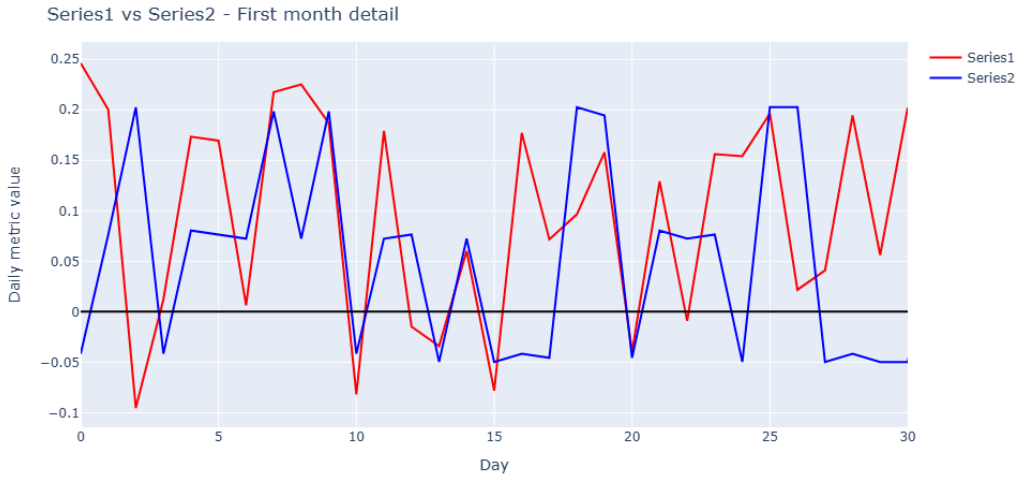

Figure 5.2: KDE Area example - First month detail

	Mean μ	Standard deviation σ	Sharpe	KDE Area
Series1	0.0764	0.1	0.764	0.72105
Series2	0.0764	0.1	0.764	0.69455

Table 5.1: KDE Area example

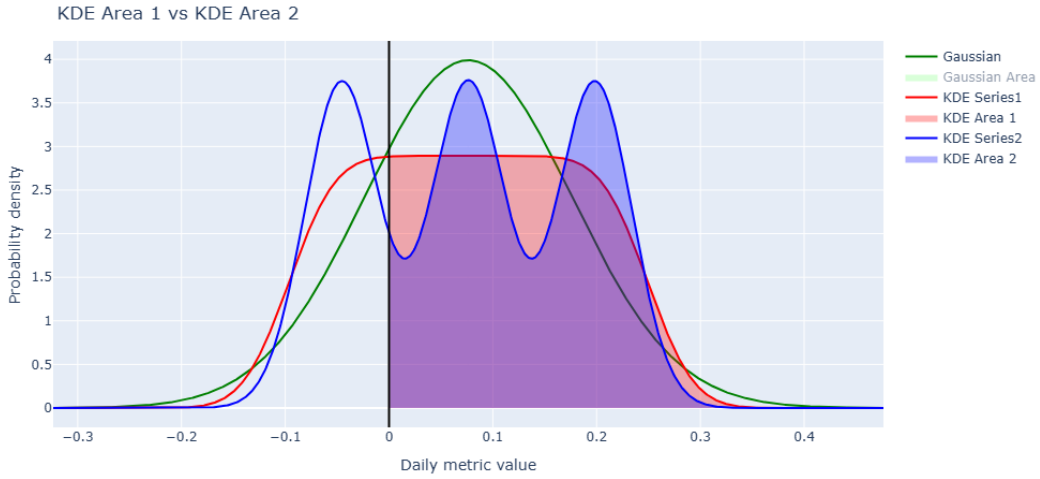

Figure 5.3: KDE Area example - KDE Areas comparison

According to the Sharpe ratio, Series1 and Series2 are equal, and both are represented by the same Gaussian - in green in Figure 5.3 - with $\mu = 0.0764$ and $\sigma = 0.1$.

When we approximate the two series using Kernel Density Estimation, instead, we get two different curves that better represent the underlying distribution of the two series values.

NOTE: Please note that an equal Sharpe ratio can be achieved even if $\mu_1 \neq \mu_2$ and $\sigma_1 \neq \sigma_2$, for example if we have $a = (\mu_1 = 0.1, \sigma_1 = 0.1)$ and $b = (\mu_2 = 0.2, \sigma_2 = 0.2)$, both have $Sharpe(a) = Sharpe(b) = 1$. In this case, a and b would be represented by two Gaussians that are graphically different, but with the same result for the definite integral of Formula 5.1.

5.4 Experiments

Taking into consideration the results of the previous chapter, and the fact that performing a full selection, generation of OOS signals and their evaluation is a process that takes up a lot of resources, both in terms of time and monetary cost, it was decided to perform only one selection on KDE Area Last, as it was deemed that selecting on Average doesn't produce any performance improvement.

So, the following sections present the results, both for a 20-day horizon and a 60-day horizon, of two experiments:

1. Selection on BDCG KDE Area Last - Last aggregation
2. Selection on BDCG KDE Area Last - Average aggregation

The results for these two experimental procedures are compared with their respective 20-day and 60-day baselines.

NOTE: Regarding this chapter, once the KDE Area selections processes were performed, 3 clones of each selected model were created and trained in order to obtain the Out-Of-Sample results presented. The baselines, instead, had 5 clones each. The definition of clone was provided in 4.3.

5.5 Results - 20d

5.5.1 Performance - RIC

	Mean RIC	Sharpe RIC	KDE Area RIC
Baseline 20d	0.03346	0.23053	0.61738
KDE Area Last - Last	0.03580	0.24154	0.63062
KDE Area Last - Average	0.03532	0.24340	0.63126

