Predicting Hold-until-Maturity Option Returns with Machine Learning Methods in a Controlled Environment

Bart Verver and Arnaud Kunkel

Master's Thesis in Finance

Stockholm School of Economics

Abstract: We investigate the performance in predicting hold-until-maturity option returns of a selection of Machine Learning methods, consisting of penalized linear regressions, dimensionality-reduction techniques, and non-linear tree-based regressions. Assuming a factor model for stock returns, we simulate data by drawing from fitted empirical distributions. We find that non-linear models outperform linear models in predicting option holding period returns and observe that this result persists with increased complexity in the assumed stock return dynamics. A trading strategy based on the predictions of the non-linear ensemble generates an annualized Sharpe Ratio of 1.1.

Keywords: Option Returns, Simulation Study, Machine Learning, Regression Trees, Return Forecasting.

Supervisor: Tobias Sichert, Assistant Professor of Finance at the Stockholm School of Economics.

Contents

1 Introduction

Machine Learning models have shown to be powerful prediction tools for complex, non-linear, and high-dimensional data (Friedman et al., 2008). First applied by Samuel (1959) in an attempt to beat a human player in a game of checkers, such models have developed significantly with the recent advancements in cloud computing and the ever-increasing availability of data. Already today, Machine Learning methods are commonly employed in health care, manufacturing, financial modeling, and more (Jordan and Mitchell, 2015).

Given its popularity, Machine Learning has increasingly become a topic of interest for financial research, with fields of research ranging from risk forecasting to structural financial market analysis (Aziz et al., 2022). Particularly its application to empirical asset pricing is attractive. Firstly, Machine Learning methods are ideally suited for handling high-dimensional data, and therefore for predicting asset returns. There have been a large number of possible factors proposed for explaining risk premiums (Welch and Goyal, 2008; Green et al., 2012). Many of these predictors are correlated. Conventional estimation methods struggle to capture the correlations among predictors. On the contrary, Machine Learning methods are built to handle high-dimensional, correlated data by selecting only relevant predictors (Friedman et al., 2008). Therefore, they are attractive as estimation techniques. Second, conventional estimation methods require the functional form of the return model to be explicitly specified. If the hypothesized relation is non-linear, specifying interactions rapidly expands the predictor set (Gu et al., 2020). Tree-based Machine Learning methods, on the other hand, offer a non-parametric, non-linear approach to mapping complex non-linear relations; a functional form does not need to be specified. Lastly, the ability to impose constraints and penalize Machine Learning models helps to prevent identifying spurious relations and limits over-fitting.

In this paper, we analyze the performance of Machine Learning models in predicting option hold-until-maturity returns, or option holding period returns, following the Machine Learning framework of Gu et al. (2020) and Bali et al. (2023). Our objective is to determine whether Machine Learning methods can learn complex, non-linear relationships often observed in empirical asset pricing, and to identify which methods perform best under which conditions. We conduct a simulation study, as this is the only setting in which the true generating process of the option returns is known. This allows us to observe

the maximum achievable performance, against which we can benchmark the candidate models. We distinguish between linear and non-linear classes of models, and two respective ensembles. The linear models consist of Lasso Regression, Ridge Regression, Elastic Net Regression (ENet), Principal Component Regression (PCR), and Partial Least Squares (PLS). The non-linear class consists of three tree-based models, namely Random Forest Regression, Gradient Boost Regression, and Dropout meets multiple Additive Regression Trees (Dart). We perform the analysis on simulated data instead of actual data. Using simulated data allows us to benchmark the performance of the candidate models and to dissect their ability to predict option holding period returns under different conditions. We assume a ten factor model for the returns of the underlying stock, sampled according to fitted historical distributions. Assuming normally distributed factor returns allows us to use the Black-Scholes closed-form solution for the price of an option to compute the expected holding period return. After simulating realized returns, we compare the out-of-sample performance of the candidate models.

First, our base simulation study consists of monthly factor and call option data for a thousand hypothetical periods and firms. We train the set of candidate models on return-relevant data to predict the realized option holding period return. As stock derivatives, the payoff of an option is directly tied to the return of the underlying stock. Because of that, the parameters of the stock return distribution contain most information about the expected option return. Thus, candidate models are trained on a subset of data including the two parameters defining the stock return distribution, and the moneyness of the call option. We evaluate performance based on out-of-sample R^2_{OOS} and compare it to the predictive power of the Black-Scholes hold-until-maturity return expectation, serving as an upper boundary. In addition, we analyze the performance of the candidate models' performance on three different sorts, namely expected stock return, stock return variance, and moneyness.

Second, we analyze the effects of several alterations to the base model. We implement imperfect information by making changes to the training set of the candidate models. By removing one predictor from the training data at a time, we can observe the individual feature importance of the three predictors. Moreover, we analyze the effect of replacing the expected stock return with the ten factor-betas used to compute the expected stock return. To analyze the relative importance of the time dimension over the firm dimension

in the data, we alter the ratio between the number of periods and the number of firms. Further, we add a case where there are two option contracts with different strike prices per firm and period. Lastly, we analyze the effect of predicting put options instead of call options.

Third, we implement a simple trading strategy based on the predictions of the candidate models. This allows us to estimate the economic gains from using Machine Learning forecasts. Each month, the portfolio is constructed by buying the decile of option contracts with the highest predicted return and writing the contracts with the lowest predicted return. We measure the performance of the strategy over a period of ten years in terms of the Sharpe Ratio. Furthermore, we provide the Sharpe Ratios of the non-linear ensemble strategy across three sorts, namely moneyness, expected stock return, and stock return variance.

Fourth, we account for the fact that the Black-Scholes model comes with certain shortcomings. Therefore, we alter the stock dynamic to allow for random jumps in the stock prices. This results in skewness and kurtosis in the stock returns. We use the Merton Jump Diffusion model to calculate the expected option holding period return. The models are trained on the moneyness and the first four moments of the new stock return distribution. To check the feature importance of the prediction models, we apply the same feature-removing procedure as in the base model.

We contribute to the current research by dissecting the performance of Machine Learning models in a controlled environment. Our research is most closely aligned with the work of Bali et al. (2023) in that we compare the performance of a similar set of candidate models. However, Bali et al. (2023) study the predictability of delta-hedged option returns, whereas we aim to predict the hold-until-maturity return of options. We believe that this is insightful as hold-until-maturity returns of options are potentially harder to predict due to their more pronounced skewed and leptokurtic nature (Zhan et al., 2022). Instead of using actual data, we simulate data based on historical distributions. This creates a controlled environment with perfect data quality, similar to a laboratory in which all parameters are known, allowing us to precisely isolate the effect of changes to the simulated data set. Further, by assuming the Black-Scholes option pricing model, we can calculate the true expected return in closed form, which serves as an upper boundary

for the performance of the candidate models. This allows us to precisely benchmark the performance of the models.

We find that the best non-linear model outperforms all linear models under the Black-Scholes assumption. An ensemble consisting of the tree-based models outperforms a linear ensemble. Further, we find that the performance is decreasing in moneyness and stock variance, while increasing in expected stock return. Different simulations show that the expected stock return is the most important predictor and that the time dimension contains more information than the firm dimension. In addition, we find that a long-short trading strategy based on Machine Learning predictions earns an annualized Sharpe Ratio of 1.1. Lastly, we note that under increased complexity in the stock price dynamics, the non-linear ensemble continues to outperform the linear ensemble.

The rest of this paper is structured as follows: Section 2 reviews the existing literature on the topic. Section 3 explains each step of our methodology in detail, outlining the simulation study framework and the Machine Learning methods. Section 4 briefly describes the empirical data used to construct the simulation. In Section 5 we report the main findings and in Section 6 we extend our analysis to the Merton Jump Diffusion model. Lastly, we conclude by contrasting our results to the existing literature in Section 7.

2 Literature Review

Research on stock options has been initiated by Bachelier (1900) and revolutionized by Black and Scholes (1973). Since then, a great deal of research in finance has been devoted to extending the Black-Scholes option pricing model. Influential models include Merton (1976) and Heston (1993). Another body of financial research examines the properties of the returns of options. Coval and Shumway (2001) find that the holding return of a call option exceeds that of the underlying and increases with the strike price under mild assumptions. The authors observe that returns of S&P 500 index call options exhibit these characteristics. In addition, they find that holding returns on zero-delta straddles are negative and hypothesize that shocks to market volatility are priced. Büchner and Kelly (2022) investigate index options and aim to explain the cross-section of option returns. The authors propose a latent three factor model with time-varying factor loadings and are able to explain a significant share of the cross-sectional variation in option returns.

Merton et al. (1978, 1982) simulate the returns of several option trading strategies and find that they produce return distributions different from those attainable by only investing in stocks and bonds. Goyal and Saretto (2009) sort options on the difference between implied and realized historical volatility. Based on these sorts, they construct delta-neutral long-short portfolios and observe significant returns.

The current literature on the application of Machine Learning in finance and asset pricing has mostly been focused on predicting equity returns. Rapach et al. (2013) apply Lasso Regressions to analyze cross-country stock predictability because of its shrinking- and selecting-properties. They find that lagged U.S. stock returns can predict stock returns in non-U.S. countries. In contrast, they find limited evidence for the inverse. Moritz and Zimmerman (2016) use tree-based methods to analyze the predictability of the cross-section of future equity returns. The authors average over multiple tree-based portfolio sorts following the literature on Random Forests by Breiman (2001). They find that short-term returns are the best predictors for forecasting future returns. Kelly et al. (2019) develop an extension of Principal Component Analysis to analyze the cross-section of returns, namely IPCA.[1] This method instruments observable characteristics for the unobservable loadings from PCA. They find a low-dimension factor model sufficiently explaining the riskiness and risk compensation of stock returns that outperforms existing observable factor models such as the Fama French Five-Factor Model. Freyberger et al. (2020) use Group Lasso to estimate the effect of stock characteristics on expected returns in a controlled environment. Gu et al. (2020) apply an entire framework of Machine Learning models to the problem of measuring equity risk premiums. The authors use 12 different model specifications, with more than half consisting of complex non-linear models. Methods used include Random Forest Regressions, Gradient Boost Regressions, and Neural Networks with up to 5 layers. They find that there are large economic gains in return forecasting when using a Machine Learning approach. In addition, they observe that tree-based models and neural networks are the methods with the highest predictive power.

Recently, there has been more literature regarding the application of Machine Learning methods to other asset classes. Kelly et al. (2023) apply their earlier developed IPCA framework to corporate bond returns. Similarly, they propose a five-factor model and find

[1]The methodology of IPCA is described in detail in Kelly et al. (2020).

that it outperforms existing models in explaining the returns of bonds. Bali et al. (2020) analyze the predictability of corporate bond returns using a variety of Machine Learning methods and Big Data following the framework of Gu et al. (2020). They use eight Machine Learning methods including Penalized Linear Regressions, dimension reduction techniques, Random Forest Regressions, and Feed-Forward Neural Networks. The authors find that Machine Learning methods can significantly improve the out-of-sample predictive power of bond and stock characteristics in forecasting bond returns. Bali et al. (2023) analyze the predictability of option returns following the framework of Bali et al. (2020). In addition to the methods named earlier, the authors introduce Gradient Boost Regressions and Dart. They find that introducing non-linear models significantly improves the out-of-sample predictive power of stock and option characteristics in forecasting option returns. In addition, they find that trading strategies based on the predictions of the non-linear Machine Learning models can generate significant Sharpe Ratios.

Using simulated data instead of real data to analyze model performance and robustness is a common procedure in research on asset returns. For example, Broadie et al. (2009) analyze index option returns by using simulated data. The authors estimate the model parameters of the Black-Scholes and Heston option pricing model from historical data. Subsequently, they draw index realizations to compare actual option returns against the simulated finite-sample return distribution. They find that excessive option returns do not reveal mispricings. Freyberger et al. (2020) assume both a linear and a non-linear return model with thirteen characteristics and simulate returns by drawing from this model. They proceed with feeding the Lasso models a set of 62 characteristics and find that adaptive Group Lasso performs best in identifying the true characteristics of the non-linear model. Further, they study the performance of the Lasso models in predicting simulated out-of-sample returns and find that adaptive group Lasso generates the highest R^2 for the non-linear, but not the linear return model. Gu et al. (2020) perform a simulation study to demonstrate the performance of the Machine Learning framework considered in predicting stock returns. They simulate stock return time series from two latent three-factor models, in which the characteristics enter the model linearly and non-linearly. The authors report out-of-sample R^2 and find that the non-linear Machine Learning models dominate under the non-linear factor model.

3 Methodology

In this Section, we describe the procedure of simulating data and lay the theoretical foundations for the methods we use in this paper. We begin by describing the factor model and the simulation process. We then derive the holding period return of an option by introducing the Black-Scholes option pricing formula and the concept of risk-neutral valuation. Next, we discuss the rationale behind our choice of performance metric. Lastly, we give a overview of all the Machine Learning methods implemented in this paper and discuss the hyperparameter tuning procedure we employ.

3.1 Simulation Study

The general approach to simulating data is to specify the probability density function of some characteristic to be analyzed and simulate random samples from this distribution. In our case, we aim to simulate the expected hold-until-maturity return of an option. The advantage of generating artificial data over using real data is that we have full control over all distribution parameters. This approach allows us to generate large quantities of data with perfect quality, something that is often not available with real data. Especially option return data can be extremely noisy and inaccurate for less liquid contracts (Duarte et al., 2023). It is beneficial if the simulated data shares some characteristics with real data to increase the relevance of any conclusions drawn from the simulation study. As we are studying holding period returns of stock options, constructing the simulation reduces to building a realistic model of stock returns.

A common approach to modeling stock returns is to assume a factor model. Under a factor model, the return of a stock is assumed to be a linear combination of the returns of some risk factors. Factor models were made popular by Fama and French (1993). The authors aim to explain the cross-section of stock returns using three factors, namely the market return, a size factor based on market capitalization, and a value factor based on the book-to-market ratio. Since then, a myriad of alternative factor models have been proposed (Feng et al., 2020). In this paper, we assume a factor model of the following form:

$$r_{t,n} = r_t^f + \sum_k \beta_{k,n} f_{k,t} + \epsilon_{t,n}, \tag{1}$$

where $r_{t,n}$ denotes the log-return of stock n in period t, r_t^f the risk-free rate, $f_{k,t}$ the log-return on the k-th risk factor, and $\beta_{k,n}$ the individual exposure of firm n to risk factor k.[2] Note, that we assume the factor exposure to be constant over time. Furthermore, $\epsilon_{t,n}$ denotes the idiosyncratic error term, or the portion of the return that is pure noise and bears no risk premium. If the factor model describes the true data-generating process of returns, meaning that returns are fully explained by the factor model, the expected value of $\epsilon_{t,n}$ is zero. Then, the conditional expected return of a stock can be decomposed into the sum of the conditional expected returns of each risk premium:

$$\mathbb{E}_t[r_{t+1,n}] = r_{t+1}^f + \sum_k \beta_{k,n} \mathbb{E}_t[f_{k,t+1}]. \tag{2}$$

Note, that we assume the conditional expected factor return to be constant. By that, we implicitly assume that the conditional expected stock return is constant too. We obtain:

$$\mathbb{E}[r_{t+1,n}] = r_{t+1}^f + \sum_k \beta_{k,n} \mathbb{E}[f_k], \tag{3}$$

where $\mathbb{E}[f_k]$ is commonly referred to as the price of risk. The conditional variance, on the other hand, is not assumed to be constant and can be computed as:

$$\mathrm{Var}_t[r_{t+1,n}] = \sum_k \beta_{k,n}^2 \, \mathrm{Var}_t[f_{k,t+1}] + \mathrm{Var}_t[\epsilon_{t+1,n}]. \tag{4}$$

Assuming that the error terms are uncorrelated, the factor model implies the following covariance between the returns of stock p and q:

$$\mathrm{Cov}_t[r_{t+1,n}] = \sum_k \beta_{k,p} \beta_{k,q} \, \mathrm{Var}_t[f_{k,t+1}]. \tag{5}$$

Next, we describe in detail how we simulate the data that is used to train the Machine Learning models. This applies to all simulation specifications unless specified otherwise. The general structure of the simulated data under the base model can be seen in Table 1. We define T as the number of hypothetical periods considered, with corresponding period identifier t. In our analysis, we simulate monthly data. Therefore, one period refers to

[2] We assume $r_t^f = 0$ to be zero in the simulation study. When fitting the factor model to empirical data, actual risk-free rates from $2003 - 2023$ are used.

one month. We emphasize that all indices t used going forward refer to these discrete periods, and not to continuous time. We define N as the number of hypothetical firms considered, with corresponding firm identifier n. The total number of observations will be $T * N$. Furthermore, we assume $k = 10$, mainly due to data-availability. Machine Learning models thrive on large amounts of data and predictors. For example, Bali et al. (2023) use a total of 273 characteristic features. In our analysis, we fit the parameters of our factor model to historical factor data. Thus, there must exist historical factor data for each factor we assume.

t	n	$\beta_{1,n}$	$\cdots$	$\beta_{10,n}$	$\mathbb{E}[f_1]$	$\cdots$	$\mathbb{E}[f_{10}]$	$\text{Var}_t[f_{1,t+1}]$	$\cdots$	$\text{Var}_t[f_{10,t+1}]$
1	1	$\beta_{1,1}$	$\cdots$	$\beta_{10,1}$	$\mathbb{E}[f_1]$	$\cdots$	$\mathbb{E}[f_{10}]$	$\text{Var}_1[f_{1,2}]$	$\cdots$	$\text{Var}_1[f_{10,2}]$
1	2	$\beta_{1,2}$	$\cdots$	$\beta_{10,2}$	$\mathbb{E}[f_1]$	$\cdots$	$\mathbb{E}[f_{10}]$	$\text{Var}_2[f_{1,2}]$	$\cdots$	$\text{Var}_2[f_{10,2}]$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
T	N-1	$\beta_{1,N-1}$	$\cdots$	$\beta_{10,N-1}$	$\mathbb{E}[f_1]$	$\cdots$	$\mathbb{E}[f_{10}]$	$\text{Var}_T[f_{1,T+1}]$	$\cdots$	$\text{Var}_T[f_{10,T+1}]$
T	N	$\beta_{1,N}$	$\cdots$	$\beta_{10,N}$	$\mathbb{E}[f_1]$	$\cdots$	$\mathbb{E}[f_{10}]$	$\text{Var}_T[f_{1,T+1}]$	$\cdots$	$\text{Var}_T[f_{10,T+1}]$

...

