ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master Thesis Quantitative Finance

Non-parametric error correction for the structural modelling of Implied Volatility Surfaces

---

Name student: Jaron Biccler
Student ID number: 619184

Supervisor: Dr. G. Freire
Second assessor: Dr. A. Camehl

Date final version: 6th July, 2022

Abstract

Given the importance of implied volatility measures in finance for assessing risk, we investigate a two-step approach for predicting implied volatility surfaces through machine learning (ML) models. Random Forests, XGBoosted Trees, Support Vector Regressors and Feedforward Neural Networks (NN) are trained to correct the residuals of parametric models, ranging from Black-Scholes to the more recent Carr-Wu option pricing framework, calibrated on the S&P 500 index options from 2016 up to 2021. The interpolation and predictive empirical experiments conducted showcase the promising use of ML in adjusting these theoretically underpinned frameworks. XGBoost-based methods consistently ended up among the top performers, which showed a perceived lack of benefit to the increased complexity provided by NNs. The generality of the method allows for wide-scale use and several extensions, of which a novel ensemble method is explored here. This involves combining the predictions made by multiple parametric models to predict the overall IVS, rather than correcting their residuals on an individual basis. Said approach led to improved performance in settings where interpolation was assessed during future time frames, which we attribute to an improved resilience to changes in the financial environment. Feature importance scoring using SHapley Additive exPlanations (SHAP) showed, next to a preference for option specific variables such as moneyness and time to maturity, the Spot Volatility Index by Todorov (2019) to be the most important among the volatility-based measures. Furthermore, the inclusion of the calibrated coefficients of the respective parametric models proved beneficial in our attempts to learn their misspecification tendencies.

Contents

List of Tables

List of Figures

List of Abbreviations

AHBS	Ad-hoc Black-Scholes
ATM	At-the-money
BN	Batch Normalization
BS	Black-Scholes
BSM	Black-Scholes-Merton
CBOE	Chicago Board Options Exchange
CW	Carr-Wu
DE	Differential Evolution
DOTMC	Deep out-the-money call
DOTMP	Deep out-the-money put
DT	Decision Tree
EBA	European Banking Authority
FFNN	Feed Forward Neural Network
FRED	Federal Reserve Economic Data
FX	Foreign exchange
H	Heston
ITM	In-the-money
IV	Implied Volatility
IVS	Implied Volatility Surface
LASSO	Least Absolute Shrinkage and Selection Operator
LIME	Local Interpretable Model-Agnostic Explanations
MAE	Mean Absolute Error
ML	Machine Learning
MN	Moneyness
MSE	Mean Squared Error
NLS	Non-linear Least Squares
NN	Neural Network
OLS	Ordinary Least Squares
OM	OptionMetrics
OOS	Out-of-sample
OTM	Out-of-the-money
OTMC	Out-the-money call
OTMP	Out-the-money put
PCP	Put-call-parity
PES	Prediction Error Surface
ReLU	Rectified Linear Unit
RF	Random Forest
RMSE	Root Mean Squared Error
RT	Regression Tree
SHAP	Shapley Additive Explanations
SVI	Spot Volatility Index
SVR	Support Vector Regression
TTM	Time to maturity
VIX	CBOE's Volatility Index
WRDS	Wharton Research Data Services
XAI	Explainable AI
XGB	XGBoosted Trees

1 Introduction

The pricing of options has long been of interest to the financial world, which has led to the development of a plethora of structural models, of which the Black-Scholes model (BS) is the most famous one. The implied volatility (IV) as extracted from BS is frequently used as an alternative to the option price itself because it is easier to compare across options. However, the observed IVs, as a backed-out parameter given today's option prices of a particular underlying, often deviate from the fundamental assumption of constant volatility under BS. This has given rise to a wide array of different structural models that were developed, of which perhaps the most notable is the Heston model which incorporates the ideas of stochastic volatility modelling into a similar framework as the BS model, yet still has a quasi-closed-form solution (Heston, 1993).

As is common in derivative pricing, one tends to look at the cross-section of options available at a given time on a particular asset to calibrate the aforementioned structural models. Such a set, containing various times to maturity and strike prices, can be used to compose the implied volatility surface (IVS), where one plots the IVs against the times to maturity and strike prices.

These IVs play an important role in finance ranging from how institutional investors manage their open option positions (Carr & Wu, 2016) to constructing forward-looking volatility measures on which the wider finance community relies such as the VIX index. However, as models are never perfect, errors exist between the IV extracted from the calibrated structural models and the observed IV in the market. Hence, various attempts have been made to improve upon these structural models using machine learning techniques, ranging from using neural networks (NN) to directly estimate the option pricing function and IVS (cfr. literature overview by Ruf and Wang (2020)), to attempting to correct the errors of the structural models in a two-step estimation approach (Almeida et al., 2022). For the latter, the literature on this type of error correction is quite limited and the most appropriate (non)parametric techniques remain unclear, which proves to be an interesting case to explore further. Not only to provide support for the common choice of neural networks to capture the highly non-linear IVS, but also to explore a broader set of network architectures. Furthermore, as is shown by Ruf and Wang (2020), many different covariates have been used over the years in literature to predict option prices, which adds to the need to inspect their feature importance in greater detail.

Hence, this paper seeks to answer the research question: What are the most appropriate non-parametric techniques for the error correction of structural models for the IVS? This gives rise to the following subquestions that determine the general outline of the thesis:

1. Does the highly non-linear shape of the IVS favour non-parametric NNs over simpler ML

models?

2. Which NN architectures and regularization schemes are the most appropriate for correcting the errors made by structural models?
3. Which covariates are important in capturing the dynamics of the IVS?
4. Does combining the predictions by multiple structural models lead to a meaningful improvement in predicting the overall IVS?

The remainder of the paper is outlined as follows: Section 2 gives an overview of the existing literature, Section 3 describes the data, Section 4 proceeds to set out the methodology, while Section 5 states the empirical results.

2 Related Literature

Over the last few decades a wide array of theoretical option pricing models have been developed to satisfy the needs of the ever-growing space of financial players that require accurate option pricing for activities ranging from trading to hedging and market-making. These models start by specifying the dynamics, or lack thereof, of the price of the underlying and its variance. The most famous example remains Black-Scholes (BS) (Black & Scholes, 1973), which given its assumptions allows for the derivation of a closed-form solution. BS then sprouted a large set of variants that included extensions to different underlyings such as the Black model (Black, 1976), still commonly used for the pricing of futures, bond options and interest rate caps and floors. Similarly, Garman and Kohlhagen (1983) extended BS to foreign exchange (FX) option valuation. Others derived theoretical prices under jump-diffusion processes (Bates, 1991), while the well-known Heston model (Heston, 1993) utilized the ideas of stochastic volatility modelling, which was preceded by, amongst others, Hull and White (1987). Later on, these evolutions gave rise to attempts to combine the ideas such as in the co-jump models with self-exciting jump intensity by Andersen et al. (2015), Bates (2019), and Carr and Wu (2019). Given that this is just a subset of some of the more well-known option pricing models, it becomes clear that the literature is vast and continues to grow. More extensive overviews and empirical comparisons can be found in Bakshi et al. (1997) and Bates (2021).

Every one of these option pricing models has to make a set of assumptions to come to a (quasi) closed-form expression for the option prices. This includes distributional assumptions regarding the underlying price process, the interest rate process and the market price of factor risks (Bakshi et al., 1997). However, as is common for institutional investors, the management of option positions is often done based upon the option's implied volatility (IV) quote, rather than the option price itself (Carr & Wu, 2016). This IV is the backed-out volatility parameter

from the BS model given today's option specification and its price. As the BS assumptions state a constant and known risk-free rate and volatility, an underlying asset with a log-normal distributed price following a geometric Brownian motion, as well as frictionless markets (Black & Scholes, 1973), the frequently found empirical result of varying IVs for a set of options on the same underlying is a clear violation of one of its most fundamental assumptions. Investors, however, still use these IVs as a way to assess the cost of the option rather than the price itself, as it is more comparable across the option panel which might boast a wide array of different option specifications. Furthermore, investors do not observe the instantaneous variance rate, whilst it requires definition in most option pricing models, but instead, they observe these IVs as quoted in the market (Carr & Wu, 2016). The Implied Volatility Surface (IVS), as a map of the IVs against the times to maturity and strike prices (or moneyness), has henceforth become an extremely important tool for institutionals.

Many attempts have been made to steer away from the structural models to model the option pricing relationship or IVS directly in non-parametric ways. Ruf and Wang (2020) provide an extensive overview and meta-analysis of the vast literature surrounding NN-based approaches to achieving this. Starting with the first papers by Malliaris and Salchenberger (1993) and Hutchinson et al. (1994), the interest in this non-parametric modelling grew and gave birth to a large collection of different proposed architectures, features, datasets, and goals such as modelling either price or IV directly, outputting hedging strategies (Carverhill & Cheuk, 2003), enforcing no-arbitrage conditions (Dugas et al., 2009), or even using NNs to calibrate the structural models (Andreou et al., 2010).

One of the more novel ways of utilizing NNs in combination with structural models was proposed by Almeida et al. (2022). They showcase how NNs taking moneyness and time to maturity, potentially augmented with time-varying covariates, as input can be used to provide estimates of the errors made by calibrated structural models in terms of their IVS predictions. Their two-step estimation approach of calibrating a structural model, using it to predict an IV and thereafter correcting it with a prediction of its residual by a secondary NN-based model, allows them to boast impressive improvements over a collection of different prediction configurations. They discuss a range of structural models that cover the main aforementioned extensions and variants that have been introduced in theoretical option-pricing models. Starting with the standard BS model (Black & Scholes, 1973), followed by the Ad-hoc Black-Scholes (AHBS) model as a more practical implementation of the former (Dumas et al., 1998), the stochastic volatility Heston model (Heston, 1993) and the novel Carr-Wu option pricing framework (Carr & Wu, 2016). Almeida et al. (2022) explore the performance of their approach from different

angles. Firstly, taking a subset of the options on a given day, how well does their method predict the IVs of the remaining options. Secondly, given the set of options today, how well does it predict the IVS n-days-ahead. Thirdly, given all option data up to day t as training data, how well can a fitted Heston model, with potential error correction, use future state variables to predict IVs of the test set.

Hence, it should be clear to the reader that the work by Almeida et al. (2022) provides a great basis to showcase the viability of their approach. Yet, as is always the case, there exist certain limitations to their work that we wish to improve upon and generalize further in this paper. One of which, is the fact that only 5 different NN architectures ranging from 1 to 5 hidden layers are explored, following the proposed architectures by Gu et al. (2020) using sigmoid activation functions. However, NNs are notorious for being difficult to tune and tend to not reach their full potential when such a limited range of predefined architectures is explored. The authors themselves are aware of this limitation as they state:

> "As one could still try to optimize over different choices of network architecture, activation function and optimization algorithm, our results can be seen as a lower bound on the performance of our model-guided approach..." (Almeida et al., 2022)

This becomes even more important considering the absence of regularization terms and the use of the sigmoid activation function. As the latter results in what is known as vanishing gradients which refers to the fact that NNs showcase exponentially decreasing backpropagated error terms as a function of the distance to the output layer (Sussillo & Abbott, 2014). Hence, when not properly accounted for, this can prevent good convergence of deeper NNs. A common way to (partially) resolve this is through the use of different activation functions such as the Rectified Linear Unit (ReLU) (Sussillo & Abbott, 2014). Furthermore, the desire for deeper neural networks rather than shallower ones (Poggio et al., 2017) and the inherent tendencies of more complex networks to overfit, creates a need to explore regularization techniques in greater detail. Popular approaches in general NN literature include L1 and L2 regularization, dropout, batch normalization, early stopping, etc. which can then be combined with the aforementioned ReLU activation functions rather than the sigmoid functions to avoid vanishing gradients. Hence, one of the goals of this paper will be to explore the potential gains of further NN tuning and more complex architectures.

Lastly, one of the main reasons for the popularity of the NN as a non-parametric approach to modelling the IVS, is the highly complex shapes the IVS itself can take, which a general approximator such as a NN can theoretically model. This has given rise to hundreds of papers covering NNs in the option pricing space (Ruf & Wang, 2020). However, there seems to be

lacking literature on other parametric and non-parametric models in attempting the same feat, potentially with lower complexity. Various common machine learning techniques that could presumably do this as well include, but are not limited to, Support Vector Regressors (SVR), Random Forests (RF) and eXtreme Gradient Boosted Trees (XGBoost) (Gu et al. (2020)). This ties back to the secondary goal of this paper, namely the desire to compare more complex NN-based architectures to simpler models and assess the potential gains, or lack thereof, that the added complexity might bring in correcting the residuals of the structural models.

3 Data

As main data source we use European-style S&P 500 (SPX) options traded on the Chicago Board Options Exchange (CBOE) ranging from January 4, 2016 to December 31, 2021, acquired from OptionMetrics (OM) Ivy DB through Wharton Research Data Services (WRDS). These SPX options are highly liquid and are known to have a large number of different contract specifications available at any given time, which highly facilitates the calibration of our models. Additionally, this is a common choice in literature which aids the comparability of our work. OM's data includes both standard (settled using the opening price of the expiration date) and weekly SPX options (settled using the closing price of the previous business day) (CBOE, 2022), with best bid and ask prices, volume, open interest, as well as OM's own IV measure and their Greeks (delta, gamma, vega, theta).

The data cleaning involves filtering out zero-volume options and options in violation of no-arbitrage conditions. The time to maturity (TTM) is calculated in days by subtracting the current date from the expiry date, where options with AM settlement receive a correction by subtracting 1 day. Afterwards, this is transformed to a yearly TTM by dividing by 365, assuming a standard calendar year. Furthermore, it is common in the literature to focus on out-of-the-money (OTM) calls and puts, given their better liquidity than deep in-the-money (ITM) options, which is a practice also followed by Almeida et al. (2022) and hence will be replicated here as well. When discussing the price of an option, we consider the mid price as the average between the best bid and ask quotes provided by OM at end-of-day.

As Wallmeier (2022) shows, several issues exist with the current implementation of the IV calculation in the widely used OM Ivy DB (OptionMetrics, 2021) for options on underlyings that pay dividends such as the S&P 500 index used here. Namely, OM uses non-synchronous index and option prices, combined with an average implied dividend yield rather than specific dividend yields per option term. This causes violations of the put-call parity (PCP) to occur in OM's IVs, even though the actual option prices did not violate said parity. Hence, we follow the

practice used by, amongst others, Aït-Sahalia and Lo (1998), Almeida et al. (2022), Binsbergen et al. (2012), Chen and Xu (2014), Fan and Mancini (2009), and Golez (2014), which is also proposed by Wallmeier (2022) as a common solution. Following the notation of Wallmeier (2022), this involves creating a synthetic dividend-adjusted index level, $A_t(K,T)$, from the most liquid options, which we choose to be the options that are closest to being at-the-money (ATM) (strike K^{ATM}), at a given time t with a given expiry date T and across all strikes using the PCP,

$$A_t(K,T) = C_t(K,T) - P_t(K,T) + Ke^{-r(T-t)}, \tag{1}$$

with K as the strike price, $C_t(K,T)$ as the call price, $P_t(K,T)$ as the put price and r as the risk-free rate. This adjusted index level per term for $K = K^{ATM}$ can then be used with a zero dividend rate in e.g. BS, or the original index level with the dividend rate that corresponds with it. Thus, in what follows, when we refer to the observed IVs, we are not referring to the IV measures provided by OM, but rather the IVs that were extracted through the BS model given this dividend-adjusted index level, $A_t(K^{ATM},T)$. Similarly, when talking about the dividend yield, this refers to the dividend yield that links $A_t(K^{ATM},T)$ to the current index level, not the dividend yield that can be found in OM's Ivy DB.

As stated earlier, we solely use OTM options for their increased liquidity. Next, we follow the approach by Fan and Mancini (2009) and Almeida et al. (2022) for comparability sake and only keep options with TTM, $\tau = T - t$, between 20 and 240 calendar days, as well as moneyness ,$m_{i,t} = S_t/K_{i,t}$, between 0.80 and 1.60. By doing so, our final dataset consists of 2,621,192 observations, with an average of 1,747 options in the cross-section per trading day. Furthermore, a classification is made with respect to moneyness and TTM. Options are grouped as deep OTM calls (DOTMC) for $m_{i,t} \in [0.80, 0.90)$, OTM calls (OTMC) for $m_{i,t} \in [0.90, 0.97)$, ATM options for $m_{i,t} \in [0.97, 1.03)$, OTM puts (OTMP) for $m_{i,t} \in [1.03, 1.10)$ and finally, deep OTM puts (DOTMP) for $m_{i,t} \in [1.10, 1.60]$. Similarly, in terms of maturity we define options as short-term for a number of days to expiration $\in [20, 60]$ and long-term $\in (60, 240]$. Table 1 gives the descriptives of the option dataset following this classification. Additionally, more extensive descriptives and visualizations can be found in Appendix A.

Furthermore, as the risk-free rate we use the 3-Month Constant Maturity U.S. Treasury Securities data, while as spread measure we use the 3-Month Treasury Constant Maturity Minus Federal Funds Rate. Both are acquired from the St. Louis Federal Reserve Economic Data (FRED).[1] As additional covariates, we include the CBOE VIX index which can be found through WRDS. Similarly, the popular Realized Variance measure, as well as the Bipower

[1] https://fred.stlouisfed.org/series/DGS3MO

Table 1: Summary statistics of OptionMetrics data aggregated by moneyness and TTM categories.

	Number		Mean IV (%)		Std. IV (%)	
	Short	Long	Short	Long	Short	Long
DOTMC	29,194	43,977	24.638	16.831	10.921	6.604
OTMC	270,540	155,914	13.535	13.807	6.661	5.501
ATM	477,403	182,795	14.240	16.254	6.613	5.766
OTMP	421,861	167,725	20.482	20.452	6.710	5.812
DOTMP	501,138	370,645	33.500	29.589	11.172	7.940
Total	1,700,136	921,056	21.532	21.998	11.710	9.388

This table reports the number of option specifications and the mean and standard deviation of IV per category of moneyness and TTM for our data sample ranging from January 4, 2016 to December 31, 2021.

Variation, can be found on the website of the Oxford-Man Institute of Quantitative Finance.[2] All of these external datasets are considered with a daily frequency over the same sample period as the OptionMetrics data. Lastly, we include the Spot Volatility Index (SVI) by Todorov (2019) for which data is only available until 31/12/2020. Hence, we reuse the `MATLAB` code made available by them on their website to acquire the data for the year 2021 and merge our data with theirs.[3]

4 Methodology

The proposed methodology will be explained in greater detail throughout the next subsections, starting with a succinct coverage of the different parametric models that will be calibrated, followed by the proposed ML models to correct their residuals. Next, a description of the different empirical experiments is given, succeed by how the performance of our methods will be evaluated and finally, the used feature importance measures are discussed.

4.1 Parametric Models

Given the desire to maintain comparability with the earlier results by Almeida et al. (2022), we opt to work with the same set of parametric models, as these cover the main different evolutions in option pricing over time. Furthermore, as the generality of the error correction methodology is of key importance, the secondary models should preferably work well across a wide array of different parametric models. Therefore, the exact choice of which parametric models are used is of lesser importance than making sure the choices are diverse enough to argue the generality of the proposed methods. Hence, in the following sections we will briefly discuss the Black-Scholes, Ad-hoc Black-Scholes, Heston and Carr-Wu models which will be explored here and,

[2] https://realized.oxford-man.ox.ac.uk/
[3] https://tailindex.com/index.html

when possible, we follow the notation used by Almeida et al. (2022). Note that, although Ad-hoc Black-Scholes is not technically a structural model, when we refer to these types of models as a set in the remainder of the paper we will use structural, parametric or primary models interchangeably.

4.1.1 Black-Scholes

The well-known Black-Scholes (BS) model (Black & Scholes, 1973) allows for the derivations of a closed-form solution for the European option price given a set of assumptions on the underlying:

1. The short-term interest rate, r, is known and constant over time.
2. The stock price, S_t, follows a geometric Brownian motion with constant drift, μ, and volatility, σ.
3. The stock pays no dividends.
4. There are no transaction costs.
5. There are no arbitrage opportunities.
6. You can borrow or lend any amount at the riskless rate, r.
7. You can buy or sell any amount, even fractional, of the stock.

Hence, the stochastic differential equation of the stock price, S_t, with W_t as a standard Brownian motion, can be written as

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t \tag{2}$$

Following the derivation in Black and Scholes (1973) leads to the following theoretical price of a European call option, C^{BS}, with strike K and TTM τ,

$$C^{BS}(S_t, K, \tau, r, \sigma) = \Phi(d_1)S_t - \Phi(d_2)Ke^{-r\tau}, \tag{3}$$

$$d_1 = \frac{\ln \frac{S_t}{K} + (r + \frac{\sigma^2}{2})\tau}{\sigma\sqrt{\tau}}, \tag{4}$$

$$d_2 = d_1 - \sigma\sqrt{\tau}, \tag{5}$$

with $\Phi(\cdot)$ as the CDF of the standard normal distribution. This implies that $\Phi(d_2)$ can be interpreted as the probability that the option expires ITM and hence will be exercised. Whilst $\Phi(d_1)S_t$ can be seen as the present value (discounted with r) of the expected value of the stock price at maturity, S_T, given that the option expires ITM.

The volatility parameter, σ, which is assumed to be known and constant, is in practice the only parameter that is not known or does not have a clear proxy when observing the option price, C_t. Hence, solving the equation $C_t - C^{BS}(S_t, K, \tau, r, \sigma) = 0$ for σ yields what is known as

the IV of the option. This, however, does not have a closed-form solution and requires numerical solutions instead. Given the importance of IV and the desire for fast calculation in settings such as high-frequency trading (HFT), different approaches have been developed to cut down on the computation time whilst maintaining precision. We will use the popular implementation by Jäckel (2015), which calculates IV with as little as two iterations whilst maintaining maximum attainable precision on 64-bit floating-point hardware.[4] The exact implementation used is the one available in the Python wrapper for the `Vollib` library.

A common fact in option data is the violation of the constant volatility assumption that the BS model makes. When one calculates the IVs and maps them out as the IVS, this will usually differ significantly from the flat IVS we would expect under the BS assumptions. As a matter of fact, typical shapes for the IVS include what is referred to as the volatility smiles, which imply a deviation from the assumed log-normal distribution for the underlying (Hull, 2017). Furthermore, Andersen et al. (2015) showcase that the IVS displays highly persistent and non-linear dynamics and is thus far from constant and flat over time. A visualization of the IVS on the last date of the data, which shows this violation, can be found in the Appendix Figures A.3 and A.4. Calibration of the BS model on the daily cross-section can be done by finding the single σ value which minimizes the RMSE of the predicted IVs against the observed IVs. In practice this can simply be implemented by regressing the observed IVs of day t, $\sigma_{i,t}$, on a constant using Ordinary Least Squares (OLS) which yields a type of average IV for the day and can be specified as

$$\sigma_{i,t} = \alpha_{0,t} + \epsilon_{i,t} \qquad i = 1, \ldots, n \tag{6}$$

Hence, you end up with a flat IVS prediction on the day, or thus the same IV for all option specifications that day, $\hat{\sigma}_t^{BS} = \hat{\alpha}_{0,t}$, which is in accordance with the BS assumptions.

4.1.2 Ad-hoc Black-Scholes

The Ad-hoc Black-Scholes (AHBS) model can be seen as a practitioner's implementation of the aforementioned theoretical BS model. Here, Dumas et al. (1998) try to tackle some of the downsides they see in the BS model and models such as the Heston model that use stochastic volatility. The main issue with the former is as stated earlier, the failure to describe the structure of observed option prices due to the incorrect constant volatility assumption. While stochastic volatility models generally require some market price of risk parameters, which, among other things, are difficult to estimate (Dumas et al., 1998). They focus directly on modelling the IV

[4]Note that this is also the standard implementation in `MATLAB` and `Vollib`.

to be more in line with how financial institutions put the theory to practice, i.e. they tend to model the IV first and subsequently translate this to option prices, rather than the other way around. Hence, Dumas et al. (1998) propose to model IV as a function of the strike and TTM. In practice, their model becomes a simple linear regression that can be fit on an option panel on day t, where we opt to use moneyness instead of the strike price itself, which we specify as

$$\sigma_{i,t} = \alpha_{0,t} + \alpha_{1,t} m_{i,t} + \alpha_{2,t} m_{i,t}^2 + \alpha_{3,t} \tau_{i,t} + \alpha_{4,t} \tau_{i,t}^2 + \alpha_{5,t} m_{i,t} \tau_{i,t} + \epsilon_{i,t}, \qquad i = 1, \ldots, n, \tag{7}$$

where $\sigma_{i,t}$ is the observed IV of option i on day t, similarly, $m_{i,t}$ and $\tau_{i,t}$ are respectively the option's moneyness and TTM. This is only one of the quadratic specifications that they test, models excluding the TTM and/or the quadratic term for TTM are left out here for brevity sake. Estimation using OLS comes down to minimizing the RMSE of the predicted IVs against the observed IVs, which yields the set of estimated parameters $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_{0,t}, \hat{\alpha}_{1,t}, \hat{\alpha}_{2,t}, \hat{\alpha}_{3,t}, \hat{\alpha}_{4,t}, \hat{\alpha}_{5,t})$ with predicted IVs, $\hat{\sigma}_{i,t}^{AHBS}$. As Almeida et al. (2022) correctly note, by fixing all coefficients to zero except the intercept, $\alpha_{0,t}$, we get the exact same calibration scenario as discussed during Subsection 4.1.1 resulting in a flat BS-type IVS prediction.

4.1.3 Heston

In order to improve upon the constant volatility assumption of the BS model, Heston (1993) followed earlier work by, amongst others, Hull and White (1987) to introduce the concept of stochastic volatility modelling. This implies that not only the asset price process is defined as a stochastic process, but the volatility process as well. Commonly referred to as the Heston model, the dynamics under risk-neutrality are modelled as

$$\frac{dS_t}{S_t} = r dt + \sqrt{V_t} dW_{1,t}, \tag{8}$$

$$dV_t = \kappa \left(\bar{v} - V_t\right) dt + \sigma_v \sqrt{V_t} dW_{2,t}, \tag{9}$$

with V_t as the spot variance and κ the rate at which it mean reverts to the long-run variance $\bar{v}$. Meanwhile, σ_v represents the volatility of the volatility generating process itself in which randomness is introduced through the standard Brownian motion $W_{2,t}$, which shares a correlation ρ with $W_{1,t}$. Note, that the volatility process follows an Ornstein-Uhlenbeck process rewritten to the familiar square-root style used by Cox et al. (1985). Furthermore, if the parameters obey the Feller condition, $2\kappa\bar{v} > \sigma_v^2$, then the process V_t is strictly positive. Following the derivations in Heston (1993), a quasi-closed-form solution for the pricing of European options can be found (Appendix B). It remains quasi-closed due to the presence of an integrand that can not be

solved analytically but needs to be numerically approximated instead.