Table 5.2: 20d - KDE Area - Mean and Sharpe RIC

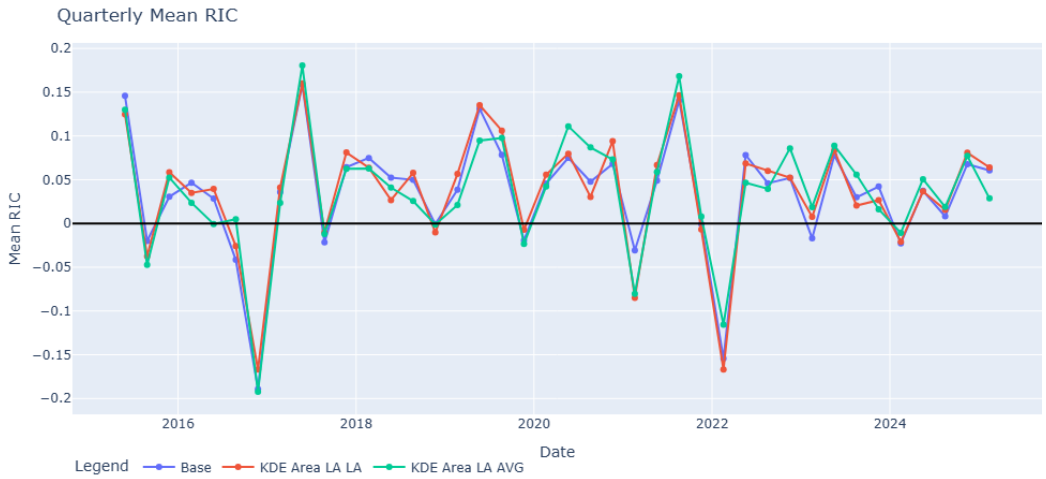

Figure 5.4: 20d - KDE Area - Quarterly Mean RIC

5.5.2 Performance - BDCG

	Mean BDCG	Sharpe BDCG	KDE Area BDCG
Baseline 20d	0.05391	0.16889	0.60340
KDE Area Last - Last	0.05775	0.17690	0.61012
KDE Area Last - Average	0.05391	0.16899	0.60655

Table 5.3: 20d - KDE Area - Mean and Sharpe BDCG

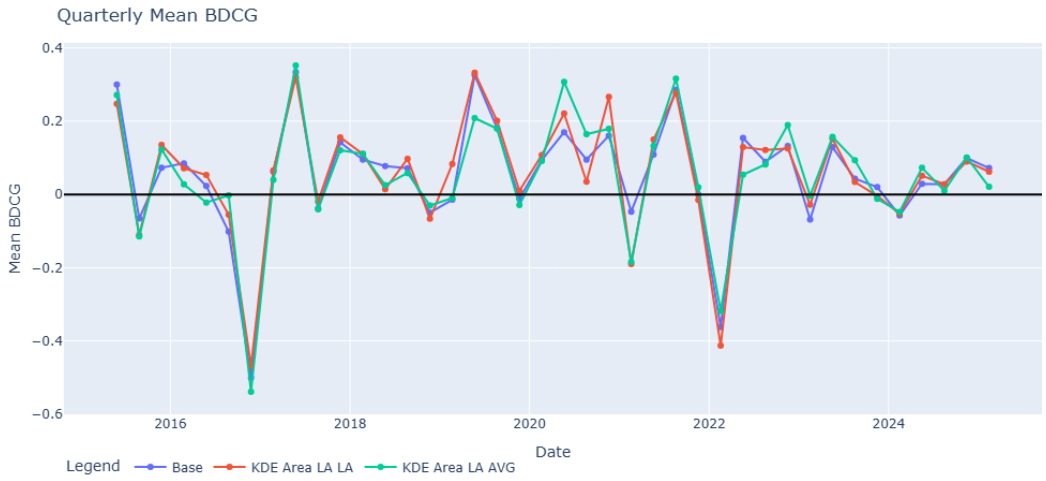

Figure 5.5: 20d - KDE Area - Quarterly Mean BDCG

5.5.3 Performance - Backtest

The "Sharpe" column in the table refers to the Sharpe ratio of daily returns.

	Cumulative return	Sharpe
Baseline 20d	40.54%	0.61
KDE Area Last - Last	46.72%	0.68
KDE Area Last - Average	35.18%	0.59

Table 5.4: 20d - KDE Area - Backtest

5.5.4 Correlation

	Mean	Maximum	Minimum
Base vs KDE Area Last - Last	0.93875	0.98343	0.82981
Base vs KDE Area Last - Average	0.84641	0.94818	0.50372

Table 5.5: 20d - KDE Area - Correlation

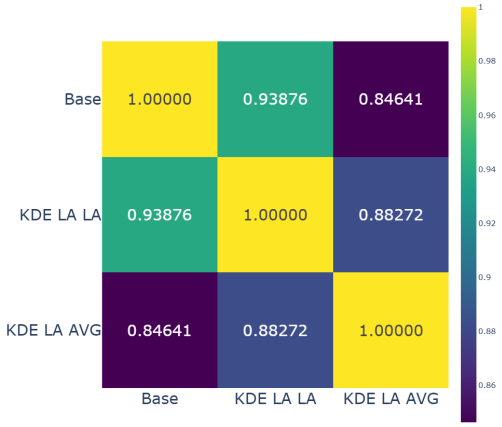

Figure 5.6: 20d - KDE Area - Correlation heatmap

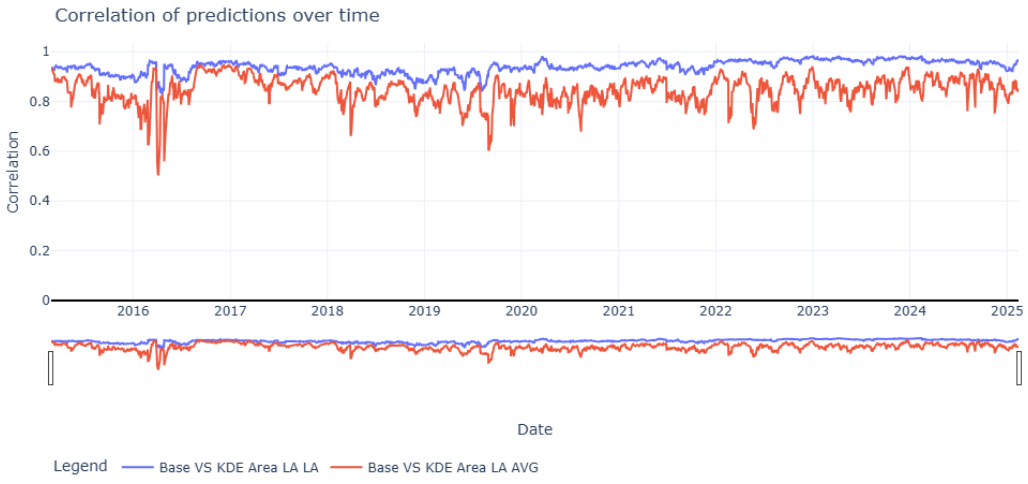

Figure 5.7: 20d - KDE Area - Correlation

5.5.5 Selection analysis

In addition to the baseline, already presented in Chapter 4, one more selection was executed to obtain the results in this section: on BDCG KDE Area Last - 20 days.