$\frac{K_{t,n}}{S}$	$\text{Var}_t[\epsilon_{t+1,n}]$	$\mathbb{E}[r_{t+1,n}]$	$\text{Var}_t[r_{t+1,n}]$	$\mathbb{E}_t[R^{\text{call}}_{t+1,n}]$	$R^{\text{call}}_{t+1,n}$
$\frac{K_{1,1}}{S}$	$\text{Var}_1[\epsilon_{2,1}]$	$\mathbb{E}[r_{2,1}]$	$\text{Var}_1[r_{2,1}]$	$\mathbb{E}_1[R^{\text{call}}_{2,1}]$	$R^{\text{call}}_{2,1}$
$\frac{K_{1,2}}{S}$	$\text{Var}_1[\epsilon_{2,2}]$	$\mathbb{E}[r_{2,2}]$	$\text{Var}_1[r_{2,2}]$	$\mathbb{E}_1[R^{\text{call}}_{2,2}]$	$R^{\text{call}}_{2,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\frac{K_{T,N-1}}{S}$	$\text{Var}_T[\epsilon_{T+1,N-1}]$	$\mathbb{E}[r_{T+1,N-1}]$	$\text{Var}_T[r_{T+1,N-1}]$	$\mathbb{E}_T[R^{\text{call}}_{T+1,N-1}]$	$R^{\text{call}}_{T+1,N-1}$
$\frac{K_{T,N}}{S}$	$\text{Var}_T[\epsilon_{T+1,N}]$	$\mathbb{E}[r_{T+1,N}]$	$\text{Var}_T[r_{T+1,N}]$	$\mathbb{E}_T[R^{\text{call}}_{T+1,N}]$	$R^{\text{call}}_{T+1,N}$

Table 1: Structure of Simulated Data
This Table shows the general structure of the simulated data following the factor model. T is the number of periods considered, and t the respective period identifier; N is the number of firms considered, and n the respective firm identifier; $\beta_{k,n}$ is the beta coefficient; $\mathbb{E}[f_k]$ is the expected factor return; $\text{Var}_t[f_{k,t+1}]$ is the conditional factor return variance; $\frac{K_{t,n}}{S}$ refers to the moneyness of the option; $\text{Var}_t[\epsilon_{t+1,n}]$ is the idiosyncratic variance; $\mathbb{E}[r_{t+1,n}]$ is the expected stock return; $\text{Var}_t[r_{t+1,n}]$ is the conditional total variance; $\mathbb{E}_t[R^{\text{call}}_{t+1,n}]$ is the conditional expected hold-until-maturity option return; and $R^{\text{call}}_{t+1,n}$ is the realized option return.

For each factor, we use a time-series of historical factor returns $f_{k,t}$ to obtain a cross-sectional distribution of $\beta_{k,n}$. Specifically, we apply a simple time-series OLS regression

on historical stock return data to obtain the empirical distribution of betas.[3] To this distribution, we fit a normal distribution and obtain $\beta_{1,n}$ to $\beta_{10,n}$ by random sampling. We set the expected factor returns, $\mathbb{E}[f_1]$ to $\mathbb{E}[f_{10}]$, to the means of the historical log-return of each factor.

Further, we assume the factor variances to follow a simple $AR(1)$ model. We pay tribute to the fact that stock returns exhibit heteroskedasticity and are time-varying (French et al., 1987). For t in $t = (2, ..., T)$, we obtain:

$$\log \text{Var}_t[f_{k,t+1}] = \phi \log \text{Var}_{t-1}[f_{k,t}] + (1 - \phi) \log \text{Var}[\hat{f}_k] + \epsilon_t, \tag{6}$$

where $\text{Var}_t[f_{k,t+1}]$ is the simulated variance for factor k and $\text{Var}[\hat{f}_k]$ is the long-run historical mean of the factor return variance. The parameter ϕ controls the speed of mean reversion and ϵ_t is the error term. The error term is drawn from a normal distribution with mean zero and standard deviation $\sigma = 5\%$. The parameter σ is not fitted to historical variance data. As our main objective is to have a time-varying variance that is mean reverting, the standard deviation σ is set to a relatively small value to ensure stability and prevent the variance estimates from fluctuating too widely. We have modeled the log-variances to ensure non-negativity, since $\exp[\log \text{Var}_t[f_{k,t+1}]] = \text{Var}_t[f_{k,t+1}] > 0$. We set $\phi = 0.5$ to make the variance process mean-reverting. The initial factor variance in period $t = 1$ is set to the historical factor return variance:

$$\log \text{Var}_1[f_{k,2}] = \log \text{Var}[\hat{f}_k]. \tag{7}$$

The idiosyncratic variances $\text{Var}_t[\epsilon_{t+1,n}]$ are randomly sampled from a log-normal distribution fitted to residuals from the time-series OLS regression, and are simulated period and firm-specific.

In our analysis, we define moneyness as $\frac{K_{t,n}}{S}$, where $K_{t,n}$ equals the strike price of the option with underlying stock n in period t, and S the underlying firm's stock price. We assume $S = 1$ and sample moneyness uniformly on the interval of $[0.9, 1.1]$.[4] Given the sampled betas, we simulate the conditional expected log-return for each firm and

[3] We perform an OLS Regression without intercept, as we assume the factor model in Equation (3) to be the true model.

[4] We decide to set the sampling boundaries of moneyness to fixed values for simplicity. We acknowledge that the moneyness interval can alternatively be set dynamically based on stock variance.

period according to Equation (3). In addition, we obtain the conditional return variance following Equation (4). We calculate the expected holding period return of an option $\mathrm{E}_t[R^{\text{call}}_{t+1,n}]$ using the Black-Scholes closed-form solution as outlined in Section 3.2. Lastly, we simulate realized betas and factor returns from the fitted distributions created earlier. This allows us to calculate a realized stock price. The realized option return is given by:

$$R^{\text{call}}_{t+1,n} = \frac{(S_{t+1,n} - K_{t,n})^+}{C_{t,n}}, \tag{8}$$

where $S_{t+1,n}$ denotes the realized stock price of firm n, and $C_{t,n}$ refers to the call option price. The option price is calculated by applying the Black-Scholes formula, which is explained in detail in Section 3.2.

3.2 Black-Scholes Option Pricing Model

The most prominent model for pricing European options is the one derived by Black and Scholes (1973). Going forward, we will refer to this model as the BS model, or simply BS. In their original paper, Black and Scholes (1973) derive the price of an option by constructing a continuously delta-hedged portfolio consisting of a stock and a bond. The resulting partial differential equation equals the heat equation, for which there is a well-known solution in physics. We outline an alternative way of deriving the Black-Scholes price of an option, namely by straightforward integration over the risk-neutral measure.

Black and Scholes (1973) assume the stock price to follow a Geometric Brownian Motion. We continue to use the notation introduced in Section 3.1 and refer to $t = 1, 2, ..., T$ as the period identifier of our simulated data. The price process of a non-dividend-paying stock of firm n can then be written as:

$$S_{t+\tau,n} = S_{t,n} e^{\left(\mu_{t,n} - \frac{1}{2}\sigma^2_{t,n}\right)(t+\tau) + \sigma_{t,n} W_{t+\tau,n}}, \tag{9}$$

where $0 \leq \tau \leq 1$. The stock price of of firm n at time t is denoted by $S_{t,n}$. In our case, $S_{t,n} = 1$. Further, $\mu_{t,n}$ denotes the instantaneous stock return, $\sigma_{t,n}$ the standard deviation of the stock returns, and $W_{t+\tau,n}$ a Wiener process with mean zero and standard deviation $\sqrt{\tau}$. From Equation (9) we see that $\mu_{t,n} = \mathbb{E}[r_{t+1,n}] + \frac{\sigma^2_{t,n}}{2}$ and $\sigma^2_{t,n} = \mathrm{Var}_t[r_{t+1,n}]$. Since $W_{t+\tau,n}$ is normally distributed, $S_{t+\tau,n}$ is log-normally distributed. This is a crucial

assumption of the Black-Scholes model. As we see in Section 6, extensions to the BS model relax this assumption.

Options cannot be priced by discounting the expected payoff under the physical probability measure P, as the discount rate is dependent on the individual risk preference of each investor. Therefore, this approach will only yield a lower boundary for the price of the option. To obtain the arbitrage free price of an option, the P-measure must be transformed to the risk-neutral probability measure Q. If stock returns are normally distributed, transforming from the P-measure to the Q-measure is equivalent to setting $\mu_{t,n} = r^f$ (Rubinstein, 1976). Under Q, the stock price process becomes a martingale, and the price of an option can be calculated by discounting the expected payoff with the risk-free rate (Cox and Ross, 1976). This leads to the Black-Scholes formula for the price of a call option:

$$C_{t,n}\left(\tau, S_{t,n}, \sigma_{t,n}, K_{t,n}, r^f\right) = S_{t,n}\Phi\left(d_1\right) - K_{t,n}e^{-r^f\tau}\Phi\left(d_2\right), \tag{10}$$

$$d_1 = \frac{\log\frac{S_{t,n}}{K_{t,n}} + \left(r^f + \frac{\sigma_{t,n}^2}{2}\right)\tau}{\sigma_{t,n}\sqrt{\tau}},$$

$$d_2 = d_1 - \sigma_{t,n}\sqrt{\tau},$$

where $K_{t,n}$ denotes the strike price of the option contract, τ the time to maturity, and $\Phi\left(\cdot\right)$ the cumulative distribution function of the standard normal distribution. Because we only consider monthly options in this paper, we have $\tau = 1$. The price of a put option is provided in Appendix A1.

By utilizing the transformation of measure, the expected hold-until-maturity return can be computed. The expected hold-until-maturity return refers to the expected return that is achieved from buying the option and holding it until maturity, and is given by:

$$\mathbb{E}_t\left[R_{t+\tau,n}^{\text{call}}\right] = \frac{\mathbb{E}_t^P\left[\left(S_{t+\tau,n} - K_{t,n}\right)^+\right]}{C_{t,n}\left(\tau, S_{t,n}, \sigma_{t,n}, K_{t,n}, r^f\right)} = \frac{e^{\mu_{t,n}\tau}C_{t,n}\left(\tau, S_{t,n}, \sigma_{t,n}, K_{t,n}, \mu_{t,n}\right)}{C_{t,n}\left(\tau, S_{t,n}, \sigma_n, K_{t,n}, r^f\right)}, \tag{11}$$

where $C_{t,n}$ is given by Equation (10), and $\mathbb{E}_t^P\left[R_{t+\tau,n}^{\text{call}}\right]$ is the expected payoff of the option under the measure P (Boyer and Vorkink, 2014). We transform the measure Q to P by

setting $r^f = \mu_{t,n}$. The holding period return for a put option can be calculated analogously and is provided in Appendix A2.

3.3 Performance Measurement

There is a plethora of performance measures that can be used to assess the predictive power of Machine Learning methods. Most metrics evaluate the predictive power in terms of the distance between the prediction and the realization. Popular metrics include the mean absolute error (MAE) and the mean squared error (MSE). As they are not bounded, these measures come with the drawback that they provide little information about the actual model performance (Chicco et al., 2021). Metrics that instead measure performance based on relative distance, such as the mean absolute percentage error (MAPE), do give a more intuitive indication of performance. However, they are not suitable if the realizations can become zero, as is the case for option returns. Given these drawbacks, numerous studies researching the application of Machine Learning in finance apply the coefficient of determination, or R^2 (Gu et al., 2020; Bali et al., 2023). An advantage of R^2 is that it is bounded by a value of 1, corresponding to a perfect fit. We follow this approach and use R^2 to measure performance in this paper. We define the out-of-sample R^2 as:

$$R^2_{OOS} = 1 - \frac{\sum_{(t,n)} \left(R_{t+1,n} - \hat{R}_{t+1,n} \right)^2}{\sum_{(t,n)} \left(R_{t+1,n} - \bar{R} \right)^2}, \tag{12}$$

where $R_{t+1,n}$ is the simple hold-until-maturity return of the option with underlying stock n at time $t+1$, $\hat{R}$ is the prediction made by the model, and $\bar{R}$ the out-of-sample mean of the returns. We use out-of-sample R^2 as it is more generalizable than in-sample R^2. We refrain from using a naive benchmark of zero as option returns are significantly different from zero (Hu and Jacobs, 2020). Thus, we benchmark against the sample mean $\bar{R}$.

The strength of a simulation study lies in the fact that we observe the true return distribution, and thus the true expected option return. This allows us to calculate the R^2_{OOS} of the true expected return under Black-Scholes, R^2_{BS}, which serves as a performance benchmark for the candidate models. Due to randomness, R^2_{BS} can vary significantly, along with the performance of the candidate models. To get a more stable indication of

R^2_{BS}, we compute the expected value of R^2_{BS}. Using Equation (12) we obtain:

$$\mathbb{E}[R^2_{BS}] = 1 - \frac{\mathbb{E}\left[\sum_{(t,n)}\left(R_{t+1,n} - \hat{R}_{t+1,n}\right)^2\right]}{\mathbb{E}\left[\sum_{(t,n)}(R_{t+1,n} - \bar{R})^2\right]} \tag{13}$$

$$= 1 - \frac{\sum_{(t,n)}\mathbb{E}_t\left[\left(R_{t+1,n} - \mathbb{E}_t\left[R_{t+1,n}\right]\right)^2\right]}{\sum_{(t,n)}\mathbb{E}_t\left[R^2_{t+1,n} - 2R_{t+1,n}\bar{R} + \bar{R}^2\right]} \tag{14}$$

$$= 1 - \frac{\sum_{(t,n)}\mathrm{Var}_t\left[R_{t+1,n}\right]}{\sum_{(t,n)}\left(\mathbb{E}_t\left[R^2_{t+1,n}\right] - 2\bar{R}\mathbb{E}_t\left[R_{t+1,n}\right] + \bar{R}^2\right)} \tag{15}$$

$$= 1 - \frac{\sum_{(t,n)}(\mathbb{E}_t\left[R^2_{t+1,n}\right] - \mathbb{E}_t\left[R_{t+1,n}\right]^2)}{\sum_{(t,n)}\left(\mathbb{E}_t\left[R^2_{t+1,n}\right] - 2\bar{R}\mathbb{E}_t\left[R_{t+1,n}\right] + \bar{R}^2\right)}, \tag{16}$$

where $\mathbb{E}_t\left[R_{t+1,n}\right]$ refers to the expected hold-until-maturity option return and is calculated according to Equation (11). The expected value of the squared option holding period returns, $\mathbb{E}_t\left[R^2_{t+1,n}\right]$, is provided in Boyer and Vorkink (2014):

$$\mathbb{E}_t\left[R^2_{t+1,n}\right] = \frac{S^2_{t,n}\exp\left[2{\sigma_{t,n}}^2 + 2\mu_{t,n}\right]\Phi\left(d_3\right) - 2KS_{t,n}\exp\left[\frac{{\sigma_{t,n}}^2}{2} + \mu_{t,n}\right]\Phi\left(d_1\right)}{C^2_{t,n}} + \frac{K^2\Phi\left(d_2\right)}{C^2_{t,n}}, \tag{17}$$

where d_1 and d_2 are defined as above, and $d_3 = d_1 + \sigma_{t,n}\sqrt{\tau}$. The expected R^2 for the true expected return is more stable and less sensitive to the random realization of returns. Further, it serves as an upper boundary to the expected R^2 of any other model. The proof can be found in Appendix B.

3.4 Machine Learning Methods

In this Section, we present the Machine Learning methods used in our analysis in increasing complexity. We aim to provide the intuition behind each of the models and outline the mathematical foundations. We distinguish between linear and non-linear models and present the framework for tuning model parameters. Our selection of prediction models is closely aligned with the models used in Gu et al. (2020) and Bali et al. (2023), but we exclude Neural Networks from our analysis due to computational limitations.

3.4.1 Linear Models

We consider three penalized linear models and two dimensionality-reduction models. The penalized linear models are based on a simple linear regression, but add a penalization term to the loss function $\mathcal{L}$:

$$\mathcal{L}(\theta; .) = \mathcal{L}(\theta) + \phi(\theta; .), \tag{18}$$

where θ denotes the model coefficient estimates, $\phi(\theta)$ is the arbitrary penalty term. The regularization is introduced to avoid over-fitting to the training data and aims to improve the predictions on out-of-sample data. Similar to many models presented here, it adds a degree of bias to the model and aims to reduce the variance in predictions. We now describe the methodology of the three penalized linear models: Ridge Regression, Lasso Regression, and Elastic Net Regression.

Ridge Regression

The first penalized linear model, Ridge, uses l_2 penalization in the loss function based on the sum of squared coefficients. The penalization term takes the following form:

$$\phi(\theta; .) = \lambda \frac{1}{2} \sum_{j=1}^{p} \theta_j^2, \tag{19}$$

where λ controls the strength of the penalization. As described earlier, the model adds a degree of bias, making predictions less sensitive to changes in the independent variables and effectively shrinking the coefficients. Increasing the strength of the penalty can shrink coefficient estimates to values close to zero, but not exactly zero. Hence, Ridge effectively addresses the issue of excessively large coefficients by shrinking them.

Lasso Regression

The second penalized linear model, Lasso, is similar to Ridge but uses l_1 penalization in the loss function. The penalization is based on the absolute value of the coefficients and takes the following form:

$$\phi(\theta; .) = \lambda \sum_{j=1}^{p} |\theta_j|. \tag{20}$$

Increasing the strength of penalty λ can shrink coefficient estimates to zero, effectively removing the predictor. For this reason, Lasso is often described as a variable selection method, dealing with uninformative predictors (Gu et al., 2020).