The implementation used here for the calibration of Heston models, given a set of options, relies on the default implementation of the `QuantLib` library. More specifically, this numerically solves the expressions found by Fourier transform, which can yield two different, but equivalent, algebraic formulations of the complex logarithm seeing as that the Heston model belongs to the class of affine, stochastic volatility models (Albrecher et al., 2006; Duffie et al., 2000). Using `QuantLib`'s pricing engine we can thus find a price for the option, assuming that all parameters are given, $\boldsymbol{\xi}_t = (V_t, \bar{v}, \kappa, \sigma_v, \rho)$. These parameters, however, need to be calibrated, for which we use an iterative produce. Given an initial estimate for the parameter values for day t, $\hat{\boldsymbol{\xi}}_t^0 = \left(\hat{V}_t^0, \hat{\bar{v}}^0, \hat{\kappa}^0, \hat{\sigma}_v^0, \hat{\rho}^0\right)$, we calculate the option prices these would yield for the different option specifications on the day. These prices are subsequently translated to IVs using Jäckel (2015)'s formula, which allows us to calculate the MSE of the predicted IVs of the day as

$$\frac{1}{n}\sum_{i=1}^{n}\left[\sigma_{i,t} - \hat{\sigma}_H^j\left(\hat{\boldsymbol{\xi}}_t^j, S_t, K_{i,t}, \tau_{i,t}, r_t\right)\right]^2, \qquad j = 0, 1, \ldots, \tag{10}$$

with $\hat{\sigma}_H^j(\hat{\boldsymbol{\xi}}_t^j, S_t, K_{i,t}, \tau_{i,t}, r_t)$ as fitted Heston IV values for the j^{th} iteration. With this objective function value, we resort to the default implementation of Non-Linear Least Squares (NLS) found in the `SciPy` library to provide us with the parameter estimate of the next iteration, $j+1$, which continues until one of the default stop conditions is met.

When fitting Heston models on the daily cross-section, we initially use the differential evolution (DE) optimisation algorithm found in the `SciPy` library for the very first day rather than NLS, but with the same objective function. This allows us to presumably find the global optimum and avoid having to specify an initial estimate for the first day of the dataset. After which the subsequent days use the previous day's parameters as an initial guess using NLS to find the optimum following the procedure discussed above, as DE is too computationally expensive to run every day.

4.1.4 Carr-Wu

Starting from the near-term dynamics of the IVS, Carr and Wu (2016) derive no-arbitrage conditions on the current shape of the IVS. This allows them to propose a novel option pricing framework that is more in line with how institutional investors manage their outstanding option positions. They attempt to tackle the downside that traditional option pricing theory tends to require a full specification of the instantaneous variance rate dynamics, ranging from flat and constant under BS to the stochastic volatility equation under Heston, while in practice the

instantaneous variance is unobserved. Furthermore, the map between the IVS and the instantaneous variance rate dynamics is not always clear, which requires institutions to frequently recalibrate their models to deal with changing market conditions, although these parameters are theoretically fixed over time (Carr & Wu, 2016).

Given their set of assumptions on the IV dynamics, Carr and Wu (2016) show that the shape of the entire IVS can be cast as a solution to a quadratic equation. This allows them to model the shape of the current IVS as a function of the current levels of drift and diffusion processes, without depending on how these processes will evolve in the future.

The risk-neutral dynamics of the stock price under the Carr-Wu (CW) model are

$$\frac{dS_t}{S_t} = \sqrt{v_t} dW_t, \tag{11}$$

where v_t denotes the instantaneous variance rate at time t, which follows a positive, real-valued stochastic process. However, they do not specify what this instantaneous variance rate looks like under risk-neutrality, rather they directly specify the risk-neutral dynamics of the BS IV instead as

$$d\sigma_t(K,T) = \mu_t dt + \omega_t dZ_t, \tag{12}$$

with μ_t and ω_t as the drift and volatility of the IV process. Both W_t and Z_t are standard Brownian motions with a correlation of $\rho_t \in [-1,1]$: $E_t[dW_t dZ_t] = \rho_t dt$. Choosing both the drift, μ_t, and diffustion, ω_t, to be proportional to the IV level yields

$$\frac{d\sigma_t(K,T)}{\sigma_t(K,T)} = e^{-\eta_t(T-t)}(m_t dt + w_t dZ_t), \tag{13}$$

where m_t, w_t, η_t are stochastic processes that do not depend on K, T or $\sigma_t(K,T)$. As Carr and Wu (2016) state, the exponential dampening $e^{-\eta_t(T-t)}$ term is used to accommodate the empirical observation that IVs of longer dated options tend to move less.

Defining $\log K/S_t = k$ as the relative strike price and following their derivations under no-arbitrage assumptions with this proportional specification of drift and diffusion leads to a quadratic equation, which formulates the IVS as a function of k and TTM, $\tau = T - t$, as

$$\begin{aligned} &\frac{1}{4}e^{-2\eta_t\tau}w_t^2\tau^2\sigma_t^4 + \left(1 - 2e^{-\eta_t\tau}m_t\tau - e^{-\eta_t\tau}w_t\rho_t\sqrt{v_t}\tau\right)\sigma_t^2 \\ &- \left(v_t + 2e^{-\eta_t\tau}w_t\rho_t\sqrt{v_t}k + e^{-2\eta_t\tau}w_t^2k^2\right) = 0 \end{aligned} \tag{14}$$

Equation 14 shows that this no-arbitrage constraint only depends upon the current levels of the stochastic processes $(m_t, w_t, \eta_t, v_t, \rho_t)$ and thus does not depend on the exact dynamics of

these processes. Hence, one can leave the dynamics of these processes unspecified when fitting the relation to observed IV surfaces, as extracting the levels of the five states is sufficient. More specifically, we can fit the IVS on day t by treating the states as parameter values that we estimate by numerically solving Equation 14 on day t's cross-section of options. Thus, given the cross-section of options on day t, we attempt to find the optimal set of parameters $\boldsymbol{\theta}_t = (v_t, m_t, w_t, \eta_t, \rho_t)$ such that the left hand side of Equation 14 approximates zero as closely as possible. Hence, the objective is to find the estimated parameters $\hat{\boldsymbol{\theta}}_t$ satisfying

$$\begin{aligned}\hat{\boldsymbol{\theta}}_t = \underset{\boldsymbol{\theta}_t}{\operatorname{argmin}} \sum_{i=1}^{n} \big[1/4 e^{-2\eta_t \tau_{i,t}} w_t^2 \tau_{i,t}^2 \sigma_{i,t}^4 + \left(1 - 2e^{-\eta_t \tau_{i,t}} m_t \tau_{i,t} - e^{-\eta_t \tau_{i,t}} w_t \rho_t \sqrt{v_t} \tau_{i,t}\right) \sigma_{i,t}^2 \\ - \left(v_t + 2e^{-\eta_t \tau_{i,t}} w_t \rho_t \sqrt{v_t} k_{i,t} + e^{-2\eta_t \tau_{i,t}} w_t^2 k_{i,t}^2\right)\big]^2,\end{aligned} \tag{15}$$

where $\sigma_{i,t}$ represent the observed IV of the respective option with $k_{i,t}$ and $\tau_{i,t}$. The optimization again takes place iteratively using NLS given an initial estimate of the parameter set $\hat{\boldsymbol{\theta}}_t^0$. We follow a similar approach as during Subsection 4.1.3. This involves running DE optimization on the first date of the dataset to find a presumably global solution. Afterwards, the initial estimate used during subsequent days is chosen as the optimized parameters of the previous day and optimization is done using NLS. The actual Carr-Wu prediction of the IV for a calibrated parameter set $\hat{\boldsymbol{\theta}}_t$ can be obtained by solving the quadratic Equation 14 for σ_t^2 given a particular k and τ. By then taking the positive square root of the positive solution to the quadratic equation we find the estimated IV, $\hat{\sigma}^{CW}(\hat{\boldsymbol{\theta}}_t, k, \tau)$.

4.2 Error Correction Models

Following the methodology by Almeida et al. (2022), the purpose of this paper is to use ML models to attempt to correct the residuals of the previously outlined parametric models in terms of their IVS predictions. Where, as stated earlier, this IVS is the map between the implied volatilities, σ, on the one hand, and the moneyness, m, and TTM, τ, on the other hand, which can be represented as an unspecified function $\sigma(m, \tau)$ that we desire to learn. Given the fact that, although much progress has been made over the last few decades, structural models still remain misspecified depending on to what extent their set of underlying assumptions deviates from the real world. Hence, the IVS as predicted by a calibrated structural model on a given day will differ from the observed IVS. This leads Almeida et al. (2022) to coin the term pricing error surface (PES) as $\epsilon_p(m, \tau) = \sigma(m, \tau) - \sigma_p(m, \tau)$, which can be thought of as the difference between the three-dimensional observed IVS and its predicted equivalent by the parametric model, $\sigma_p(m, \tau)$.

Considering the existence of this PES, a two-step estimation approach, where a secondary non-parametric model attempts to predict the shape of the PES, allows us to correct the predicted IVS of the structural model and hence should result in an overall closer approximation of the observed IVS. Thus, given a n-sized cross-section of options, $i = 1 \dots n$, on day t, the goal becomes to calibrate a parametric model, p, to the observed IVS, $\sigma(m_{i,t}, \tau_{i,t})$, which results in the fitted values, $\hat{\sigma}_p(m_{i,t}, \tau_{i,t})$. Thenceforth, we take the residuals $\hat{\epsilon}_p(m_{i,t}, \tau_{i,t}) = \sigma(m_{i,t}, \tau_{i,t}) - \hat{\sigma}_p(m_{i,t}, \tau_{i,t})$, which can now be used as the dependent variable for the secondary non-parametric model, s, which has a second set of parameters $\boldsymbol{\theta}_s$. This implies we want to minimize the difference between our structural model's residuals, $\hat{\epsilon}_p(m_{i,t}, \tau_{i,t})$, and its predicted residuals, $\hat{f}_s(m_{i,t}, \tau_{i,t}|\hat{\boldsymbol{\theta}}_s)$ by model s, by finding the estimated optimal parameter set $\hat{\boldsymbol{\theta}}_s$. As structural models tend to be calibrated by minimizing the MSE of the observed versus the predicted IVS, we opt to fit the secondary model by minimizing the MSE as well. Hence, the secondary objective becomes

$$\hat{\boldsymbol{\theta}}_s = \underset{\boldsymbol{\theta}_s}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} [\hat{\epsilon}_p(m_{i,t}, \tau_{i,t}) - f_s(m_{i,t}, \tau_{i,t}|\boldsymbol{\theta_s})]^2, \tag{16}$$

Referring back to our structural model, p, we had, $\hat{\epsilon}_p(m_{i,t}, \tau_{i,t}) = \sigma(m_{i,t}, \tau_{i,t}) - \hat{\sigma}_p(m_{i,t}, \tau_{i,t})$, such that we now get a second residual $\hat{\nu}_{p,s}(m_{i,t}, \tau_{i,t}) = \hat{\epsilon}_p(m_{i,t}, \tau_{i,t}) - \hat{f}_s(m_{i,t}, \tau_{i,t})$. When combined this yields

$$\sigma(m_{i,t}, \tau_{i,t}) = \hat{\sigma}_p(m_{i,t}, \tau_{i,t}) + \hat{f}_s(m_{i,t}, \tau_{i,t}) + \hat{\nu}_{p,s}(m_{i,t}, \tau_{i,t}), \tag{17}$$

$$\hat{\sigma}_{p,s}(m_{i,t}, \tau_{i,t}) = \hat{\sigma}_p(m_{i,t}, \tau_{i,t}) + \hat{f}_s(m_{i,t}, \tau_{i,t}), \tag{18}$$

with the overall predicted IV for option i at time t of a particular combination of primary model, p, and secondary model, s, as $\hat{\sigma}_{p,s}(m_{i,t}, \tau_{i,t})$.

Furthermore, this paper sets out to explore the added benefit of an ensemble based approach, where rather than predicting individual residuals, we predict the IVS directly based on the combined predictions of several parametric models. This implies that given a set of calibrated parametric models, we construct a vector of IV predictions made by those parametric models per particular option specification, $\hat{\boldsymbol{\sigma}}_{i,t}$. More specifically, we choose $\hat{\boldsymbol{\sigma}}_{i,t} = (\hat{\sigma}_{i,t}^{BS}, \hat{\sigma}_{i,t}^{AHBS}, \hat{\sigma}_{i,t}^{H}, \hat{\sigma}_{i,t}^{CW})$. We then look to learn a new model, c, with functional form f_c, defined by a parameter set $\boldsymbol{\theta}_c$. We follow a similar approach as before, but now using the observed IVs directly as dependent variable, rather than a residual term of a particular parametric model. Hence, we can find the

estimated set of relevant parameters, $\hat{\boldsymbol{\theta}}_c$, by solving the following objective

$$\hat{\boldsymbol{\theta}}_c = \underset{\boldsymbol{\theta}_c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} [\sigma(m_{i,t}, \tau_{i,t}) - f_c(\hat{\boldsymbol{\sigma}}_{i,t}, m_{i,t}, \tau_{i,t} | \boldsymbol{\theta}_c)]^2 \tag{19}$$

Thereafter, the residual, $\hat{\eta}_c(\hat{\boldsymbol{\sigma}}_{i,t}, m_{i,t}, \tau_{i,t}) = \sigma(m_{i,t}, \tau_{i,t}) - \hat{f}_c(\hat{\boldsymbol{\sigma}}_{i,t}, m_{i,t}, \tau_{i,t} | \hat{\boldsymbol{\theta}}_c))$, is then directly comparable to the $\hat{\nu}_{p,s}(m_{i,t}, \tau_{i,t})$ residual of the earlier approach, and thus the performance metrics are as well. In what follows, this approach will be referred to as either the combined or ensemble approach for the modelling of the IVS.

Now that the general methodology has been outlined, the following sections will cover the types of ML models that are used for our two-step estimation approaches. Given that depending on the prediction exercise, an ML model has to be trained up to once for every day of training data, this creates a desire for general-purpose, plug-and-play ML models with limited hyper-parameters. Furthermore, as the IVS and its derived PES can take a wide variety of complex shapes, the chosen ML models need to be able to accommodate and capture sufficiently complex relationships as well. Hence, Random Forests (RF), eXtreme Gradient Boosted trees (XGBoost) and Support Vector Regressions (SVR) will be explored in greater detail here. These are then subsequently compared with the NN-based approaches to see whether there is any benefit to be found in the additional complexity of the latter.

4.2.1 Random Forest

Originally proposed by Breiman (2001), Random Forests have become very popular ML models given their ease of use and general applicability. Unlike linear models, tree-based models are fully non-parametric and can easily capture non-linearities (Gu et al., 2020). In RFs, an ensemble is built from a D-sized set of decision trees (DT) that can be trained in parallel, each given a different bootstrapped sample of the training data. During the splitting of the nodes of the DTs, based upon the chosen impurity measure such as Gini or Entropy, a random subset of available features is used to determine the next optimal split that decreases the impurity as much as possible. The final predictions by the D DTs are subsequently aggregated through a certain voting scheme for classification or by averaging in case of regression (Breiman, 2001). In the latter, the DTs are often referred to as Regression Trees (RT). These two sources of randomness that are introduced serve the purpose of decreasing the variance of and promoting more diversity in the ensemble. This reduces the overfitting tendencies that DTs have, whilst only modestly increasing the bias and hence tries to positively affect the bias-variance trade-off that exists in ML (Breiman, 2001).

The implementation used here is the one available in the `Scikit-Learn` Python library (Pedregosa et al., 2011). The main hyperparameters of interest include the number of trees D, the max depth each individual tree is allowed to reach and the minimal number of samples required to split an internal node. Whilst the first controls the size of the ensemble, the latter two options control the pruning that is done on the individual RTs during training and hence manage their tendencies to overfit and their allowed complexity.

4.2.2 XGBoost

A different approach in which RTs play a key role can be found in gradient boosted regression trees. Here the main difference with the earlier RFs results from the sequential nature of the boosted trees, as RTs are no longer trained in parallel on the same dependent variable. The trees are now trained sequentially to correct the residuals of the earlier model, with the first being trained on the original dependent variable. Secondly, the introduction of randomness during the fitting of a single DT now becomes optional as well. The idea is that recursively combining forecasts made by relatively simple, shallow trees, i.e. weak learners, results in an overall strong learner with greater stability and a net positive effect on the bias-variance trade-off.

Chen and Guestrin (2016) formalize Gradient Tree Boosting for a data set with n instances and m features using D additive functions as follows

$$\hat{y}_i = \phi\left(\mathbf{x}_i\right) = \sum_{d=1}^{D} f_d\left(\mathbf{x}_i\right), \quad f_d \in \mathcal{F}, \tag{20}$$

where our prediction $\hat{y}_i$ for instance i is formed as the addition of the D sequential models f_d that have been learnt, where $\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}$ represents the space of RTs with RT structures q: $\mathbb{R}^m \rightarrow T$ and $w \in \mathbb{R}^T$. Each f_d corresponds to an independent tree structure q with leaf weights w and T leaves in total. As these trees are RTs, each leaf has a continuous score w_i on its i-th leaf. This allows us to define the following general, regularized objective function as

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l\left(\hat{y}_i, y_i\right) + \sum_{d=1}^{D} \Omega\left(f_d\right), \tag{21}$$

with $l(\hat{y}_i, y_i)$ a differentiable, convex loss function, $\mathcal{L}(\cdot)$ as the total loss and $\Omega(\cdot)$ as a regularization term that attempts to control the complexity of the RTs. Traditional optimization schemes can not be used to optimize Equation 21 due to the sequential manner in which f_d need to be trained, i.e. f_{i+1} depends on the output of f_i and hence requires f_i to be known. Therefore, let $\hat{y}_i^t$ be the prediction of the i-th instance at iteration t, we can then add f_t to the

objective function as

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(\mathbf{x}_i\right)\right) + \Omega\left(f_t\right), \tag{22}$$

where one can seek to greedily add the f_t that improves the model the most. Several algorithms for this type of boosting exist, including Adaptive Boosting (AdaBoost) and eXtreme Gradient Boosting (XGBoost), that each use their own specific loss function and approximation for the calculation of Equation 22 during optimization, as well as potential regularization terms (e.g. XGBoost uses $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$). This paper will explore XGBoost, given its recent rise in popularity and high computational performance (Chen & Guestrin, 2016). More implementation details are out-of-scope here, but in general, it has several features and parameters of interest including:

1. The number of estimators, D.
2. Max depth of the RTs.
3. A learning rate parameter η. After each boosting step, XGBoost gets the weights of the new features and shrinks them by η to make the boosting process more conservative.
4. The minimum loss reduction required to further partition a leaf node, γ.
5. A subsample ratio that controls the randomized training data sampling that is performed prior to each boosted iteration and tree grown.
6. A column subsample ratio that determines the number of features that are considered on various different levels: for the entire tree, per node split, and per depth level of the tree.
7. L2 regularisation term on the weights, λ.
8. L1 regularisation term on the weights, α.

The XGBoost models that are listed in the remainder of the paper were all implemented using the `XGBoost` Python library (Chen & Guestrin, 2016).[5]

4.2.3 Support Vector Regression

In order to move away from the tree-based models mentioned above, yet still allow for plenty of complexity and non-linear behaviour to be captured with manageable hyperparameter counts, we propose the use of Support Vector Regression (SVR) (Drucker et al., 1996). Originally developed as an extension to Support Vector Machines (SVM) (Boser et al., 1992; Cortes et al., 1995), SVRs have seen a long-term use in ML spheres given their main benefit of the kernel trick, which can easily capture non-linearities through the use of various kernel functions and handles large dimensional problems well. The SVR problem can be stated in different and equivalent ways and we will opt to represent it as what is known as the ε-SVR formulation followed by

[5] https://xgboost.ai/

Vapnik (1998) using the notation of the popular `LIBSVM` library (Chang & Lin, 2011) for which `Scikit-Learn` provides a Python wrapper (Pedregosa et al., 2011).

Consider a set of training points $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, where $\boldsymbol{x}_i \in \mathbb{R}^m$ are the m-dimensional feature inputs and $y_i \in \mathbb{R}$ represent the dependent variable. Given regularization, $C > 0$, and range, $\varepsilon > 0$, parameters the problem can be formulated as

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad & \frac{1}{2}\boldsymbol{w}'\boldsymbol{w} + C\sum_{i=1}^{n}\xi_i + C\sum_{i=1}^{n}\xi_i^* \\ \text{subject to} \quad & \boldsymbol{w}'\phi\left(\boldsymbol{x}_i\right) + b - y_i \leq \epsilon + \xi_i, \\ & y_i - \boldsymbol{w}'\phi\left(\boldsymbol{x}_i\right) - b \leq \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, n, \end{aligned} \tag{23}$$

where $\phi(\boldsymbol{x}_i)$ maps $\boldsymbol{x}_i$ into a higher-dimensional space. Moreover, the slack parameters ξ_i are introduced to deal with otherwise potentially infeasible problems (Chang & Lin, 2011; Vapnik, 1998). This problem statement can then be rephrased into its dual problem as

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & \frac{1}{2}\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right)' Q\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right) + \epsilon\sum_{i=1}^{n}\left(\alpha_i + \alpha_i^*\right) + \sum_{i=1}^{n} y_i\left(\alpha_i - \alpha_i^*\right) \\ \text{subject to} \quad & \boldsymbol{e}'\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right) = 0, \\ & 0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, \ldots, n, \end{aligned} \tag{24}$$

where $Q \in \mathbb{R}^{n \times n}$ is a positive semi-definite matrix with $Q_{ij} = K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \equiv \phi\left(\boldsymbol{x}_i\right)'\phi\left(\boldsymbol{x}_j\right)$ and $\boldsymbol{e} = (1, \ldots, 1)'$.

After solving Equation 24 by learning the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$, the approximating function becomes:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\left(-\alpha_i + \alpha_i^*\right) K\left(\boldsymbol{x}_i, \boldsymbol{x}\right) + b \tag{25}$$

The most important hyperparameters that must be tuned in this type of ML model include:

1. The kernel function to be used and its appropriate parameters.
2. C, the regularization parameter.
3. ε, the range parameter.

In terms of the kernel function, we will explore the polynomial, radial basis and sigmoid kernels,

for which we use $\langle \cdot, \cdot' \rangle$ as inner product notation, specified respectively as

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\gamma \left\langle \boldsymbol{x}_i, \boldsymbol{x}'_j \right\rangle + r\right)^d, \qquad \text{polynomial} \tag{26}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \left\| \boldsymbol{x}_i - \boldsymbol{x}'_j \right\|^2\right), \qquad \text{radial basis} \tag{27}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh\left(\gamma \left\langle \boldsymbol{x}_i, \boldsymbol{x}'_j \right\rangle + r\right), \qquad \text{sigmoid} \tag{28}$$

with γ as scale parameter, d to control the degree of the polynomial and r as a level parameter.

Lastly, when fitting SVRs during empirical experiments on large datasets consisting of years worth of data, rather than a model per day as is done initially, the dimensions of the problem, n, can get out of hand. Hence, if this problem arises, we follow the Nyström approximation proposed by Kumar et al. (2012) and implemented in `Scikit-learn`. This is a general method for low-rank approximations of kernels which it achieves by subsampling the data for the evaluation of the kernel.

4.2.4 Feedforward Neural Networks

The final type of model that will be discussed is Neural Networks (NN). Arguably the most powerful modelling technique available today that has theoretical underpinnings as a universal approximator. The flexibility NNs can boast is due to the complexity that can be introduced through many layers of subsequent non-linear predictor interactions. The flipside to this lies in the low interpretability and high parameterization (Gu et al., 2020). The specific types of NNs that will be considered here are known as Feedforward NN (FFNN) which consist of an input layer that takes in the raw feature values, one or more hidden layers that interact and non-linearly transform the predictors, and an output layer that aggregates the output of the last hidden layer into the final prediction as Gu et al. (2020) put it. These FFNNs are fully connected in the sense that all nodes or neurons on a particular layer, are connected to all the neurons of the previous layer. This implies that all the outputs of the previous layer (one for each neuron) are aggregated through a weighted sum in the current neuron and are subsequently sent through a (non)linear activation function to create the current neuron's output. Hence, the choice of the number of neurons and number of hidden layers, as well as the type of activation function that is used in each layer, constitute the most important decisions when determining the architecture of the NN.

In general, following the notation of Almeida et al. (2022), we denote $\mathbf{x}_{i,t} = (m_{i,t}, \tau_{i,t}) \in \mathbb{R}^2$ as the input vector of moneyness and TTM for option i on day t. For a NN architecture with L

hidden layers we can then define the network, $f\colon \mathbb{R}^2 \to \mathbb{R}$, iteratively starting from $\mathbf{z}_0 = \mathbf{x}_{i,t}$ as

$$\underset{d_l \times 1}{\mathbf{z}_l} = h(\underset{d_l \times d_{l-1}}{\mathbf{A}_{l-1}} \underset{d_{l-1} \times 1}{\mathbf{z}_{l-1}} + \underset{d_l \times 1}{\mathbf{b}_{l-1}}), \quad \text{for } l = 1, \cdots, L, \qquad \underset{1 \times 1}{f(\mathbf{x}_{i,t})} = \underset{1 \times d_L}{\mathbf{A}_L} \underset{d_L \times 1}{\mathbf{z}_L} + \underset{1 \times 1}{b_L}, \tag{29}$$

where $\mathbf{z}_0$ is simply the input layer, while $h(\cdot)$ represents the (non)linear activation function applied to the weighted sum of the d_{l-1} outputs contained in vector $\mathbf{z}_{l-1}$ of the previous hidden layer using the weight matrix, $\mathbf{A}_{l-1}$, and bias vector, $\mathbf{b}_{l-1}$.

As stated earlier, Almeida et al. (2022) follow the 5 fixed FFNN architectures outlined in Gu et al. (2020) which follow the geometric pyramid rule proposed by Masters (1993) combined with sigmoid activation functions: $h(x) = \text{sigmoid}(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$. The neurons per hidden layer, 1 up to 5, are defined as follows:

1. NN1: 1 hidden layer with 32 neurons.
2. NN2: 2 hidden layers with 32 and 16 neurons respectively.
3. NN3: 3 hidden layers with 32, 16 and 8 neurons respectively.
4. NN4: 4 hidden layers with 32, 16, 8 and 4 neurons respectively.
5. NN5: 5 hidden layers with 32, 16, 8, 4 and 2 neurons respectively.

Deeper NNs, i.e. networks with a larger number of hidden layers L, frequently offer better or equal performance, if properly regularized, with a lower number of parameters than wider networks that use fewer hidden layers but more neurons per layer (Eldan & Shamir, 2015). The findings of Almeida et al. (2022) seem to conclude that there is little benefit in going beyond 4 to 5 hidden layers, or thus deeper than their proposed NN4 and NN5 architectures. However, due to their use of the sigmoid activation functions, what is known as the vanishing gradient issue might play a role in these findings. As stated earlier, vanishing gradients refers to the fact that NNs showcase exponentially decreasing backpropagated error terms as a function of the distance to the output layer (Sussillo & Abbott, 2014). Hence, when not properly accounted for, this can prevent good convergence of deeper NNs. A common way to (partially) resolve this is through the use of different activation functions such as the Rectified Linear Unit (ReLU) (Sussillo & Abbott, 2014), which can be specified as $h(x) = \text{ReLU}(x) = \max(0, x)$ and is the predominant activation function that will be used in the remainder of the paper, unless specified otherwise.

Moreover, a plethora of regularization techniques exists for the successful training of NNs, whilst keeping the overfitting tendencies at bay. Among the most commonly found techniques you have, L1 and L2 regularization, max norm, batch normalization, dropout and early stopping to name a few.

L1 and L2 regularization simply involve adding additional terms to the loss function in

question, $\mathcal{L}(\cdot)$. Denoting $||\operatorname{vec}(\mathbf{A}_{l-1})||_i$ as the i-th norm of vectorized weight matrix $\mathbf{A}_{l-1}$, we define the regularized loss as

$$\mathcal{L}^*(\mathbf{X}) = \mathcal{L}(\mathbf{X}) + \sum_{l=1}^{L} \lambda_l ||\operatorname{vec}(\mathbf{A}_{l-1})||_2 + \sum_{l=1}^{L} \alpha_l ||\operatorname{vec}(\mathbf{A}_{l-1})||_1, \tag{30}$$

with λ_l and α_l controlling respectively the level of L2 and L1 regularization that is applied to the weights of hidden layer l. Note that normally no regularization is applied to the bias vector, as this can be interpreted as a type of level or intercept parameter.