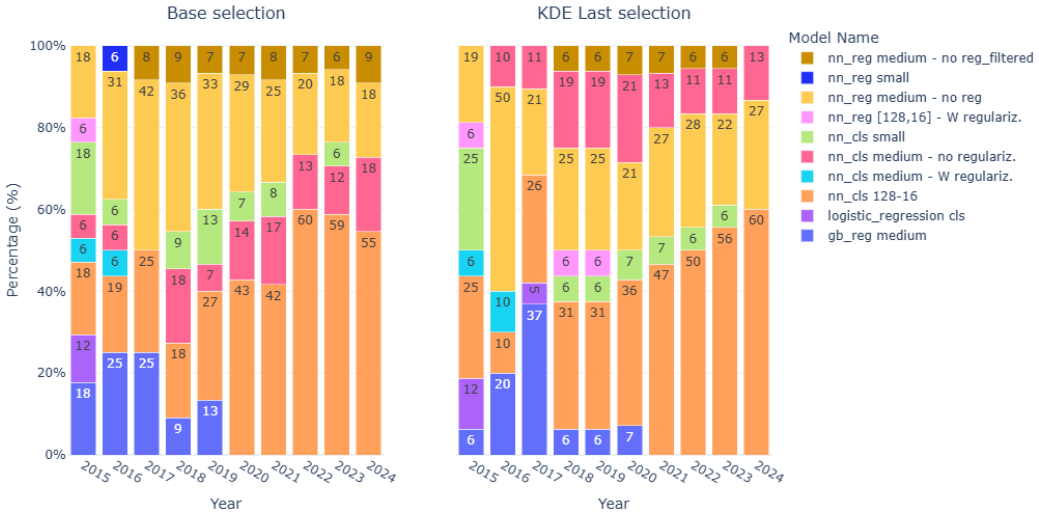

Figure 5.8: 20d - KDE Area - Models selected

5.6 Results - 60d

5.6.1 Performance - RIC

	Mean RIC	Sharpe RIC	KDE Area RIC
Baseline 60d	0.04922	0.33758	0.68641
KDE Area Last - Last	0.04705	0.32542	0.65881
KDE Area Last - Average	0.04243	0.30469	0.64118

Table 5.6: 60d - KDE Area - Mean and Sharpe RIC

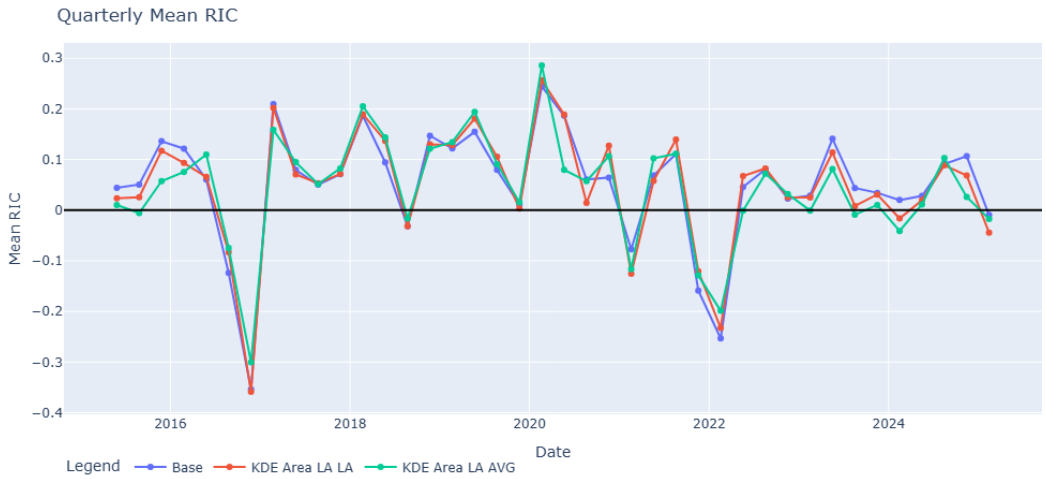

Figure 5.9: 60d - KDE Area - Quarterly Mean RIC

5.6.2 Performance - BDCG

	Mean BDCG	Sharpe BDCG	KDE Area BDCG
Baseline 60d	0.16855	0.30276	0.68683
KDE Area Last - Last	0.16100	0.28790	0.66105
KDE Area Last - Average	0.13816	0.25662	0.64403

Table 5.7: 60d - KDE Area - Mean and Sharpe BDCG

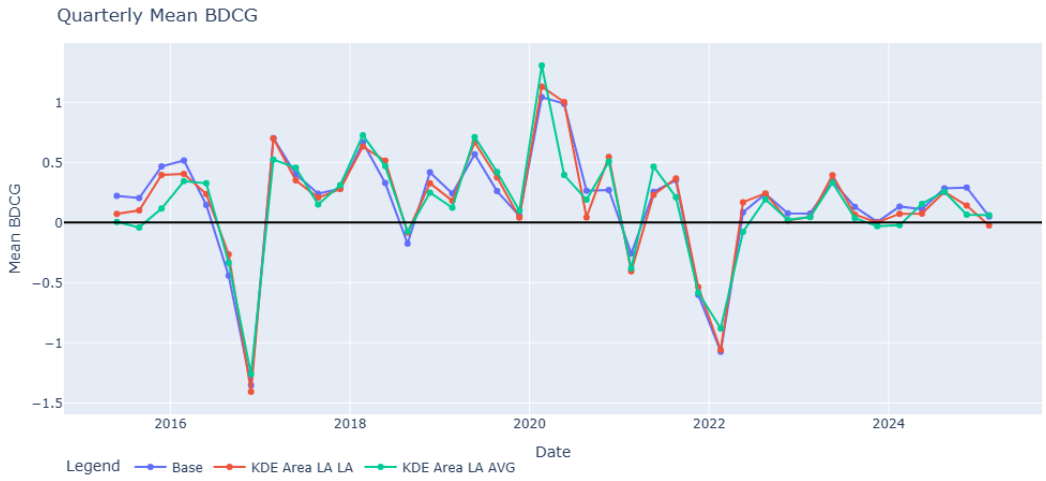

Figure 5.10: 60d - KDE Area - Quarterly Mean BDCG

5.6.3 Performance - Backtest

The "Sharpe" column in the table refers to the Sharpe ratio of daily returns.