Elastic Net Regression

The third penalized linear model, Elastic Net Regression (ENet), combines the properties of Ridge and Lasso. It uses both l_1 and l_2 penalization in the loss function. This penalization takes the following form:

$$\phi(\theta; .) = \lambda(1-\rho)\sum_{j=1}^{p}|\theta_j| + \lambda\frac{1}{2}\rho\sum_{j=1}^{p}\theta_j^2, \tag{21}$$

The parameter λ again regulates the strength of the penalty and ρ regulates the weights allocated to each penalization type. Setting $\rho = 0$ simplifies the model to a Lasso regression. Setting $\rho = 1$ simplifies the model to a Ridge regression. Setting $\rho \in (0, 1)$ leads to a combination of Ridge and Lasso. Ridge regression tends to shrink all of the coefficients for correlated variables equally, whereas Lasso tends to select certain estimates to shrink to zero. By combining the two, ENet applies shrinkage through bias and shrinkage through selection (Gu et al., 2020).

The penalized linear models deal with the dimensionality of the data by regularizing the loss function and consequently shrinking the coefficients of the predictor set P. Often, predictors are correlated and penalization leads to sub-optimal coefficient estimates. In such a case, dimensionality-reduction methods can produce better estimates. We now describe two dimensionality-reduction methods, namely Principal Component Regression and Partial Least Squares.

Principal Component Regression

The first dimensionality-reduction method, Principal Component Regression (PCR), consists of two parts. The procedure is outlined in Gu et al. (2020). First, a Principal Component Analysis (PCA) is conducted, resulting in a set of linear combinations of the predictors that maximize the covariance between the predictors. Given a predictor dataset D, PCA solves for the eigenvalues and eigenvectors of the centered data $D' = D - \overline{D}$. This procedure results in k^* principal components and k^* eigenvectors Ω_{k^*}. We choose the number of components k and construct the dimensionality-reduced matrix $D'\Omega_k$. Second,

we perform a linear regression of the following form:

$$R = (D'\Omega_k)\theta_k + E, \tag{22}$$

where θ_k are the prediction coefficients and E is the error term.

Partial Least Squares

The second dimensionality-reduction method, Partial Least Squares (PLS), reduces dimensionality by maximizing the covariance between the predictors and the target, instead of maximizing the covariance among predictors. All predictors are aggregated into a component with weights based on their covariance with the target. This component effectively captures the information in the predictors that is most relevant for predicting the target. After forming the component, the variation explained by this component is removed from the predictors, and the next component is formed. This results in k components. Similar to PCR, the final forecast model is defined by Equation (22), where $D'\Omega_k$ is now the PLS dimensionality-reduced matrix. For both PLS and PCR, the parameter k, determining the number of principal components, is tuned as described in Section 3.4.4.

3.4.2 Non-Linear Models

To accommodate the non-linear patterns in our data, we use three non-linear models. All three models are types of regression tree models, which use binary classification to separate predictors into classification nodes. This iterative process results in a tree that creates predictions based on the terminal nodes of the tree. We use the methodology from Friedman et al. (2008) to explain the intuition behind a regression tree.

Consider the dataset D that consists of $T * N$ observations for p predictors. We denote an observation by the tuple $(x_{i1}, x_{i2}, ..., x_{ip}, y_i)$, $i = 1, 2, ..., T * N$, where x_{ip} denotes the value of predictor p for observation i, and y_i the value of the variable to be predicted, also referred to as the target variable. Further, we refer to the prediction produced by the tree model as the tree output. The first step in creating a regression tree is to establish an algorithm for binary classification.

Suppose there exists an arbitrary separation in the data that results in M distinct regions. For each region $R_m \in (R_1, R_2, ..., R_M)$, the output of that region is modeled as a constant by the regression tree. The final output $f(x)$ of the regression tree for observation x is then given by:

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{1}_{R_m}(x), \tag{23}$$

where c_m is the output of the model for observations that fall in the region R_m, and $\mathbb{1}_{R_m}(x)$ is the indicator function. Note that we have only described hypothetical outputs so far and that the region R_m with corresponding output c_m is yet to be determined. To find the best output $\hat{c}_m$ for the region R_m, a loss function is minimized. In this paper, we choose the sum of squares $\sum(y_i - f(x_i))^2$ as the loss function for tree models. It follows that the output $\hat{c}_m$ that minimizes the loss function is the average of the target values in the region R_m:

$$\hat{c}_m = \text{average}(y_i | x_i \in R_m). \tag{24}$$

Having derived the best outcome when presented with a set of regions, the set of regions now has to be determined. A greedy algorithm is deployed to find the best binary separation. The algorithm starts by separating the entire dataset D into two disjoint sets and progressively separates the sets further to form a tree. The separation of the data on predictor p at separation point s results in two regions referred to as half-planes:

$$R_1(p, s) = \{X | X_p \leq s\} \quad \text{and} \quad R_2(p, s) = \{X | X_p > s\}, \tag{25}$$

where X denotes the matrix containing the predictor values x_{ip}. The optimal binary separation is obtained by searching for predictor p and separation point s that solves the following minimization problem:

$$\min_{p,s} \left[\min_{c_1} \sum_{x_i \in R_1(p,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(p,s)} (y_i - c_2)^2 \right], \tag{26}$$

where the two best outcomes at the nodes are defined as:

$$\hat{c}_1 = \text{average}(y_i | x_i \in R_1(j, s)) \text{ and } \hat{c}_2 = \text{average}(y_i | x_i \in R_2(j, s)). \tag{27}$$

The resulting optimal pair (p, s) defining the binary separation serves as the root of the tree. For each of the two resulting regions, the minimization problem (26) is repeated. Following this procedure, a tree is successively built. In theory, the algorithm can be repeated until there is only one observation left in each region. The predictions of such a tree fit the in-sample data used to train the tree with perfect accuracy. However, this leads to an over-fitted regression tree with deteriorating out-of-sample performance. For this reason, constraints are set on the tree-growing process to prevent over-fitting on the training data. The three non-linear models we use in this paper all rely on regression trees and differ only in the tree-growing algorithm and the constraints applied to the growing process. The intuition behind the growing process remains unchanged. We will now describe the methodology behind the three non-linear methods, Random Forest, Gradient Boosting Regression, and Dart.

Random Forest

The first modified regression tree we consider is the Random Forest regression introduced by Breiman et al. (1984). A Random Forest grows multiple trees and aggregates the output by averaging them into a single decision classification. Each tree of the Random Forest is built on a bootstrapped sub-sample of the training set D. This technique is referred to as bootstrap aggregation (hereafter bagging) and has been shown to reduce variance in the prediction function (Friedman et al., 2008).

Consider the dataset D. The bootstrapped dataset D^* is created by randomly selecting observations from D with replacement. One bootstrapped dataset is used to grow one single tree. The Random Forest regression creates M bootstrapped datasets to build a total of M trees T_m where $m = (1, 2, ..., M)$. To build a tree, the following steps are repeated at each node until one of the constraints imposed on the growing process is reached:

(a) Take a random subset of predictors $\hat{P}$ out of the predictor set P.

(b) Given $\hat{P}$ and D^*, solve the minimization problem in (26) to find the best splitting pair (p, s).

(c) Split the node into two daughter nodes at (p, s).

Once a constraint is reached and the growing process is stopped, tree T_m is grown. A new bootstrap sample D^* is drawn for the next tree, and the process is repeated. Given the ensemble of trees $\{T_m\}^M$, the output of the Random Forest is computed by averaging over the output of every single tree:

$$f_{RF}^M(x) = \frac{1}{M} \sum_{m=1}^{M} T_m(x). \tag{28}$$

Bagging results in identically distributed trees. Therefore, the bias of the ensemble of trees is the same as the bias of each individual tree. Hence, the issue of bias cannot be addressed with bagging, whereas variance can be reduced (Friedman et al., 2008). As mentioned, there are constraints that stop the tree growing process, ensuring that the tree does not over-fit. For example, M is tuned to limit the number of trees, and the cardinality of the predictor set, $card(\hat{P})$, is tuned to limit the number of randomly sampled predictors in each iteration. Additional constraints to be tuned include the maximum depth of the tree and the maximum number of leaf nodes, and are discussed in further detail in Section 3.4.4.

Gradient Boosting Regression

The second modified tree-growing algorithm we discuss is Gradient Boosting Regression (GBR). The Gradient Boosting algorithm uses an ensemble of trees, similar to Random Forest Regression. After initializing the tree output, GBR iteratively updates the tree output by growing new trees. In contrast to Random Forest, GBR employs a boosting algorithm rather than bagging. In other words, the prediction model is improved with each iteration by adding a new tree that reduces the error of the previous prediction model. To prevent over-fitting, a learning rate λ is applied to the new tree before being added to the ensemble (Friedman et al., 2008). We apply the methodology as introduced by (Friedman, 2001).

Given a pre-defined loss function L, the first tree output is initialized by solving the following minimization problem:

$$f_o(x) = \arg\min_\gamma \sum_{i=1}^{T*N} L(y_i, \gamma), \tag{29}$$

where γ is a constant and denotes the model prediction. In our paper, we apply the squared error $\frac{1}{2}(y_i - f(x_i))^2$ as the loss function. Solving the minimization problem in (29) results in $\gamma = \bar{y}$, the average of the target values. The squared error is most commonly used as a loss function in regression adaptations of Gradient Boosting models (Friedman et al., 2008). Other loss functions include the absolute error and the Huber loss function, as implemented in Gu et al. (2020).

After the first tree output is initialized, further trees are subsequently grown. In every iteration, the negative gradient is calculated for each prediction and evaluated on the previous prediction model f_{m-1}:

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}. \tag{30}$$

Under the assumption of the squared error loss function, Equation (30) simplifies to $r_{im} = y_i - f_{m-1}(x_i)$, the residuals of the predictions. Next, the tree T_m is fitted to the residuals r_{im}, following the same growing process as outlined earlier. The resulting tree predicts the residuals r_{im} and has regions R_{jm}, where j is the number of terminal nodes. The residual predictions at each terminal node of the tree are computed by solving the following minimization problem:

$$\hat{\gamma}_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma). \tag{31}$$

Similarly, the problem is solved by the average of the residuals pertaining to region R_{jm}. The prediction model f_{m-1} is updated with the new tree multiplied by the learning rate $\lambda \in [0, 1]$:

$$f_m(x) = f_{m-1}(x) + \lambda \sum_{j=1}^{J_m} \hat{\gamma}_{jm} \mathbb{1}_{R_m}(x). \tag{32}$$

The process described above is repeated for the next tree until M trees are grown. The final tree prediction model under Gradient Boosting Regression is given by:

$$f_{GBR}^{M}(x) = f_M(x). \tag{33}$$

To prevent over-fitting, constraints are imposed on the tree-growing process, as described in Section 3.4.4

Dart

The last modified tree regression we apply is Dropouts meets multiple Additive Regression Trees (Dart), as introduced by Rashmi and Gilad-Bachrach (2015). Dart is a boosting algorithm similar to the Gradient Boosting algorithm. In contrast, a set of trees is randomly dropped during the tree-growing process, where each tree has a set probability of being dropped. The dropout is added to avoid over-fitting and is similar to the dropout technique employed in Neural Networks (Krizhevsky et al., 2012). During each boosting iteration, only the predictions of a random subset of trees are considered.

As in the Gradient Boosting algorithm, the tree-growing process is initialized by setting the first node such that it solves the minimization problem in (29). The first tree is built following the growing procedure of Gradient Boosting, providing the first ensemble of trees. In the subsequent iterations, a random sub-sample of trees $\hat{T}$ consisting of k trees is temporarily excluded from the ensemble. The set of trees in the temporary prediction model is given by:

$$\hat{F}_{m-1} = F_{m-1} \backslash \hat{T}, \tag{34}$$

where F_{m-1} is the set of trees included in $f_{m-1}(x)$. The resulting prediction model of $\hat{F}_{m-1}$ is $\hat{f}_{m-1}$. Note, if $\hat{T} = \emptyset$, a random tree is chosen to make $\hat{T}$ non-empty. Next, for each observation, the negative gradient is computed using the predictions from the reduced ensemble $\hat{f}_{m-1}(x)$:

$$r_{im} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=\hat{f}_{m-1}}. \tag{35}$$

The output nodes are calculated by solving the minimization problem in (31) for $\hat{f}_{m-1}(x)$, and the tree T_m is obtained. The number of trees included in the ensemble $\hat{f}_{m-1}(x)$ is strictly smaller than the number of trees in $f_{m-1}(x)$. Hence, the predictions of the new tree T_m have a magnitude approximately k-times larger than the predictions of the excluded trees $\hat{T}$. Therefore, the new tree T_m and the dropout trees $\hat{T}$ are normalized to account for the change in magnitude. The updated ensemble is then given by:

$$f_m(x) = f_{m-1}^{Norm} + \lambda \frac{1}{k+1} \sum_{j=1}^{J_m} \hat{\gamma}_{jm} \mathbb{1}_{R_m}(x), \tag{36}$$

where the new tree is scaled with $1/(k+1)$. The dropout trees in $\hat{T}$ are scaled by a factor of $k/(k+1)$ and added back to $\hat{F}_{m-1}$, resulting in f_{m-1}^{Norm}. This process is repeated until M trees are grown. The final output is:

$$f_{DART}^{M}(x) = f_M(x) \tag{37}$$

3.4.3 Ensemble Models

We create two equally weighted ensemble models, implemented similarly as in Bali et al. (2023). Equally weighted ensemble methods have shown good performance in economic forecasting as documented by Rapach et al. (2010). The ensemble predictions of the hold-until-maturity return are created as follows:

$$f_{Ensemble}(x) = \frac{1}{J} \sum_{j \in \hat{J}} f_j(x), \tag{38}$$

where $\hat{J}$ is the set of prediction models used in the ensemble, J is the number of models, and $f_j(x)$ are the predictions of model j. The first ensemble, L-En consists of the predictions of the five linear models. The second ensemble model, N-En consists of the predictions of the three non-linear models.

3.4.4 Hyperparameter Tuning

The performance of the models and algorithms described above can depend heavily on the parameter specification. It is important that we set the parameters such that they maximize the performance of the models. Hyperparameters have to be set manually before training a model. As in Gu et al. (2020) and Bali et al. (2023), we use a separate sample to evaluate the out-of-sample performance of the Machine Learning models under different hyperparameter combinations. We evaluate performance based on R^2_{OOS} as defined in Equation (12). For a given model, we set up a grid of possible values for each parameter. We then iterate over parameter combinations and train the models on the simulated data. For each iteration, we measure the model's performance against an out-of-sample dataset. We use the model specification that yields the highest performance to make out-of-sample predictions. Due to the high number of parameter combinations, we employ a random grid search for the non-linear models. In other words, we randomly select a

predetermined number of parameter combinations, and for each combination evaluate the model performance.

For Ridge, Lasso, and ENet we tune the penalization parameter. We tune PCR and PLS on the number of principal components the model retains. For the tree models, we set multiple hyperparameters. First, we set the maximum number of trees M that the models can grow. Second, we set the maximum depth of the trees. The maximum depth controls the number of splits a tree can perform, and effectively manages the vertical tree size. It is the largest control for impurity in the tree. Without specifying the maximum depth of the tree, it would continue to grow until every node perfectly predicts every observation or another constraint is reached. Third, we specify the maximum number of leaf nodes the tree can grow. It controls the number of terminal nodes and significantly constraints the size of each single tree. Further, for Random Forests we specify the cardinality of set $\hat{P}$, the number of predictors to consider when determining the best split. The parameter is stated as a fraction of the cardinality of set P^*. In addition, we specify the bagging fraction for Random Forests. It refers to the fraction of the cardinality of set P^* and the cardinality of set P.

In some Random Forest model adaptations, multiple bootstrap datasets are drawn and the bagging frequency is specified, as in Bali et al. (2023). Our model implementation does not allow for this as trees are built simultaneously and independently of each other. For GBR and Dart, we use a similar concept to bagging fraction. This makes the algorithm more complicated than in the methodology described above, but the intuition is straightforward. Only a random sample of the data is used in each boosting iteration. This limits the number of observations used. We optimize the sampling fraction over a range of conventional values. Moreover, we specify the learning rate, which can take values between 0 and 1. We use similar learning rate specifications as in Bali et al. (2023).

For Dart, we additionally optimize the number of columns used in each boosting iteration. Given the parameter, a random sample of columns is used for each tree. Moreover, we tune the dropout probability and the probability that the dropout procedure is skipped. Lastly, we tune an l_1 and l_2 regularization. Similarly to Ridge and Lasso, l_1 penalization encourages the weights of the trees in the ensemble to be sparser, whereas l_2 penalization encourages the weights to be smaller. The grid of all parameter combinations can be found in Appendix C.

4 Data

As we rely on simulated data, no actual data is needed to train the candidate models. However, our objective is to create a simulated environment that contains some realistic characteristics. This involves simulating values for the betas in our factor model, and requires appropriate assumptions about the expected return and variance of each factor. To achieve this, we fit the factor model we introduced in Section 3.1 to monthly historical factor return data spanning a twenty-year period from 2003 to 2023. We decided against sourcing historical data from a longer time horizon to reduce the risk of structural breaks in the data.

For each of the ten factors we assume in our factor model, we use a corresponding time-series of actual factor data. For the first five factors, we use monthly return data of the Fama-French Five-Factors (FF5), downloaded from Kenneth French's website. The five factors are constructed as outlined in Fama and French (2015). They include the market factor (Mkt-RF), size factor (SMB), value factor (HML), profitability factor (RMW), and investment factor (CMA). Next, we use a factor based on historical 36-month return volatility (VOL), as described in Blitz and van Vliet (2007). The time-series data is downloaded from the website of Robeco. In addition, we incorporate another value factor based on the lagged book-to-market ratio (VAL) and a momentum factor (MOM) based on the past twelve-month price action, both constructed according to Asness et al. (2013). The historical monthly return data is downloaded from the website of AQR Capital Management. To be in line with FF5, we use factor return data based on U.S. equities. We include monthly returns on the quality factor (QMJ) as outlined in Asness et al. (2019), as well as monthly returns of the betting-against-beta factor (BAB) as presented in Frazzini and Pedersen (2014). Both return time series are based on U.S. equities and downloaded from the website of AQR Capital Management. We acknowledge that the SMB factor and the VAL factor are conceptually very similar. In addition, the VOL factor is closely linked to the BAB factor as volatility and beta are interrelated. However, we do not consider this a problem. The aim of constructing the factor model in this paper is to build a model that produces returns that are aligned with those observed in reality. Adding factors with less relevance does not compromise this objective.

