Mid-Curve Recommendation System: a Stacking Approach Through Neural Networks

Adriano Koshiyama, Nick Firoozye and Philip Treleaven
Department of Computer Science
University College London
Gower Street, London WC1E 6BT
Email: [adriano.koshiyama.15, n.firoozye, p.treleaven]@ucl.ac.uk

***Abstract*—Derivative traders are usually required to scan through hundreds, even thousands of possible trades on a daily-basis; a concrete case is the so-called Mid-Curve Calendar Spread (MCCS). The actual procedure in place is full of pitfalls and a more systematic approach where more information at hand is crossed and aggregated to find good trading picks can be highly useful and undoubtedly increase the trader's productivity. Therefore, in this work we propose an MCCS Recommendation System based on a stacking approach through Neural Networks. In order to suggest that such approach is methodologically and computationally feasible, we used a list of 15 different types of US Dollar MCCSs regarding expiration, forward and swap tenure. For each MCCS, we used 10 years of historical data ranging weekly from Sep/06 to Sep/16. Then, we started the modelling stage by: (i) fitting the base learners using as the input sensitivity metrics linked with the MCCS at time t, and its subsequent annualized returns as the output; (ii) feeding the prediction from each base model to a particular stacker; and (iii) making predictions and comparing different modelling methodologies by a set of performance metrics and benchmarks. After establishing a backtesting engine and setting performance metrics, our results suggest that our proposed Neural Network stacker compared favourably to other combination procedures.**

I. Introduction

Inside an investment bank, a quantitative research team plays an important role as a hub for the demands and inputs of different stakeholders: traders, structurers, bankers, salespeople, etc. In general these actors are interested on trading ideas, portfolio analysis, economic outlooks, and new products to be sold for the bank's clients: pension funds, wealth managers, commercial banks, that is, institutional investors in general. A concrete case is the so-called Mid-Curve Calendar Spread (MCCS), a derivatives package that involves selling an option on a forward-starting swap and buying an option on a spot-starting swap with longer expiration [5], [22]. In such a package, traders usually look for the historical carry and the breakeven width levels, in order to rank the most prominent ones to offer a client or to proceed in some proprietary trading.

However, one might notice that the main downsides of such approach are: (i) substantial information on the underlying are usually not taken into account; (ii) using the previous example, high historical values for carry and breakeven widths are more necessary rather than sufficient conditions for a profitable MCCS trade; (iii) a trader can quickly judge if an individual trade is worthwhile to invest, but may take some time to find it; and (iv) after a given period, traders tends to only look at a small subset of possible trades. Hence, a systematic approach where more information at hand is crossed and aggregated to find good trading picks can be highly useful and undoubtedly increase the trader's productivity.

Therefore, in this work we propose a MCCS Recommendation System based on a stacking approach through Neural Networks. Since we are looking for an omnibus methodology that can work for the different profiles of a MCCS, this ensemble-based methodology [13], [28] can provide us with the capacity to highlight each base model where it fares best and discredit each base model where it performs poorly. By having many distinct base learners, we are able to provide a reasonable solution to different MCCSs, while at the same time give more chance to the Neural Network stacker to outperform the base learners across all MCCSs.

In order to suggest that such approach is methodologically and computationally feasible, we used a list of 15 different types of US Dollar MCCSs regarding expiration, forward and swap tenure. For each MCCS, we used 10 years of historical data ranging weekly from Sep/06 to Sep/16. Then, we started the modelling stage by: (i) fitting the base learners using as the input sensitivity metrics linked with the MCCS at time t, and its subsequent annualized returns as the output (in this case, holding 1-year-ahead the trade); (ii) feeding the prediction from each base model to a particular stacker; and (iii) making predictions and comparing different modelling methodologies by a set of performance metrics and benchmarks. Precisely, we used a total of 13 base models, with some being commonly used by the traders to decide which MCCS look more attractive to a selling pitch. We compared the Neural Network stacker with three other traditional combination approaches common in the financial literature. After establishing a backtesting engine and setting performance metrics, our results suggest that our proposed Neural Network stacker compared favourably to other stacking procedures.

In this sense, we organised this work as follows: next section presents a literature review on existing approaches to return/price prediction/estimation in different areas and instruments, as well as a brief description on MCCS trades. The third section displays the stacking approach through Neural Networks, the base learners used and the calibration process that each undergone, the dataset that comports the MCCS trades and a set of performance metrics applied.

Finally, we exhibit the results and discussions linked with the performance analysis of each model through different metrics and perspectives. We close this work with some concluding remarks and future directions for research.

II. Related Works and Mid-Curve Calendar Spread

Literature provides a growing body of evidence that price changes can be predicted, that is, in particular circumstances and periods securities violate the Efficient Market Hypothesis [3], [20]. This hypothesis states that price changes must be unforecastable if they are properly anticipated, that is if they fully incorporate the expectations and information of all participants. In this sense, researchers have employed different modelling approaches and information sets to predict price changes across a range of assets; for cash instruments (equities, bonds, foreign exchange, etc.) we can find a vast amount of research using statistical and machine learning methods [4], [6], [11], [15], [19], [29], [30].

Contrasting with the emphasis that researchers in cash instruments put on return predictability, when we devote our attention to research in derivatives instruments (options, swaps, swaptions, etc.) it is clear that most of the effort is concentrated on pricing these contracts via stochastic calculus [10], [16], [22], [25], with some few exceptions using Neural Networks and other machine learning models for price estimation [14], [23], [27]. When we devote our attention to the asset type that this work is dedicated, interest rate swaptions, a similar pattern persists: most of the research is related to pricing and not to return prediction.