	Cumulative return	Sharpe
Baseline 60d	56.46%	0.81
KDE Area Last - Last	53.95%	0.78
KDE Area Last - Average	36.47%	0.59

Table 5.8: 60d - KDE Area - Backtest

5.6.4 Correlation

	Mean	Maximum	Minimum
Base vs KDE Area Last - Last	0.92865	0.98653	0.78214
Base vs KDE Area Last - Average	0.83419	0.94926	0.53286

Table 5.9: 60d - KDE Area - Correlation

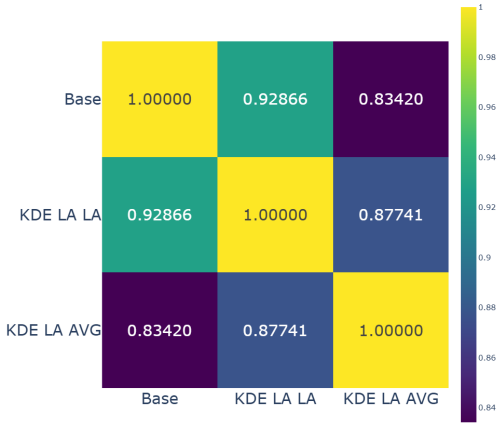

Figure 5.11: 60d - KDE Area - Correlation heatmap

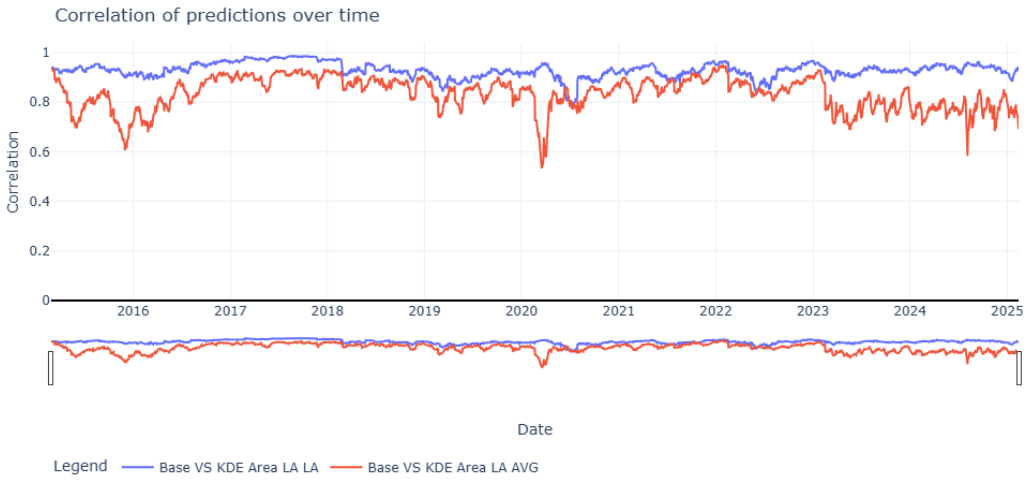

Figure 5.12: 60d - KDE Area - Correlation

5.6.5 Selection analysis

In addition to the baseline, already presented in Chapter 4, one more selection was executed to obtain the results in this section: on BDCG KDE Area Last - 60 days.

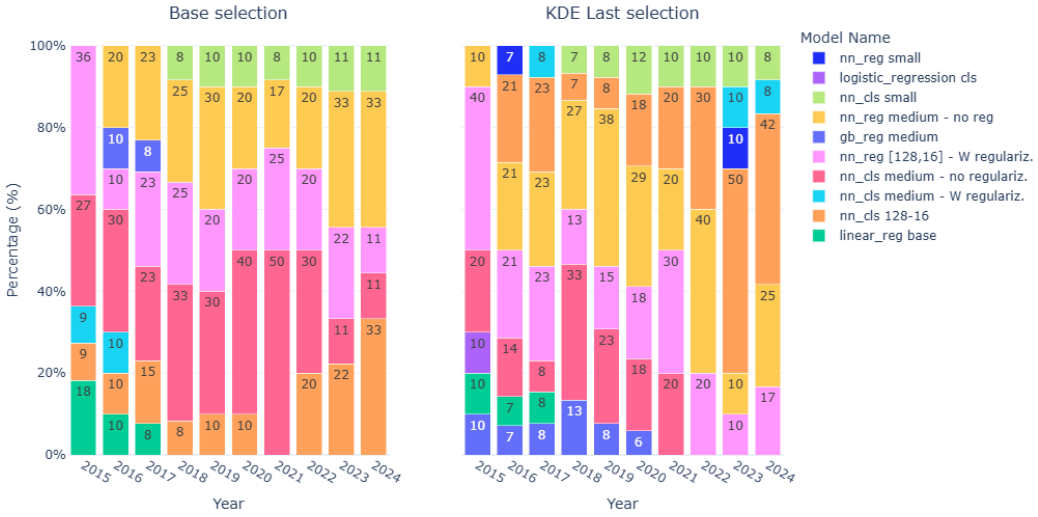

Figure 5.13: 60d - KDE Area - Models selected

5.7 Experiments in numbers

Please note that the baselines coincide with those already presented in Chapter 4.

The following table details the number of individual models trained, in order to offer an idea of the scale of the experiments carried out in this chapter:

	Baselines	KDE Area Last selections
Random search models trained	1,000	0
OOS models trained	1,285	855

Table 5.10: KDE Area - Models trained

The random search models were trained for 44 runs each. The OOS models, instead, were trained for 4 runs each.

The 855 OOS models trained for KDE Area Last selections are 3 clones each per 285 individual models.

The following table, instead, presents the number of individual ensemble selections performed in order to complete the experiments carried out and presented in this chapter:

		n
Baselines	20	500
KDE Area Last - Last	20	500
KDE Area Last - Average	0	

Table 5.11: KDE Area - Ensemble selections performed

where n is the number of candidates for each selection.

No additional ensemble selections were required for KDE Area Last - Average, as the needed selections coincide with those of KDE Area Last - Last.

6. Horizon Union

6.1 Introduction

The second main goal of this study was to verify whether performing "Horizon Union", so allowing the ensemble selection process, when trying to predict a specific horizon, to use models that are trained to predict other horizons, would improve performance.