Historical monthly stock returns are retrieved from the CRSP database. We follow the filtering approach of Bali et al. (2023) and only consider stocks with share codes 10 and 11 and exchange codes 1, 2, 3, 30, 31, and 32 – firms that are traded on the NYSE, NYSE American, or NASDAQ. Furthermore, we only consider stocks with a share price greater than five. To produce robust OLS results, we filter out any stocks for which there are less than 100 return data entries available. This leaves us with a total of $2,850$ firms and $524,498$ return observations in the historical sample. Lastly, monthly data on the risk-free rate is retrieved from Kenneth French's website. Summary statistics of all historical data used can be found in Appendix D.

The empirical distribution of the betas for each of the ten factors can be found in Appendix E. For simplicity, we assume that each of them is normally distributed. Hence, we fit a normal probability density function. Under the Black-Scholes assumption that the log-returns are normally distributed, the factor returns $f_{k,t}$ in the factor model in Equation (1) must also be normally distributed. We use the sample mean and variance of the historical sample of factor returns as the expected return and the true variance of the normal distribution of factor returns. Furthermore, we fit a log-normal distribution to the empirical distribution of the idiosyncratic variance of returns and a t-distribution to the skewness of returns. Both empirical distributions including the fitted distributions can be found in Appendix E.

To simulate stock return data, we randomly sample the betas, realized factor returns, and idiosyncratic shocks from the fitted distributions. In Appendix F, we show summary statistics of all simulated data under the BS model. The distribution of simulated log-returns from a sample of size $T, N = 1000$ is shown in Figure 1. In addition, the empirical distribution of historical stock returns is shown. By construction, the simulated log-returns are normally distributed. It is interesting to see that the histogram of the simulated returns closely aligns with the fitted normal distribution of the *actual* returns. This is because our simulation procedure effectively equals fitting a normal distribution to the empirical distribution, and then sampling from the fitted distribution. The fact that the fitted normal distribution and the histogram of the simulated returns closely match serves as validation of our simulation procedure.

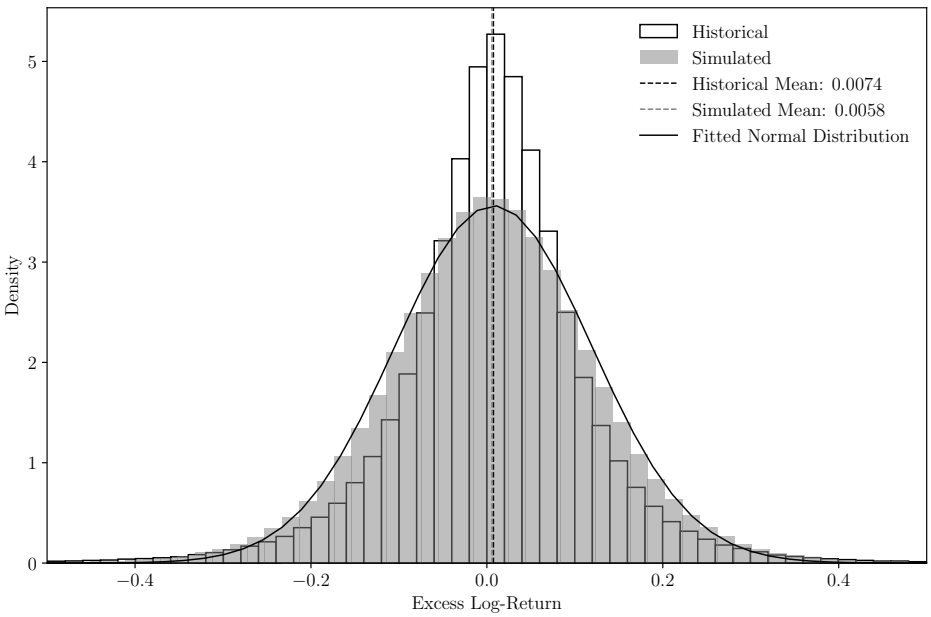


Figure 1: Empirical Return Distribution
This Figure shows the empirical distribution of historical excess returns and the sample distribution of excess returns generated by our simulation. The black line shows the probability density function of a normal distribution fitted to the empirical distribution of actual returns. The dotted lines show the means of the respective distributions.

5 Results

In Figure 2, the performance of the Machine Learning models under the base simulation study is displayed. The models are trained on three predictors, namely the two first moments of the return distribution of the underlying and the moneyness of the option. The simulation study specification consists of $T = 1000$ and $N = 1000$, a thousand time periods and firms. The first bar in Figure 2 labeled *BS* refers to the performance of a model using the true expected holding period returns under Black-Scholes as predictions. We refer to this model as the BS prediction model, or BS model, interchangeably.

The R^2 under the BS prediction model serves as an upper boundary to the realized R^2 of the candidate models given the simulated realized returns.[5] Furthermore, the expected R^2 under the BS prediction model is given by the black dotted line and represents the maximum possible expected R^2 achievable given the simulated data.

Firstly, we note that the BS model yields an R^2 of 0.211%. The performance of other models should be evaluated relative to BS. Most of the machine learning models have R^2

[5]It is possible that a model outperforms R^2 of the expected return under BS by pure randomness. Since we use a total of 10^6 observations, this is highly unlikely.

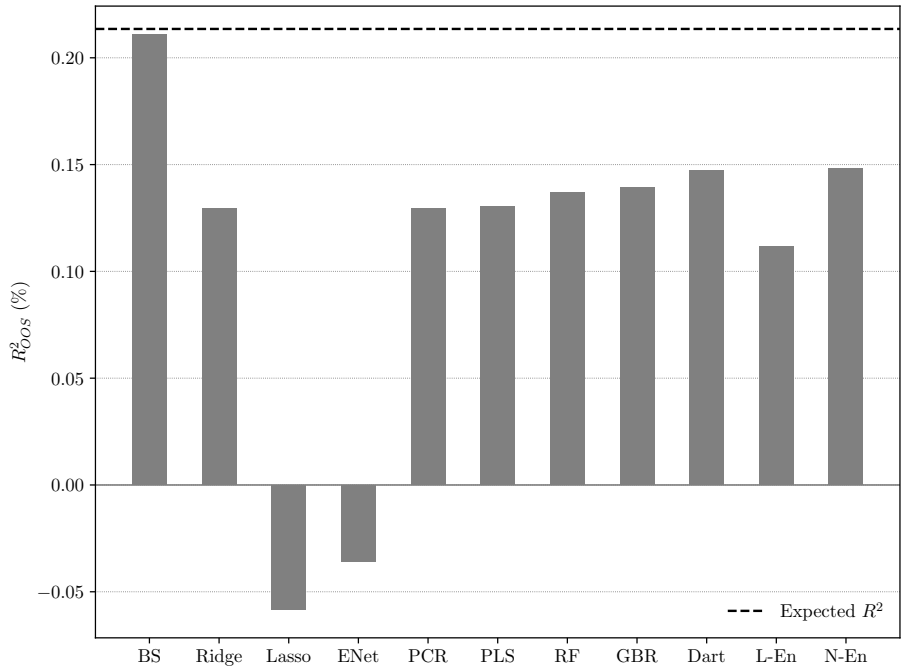


Figure 2: Base Model Performance
This Figure shows the R^2_{OOS} of ten Machine Learning models, namely, Ridge Regression (Ridge), Lasso Regression (Lasso), Elastic Net (ENet), Principal Component Regression (PCR), Partial Least Squares (PLS), Random Forest Regression (RF), Gradient Boosted Regression (GBR), Dropout meets multiple Additive Regression Trees (DART), the linear ensemble (L-En), and the non-linear ensemble (N-En). In addition, the R^2_{OOS} of the Black-Scholes prediction model (BS) is shown. The black dotted line represents the expected R^2_{OOS} of BS. The models are trained and tested on simulated data of size $T, N = 1000$.

scores in the range of 0.110% to 0.150%. We observe that, except for Lasso and ENet, the linear models perform strikingly well: Ridge, PCR, and PLS all yield R^2 scores of 0.129%. Most importantly, we find that non-linear models perform best in predicting option hold-until-maturity returns and that all three non-linear models outperform the linear models. The ensemble of the non-linear models yields an R^2 of 0.152% while the linear ensemble achieves an R^2 of 0.112%. Of the non-linear models, the performance is highest for the Dart model and lowest for the Random Forest model. However, the difference is small.

The R^2 scores we obtain are lower than in comparable literature. Bali et al. (2023) test the predictability of daily delta-hedged option returns and show R^2 scores in the region of 2% for the non-linear models. As delta-hedged option returns are said to be less skewed and less leptokurtic than raw holding period returns, predicting holding period returns could be harder than predicting delta-hedged returns (Zhan et al., 2022). In addition,

since we have modeled Black-Scholes-type stock dynamics with log-normally distributed stock returns, performance scores obtained in our study are not fully comparable. Thus, the level of R^2 is not as indicative as the relative performance between models. Gu et al. (2020) find R^2 scores in the region of 0.3% when predicting stock returns, slightly higher than the performance scores in our simulation. Apart from differences due to our simulation assumptions, this aligns with our expectation as the truncated nature of option payoffs adds complexity.

Furthermore, the strong performance of linear models leads us to believe that a large portion of the predictable share of returns under BS could be of linear nature. The results indicate that this linearity is not captured by the Lasso regression, which produces a negative R^2. We believe that the performance of Lasso can be attributed to its method of predictor selection, which is sub-optimal when all predictors contain relevant information. The negative performance score of ENet originates from the equal weight it puts on Lasso and Ridge. Lastly, the non-linear models perform better, indicating that they are finding non-linear patterns that cannot be explained by the linear models. It is not surprising that the most complex tree-algorithm, Dart, outperforms the other non-linear models.

In Figure 3 the performance of the models under three sorts is shown. For the purpose of readability, we restrict our analysis to the two ensemble models. We sort for moneyness, expected stock return, and stock return variance. The upper left graph shows the dynamics of R^2 for ten buckets sorted on moneyness. Recall that we defined moneyness as $\frac{K_{t,n}}{S}$. Therefore, high values of moneyness correspond to out-the-money options and low values to in-the-money options. The top-right graph shows the difference between the R^2 of the ensemble models and that of the Black-Scholes prediction model.

We observe that R^2 of the BS prediction model is declining in moneyness. The return distribution of the option converges to the return distribution of the underlying, namely a log-normal distribution, as moneyness converges to zero. On the other hand, the further an option is out-of-the-money, the more negatively skewed the return distribution becomes. A far out-of-the-money option will yield a return of zero almost with certainty. In the rare case that the option does expire in the money, the option yields an extreme return. This is due to the initially low price of the option. We argue that the expected return of such a skewed distribution is a worse predictor of realized returns. Interestingly, the performance for the two ensemble models does not decrease in moneyness. In fact, the

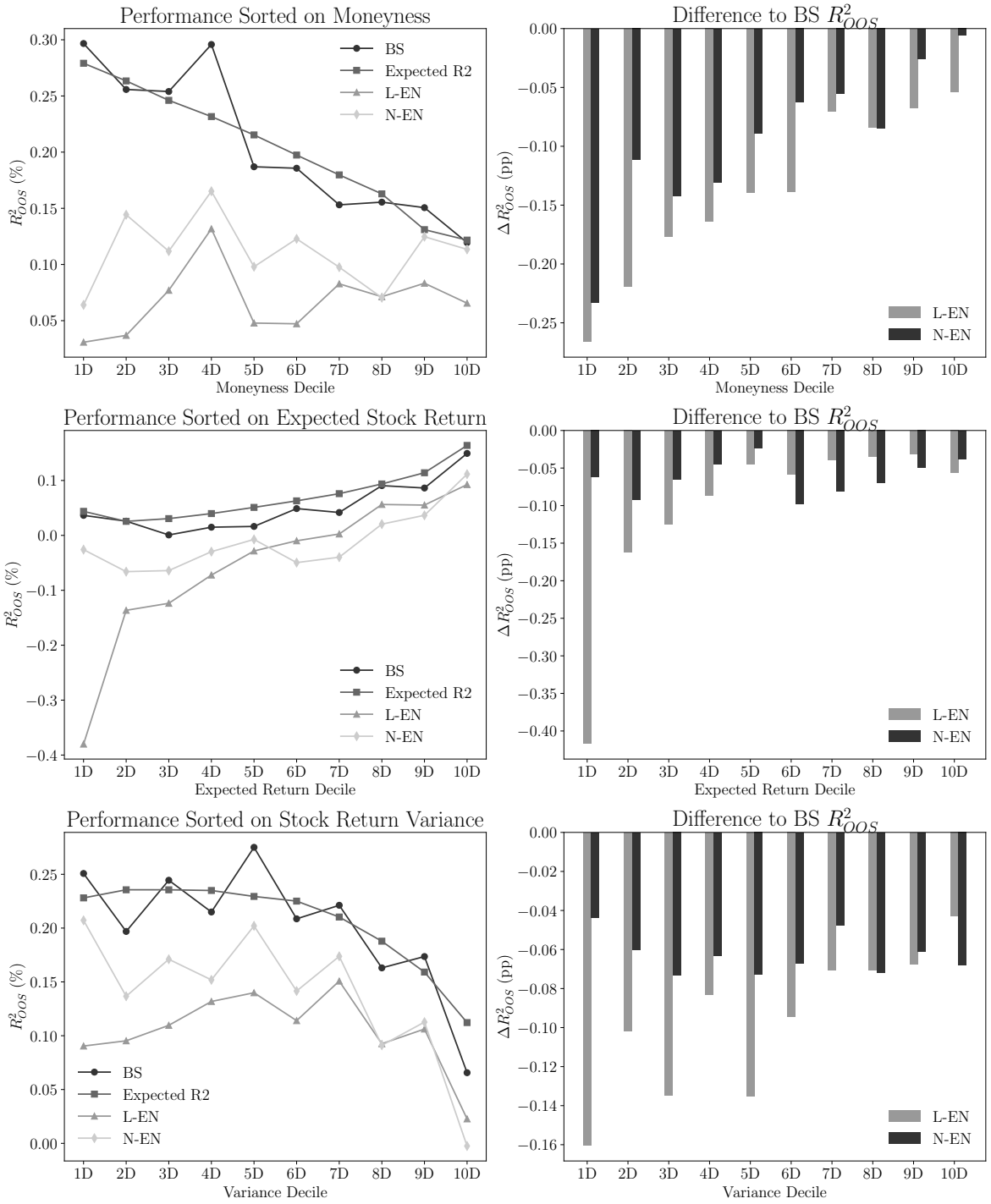


Figure 3: Base Model Performance Sorts
This Figure shows the R^2_{OOS} in percent of the Black-Scholes prediction model (BS), the linear ensemble (L-En), the non-linear ensemble (N-En), and the expected R^2_{OOS} of BS, sorted on three predictors. The top left Figure shows the R^2_{OOS} for ten buckets of moneyness. The middle left Figure shows the R^2_{OOS} for ten deciles of the expected stock return. The bottom left Figure shows the R^2_{OOS} for ten deciles of stock return variance. For each of the three Figures on the left, the corresponding Figure on the right shows the difference between R^2_{OOS} of the ensembles and the BS prediction model.

relative performance to Black-Scholes increases in moneyness for both models. Still, the non-linear ensemble outperforms the linear ensemble across the moneyness range.

If the ensemble models fully captured the downward trend between R^2 and moneyness, the relative performance of the models would stay constant. However, the models struggle to predict the returns of in-the-money options in particular. We argue that both models fail to fully capture the dynamics between moneyness and expected hold-until-maturity return.

In the two middle graphs in Figure 3 the results are shown for sorts across ten deciles of expected stock return. We observe that R^2 is increasing in the expected return of the underlying, as indicated by the expected R^2 of the Black-Scholes prediction model. Further, we see that the relative performance of the linear ensemble increases in the expected stock return, as indicated by the middle right graph. On the other hand, the relative performance of the non-linear models remains constant. As argued before, the relative performance of the models stays constant if the upward trend is captured. It appears that the linear ensemble fails to capture the dynamics between the expected stock return and the expected option return, especially for low expected returns. On the contrary, the non-linear ensemble succeeds in capturing this dynamic.

In the two bottom graphs in Figure 3 the results are shown for sorts across ten deciles of stock variance. We observe that the predictability of option holding period returns decreases in stock variance. We argue that this is because increased return variance of the underlying corresponds to increased variance of holding period option returns. Increased variance of the holding period returns intuitively leads to increased prediction difficulty. Similar to the sorts on moneyness, the relative performance of the linear model increases with variance. In addition, we observe that the relative performance of the non-linear models stays constant. Again, this indicates that the linear ensemble fails to capture the dynamics between the stock variance and the expected hold-until-maturity period return of the option. Finally, the non-linear ensemble outperforms the linear ensemble for almost all variance deciles.

5.1 Alterations

We now alter the simulation specifications to get a better understanding of the candidate models' performance under different conditions. Table 2 shows the R^2 scores for five simulation alterations. The top row displays the base model performance scores obtained earlier, and serves as a benchmark for comparison. Panel A in Table 2 shows the performance under different predictor sets than in the base model. This allows us to observe the strength of the individual predictors. Note that under the different feature specifications, the candidate models have been trained on the same simulated sample as in the base model. Any effects we observe can thus be attributed to the feature specification, and not to randomness. Conveniently, using the same random sample leaves the R^2 of the Black-Scholes prediction model unchanged, as it does not depend on the predictors used by the Machine Learning methods.