Based on this short review of existing approaches to return/price prediction/estimation in different areas and instruments, to the best of our knowledge, our work is the first attempt to apply computational statistics and machine learning techniques to build trading strategies in the context of interest rate swaptions. Our approach is not the only novel from a modelling perspective, but instead of trading the vanilla product (receiver/payer interest rate swaption), we prefer to focus on options strategies (calendar spreads, straddles, etc.) which in many cases is the package that is in practice traded. By thinking in terms of the package, in this case, a Mid-Curve Calendar Spread, rather than the individual constituents we unlock some features that can only be computed in this situation, like the carry at expiry, breakeven width and so on.

A Mid-Curve Calendar Spread (MCCS) is a package involving short selling an option on a forward-starting swap and going long a longer-expiry swaption on the same underlying swap [5], [12]. Investors typically use MCCS to take a view on forwarding volatility. This comes from the fact that, conceptually, spot volatility can be decomposed into forward volatility and mid-curve volatility. Taking 10y10y[1] for example, we outline below a relation that hold for those volatility exposures of different time periods for the underlying 10y10y rate [26]:

$$10y \times \sigma^2_{10y10y} = 5y \times \sigma^2_{5y5y10ymid-curve} + 5y \times \sigma^2_{5yfwd5y10y} \tag{1}$$

With the above (assuming flat skews where volatility is the standard deviation), 5y fwd 5y10y volatility can be backed out if we know 10y10y swaption volatility and 5y5y10y mid-curve volatility. Therefore, a forward strike swaption, in nature a pure exposure to forward volatility, can be approximated by short selling a mid-curve and buying a plain vanilla swaption, which is an MCCS. Due to put-call parity, at the expiry date, an MCCS becomes either an out of the money payer or a receiver no matter it is payer or receiver at the inception.

In Figure 1 is presented the payoff profile for an USD 1m1y2y. We plot the payoff profiles for current volatility and up and down volatility scenarios, noting that the long vega position means that the payoff profile shifts up in a rising volatility environment and correspondingly shifts down in a falling vol environment. We calculate the (volatility adjusted) breakevens as being $0.41\% - 0.47\%$, giving very little protection against selloffs, but maintaining a decent margin for a rally. We note that forwards in a ± 1 volatility band leave them at $0.40\% - 0.49\%$, a range just marginally larger than our breakeven range (i.e., the trade should pay off just slightly less than 66% of the time). From this payoff profile many of the input variables, outlined in the next section, are backed, such as the Gamma, Delta, Breakeven Width (BE-Width), and so on.

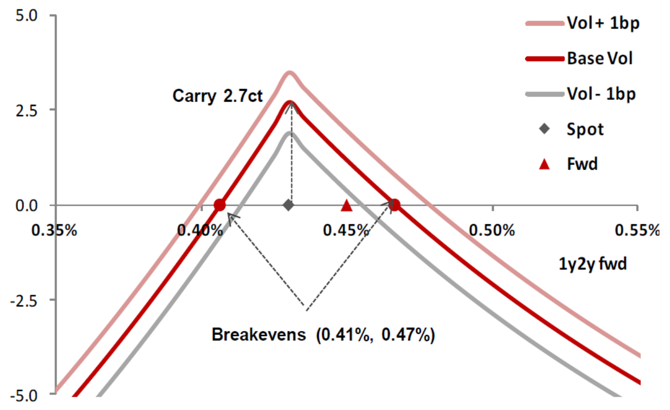

Fig. 1. Payoff profile for an USD 1m1y2y.

III. Methodology

In summary, our solution develops the following roadmap:

1) **Data**: On a certain trade date, we **calculate metrics and sensitivities** related to an MCCS package;
2) **Modelling**: These metrics are **feed in a predictive model** that outputs its expected return for a given holding period (e.g., one year);
3) **Recommendation**: After repeating (i) and (ii) for all MCCS we (iii) rank them based on the expected returns using some criteria.

[1]This notation is extensively used during this work. In this case, the first 10y means a spot swaption with 10 year of expiration, while the second 10y refers to the swap tenure.

Hence, the next three subsections describes in details the MCCS trades and ensemble members used (Dataset and Models section); how these base learners were stacked, and specifically how they were assembled in a Neural Network stacker (Stacking Approach section); and finally, last subsection presents which metrics were used to evaluate the recommendation system performance when a certain predictive model candidate is underpinning it (Performance Metrics).

A. Dataset and Models

During our experiments, we opted to use the trades displayed in Table I.

TABLE I
CONFIGURATION OF THE MCCS TRADES USED. REMEMBER THAT THE PACKAGE INVOLVES SELLING AN OPTION ON A FORWARD-STARTING SWAP (USING THE EXPIRATION AND FORWARD TENURES) AND BUYING A LONG EXPIRATION OPTION (SWAP TENURE) ON A SPOT-STARTING SWAP.

Currency	Expiry	Forward	Swap	Currency	Expiry	Forward	Swap
USD	1y	1y	1y	USD	1y	10y	20y
USD	1y	5y	5y	USD	2y	1y	1y
USD	2y	5y	5y	USD	2y	10y	20y
USD	3y	1y	1y	USD	3y	5y	5y
USD	3y	10y	20y	USD	4y	1y	1y
USD	4y	5y	5y	USD	4y	10y	20y
USD	5y	1y	1y	USD	5y	5y	5y
USD	5y	10y	20y				

Although many other configurations are available in practice, these are the ones with longest historical data available, which is important when it is necessary to fit a predictive model. As it can be seen, all trades are in US Dollar, ranging from different expiries (1y-5y), forwards (1y, 5y and 10y) and swap tenures (1y, 5y and 20y).

For each configuration, at time t we agree with a counterpart to trade this package using the At the Money Forward (ATMF) rate as the strike, paying or receiving the present value PV_t. The PV_t is computed via SABR model [24], using information and parameters (e.g., spot, forward rates and rate-rate correlation) calibrated using market data on a daily basis. From the same model that computed the PV_t, we can obtain other metrics as those displayed in Table II.