In a similar vein, max norm places an absolute upper bound on the magnitude of the vectorized weight matrix, restricting it in such a way that the second norm can not exceed a user-defined value per hidden layer: $||\operatorname{vec}(A_l)||_2 \leq C_l$. This is often used in combination with dropout as they have been reported to work well together (Srivastava et al., 2014).

Alternatively, batch normalization (BN) was introduced by Ioffe and Szegedy (2015) as a way to improve the convergence speed of deeper NNs, as well as provide a modest amount of regularization. BN layers can essentially be added anywhere in the network, although Ioffe and Szegedy (2015) originally proposed to do so before the activation function. During training, the BN layer calculates the mean and variance of its inputs across the mini-batch, after which it uses the learnt mini-batch mean and variance to standardize the inputs. This standardized input is subsequently scaled and shifted and fed to the next layer. Hence, if we denote $\mathcal{B}$ as the mini-batch with size m, we calculate the mean and variance of the mini-batch, $\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i$ and $\sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu_{\mathcal{B}})^2$ respectively. Assuming that each input $\mathbf{x_i}$ is d dimensional, each dimension is normalized seperately as

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{2(k)} + \epsilon}}, \qquad \text{for } k = 1, \ldots, d, \quad i = 1, \ldots, m, \tag{31}$$

where ϵ is a small arbitrary constant added for numerical stability. The normalized vector $\hat{\mathbf{x}}^{(k)}$ with zero mean and unit variance (neglecting the influence of ϵ) then undergoes the final scaling and shifting step which yields the layer's output, $y_i^{(k)} = \gamma^{(k)}\hat{x}_i^{(k)} + \beta^{(k)}$, where $\gamma^{(k)}$ and $\beta^{(k)}$ are learnt as part of the optimization process. When appropriate, BN has been shown to greatly speed up convergence due to the possibility for higher learning rates and decreased sensitivity to initialization (Ioffe & Szegedy, 2015), whilst providing regularization tendencies.

Another regularization technique commonly found in NN literature is dropout. This involves adding dropout layers to the proposed NN architecture which randomly drop the output of several units of the previous layer, in accordance with a specified dropout rate, by setting them

to 0. The subset of units that is effectively dropped is randomized for every training iteration and for every training instance (Hinton et al., 2012; Srivastava et al., 2014). Afterwards, once the model has been fit, the dropout is no longer applied and the model estimates as if the dropout layers were not present. Although this tends to slow down the rate of convergence, it effectively forces the network to become more robust to noise. Furthermore, an interesting use case exists where one applies dropout to the high dimensional input as a way of incorporating random feature selection during the training of NNs.

Lastly, early stopping involves the use of a validation set during the training of the NN, on which the performance is evaluated at the end of each epoch. Once the performance has not improved for a certain number of epochs, P, called the patience, the training stops and the weights that resulted in the best validation performance can be reinstated. Afterwards, the network can be refit on the total training data, i.e. including the validation data, with the desired number of epochs set to the epoch that gave the best performance on the validation set (Prechelt, 2012; Yao et al., 2007). The reasoning is that after a certain number of epochs, the network starts to overfit more and more, which hurts the generalization error and hence one should refrain from going past this point. Early stopping is not just a NN regularization technique, as the idea can be applied to essentially any iteratively learnt method (Zhang & Yu, 2005).

Given the reasoning above, we investigate several different types of NN architectures, which follow a different design philosophy depending on the use case. A complete list of the different specifications used will be listed in the respective appendices when discussed during the result section. In general, we train the NNs proposed by Almeida et al. (2022) as a benchmark, to which we compare similar architectures will simple changes such as the use of ReLU activation function or the addition of batch normalization, as well as iteratively more complex models. With regards to the latter, we assess the benefits of increasing the complexity in terms of an increased number of neurons, which we will refer to as the collection of wider networks, as well as networks with an increased depth, which we will call the deeper networks. The choices of which exact architectures to include per set, as well as the defined regularization and optimization parameters are made based on an iterative approach. This involves exploring a much larger set of different configurations on a subset of the data to select what tends to work in terms of the validation performance, before running it on the entire dataset. As is usually done in NN tuning, this implies building networks that are complex enough to model the problem appropriately on the training data, before adding regularization to limit the overfitting and benefit the generalization capability of the model. This can largely be done by assessing the

learning curves in terms of proper convergence and the deviation between train and validation scores.

All networks are implemented in the high-level API named `Keras` of the `TensorFlow 2` library. The NNs are trained using the well-known Adam optimizer (Kingma & Ba, 2014) with varying learning rates depending on the empirical experiment and the used batch size as will be listed later, whilst the remainder of the Adam parameters remain on their default values. Lastly, for the networks using ReLU activation functions, we opt to use the He normal weight initializer as this has been shown to provide robust results when combined (He et al., 2015). While for the benchmarks with the sigmoid function we initialize using the Glorot uniform distribution (Glorot & Bengio, 2010).

4.3 Empirical Experiments

There exist several ways in which this proposed methodology can be applied in practice. One could calibrate and fit both the parametric and ML models on the daily cross-section to assess their interpolation performance. Alternatively, a more general approach involves calibrating parametric models on the daily cross-section while fitting ML models over longer data ranges in an attempt to better capture the general IVS as well as its time-variability. Or one may even calibrate the parametric models themselves over option data covering multiple days. Hence, the following paragraphs will discuss the several empirical exercises conducted in this paper in greater detail.

4.3.1 Daily Cross-Section

First, we follow Almeida et al. (2022) and conduct an exercise in the cross-section. Namely every day of the dataset, we divide the available cross-section of options in two based on the divisibility of the strike price. If K is divisible by 10, that option belongs to the training set, otherwise, it belongs to the out-of-sample (OOS) test set. Doing so results in 1,603,810 (61.19%) and 1,017,382 (38.81%) options for training and testing respectively, while the daily training and testing sets are guaranteed to consist of options spread across the entire IVS such that they remain representative. By then calibrating the parametric models and fitting the ML models every day on given set of training options, we can attempt to derive a consistent price for the daily test set of different option specifications and thus assess the interpolation performance.

However, as we intend to explore a wide array of models that each require certain hyperparameters to be tuned, an extension has to be made to avoid information leakage. As is common in ML literature, a grid search approach is deployed to automatically find the best set of hy-

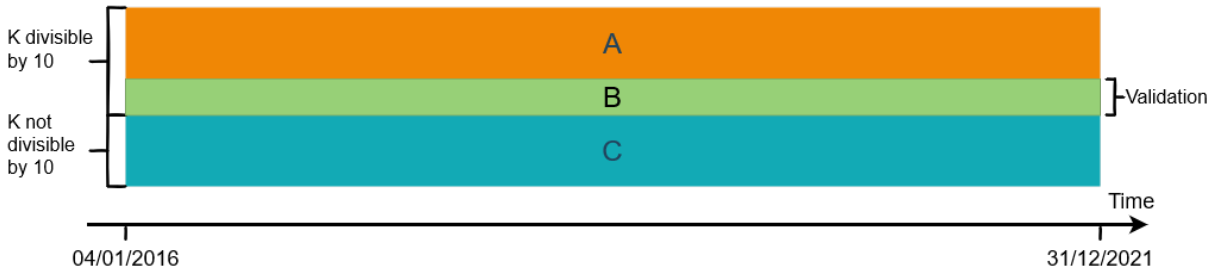


Figure 1: This diagram showcases the train-val-test split for the first cross-section exercise, where each set, A, B, and C, should be interpreted as consisting of multiple individual subsets, one for each trading day. The parametric models are calibrated daily on options with a strike, K, divisible by 10 represented as the daily subset of $A \cup B$. A secondary model is then tuned every day through a grid search using the residuals in the 80% of the daily training data, A, with the remaining 20% represented by set B as daily validation data, and is afterwards refit with the daily optimal parameter set on the entire daily cross-section of $A \cup B$. This model is then evaluated on the daily test set C.

perparameters for each model on a given day. The complexity of, mainly, the NNs makes a 5- or 10-fold cross-validation grid search infeasible, instead the more standard approach of a fixed validation and training set per day will be used to select hyperparameters. Furthermore, in order to maintain comparability across the models, these validation sets remain the same during all grid searches. The exact implementation involves selecting roughly 20% of the training data per day as validation data and keeping this fixed across the different models and grid searches. In order to achieve daily train and validation sets that cover all regions of the IVS and hence are representative, we opt to bin the data along the moneyness and TTM values into five bins each using equal frequency binning. This results in a 5×5 grid which forms the basis for a stratified sampling approach where we randomly select 20% of the data per stratum (a bin combination) as validation data that day.

Figure 1 gives a visual representation of the proposed train-val-test split. During the grid search for a particular model on a particular day, one model is fit on the remaining 80% of the daily training data (that day's subset of A) per combination of hyperparameters and the performance is evaluated on the validation data (that day's subset of B). Subsequently, the optimal hyperparameters for that model that day are selected based on the best validation performance measure. Afterwards, the model in question is refit using that set of hyperparameters on the entire training data that day (100%, $A \cup B$) and the performance can be assessed on the so far unseen daily test data in C. Note that the use of validation sets for tuning is one of the main differences with Almeida et al. (2022) for this exercise, as they only evaluate a fixed set of NNs architectures without any tuning, nor evaluate other types of models.

Overall this approach involves taking the subset of training options that day and calibrating our set of parametric models (BSM, AHBS, H, CW). Each one then results in a set of predicted IVs and hence residuals against the observed IVS. Subsequently, we grid search our secondary

models every day on these training residuals, following the procedure described above, and evaluate the final predicted IVS on the test set each day by correcting the predicted IVS of the parametric models with the predicted residuals.

4.3.2 Multiday Cross-Section Interpolation

A second exercise also involves daily calibrated parametric models using the same training, validation and testing data as during Subsection 4.3.1, yet differs in the way the secondary models are fit. We calibrate the set of parametric models on the training set of options every day that have a strike divisible by 10 and then have them predict on the test set of strikes that are not divisible by 10 to get the test set residuals. Subsequently, we now train our secondary models on the training residuals across all trading days at once. Afterwards, these secondary models are then used to predict on the testing data to assess their interpolation performance. Yet again, because of the need to tune our models, a validation set is used in a similar way as above to grid search the hyperparameters. For the sake of comparability, we use the same training, validation and testing data as during Subsection 4.3.1. However, this is now combined into a single training, validation and test set each, rather than a set per day. Hence, the main difference with the earlier experiment is that we now end up with a single, tuned ML model per model type for the entire time frame of the data, rather than a single ML model per model type per day as earlier. The visual representation of the approach remains the same as in Figure 1, but now sets A, B, and C, are one set each during the secondary modelling, instead of sets consisting of daily subsets.

The reasoning behind this experiment is that it allows us to use the full time frame of the data available to build secondary models that can hopefully learn the misspecification tendencies of the individual parametric models. Furthermore, fitting a single ML model over a longer time span than a day allows us to introduce time-varying covariates (Section 3), as well as the parameters of the calibrated parametric models. When using time-varying covariates here, or in the following experiments, we consider them to be state variables, which means that they are not to be forecasted themselves but are treated as a given each day.

4.3.3 Multiday Cross-Section Reusability

A third empirical experiment has a similar setup to the one described in Subsection 4.3.2, but now focuses more on the reusability of the models for interpolation purposes, rather than evaluating the model's performance on a hold-out test set during the same time window as the one it was trained on. The benefit here is that it allows us to compare how well the corrective

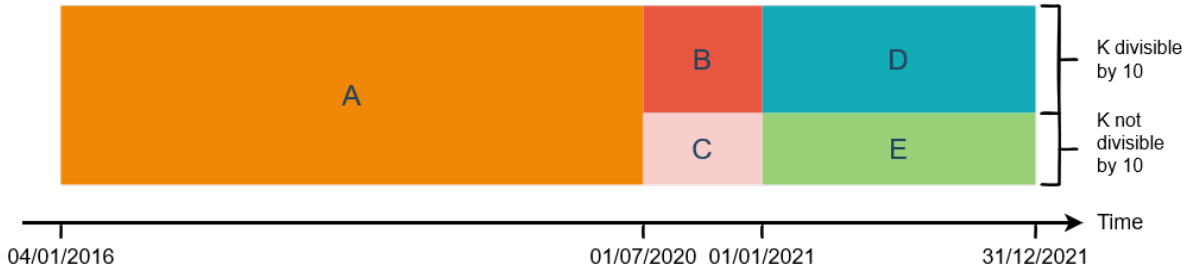


Figure 2: This diagram shows the train-val-test split for the third exercise. The parametric models are calibrated daily on all available options in the cross-section represented by set A. A secondary model is then tuned through a grid search where we train on all residuals in the training set A, while we assess their interpolation performance on validation set C. Here C contains residuals of parametric models calibrated daily only on B, with K divisible by 10. The optimal model is then refit on a new set of residuals in $A \cup B \cup C$ of parametric models recalibrated on the entire daily cross-sections, now including the non-divisible by 10 strikes in C. Lastly, the performance is evaluated on set E of parametric models calibrated on D.

models have learnt the misspecification tendencies of the parametric models in a general way such that they can be potentially reused over time, rather than having to be retrained at a daily frequency. We reserve an OOS test set towards the end of the data of one year in length spanning from 01/01/2021 until 31/12/2021. This is preceded by a slightly shorter validation set of six months from 01/07/2020 until 31/12/2020. This was chosen in such a way that the start of the COVID-19 pandemic still belongs to the training data, in order to have validation and test sets that are presumably representative of the data in the training set. During the training time window, parametric models are calibrated daily on the entire cross-section and their predictions give rise to the training residuals used for the secondary ML models as dependent variable. Again we use a similar grid search approach for their tuning.

Figure 2 gives a diagram of the exact sets used for training, validation and testing. More specifically, hyperparameters are selected by training the ML models on all training residuals of parametric models which were calibrated daily on the entire cross-section (set A), while we evaluate on the interpolation and validation set C of residuals by parametric models calibrated daily only on options with strike divisible by 10 in B. Afterwards, the optimal hyperparameter combination is used to refit the ML model on the residuals of parametric models calibrated on the entire daily cross-sections found in the combined training and validation data ($A \cup B \cup C$). Lastly, we evaluate the performance of the refit model on the test set E, which again contains residuals of parametric models only fit on strikes divisible by 10, set D.

The critical reader might see this as a case where the training and evaluation settings differ strongly, given that during training the residuals of parametric models are used that were calibrated on all options that day, whereas during testing and validation they are only calibrated on the divisible by 10 strikes to be able to assess the interpolation performance. One could argue that smaller daily cross-sections might lead to different misspecification tendencies

of the underlying parametric models, and hence, what the ML models learn during training might become less relevant during validation and testing. However, we do not find significant differences in the coefficients of the parametric models when calibrated on all options or solely on the divisible by 10 subset. This can largely be attributed to the way in which the options are subdivided, as doing so based on the divisibility of the strikes yields well spread out option specifications in terms of the IVS and thus maintains representability within the subsets. Furthermore, the number of option specifications on a given day does not remain constant, but differs over time and shows a positive trend (Figure A.2). Hence, if this were to lead to different misspecification tendencies, using residuals of parametric models solely calibrated on divisible by 10 strikes would not resolve the issue entirely. Moreover, doing so would leave out roughly 40% of the available data during training. Ergo, we assume that using all data improves ML performance more so than hurting it by this arguably slight difference in training and evaluation environment.

In the end what one ends up assessing is, if we were to fit an ML model on the residuals of daily parametric models up to a certain point (here 31/12/2020), how well can we expect that ML model to be able to predict residuals for interpolating consistent IVs for non-existent options today, given today's time-varying covariates and calibrated parametric models. Furthermore, we can evaluate how the performance deteriorates over time as we move further away from the training data by calculating cumulative performance measures. This is done by evaluating how the performance measure worsens over the subperiods starting on 01/01/2021, up to each and every day until the end of the test set, 31/12/2021.

4.4 Evaluation

In terms of performance evaluation, (root) mean squared error ((R)MSE) and mean absolute error (MAE) measures are clearly the most popular in literature as is shown by Ruf and Wang (2020). However, the dependent on which this is calculated differs from paper to paper, most either use the ratio of the predicted call price over the strike ($\frac{C}{K}$) or the IV. As mentioned earlier, the calibration of our parametric models is done based on the IVs directly. For the sake of consistency, we can then fit the secondary models using MSE-based performance measures on the IV residuals as well, which is in line with the earlier work by Almeida et al. (2022). Hence, as main evaluation criterion, we propose to use the RMSE of the overall IV prediction. Given an arbitrary set of n options with their observed IVs, σ, and predicted IVs, $\hat{\sigma}$, we define the

RMSE as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\sigma_i - \hat{\sigma}_i)^2} \tag{32}$$

This approach does, however, introduce a mild bias towards favouring the performance on more recent data, as there exists an increasing trend in the number of option specifications (Figure A.2) and thus there are more recent IV predictions, which a simple mean does not account for.

4.5 Feature Importance

Understanding why a model makes certain predictions is of utmost importance when assessing its quality and potential to be put into production, but is also of increasing concern given the ever-strengthening regulatory environment in which financial players operate. Several of the prediction exercises mentioned in Subsection 4.3 allow for the use of time-varying covariates which, given the wide variety of possibilities that exist, give reason to define a way in which we can evaluate their importance in terms of PES or IVS prediction. The nature of this paper involves the use of several different ML models classes and techniques which results in the need for model-agnostic feature importance measures that allow us to compare results not only between models with different sets of hyperparameters but also across model types. Below we argue the choice of our proposed feature importance measure succinctly, however, an elaboration on some key points is offered in Appendix C.

One of the two measures proposed by Gu et al. (2020) satisfies this model-agnostic criterion. They determine the reduction in the model's R^2 measure when setting all values of a particular predictor to zero while keeping the rest fixed. In line with this, Almeida et al. (2022) use a similar approach where the R^2 measure is replaced with their performance measure of choice, the RMSE of the IVs. These measures can be seen as simplifications of the more popular permutation-based approaches as originally implemented for RFs (Breiman, 2001) and later adopted by others and generalised to a model-agnostic version such as by Fisher et al. (2019). These latter methods permute the feature in question, rather than setting it to zero, to break the relationship with the remaining features (Appendix C.a).

However, in both cases, these approaches hold several severe disadvantages as discussed by Fisher et al. (2019). First of all, correlated features can bias and underestimate the feature importance scores (Appendix C.a). Secondly, permutation, but certainly setting feature values to zero, will often create unrealistic samples and hence has the model predict instances that would not occur in practice, which is not something the model should be able to do and therefore

its decrease in performance is not as relevant anymore.

Given the need for explainable AI (XAI) in model-agnostic settings and for very complex models such as deep NNs, the literature has progressed towards more advanced methods that deal with both the increase in complexity and some of the aforementioned issues. Two of the most promising techniques that are gaining traction within European regulatory bodies for approval (European Banking Authority, 2020, 2021; European Commission, 2021) are Local-Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016) and Shapley values (Lundberg & Lee, 2017). The former deploys local surrogate models that are trained to approximate predictions made by the underlying black-box model. LIME does this by generating a new dataset, through small, randomized perturbations of an original sample, on which a secondary, interpretable model is trained using the predictions made by the original black-box model as dependent variable (Ribeiro et al., 2016) (Appendix C.b). One of the main issues, however, is not particularly a shortcoming of the LIME technique itself, but rather the lack of theoretical underpinning as to why it provides sound explanations.

Hence, this paper will focus on the use of Shapley values, which find their origin in coalitional game theory and were first proposed by the Nobel prize-winner Lloyd Shapley (Shapley, 1953). In a coalition in which several players cooperate to obtain a benefit from the cooperation, certain players might contribute more to the coalition than others. Shapley values attempt to address how important each one of these players is to the overall cooperation and what payoff the player should fairly receive in accordance to their contribution (Shapley, 1953). Translating these ideas to feature importances involves seeing the features as players in a predictive game in which each one of the feature values contributes a certain amount to the end result of the prediction by the black-box model, f. Formally, we define a total set of players P, of which subsets of players, or thus features, are referred to as coalitions $S \subseteq P$, with $p = |P|$ the total number of players/features, which yields the Shapley value for the j^{th} player/feature, ϕ_j as

$$\phi_j(v) = \sum_{S \subseteq \{1,\dots,p\} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left(v\left(S \cup \{j\}\right) - v(S)\right), \tag{33}$$

by summing over all possible coalitions of players, S, that do not contain feature j. Furthermore a value function, v, is defined as a characteristic function that maps subsets of players to real numbers, v: $2^p \to \mathbb{R}$, satisfying the constraint $v(\emptyset) = 0$. Which, according to Shapley (1953), can be interpreted such that $v(S)$, for S a coalition of players, represents the total worth of coalition S, which is the total expected sum of payoffs the members of S can obtain by cooperation.

The Shapley values have many desirable properties, of which efficiency, symmetry and additivity, as defined in Appendix C.c, are used to state that they do indeed represent fair payouts (Shapley, 1953). Furthermore, it can be shown that they are the only distribution that has those properties and hence is the only fair distribution satisfying efficiency, symmetry and additivity. Next to this guarantee, Shapley values also boast many other advantages. Such a key benefit is its theoretical underpinnings in game theory, which proves a necessity in regulatory settings. Furthermore, it allows for contrastive explanations where one compares a prediction to a subset of other predictions or even a single data point, rather than the average as is common in other explanation methods. However, Shapley values require a lot of compute as there are 2^p possible coalitions of feature values, where the features deemed absent have to be dealt with through simulation to still allow prediction by the black-box model. The potential simulation of unrealistic data instances and lack of applicability in sparse feature sets are other disadvantages of the method. Moreover, it should be clear that this theoretical framework relies upon the definition of the characteristic function v, which in practice, is far from straightforward.

In recent years, several methods have been proposed for the estimation of these Shapley values through the assumption of additivity of the characteristic function, which are sometimes referred to as additive feature attribution methods in the literature (Datta et al., 2016; Lipovetsky & Conklin, 2001; Lundberg & Lee, 2017; Štrumbelj & Kononenko, 2014). Perhaps the most widely used technique is known as SHapley Additive exPlanations (SHAP) by Lundberg and Lee (2017). SHAP combines ideas of LIME and the theoretical concept of Shapley values to generate both local as well as global feature interpretation in a more efficient manner. Following their notation, we define a linear, surrogate model, g, also known as the explanation model as

$$g(\mathbf{z}) = \phi_0 + \sum_{j=1}^{p} \phi_j z_j, \tag{34}$$

with $\mathbf{z} \in \{0,1\}^p$ as the binary, coalition vector with maximum size $p = |P|$, which indicates when a particular feature j is present and hence gives a contribution of $\phi_j \in \mathbb{R}$. When assessing a particular data instance of interest, $\mathbf{x}$, its coalition vector $\mathbf{z}$ becomes a vector of ones and hence the linear model simplifies to

$$g(\mathbf{x}) = \phi_0 + \sum_{j=1}^{p} \phi_j \tag{35}$$

They go on to show that this formulation satisfies similar properties as the ones listed above, namely Local Accuracy, Missingness and Consistency, that allow them to prove in a similar

manner that there is only one possible additive feature attribution method for a particular input $\mathbf{x}$. Their initial implementation that follows the linear explanation model above was coined as Kernel SHAP, which they themselves see as a linear LIME model on Shapley values to locally approximate the explanation function. A full discussion of the exact implementation of Kernel SHAP is out-of-scope here but can be found in the original paper, Lundberg and Lee (2017), while their tree-based extension known as Tree SHAP, which can vastly reduce the computational complexity, was introduced in Lundberg et al. (2020).

However, its main advantages should be clear, including that it connects the widely used LIME to Shapley values that have a solid theoretical foundation. A noteworthy disadvantage of Kernel SHAP is that it continues to rely on sampling from the marginal distribution for individual feature values which does not resolve the issue of correlated features. However, this was tackled by Tree SHAP as there the authors model the conditional expected prediction rather than the marginal and hence it can capture interaction effects correctly (Lundberg et al., 2020).

When we report SHAP values and some of their extensive visualisations in the following sections, these were generated following the Tree SHAP methodology using the Python library `SHAP` maintained by the original authors themselves.[6]

5 Empirical Results

The paragraphs below will cover the results following the methodology outlined above for the empirical experiments: Daily Cross-Section (5.1), Multiday Cross-Section Interpolation (5.2) and Multiday Cross-Section Reusability (5.3).

5.1 Results Daily Cross-Section

The first daily cross-section exercise, as stated earlier, involves calibrating parametric models on a daily basis and similarly fitting the secondary models on their residuals on a daily basis as well. The results of doing so, with the train-val-test split and grid search approach described in Subsection 4.3.1, gives the RMSE values reported in Table 2. The first four rows follow the standard approach outlined in Subsection 4.2 where we use the TTM and moneyness as variables for the ML models to predict the respective parametric model's PES. While the last row reports the results of the combined method (Comb.), which uses moneyness, TTM as well as the IV predictions of the 4 parametric models as features and predicts the overall IVS. Furthermore,

[6] https://github.com/slundberg/shap

Table 2: Test set RMSE (%) of the IVS for the Daily Cross-Section exercise.

ML Param.	Param.	SVR	RF	XGB	SA NN	RA NN	BA NN	W NN	D NN
BS	8.855	**0.176**	0.293	0.238	2.026	1.013	0.868	7.390	6.437
AHBS	1.369	0.173	0.165	**0.149**	1.286	1.126	0.702	1.526	1.295
H	1.247	0.959	**0.687**	0.708	1.299	1.173	0.915	4.682	1.375
CW	1.803	0.169	0.150	**0.137**	1.350	1.117	0.556	2.470	1.487
COMB		**0.221**	0.355	0.276	0.905	0.400	0.681	9.196	5.646

Each row represents a certain underlying parametric model whose residuals were used to fit the models listed per column. Where the Param. column refers to using the parametric model as is, without error correction as benchmark. While SVR, RF, XGB, SA NN, RA NN, BA NN, W NN and D NN refer to respectively Support Vector Regression, Random Forest, XGBoost, sigmoid and ReLU Almeida et al. (2022) NNs, batch normalized ReLU Almeida et al. (2022) NNs and Wider and Deeper NNs. Lastly, comb. follows the ensemble approach where we predict the IVS directly. The test set spans the entire time frame from 04/01/2016 to 31/12/2021. Bold values indicate the best result per underlying parametric model (per row), while the underlined value is the best overall.

the exact sets of hyperparameters that were included in each grid search per model can be found in Appendix D.

As will be the case in later experiments as well, the explored sets of hyperparameters were chosen based on preliminary results for each model class ran on a wider set of hyperparameters, but on a much smaller subset of data to limit the computation time. For instance, SVR with different kernel functions such as the polynomial or sigmoid functions consistently gave worse results in all preliminary tests, and hence only the radial basis function kernel was used on the entire dataset. With regards to the NNs, we explore the architectures defined by Almeida et al. (2022) using the sigmoid activation function (SA NN), as well as with the ReLU function (RA NN) as our proposed way to avoid vanishing gradients, followed by the same architectures augmented with batch normalization layers (BA NN). Furthermore, we explore increased complexity by either focusing on increased width (W NN) or depth (D NN). Different levels of regularization are applied depending on the exact architecture. Where the values for max norm and the dropout rates when applicable were again chosen based on preliminary test runs. The exact configuration of each NN architecture can be found in Appendix N.

Table 2 showcases a dominance of the simpler ML models over the more complex and heavily parameterized NNs. The overall best performance is given by the XGBoost models that correct the CW model's residuals. Whereas, the largest relative improvement is to be found for the SVR-type models on the BS's residuals. Note, however, that as the BS models give essentially a flat IVS prediction on a given day, the models that correct these residuals can be interpreted as if they are modelling the IVS directly, up to a constant given by the BS flat prediction. Overall, there does not seem to be a clear winner between SVR, RF and XGBoost models, as they all deliver largely the same performance with arguably rather modest differences

depending on the configuration. On a row-by-row basis, the CW followed by AHBS parametric models seemingly are the easiest to correct. Meanwhile, the Heston-based models showcase less sizeable improvements. Surprisingly, the RMSE of these secondary models trained on the Heston residuals is consistently higher than those of the BS corrected models. This seems to indicate that flat-out modelling the IVS is an easier task for these ML models than correcting the Heston PES.