First, we exclude the expected stock return $\mathbb{E}_t[r_{t+1,n}]$ from the feature set and restrict the training data to moneyness and total variance. We observe that the models cannot make meaningful predictions anymore as performance has effectively dropped to zero. We conclude that the expected stock return contains significant information about the hold-until-maturity return of options.

Second, we exclude the stock variance $\text{Var}_t[r_{t+1,n}]$ from the feature set. We observe that the performance remains almost identical for the linear models. On the contrary, the performance for the non-linear models decreases slightly. Since non-linear models thrive in larger training datasets, we argue that the reduction in performance is due to the mere reduction of the number of predictors. Despite the performance decline, non-linear models continue to outperform linear models, both individually and in the ensemble. Interestingly, the performance of Dart drops below that of Random Forest and is no longer the best performing model. This indicates that the tree-algorithm of Dart finds meaningful patterns in the variance. Thus, excluding it reduces its predictive power.

Third, we remove the moneyness $K_{t,n}/S$ from the feature set. We observe that the performance decreases for all models and that scores converge to each other. Despite this, Random Forest continues to be the best predictor model, and all non-linear models outperform the linear models. Further, the non-linear ensemble performs better than the linear ensemble.

Lastly, we alter the feature specification by replacing the expected stock return with the ten factor betas, while retaining the moneyness and stock variance in the predictor set. Again, we observe a decline in performance for all models compared to the base specification, though more pronounced for the non-linear models. Further, we see that non-linear models do not outperform linear models under this feature specification. We note that Lasso and ENet perform similarly to other linear models. Under the base specification, the models clearly underperform and yield negative R^2 scores. The strong performance of the linear models under this alteration is likely due to the linear nature of the factor model assumed. To generalize this finding, the performance of the models under non-linear return models has to be analyzed. We refrain from doing so in this paper and leave this to future research.

Next, we alter the ratio between the number of firms and periods in the simulated data set. The results for different ratios are shown in Panel B in Table 2. A graph of the results can be found in Appendix G. The performance is shown relative to the performance of the BS prediction model for each alteration. In the base model, we set $T = 1000$ and $N = 1000$, or $T = 50\%$. We explore eight different T specifications, where we keep the amount of total observations $T * N$ constant at 10^6. As the shape of the simulated data changes when the T/N ratio is changed, it is not possible to recycle the sampled data from the base specification. Therefore, the results are based on different random samples and results likely include some degree of randomness. For this reason, we aim to look at the overarching trend rather than individual R^2 scores. We observe that the relative performance to the Black-Scholes prediction decreases on average for lower percentages of T across all models. Hence, time-series data on option returns contains inherently more information than cross-sectional data. We argue that this is because realized factor returns are drawn only once per period, and because the cross-section of stock returns is correlated. By that, datasets with greater time dimensions are more varied. We conclude that longer time series data are preferred over cross-sectional breadth to improve the performance of the Machine Learning models.

In Panel C in Table 2 we show the performance distribution under a reduced sample size. We aim to account for the fact that time-series data on real option returns are limited. The time dimension of the training dataset is reduced to $T = 100$, while the firm dimension is kept at $N = 1000$. The performance is measured against simulated

Specification	**BS**	**Expected BS**	**Ridge**	**Lasso**	**ENet**	**PCR**	**PLS**	**RF**	**GBR**	**Dart**	**L_En**	**N_En**
Base	0.211	0.214	0.129	-0.058	-0.036	0.129	0.129	0.141	0.144	0.147	0.112	0.152
Panel A: Feature Selection Specification												
-$\mathbb{E}[r_{t+1,n}]$	0.211	0.214	-0.022	-0.046	-0.03	-0.022	-0.022	-0.019	-0.019	-0.023	-0.019	-0.017
-$\text{Var}_t[r_{t+1,n}]$	0.211	0.214	0.129	-0.046	-0.03	0.129	0.129	0.138	0.134	0.137	0.112	0.139
-$K_{t,n}/S$	0.211	0.214	0.099	-0.058	-0.058	0.099	0.099	0.110	0.103	0.104	0.077	0.108
+$\beta_{k,n}$; -$\mathbb{E}[r_{t+1,n}]$	0.211	0.214	0.111	0.099	0.113	0.111	0.122	0.092	0.046	0.092	0.120	0.092
Panel B: T/N Ratio Specification in % relative to BS												
T: 90%	100.0	99.8	88.4	-0.2	-0.2	88.4	88.4	72.5	77.6	80.3	80.8	82.8
T: 80%	100.0	111.1	75.5	-10.2	-0.5	75.4	75.4	71.6	73.7	71.6	68.7	78.0
T: 70%	100.0	107.9	88.1	-2.4	8.7	88.1	88.2	74.8	66.4	96.8	92.0	92.4
T: 60%	100.0	107.1	75.3	-4.3	3.0	74.9	75.9	75.0	27.5	69.7	76.8	69.2
T: 40%	100.0	78.8	75.0	-13.1	-6.7	75.0	75.0	52.7	51.3	63.4	67.6	62.9
T: 30%	100.0	100.1	57.6	-29.7	-21.2	57.4	58.6	65.0	58.7	61.0	67.9	67.9
T: 20%	100.0	107.0	30.1	-63.0	-52.9	30.1	30.2	13.2	14.3	28.0	23.3	29.8
T: 10%	100.0	122.1	66.2	-47.4	-39.3	66.2	66.3	47.5	68.3	66.6	72.0	75.7
Panel C: Reduced Sample Size												
25%-Q.	0.172	0.218	-0.021	-0.141	-0.134	-0.025	-0.025	-0.013	-0.065	-0.042	-0.034	-0.032
50%-Q.	0.214	0.224	0.118	-0.038	-0.037	0.112	0.112	0.100	0.033	0.054	0.071	0.069
75%-Q.	0.243	0.229	0.166	-0.005	-0.001	0.164	0.166	0.140	0.103	0.114	0.118	0.129
Panel D: Extra Contracts												
	0.211	0.214	0.169	-0.017	0.008	0.169	0.169	0.182	0.166	0.183	0.147	0.182
Panel E: Put Options												
	0.244	0.249	0.166	-0.051	-0.051	0.166	0.168	0.165	0.150	0.180	0.135	0.171

Table 2: Model Performance under Altered Specifications
This Table shows the R^2_{OOS} in percent for five alterations to the model specifications. The top row shows the R^2_{OOS} of the base model specification. Panel A shows the performance of alterations to the feature set. A minus indicates that a feature is dropped, and a plus that a feature is added. Panel B shows the performance for eight altered time and firm dimensions. Panel C shows the performance quartiles of 100 iterations of size $T = 120, N = 1000$. Panel D shows the performance under one extra option contract per firm and period. Panel E shows the performance of predicting put options.

out-of-sample data of size $T, N = 1000$. This simulation specification has been iterated one hundred times, and every model is parameter-tuned at each iteration. We see that the R^2 of the BS model ranges between 0.172% and 0.243 % from the upper to the lower quartile. As expected, the median of 0.214% is close to the performance under the base specification. We make two observations. First, we find that the performance of the linear and non-linear models varies significantly across the iterations under reduced sample size. Second, we observe that the median performance of the non-linear models decreases significantly, while that of the linear models decreases only slightly. Non-linear models are outperformed by most linear models, and even the highest quartile of all non-linear models does not reach the benchmark performance under the base specification. We conclude that non-linear models suffer more from reducing the available data than linear models. This is because tree-based models need enough data to reveal non-linear structures and excel when provided with large amounts of data. Further, this demonstrates that the performance of non-linear models is greatly dependent on data availability.

In Panel D in Table 2 we show the outcomes of adding additional option contracts with different strike prices. Specifically, we use the same simulated dataset of size $T, N = 1000$ as in the base specification and introduce one other option contract per firm and period, differing only in moneyness. This procedure effectively doubles the size of our training set. We observe that R^2 increases for all models. Except for GBR, non-linear models continue to outperform all other linear models. Moreover, the non-linear ensemble yields an R^2 of 0.182%, coming close to the maximum attainable R^2 of 0.211%. The results suggest that, by including several option contracts per underlying stock and period, the models gain more insights into the dynamics between moneyness and option returns. We argue that, with multiple options differing only in the strike price, the effect of moneyness on option holding period returns is presented ceteris paribus to the candidate models.

Lastly, in Panel E the results of the base specification with put-options are shown. We note that the realized and expected R^2 of the BS prediction model are higher than for call options. We suggest that this is because put option holding period returns are bounded as the price of the underlying stock cannot become negative, whereas call option returns are unbounded. This leads to more extreme returns for call options than for put options. For example, the 99th percentile of realized call option returns in our simulated test set is 806%, whereas it is 564% for put options. Such extreme values are penalized

more heavily by the R^2 measure. Along with the R^2 of the BS prediction model, the performance of all candidate models increases. This is similar to Bali et al. (2020), who find that delta-hedged put option returns are easier to predict than call option returns. We observe that the non-linear ensemble continues to outperform the linear ensemble.

5.2 Trading Strategy

In this section, we implement a trading strategy using the predictions from the candidate models to assess their economic value. We measure the performance in terms of the Sharpe Ratio, a measure of risk-adjusted performance. The Sharpe Ratio is defined by:

$$SR = \frac{E[r_p] - r_f}{\sigma_p}, \tag{39}$$

where $E[r_p] - r_f$ is the expected excess return of the trading strategy and σ_p is the volatility of returns. We construct the Sharpe Ratio based on annual returns in this paper.

Turning away from R^2 and towards the Sharpe Ratio for performance measurement is particularly interesting as a high R^2 does not necessarily translate to high Sharpe Ratios (Kelly et al., 2024). As before, the training data for the Machine Learning models consists of a thousand time periods and firms. We continue to use the expected stock return, the stock variance, and the moneyness as predictors. The performance of the trading strategies is measured against an out-of-sample data set of $T = 120$ and $N = 1000$, representing a hypothetical investment horizon of ten years. To construct the portfolio, we solely use the predictions produced by the candidate models as the investment criterion. Each month, we sort option contracts into deciles based on their predicted holding period return, buying the options in the highest decile and writing the options in the lowest decile. The options are equally weighted, and the resulting portfolio is self-financing. After simulating the realized returns for each option using the factor model, we calculate the realized return of the portfolio for each period. This procedure generates a series of monthly portfolio returns for each candidate model. We then compute the Sharpe Ratio by using the sample mean and standard deviation of the realized returns.

To better understand the magnitudes of Sharpe Ratios obtained from various trading strategies, we conducted 100 iterations for each candidate model. In each iteration, we simulate a new out-of-sample data set. Figure 4 shows the annualized Sharpe Ratios

obtained from the iterations, as well as the mean annualized return for each candidate model. The median Sharpe Ratio of a trading strategy based on the Black-Scholes prediction is 1.1. The Sharpe Ratios of the Ridge, PCR, PLS, and the three non-linear models exhibit a very similar magnitude. The same holds for the two ensemble models. In contrast, the Sharpe Ratio for the Lasso regression fluctuates around zero, suggesting it is less suitable for building a trading strategy. We believe that the tendency of the Lasso model to generate constant predictions results in the random selection of options when constructing the portfolio.

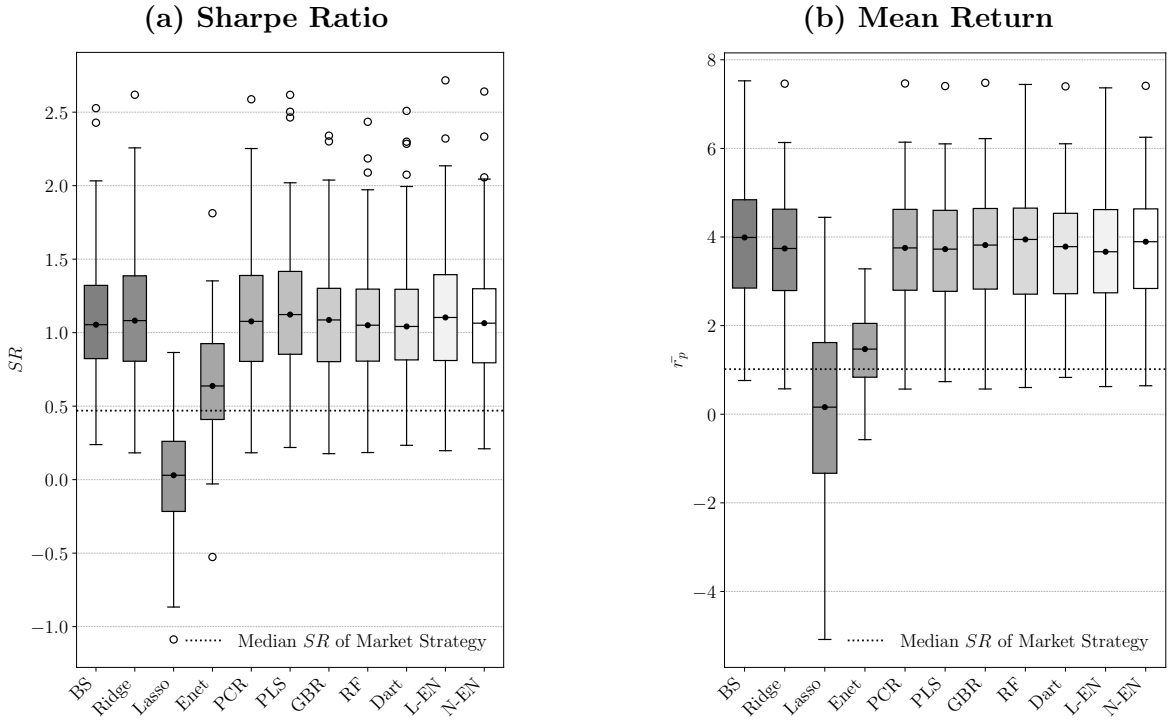


Figure 4: Trading Strategy Performance
This Figure shows the performance of the trading strategy across the candidate models in terms of (a) annual Sharpe Ratio and (b) annual mean annual return in decimal form over 100 iterations. Options are sorted into deciles based on the predicted return. A long-short portfolio is constructed by buying the top decile and writing the bottom decile. The simulated training is of size $T, N = 1000$, and the testing data is of size $T = 120, N = 1000$. The median of the annual Sharpe Ratio and mean return of a simple market strategy is also provided. Under this strategy, all available options are bought to form equally-weighted portfolios.

We observe that the relative performance in terms of R^2 scores obtained earlier is not preserved when measuring performance in Sharpe Ratios. While non-linear models previously dominated the linear models in terms of R^2, they now perform slightly worse. Further, the median Sharpe Ratio of the three non-linear models is lower than that of

PLS, and the linear ensemble slightly outperforms the non-linear ensemble. This confirms our earlier statement that high R^2 does not necessarily translate to high Sharpe Ratios and vice versa. In contrast, we find that the relative performance measured in terms of mean return is more similar to the relative performance in terms of R^2.

We are surprised by how little variation the mean return shows among the candidate models other than Lasso and ENet. Similarly, we observe little variation in the Sharpe Ratios between the candidate models. We argue that, in contrast to R^2, achieving a high mean return requires correctly classifying the option returns into their deciles rather than making accurate return predictions. Take for example a hypothetical model which predicts returns to be precisely twice as high as another model's return predictions. Both models will yield the same mean return and Sharpe Ratio, but not the same R^2. It appears that except for Lasso and ENet, all candidate models perform similarly well in classifying the option returns into deciles.

Next, we analyze the Sharpe Ratio of the non-linear ensemble more in-depth and look at three sorts on moneyness, stock variance, and expected stock return. The results are shown in Figure 5. Interestingly, the Sharpe Ratio dynamics closely mirror the dynamics of R^2 we observed in Section 5.1 for the sorts on moneyness and expected return. First, we observe that the Sharpe Ratio is smaller for out-the-money options than for in-the-money options, as presented in Figure 5a. Second, in Figure 5c we see that the Sharpe Ratio is increasing in the expected stock return for all deciles except for the first two. We observed a similar dynamic in Figure 3, seeing an uptick in the R^2 in the lowest decile. However, this uptick is more pronounced for the Sharpe Ratio. In contrast to moneyness and expected stock return, the Sharpe Ratio does not seem significantly influenced by the stock variance, as shown in Figure 5b. Except for the last decile, all Sharpe Ratios appear to fluctuate around 0.85.