TABLE II
METRICS AND SENSITIVITIES COMPUTED FOR EACH MCCS AT TIME t.

Features	
PV	Strike
Carry at Expiry	Breakeven Width (BE-Width)
Aged 1y Carry	Theta
ATMF Implied Volatility (Implied Vol)	Gamma
Vega	Curve Carry (Aged 1y)
Time Carry (Aged 1y)	Volatility Carry (Vol Carry)

Carry at Expiry and BE Width are those obtained looking at the payoff profile at expiry. The Aged 1y Carry is produced by ageing the trade by one year (moving closer to the expiration) and estimate the payoff profile computing the carry. Theta, Vega and Gamma are the sensitivities of the instruments by a change in time, volatility and a wider range of underlying rate movements, respectively. These and the ATMF Implied Vol are backed by the SABR model too. Curve, Time and Volatility Carry are the amount of Aged 1y Carry that can be attributed to the changes in certain sensitivities from spot to forward, such as the Delta (Curve), Theta (Time) and Vega (Volatility). These can also be used as tools to understand which factors most influence the instrument value over time.

After computing all these metrics at time t, we hold the trade until $t+h$ where h can be two weeks, one month, one year, and so on, as long as $t+h$ is before or at expiration. In time $t+h$ we compute the PV_{t+h} of the same trade again, using the new economic scenario available (e.g. rates, change in model parameters). By agreeing on buying back or selling the current trade for PV_{t+h} we can compute the Holding k-period Return of the trade started at time t by:

$$R_t^{(h)} = \frac{PV_{t+h} - PV_t}{PV_t} \tag{2}$$

Based on these procedures, metrics and observations, Table III express other details that we used during our experiments to generate the dataset.

TABLE III
DETAILS USED TO GENERATE THE MCCS TRADE DATASET.

Detail	Value
Period	September 2006 to September 2016
Holding Period (h)	1 year
Trade Frequency	Weekly (usually on Wednesday)
Strike	At the Money Forward (ATMF)
Lagged data ($h-p$)	$p = 1, 2$ and 3 lagged returns
Assumption	Characteristic
Bid-Ask	Middle Rate
Transaction Costs	Entry and Unwind = 0.75 $\times$ $Vega_t$
Funding Rate	3 month Treasury bill

Therefore, we gathered data from trades entered on a weekly basis from September 2006 to September 2016. These trades are struck ATMF, using the PV_t computed from the Middle Rate (in practice, some bid-ask spread would be imbued proportional to the Vega). After holding for one year ($h = 1y$) the trade, we compute the arithmetical returns that are, therefore by definition, automatically annualised. These returns are gross, and so we need to take into account the transaction costs (hedging costs and fixed fees charged by the derivatives desk) as well as some future funding rate. These values are also outlined in Table III, where the transaction costs of 0.75 as a fraction of Vega were chosen not only to taken into account the transaction cost, but also some potential bid-ask spread on the start/unwind of the trade. The 3-month Treasury bill rate was chosen as the funding cost/benchmark rate to compute excess returns.

In relation to modelling, our general model is a system of uncoupled equations:

$$R_{t,1}^{(1y)} = f_1(features_{t,1}) + \varepsilon_{t,1} = \hat{R}_{t,1}^{(1y)} + \varepsilon_{t,1} \tag{3}$$

$$R_{t,2}^{(1y)} = f_2(features_{t,2}) + \varepsilon_{t,2} = \hat{R}_{t,2}^{(1y)} + \varepsilon_{t,2} \tag{4}$$

$$\cdots$$

$$R_{t,n}^{(1y)} = f_n(features_{t,n}) + \varepsilon_{t,n} = \hat{R}_{t,n}^{(1y)} + \varepsilon_{t,n} \tag{5}$$

where for each MCCS trade ($i = 1, ..., n$) there is an i-th predictive model f_i that is feed with a set of pre-calculated features (BE Width, Carry, etc.) and returns an estimate of the holding 1y-period return $\hat{R}_{t,i}^{(1y)}$. As the model is an approximation, some noise/error is expected, and in the modelling aspect, this is expressed as the $\varepsilon_{t,i}$ component. After defining which variable is intended to be predicted, the remaining points are: which models are available to embody f_i and how the fitting, validation and selection of these models are going to be made.

About the first point, in the first rows of Table IV we display the base learners that we used during our experiments. We should mention that these models are standard techniques commonly found in the computational statistics and machine learning literature, with their mathematical descriptions and usage can be consulted in the following references [2], [8], [9], [13], [17].

Table IV Model column presents a plethora of models that this work has fitted for this prediction purpose: we started from simple predictive models such as Classical Linear Regression, Regression Tree, towards those that can seamlessly exhibit nonlinear behaviours, like Random Forest, Kernel Ridge Regression, Multi-Layer Perceptron and Support Vector Regression. Some of these methods had their hyperparameters held constant across all experiments (Fixed Hyperparameters column), or because we wanted to apply a particular form of a method (RBF kernel, single hidden layer, etc.) or because during a warm-up phase we noticed that they did not affect substantially the results (hyperbolic tangent, increasing number of trees, etc.).

For certain models, the Cross-Validated Parameters column shows which hyperparameters were optimised before the prediction step. For instance, suppose the case of Ridge Regression and the need to define the regularisation value (λ) appropriately. Consider that we have a set of training pairs $(features_t, R_t^{(1y)})_{t=1}^{L}$[2] of size L, and for this sample we subset it in k-rolling-cross-validation (k-rolling-cv) folders (better explained later in this subsection). Then, we train and test using this scheme the Ridge Regression model with one of the predefined λ, say $\lambda = 10^0$. We compute some performance function on the test set (Mean Squared Error – MSE) and repeat this process for all λ values available. We use in the final model the λ that on average had the lowest MSE.