The distributions of the selected hyperparameters across the different days of the dataset during the grid search are depicted in the figures of Appendix E. For the RFs we find the common result that as soon as a sufficiently high number of estimators is selected, the differences in performance end up being modest. However, for the max depth, there is a clear preference for the largest possible value we provided, 12. This seems to indicate that quite complex individual trees are desirable and increasing the allowed values even further might pose to be interesting. The XGBoost models, on the other hand, tend to max out their number of estimators at 200, whereas the lowest η is never selected, and the subsampling of the data seems preferable. Lastly, for the SVR models, there seems to be less of a common trend, except for a preference for high gamma values, especially for the Heston correction.

With regards to the more complex NN approaches, none of the explored sets of architectures and parametric model combinations provide better performance than the other ML models. The replacement of the sigmoid activation function by the ReLU for the Almeida et al. (2022) architectures performs consistently better. Furthermore, reducing the batch size but including a batch normalization layer succeeding the first hidden layer yields the best performance out of the different NN sets due to the improved convergence and reduced sensitivity to the initialized weights, except for the combined approach. With regards to both the wider and deeper models, the increased complexity results in troublesome convergence and thus subpar results. Not only the test scores but also the train scores are consistently worse than the much simpler Almeida et al. (2022) based architectures, whether batch normalized or not, which shows their tendency to get stuck in local optima. Hence, across the NN sets there exists a preference for simplicity when fit on the daily cross-sections, which we deem to be due to relatively limited availability of data on a given day giving rise to a preference for more parsimonious models overall. Appendix E showcases the distributions of the chosen NN architecture for each NN architecture set.

The proposed combination of parametric model IVs in order to predict the overall IVS rather than the PES per model does not result in improved performance during the daily cross-section case. We hypothesise this might again be due to the vastly increased parameterization that this causes for all secondary models, whilst the available data on a given day remains limited.

Lastly, these results are directly comparable to the ones found by Almeida et al. (2022) as the empirical setup was chosen to be largely identical. The key difference to consider is the use of a more recent dataset here, which does include the volatile COVID-19 period and is hence arguably more difficult to model. Nonetheless, the performance of the parametric models by themselves is within the same orders of magnitude. Additionally, they also show the same ranking. In terms of the correction offered by the NNs, we find a very different result. Although we do see sizeable improvements over the base parametric models, these are nowhere near as large as the ones shown in Almeida et al. (2022), nor do they come close to the consistent and large-scale improvements given by our other classes of ML models, which boast RMSEs more in line with their NN results. Lastly, regarding which parametric model to use as underlying, Almeida et al. (2022) find the best results for the Heston model, with CW and AHBS being very close follow-ups. We, however, find poor performance for the Heston-based correctors, whereas CW, followed closely by AHBS, seems to yield the greatest results.

5.2 Results Multiday Cross-Section Interpolation

Table 3 reports the results achieved following the approach outlined in Subsection 5.2, where the grid searches were run using the hyperparameters listed in Appendix G. Note that in certain cases we allow for more complexity such as larger max depth and number of estimator values for RF and XGB based on the results of the previous exercise. In a similar vein, we omit the sigmoid versions of the Almeida et al. (2022) architectures given their relatively poor performance in the previous exercise, as well as during preliminary testing here, compared to the ReLU-based implementations. Different sets of features are explored to assess their added value, starting with the so-called 'base' features which correspond to just moneyness and TTM. The inclusion of this base set, is mostly as a benchmark, as it seems unlikely that given the wide array of different and complex shapes that the IVS and hence the PES can take, simply using a set of moneyness and TTM could produce consistent results over time. The time-varying covariates which we include are:

1. 3-Month Treasury constant maturity minus Federal Funds Rate (rate_spread)
2. The implied dividend yield of the S&P 500 (div_yield)
3. Difference between the daily close and open price of the VIX (VIX_close_open)
4. Difference between the daily high and low of the VIX (VIX_high_low)
5. Realized Variance of the S&P 500 (realized_var)
6. Bipower Variation of the S&P 500 (bipower_var)
7. Spot Volatility index by Todorov (2019) (SVI)

which are referred to with the naming scheme 'tv', which stands for time-varying. Note that we still include moneyness and TTM in this and all other sets of features as well. These can then subsequently be augmented with the calibrated parameters of the underlying model referred to as 'tv_op', which stands for time-varying with its own parameters. Alternatively, we can augment the time-varying set with the calibrated parameters of all parametric models that have them, i.e. AHBS, H, and CW, resulting in feature set 'tv_ap', or thus, time-varying with all parameters. Note that no parameters of BS are included here, as the only parameter that requires calibration is the BS IV, which is essentially already included as the intercept of AHBS. In the interest of computational time, the 'tv' set is only run for the BS-based residual models, as there is no relevant 'tv_op' set in this case. While for the others we skip running the models on solely the 'tv' set and immediately assess the 'tv_op' set to save compute in line with our hypothesis that the calibrated parameters are very valuable when attempting to learn the misspecification tendencies of the underlying parametric model. Lastly, for the combined approach, comb_base_ap refers to moneyness, TTM, all parametric model predictions and calibrated parameters on the one hand. While, on the other hand, comb_tv_ap refers to the same features set but augmented with the time-varying covariates listed above.

Similar to the earlier experiment above, we again see that XGBoost models perform very well, and in this setting even dominate. The only cases for which they do not deliver the best overall performance, are the configurations in which only the base features are used, i.e. only moneyness and TTM. In all other settings, where time-varying covariates, as well as parametric model parameters, are included, the XGBoost models deliver the best performance. In general, we again find that relatively simple RF models perform quite adequately compared to the best-performing XGBoost models. Their performance is quite similar and they outperform SVR in most cases. The latter now shows quite a subpar performance compared to the other models. This leads us to state that XGBoost and RF are more appropriate for modelling PES behaviour over time, whereas the radial basis function used as kernel for SVR does not seem to capture the time variation as well. Although, it has to be noted that in all cases there still exists some improvement over just the parametric model by itself.

If we look at the table in Appendix H, which showcases the selected hyperparameters per model class for each combination of feature set and parametric model, we see a clear preference for the most complex models possible. Given the explored hyperparameters listed in Appendix G, we notice that most RF and XGBoost models end up maxing out the number of estimators. Additionally, the RFs also prefer the largest allowed max depth of 24, whereas for the XGB models there's a spread between 12 and 24 augmented with a regularization parameter η of

Table 3: Test set RMSE (%) of the IVS for the Multiday Cross-Section Interpolation exercise.

Param.	ML Features	Param.	SVR	RF	XGB	RA NN	BA NN	W NN	D NN
BS	base	8.855	1.691	2.012	2.005	**1.655**	1.664	1.682	1.685
	tv		1.467	0.345	**0.153**	0.612	0.677	0.365	0.307
	tv_ap		1.253	0.239	**0.137**	0.352	0.375	0.220	0.263
AHBS	base	1.369	0.886	1.024	1.098	0.894	0.894	**0.871**	0.889
	tv_op		0.921	0.141	**0.095**	0.354	0.344	0.102	0.170
	tv_ap		0.885	0.132	**0.095**	0.319	0.316	0.129	0.213
H	base	1.247	1.087	1.086	1.198	1.066	1.071	1.062	**1.059**
	tv_op		1.013	0.640	**0.637**	0.765	0.799	0.704	0.718
	tv_ap		1.030	0.732	**0.650**	0.771	0.742	0.664	0.728
CW	base	1.803	1.307	**1.196**	1.479	1.228	1.245	1.204	1.278
	tv_op		1.140	0.132	**0.097**	0.367	0.366	0.108	0.201
	tv_ap		1.372	0.127	**0.096**	0.332	0.342	0.140	0.196
COMB	comb_base_ap		0.563	0.417	**0.164**	0.293	0.346	0.233	0.337
	comb_tv_ap		0.566	0.415	**0.160**	0.293	0.354	0.218	0.208

Each row represents a combination of a feature set and an underlying parametric model, whose residuals were used to fit the models listed per column. Where the Param. column refers to using the parametric model as is, without error correction. While SVR, RF, XGB, RA NN, BA NN, W NN and D NN refer to respectively Support Vector Regression, Random Forest, XGBoost, ReLU Almeida et al. (2022) NNs, batch normalized ReLU Almeida et al. (2022) NNs and Wider and Deeper NNs. Lastly, comb. follows the ensemble approach where we predict the IVS directly. Base features refer to only moneyness and TTM, while tv, tv_op and tv_ap are respectively the collections of only the time-varying covariates (including the base features), the time-varying covariates with the parametric model's own parameters and the time-varying covariates with the calibrated parameters of all parametric models. Next, comb_base_ap is a feature set consisting of moneyness, TTM, all parametric model predictions, and all parametric model parameters, whereas comb_tv_ap augments this with the time-varying covariates. The test set spans the entire time frame from 04/01/2016 to 31/12/2021. Bold values indicate the best result per underlying parametric model and feature combination (per row), while the underlined value is the best overall.

either 0.1 or 0.3. For SVR, the only consistency is to be found in the choice of gamma, being one of the 2 smallest allowed values.

Furthermore, in terms of the NN-based approaches, we now find the opposite result as earlier in terms of the Almeida et al. (2022) architectures (Appendices O.a and O.b) compared to ones with increased complexity. Both the wider and deeper architecture sets (Appendices O.c and O.d) consistently offer better performance than the Almeida et al. (2022) sets, either batch normalized or not, except for a few cases where solely the base features are used. Hence, the increase in available data during training, compared to the earlier exercise, facilitates the use of more complex NNs and allows their stronger parameterization to shine. Except in two settings, the wider networks showcase better performance that the set offering more depth, although in general, the performance of both sets is rather similar. Appendix H shows the elected NN architecture per set based on the grid search results. with regards to the Almeida et al. (2022) style architectures, we again find that three or four hidden layers seem to be largely sufficient. For the wider networks, we see a general preference for W NN4, which is arguably the most complex model tested here, as it has not only a much wider base of 1024 neurons on the first

hidden layer, but it also goes as deep as 8 hidden layers. While for the deeper, but more linear models, we find a less clear preference, although D NN2 seems to be the most popular. In general, though, the wider approach seems to be more beneficial and is rarely beat by the deeper networks, and when it happens the difference is modest at best. Our preliminary testing approach for selecting regularization parameters here often resulted in only a modest amount of regularization being sufficient as the training and validation scores tended to be very similar. Hence, a max norm of 1, combined with early stopping and batch normalization was plentiful to achieve training and validation scores that were in line with one another (see Appendix O for exact specifications). Moreover, dropout and L1 or L2 regularization, even at low rates, resulted in both lower training and validation scores and was henceforth left out.

Both for the tree and NN-based models, we see similar tendencies in terms of performance differences across different feature and parametric model sets. Our overall best performer is now the XGB model based on AHBS residuals with tv_op features, although very closely followed by the XGB models on CW residuals. Moreover, this very close performance of the AHBS and CW underlying models with either tv_op or tv_ap seems to be the same for the different ML model types as well, where either one of those two parametric models results in the best performance per ML class (neglecting the combined approach). The added value of the calibrated parameters of all parametric models over just the ones of the underlying parametric model seems dubious, i.e. tv_ap over tv_op. Only in certain cases does it improve performance, and even when this occurs the increase is quite limited. This seems to indicate that there is little correlation between how different parametric models end up being misspecified given a certain set of options, at least not enough that the ML models can exploit it in a meaningful way.

Again we find rather poor performance for the Heston-based approaches. The best performance per feature set is rather meagre compared to the others, both in absolute as well as relative terms. This is especially true when we consider the results of the BS-based approaches, which, again, can be seen as directly modelling the IVS up to a constant provided by the calibrated BS model that day. Hence, the Heston PES appears vastly more difficult to predict than the IVS itself. We hypothesise this might have something to do with the difficult calibration of Heston models, rather than the two-step estimation approach promoted here. Especially given that the Heston-based ML models still improve upon the Heston model by itself under all circumstances.

When inspecting the combined approach, we again find no presumed benefit to this way of modelling for the tree-based models, RF and XGB, whereas, for SVR and the ReLU Almeida et al. (2022) NN architectures, this gives us the best performance within those model types.

However, as in the latter cases, the RMSE scores are still substantially larger than those of the overall best-performing tree-based models, we deem this way of modelling rather undesirable.

Given that the same test set is used here as for the results in Subsection 5.1, we can see a noticeable improvement in the RMSE by using a singular model fit over a larger time frame with time-varying covariates, rather than using secondary models trained on the daily. For all underlying parametric models as well as the combined approach, the best performing models on the longer time frame here boast lower RMSE scores than their equivalents of Table 2. However, this is only the case when time-varying covariates are actually included, solely using moneyness and TTM (base features) is not sufficient to outperform daily fitted models, which is in line with the hypothesis that there exists too much variation over time for the base features to appropriately model the PES during a longer time frame.

In order to assess the feature importance scores, we extract the SHAP values using the methodology outlined in Subsection 4.5. Given that we have many different models here, for the sake of brevity, we will mostly focus on the feature importance of the best performing one, the XGB model on AHBS residuals with the tv_op feature set. However, the results remain largely the same for the other models as well. Figure 3 showcases the mean, absolute SHAP values per feature, which can be interpreted as the global importance of a particular feature. Moneyness and TTM are the most important features by far, followed by the Spot Volatility Index of Todorov (2019). Next, we find the squared moneyness coefficient of the AHBS model. Afterwards, the difference in feature importances becomes negligible, as the importance of the remaining AHBS coefficients, interest rate spread, dividend yield and the other variance measures is largely the same.

Figure 5 shows how the residuals of the parametric AHBS model itself are distributed in terms of moneyness and TTM. This reflects the misspecification present in the AHBS-type models, which clearly depicts how the residuals become larger as the option specifications become more extreme in terms of both sides of the moneyness and TTM scales. This is particularly interesting when compared to Figure 4 that depicts how the most important features, moneyness and TTM, contribute, measured as the SHAP value, to the prediction of our best performing XGB model on these AHBS residuals. Both relationships are clearly non-linear. Panel 4a shows that for the lowest moneyness values, i.e. deep OTM calls, we find a very strong positive contribution to the residual prediction. As the moneyness increases to ATM values, we almost linearly shift towards quite strong negative contributions, up to a SHAP of around -2 for ATM options with short TTM, or thus close to expiry ATM options. Whereas, options with a larger TTM, but also ATM, have a contribution much closer to 0. This seems to indicate that the

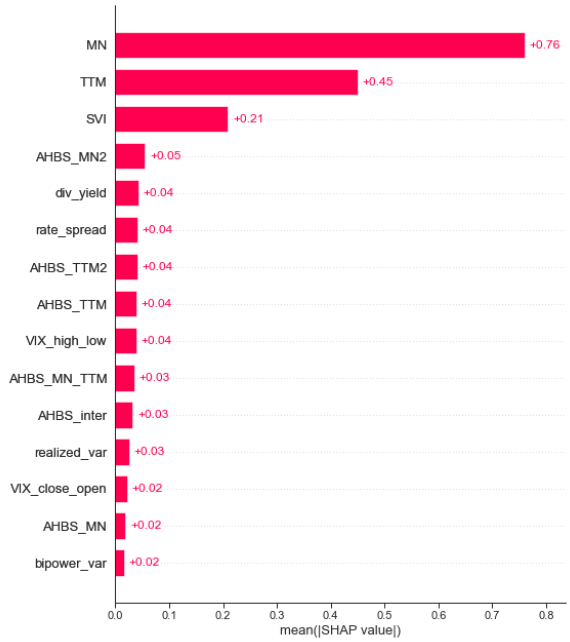
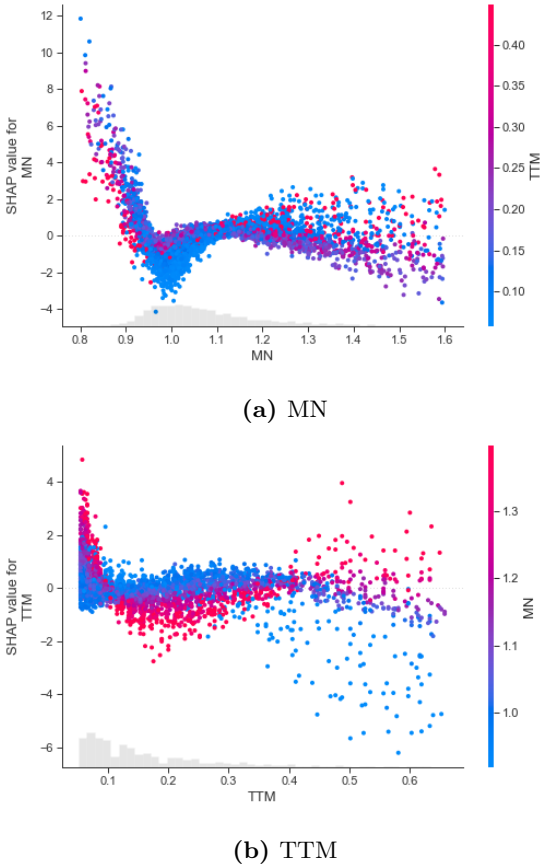


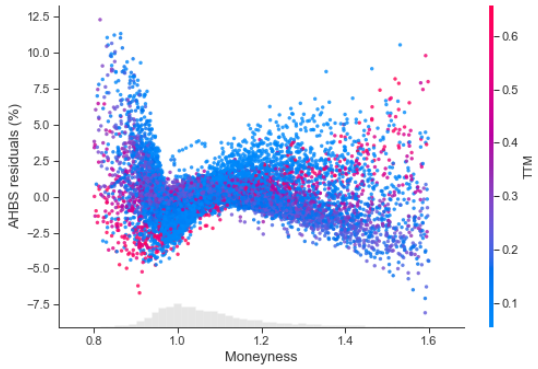
Figure 3: Bar plot of mean, absolute SHAP values (in %) per feature for the best performing model, XGB on AHBS with tv_op features.

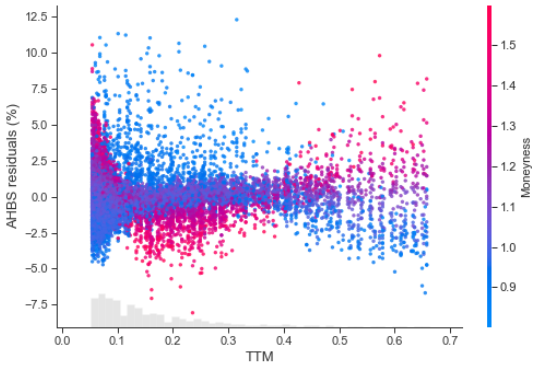


(a) MN

(b) TTM

Figure 4: Scatter plot of moneyness or TTM against SHAP values (in %), coloured by respectively TTM and moneyness for the best performing model, XGB on AHBS with tv_op features.





(a) AHBS moneyness

(b) AHBS TTM

Figure 5: Scatter plots of AHBS residuals along moneyness and TTM, coloured by respectively TTM and moneyness. Note that a subset of the data spanning the entire time frame is used for the sake of clarity.

XGB model corrects the predictions made by the calibrated AHBS model more heavily for these very deep OTM calls, as well as for the close-to-expiry ATM options. As the moneyness increases further, we find a negative parabolic relationship, where the differences in terms of TTM become less clear. Furthermore, compared to Panel 5a we find roughly the same shape with positive SHAP contributions corresponding with positive residuals and vice versa. The main difference lies in the tighter spread of the SHAP values compared to the raw residuals. This reduced spread can largely be explained by the fact that only two features of the XGB model are depicted here, whereas the final prediction is the combination of the total tv_op set, where the other features are used to capture some of the remaining variance.

Panels 4b and 5b show the same idea but for TTM coloured in accordance to the moneyness values. Again we find a non-linear shape for the SHAP values of 4b, where we start with strong positive contributions for the lowest TTM values, which decrease as we move to slightly longer-dated maturities. Interestingly enough, this mainly holds for the deep OTM puts, shown by options with a high moneyness value, where we start with a much stronger positive contribution and shift again to stronger negative contributions than their lower moneyness counterparts. Whereas the latter tend to be more centred around the 0 contribution line initially. These findings again indicate the relatively poor performance of the AHBS model for the more extreme contract specifications, particularly deep OTM puts, for which the XGBoost clearly has to provide stronger corrections. As the TTM increases, we find a somewhat negative parabolic shape for the lower moneyness values, whereas this seemingly increases linearly for the deeper OTM puts. Compared to the actual residuals shown in Panel 5b we find the same conclusion as earlier, being that the overall relationship looks very similar, but differs in the smaller spread of the SHAP values compared to the residuals. Hence, considering the great performance of this XGB model, it clearly tends to correctly use the TTM values to determine if they should have a positive or negative contribution to the overall predicted residual when accounting for the remaining feature values as well.

Lastly, Appendices J and K repeat this exercise for some of the remaining models. The former shows the misspecification tendencies of all parametric models, whereas the latter shows the SHAP relationships for the most similar XGB model to the best performing one above, for each underlying parametric model. Panels J.1a and J.1b show that the flat IV predictions of BS are highly problematic, especially for deep OTM puts and short-dated options. Panel J.1e showcases an abrupt change around the ATM moneyness for the Heston residuals, which is accurately corrected by the relevant XGB model shown in Panel K.1e. Here, deep OTM puts with short TTMs again represent some of the most troublesome option specifications for

these Heston models. Lastly, the CW residuals of Panel J.1g and J.1h are largely similar to the ones of the AHBS models, with as main exception that the CW models now not only tend to underestimate but also overestimate the deepest OTM calls.

In general, these misspecification tendencies in terms of moneyness and TTM shown in Appendix J and the contributions scores for the similar XGBoost models depicted in Appendix K, yield largely the same outcomes as before. Namely, the general relationship between the parametric model's residuals and the moneyness and TTM values are well-captured by the XGB models as shown by their respective SHAP contributions and good RMSE performance. Again, with a smaller spread as some of the remaining variance is captured through the omitted features. The main exception to this similar shape lies in the TTM plots for the BS models, Panels J.1b and K.1b. The former shows a distribution that is largely overlapping with how the observed IV values are actually spread in terms of TTM (Figure A.6) as the BS prediction on a given day is simply flat. The latter, however, shows that the XGBoost model utilizes the correlation that exists with the remaining time-varying and parametric model coefficient features to yield the relation shown by Panel K.1b. This becomes particularly clear once we denote that it now more closely resembles the corrective tendencies of the other XGB models in terms of TTM (Panels J.1d, J.1f, J.1h).

5.3 Results Multiday Cross-Section Reusability

In order to assess the reusability of fitting a single ML model on a longer window of data, we conduct the research explained in Subsection 4.3.3. Given the strong similarity between this empirical experiment and the one in Subsection 5.2, we opt to use similar configurations when possible to aid comparability. In terms of the grid search, this implies using the same hyperparameters, hence the ones listed in Appendix G. Moreover, the same feature sets are used, as well as the same naming schemes for the models.

Table 4 shows that moving towards an interpolation exercise where the OOS set lies in the future, compared to during the same time window as in Subsection 5.2, reduces the performance of the models substantially. However, we can still consistently outperform the parametric models by themselves. Again, we find that the XGBoost models seem to give the best performance in most setups. The main exceptions are the configurations that only use the base features, as well as two cases where the RF models provide marginally better performance. Moreover, in most cases, the performance of the RF models is very similar to that of the XGB models. Meanwhile, SVR-based approaches show a similarly poor performance to what we had in the previous empirical experiment, although they still always outperform the parametric models.

Table 4: Test set RMSE (%) of the IVS for the Multiday Cross-Section Reuseability exercise.

Param.	ML features	Param.	SVR	RF	XGB	RA NN	BA NN	W NN	D NN
BS	base	9.691	3.091	1.614	1.626	1.620	1.587	1.660	**1.564**
	tv		1.905	**1.480**	1.490	1.883	1.680	1.609	1.710
	tv_ap		1.746	1.134	**0.998**	1.883	1.680	1.609	1.710
AHBS	base	1.464	0.783	0.763	0.760	0.811	0.764	**0.750**	0.766
	tv_op		1.214	0.602	**0.574**	1.627	0.796	0.821	0.875
	tv_ap		1.301	**0.500**	0.502	0.739	0.753	0.892	0.947
H	base	1.172	**0.908**	0.954	1.027	0.949	0.925	0.925	0.942
	tv_op		1.024	0.917	**0.907**	1.050	1.002	1.704	1.204
	tv_ap		1.014	0.914	**0.880**	1.357	1.410	1.362	1.583
CW	base	1.656	1.424	0.906	0.898	0.969	**0.893**	0.919	0.925
	tv_op		1.166	0.686	**0.641**	0.788	0.783	0.753	1.020
	tv_ap		1.240	0.632	**0.606**	0.827	0.870	0.877	1.109
COMB	comb_base_ap		0.747	0.548	**0.493**	0.755	0.699	1.565	1.583
	comb_tv_ap		1.314	0.550	**0.486**	0.839	0.943	1.615	1.513

Each row represents a combination of a feature set and an underlying parametric model, whose residuals were used to fit the models listed per column. Where the Param. column refers to using the parametric model as is, without error correction. While SVR, RF, XGB, RA NN, BA NN, W NN and D NN refer to respectively Support Vector Regression, Random Forest, XGBoost, ReLU Almeida et al. (2022) NNs, batch normalized ReLU Almeida et al. (2022) NNs and Wider and Deeper NNs. Lastly, comb. follows the ensemble approach where we predict the IVS directly. Base features refer to only moneyness and TTM, while tv, tv_op and tv_ap are respectively the collections of only the time-varying covariates (including the base features), the time-varying covariates with the parametric model's own parameters and the time-varying covariates with the calibrated parameters of all parametric models. Next, comb_base_ap is a feature set consisting of moneyness, TTM, all parametric model predictions, and all parametric model parameters, whereas comb_tv_ap augments this with the time-varying covariates. The test set ranges from 01/01/2021 to 31/12/2021. Bold values indicate the best result per underlying parametric model and feature combination (per row), while the underlined value is the best overall.

In terms of the NN-based approaches, the wider and deeper networks (Appendix O.c and O.d) end up performing very similarly. However, there is no longer a clear preference for these models with increased complexity over the ones offered by either the ReLU Almeida et al. (2022) or the batch normalized architectures (Appendix O.a and O.b). Furthermore, the addition of such a batch norm layer has a strong beneficial effect in certain cases such as for AHBS with tv_op reducing the RMSE from 1.627% to 0.796%. Yet, this is not always the case as the inclusion of a batch norm layer even ends up hurting the performance in certain situations. Hence, when it causes significantly better performance, this is likely due to the improved consistency during convergence, which ended up yielding a better local optimum when batch normalized than when not, but this is probably not a consistent result given a different random weight initialization. Comparing the results of the Almeida et al. (2022) architectures to the more complex ones seems to indicate that the additional complexity of the latter has a negative effect on the bias-variance trade-off. The stronger overfitting tendencies of the increased parameterization end up hurting the generalization performance more so than it seemed to do during the previous empirical experiment. Hence, simpler or more regularized models seem preferable when extending the

time frame past the one of the training data.