6.2 Hypothesis

The idea to unite horizons derives from previous R&D studies performed by Axyon AI. During these studies it was observed that, generally, adding heterogeneity and different informational patterns in the signals used by the ensembles would improve performance.

Therefore, it was hypothesized that somehow integrating the informative power of signals trying to predict longer horizons into the shorter horizons would lead to performance improvements.

This idea could have been implemented in multiple different ways, but it was specifically requested to me to integrate it inside the selection process.

6.3 Experiments

To obtain the results presented in this chapter, the Horizon Union between models targetizing $h = 20$ and models targetizing $h = 60$ was performed. In other words, I united in a single weak learner pool, models trying to predict what will happen in 20 days with models trying to predict what will happen in 60 days.

Given the fairly inconclusive results obtained in the previous chapter, which

had opposite results for the two tested horizons, it was decided to stick to BDCG Sharpe as the metric to use and optimize.

In order to perform Horizon Union, two main problems had to be solved:

1. The temporal splitting of the data performed by the sliding fold data splitting method presented in 3.2 is different for different horizons. This leads to a temporal misalignment between datasets split according to a 20-day horizon and datasets split according to a 60-day horizon.

2. Once Horizon Union is performed, the resulting signals are no longer strictly related to a specific horizon, but they can be broadly considered as "valid" between the shortest horizon and the longest horizon. This fact poses the issue of how to evaluate the overall quality of these signals, both during the training phase and the evaluation phase.

The first issue was solved by aligning the temporal splits of all the datasets involved on the basis of the longest horizon targetized, $h = 60$. In particular, this was achieved by setting the $json\_split\_settings$ - presented in 3.2.2 - of all models to the standard values requested by the longest horizon.

The second issue, instead, was solved by:

1. Leaving the labels, used during the training phase, as they were. This means that each model is still trying to learn with respect to its original horizon.

2. Setting the so called "target deltas", which are the stock returns used to compute BDCG in the ensemble selection phase, to those of the shortest horizon, $h = 20$, for all models, regardless of if they're 20-day models or 60-day models.

3. Evaluating the Horizon Union signals with respect to the shortest horizon considered, $h = 20$, and the corresponding baseline ensemble, Baseline 20d.

	$json\_split\_settings$	Labels	Target deltas
Baseline 20d	20-day	20-day	20-day
Horizon Union	60-day	20-day and 60-day	20-day

Table 6.1: Horizon Union settings

In short, these settings allow the ensemble selection process that is trying to pick the optimal ensemble for a 20-day horizon, to pick both models that were trained on a 20-day horizon and on a 60-day horizon, which is exactly what we wanted.

NOTE: Regarding this chapter, once the selection process was performed, 5 clones of each selected model were created and trained in order to obtain the Out-Of-Sample results presented. The baseline also had 5 clones each. The definition of clone was provided in 4.3.

6.4 Results

6.4.1 Performance - RIC

	Mean RIC	Sharpe RIC
Baseline 20d	0.03346	0.23053
Horizon Union Last - Last	0.03673	0.22951
Horizon Union Last - Average	0.03481	0.21953

Table 6.2: Horizon Union - Mean and Sharpe RIC

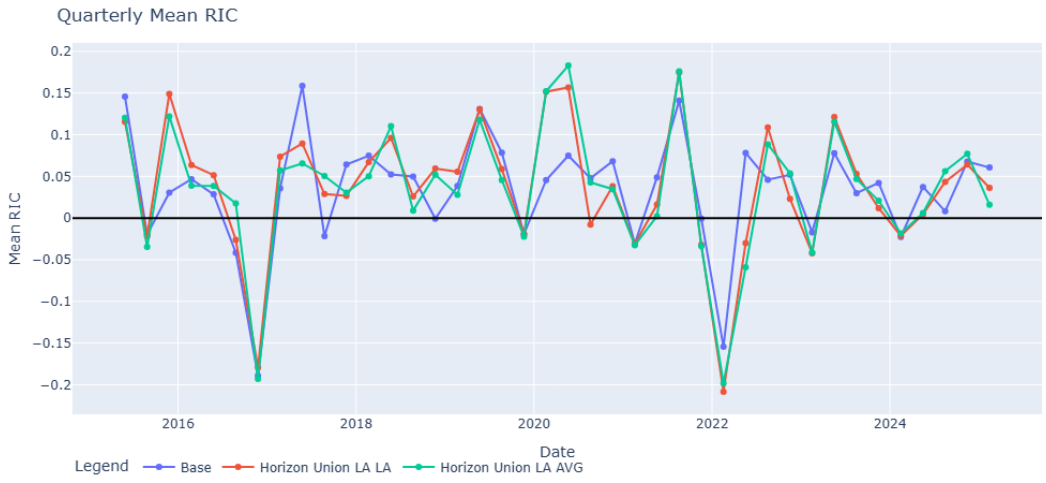

Figure 6.1: Horizon Union - Quarterly Mean RIC

6.4.2 Performance - BDCG

	Mean BDCG	Sharpe BDCG
Baseline 20d	0.05391	0.16889
Horizon Union Last - Last	0.06928	0.19627
Horizon Union Last - Average	0.06850	0.19191

Table 6.3: Horizon Union - Mean and Sharpe BDCG

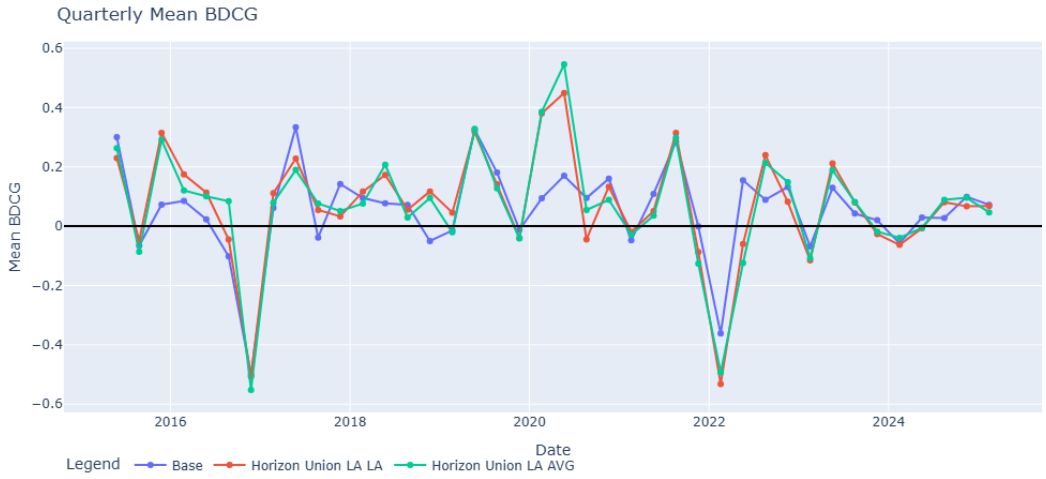

Figure 6.2: Horizon Union - Quarterly Mean BDCG

6.4.3 Performance - Backtest

The "Sharpe" column in the table refers to the Sharpe ratio of daily returns.