The process explained previously is repeated for all cross-validated parameters in each model. When a model needs to cross-validate a pair or more of hyperparameters, the procedure is to perform all the feasible combinations of them (e.g., KRR-RBF with $(\lambda, \gamma) = \{(1, 0.01), (0.1, 0.01), ..., (0.001, 100)\}$). Another relevant point is that this process is in fact made inside a bigger loop called nested resampling [1], and that is the reason we have k and L inner and outer.

We fitted an usual benchmark found in the literature for forecasting modelling: the Naive model [18]. We also implemented the benchmarks that traders use to assess whether a particular MCCS is worth to be pitched or traded: BE Width and Carry at Expiry. We replicated the way traders look to these features, by computing z-scores[3] based on average and standard deviation on rolling window of size equal to 1 year. The signal for going long/short (S_t) is given by a thumb rule with a simple rationale: if a certain metric has a z-score above or equal to ± 3, the trader goes fully long (+)/short(-) in the trade, since it is a very extreme event. Otherwise, it reduces the leverage on it, until it below one standard deviation of distance from the rolling average.

We removed any missing data, and clipped extremes values, mainly in returns above the 95% percentiles (in our case it can be due to some numerical problems, or some extreme scenarios related to 2008-2009 financial crisis period). Next subsection presents the Stacking Approach, made to assemble all the base learners in a final model.

B. Stacking Approach

In general, the stacking approach consists of combining the underlying base learners in way that a consensual output is generated from it. Generically, for a particular trade we can express this combination by:

$$\hat{R}_t^{(1y)} = g(\hat{R}_{t,1}^{(1y)}, ..., \hat{R}_{t,k}^{(1y)}, ..., \hat{R}_{t,K}^{(1y)}) \tag{6}$$

where $\hat{R}_t^{(1y)}$ is the final output, comprising the combination of $k = 1, ..., K$ different base learners input ($\hat{R}_{t,k}^{(1y)}$). In this work, the way this combination is made, that is the functional form of g, is established by one of the four different forms: three linear and one nonlinear. Starting from the linear approaches, we can cast the general expression of g by:

$$\hat{R}_t^{(1y)} = \sum_{k=1}^{K} w_k \hat{R}_{t,k}^{(1y)} \tag{7}$$

with the weights w_k being defined by each approach as:

- Equally Weighted (Eq Weighted): all base learners wield the same weight, that is, or more precisely $w_k = 1/K$.
- Minimum Volatility Portfolio (Min Vol): the weights are defined by the Minimum Volatility estimation technique [21], in which the weights of each base learner are proportional to the inverse of the ensemble covariance matrix. Intuitively, base learners that provide more volatile returns and are highly-correlated are weighted less in the ensemble.
- Tangent Portfolio (Tang Port): the weights are defined by the Tangent Portfolio technique [21], where roughly speaking each base learner is weighted by its risk-adjusted return, penalised by how much diversity (low-correlation) is bringing to the portfolio.

It should be mentioned that all these approaches are traditional benchmarks used by the traders and found in the

[2]For the sake of brevity we dropped the subscript (i).

[3]a z-score is defined by: $Z - score = \frac{X-\mu}{\sigma}$ where X represent the actual value of a certain variable, μ and σ the average and standard deviation of X in a period.

TABLE IV
BASE LEARNERS USED TO MODEL THE MCCS TRADE DATASET.

Abbreviation	**Base learners**	**Fixed Hyperparameters**	**Cross-Validated Hyperparameters**
Classical	Classical Linear Regression	None	None
Ridge	Ridge Regression	None	$\lambda = \{10^0, 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}\}$
Bayes Ridge	Bayesian Ridge Regression	$\Gamma \sim (10^{-6}, 10^{-6})$ prior for noise	$\Gamma \sim (\{10^{-6}, 10^{-4}, 10^{-2}, 10^0\}, \{10^{-6}, 10^{-4}, 10^{-2}, 10^0\})$ prior for coefficients
Lasso	Lasso Regression	None	$\lambda = \{10^0, 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}\}$
Elastic Net	Linear regression with combined L1 and L2 priors as regularizer	None	$l_1 = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $\lambda = \{10^0, 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}\}$
KRR-RBF	Kernel Ridge Regression	Radial-Basis Function kernel	$\lambda = \{10^0, 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}\}$ and $\gamma = \{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$
CART	Classification and Regression Tree	MSE Function	Max depth $= \{2, 3, 5, 7\}$
Random Forest	Random Forest	Max depth $= 5$	Number of trees $= \{50, 100, 200\}$
Grad Boost	Gradient Boosting Tree	None	Number of trees $= \{50, 100, 200\}$ and Learning Rate: $\{0.1, 0.3, 0.5\}$
Extreme RF	Extra-Trees Regression	None	Number of trees $= \{50, 100, 200\}$
MLP	Multi-Layer Perceptron	Single hidden layer with hyperbolic tangent as transfer function	$\lambda = \{10^0, 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}\}$ and number of neurons $= \{5, 7, 12\}$
SVR-RBF	Support Vector Regression	Radial-Basis Function kernel	$C = \{10^0, 10^1, 10^2, 10^3\}$ and $\gamma = \{10^{-2}, 10^{-1}, 10^0, 10^1, 10^2\}$
AdaBoost	AdaBoost Regression	None	Number of estimators $= \{50, 100, 200\}$ and Learning Rate: $\{0.1, 0.3, 0.5\}$
Abbreviation	**Baseline Model**	**Parameter**	
Naive	Naive Model		
BE-Width	BE Width feature	Rolling window of size = 1 year	$S_t = \lfloor_{-1}((Z > 1) * Z)/(3)\rceil^{+1}$
CarryAtExpiry	Carry at Expiry feature	Rolling window of size = 1 year	$S_t = \lfloor_{-1}((Z > 1) * Z)/(3)\rceil^{+1}$
	Other Parameters	**Values**	
	Warm-up Period (L)	$L_{outer} = 2$ years	$L_{inner} = 1$ year
	k-rolling-cv	$k_{outer} = 1$ week for outer	$k_{inner} \approx (T_{train} - L_{inner})/5$ for inner
	Outlier Treatment	Winsorizing	
	Winsorizing Quantiles	0.01 and 0.95	
	Missing Data Treatment	Remove	