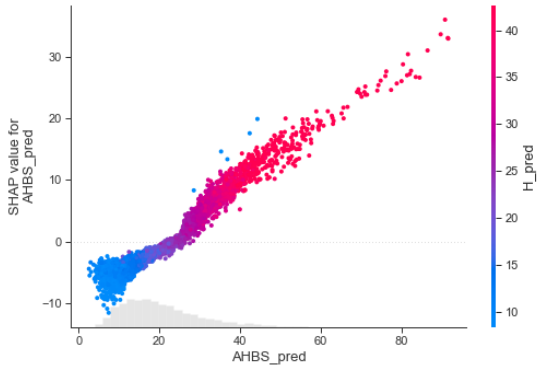
This last statement can further be argued when looking at the grid search hyperparameter choices found in Appendix L, when compared to those of Appendix H. It showcases a clear tendency towards simpler models. For instance, for the RF and XGB models this means that a lower max depth and a lower number of trees become much more frequent. Similarly, the NN-based approaches seem to more frequently choose lower complexity architectures from each provided set.

Contrary to the results of the previous two empirical exercises, the best overall performer is now to be found using the combined approach. More specifically, an XGBoost model ran on the predictions of the four different parametric models, augmented with the time-varying features, moneyness, TTM and the calibrated coefficients (comb_tv_ap). So although using the ensemble approach does not yield better results when considering the same time frame as used during training for interpolation, it considerably improves the performance when considering OOS time frames in the future. Hence, including the predictions of daily calibrated models seems to improve the generalization capabilities of the ML models producing more robust results. We hypothesize that directly relying upon the parametric model predictions as features improves the overall resilience of the ML model to changes in the greater financial environment, as these changes are partially reflected by, and thus included through, the daily calibrated parametric models themselves.

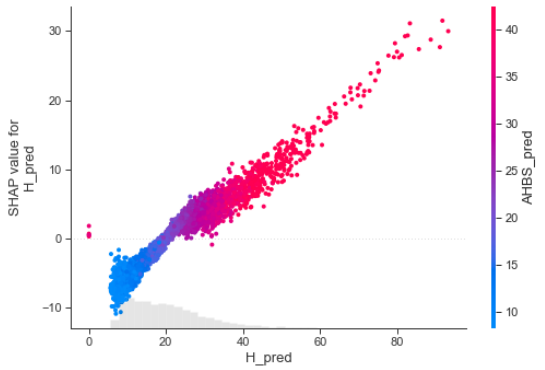
Another interesting difference with the earlier results can be found in the outperformance of the tv_ap feature set compared to the tv_op set. If we focus on the best performing model class here, the XGBoost models. We find that for the three parametric models where both sets were tested (AHBS, H, CW), we see a noticeable performance improvement in each case by including the coefficients of all parametric models rather than those of the single underlying one. Hence, this leads us to believe that these ML models do learn the correlation or relationship that might exist in the misspecification tendencies of these different parametric models.

Figure 6 again showcases the general feature importance for the best model. The XGBoost model relies mostly upon the AHBS and Heston model predictions to form the overall IV prediction. Augmented with the noticeably less important CW prediction. Moreover, moneyness and TTM are still among the most important features overall. In terms of the calibrated parameters of the different parametric models, we find that although the AHBS prediction is the most important, its coefficients are rather far down the list, whereas the Heston θ parameter still boasts an importance similar to that of the BS prediction and even higher than the Spot Volatility Index. Again the other variance-based measures, such as the spread in the VIX, re-

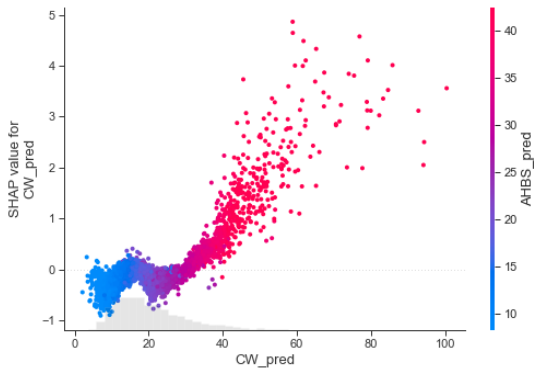
Figure 6: Bar plot of mean, absolute SHAP values (in %) per feature for the best performing model, XGB using the combined approach with comb_tv_ap feature set.



(a) AHBS

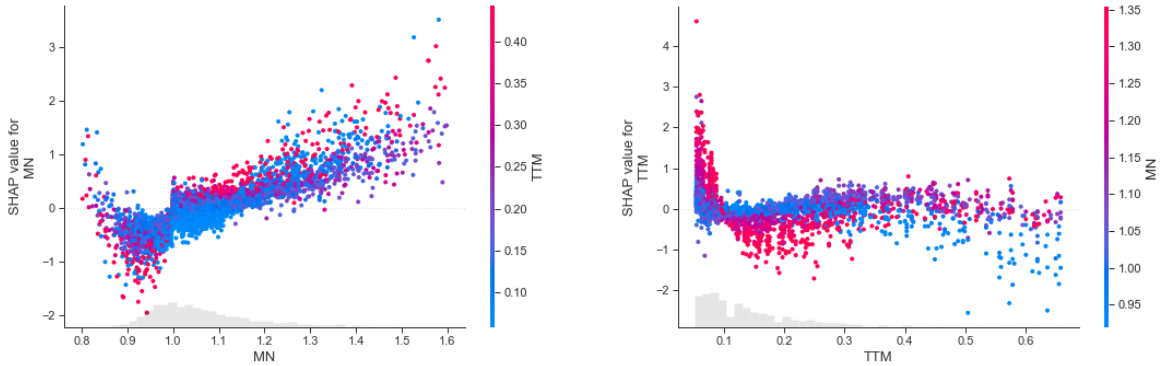


(b) H



(c) CW

Figure 7: Scatter plot for AHBS, H and CW predictions against SHAP values (in %) used as features in the XGB model of the combined approach with comb_tv_ap feature set.



(a) MN **(b)** TTM

Figure 8: Scatter plot of moneyness and TTM against SHAP values (in %), coloured by respectively TTM or moneyness for the best performing model, XGB with comb_tv_ap features.

alized variance and bipower variation showcase a rather low overall importance. This indicates that the value of these features is already largely captured by the calibrated parametric model predictions themselves.

Figures 7 and 8 show the relationship between the SHAP values and the individual feature values for some of the most important features. As expected the AHBS and Heston predictions showcase rather linear relationships and the correlation between both is high. Panel 7c shows that not only are the SHAP values more modest for the CW predictions compared to AHBS or H, but the relationship is also far from linear. It showcases a substantial drop around predicted values of 15-20% reaching negative SHAP values again, which indicates that those predictions tend to be too great and corresponds with the hump we denoted earlier in Figure J.1e of the actual Heston residuals.

Contrary to Subsection 5.2, we now find a rather linearly increasing relationship for moneyness after the initial drop for deep OTM calls, although there does again seem to be a difference depending on the TTM value. Lastly, the scatter plot of TTM is quite similar to Panel 4b. It shows the same difference for high versus low moneyness values and the same initial drop in SHAP values as TTM increases followed by a slightly negative parabolic relationship.

Given the best model of Table 4, we can check how the performance on the test set deteriorates over time as we move further away from the last date of training. Figure 9 visualizes this increase in cumulative RMSE on the test set. There is a clear increase around the end of January and the beginning of February 2021, which coincides with the volatility in the GameStop stock and the wider effects it had on the rest of the financial markets. After which, we see a slight improvement in performance during April and May, which is followed by the model getting slowly but consistently worse as time progresses. As this result seems to be general across the different top-performing models of Table 4, it indicates that this approach does not handle

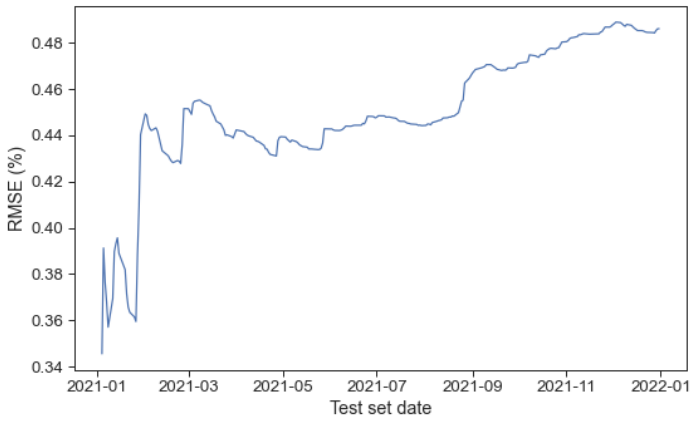


Figure 9: Cumulative RMSE (%) over time of the XGBoost model using the combined approach and comb_tv_ap feature set on the Multiday Cross-Section Reusability test set.

large changes in financial conditions well, but during less volatile periods the performance losses remain modest and hence the models maintain their reusability. So, even though we still vastly improve upon the parametric models, these secondary models fit over longer time windows do seemingly require at least semi-frequent refitting to maintain desirable performance. Although, we should refrain from drawing strong conclusions based on this single test set, as ideally, further research with a moving window approach would allow for more general recommendations regarding the deterioration by assessing it over multiple test sets under varying conditions.

6 Conclusion

Using the novel approach of Almeida et al. (2022), this paper set out to explore if ML models can correct the residual terms of parametric option pricing models, and if so, which techniques and ML models are most appropriate to do so. Given a vast dataset of recent and liquid options on the S&P 500 index, we set up different empirical experiments. Starting by assessing if daily calibrated parametric models can be combined with daily fit ML models to correct their residuals on an unseen subset of the cross-section as an interpolation exercise. Secondly, we inspect if moving towards longer time frames on which the ML models are fit, which opens up avenues for numerous time-varying covariates, proves to be beneficial. And lastly, we investigate how the ML models maintain their interpolation performance as we move further into the future and attempt to reuse the ML models on time frames beyond those used during training. As parametric model set we follow Almeida et al. (2022) and use Black-Scholes (Black & Scholes, 1973), Heston (Heston, 1993), Ad-hoc Black-Scholes (Dumas et al., 1998) and Carr-Wu (Carr & Wu, 2016) models calibrated on the daily cross-section. These are then subsequently corrected using various configurations of SVR, RF, XGBoost and NN models.

Firstly, we consistently found a preference for the simpler ML models over the NNs. In the vast majority of the setups, XGBoost-based methods yielded the best results, with the other tree-based models, RFs, being a close follow-up. SVR performed well in the daily cross-section setting, yet failed to live up to the earlier models when expanding the time frame, although still always improving upon the base parametric performance. On the contrary, the performance of NN-based settings improved when shifting away from the daily cross-sectional exercise due to the larger availability of data, both in terms of the sheer number of option specifications as well as the potential features to include, which facilitates their higher parameterization, resulting in reduced sensitivity to weight initialization, overfitting and improved convergence. In general, their performance remained subpar compared to the tree-based methods and this could not be resolved further through the use of increasingly complex architectures, nor through the different proposed regularization techniques. In most cases, either a modest amount of regularization offered by max norm, batch normalization and early stopping or none at all proved ideal. Moreover, as hypothesized, the replacement of sigmoid by ReLU activations functions in the Almeida et al. (2022) NN architectures gave desirable results through the partial resolution of vanishing gradients for the daily cross-sectional exercise, which led to the choice of ReLU for the remainder of the architectures. The vast set of architectural choices required for NN modelling, combined with their large computational complexity and often troublesome convergence, further indicates the attractiveness of XGBoost models not only being sufficient, but often preferred.

Comparing the results of the first to the second exercise, which reflects these aforementioned differences in available data, yields our second conclusion that increasing the time frame for fitting secondary models poses to be desirable overall in terms of RMSE. The time-varying nature of the second problem resulted in a preference for more complex hyperparameter choices, but still, XGBoost models performed better than even the most complex NN architectures tested.

Thirdly, the results of the last empirical experiment showed that the reusability of such ML models is rather limited during times of distress. When the financial environment changes strongly, the fit ML models show rapid deterioration in terms of their interpolation performance to price new and unseen options given a future state. However, during less volatile periods, the decline is more modest and their reusability remains acceptable. Hence, at least semi-frequent refitting of these models seems in order.

The second and third exercise do show a similar limitation, namely the fixed nature of their test sets. In the former this predominately shows itself by the fact that only one longer time frame was explored during both training and testing. While similarly, a predictive setting

with only a single test set such as during the third experiment limits the drawing of strong conclusions. Hence, further research could explore the use of a moving window approach, either for interpolation or more short-term n-day-ahead IVS predictions. Additionally, one could vary the length of the window to help determine an ideal periodicity to deal with a changing financial environment. This was avoided here due to computational limitations.

Moreover, we explored a plethora of time-varying covariates as well as the inclusion of the calibrated coefficients of the parametric models in our attempt to learn how to correct their predictions. Of the different volatility-based measures, the Spot Volatility Index reigned supreme, while the VIX-based measures, the Realized Variance and Bipower Variation yielded consistently lower feature importances. The inclusion of the underlying parametric model's calibrated coefficients was deemed beneficial. However, the calibrated parameters of other parametric models than the one on which the residuals were based only benefited the final experiment. Accordingly, this prevents us from definitively stating whether these ML models successfully learn to capture correlation across the misspecification of parametric models. The highly non-linear relationships between the numerous features and the PES did not require the more complex NN architectures as initially thought, and were appropriately captured by the XGBoost models. Furthermore, comparing these relationships to the residuals of parametric models indicated a strong benefit to using the proposed two-step estimation approach for more extreme option specifications, for which the parametric models often leave much to be desired.

In terms of the different modelling approaches, our novel ensemble method only improved upon the technique of modelling the PES of individual parametric models during the last experiment. We hypothesize that directly relying upon the parametric model predictions as features improves the overall resilience of the ML model to changes in the financial environment, as these changes are partially reflected by the daily calibrated parametric models themselves. In addition, the predictions by the various parametric models showed strong differences in feature importance scores, where the AHBS predictions, followed closely by the Heston ones, were given substantially higher weights than those of CW or BSM. As only four parametric models were explored in this paper, this raises the question for further research on whether a different, larger or more diverse set could yield even better results.

Nonetheless, in all settings, we managed to vastly improve on just using the parametric models by themselves through the proposed two-step estimation approach. Hence, this easily adaptable framework proves to be highly valuable for practitioners for the pricing of options under a wide array of different circumstances, while being largely automatable and relatively limited in computational complexity when using the proposed XGBoost models.

References

Aït-Sahalia, Y., & Lo, A. W. (1998). Nonparametric Estimation of State-Price Densities Implicit in Financial Asset Prices. *The Journal of Finance*, *53*(2), 499–547. https://doi.org/10.1111/0022-1082.215228

Albrecher, H., Mayer, P., Schoutens, W., & Tistaert, J. (2006). *The Little Heston Trap*.

Almeida, C., Fan, J., Freire, G., & Tang, F. (2022). Can a Machine Correct Option Pricing Models? *Working Paper*. https://dx.doi.org/10.2139/ssrn.3835108

Andersen, T. G., Fusari, N., & Todorov, V. (2015). The risk premia embedded in index options. *Journal of Financial Economics*, *117*(3), 558–584. https://doi.org/10.1016/j.jfineco.2015.06.005

Andreou, P. C., Charalambous, C., & Martzoukos, S. H. (2010). Generalized parameter functions for option pricing. *Journal of Banking and Finance*, *34*(3), 633–646. https://doi.org/10.1016/J.JBANKFIN.2009.08.027

Bakshi, G., Cao, C., & Chen, Z. (1997). Empirical Performance of Alternative Option Pricing Models. *The Journal of Finance*, *52*(5), 2003–2049. https://doi.org/10.1111/j.1540-6261.1997.tb02749.x

Bates, D. S. (1991). The Crash of '87: Was It Expected? The Evidence from Options Markets. *The Journal of Finance*, *46*(3), 1009–1044. https://doi.org/10.1111/j.1540-6261.1991.tb03775.x

Bates, D. S. (2019). How Crashes Develop: Intradaily Volatility and Crash Evolution. *Journal of Finance*, *74*(1), 193–238. https://doi.org/10.1111/JOFI.12732

Bates, D. S. (2021). Empirical Option Pricing Models. *NBER Working Paper Series*, *29554*. https://doi.org/10.3386/w29554

Binsbergen, J. v., Brandt, M., & Koijen, R. (2012). On the Timing and Pricing of Dividends. *American Economic Review*, *102*(4), 1596–1618. https://doi.org/10.1257/aer.102.4.1596

Black, F. (1976). The pricing of commodity contracts. *Journal of Financial Economics*, *3*(1), 167–179. https://doi.org/10.1016/0304-405X(76)90024-6

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, *81*(3), 637–654. https://doi.org/10.1086/260062

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). Training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 144–152. https://doi.org/10.1145/130385.130401

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Carr, P., & Wu, L. (2016). Analyzing volatility risk and risk premium in option contracts: A new theory. *Journal of Financial Economics*, *120*(1), 1–20. https://doi.org/10.1016/j.jfineco.2016.01.004

Carr, P., & Wu, L. (2019). Leverage Effect, Volatility Feedback, and Self-Exciting Market Disruptions. *Journal of Financial and Quantitative Analysis*, *52*(5), 2119–2156. https://doi.org/10.1017/S0022109017000564

Carverhill, A. P., & Cheuk, T. H. F. (2003). Alternative Neural Network Approach for Option Pricing and Hedging. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.480562

CBOE. (2022). S&P 500 Index Options Fact Sheet. https://cdn.cboe.com/resources/spx/spx-fact-sheet.pdf

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 1–27. https://doi.org/10.1145/1961189.1961199

Chen, S. X., & Xu, Z. (2014). On implied volatility for options - Some reasons to smile and more to correct. *Journal of Econometrics*, *179*(1), 1–15. https://doi.org/10.1016/j.jeconom.2013.10.007

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Cortes, C., Vapnik, V., & Saitta, L. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Cox, J. C., Ingersoll, J. E., & Ross, S. A. (1985). A Theory of the Term Structure of Interest Rates. *Econometrica*, *53*(2), 385–407. https://doi.org/10.2307/1911242

Datta, A., Sen, S., & Zick, Y. (2016). Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems. *2016 IEEE Symposium on Security and Privacy (SP)*, 598–617. https://doi.org/10.1109/SP.2016.42

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1996). Support Vector Regression Machines. *Proceedings of the Nineth Advances in Neural Information Processing Systems*.

Duffie, D., Pan, J., & Singleton, K. J. (2000). Transform Analysis and Asset Pricing for Affine Jump-Diffusions. *Econometrica*, *68*(6), 1343–1376. https://doi.org/10.2139/SSRN.157733

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2009). Incorporating Functional Knowledge in Neural Networks. *Journal of Machine Learning Research*, *10*, 1239–1262.

Dumas, B., Fleming, J., & Whaley, R. E. (1998). Implied volatility functions: Empirical tests. *Journal of Finance*, *53*(6), 2059–2106. https://doi.org/10.1111/0022-1082.00083

Eldan, R., & Shamir, O. (2015). The Power of Depth for Feedforward Neural Networks. *Journal of Machine Learning Research*, *49*, 907–940. https://doi.org/10.48550/arxiv.1512.03965

European Banking Authority. (2020). *Report on Big Data and Advanced Analytics* (tech. rep.).

European Banking Authority. (2021). *Discussion Paper On Machine Learning For IRB Models* (tech. rep.).

European Commission. (2021). Laying down harmonised rules on Artificial Intelligence (Artificial Intelligence Act) and amending certain Union Legislative Acts.

Fan, J., & Mancini, L. (2009). Option Pricing with Model-Guided Nonparametric Methods. *Journal of the American Statistical Association*, *104*(488), 1351–1372. https://doi.org/10.2139/ssrn.955740

Fisher, A., Rudin, C., & Dominici, F. (2019). All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research*, *20*, 1–81. https://doi.org/10.48550/arxiv.1801.01489

Garman, M. B., & Kohlhagen, S. W. (1983). Foreign currency option values. *Journal of International Money and Finance*, *2*(3), 231–237. https://doi.org/10.1016/S0261-5606(83)80001-1

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, *9*, 249–256. http://proceedings.mlr.press/v9/glorot10a.html

Golez, B. (2014). Expected Returns and Dividend Growth Rates Implied by Derivative Markets. *The Review of Financial Studies*, *27*(3), 790–822. https://doi.org/10.1093/RFS/HHT131

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, *33*(5), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV*, 1026–1034. https://doi.org/10.1109/ICCV.2015.123

Heston, S. L. (1993). A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *Review of Financial Studies*, *6*(2), 327–343. https://doi.org/10.1093/rfs/6.2.327

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. https://doi.org/10.48550/arXiv.1207.0580

Hull, J. C. (2017). *Options, Futures, and Other Derivatives* (9th ed.). Pearson Education.

Hull, J. C., & White, A. (1987). The Pricing of Options on Assets with Stochastic Volatilities. *The Journal of Finance*, *42*(2), 281–300. https://doi.org/10.1111/J.1540-6261.1987.TB02568.X

Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994). A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. *The Journal of Finance*, *49*(3), 851. https://doi.org/10.2307/2329209

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 448–456. https://doi.org/10.48550/arxiv.1502.03167

Jäckel, P. (2015). Let's Be Rational. *Wilmott*, *2015*(75), 40–53. https://doi.org/10.1002/wilm.10395

Kahl, C., & Jäckel, P. (2005). Not-so-complex logarithms in the Heston model. *Wilmott*, *19*, 94–103. https://web.math.ku.dk/~rolf/KahlJackel.pdf

Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *Proceedings of the Third International Conference on Learning Representations, ICLR*. https://doi.org/10.48550/arxiv.1412.6980

Kumar, S., Mohri, M., & Talwalkar, A. (2012). Sampling Methods for the Nyström Method. *Journal of Machine Learning Research*, *13*, 981–1006. https://www.jmlr.org/papers/v13/kumar12a.html

Lipovetsky, S., & Conklin, M. (2001). Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, *17*(4), 319–330. https://doi.org/10.1002/asmb.446

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S.-I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, *2*(1), 56–67. https://doi.org/10.1038/s42256-019-0138-9

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS*, 4768–4777. https://doi.org/10.48550/arXiv.1705.07874

Malliaris, M., & Salchenberger, L. (1993). A neural network model for estimating option prices. *Applied Intelligence*, *3*(3), 193–206. https://doi.org/10.1007/BF00871937

Masters, T. (1993). *Practical Neural Network Recipies in C++*. Elsevier. https://doi.org/10.1016/C2009-0-22399-3

OptionMetrics. (2021). IvyDB File and Data Reference Manual.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. https://doi.org/10.5555/1953048.2078195

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., & Liao, Q. (2017). Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, *14*(5), 503–519. https://doi.org/10.1007/s11633-017-1054-2

Prechelt, L. (2012). Early Stopping — But When? *Neural networks: Tricks of the trade* (pp. 53–67). Springer. https://doi.org/10.1007/978-3-642-35289-8_5

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, 97–101. https://doi.org/10.48550/arxiv.1602.04938

Ruf, J., & Wang, W. (2020). Neural networks for option pricing and hedging: a literature review. *The Journal of Computational Finance*, *24*(1), 1–46. https://doi.org/10.21314/JCF.2020.390

Shapley, L. S. (1953). A Value for n-Person Games. *Contributions to the theory of games* (pp. 307–318). Princeton University Press. https://doi.org/10.1515/9781400881970-018

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. https://jmlr.org/papers/v15/srivastava14a.html

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, *41*(3), 647–665. https://doi.org/10.1007/s10115-013-0679-x

Sussillo, D., & Abbott, L. F. (2014). Random Walk Initialization for Training Very Deep Feedforward Networks. https://doi.org/10.48550/arxiv.1412.6558

Todorov, V. (2019). Nonparametric spot volatility from options. *The Annals of Applied Probability*, *29*(6), 3590–3636. https://doi.org/10.1214/19-AAP1488

Vapnik, V. (1998). *Statistical learning theory* (1st ed.). Wiley.

Wallmeier, M. (2022). Quality Issues of Implied Volatilities of Index and Stock Options in the OptionMetrics IvyDB Database. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.4025257

Yao, Y., Rosasco, L., & Caponnetto, A. (2007). On Early Stopping in Gradient Descent Learning. *Constructive Approximation*, *26*(2), 289–315. https://doi.org/10.1007/s00365-006-0663-2

Zhang, T., & Yu, B. (2005). Boosting with Early Stopping: Convergence and Consistency. *The Annals of Statistics*, *33*(4), 1538–1579. https://doi.org/10.1214/009053605000000255

Appendix

A Data Descriptives

Table A.1: Summary statistics of OptionMetrics data aggregated by trading day.

	# Options	# Expiries	Volume	IV (%)	% Calls
mean	1747.461	19.640	444,106.533	20.911	31.488
std	524.850	1.765	174,773.153	6.239	4.766
min	785	15	121,846.000	13.157	22.395
25%	1327	18	313,458.750	16.877	28.372
50%	1752	20	411,750.500	19.426	30.321
75%	2073	21	535,696.500	23.152	33.337
max	3896	24	1,648,554.000	68.651	57.650

This table reports the mean, standard deviation and quantiles of the number of option specifications, number of expiration dates, total traded volume, mean IV and the proportion of calls per day for the data sample ranging from 04/01/2016 to 31/12/2021.

Table A.2: Relative distribution of moneyness categories as defined in Section 3.

	%
DOTMP	33.259
ATM	25.187
OTMP	22.493
OTMC	16.269
DOTMC	2.792

Table A.3: Relative distribution of calls and puts.

	%
Calls	31.850
Puts	68.150

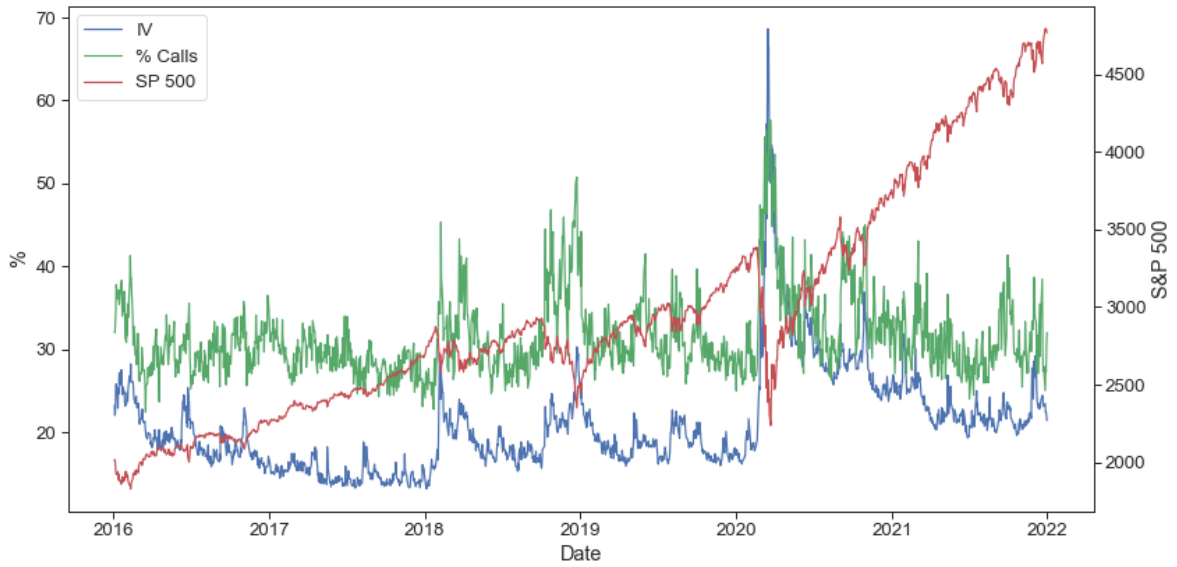


Figure A.1: The mean IV and fraction of options that are calls (left axis) and the S&P 500 index level (right axis) over time.