Lastly, we aim to benchmark the Sharpe Ratio obtained from the non-linear ensemble. The dotted line in Figure 4 shows the Sharpe Ratio and mean return of a simple market strategy, in which the portfolio consists of all available stocks. We see that all models except Lasso outperform the market strategy, both in terms of Sharpe Ratio and mean return. Further, we implement a benchmark approach similar to Bali et al. (2023) and create simple alternative trading strategies built on univariate sorts on one of the available predictors. We consider all betas, mean risk premiums, risk premium variances, as well as

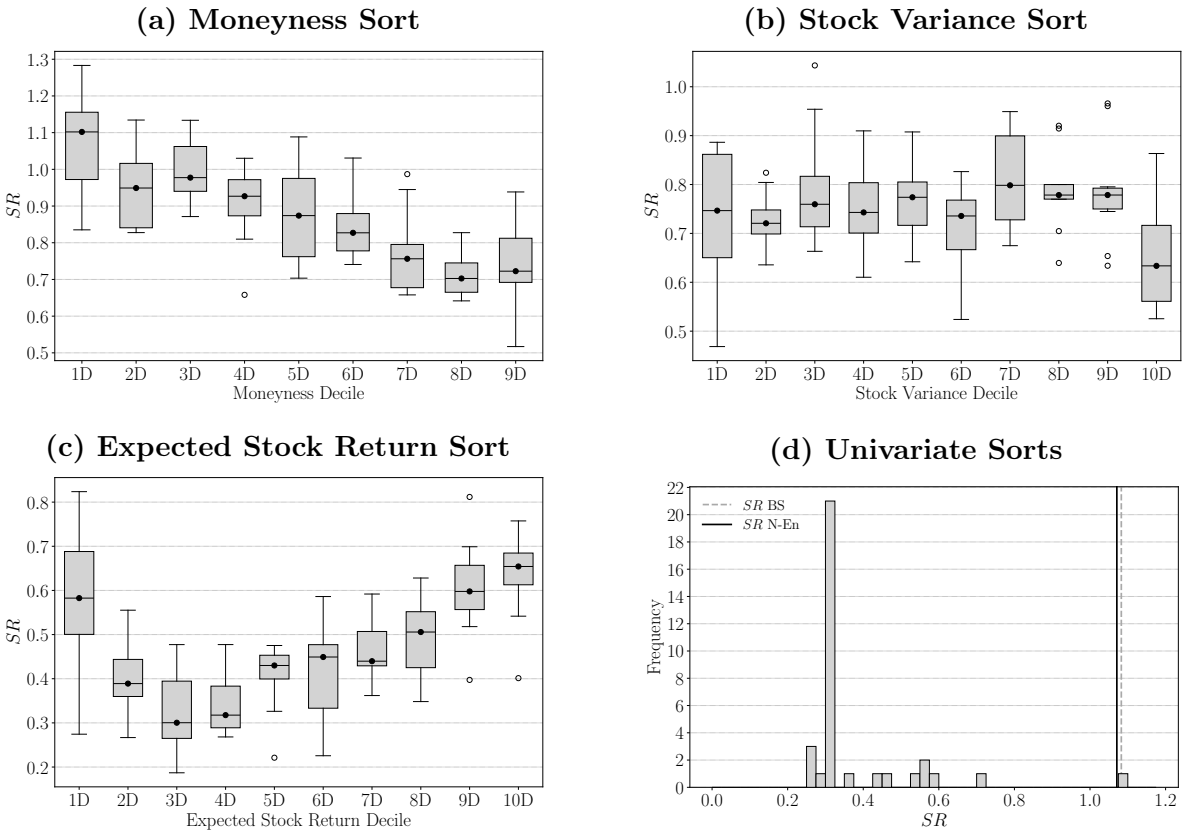


Figure 5: Trading Strategy Analytics of the Non-Linear Ensemble
Figure (a) shows the annualized Sharpe Ratio of the trading strategy based on the predictions of the non-linear ensemble, sorted on moneyness. Figure (b) shows the annualized Sharpe Ratio sorted on stock variance. Figure (c) shows the annualized Sharpe Ratio sorted on the expected stock return. For each bucket, the simulated training data is of size $T, N = 1000$, and the testing data is of size $T = 120, N = 1000$. Results are shown for 10 iterations. Figure (d) shows the mean Sharpe Ratios of a strategy based on univariate sorts on each of the available predictors over 100 iterations.

the moneyness, idiosyncratic variance, expected stock return, and stock return variance. As before, we run the trading strategy on out-of-sample data of size $T = 120, N = 1000$ and run 100 iterations. For each iteration, we compute a total of 34 Sharpe Ratios, one for each sorting criterion. To obtain the mean Sharpe Ratio for each sorting criterion, we average over all iterations. In addition, we obtain the mean Sharpe Ratio of the BS prediction and of the non-linear ensemble. The results, as shown in Figure 5d, reveal that the Sharpe Ratio of the non-linear model significantly exceeds that of nearly all other strategies. The one exception is the strategy using the expected stock return as a sorting criterion, which produces a similar Sharpe Ratio to the non-linear ensemble. This again highlights the exceptional informational value of the expected stock return.

6 Merton Jump Diffusion Model

So far, we have only studied option returns under the Black-Scholes model and its assumptions, one of which is that stock returns are log-normally distributed. However, it is considered a stylized fact that stock returns do not follow a log-normal distribution. Instead, stock returns are observed to be leptokurtic. This has first been brought to attention by Mandelbrot (1963) for asset prices in general, and subsequently confirmed by Fama (1965) for stock prices. Furthermore, stock returns are skewed (Albuquerque, 2012).

Since Black and Scholes (1973) presented their model for option pricing, a large body of research has emerged aiming to extend the BS model to better account for stylized facts in stock returns (Heston, 1993; Bates, 1996; Carr and Sun, 2007). The purpose of this Section is to implement one of these extensions, namely the Merton Jump Diffusion (MJD) model as presented in Merton (1976). We choose MJD because it provides a closed-form solution for the option price and the first three moments of the underlying stock return are defined. This allows us to apply the factor model used in this paper and directly compare the performance of the candidate models under MJD with the performance under BS.

6.1 Methodology

Merton (1976) aims to capture the skewness and kurtosis in stock returns by allowing for discrete jumps. Under MJD, the stock price process consists of two components. First, a Geometric Brownian Motion similar to BS to account for the continuous changes in price. The continuous price process is overlaid with discontinuous jumps. These jumps represent price changes of the stock after the release of material information about the underlying company. Such information includes unexpected announcements of a take-over, an earnings miss, or the loss of a court suit. Merton (1976) assumes that the jump sizes are independently and identically distributed, and that the number of jumps is Poisson distributed. The discontinuous stock price process is then driven by the following stochastic differential equation (SDE):

$$\delta S_t = S_t(\mu - \lambda k)\delta t + S_t \sigma \delta W_{t,n} + \delta q_t. \tag{40}$$

The parameter μ denotes the instantaneous return of the continuous price process; σ is the instantaneous standard deviation of the stock return conditional on no jumps occurring; λ is the mean number of jumps per unit of time; k is the expected percentage change in the stock due to the jump. We define $Y - 1$ as the percentage change in the stock due to the jump. Hence, $k = \mathbb{E}[Y - 1]$. Furthermore, q_t is a Poisson process described by the following probabilities in the limit $\delta t \to 0$:

1. Prob(no jump occurs in the time interval $(t, t + \delta t)) = 1 - \lambda \delta t$,
2. Prob(one jump occurs in the time interval $(t, t + \delta t)) = \lambda \delta t$,
3. Prob(more than one jump occurs in the time interval $(t, t + \delta t)) = 0$.

The process takes the value $q_t = (Y - 1)S_t$ if the jump occurs at time t, and zero otherwise. The Poisson process and the Wiener process in Equation (40) are assumed to be independent. Solving the SDE in Equation (40) yields the following stock price process for firm n:

$$S_{t+\tau,n} = S_{t,n} e^{\left(\mu_{t,n} - \frac{1}{2}\sigma_{t,n}^2\right)\tau + \sigma_{t,n} W_{t+\tau,n}} Y_{t+\tau,n}(s_{t+\tau,n}), \tag{41}$$

where $s_{t+\tau,n}$ is the number of price jumps that occurred to the stock of firm n during time period τ, and $Y_{t+\tau,n}(s_{t+\tau,n}) = \prod_{j=1}^{s_{t+\tau,n}} Y_{j,n}$, if $s_{t+\tau,n} \geq 1$ and $Y_{t+\tau,n}(s_{t+\tau,n}) = 1$ otherwise. Here, $Y_{j,n}$ are independently and identically distributed jumps as defined above. The number of jumps $s_{t+\tau,n}$ is Poisson distributed with intensity $\lambda_{t,n}$. Using $W_{t+\tau,n} = \sqrt{\tau} z_{t+\tau,n}$, we can simplify Equation (41) to

$$S_{t+\tau,n} = S_{t,n} e^{\left(\mu_{t,n} - \frac{1}{2}\sigma_{t,n}^2\right)\tau + \sigma_{t,n}\sqrt{t} z_{t+\tau,n}} Y_{t+\tau,n}(s_{t+\tau,n}), \tag{42}$$

where z_n is a standard normally distributed random variable.

Up to this point, we haven not made any assumption about the distribution of the jump sizes $Y_{j,n}$, except that they are independently and identically distributed. Under the assumption that the jump sizes are log-normally distributed with $\log(Y_{j,n}) \sim N(\alpha_{t,n}, \delta_{t,n}^2)$, Merton (1976) provides a closed-form solution for the price of a call option with underlying stock n and time to maturity τ:

$$C_{t,n}^{MJD}\left(\tau, S_{t,n}, \sigma_{t,n}, \delta_{t,n}, \lambda_{t,n}, K_{t,n}, r^f, \alpha_{t,n}\right) = \sum_{s=0}^{\infty} \frac{e^{-\lambda'\tau}\left(\lambda'\tau\right)^s}{s!} C_{t,n}^{BS}\left(\tau, S_{t,n}, \sigma_s, K_{t,n}, r_s\right), \tag{43}$$

where:

$$\lambda' = \lambda_{t,n} e^{\alpha_{t,n}+\frac{1}{2}\delta_{t,n}^2}, \tag{44}$$

$$\sigma_s^2 = \sigma_{t,n}^2 + \frac{s\delta_{t,n}^2}{\tau}, \tag{45}$$

$$r_s = r^f - \lambda_{t,n}\left(e^{\alpha_{t,n}+\frac{1}{2}\delta_{t,n}^2} - 1\right) + \frac{s\left(\alpha_{t,n}+\frac{1}{2}\delta_{t,n}^2\right)}{\tau}, \tag{46}$$

and where $C_{t,n}^{BS}$ denotes the Black-Scholes call option price as given by Equation (10). The price of the call option under MJD is the infinite sum of the call option prices under BS, conditional on that there are s jumps. The call prices are weighted with the Poisson probability that s jumps occur until maturity.[6] The option price formula under MJD takes three additional input parameters characterizing the the jump process, namely $\lambda_{t,n}$, $\delta_{t,n}$, and $\alpha_{t,n}$.

It is important to note that jump-risk cannot be hedged. Therefore, there exist infinitely many risk-neutral measures, under each of which the price of the option is arbitrage-free. The concrete choice of the risk-neutral measure depends on the individual price of jump-risk. Merton (1976) assumes that jump-risk is fully non-systematic. Therefore, the price of jump-risk is zero. Under this assumption, the P-measure is transformed to the Q-measure by setting $\mu_{t,n} = r^f$, similar to BS. The expected holding period return of a call option under MJD is then given by:

$$\begin{aligned}\mathbb{E}_t\left[R_{t+\tau,n}^{\text{call}}\right] &= \frac{\mathbb{E}_t^P\left[\left(S_{t+\tau,n} - K_{t,n}\right)^+\right]}{C_{t,n}^{MJD}\left(\tau, S_{t,n}, \sigma_{t,n}, \delta_{t,n}, \lambda_{t,n}, K_{t,n}, r^f, \alpha_{t,n}\right)} \\ &= \frac{e^{\mu_{t,n}\tau} C_{t,n}^{MJD}\left(\tau, S_{t,n}, \sigma_{t,n}, \delta_{t,n}, \lambda_{t,n}, K_{t,n}, \mu_{t,n}, \alpha_{t,n}\right)}{C_{t,n}^{MJD}\left(\tau, S_{t,n}, \sigma_{t,n}, \delta_{t,n}, \lambda_{t,n}, K_{t,n}, r^f, \alpha_{t,n}\right)}.\end{aligned} \tag{47}$$

Now that we have derived the expected holding period return of an option under MJD, we briefly describe the simulation procedure. To compare the performance of the simulation study under MJD to BS, we use the same simulated sample. We extend this sample by randomly simulating values of skewness from a fitted empirical distribution.

[6] The infinite sum in (43) is quickly converging. Therefore, we cut off the sum at $s = 40$ when implementing the MJD model in Python. We find that this yields sufficiently accurate results.

Given the first three moments of the stock return distribution, we derive the parameters of the MJD model. The kurtosis is calculated from the obtained MJD parameters. This procedure is outlined in detail in section 6.2. Next, we simulate realized stock returns that follow the MJD price dynamic in Equation (41).

We already used the factor model to determine the moments of the stock return distribution. Therefore, using the factor model to simulate realized returns and subsequently overlaying it with simulated jumps leads to incorrect moments of the stock distribution. Instead, we simulate the realized stock price using Equation (42). This entails sampling from a standard normal distribution to obtain $z_{t+1,n}$ and sampling the jumps to obtain $Y_{t+1,n}(s_{t+1,n})$. We simulate the number of jumps from a Poisson distribution and the individual jump size from a log-normal distribution, in line with the assumptions. To mirror the correlations implied by the factor model, correlated shocks $z_{t+1,n}$ would need to be drawn based on a scaled $N \times N$ covariance matrix for each period. This entails decomposing each of the T covariance matrices by Cholesky decomposition. We face computational issues when implementing this approach. Therefore, we attempt to preserve the correlations by setting $z_{t+1,n}$ to the normalized stock return we obtain from the factor model when simulating Black-Scholes returns. For firm n we obtain:

$$z_{t+1,n} = \frac{1}{\sqrt{\text{Var}_t[r_{t+1,n}]}}\left(\sum_k \beta_{k,n} f_{k,t+1} + \epsilon_{t+1,n} - \mathbb{E}[r_{t+1,n}]\right), \tag{48}$$

where $\mathbb{E}[r_{t+1,n}]$, $\text{Var}_t[r_{t+1,n}]$, $\beta_{k,n}$, $f_{k,t+1}$, and $\epsilon_{t+1,n}$ are obtained from the simulation under BS. The innovation $z_{t+1,n}$ is normally distributed, as the factor returns $f_{k,t+1}$ and the idiosyncratic shock $\epsilon_{t+1,n}$ are assumed to be normally distributed. This procedure leads to the same correlation among the continuous components of the stock returns as implied by the factor model. This is shown in detail in Appendix H. However, we note that the total correlation among stock returns is lower, as the jumps add an uncorrelated component to the stock return variance.

Finally, we train the candidate models on the moneyness of the option contract and the first four moments of the stock return distribution. These are the expected value, variance, skewness, and kurtosis of the stock returns. Again, we use R^2_{OOS} to measure performance. Using a similar set of predictors lets us directly compare the performance of the models under MJD against their performance under BS. In addition, we show the expected R^2 of

the MJD prediction model to obtain a more robust benchmark. Calculating the expected R^2 is more advanced than under BS and involves applying the law of total variance. We provide a complete derivation in Appendix I.

6.2 Parameter Calibration

There are five parameters to be set to calculate the option holding period return under MJD, namely $\sigma_{t,n}$, $\delta_{t,n}$, $\lambda_{t,n}$, $\alpha_{t,n}$, and $\mu_{t,n}$. One approach to calibrating the parameters is to fit the pricing formula in Equation (6) to historical data (He et al., 2007). In this paper, we follow a different approach. We obtain the first three moments of the returns from historical stock prices and then solve for the MJD parameters. The closed-form solution for the first three moments of the stock return distribution under an MJD price process is given by:

$$\mathbb{E}[r_{t+1,n}] = \mu_{t,n} - \frac{\sigma_{t,n}^2}{2} - \lambda_{t,n}\left(e^{\alpha_{t,n}+\frac{1}{2}\delta_{t,n}^2} - 1\right) + \lambda_{t,n}\alpha_{t,n}, \tag{49}$$

$$\mathrm{Var}_t[r_{t+1,n}] = \sigma_{t,n}^2 + \lambda_{t,n}\delta_{t,n}^2 + \lambda_{t,n}\alpha_{t,n}^2, \tag{50}$$

$$\mathrm{Skew}_t[r_{t+1,n}] = \frac{\lambda_{t,n}\left(3\delta_{t,n}^2\alpha_{t,n} + \alpha_{t,n}^3\right)}{\mathrm{Var}_t[r_{t+1,n}]^{3/2}}, \tag{51}$$

Further, we assume that:

$$\lambda_{t,n}, \sigma_{t,n}^2, \delta_{t,n}^2 > 0, \tag{52}$$

as the MJD pricing formula only allows for real-valued input parameters. Further, this ensures that we obtain economically reasonable results.

We obtain $\mathbb{E}[r_{t+1,n}]$ and $\mathrm{Var}_t[r_{t+1,n}]$ from the factor model under BS. This allows us to directly compare the performance of the candidate models under MJD with the performance under BS. Further, we sample $\mathrm{Skew}_t[r_{t+1,n}]$ from a log-normal distribution fitted to historical stock return data. The empirical distribution and the fitted distribution of return skewness are provided in Appendix E. To obtain a unique solution to the system of equations spanned by Equations (49)-(51), we need to set two parameters to a self-chosen value. For convenience, we decide to set $\alpha_{t,n}$ and $\delta_{t,n}^2$.

Combining Equations (49)-(51) and the non-negativity conditions (52), we obtain the following boundary conditions for $\lambda_{t,n}$ and $\alpha_{t,n}$:

$$0.1 \leq \lambda_{t,n} < \min\left(1, \frac{3\alpha_{t,n} \cdot \text{Var}_t[r_{t+1,n}] - \text{Skew}_t[r_{t+1,n}] \cdot \text{Var}_t[r_{t+1,n}]^{3/2}}{2\alpha_{t,n}^3}, \frac{\text{Skew}_t[r_{t+1,n}] \cdot \text{Var}_t[r_{t+1,n}]^{3/2}}{\alpha_{t,n}^3}\right) \quad (53)$$

$$\begin{aligned} \alpha_{t,n} &> \frac{\text{Skew}_t[r_{t+1,n}]\sqrt{\text{Var}_t[r_{t+1,n}]}}{3} \quad \text{, if } \text{Skew}_t[r_{t+1,n}] \geq 0, \\ &< \frac{\text{Skew}_t[r_{t+1,n}]\sqrt{\text{Var}_t[r_{t+1,n}]}}{3} \quad \text{, else.} \end{aligned} \quad (54)$$

A detailed derivation can be found in Appendix J. Note that we set a self-chosen upper boundary of 1 for the parameter $\lambda_{t,n}$ to prevent obtaining unreasonably high jump intensity rates. Further, we set the lower boundary of $\lambda_{t,n}$ to 0.1. This ensures that, for given values of $\mathbb{E}[r_{t+1,n}]$, $\text{Var}_t[r_{t+1,n}]$, and $\text{Skew}_t[r_{t+1,n}]$, we do not obtain exploding values of kurtosis. Lastly, we decided to set $\alpha_{t,n} = 0.5 \cdot \text{Skew}_t[r_{t+1,n}]\sqrt{\text{Var}_t[r_{t+1,n}]}$. This ensures that condition (54) is always met.