Observation: λ represents the regularisation parameter, while γ the precision parameter for the Radial-Basis Function (RBF) kernel.

financial literature. These approaches were contrasted with the Neural Network Stacker. This stacking approach make use of a Multi-Layer Perceptron (MLP Stacker), with hyperbolic tangent neurons composing it single-hidden layer. The consensus is, therefore, computed by:

$$\hat{R}_t^{(1y)} = \sum_{m=1}^{m} r_m \tanh \sum_{k=1}^{K} w_k \hat{R}_{t,k}^{(1y)} + c_k \tag{8}$$

with M representing the number of neurons, r_m and w_k the MLP's weights. The MLP Stacker required an extra-round of fine-tuning. Its parameters are the same used by the base learner as in Table IV, and we have used an extra set of data for this purpose. We also investigated the importance of each underlying model. Since an MLP do not provide seamlessly the relevance of each base learner, we have estimated it using a perturbation method:

$$\frac{d\hat{R}_t^{(1y)}}{d\hat{R}_{t,k}^{(1y)}} \approx \frac{\hat{R}_t^{(1y)}(\hat{R}_{t,k}^{(1y)} + h) - \hat{R}_t^{(1y)}(\hat{R}_{t,k}^{(1y)} - h)}{2h} \tag{9}$$

hence, we approximate the partial derivative by taking a central-difference for the base learner k. More precisely, by adding and subtracting a small perturbation in its prediction at time t, and evaluating its results in the output of the MLP Stacker. By doing it across all base learners, we are able to not only check which model plays a major role in the final prediction, but also check in which sense its input is being re-weighted (positively or negatively) .

Using the consensus prediction of a given stacking/base learner, the recommendation of a certain trade can be made solely on some normalised version of the expected return for holding 1y-period the i-th trade ($\hat{R}_{t,i}^{(1y)}$). Given that each model will be providing individual forecasts for each MCCS and after that their performance will be assessed locally and globally, a more suitable manner to proceed would be to assign a credit based on the tracking record of a model to predict a particular MCCS trade. Hence, we will be weighted up or down a signal not only based on the magnitude of a model prediction but also by its quality. Then, consider as $\hat{R}_{t,i}^{(1y)}$ the expected return for holding 1y-period the i-th MCCS trade. Now, define the new signal function $S_{t,i}$ by:

$$S_{t,i} = \frac{\hat{R}_{t,i}^{(1y)} \times Rho_{\hat{R}_{t,i}^{(1y)}, R_{t,i}^{(1y)}}}{\max(|\hat{R}_{t,i}^{(1y)} \times Rho_{\hat{R}_{t,i}^{(1y)}, R_{t,i}^{(1y)}}|, \ldots, |\hat{R}_{t-h,i}^{(1y)} \times Rho_{\hat{R}_{t-h,i}^{(1y)}, R_{t-h,i}^{(1y)}}|)} \tag{10}$$

where the strength of the i-th long/short signal is given by its expected return, scaled by the maximum weighted return that a long/short position on the same trade (that is why the returns are in absolute terms) was expected to yield in the previous h-period (in this case 1 year). Therefore, the trade with the maximum weighted return in absolute terms will have $|S_{t,i}| = 1$ as well as those close to zero will yield $S_{t,i} \approx 0$. The weight/credit of a certain prediction is based on the historical adherence between the actual and predicted values, that is, through the Pearson correlation coefficient (Rho).

C. Performance Metrics

Below we outline two types of metrics: one that focuses on the predictive performance that the model provided, and other four that are based on the profit/loss that its application harvested during the backtest. Set by $R_t^{(S_i)} = R_{t,i}^{(1y)} \times S_{t,i}(\hat{R}_{t,i}^{(1y)})$ the strategy return (combination of the realized/observed excess returns and the signal – function of a model prediction), we can compute the following metrics:

- Average Return (Avg Return): is the arithmetic average of the strategy returns:

$$\bar{R}^{(S_i)} = \frac{\sum_{t=1}^{T} R_t^{(S_i)}}{T} \tag{11}$$

- Standard Deviation: is the estimator of the dispersion around the strategy average returns (a risk measure in certain sense):

$$\sigma_{R^{(S_i)}} = \sqrt{\frac{\sum_{t=1}^{T}(R_t^{(S_i)} - \bar{R}^{(S_i)})^2}{T}} \tag{12}$$

- Information Ratio: is the average annualized return of a strategy earned in excess of a particular benchmark per unit of risk (measured in terms of standard deviation):

$$IR = \frac{\bar{R}^{(S_i)} - \bar{B}}{\sigma_{R^{(S_i)}}} \tag{13}$$

  where $\bar{B}$ is the average return of the benchmark (e.g., treasury bond, equity index). In our case, it was already set to the 3-month Treasury bill (Table III). It should be mentioned that Information Ratio makes each strategy performance comparable: since we are adjusting average returns by the risk assumed for each strategy, it removes the leverage component that is magnifying/shrinking the returns provided by a certain strategy. In analogy, it can be viewed as standardising a random variable by stripping it from its location and scaling factor.