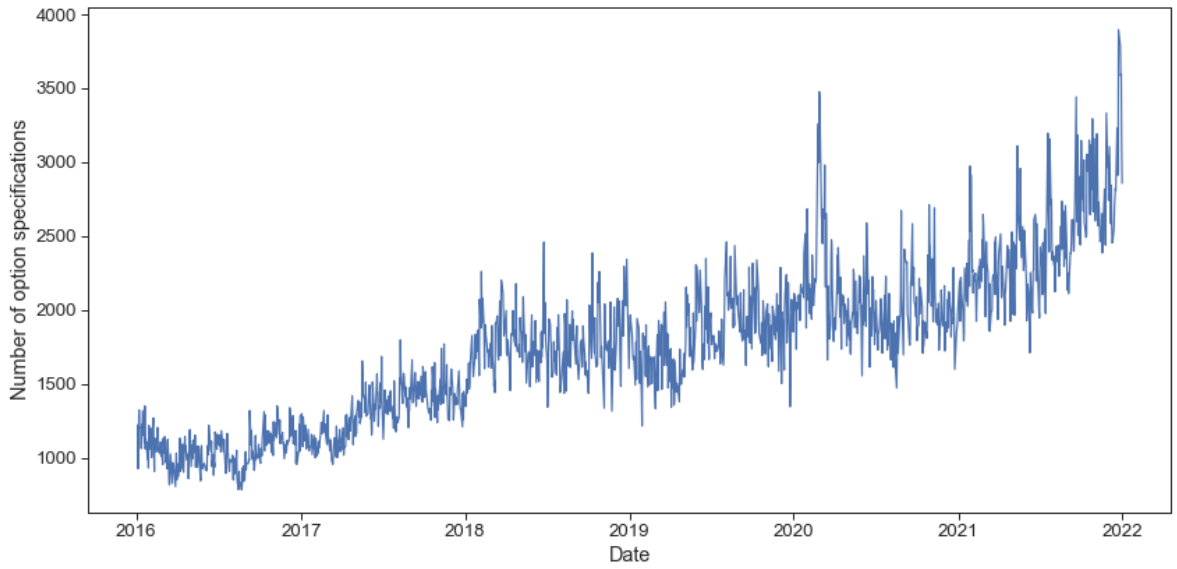


Figure A.2: The number of different option specifications available on the S&P 500 over time.

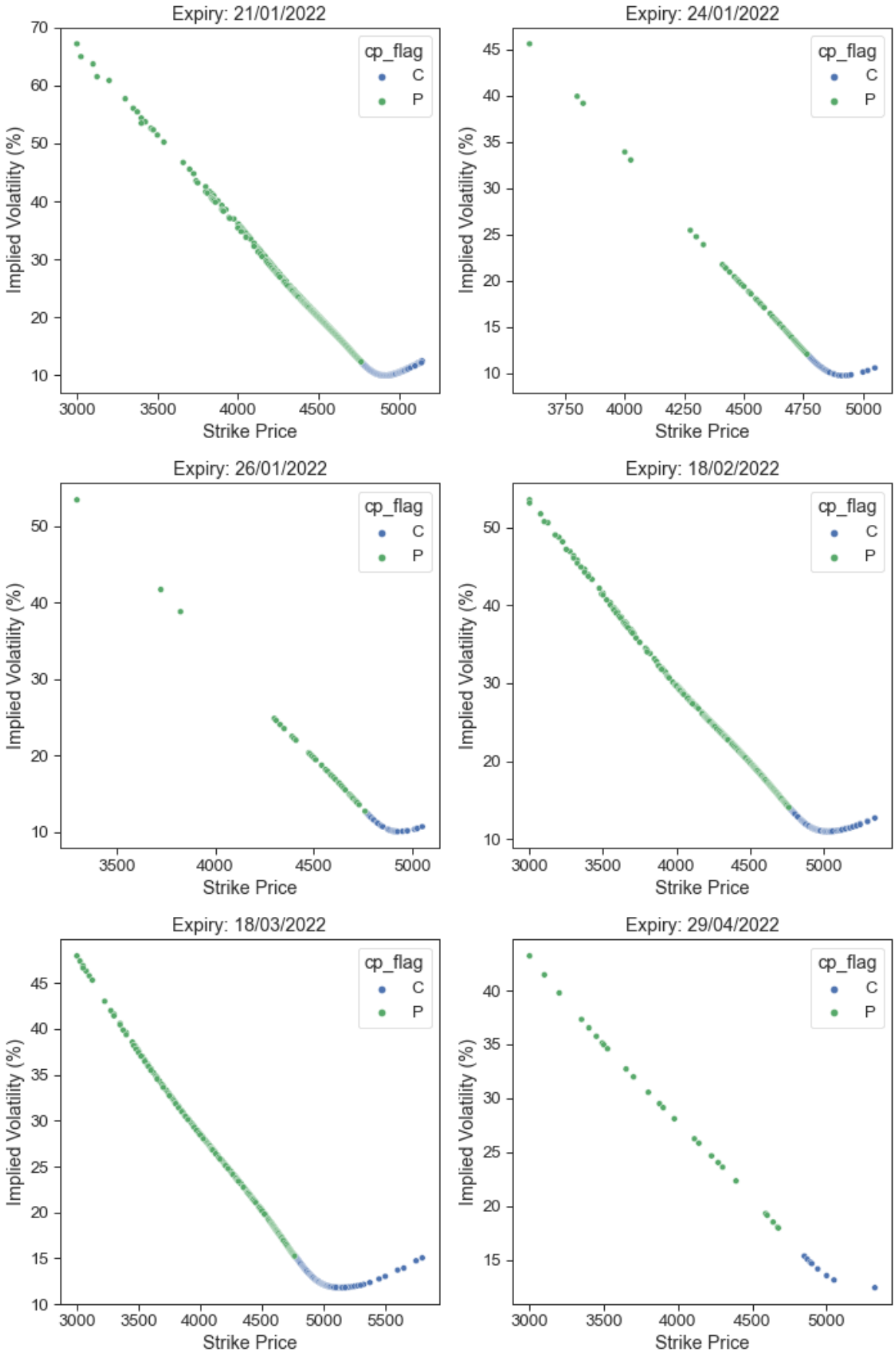


Figure A.3: IV skew on a subset of the S&P 500 index options on the most recent date of the data, 31/12/2021, where cp_flag is an indicator for calls and puts.

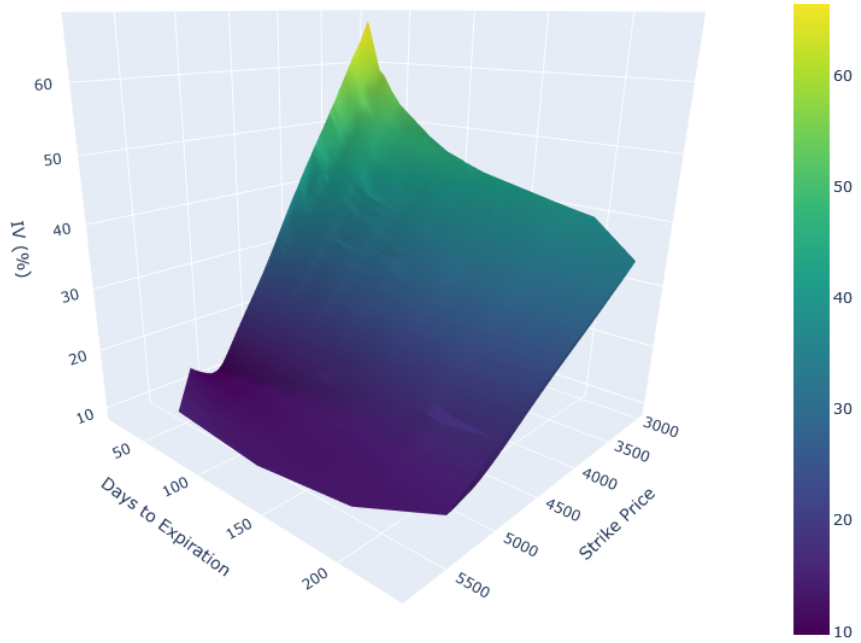


Figure A.4: IVS of the S&P 500 index options on the most recent date of the data, 31/12/2021.

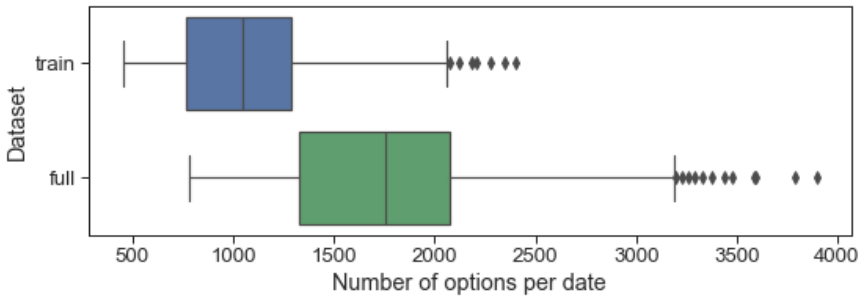
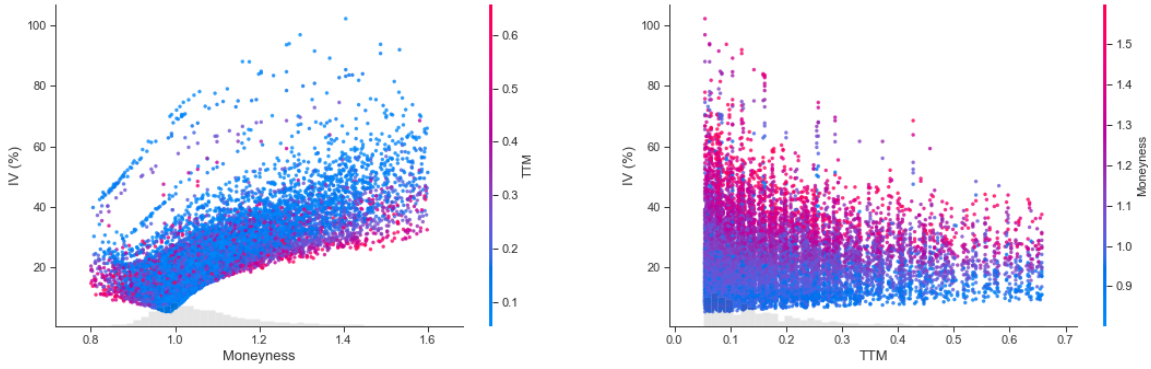


Figure A.5: Box plot of the number of option specifications per day for the training data, with K divisible by 10, and the full dataset.



(a) Moneyness **(b)** TTM

Figure A.6: Observed IV (%) distribution in terms of moneyness (a) and TTM (b), coloured by respectively TTM and moneyness for a subset of the data spread across the entire time frame.

B Heston Pricing Formula

Following the notation of Kahl and Jäckel (2005), we can define the quasi-closed-form solution for European options according to the Heston dynamics as

$$C\left(S_t, K, V_0, \tau\right) = P(\tau) \cdot \left[\frac{1}{2}(F - K) + \frac{1}{\pi}\int_0^\infty \left(F f_1 - K f_2\right) \mathrm{d}u\right], \tag{36}$$

where f_1 and f_2 are

$$f_1 = \mathrm{Re}\left(\frac{\mathrm{e}^{-iu\ln K}\varphi(u-i)}{iuF}\right) \text{ and } f_2 = \mathrm{Re}\left(\frac{\mathrm{e}^{-iu\ln K}\varphi(u)}{iu}\right), \tag{37}$$

with $F = S\mathrm{e}^{\mu\tau}$ and $P(\tau)$ as the discount factor to the option expiry date where we use $P(\tau) = e^{-r\tau}$ (Kahl & Jäckel, 2005). The function $\varphi(\cdot)$ is defined as the log-characteristic function of the underlying asset value S_τ at expiry,

$$\varphi(u) = \mathrm{E}\left[\mathrm{e}^{iu\ln S_\tau}\right], \tag{38}$$

where by virtue of definition it holds that $\varphi(0) = 1$ and $\varphi(-i) = F$. Following the derivation of Heston (1993) results in

$$\varphi(u) = \mathrm{e}^{C(\tau,u)+D(\tau,u)V_0+iu\ln F}, \tag{39}$$

with

$$C(\tau,u) = \frac{\kappa\bar{v}}{\sigma_v^2}\left((\kappa - \rho\sigma_v ui + d(u))\tau - 2\ln\left(\frac{c(u)\mathrm{e}^{d(u)\tau}-1}{c(u)-1}\right)\right), \tag{40}$$

$$D(\tau,u) = \frac{\kappa - \rho\sigma_v ui + d(u)}{\sigma_v^2}\left(\frac{\mathrm{e}^{d(u)\tau}-1}{c(u)\mathrm{e}^{d(u)\tau}-1}\right), \tag{41}$$

$$c(u) = \frac{\kappa - \rho\sigma_v ui + d(u)}{\kappa - \rho\sigma_v ui - d(u)}, \tag{42}$$

$$d(u) = \sqrt{(\rho\sigma_v ui - \kappa)^2 + iu\sigma_v^2 + \sigma_v^2 u^2}. \tag{43}$$

C Elaboration on Feature Importance Measures

The following appendix entry covers some more in-depth discussion of and reasoning for the choice of the SHAP feature importance measure and should be read as an extension to the Subsection 4.5 if greater detail is desired.

C.a Gu et al. (2020), Almeida et al. (2022) and Permutation-based Approaches

Measuring the reduction in R^2 in Gu et al. (2020), or similarly in RMSE in Almeida et al. (2022), when setting a particular feature to zero, can be seen as a simplification of the more popular permutation-based approaches. These were originally implemented for RFs (Breiman, 2001) and later adopted by others and generalized to a model-agnostic version such as by Fisher et al. (2019). They work by assessing the initial model's error, subsequently iterating through every feature, j, and permuting the values of feature j of the original dataset X resulting in a new dataset denoted as X_j. After which, the model predicts on X_j and you calculate how much worse the error on X_j got compared to the error on the original dataset X, which gives a sense of feature importance. Namely, the stronger the increase in the error, the more the model relies upon the values of that feature j to make its prediction and hence the more important it is. This is essentially the same as the methods used by Gu et al. (2020) and Almeida et al. (2022), but rather than permuting feature j, they set it to 0 instead to break the relationship with the other features present.

Both approaches hold similar disadvantages (Fisher et al., 2019), of which we will cover the main ones here. First of all, correlated features can bias the feature importance scores. If we assume that feature A and B are correlated and there exists a strong relationship of A and B with the dependent variable y. Then permutation-based approaches on A will underestimate its importance as some of the relationship continues to be present in the data through B or vice versa. Hence, it would be more accurate to assess the importance of A and B simultaneously by permuting both at once. It, however, becomes rather arbitrary to determine when two or more features are correlated highly enough to warrant co-permutation and, furthermore, correlation only captures linear relationships, but the same reasoning holds for other types of relationships as well. Secondly, permutation, but certainly setting feature values to zero, will often create unrealistic samples and hence has the model predict instances that would not occur in practice, which is not something the model should be able to do anyway and therefore its decrease in performance is not as relevant anymore.

C.b LIME

Local-Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016) deploys local surrogate models that are trained to approximate predictions made by the underlying black-box model. LIME does this by generating a new dataset, through small, randomized perturbations of an original sample, on which a secondary, interpretable model is trained using the predictions made by the original black-box model as dependent variable. Furthermore, the perturbed samples are weighted in accordance to their proximity to the original sample of interest. Hence, this results in a smaller, surrogate and directly interpretable model (often LASSO or DTs) that attempts to explain the predictions made by the black-box on particular samples of interest (Ribeiro et al., 2016). Moreover, these surrogate models can be trained on the same or different features and/or transformations than the black-box model, which further increases the flexibility of the method.

C.c Shapley Properties

As was shown by Shapley (1953), Shapley values showcase many desirable properties, of which the most important ones are listed below.

Efficiency states that the contributions of all players must add up to the value of the grand coalition of all players P:

$$\sum_{j=1}^{p} \phi_j = v(P) \tag{44}$$

Secondly, symmetry states that two feature values j and k should be identical if they share an equal contribution among all possible coalitions S:

$$\begin{aligned} &\text{if } v(S \cup \{j\}) = v(S \cup \{k\}), \qquad \forall S \subseteq \{1, \dots, p\} \setminus \{j, k\} \\ &\qquad \text{then } \phi_j = \phi_k \end{aligned} \tag{45}$$

Lastly, Shapley values fulfil additivity such that for two coalition games, with value functions v and w respectively, the total distributed gains for player j should correspond to the sum of the gains of both individual games:

$$\phi_j(v + w) = \phi_j(v) + \phi_j(w) \tag{46}$$

While for every player $j \in P$ and any $a \in \mathbb{R}$ it similarly holds that $\phi_j(av) = a\phi_j(v)$.

D Hyperparameters Daily Cross-Section

Below we list the hyperparameters used during the Daily Cross-Section exercise, settings left unspecified follow the defaults of Python libraries `Scikit-Learn v1.1.1`, `XGBoost v1.6.1` or `TensorFlow v2.9.1`. Let us denote w as a vector of linearly spaced values from 0.01 up to 5 with size 10, and $v = e^w - 1$.:

1. SVR:
   (a) kernel: radial basis
   (b) gamma: v
   (c) C: v
   (d) epsilon: 0.001, 0.0001
2. RF:
   (a) n_estimators: 100, 200, 500
   (b) max_depth: 6, 8, 10, 12
3. XGBoost:
   (a) eta: 0.01, 0.1, 0.3
   (b) n_estimators: 25, 50, 100, 200
   (c) max_depth: 6, 12
   (d) subsample: 0.5, 0.75, 1
4. Almeida et al. (2022) NN1, NN2, NN3, NN4, NN5:
   (a) with sigmoid activation functions (Appendix N.a)
   (b) with ReLU activation functions (Appendix N.b)
5. Batch Normalized Almeida et al. (2022):
   (a) ReLU NN1, NN2, NN3, NN4, NN5 each augmented with a single BN layer after the first hidden layer (Appendix N.c)
6. Wider NNs:
   (a) Wider architectures W NN1 up to W NN4 (Appendix N.d)
7. Deeper NNs:
   (a) Deeper architectures D NN1 up to D NN4 (Appendix N.e)

E Grid Search Results for Daily Cross-Section

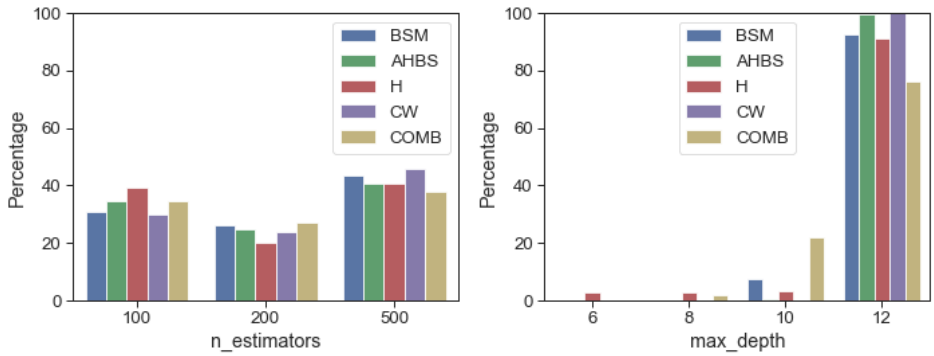


Figure E.1: Distribution of chosen RF hyperparameters per grid search for the Daily Cross-Section exercise.

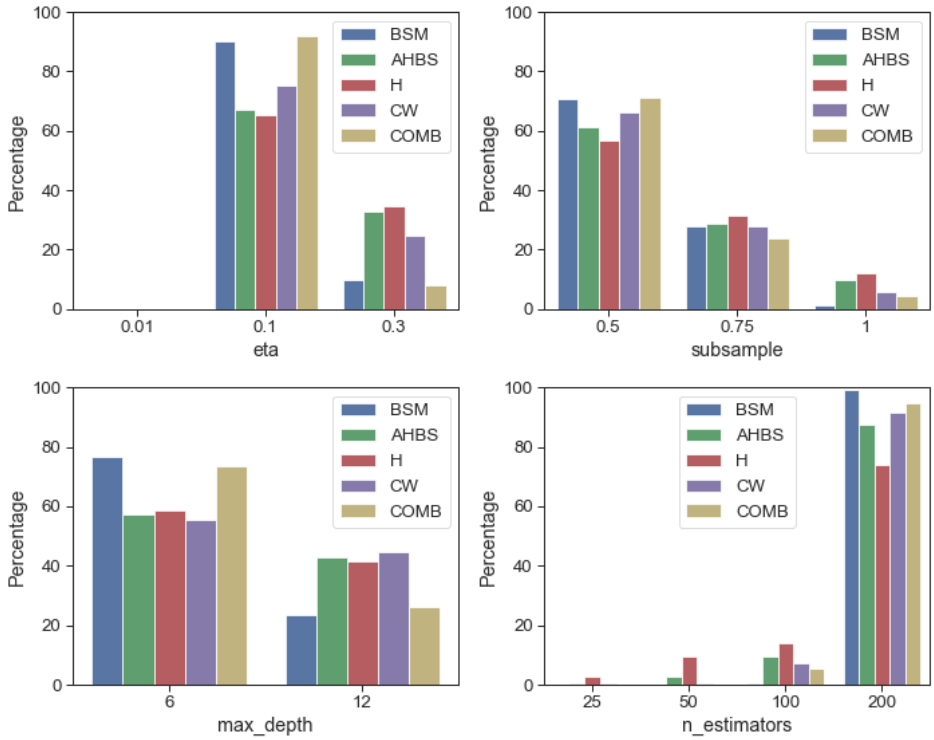


Figure E.2: Distribution of chosen XGB hyperparameters per grid search for the Daily Cross-Section exercise.

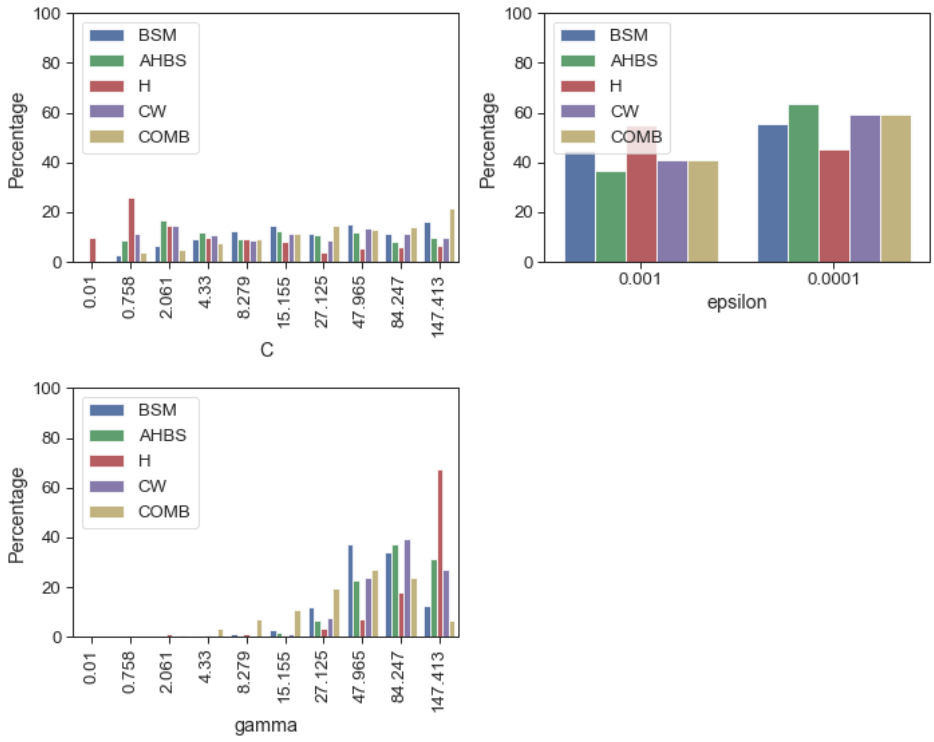
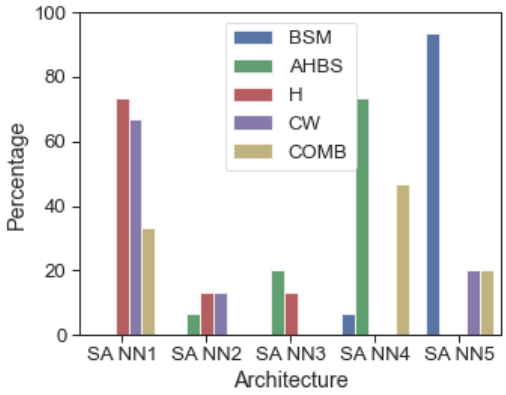
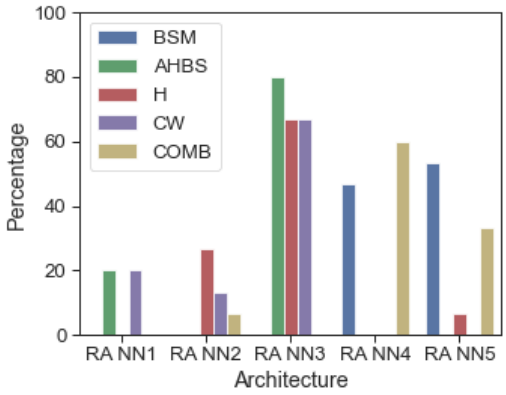


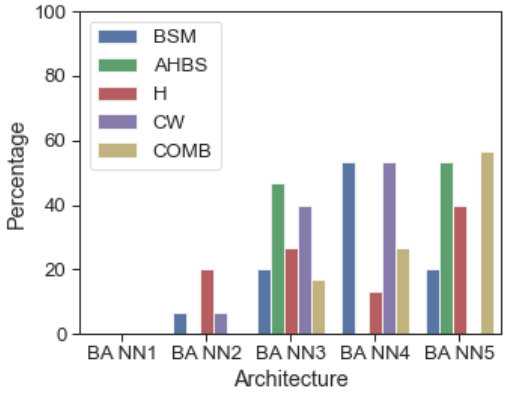
Figure E.3: Distribution of chosen SVR hyperparameters per grid search for the Daily Cross-Section exercise.



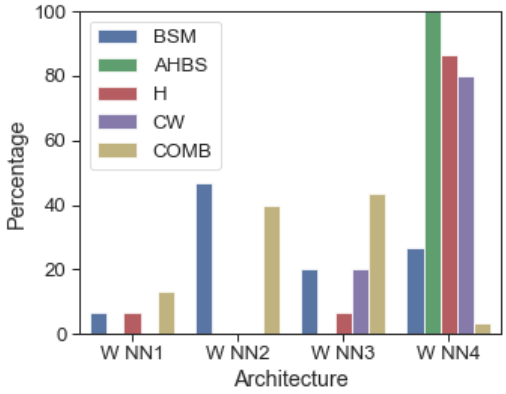
(a) SA NN



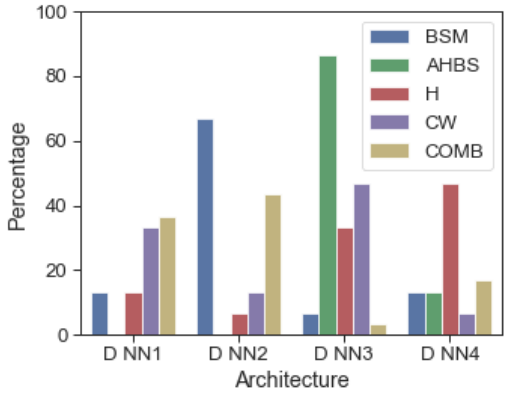
(b) RA NN



(c) BA NN



(d) W NN



(e) D NN

Figure E.4: Distribution of chosen NN architectures per grid searched set for the Daily Cross-Section exercise.

F Train, Val. and Test Results of Daily Cross-Section

Table F.1: Train, validation and test set RMSE scores (%) of the Daily Cross-Section exercise.