Given the MJD parameters, we calculate the implied kurtosis using:

$$\text{Kurt}_t[r_{t+1,n}] = \frac{\lambda_{t,n}(3\delta_{t,n}^4 + 6\mu_{t,n}\delta_{t,n}^2 + \mu_{t,n}^4)}{\text{Var}_t[r_{t+1,n}]^{3/2}}. \quad (55)$$

The empirical distribution and sampled distribution of kurtosis can be found in Appendix K. They are sufficiently matching for our purpose.

6.3 Results and Discussion

The performance of the Machine Learning models in predicting option holding period returns under the MJD-type stock returns is shown in Figure 6. In addition, the Figure shows the performance of the candidate models under BS-type stock returns. The models are trained on the same simulated training set as used under BS. Only simulated values of skewness and kurtosis are added to the set of predictors. Similar to before, the performance is then measured on true out-of-sample data of size $T, N = 1000$. The R^2 of the MJD predictions, again serving as an upper boundary to all other models, declines slightly to 0.169% from 0.211% under BS. The R^2 of the Ridge Regression and the three tree-based models decreases in similar magnitude. Further, the R^2 of the Lasso Regression and ENet increases slightly but remains close to zero. We observe that PCR is unable to cope with the extra layer of complexity added by the MJD model: the R^2 decreases drastically to

only 0.019%. Lastly, we observe that both ensemble models decrease in performance. However, the decline is more pronounced for the linear ensemble model than the non-linear ensemble. The linear ensemble model achieves an R^2 of 0.086%, while the R^2 of the non-linear ensemble is 0.132%.

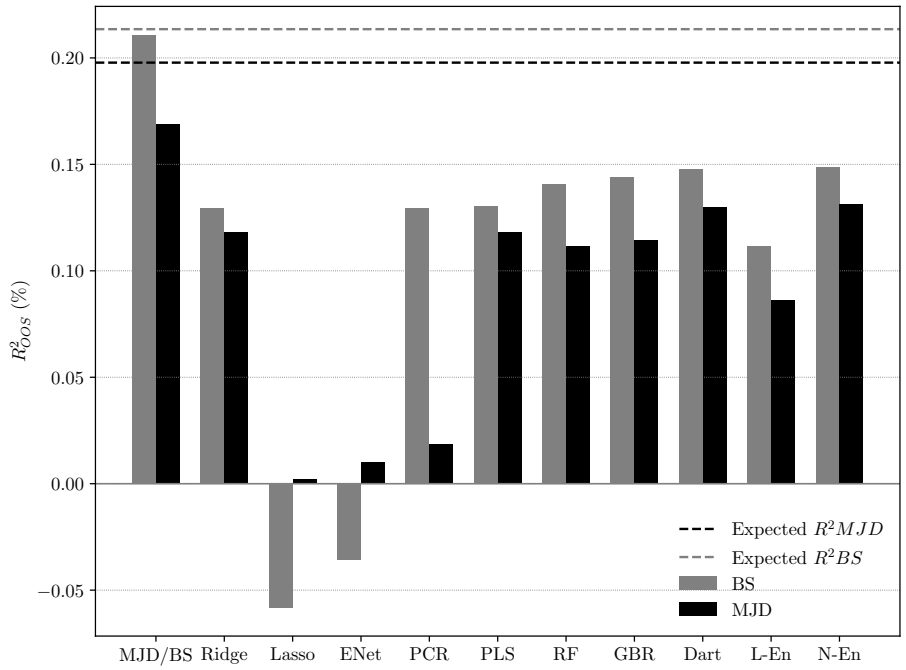


Figure 6: Merton Jump Diffusion Model Performance
This Figure shows the R^2_{OOS} of the candidate models under the assumption of MJD-type stock returns. Results are compared against the R^2_{OOS} of the base model. The models are trained and tested on simulated data of size $T, N = 1000$. In addition, the expected R^2_{OOS} for the MJD prediction model and the BS prediction model are presented.

In line with our expectations, the expected R^2 declines when introducing jumps to stock returns. Similarly, R^2 decreases for most models under MJD. We argue that the skewness and kurtosis embedded in MJD bring an extra layer of complexity and make returns harder to predict. We observe that Random Forest and GBR perform slightly worse under MJD than PLS and Ridge, the best performing linear models. Still, the best performing non-linear model, Dart, outperforms all other linear models, and the performance gap between the linear ensemble model and the non-linear ensemble widens. We suggest that non-linear models are better capable of capturing the skewness and kurtosis of returns under the MJD model. Furthermore, we hypothesize that the out-performance of the non-linear models will become even more pronounced the more complexity is added to the stock return process.

Next, we investigate the importance of each of the predictors for the candidate models. Similar to before, we drop one feature at a time and compare the performance with the R^2 under full feature availability. The results can be found in Table 3. We observe that the non-linear models are better capable of predicting returns under incomplete information and that they are less sensitive to feature dropping. For example, the non-linear ensemble continues to outperform the linear ensemble across all alterations. Further, Dart remains the strongest model across the three tree-regression models. However, Ridge and PLS continue to perform well and beat the non-linear ensemble slightly on two of the five alterations. Lastly, we observe that the expected stock return appears to be the strongest predictor, reaching the same conclusion as under BS-type stock returns. If the expected return is removed as a predictor, the performance of all models decreases drastically. Therefore, the expected return remains the crucial component in predicting returns, even after adding complexity to the return distribution.

Specification	**MJD**	**Exp. MJD**	**Ridge**	**Lasso**	**ENet**	**PCR**	**PLS**	**RF**	**GBR**	**DART**	**L_En**	**N_En**
Base MJD	0.169	0.198	0.118	0.002	0.010	0.019	0.118	0.112	0.115	0.130	0.086	0.132
Feature Selection Specification												
-$\mathbb{E}[r_{t+1,n}]$	0.169	0.198	0.009	-0.007	-0.002	0.005	0.009	0.008	0.013	0.014	0.011	0.014
-$\text{Var}_t[r_{t+1,n}]$	0.169	0.198	0.105	-0.007	-0.002	0.005	0.105	0.100	0.099	0.102	0.072	0.103
-$K_{t,n}/S$	0.169	0.198	0.091	-0.007	-0.007	0.003	0.091	0.090	0.091	0.091	0.057	0.093
-$\text{Skew}_t[r_{t+1,n}]$	0.169	0.198	0.106	-0.007	-0.002	0.017	0.106	0.089	0.101	0.105	0.075	0.103
-$\text{Kurt}_t[r_{t+1,n}]$	0.169	0.198	0.104	-0.008	-0.004	0.015	0.104	0.088	0.101	0.111	0.073	0.107

Table 3: Merton Jump Diffusion Performance under Altered Specifications
This Table shows the R^2_{OOS} in percent for five alterations to the feature set. The top row shows the R^2_{OOS} of the base MJD specification. The alterations include removing expected stock return, stock return variance, moneyness, stock return skewness, and stock return kurtosis.

7 Conclusion

In this paper, we examine the predictability of option holding period returns using Machine Learning methods. There exists an extensive literature on predicting option returns. However, the application of Machine Learning methods has only recently been explored. We contribute to the current literature by extending the framework of Bali et al. (2023) to a controlled environment. It is impossible to benchmark the performance of

prediction models when the true expected return is unknown. Using simulated data allows us to compute the maximum possible performance of any model under different stock price dynamics. By that, we can accurately evaluate the performance of Machine Learning methods against the maximum possible predictive power. Further, we can isolate any effect of changes to the simulation setup, helping us to understand the model performance under different conditions.

We analyze the performance of eight Machine Learning models and two ensemble models on simulated data and find that non-linear models outperform linear models in predicting hold-until-maturity option returns when assuming Black-Scholes-type stock dynamics. The non-linear ensemble yields an R^2_{OOS} of 0.152%, the highest performance across all models considered. We find that non-linear models continue to outperform when introducing discontinuous jumps to the stock price process. In addition, we find that the expected stock return is most useful in predicting hold-until-maturity option returns and that the time dimension in data is more important than the cross-sectional dimension. Lastly, non-linear models suffer heavily when data availability is reduced and when the expected stock return predictor is replaced with factor betas. A long-short trading strategy constructed on the predictions of the non-linear ensemble yields an annualized Sharpe Ratio of 1.1.

Compared to Bali et al. (2020), we find a lower magnitude of performance across all models, including the true expected return benchmark. We acknowledge that this is likely because different data is used and hold-until-maturity returns are more difficult to predict than delta-hedged returns. However, our results confirm that non-linear models are better suited for predicting option returns.

As discussed, we excluded Neural Networks from our analysis due to computational limitations. In addition, we tune over a sparser grid of hyper-parameters than in Bali et al. (2023). We suggest that future research extends the model framework and hyperparameters, potentially yielding even higher performance for non-linear models. Further, we hypothesize that the functional form of the assumed factor model can affect the model performance when the factor betas are used as predictors. We suggest a more in-depth analysis of introducing non-linearity in the factor model.

References

Albuquerque, R. (2012). Skewness in stock returns: Reconciling the evidence on firm versus aggregate returns. *The Review of Financial Studies*, 25:1630—1673.

Asness, C. S., Frazzini, A., and Pedersen, L. H. (2019). Quality minus junk. *Review of Accounting Studies*, 24:34–112.

Asness, C. S., Moskowitz, T. J., and Pedersen, L. H. (2013). Value and momentum everywhere. *The Journal of Finance*, 68:929–985.

Aziz, S., Dowling, M., Helmi, H., and Piepenbrink, A. (2022). Machine learning in finance: A topic modeling approach. *European Financial Management*, 28:744–770.

Bachelier, L. (1900). The theory of speculation. *Annales scientifiques de l'E cole Normale Superieur*, 17:21–86.

Bali, T. G., Beckmeyer, H., Mörke, M., and Weigert, F. (2023). Option return predictability with machine learning and big data. *The Review of Financial Studies*, 36:3548–3602.

Bali, T. G., Goyal, A., Huang, D., Jiang, F., and Wen, Q. (2020). Predicting corporate bond returns: Merton meets machine learning. *Working Paper*.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, 9:69—107.

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654.

Blitz, D. and van Vliet, P. (2007). The volatility effect: Lower risk without lower return. *Journal of Portfolio Management*, Fall 2007.

Boyer, B. H. and Vorkink, K. (2014). Stock options as lotteries. *The Journal of Finance*, 69:1485–1527.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. J. (1984). *Classification and Regression Trees*. Chapman and Hall.

Broadie, M., Chernov, M., and Johannes, M. (2009). Understanding index option returns. *The Review of Financial Studies*, 22:4493–4529.

Büchner, M. and Kelly, B. (2022). A factor model for option returns. *Journal of Financial Economics*, 143:1140–1161.

Carr, P. and Sun, J. (2007). A new approach for option pricing under stochastic volatility. *Review of Derivatives Research*, 10:87—150.

Chicco, D., Warrens, M., and Jurman, G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science, Working Paper*.

Coval, J. D. and Shumway, T. (2001). Expected option returns. *The Journal of Finance*, 56:983–1009.

Cox, J. C. and Ross, S. A. (1976). The valuation of options for alternative stochastic processes. *Journal of Financial Economics*, 3:145–166.

Duarte, J., Jones, C. S., and Wang, J. L. (2023). Very noisy option prices and inference regarding the volatility risk premium. *The Journal of Finance, forthcoming*.

Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, 38(1):34–105.

Fama, E. F. and French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33:3–56.

Fama, E. F. and French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics*, 116:1–22.

Feng, G., Giglio, S., and Xiu, D. (2020). Taming the factor zoo: A test of new factors. *The Journal of Finance*, 75:1327–1370.

Frazzini, A. and Pedersen, L. H. (2014). Betting against beta. *Journal of Financial Economics*, 111:1–25.

French, K. R., Schwert, G. W., and F. Stambaugh, R. (1987). Expected stock returns and volatility. *Journal of Financial Economics*, 19:2–29.

Freyberger, J., Neuhierl, A., and Weber, M. (2020). Dissecting characteristics nonparametrically. *The Review of Financial Studies*, 33:2326–2377.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29:1189–1232.

Friedman, J., Tibshirani, R., and Hastie, T. (2008). *The Elements of Statistical Learning*. Springer.

Goyal, A. and Saretto, A. (2009). Cross-section of option returns and volatility. *Journal of Financial Economics*, 94:310–326.

Green, J., Hand, J. R., and Zhang, F. (2012). The supraview of return predictive signals. *Review of Accounting Studies*.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33:2223–2273.

He, C., Kennedy, J. S., Coleman, T. F., Forsyth, P. A., Li, Y., and R., V. K. (2007). Calibration and hedging under jump diffusion. *Review of Derivatives Research*, 9:1–35.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6:327–343.

Hu, G. and Jacobs, K. (2020). Volatility and expected option returns. *Journal of Financial and Quantitative Analysis*, 55:1025—1060.

Jordan, M. L. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349:255–260.

Kelly, B., Malamud, S., and Zhou, K. (2024). The virtue of complexity in return prediction. *The Journal of Finance*, 79.

Kelly, B., Palhares, D., and Pruitt, S. (2023). Modeling corporate bond returns. *The Journal of Finance*.

Kelly, B., Pruitt, S., and Su, Y. (2019). Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics*, 134:501–524.

Kelly, B., Pruitt, S., and Su, Y. (2020). Instrumented principal component analysis. *Working paper*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105,.

Mandelbrot, B. (1963). The variation of certain speculative prices. *The Journal of Business*, 36(4):394–419.

Merton, R., Scholes, M., and Gladstein, M. L. (1978). The returns and risk of alternative call option portfolio investment strategies. *The Journal of Business*, 51:183–24.

Merton, R., Scholes, M., and Gladstein, M. L. (1982). The returns and risks of alternative put-option portfolio investment strategies. *The Journal of Business*, 55:183–24.

Merton, R. C. (1976). Option pricing when underlying stock returns are discountinous. *The Journal of Financial Economics*, 3:125–144.

Mixon, S. (2011). What does implied volatility skew measure? *The Journal of Derivatives*, 18(4):9–25.

Moritz, B. and Zimmerman, T. (2016). Tree-based conditional portfolio sorts: The relation between past and future stock returns. *Working Paper*.

Rapach, D. E., Strauss, J. K., and Zhou, G. (2010). Out-of-sample equity premium prediction: Combination forecasts and links to the real economy. *The Review of Financial Studies*, 23:821–862.

Rapach, D. E., Strauss, J. K., and Zhou, G. (2013). International stock return predictability: What is the role of the united states? *The Journal of Finance*, 68:1633–1662.

Rashmi, K. and Gilad-Bachrach, R. (2015). Dart: Dropouts meet multiple additive regression trees. *arxiv*.

Rubinstein, M. (1976). The valuation of uncertain income streams and the pricing of options. *The Bell Journal of Economics*, 7:407—425.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:210–229.

Welch, I. and Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies*, 21:1455–1508.

Zhan, X., Han, B., Cao, J., and Tong, Q. (2022). Option return predictability. *The Review of Financial Studies*, 35:1394—1442.

Appendix

Appendix A1: Put Option Price

The pay-off of a put option at maturity date $t+\tau$ is given by:

$$P_{t+\tau,n} = \left(K_{t,n} - S_{t+\tau,n}\right)^{+}. \tag{A1}$$

Black and Scholes (1973) derive the price $P_{t,n}$ of a put option with a non-dividend-paying underlying with time to maturity τ. The price is computed as follows:

$$P_{t,n}\left(\tau, S_{t,n}, \sigma_{t,n}, K_{t,n}, r^f\right) = e^{-r^f\tau} K_{t,n} \Phi\left(-d_2\right) - S_{t,n}\Phi\left(-d_1\right), \tag{A2}$$

$$d_1 = \frac{\log \frac{S_{t,n}}{K_{t,n}} + \left(r^f + \frac{\sigma_{t,n}^2}{2}\right)\tau}{\sigma_{t,n}\sqrt{\tau}},$$

$$d_2 = d_1 - \sigma_{t,n}\sqrt{\tau},$$

where τ denotes the time to maturity, $S_{t,n}$ is the stock price of firm n at time t, $\sigma_{t,n}$ is the stock return volatility, $K_{t,n}$ is the strike price, and r^f the risk-free rate.