Based on the methodology developed in this section, next one presents the results and discussions of this work.

IV. Results and Discussions

Table VI present the main results for each base learner and stacking approaches in terms of Information Ratio in the test set per MCCS. Overall, all methods outperformed the Naive model, implying that some predictability exist and can be achieved from the other features (Carry at Expiry, Breakevens, etc.) we have selected to compose each base learner. Traders benchmarks fared in average better than the machine learning approaches, especially the BE-Width Z-score scheme though CarryAtExpiry had controlled better the dispersion of the results across MCCS. From the predictive methods, Bayesian Ridge and Classical Ridge Regression outperformed in average the remaining piecewise linear and nonlinear methods (like SVR-RBF, KRR-RBF, etc.).

About the main findings of the stacking approaches (the last four rows of Table VI), in average the MLP Stacker outperformed the remaining traditional approaches, and in just very few occasions obtained a result lower than one the approaches (such as for USD 1y5y5y). To check whether this difference was substantial, Table V presents a statistical analysis using the average aligned ranks, Aligned-rank Friedman test and Holm posthoc procedure [7].

TABLE V
AVERAGE RANKS, FRIEDMAN AND HOLM POST-HOC STATISTICAL TESTS AND ANALYSIS FOR INFORMATION RATIO.

Model	Rank	Z-score	p-value	Holm Correction
Tang Port	38.16	4.23	**<0.0001**	0.016
Min Vol	36.76	4.01	**<0.0001**	0.025
Eq Weighted	35.93	3.88	**<0.0001**	0.050
MLP Stacker	**11.13**	-	-	-
Friedman Chi-Square	11.96		**0.0075**	

When we look at the average rank, MLP Stacker was the top positioned (11.13) while Tang Port remained most of the time as the worst choice (38.16), very close to the remaining approaches. In general, all Z-scores were above 3.8, resulting in very low p-values (<0.0001). If we set our initial significance level as 0.05 and correct using the Holm procedure (last column) we can assert that MLP Stacker performed significantly better than the other models. Therefore, MLP Stacker is capturing some information beyond that is not being spanned by the trader's benchmark, probably due to its nonlinear combination of the base learners – a statement that might require further investigations.

Delving into the MLP Stacker results, Figure 2 presents the normalised relevance of each base learner (following equation 9), aggregating across all MCCS. Overall, most machine learning approaches had low relevance, apart from the Lasso Regression and MLP (keeping as estimated the position signal), and Classic Regression and AdaBoost (reversing the position signal). Both trader's benchmark methods (Breakeven and Carry) accounts roughly for 15-20% of the final predicted value – as mentioned before these models in particular performed in average better than the remaining predictive models.

Figure 3 presents the observed annualised returns (red line), with the overlapped long-short positioning (area chart) of the MLP Stacker for the USD 3y5y5y MCCS trade. The MLP stacker was able to capture most opportunities to short the trade (when the return would have been negative) or go long in it (when the return was positive), avoiding volatile moments (Feb-2012 to Feb-2013) by keeping the exposure as neutral as possible.

V. Conclusions

This work proposed a trading recommendation system for Mid-Curve Calendar Spread Trades (MCCS) through a Neural Network stacking approach. We started with the main motivations, highlighting the bottlenecks that the derivatives traders face in their day-to-day routines. To tackle these issues, we propose a recommendation system that could analyse and rank a set of fixed income derivatives trades. Therefore, we started the methodology by showing the dataset: it comprised of 15 US Dollar MCCS trades, ranging from September 2006

TABLE VI
AVERAGE INFORMATION RATIO OF EACH BASE LEARNER AND STACKING APPROACH PER MCCS IN THE TEST SET.