ML Param.	Param.	SVR	RF	XGB	SA NN	RA NN	BA NN	W NN	D NN
BS	10.718	0.158	0.145	**0.076**	2.283	1.096	0.934	7.772	7.253
AHBS	1.584	0.151	0.102	**0.060**	1.477	1.261	0.765	1.721	1.497
H	1.484	0.955	0.388	**0.245**	1.638	1.494	1.306	4.791	1.702
CW	1.911	0.151	0.090	**0.058**	1.504	1.274	0.572	2.731	1.605
COMB.		0.143	0.169	**0.066**	1.099	0.439	0.718	10.115	6.517

(a) train

ML Param.	Param.	SVR	RF	XGB	SA NN	RA NN	BA NN	W NN	D NN
BS	10.737	**0.184**	0.437	0.254	2.272	1.110	0.718	5.946	7.133
AHBS	1.581	**0.177**	0.308	0.209	1.425	1.304	0.753	1.472	1.387
H	1.490	**0.857**	1.050	0.944	1.562	1.438	1.184	1.606	1.547
CW	1.906	0.178	0.245	**0.173**	1.453	1.232	0.624	1.497	1.440
COMB.		**0.163**	0.504	0.371	1.037	0.381	0.696	4.369	6.590

(b) val

ML Param.	Param.	SVR	RF	XGB	SA NN	RA NN	BA NN	W NN	D NN
BS	8.855	**0.176**	0.293	0.238	2.026	1.013	0.868	7.390	6.437
AHBS	1.369	0.173	0.165	**0.149**	1.286	1.126	0.702	1.526	1.295
H	1.247	0.959	**0.687**	0.708	1.299	1.173	0.915	4.682	1.375
CW	1.803	0.169	0.150	**0.137**	1.350	1.117	0.556	2.470	1.487
COMB.		**0.221**	0.355	0.276	0.905	0.400	0.681	9.196	5.646

(c) test

G Hyperparameters Multiday Cross-Section Interpolation and Reusability

Below we list the hyperparameters used during the multiday exercises, settings left unspecified follow the defaults of Python libraries `Scikit-Learn v1.1.1`, `XGBoost v1.6.1` or `TensorFlow v2.9.1`. Let us denote w as a vector of linearly spaced values from 0.01 up to 5 with size 5, and $v = e^w - 1$.:

1. SVR with Nystrom Approximation:
   (a) kernel: radial basis
   (b) gamma: v
   (c) Nystrom n_components: 10000
   (d) C: v
   (e) epsilon: 0.001, 0.0001
2. RF:
   (a) n_estimators: 100, 200, 500
   (b) max_depth: 6, 8, 10, 12, 24
3. XGBoost:
   (a) eta: 0.01, 0.1, 0.3
   (b) n_estimators: 50, 100, 200, 500
   (c) max_depth: 6, 12, 24
4. Almeida et al. (2022) NN1, NN2, NN3, NN4, NN5:
   (a) with ReLU activation functions (Appendix O.a)
5. Batch Normalized Almeida et al. (2022):
   (a) ReLU NN1, NN2, NN3, NN4, NN5 each augmented with a single BN layer after the first hidden layer (Appendix O.b)
6. Wider NNs:
   (a) Wider architectures W NN1 up to W NN4 (Appendix O.c)
7. Deeper NNs:
   (a) Deeper architectures D NN1 up to D NN4 (Appendix O.d)

H Grid Search Results for Multiday Cross-Section Interpolation

Table H.1: Optimal hyperparameters according to grid search for the different ML models and NN architecture sets for Multiday Cross-Section Interpolation.

Param.	Model Hyperparam Features	RF max depth	n_estimators	XGB max depth	n_estimators	eta	SVR C	epsilon	gamma	RA NN Arch.	BA NN Arch.	W NN Arch.	D NN Arch.
BS	base	24	500	12	500	0.3000	0.0101	0.0001	2.5166	RA NN5	BA NN2	W NN1	D NN2
	tv	24	500	24	500	0.1000	41.6275	0.0010	0.0101	RA NN4	BA NN5	W NN2	D NN2
	tv_ap	24	500	24	500	0.1000	41.6275	0.0010	0.0101	RA NN5	BA NN4	W NN3	D NN1
AHBS	base	24	500	12	500	0.3000	0.0101	0.0001	2.5166	RA NN3	BA NN3	W NN4	D NN1
	tv_op	24	500	24	500	0.1000	147.4132	0.0001	0.0101	RA NN3	BA NN3	W NN3	D NN4
	tv_ap	24	500	24	500	0.1000	2.5166	0.0010	0.0101	RA NN3	BA NN3	W NN4	D NN2
CW	base	12	200	12	500	0.3000	0.0101	0.0010	2.5166	RA NN3	BA NN5	W NN4	D NN2
	tv_op	24	500	24	500	0.1000	147.4132	0.0001	0.0101	RA NN4	BA NN5	W NN4	D NN2
	tv_ap	24	100	24	500	0.1000	11.2436	0.0010	0.0101	RA NN4	BA NN2	W NN3	D NN2
H	base	10	200	6	500	0.3000	0.0101	0.0001	2.5166	RA NN3	BA NN2	W NN4	D NN3
	tv_op	24	100	12	500	0.1000	11.2436	0.0001	0.0101	RA NN3	BA NN2	W NN4	D NN1
	tv_ap	12	200	12	500	0.1000	2.5166	0.0010	0.0101	RA NN2	BA NN3	W NN3	D NN3
COMB	comb_base_ap	12	500	12	500	0.1000	41.6275	0.0001	0.0101	RA NN5	BA NN3	W NN4	D NN1
	comb_tv_ap	12	500	12	500	0.1000	2.5166	0.0010	0.0101	RA NN4	BA NN3	W NN3	D NN4

I Train, Val. and Test Results of Multiday Cross-Section Interpolation

Table I.1: Train, validation and test set RMSE scores (%) of the Multiday Cross-Section Interpolation exercise.

Param.	ML Features	Param.	SVR	RF	XGB	RA NN	BA NN	W NN	D NN
BS	base	10.718	1.825	0.666	**0.562**	1.798	1.801	1.822	1.795
	tv		1.045	0.182	**0.049**	0.578	0.620	0.204	0.230
	tv_ap		1.076	0.128	**0.048**	0.318	0.386	0.265	0.262
AHBS	base	1.584	0.985	0.401	**0.348**	0.973	0.971	0.974	0.973
	tv_op		0.826	0.085	**0.042**	0.363	0.351	0.172	0.166
	tv_ap		1.049	0.080	**0.042**	0.325	0.321	0.150	0.169
H	base	1.484	1.408	0.994	**0.607**	1.333	1.337	1.385	1.329
	tv_op		1.355	0.382	**0.096**	0.990	0.936	0.894	0.763
	tv_ap		1.362	0.547	**0.098**	0.934	0.945	0.694	0.674
CW	base	1.911	1.360	1.104	**0.396**	1.308	1.334	1.339	1.322
	tv_op		0.828	0.085	**0.043**	0.357	0.372	0.135	0.202
	tv_ap		0.981	0.085	**0.043**	0.318	0.343	0.159	0.192
COMB	comb_base_ap		0.475	0.376	**0.082**	0.311	0.394	0.464	0.280
	comb_tv_ap		0.646	0.374	**0.074**	0.290	0.366	0.345	0.266

(a) train

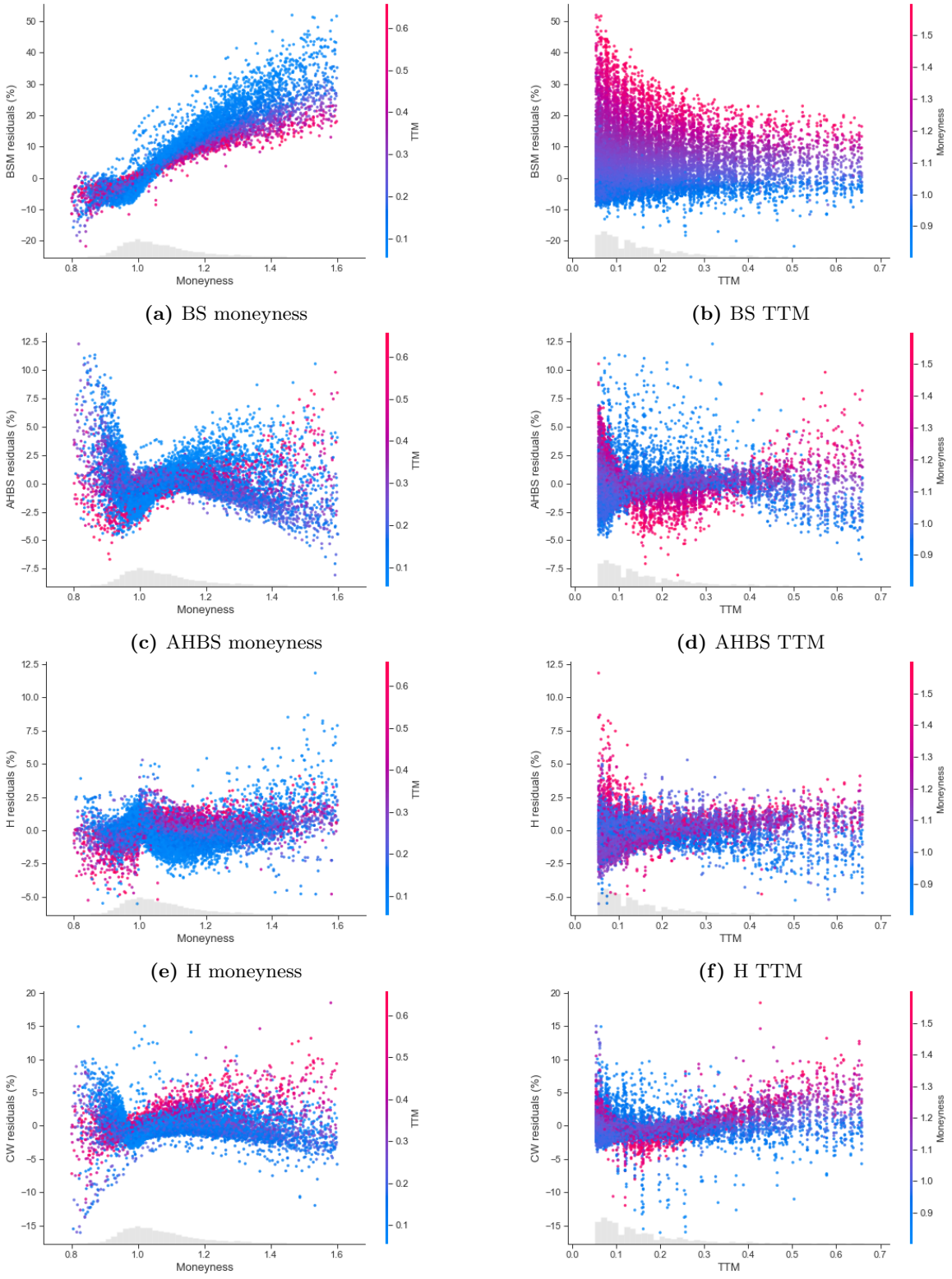
Param.	ML Features	Param.	SVR	RF	XGB	RA NN	BA NN	W NN	D NN
BS	base	10.737	1.825	1.576	**1.130**	1.804	1.806	1.826	1.803
	tv		1.057	0.502	0.237	0.595	0.629	**0.213**	0.243
	tv_ap		1.076	0.350	**0.191**	0.330	0.395	0.281	0.271
AHBS	base	1.581	0.972	0.913	**0.706**	0.985	0.984	0.966	0.986
	tv_op		0.834	0.225	**0.136**	0.371	0.357	0.190	0.183
	tv_ap		1.025	0.214	**0.132**	0.335	0.332	0.174	0.181
H	base	1.490	1.242	1.417	1.364	1.446	1.448	**1.232**	1.447
	tv_op		1.186	1.098	**1.065**	1.169	1.137	1.192	1.137
	tv_ap		1.188	1.143	1.084	1.136	**1.075**	1.092	1.081
CW	base	1.906	1.357	1.298	**0.942**	1.338	1.356	1.337	1.344
	tv_op		0.848	0.204	**0.128**	0.366	0.378	0.161	0.215
	tv_ap		0.994	0.201	**0.129**	0.327	0.351	0.179	0.203
COMB	comb_base_ap		0.481	0.455	**0.203**	0.318	0.399	0.483	0.295
	comb_tv_ap		0.622	0.453	**0.197**	0.298	0.376	0.361	0.296

(b) val

Param.	ML Features	Param.	SVR	RF	XGB	RA NN	BA NN	W NN	D NN
BS	base	8.855	1.691	2.012	2.005	**1.655**	1.664	1.682	1.685
	tv		1.467	0.345	**0.153**	0.612	0.677	0.365	0.307
	tv_ap		1.253	0.239	**0.137**	0.352	0.375	0.220	0.263
AHBS	base	1.369	0.886	1.024	1.098	0.894	0.894	**0.871**	0.889
	tv_op		0.921	0.141	**0.095**	0.354	0.344	0.102	0.170
	tv_ap		0.885	0.132	**0.095**	0.319	0.316	0.129	0.213
H	base	1.247	1.087	1.086	1.198	1.066	1.071	1.062	**1.059**
	tv_op		1.013	0.640	**0.637**	0.765	0.799	0.704	0.718
	tv_ap		1.030	0.732	**0.650**	0.771	0.742	0.664	0.728
CW	base	1.803	1.307	**1.196**	1.479	1.228	1.245	1.204	1.278
	tv_op		1.140	0.132	**0.097**	0.367	0.366	0.108	0.201
	tv_ap		1.372	0.127	**0.096**	0.332	0.342	0.140	0.196
COMB	comb_base_ap		0.563	0.417	**0.164**	0.293	0.346	0.233	0.337
	comb_tv_ap		0.566	0.415	**0.160**	0.293	0.354	0.218	0.208

(c) test

J Scatter Plots of Parametric Model Residuals



(a) BS moneyness **(b)** BS TTM

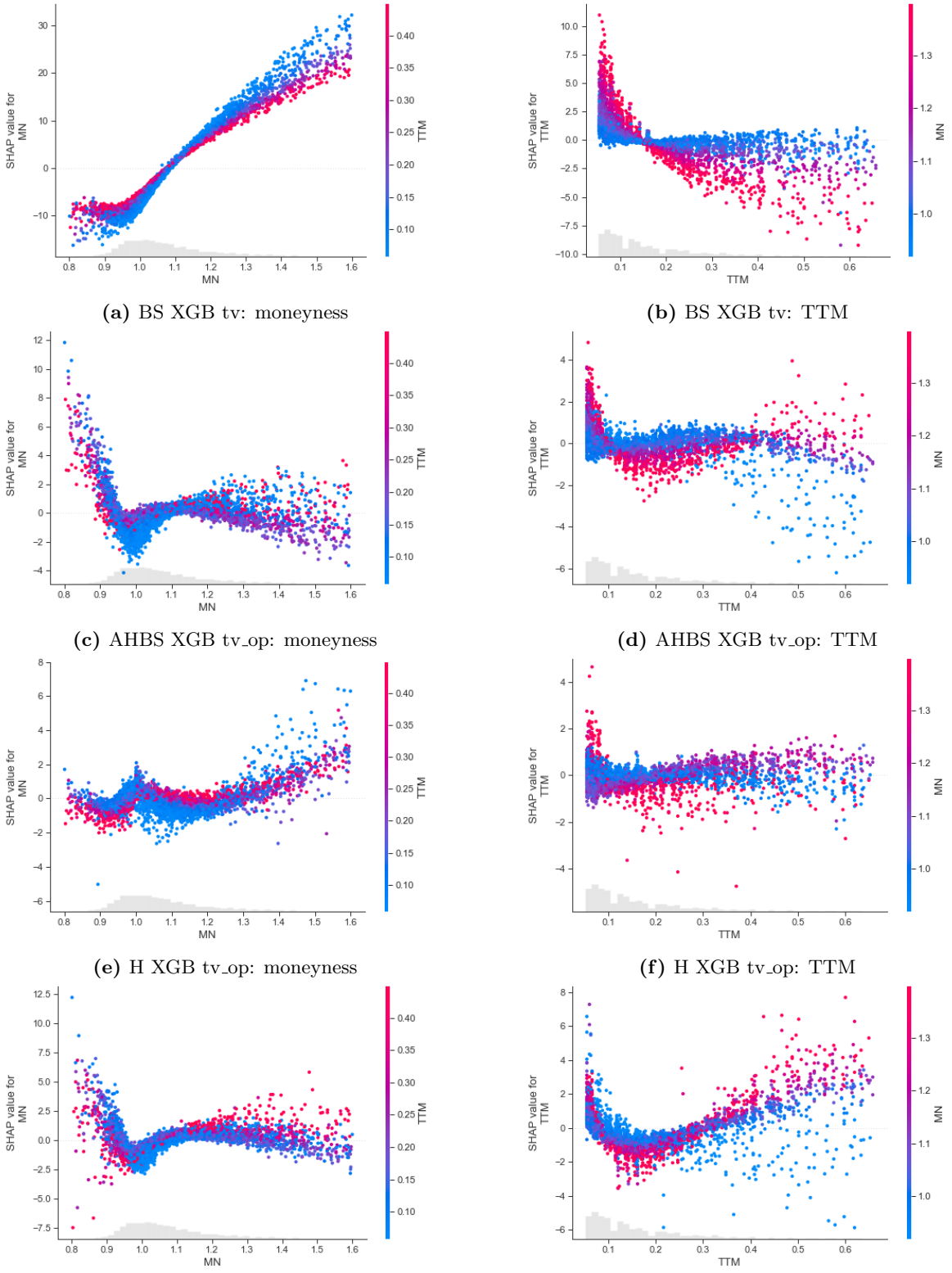
(c) AHBS moneyness **(d)** AHBS TTM

(e) H moneyness **(f)** H TTM

(g) CW moneyness **(h)** CW TTM

Figure J.1: Scatter plots for each parametric model's residuals along moneyness and TTM, coloured by respectively TTM and moneyness. Note that a subset of the data is used for the sake of clarity.

K SHAP Scatter Plots of Multiday Cross-Section Interpolation XGB Models



(a) BS XGB tv: moneyness

(b) BS XGB tv: TTM

(c) AHBS XGB tv_op: moneyness

(d) AHBS XGB tv_op: TTM

(e) H XGB tv_op: moneyness

(f) H XGB tv_op: TTM

(g) CW XGB tv_op: moneyness

(h) CW XGB tv_op: TTM

Figure K.1: Scatter plots of moneyness or TTM against SHAP values (in %), coloured by respectively TTM or moneyness for each parametric model's XGB model with either tv or tv_op features. Note that a subset of the data is used for the sake of clarity.

L Grid Search Results for Multiday Cross-Section Reusability

Table L.1: Optimal hyperparameters according to grid search for the different ML models and NN architecture sets for Multiday Cross-Section Reusability.

Param.	Model Hyperparam Features	RF max depth	n_estimators	XGB max depth	n_estimators	eta	SVR C	epsilon	gamma	RA NN Arch.	BA NN Arch.	W NN Arch.	D NN Arch.
BS	base	10	200	6	200	0.1000	41.6275	0.0001	0.0101	RA NN1	BA NN2	W NN4	D NN4
	tv	10	100	12	200	0.1000	0.0101	0.0001	0.0101	RA NN5	BA NN4	W NN3	D NN1
	tv_ap	24	500	6	500	0.1000	2.5166	0.0010	0.0101	RA NN5	BA NN4	W NN3	D NN1
AHBS	base	10	100	6	100	0.1000	0.0101	0.0001	2.5166	RA NN2	BA NN4	W NN3	D NN4
	tv_op	24	100	6	100	0.1000	0.0101	0.0010	0.0101	RA NN2	BA NN1	W NN4	D NN3
	tv_ap	24	500	24	500	0.1000	0.0101	0.0001	0.0101	RA NN1	BA NN4	W NN4	D NN3
CW	base	8	100	6	100	0.1000	41.6275	0.0001	0.0101	RA NN1	BA NN5	W NN4	D NN1
	tv_op	24	100	6	100	0.1000	0.0101	0.0001	0.0101	RA NN5	BA NN4	W NN4	D NN2
	tv_ap	24	200	6	500	0.1000	0.0101	0.0010	0.0101	RA NN1	BA NN4	W NN1	D NN4
H	base	6	500	6	500	0.0100	0.0101	0.0001	2.5166	RA NN3	BA NN5	W NN3	D NN3
	tv_op	12	500	6	100	0.1000	0.0101	0.0010	0.0101	RA NN3	BA NN3	W NN4	D NN3
	tv_ap	8	100	6	100	0.1000	0.0101	0.0001	0.0101	RA NN4	BA NN5	W NN4	D NN4
COMB	comb_base_ap	24	500	6	500	0.1000	0.0101	0.0010	0.0101	RA NN4	BA NN4	W NN4	D NN4
	comb_tv_ap	24	500	6	500	0.1000	0.0101	0.0001	0.0101	RA NN5	BA NN5	W NN4	D NN3

M Train, Val. and Test Results of Multiday Cross-Section Reusability

Table M.1: Train, validation and test set RMSE scores (%) of the Multiday Cross-Section Reusability exercise.

Param.	ML features	Param.	SVR	RF	XGB	RA NN	BA NN	W NN	D NN
BS	base	9.604	2.027	1.644	**1.638**	1.844	1.833	1.796	1.799
	tv		1.534	0.839	**0.093**	0.555	0.608	0.295	0.299
	tv_ap		0.925	**0.095**	0.202	0.555	0.608	0.295	0.299
AHBS	base	1.423	0.973	0.892	**0.892**	0.988	0.959	0.954	0.921
	tv_op		1.280	**0.063**	0.362	0.509	0.676	0.100	0.159
	tv_ap		1.351	0.059	**0.040**	0.605	0.436	0.110	0.186
H	base	1.427	1.320	1.246	**1.180**	1.317	1.295	1.294	1.295
	tv_op		1.368	**0.569**	0.720	0.922	0.928	0.688	0.669
	tv_ap		1.370	0.884	0.722	0.990	0.948	0.971	**0.670**
CW	base	1.936	1.546	1.335	**1.246**	1.466	1.433	1.347	1.350
	tv_op		1.310	**0.063**	0.381	0.389	0.366	0.085	0.196
	tv_ap		1.343	**0.061**	0.168	0.692	0.513	0.098	0.186
COMB	comb_base_ap		0.975	**0.103**	0.188	0.299	0.415	0.403	0.336
	comb_tv_ap		1.058	**0.101**	0.180	0.308	0.389	0.420	0.412

(a) train

Param.	ML features	Param.	SVR	RF	XGB	RA NN	BA NN	W NN	D NN
BS	base	10.611	**1.858**	2.036	1.990	1.978	2.008	2.008	2.010
	tv		2.329	2.054	**2.002**	2.425	2.178	2.023	2.300
	tv_ap		2.173	1.861	**1.714**	2.425	2.178	2.023	2.300
AHBS	base	1.769	1.227	1.234	1.199	**1.170**	1.209	1.243	1.217
	tv_op		1.617	0.833	**0.808**	1.467	1.444	1.237	2.059
	tv_ap		1.739	0.871	**0.782**	1.037	0.977	1.029	1.127
H	base	1.170	**0.834**	0.918	1.020	0.900	0.870	0.929	0.844
	tv_op		0.963	**0.766**	0.813	1.035	0.819	1.240	0.983
	tv_ap		0.969	**0.831**	0.902	0.909	0.949	1.144	0.983
CW	base	1.681	**1.155**	1.333	1.316	1.243	1.205	1.325	1.316
	tv_op		1.425	1.008	1.270	1.172	1.111	**0.904**	1.281
	tv_ap		1.503	**0.866**	1.131	1.614	1.490	1.030	1.195
COMB	comb_base_ap		1.473	0.787	**0.753**	1.079	1.216	2.512	3.145
	comb_tv_ap		1.813	0.817	**0.742**	2.129	1.759	2.306	3.024

(b) val

Param.	ML features	Param.	SVR	RF	XGB	RA NN	BA NN	W NN	D NN
BS	base	9.691	3.091	1.614	1.626	1.620	1.587	1.660	**1.564**
	tv		1.905	**1.480**	1.490	1.883	1.680	1.609	1.710
	tv_ap		1.746	1.134	**0.998**	1.883	1.680	1.609	1.710
AHBS	base	1.464	0.783	0.763	0.760	0.811	0.764	**0.750**	0.766
	tv_op		1.214	0.602	**0.574**	1.627	0.796	0.821	0.875
	tv_ap		1.301	**0.500**	0.502	0.739	0.753	0.892	0.947
H	base	1.172	**0.908**	0.954	1.027	0.949	0.925	0.925	0.942
	tv_op		1.024	0.917	**0.907**	1.050	1.002	1.704	1.204
	tv_ap		1.014	0.914	**0.880**	1.357	1.410	1.362	1.583
CW	base	1.656	1.424	0.906	0.898	0.969	**0.893**	0.919	0.925
	tv_op		1.166	0.686	**0.641**	0.788	0.783	0.753	1.020
	tv_ap		1.240	0.632	**0.606**	0.827	0.870	0.877	1.109
COMB	comb_base_ap		0.747	0.548	**0.493**	0.755	0.699	1.565	1.583
	comb_tv_ap		1.314	0.550	**0.486**	0.839	0.943	1.615	1.513

(c) test

N Neural Network Architectures for Daily Cross-Section

The next few appendices cover the different NN architecture sets that were used during the modelling of the Daily Cross-Section exercise. Each diagram includes the type of layer, followed by the activation function when relevant, after which the input and output shapes are mentioned, and lastly the possible regularization parameters are listed where 'MN' denotes max norm. Note that a max norm of infinity or an L1, L2 or dropout rate of 0 corresponds with that type of regularization not being applied at all.

N.a Sigmoid Almeida et al. (2022) Architectures

The architectures below follow the proposed geometric pyramid rule of Masters (1993) as implemented by Almeida et al. (2022) with sigmoid activation functions.