	Cumulative return	Sharpe
Baseline 20d	40.54%	0.61
Horizon Union Last - Last	53.71%	0.78
Horizon Union Last - Average	49.68%	0.74

Table 6.4: Horizon Union - Backtest

6.4.4 Correlation

	Mean	Maximum	Minimum
Base vs Horizon Union Last - Last	0.81911	0.93878	0.54187
Base vs Horizon Union Last - Average	0.74594	0.92547	0.26742

Table 6.5: Horizon Union - Correlation

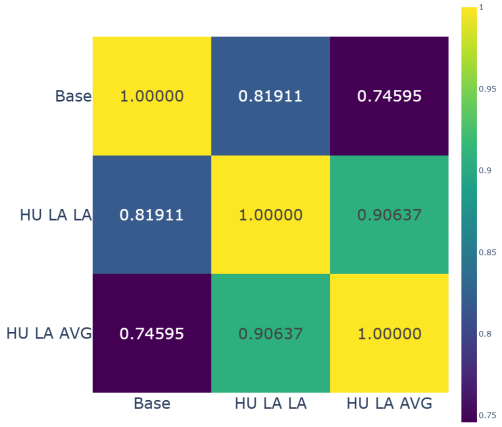

Figure 6.3: Horizon Union - Correlation heatmap

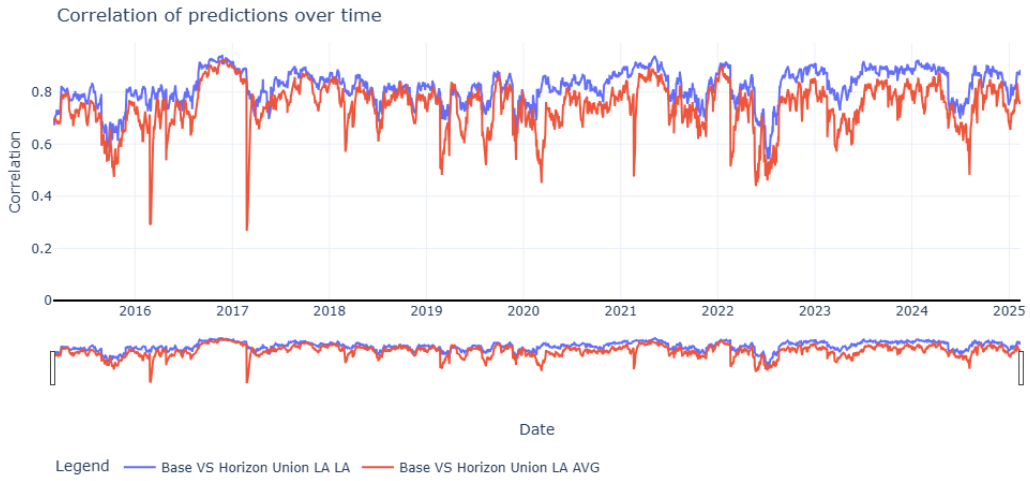

Figure 6.4: Horizon Union - Correlation

6.4.5 Selection analysis

In addition to the baseline, already presented in Chapter 4, one more selection was executed to obtain the results in this chapter: on BDCG Sharpe Horizon Union Last.

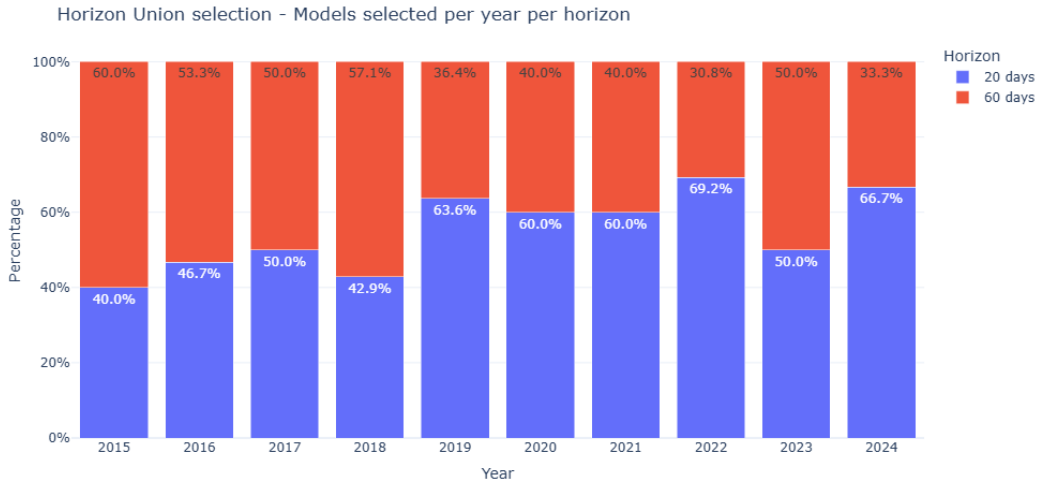

Figure 6.5: Horizon Union selection - Models selected per horizon

It's interesting to note how there's a certain stability in the share of 20-day

and 60-day models selected. It's also important that the selection process doesn't collapse into selecting models predicting only one of the two horizons, but the split is almost even.