Model	1y10y20y	1y1y1y	1y5y5y	2y10y20y	2y1y1y	2y5y5y	3y10y20y	3y1y1y	3y5y5y	4y10y20y	4y1y1y	4y5y5y	5y10y20y	5y1y1y	5y5y5y	Avg	Std
Classical	0.452	-0.224	0.432	0.003	0.457	-0.148	-0.151	0.750	-1.025	0.236	0.690	0.031	0.305	0.384	-0.150	0.136	0.449
Ridge	0.460	0.106	0.780	0.550	-0.103	-0.077	0.463	0.497	0.217	0.414	0.555	0.330	0.447	0.657	0.186	0.366	0.254
Bayes Ridge	0.452	0.021	0.708	0.310	-0.047	0.496	0.434	0.608	0.293	0.272	0.552	-0.172	0.400	0.707	0.035	0.338	0.273
Lasso	0.459	0.347	0.728	0.034	-0.187	0.596	-0.161	0.562	0.287	-0.180	0.775	0.022	-0.023	0.659	-0.356	0.237	0.380
Elastic Net	0.331	0.164	0.700	-0.346	-0.118	0.524	0.259	0.334	0.324	0.078	0.536	-0.090	0.415	0.634	-0.329	0.228	0.329
KRR-RBF	0.061	-0.066	0.218	-0.026	-0.271	-0.176	0.106	-0.071	0.131	0.244	0.218	-0.581	0.317	0.648	-0.239	0.034	0.292
CART	0.105	0.240	0.468	0.439	-0.523	0.183	0.143	0.055	0.131	0.021	0.257	-0.125	0.251	-0.025	0.276	0.126	0.240
Random Forest	0.120	0.323	0.629	0.490	-0.511	0.211	0.410	0.007	0.121	0.092	0.313	-0.218	0.062	-0.505	0.360	0.127	0.331
Grad Boost	0.211	0.340	0.793	0.472	-0.561	0.413	0.228	0.414	0.139	0.332	0.247	-0.118	0.090	-0.239	0.295	0.204	0.322
Extreme RF	0.145	0.027	0.779	0.373	-0.545	0.555	0.314	0.264	0.033	0.331	0.298	-0.093	0.442	0.320	-0.004	0.216	0.310
MLP	0.426	-0.175	0.417	0.083	-0.381	0.379	-0.032	0.147	0.110	0.076	0.413	0.000	0.100	0.469	-0.255	0.119	0.264
SVR-RBF	0.053	0.068	1.397	-0.247	-0.504	0.173	0.674	-0.228	-0.540	0.465	0.133	-0.141	0.212	0.721	0.078	0.154	0.504
AdaBoost	0.160	-0.055	0.512	0.386	-0.683	0.443	0.327	0.074	0.083	0.385	0.213	0.053	0.419	-0.094	0.206	0.162	0.298
CarryAtExpiry	0.686	0.302	0.427	0.367	0.253	0.428	0.295	-0.123	0.588	0.165	0.320	0.620	0.061	0.826	0.592	0.387	0.250
BE-Width	0.688	0.054	0.328	0.927	0.384	0.430	0.592	0.403	0.660	0.409	0.407	0.849	0.140	-1.005	1.093	0.424	0.485
Naive	0.370	0.127	0.363	0.214	-0.766	0.265	0.086	-0.448	0.372	0.068	-0.409	0.140	0.156	-0.758	0.166	-0.004	0.392
Eq Weighted	0.260	0.114	0.688	0.175	-0.243	0.105	0.071	0.266	-0.232	0.134	0.394	-0.028	0.259	0.672	0.096	0.182	0.265
Tang Port	0.052	-0.263	0.570	0.006	0.148	0.257	0.120	0.196	0.050	-0.214	0.491	-0.075	0.274	0.403	0.259	0.152	0.237
Min Vol	-0.658	0.457	0.725	0.117	-0.295	0.206	0.135	0.196	0.132	0.110	0.123	0.090	0.174	0.284	0.393	0.146	0.312
MLP Stacker	0.654	0.481	0.567	0.402	0.525	0.384	0.390	0.543	0.547	0.591	0.534	0.449	0.453	0.556	0.466	0.503	0.078

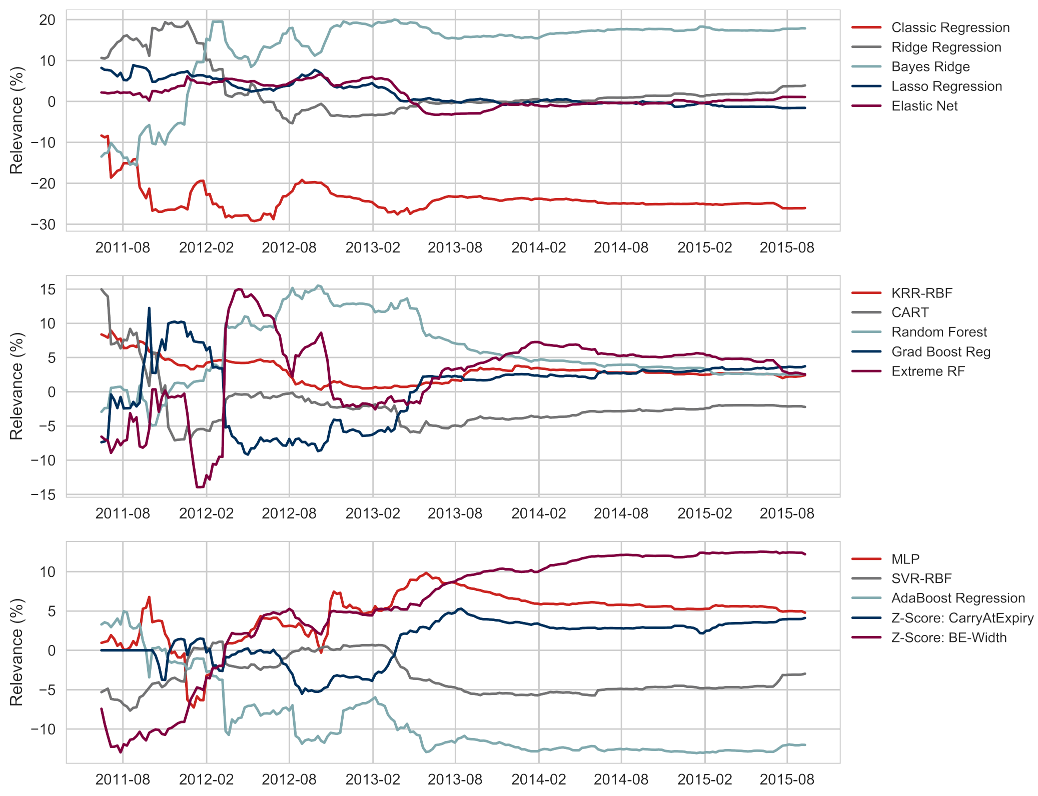

Fig. 2. Base Learner relevance for the MLP stacker, aggregated across all MCCSs.