The used training settings are:

- Initialization: Glorot Uniform
- Batch size: Entire cross-section
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 100
- Settings left unspecified imply `TensorFlow v2.9.1` defaults

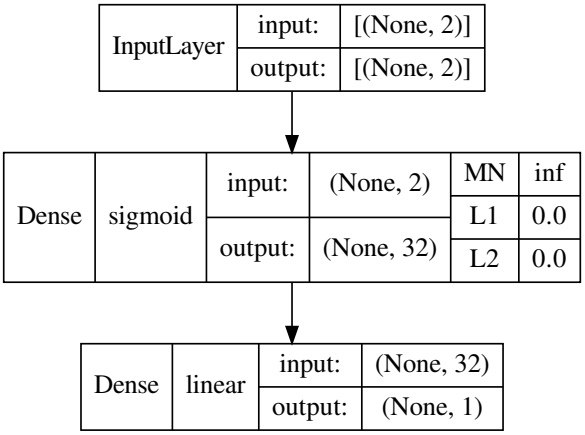


Figure N.a.1: SA NN1

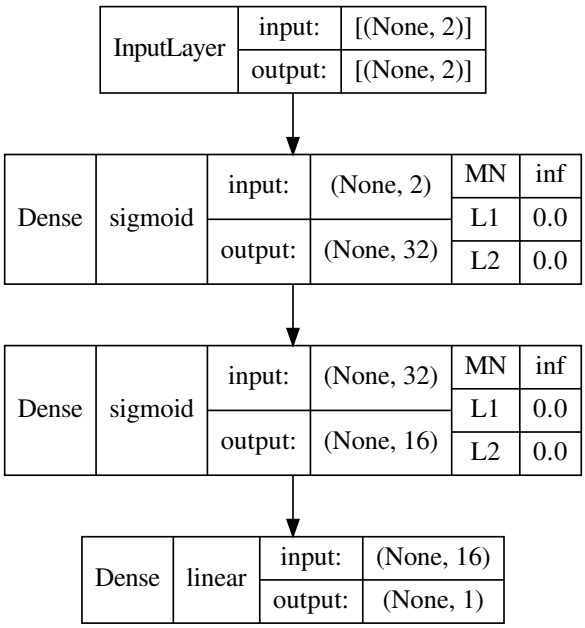


Figure N.a.2: SA NN2

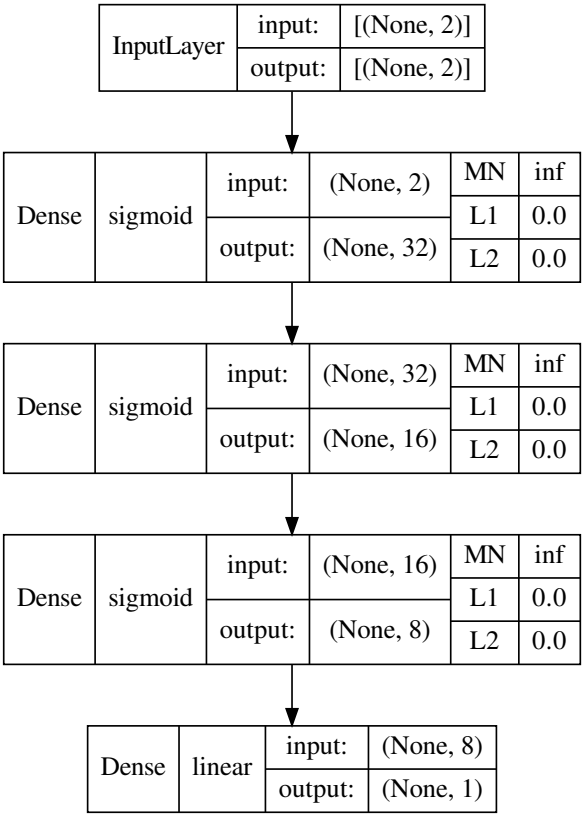


Figure N.a.3: SA NN3

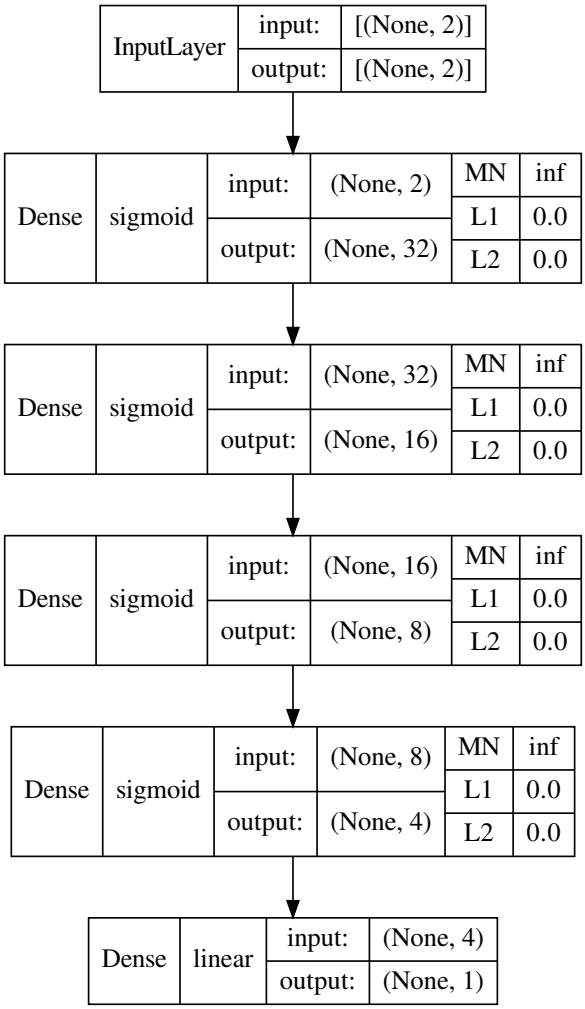


Figure N.a.4: SA NN4

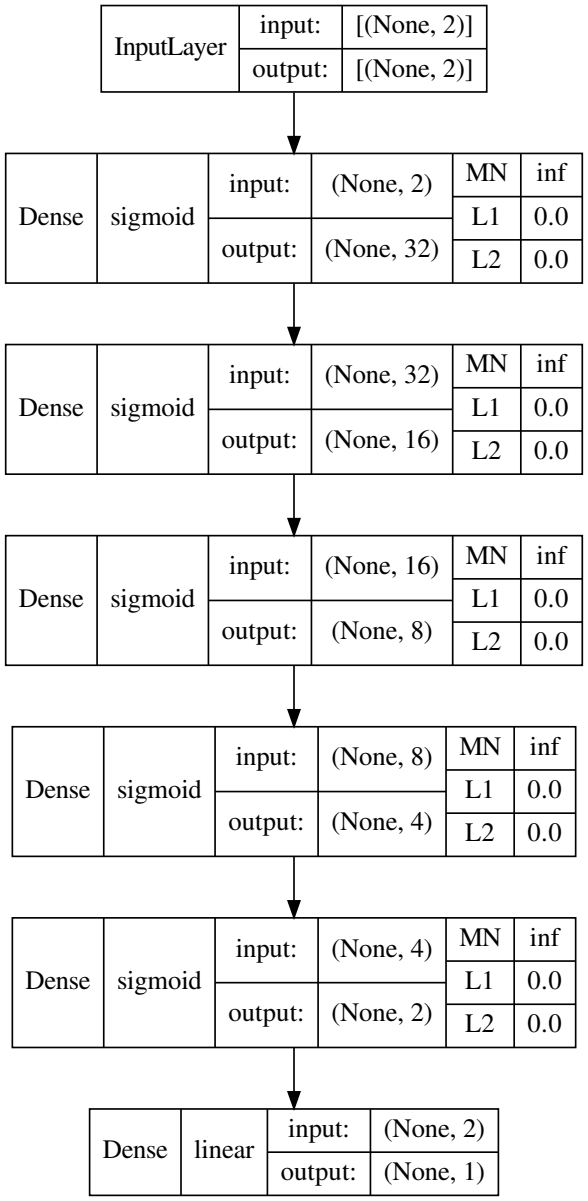


Figure N.a.5: SA NN5

N.b ReLU Almeida et al. (2022) Architectures

The architectures of this set are identical to the ones listed in Appendix N.a, with as only difference the replacement of the sigmoid activation functions by ReLU functions.

The used training settings are:

- Initialization: He Normal
- Batch size: Entire cross-section
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 100
- Settings left unspecified imply `TensorFlow v2.9.1` defaults

N.c BatchNorm Almeida et al. (2022) Architectures

The architectures of this set are identical to the ones listed in Appendix N.b, with as only difference the addition of a batch normalization layer after each first hidden layer. Note that increasing the number of batch normalization layers or placing them elsewhere did not improve the performance further.

The used training settings are:

- Initialization: He Normal
- Batch size: 256
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 100
- Settings left unspecified imply `TensorFlow v2.9.1` defaults

N.d Wider Architectures

The architectures below explore the benefits of using wider networks, i.e. with a strongly increased number of neurons for the initial hidden layer, which are then quickly reduced as we move deeper in the network.

The used training settings are:

- Initialization: He Normal
- Batch size: Entire cross-section
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 50
- Settings left unspecified imply `TensorFlow v2.9.1` defaults

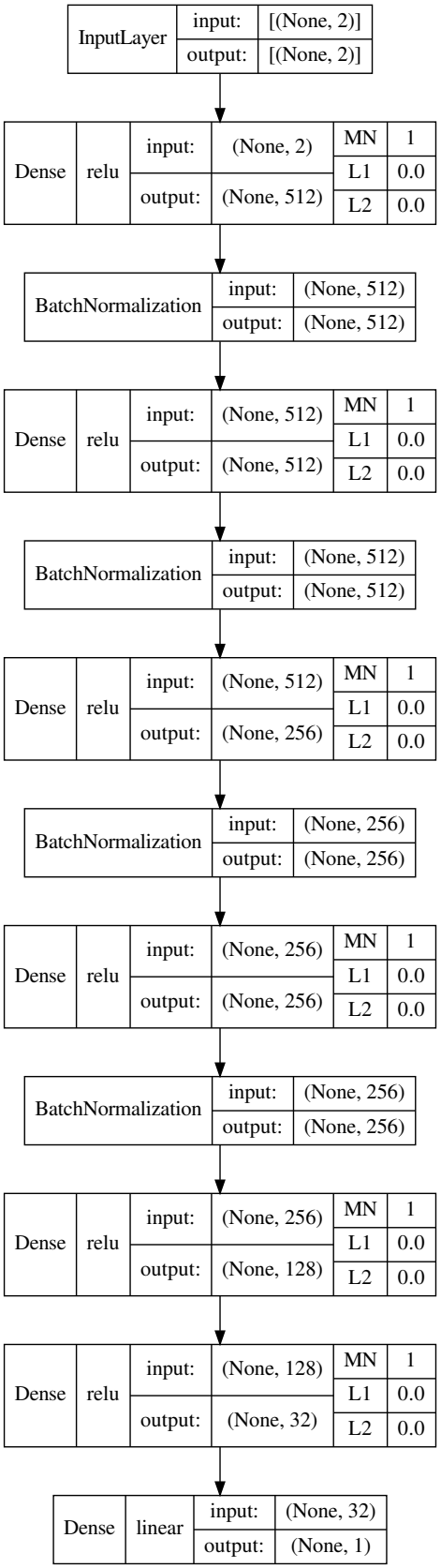


Figure N.d.1: W NN1

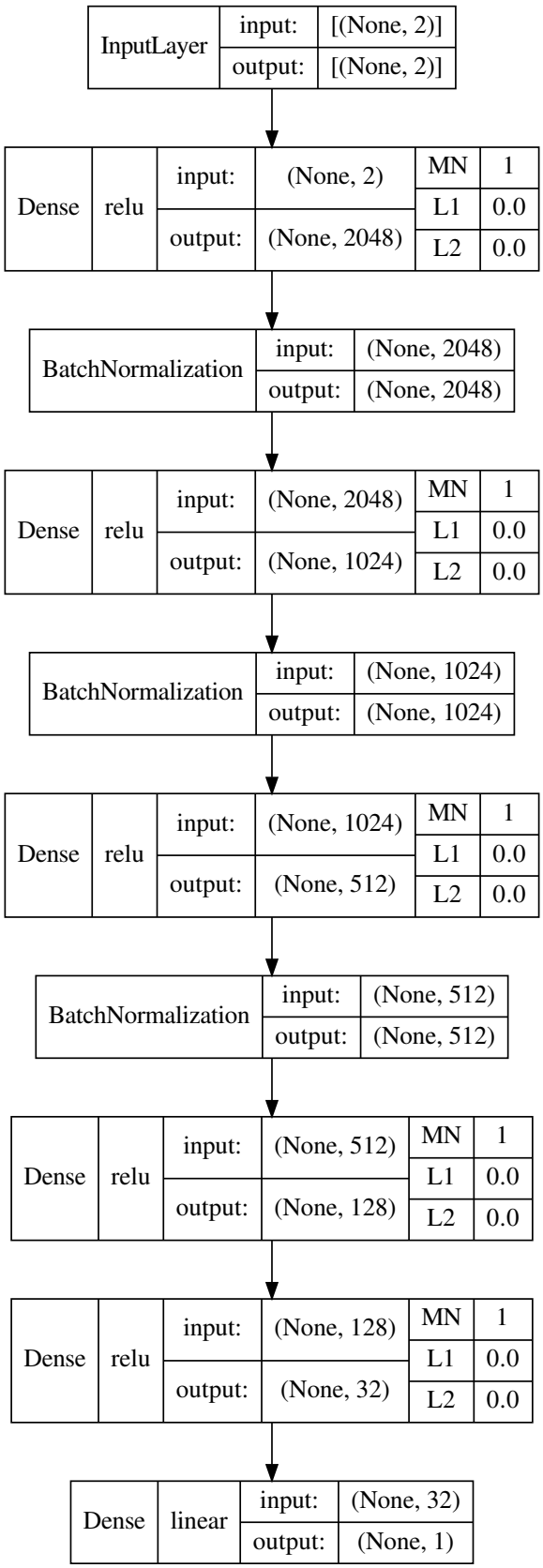


Figure N.d.2: W NN2

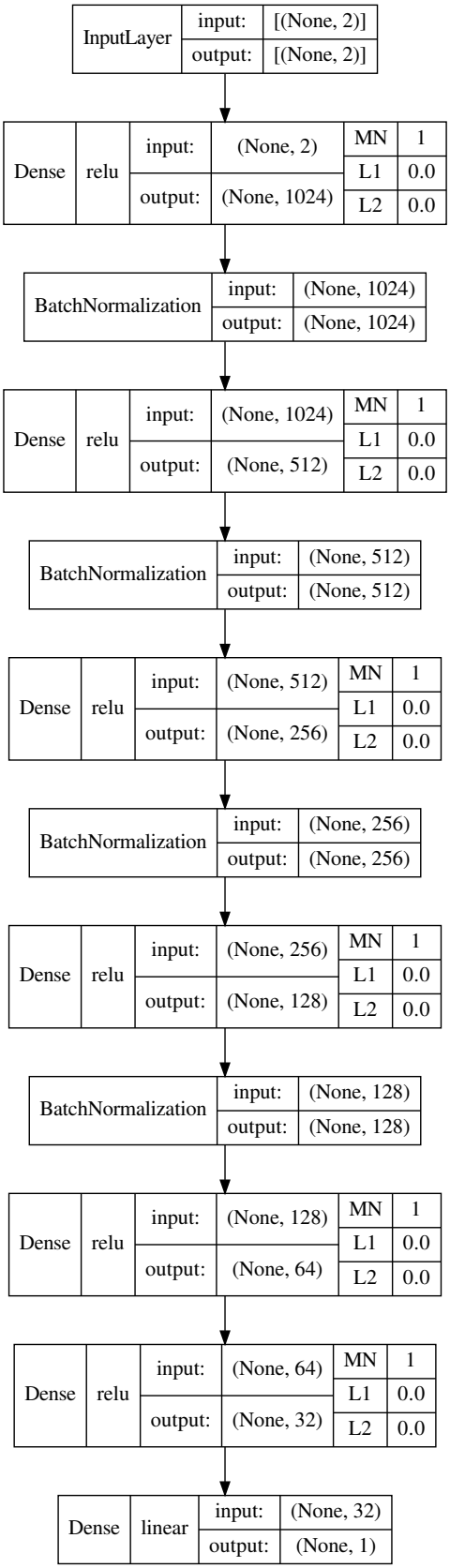


Figure N.d.3: W NN3

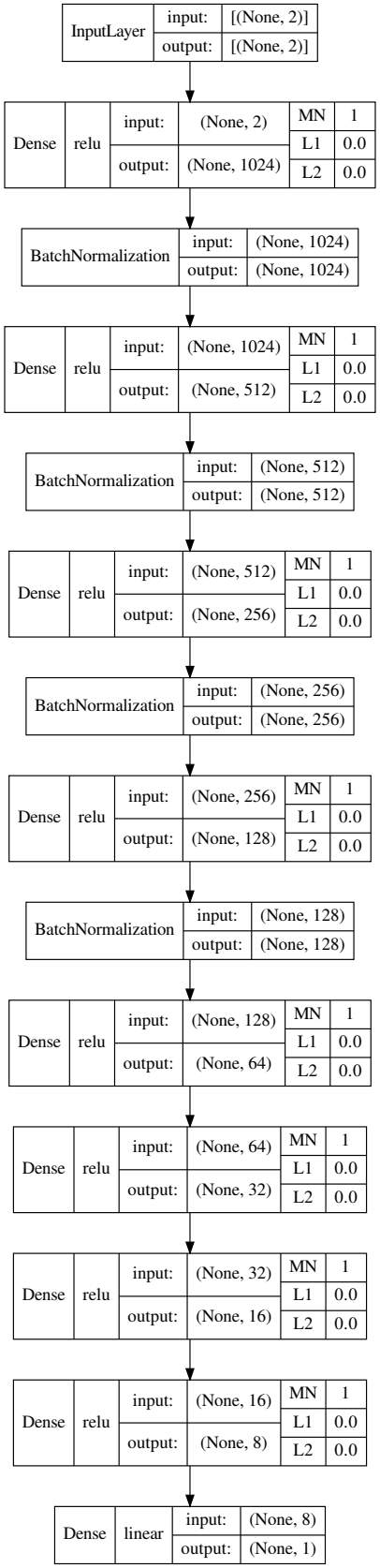


Figure N.d.4: W NN4

N.e Deeper Architectures

The architectures below explore the benefits of using deeper networks, i.e. with an increased number of hidden layers. We implement this here by increasing the number of subsequent hidden layers with 64 neurons, to keep the parameterization somewhat limited. These 64 neuron layers are succeeded by batch normalization layers as this proved to be beneficial on the validation data in initial test runs.

The used training settings are:

- Initialization: He Normal
- Batch size: Entire cross-section
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 50
- Settings left unspecified imply `TensorFlow v2.9.1` defaults

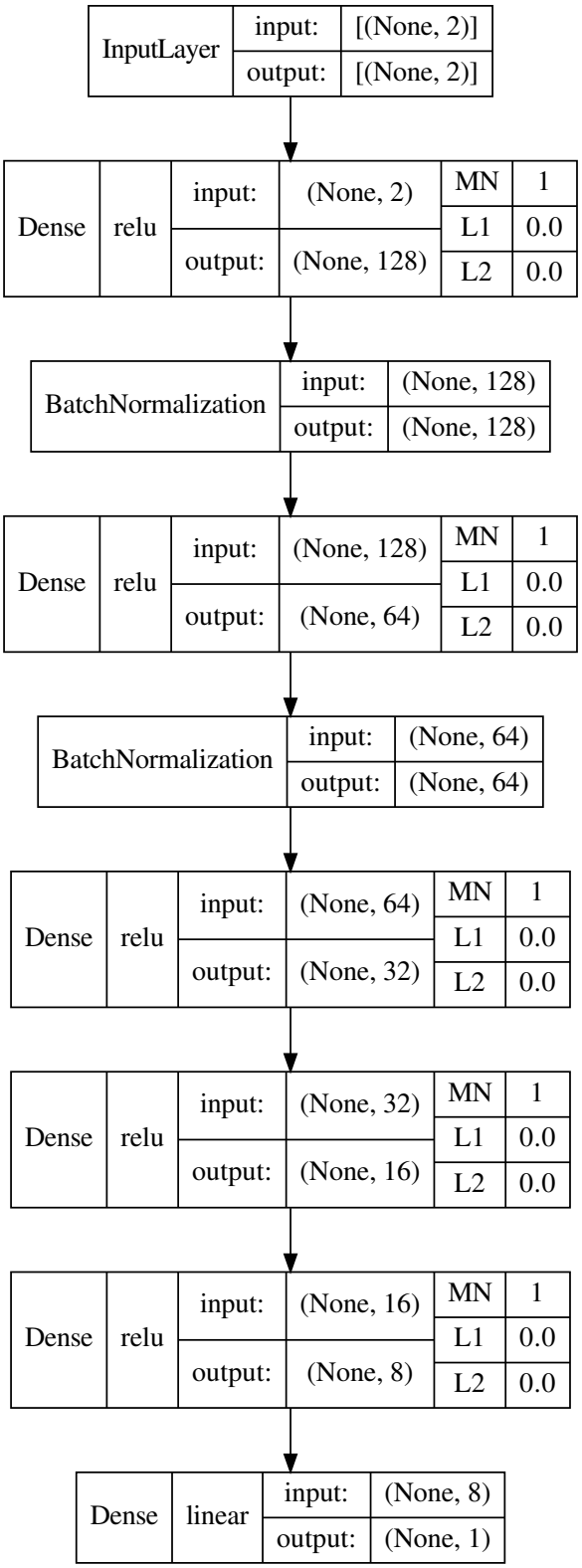


Figure N.e.1: D NN1

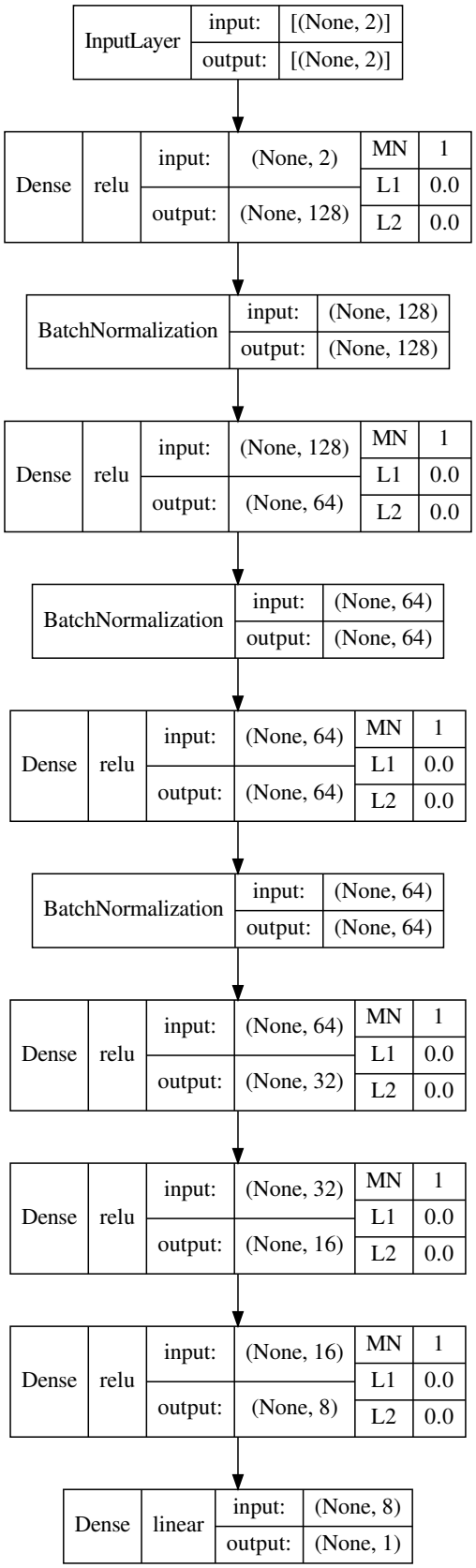


Figure N.e.2: D NN2

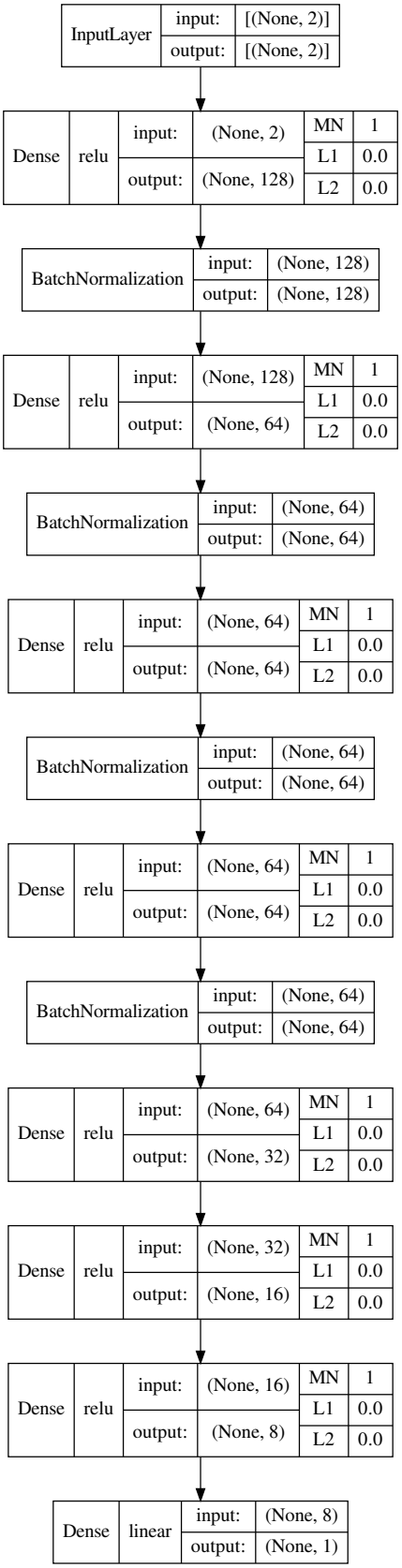


Figure N.e.3: D NN3

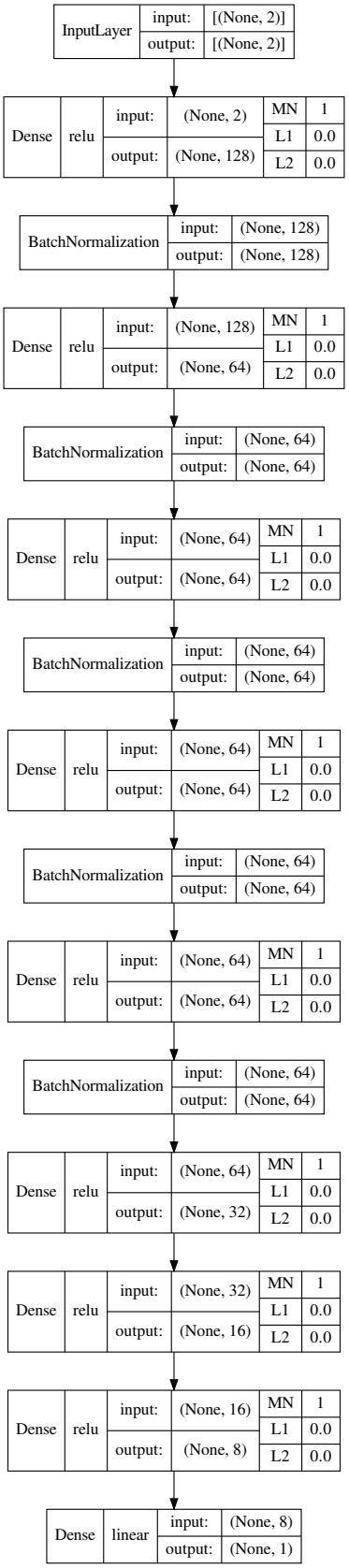


Figure N.e.4: D NN4

O Neural Network Architectures for Multiday Cross-Section Interpolation and Reusability

The next few appendices cover the different NN architecture configurations that were used during the modelling of the Multiday Cross-Section exercises. These architecture sets are largely the same as during Appendix N, but can differ in the specification of the input layer. Whereas during the Daily Cross-Section we were limited to solely using moneyness and TTM as inputs, i.e. a dimensionality of 2, this can now be extended to include time-varying covariates. Hence, when we state that the architecture set is the same as a previously listed set, this excludes the input layer. The exact input used depends on the configuration of the empirical experiment. Settings that for example solely use the base features would imply the same dimensionality of 2, models with the tv feature set would imply a dimensionality of 9 (moneyness, TTM, and the seven time-varying covariates), and so on.

O.a ReLU Almeida et al. (2022) Architectures

This architecture set is identical to the ones listed in N.b, with as sole differences the input dimensions and training settings.

The used training settings are:

- Initialization: He Normal
- Batch size: 256
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 50
- Settings left unspecified imply `TensorFlow v2.9.1` defaults

O.b BatchNorm Almeida et al. (2022) Architectures

This architecture set is identical to the ones listed in N.c, with as sole differences the input dimensions and training settings.

The used training settings are:

- Initialization: He Normal
- Batch size: 256
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 50
- Settings left unspecified imply `TensorFlow v2.9.1` defaults

O.c Wider Architectures

This architecture set is identical to the ones listed in N.d, with as sole differences the input dimensions and training settings.

The used training settings are:

- Initialization: He Normal
- Batch size: 5096
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 50
- Settings left unspecified imply `TensorFlow v2.9.1` defaults

O.d Deeper Architectures

This architecture set is identical to the ones listed in N.e, with as sole differences the input dimensions and training settings.

The used training settings are:

- Initialization: He Normal
- Batch size: 2048
- Optimizer: Adam
- Learning rate: 0.001
- Epochs: 1000
- Patience: 50
- Settings left unspecified imply `TensorFlow v2.9.1` defaults