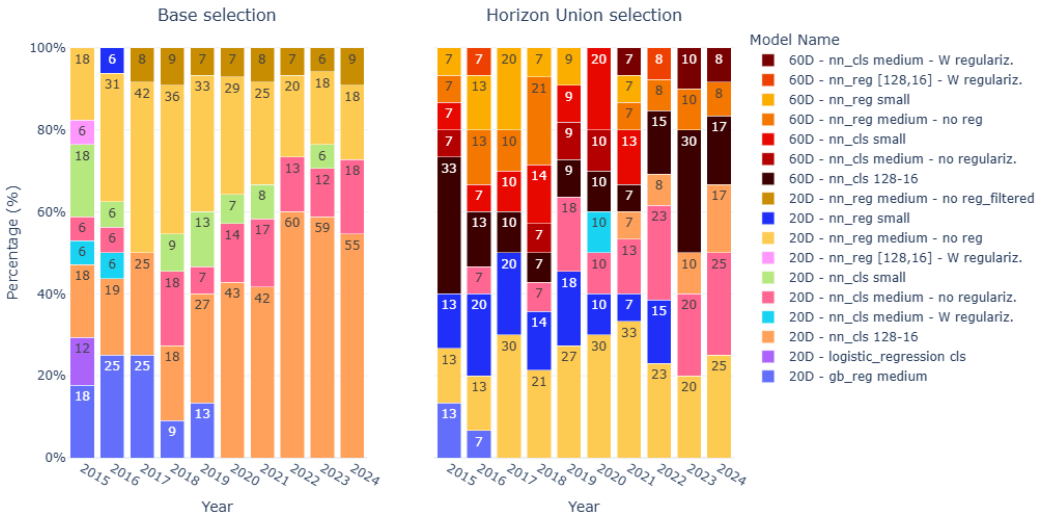

Figure 6.6: Horizon Union - Models selected

6.5 Experiments in numbers

Please note that the baseline is one of the two already presented in Chapter 4.

The following table details the number of individual models trained, to offer an idea of the scale of the experiments carried out in this chapter:

	Baseline	Horizon Union selection
Random search models trained	500	200
OOS models trained	725	625

Table 6.6: Horizon Union - Models trained

The random search models were trained for 44 runs each. The OOS models, instead, were trained for 4 runs each.

The 625 OOS models trained for the Horizon Union selection are 5 clones each per 125 individual models.

This chapter required to train additional, very computationally expensive, random search models. It was decided to perform a random search with 200 total individuals instead of the 500 of the baseline. Despite this, the Horizon Union method overperformed the baseline, as shown in the previous section.

The following table presents the number of individual ensemble selections performed in order to complete the experiments carried out and presented in this chapter:

		n
Baseline	10	500
Horizon Union Last - Last	10	200
Horizon Union Last - Average	0	

Table 6.7: Horizon Union - Ensemble selections performed

where n is the number of candidates for each selection.

No additional ensemble selections were required for Horizon Union Last - Average, as the needed selections coincide with those of Horizon Union Last - Last.

7. Conclusions

7.1 Conclusions

This thesis project, fully developed at Axyon AI, explored in a hands-on way the study of ensemble learning techniques applied to the field of financial forecasting.

The work presented includes a comprehensive examination of the Axyon AI signal generation process considered, and a set of experiments evaluating alternative ensemble selection methods and metrics, signal aggregation strategies and multi-horizon forecasting experiments.

The experimental work was carried out with a rigorous approach, in order to obtain precise and reliable results for the dataset tested, Japan Target Market.

The experiments conducted yielded variable outcomes: some produced inconclusive results; others, however, worked well and provided clear evidence of the potential benefits of combining models across multiple horizons, offering an idea of the potentialities of multi-horizon financial forecasting, in the setting of Axyon AI.

When considering the experiments performed with an investment horizon of $h = 20$, the contribution that proved to have superior performance has been the Horizon Union Last - Last method presented in 6.4.3. The beforementioned Horizon Union method, selected an almost even split of 20-day and 60-day models. This shows that, as initially hypothesized, there is performance to be extracted from uniting horizons.

The issues related to the Horizon Union problem, then, were addressed successfully, and the results of the backtest of the Horizon Union method proved that, as we can see from the following graphical comparison:

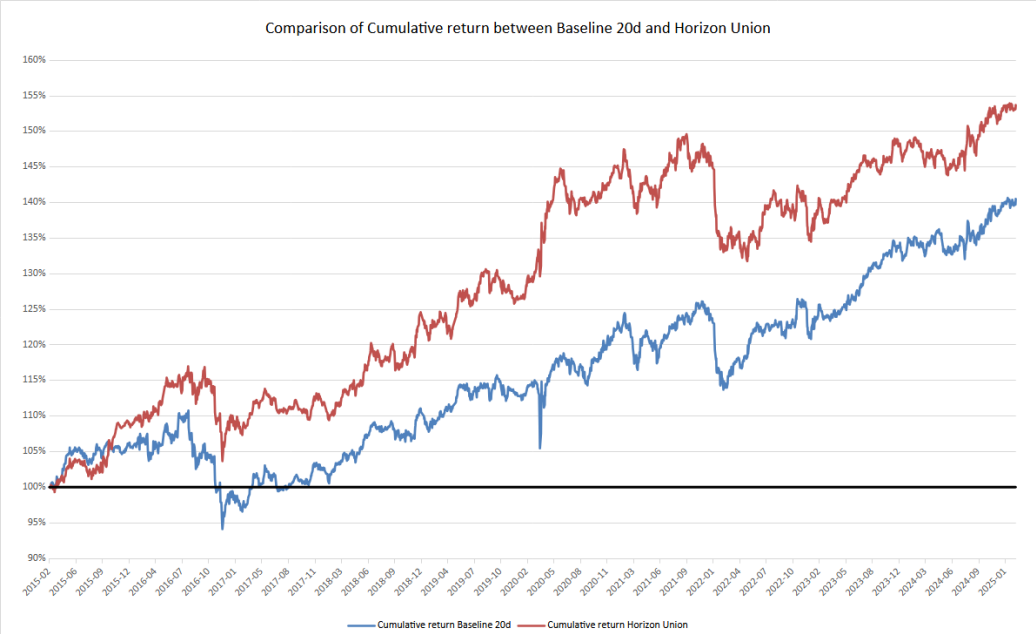

Figure 7.1: Comparison of Cumulative return between Baseline 20d and Horizon Union

7.2 Limitations and Future Work

The main limitation of this study is that the experimental evaluation was conducted on a single real-world dataset, although the experiments did consider multiple investment horizons.

As a result, the findings cannot be assumed to generalize to datasets with different characteristics, particularly those featuring a larger number of financial instruments.

A natural extension of this work would be to expand the evaluations performed to other datasets, especially focusing on the methods that demonstrated the most promising results in this study.