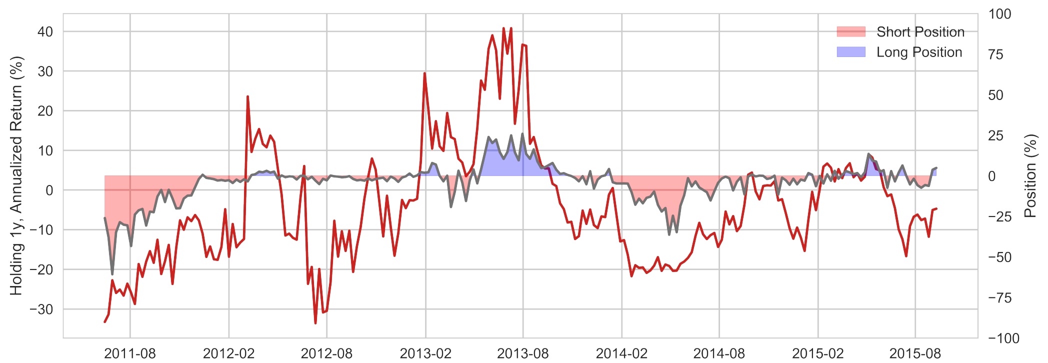

Fig. 3. Observed annualised returns (red line), and long-short signals (area chart) of the MLP stacker for the USD 3y5y5y MCCS trade.

to September 2016, with different expirations, forward and swap tenures. For each particular trade, we described how the sampling of inputs (metrics, sensitivities and lagged returns) and outputs (returns from unwinding the trade after one year of its start) were computed on a weekly basis. Then, we displayed the modelling strategy by highlighting the base learners used, and how the Stacking approaches, in special the Neural Network, were used to provide a consensual signal based on the inputs from the underlying models as well as other information to traders.

Our results suggests that our proposed Neural Network Stacking approach outperformed substantially the traditional approaches for pooling models in the financial literature. Also, regarding interpretability, our sensibility investigation on the models being most used – and the subsequent investigation that can be made over which features are more prevalent in which model – tend to make it easier to convey and convince the traders to use our recommendation system.

Future works should increase the sampling frequency: we used for this experiment data from weekly trades, and though this turned the modelling process faster, we had an undersampling close to a fifth. This, of course, reduced the power of our backtest as well as limited the range of models that can be applied. Also, some other works can devote some attention to better analyse the signal generating function used, as well as attempt to find out potential alternatives for the formulation used. Although it worked properly, we believe that some extra effort can overall improve models results and the nested resampling framework can help us find that out. Finally, other stacking approaches based on machine learning should be tried to enlarge and strengthen our findings.

Acknowledgements

Adriano Soares Koshiyama is funded by the National Research Council of Brazil (CNPq) through the Science Without Borders program. Also, the authors would like to thanks, Guillaume Andrieux, Tomoya Horiuchi, Gerald Rushton, Tam Rajendran, and Anthony Morris for all the comments and support during this research.

References

[1] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Claus Weihs. Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20(2):249–275, 2012.

[2] C Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2007.

[3] John Y Campbell, Andrew Wen-Chuan Lo, and Archie Craig MacKinlay. *The econometrics of financial markets*. Princeton University press, 1997.

[4] Eunsuk Chong, Chulwoo Han, and Frank C. Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187 – 205, 2017.

[5] Howard Corb. *Interest Rate Swaps and Other Derivatives*. Columbia University Press, 2012.

[6] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, 2017.

[7] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[8] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[9] Bradley Efron and Trevor Hastie. *Computer Age Statistical Inference*, volume 5. Cambridge University Press, 2016.

[10] Douglas S Ehrman. *The handbook of pairs trading: strategies using equities, options, and futures*, volume 240. John Wiley & Sons, 2006.

[11] Graham Elliott, Antonio Gargano, and Allan Timmermann. Complete subset regressions. *Journal of Econometrics*, 177(2):357 – 373, 2013.

[12] Nick Firoozye and Xiaowei Zheng. Market update: Forward vol and midcurve calendar spreads in usd and eur recent levels and carry and trades of note. *Nomura International plc, Nomura Research*, 2016.

[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[14] R. Gencay and Min Qi. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks*, 12(4):726–734, 2001.

[15] Eduardo A. Gerlein, Martin McGinnity, Ammar Belatreche, and Sonya Coleman. Evaluating machine learning classification for financial trading: An empirical approach. *Expert Systems with Applications*, 54:193 – 207, 2016.

[16] Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013.

[17] Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson, 2009.

[18] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[19] Andreas Karathanasopoulos, Konstantinos Athanasios Theofilatos, Georgios Sermpinis, Christian Dunis, Sovan Mitra, and Charalampos Stasinakis. Stock market prediction using evolutionary support vector machines: an application to the ase20 index. *The European Journal of Finance*, 22(12):1145–1163, 2016.

[20] Burton G Malkiel. The efficient market hypothesis and its critics. *The Journal of Economic Perspectives*, 17(1):59–82, 2003.

[21] Attilio Meucci. *Risk and asset allocation*. Springer Science & Business Media, 2009.

[22] Sheldon Natenberg. *Option volatility and pricing: advanced trading strategies and techniques*. McGraw Hill Professional, 2014.

[23] Hyejin Park, Namhyoung Kim, and Jaewook Lee. Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over {KOSPI} 200 index options. *Expert Systems with Applications*, 41(11):5227 – 5237, 2014.

[24] Riccardo Rebonato, Kenneth McKay, and Richard White. *The SABR/LIBOR Market Model: Pricing, calibration and hedging for complex interest-rate derivatives*. John Wiley & Sons, 2011.

[25] Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media, 2004.

[26] Nassim Taleb. *Dynamic hedging: managing vanilla and exotic options*, volume 64. John Wiley & Sons, 1997.

[27] Christian von Spreckelsen, Hans-Jrg von Mettenheim, and Michael H. Breitner. Real-time pricing and hedging of options on currency futures with artificial neural networks. *Journal of Forecasting*, 33(6):419–432, 2014.

[28] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[29] Tianle Zhou, Shangce Gao, Jiahai Wang, Chaoyi Chu, Yuki Todo, and Zheng Tang. Financial time series prediction using a dendritic neuron model. *Knowledge-Based Systems*, 105:214 – 224, 2016.

[30] Xiaocong Zhou, Jouchi Nakajima, and Mike West. Bayesian forecasting and portfolio decisions using dynamic dependent sparse factor models. *International Journal of Forecasting*, 30(4):963–980, 2014.