References

[1] M. Mohri, A. Rostamizadeh, and A. Talwalkar, "*Foundations of Machine Learning*", MIT Press, 2012, pp. 7–8.

[2] S. Gu, B. Kelly, and D. Xiu, "*Empirical Asset Pricing via Machine Learning*", Review of Financial Studies, vol. 33, no. 5, 2020, pp. 2223–2273.

[3] T.-Y. Liu, "*Learning to Rank for Information Retrieval*", Foundations and Trends in Information Retrieval, vol. 3, no. 3, 2009, pp. 225–331.

[4] C. Krauss, X. A. Do, and N. Huck, "*Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500*", European Journal of Operational Research, vol. 259, no. 2, 2017, pp. 689–702.

[5] L. Breiman, "*Bagging Predictors*", Machine Learning, vol. 24, no. 2, 1996, pp. 123–140.

[6] Y. Freund and R. E. Schapire, "*Experiments with a New Boosting Algorithm*", Proceedings of the 13th International Conference on Machine Learning (ICML), 1996, pp. 148–156.

[7] D. H. Wolpert, "*Stacked Generalization*", Neural Networks, vol. 5, no. 2, 1992, pp. 241–259.

[8] D. Wood, T. Mu, A. M. Webb, H. W. J. Reeve, M. Luján, and G. Brown, "*A Unified Theory of Diversity in Ensemble Learning*", Journal of Machine Learning Research, vol. 24, 2023, pp. 1–49.

[9] B. A. Marconi, "*Time Series Foundation Models for Multivariate Financial Time Series Forecasting*", Imperial College London, 2025, Chapter 1.

[10] S. Gupta and A. Gupta, "*Dealing with Noise Problem in Machine Learning Data-sets: A Systematic Review*", Procedia Computer Science, vol. 161, 2019, pp. 466–474.

[11] K. Järvelin and J. Kekäläinen, "*Cumulated Gain-Based Evaluation of IR Techniques*", ACM Transactions on Information Systems, vol. 20, no. 4, 2002, pp. 422–446.

[12] W. F. Sharpe, "*The Sharpe Ratio*", The Journal of Portfolio Management, Fall 1994.

[13] A. W. Lo, "*The Statistics of Sharpe Ratios*", Financial Analysts Journal, vol. 58, no. 4, 2002, pp. 36–52.

[14] M. Eling, "*Does the Measure Matter in the Mutual Fund Industry?*", Financial Analysts Journal, vol. 64, no. 3, 2008, pp. 42–54.

[15] J. R. H. Ornelas, A. F. Silva Júnior, and J. L. B. Fernandes, "*Yes, the choice of performance measure does matter for ranking of US mutual funds*", International Journal of Finance & Economics, vol. 17, no. 1, 2012, pp. 61–72.

[16] SciPy, "Documentation for *scipy.stats.gaussian_kde*", URL:
`https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html`

Acknowledgements

I would like to thank Prof. Carlo Augusto Grazia for accepting the role of academic supervisor and for his important advice before and during the preparation of this thesis. I also would like to thank my company supervisor Alessandro Lambertini and the whole team at Axyon AI for their help, support and for creating a welcoming and wonderful work environment.

To my dad Flaviano, my mum Veronica and my brothers Andrea and Marco: thank you for always being by my side every step of the way. None of this would have been possible without you.

To my grandparents Bepi and Grazia: thank you for helping me in so many ways, especially during difficult times, and thank you for all of your advice.

To my grandmother Mary: thank you for teaching me to always treat others as we wish to be treated, along with many other values I try to live by daily.

To my grandfather Giuseppe: thank you and rest in peace grandpa, I would have loved to have you here today. I'm sure we would have had a lot of fun.

To the entirety of my extended family: thank you for all your help and the invaluable joy you bring me at every single family reunion.

To Odessa: thank you for being my partner in crime, I'm immensely grateful to have you standing by my side.

To Ari: thank you. For the past 10 years, whether you were hundreds of kilometers away, or right next to me, I could always count on you and your friendship.

To Davide and Morel: thank you for being awesome people and friends. I feel extremely lucky to have met you. Casa Crescenzago forever.

To Paglie, Iris, Michele, Turro, Alice, Massimo, Pav, Miglia, Ale, Andre, Michi, Pier and everybody else who walked even just one step by my side on the long and twisty path that brought me here: simply, from the bottom of my heart, thank you.

Ringraziamenti

Vorrei ringraziare il Prof. Carlo Augusto Grazia per aver accettato il ruolo di relatore e per i suoi fondamentali consigli prima e durante la stesura di questa tesi. Vorrei anche ringraziare il mio supervisore aziendale Alessandro Lambertini e l'intero team di Axyon AI per il loro aiuto, supporto e per aver creato un ambiente di lavoro accogliente e fantastico.

A mio papà Flaviano, mia mamma Veronica e i miei fratelli Andrea e Marco: grazie per essere stati sempre al mio fianco in ogni passo di questo percorso. Nulla di tutto ciò sarebbe stato possibile senza di voi.

Ai miei nonni Bepi e Grazia: grazie per avermi aiutato in tantissimi modi, soprattutto nei momenti difficili, e grazie per tutti i vostri consigli.

A mia nonna Mary: grazie per avermi insegnato che è giusto trattare gli altri come vogliamo essere trattati, oltre a tantissimi altri valori fondamentali che cerco di mettere in pratica tutti i giorni.

A mio nonno Giuseppe: grazie e riposa in pace nonno, mi sarebbe piaciuto averti qui oggi. Sono certo che ci saremmo divertiti un sacco.

A tutta la mia famiglia estesa: grazie per tutto l'aiuto e per la gioia preziosa che mi regalate in ogni occasione in cui ci troviamo tutti assieme.

A Odessa: grazie per essere la mia partner in crime. Sono immensamente grato di averti al mio fianco.

Ad Ari: grazie. Negli ultimi 10 anni, che tu fossi a centinaia di kilometri o accanto a me, ho sempre potuto contare su di te e sulla tua amicizia.

A Davide e Morel: grazie per essere persone fantastiche ed amici fantastici. È una fortuna conoscervi. Casa Crescenzago forever.

A Paglie, Iris, Michele, Turro, Alice, Massimo, Pav, Miglia, Ale, Andre, Michi, Pier e chiunque altro abbia camminato anche solo un passo a fianco a me sulla lunga e tortuosa strada che mi ha portato fino a qui: semplicemente, dal profondo del mio cuore, grazie.