Appendix A2: Put Option Holding Period Return

The hold-until-maturity return is calculated analogously to the holding period return of a call option:

$$\mathbb{E}_t\left[R^{\text{put}}_{t+\tau,n}\right] = \frac{\mathbb{E}_t^P\left[\left(K_{t,n} - S_{t+\tau,n}\right)^{+}\right]}{P_{t,n}\left(\tau, S_{t,n}, \sigma_{t,n}, K_{t,n}, r^f\right)} = \frac{e^{\mu_{t,n}\tau} P_{t,n}\left(\tau, S_{t,n}, \sigma_{t,n}, K_{t,n}, \mu_{t,n}\right)}{P_{t,n}\left(\tau, S_{t,n}, \sigma_{t,n}, K_{t,n}, r^f\right)}. \tag{A3}$$

Appendix B: Upper Boundary of Expected R^2

In this Section, we explain why the expected R^2 of the true expected return is higher than the expected R^2 of any other model. To see this, we define $c_{t+1,n} = \hat{R}_{t+1,n} - \mathbb{E}\left[R_{t+1,n}\right]$, where $\hat{R}_{t+1,n}$ are the option return predictions of some arbitrary model. First, we observe that:

$$\mathbb{E}\left[\sum_{(t,n)} \left(R_{t+1,n} - \left(\mathbb{E}_t\left[R_{t+1,n}\right] + c_{t+1,n}\right)\right)^2\right] \tag{B1}$$

$$= \sum_{(t,n)} \mathbb{E}_t\left[\left(R_{t+1,n} - \left(\mathbb{E}_t\left[R_{t+1,n}\right] + c_{t+1,n}\right)\right)^2\right] \tag{B2}$$

$$= \sum_{(t,n)} \mathbb{E}_t\left[R_{t+1,n}^2 - 2R_{t+1,n}\mathbb{E}_t[R_{t+1,n}]\right. \tag{B3}$$

$$\left. -2R_{t+1,n}c_{t+1,n} + \mathbb{E}_t[R_{t+1,n}]^2 + 2\mathbb{E}_t[R_{t+1,n}]c_{t+1,n} + c_{t+1,n}^2\right]$$

$$= \sum_{(t,n)} \mathbb{E}_t\left[\left(R_{t+1,n} - \mathbb{E}_t[R_{t+1,n}]\right)^2\right] \tag{B4}$$

$$+ \sum_{(t,n)} \mathbb{E}_t\left[2\mathbb{E}_t[R_{t+1,n}]c_{t+1,n} + c_{t+1,n}^2 - 2R_{t+1,n}c_{t+1,n}\right] \tag{B5}$$

$$= \sum_{(t,n)} \mathrm{Var}_t\left[R_{t+1,n}\right] + \sum_{(t,n)} c_{t+1,n}^2 \tag{B6}$$

$$\geq \sum_{(t,n)} \mathrm{Var}_t\left[R_{t+1,n}\right]. \tag{B7}$$

Using Equation (12), we finally get:

$$\mathbb{E}[R_{arb}^2] = 1 - \frac{\mathbb{E}\left[\sum_{(t,n)}\left(R_{t+1,n} - \hat{R}_{t+1,n}\right)^2\right]}{\mathbb{E}\left[\sum_{(t,n)}(R_{t+1,n} - \bar{r})^2\right]} \tag{B8}$$

$$= 1 - \frac{\mathbb{E}\left[\sum_{(t,n)}\left(R_{t+1,n} - \left(\mathbb{E}_t\left[R_{t+1,n}\right] + c_{t+1,n}\right)\right)^2\right]}{\mathbb{E}\left[\sum_{(t,n)}(R_{t+1,n} - \bar{r})^2\right]} \tag{B9}$$

$$\leq 1 - \frac{\mathrm{Var}_t\left[R_{t+1,n}\right]}{\mathbb{E}\left[\sum_{(t,n)}(R_{t+1,n} - \bar{r})^2\right]} \tag{B10}$$

$$= \mathbb{E}[R_{BS}^2], \tag{B11}$$

where $\mathbb{E}[R_{arb}^2]$ denotes the expected R^2 of some arbitrary prediction model.

Appendix C: Hyperparameter Grid

This Table shows the grid of hyperparameters. $\mathcal{LU}$ refers to drawing from a log-uniform distribution within the specified boundaries. For the tree-based models, a random search technique is employed. Out of all possible parameter combinations, a predefined number of combinations is evaluated at random.

	Ridge, Lasso, & ENet	
	Search trials	10
	Penalization strength	$\mathcal{LU}(1e^{-6}, 1e^{-2})$
ENet	Penalization ratio	0.5
	PCR & PLS	
	Search trials	X, the number of predictors
	Number of components	$\in [1, 2, ..., X]$
	RF, GBR, & Dart	
	Max trees	$\in [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]$
	Random search trials	40
	Max depth per tree	$\in [3, 5, 7, 10]$
	Max number of leaves per tree	$\in [3, 5, 7, 10, 15, 20, 30, 40, 50, 70, 90, 100]$
	Bagging fraction	$\in [0.8, 1]$
GBR & Dart	Learning rate	$\in [0.01, 0.1, 1]$
Dart	Fraction of features per run	$\in [0.8, 1]$
Dart	L_1 regularization	$\in [0, 0.01]$
Dart	L_2 regularization	$\in [0, 0.01]$
Dart	Dropout probability	$\in [0.05, 0.1, 0.15]$
Dart	Probability of skipping dropout	$\in [0.05, 0.1, 0.15]$

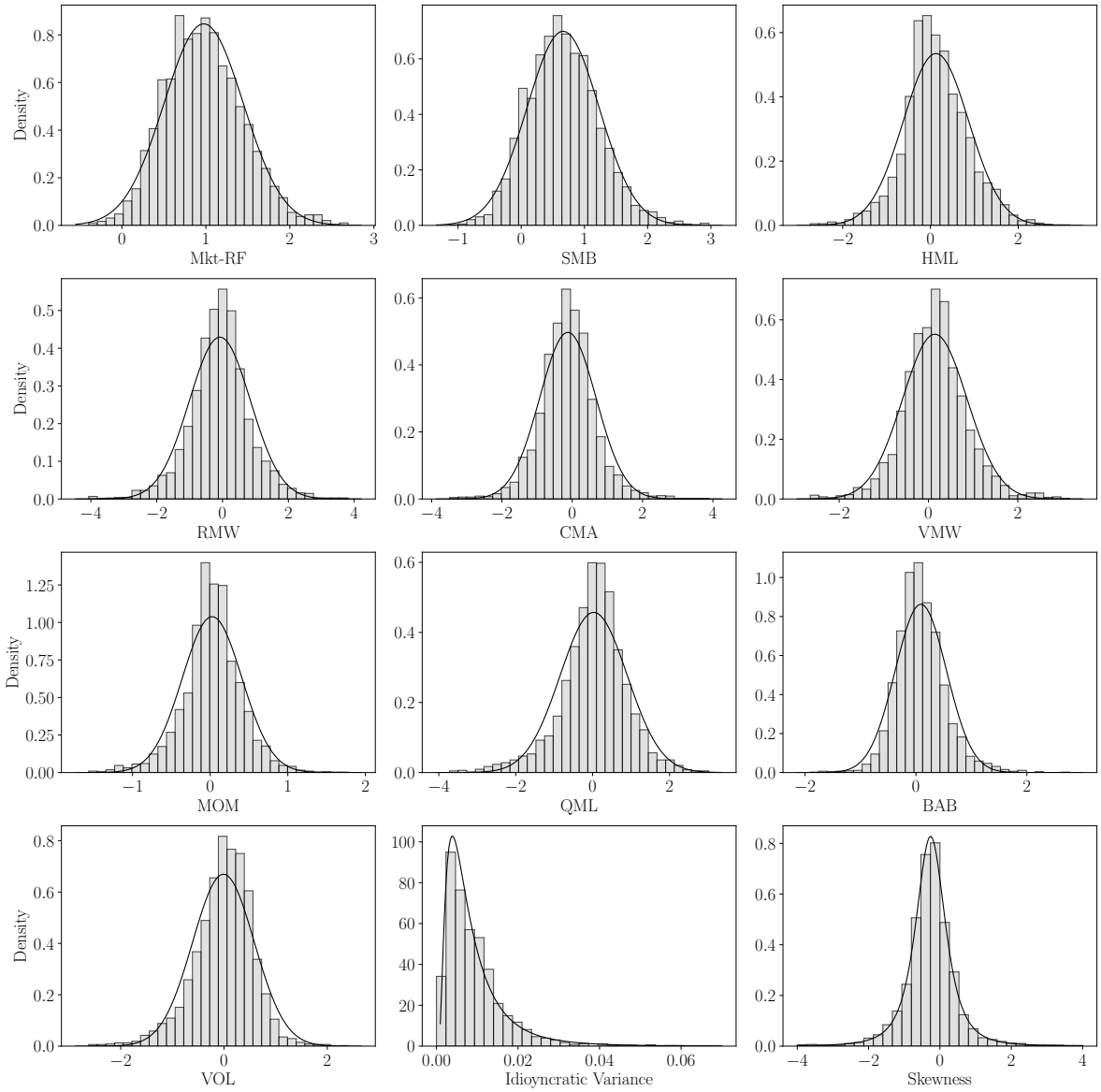
Appendix D: Summary Statistics of Empirical Data

This Table shows the summary statistics of the historical data used to simulate data.

Data	Obs.	Median	Mean	Std. Dev.	Skew.	Ex. Kurtosis
Stock Return	524,498	0.0091	0.0075	0.1134	-0.0939	10.2064
Risk-Free Rate	240	0.0002	0.0010	0.0012	1.3336	0.7170
Mkt-RF Return	240	0.0131	0.0070	0.0444	-0.7639	1.8518
SMB Return	240	0.0014	0.0011	0.0251	0.1067	0.2417
HML Return	240	-0.0018	-0.0003	0.0306	-0.1834	3.7701
RMW Return	240	0.0027	0.0026	0.0195	0.1376	1.5820
CMA Return	240	-0.0002	0.0013	0.0183	0.4525	1.8893
VAL Return	240	-0.0020	0.0001	0.0300	0.3383	3.3636
MOM Return	240	-0.0001	-0.0013	0.0366	-0.1579	1.5803
QML Return	240	0.0019	0.0029	0.0260	-0.1261	1.3105
BAB Return	240	0.0062	0.0054	0.0277	-0.3495	2.9103
VOL Return	240	0.0071	0.0046	0.0293	-0.3159	1.3216

Appendix E: Empirical Distributions

This Figure shows the empirical distributions of the beta exposures to each of the ten factors. The beta coefficients were obtained by estimating the assumed factor model through OLS regression using historical stock and factor return data. To each beta distribution, a normal distribution is fitted. Further, the Figure shows the empirical distribution of idiosyncratic variances with fitted log-normal distribution. The idiosyncratic variance of an individual stock is computed as the sample variance of the OLS residuals. Lastly, the Figure shows the empirical distribution of return skewness with a fitted t-distribution. The skewness of an individual stock is computed as the sample skewness of the historical excess log-returns of the stock.



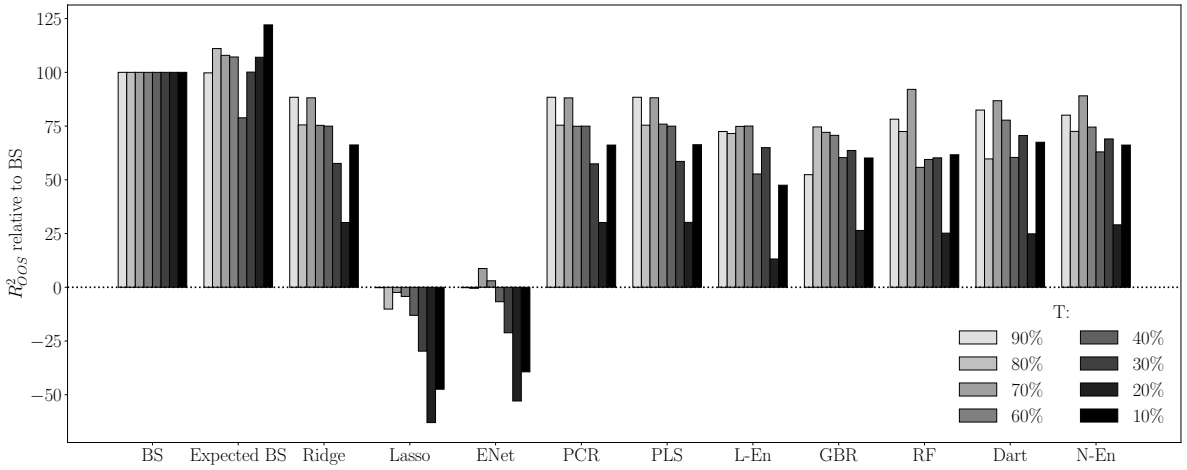
Appendix F: Summary Statistics of Simulated Data

This Figure shows the summary statistics of the in-sample simulated data. Note that, since the risk premium values are constant, no skewness and kurtosis values are provided.

	Median	**Mean**	**Std. Dev.**	**Min**	**Max**	**Skew.**	**Ex. Kurt.**
β_1	0.96	0.96	0.47	-0.50	2.23	-0.09	-0.06
β_2	0.67	0.68	0.55	-0.98	2.80	0.06	0.01
β_3	0.20	0.16	0.72	-2.28	2.17	-0.05	-0.05
β_4	-0.04	-0.09	0.91	-3.16	3.04	-0.10	0.14
β_5	-0.14	-0.15	0.75	-2.20	2.37	-0.03	-0.16
β_6	0.12	0.13	0.70	-2.23	2.33	0.05	0.31
β_7	0.04	0.04	0.37	-1.02	1.32	0.02	-0.06
β_8	0.05	0.05	0.86	-2.57	3.27	0.12	0.07
β_9	0.04	0.06	0.44	-1.23	1.35	0.17	-0.09
β_{10}	-0.04	-0.03	0.60	-2.10	2.14	0.01	-0.02
$\frac{K}{S}$	1.00	1.00	0.06	0.90	1.10	-0.00	-1.20
Variances in %							
$\text{Var}[f_1]$	0.20	0.20	0.01	0.16	0.23	0.05	-0.36
$\text{Var}[f_2]$	0.06	0.06	0.00	0.05	0.08	0.10	-0.07
$\text{Var}[f_3]$	0.09	0.09	0.01	0.08	0.11	0.14	-0.08
$\text{Var}[f_4]$	0.04	0.04	0.00	0.03	0.05	0.21	0.12
$\text{Var}[f_5]$	0.03	0.03	0.00	0.03	0.04	0.19	0.07
$\text{Var}[f_6]$	0.09	0.09	0.00	0.08	0.11	0.31	0.19
$\text{Var}[f_7]$	0.13	0.13	0.01	0.11	0.16	0.43	0.26
$\text{Var}[f_8]$	0.07	0.07	0.00	0.05	0.08	0.05	-0.02
$\text{Var}[f_9]$	0.08	0.08	0.00	0.06	0.09	0.20	-0.06
$\text{Var}[f_{10}]$	0.08	0.09	0.00	0.07	0.10	0.17	0.28
$\text{Var}[\epsilon]$	0.68	0.91	0.82	0.01	21.66	3.39	23.75
$\text{Var}[r]$	1.24	1.44	0.86	0.10	22.43	3.02	20.30
Returns in %							
$\mathbb{E}[f_1]$	0.70	0.70	0.00	0.70	0.70	-	-
$\mathbb{E}[f_2]$	0.11	0.11	0.00	0.11	0.11	-	-
$\mathbb{E}[f_3]$	-0.03	-0.03	0.00	-0.03	-0.03	-	-
$\mathbb{E}[f_4]$	0.26	0.26	0.00	0.26	0.26	-	-
$\mathbb{E}[f_5]$	0.13	0.13	0.00	0.13	0.13	-	-
$\mathbb{E}[f_6]$	0.01	0.01	0.00	0.01	0.01	-	-
$\mathbb{E}[f_7]$	-0.13	-0.13	0.00	-0.13	-0.13	-	-
$\mathbb{E}[f_8]$	0.29	0.29	0.00	0.29	0.29	-	-
$\mathbb{E}[f_9]$	0.54	0.54	0.00	0.54	0.54	-	-
$\mathbb{E}[f_{10}]$	0.46	0.46	0.00	0.46	0.46	-	-
$\mathbb{E}[r]$	0.73	0.73	0.62	-1.66	2.72	-0.09	0.27
$\mathbb{E}[R^{\text{call}}]$	16.66	17.44	9.68	-28.36	99.16	0.43	0.94
R^{call}	-79.42	20.49	204.39	-100.00	9388.85	3.70	28.97

Appendix G: Time Dimension Alteration

This Figure shows the result of eight time dimension alterations. The R^2_{OOS} of the candidate models is shown relative to BS.



Appendix H: MJD Correlations

In this Section, we examine the correlations between stock returns derived from the modeling process described in Section 6. Recall the assumed factor model:

$$r_{t,n} = r_t^f + \sum_k \beta_{k,n} f_{k,t} + \epsilon_{t,n}. \tag{G1}$$

Under the assumption that the factor returns and the error terms are uncorrelated, the factor model implies the following correlation $\text{corr}_t^{fact}(r_{t+1,p}, r_{t+1,q})$ between the returns of firm p and q:

$$\begin{aligned}
\text{corr}_t^{fact}(r_{t+1,p}, r_{t+1,q}) &= \frac{\text{Cov}\left[\sum_k \beta_{k,p} f_{k,t+1} + \epsilon_{t+1,p}, \sum_k \beta_{k,q} f_{k,t+1} + \epsilon_{t+1,q}\right]}{\sqrt{\text{Var}_t[r_{t+1,p}]}\sqrt{\text{Var}_t[r_{t+1,q}]}} && \text{(G2)} \\
&= \frac{1}{\sqrt{\text{Var}_t[r_{t+1,p}]}\sqrt{\text{Var}_t[r_{t+1,q}]}} \sum_k \beta_{k,p}\beta_{k,q} \,\text{Cov}_t[f_{k,t+1}, f_{k,t+1}] && \text{(G3)} \\
&= \frac{1}{\sqrt{\text{Var}_t[r_{t+1p}]}\sqrt{\text{Var}_t[r_{t+1q}]}} \sum_k \beta_{k,p}\beta_{k,q} \,\text{Var}_t[f_{k,t+1}]. && \text{(G4)}
\end{aligned}$$

In Section 6 we simulate stock returns by sampling from the following MJD-type stock dynamic:

$$S_{t+1,n} = S_{t,n} e^{\left(\mu_{t,n} - \frac{1}{2}\sigma_{t,n}^2\right) + \sigma_{t,n} W_{t+1,n}} Y_{t+1,n}(s_{t+1,n}). \tag{G5}$$

This implies the following log-return:

$$r_{t+1,n}^{MJD} = \left(\mu_{t,n} - \frac{1}{2}\sigma_{t,n}^2\right) + \sigma_{t,n}\sqrt{t} z_{t+1,n} + \log(Y_{t+1,n}(s_{t+1,n})). \tag{G6}$$

Hence, the stock return is composed of a constant, a continuous diffusion $r_{t,n}^{cont} = \sigma_{t,n}\sqrt{t} z_{t,n}$ and a discontinuous jump component $r_{t,n}^{jump} = \log(Y_{t+1,n}(s_{t+1,n}))$. We assume the jumps and the continuous diffusion to be uncorrelated and set

$$z_{t+1,n} = \frac{1}{\sqrt{\text{Var}_t[r_{t+1,n}]}} \left(\sum_k \beta_{k,n} f_{k,t+1} + \epsilon_{t+1,n} - \mathbb{E}[r_{t+1,n}]\right). \tag{G7}$$

We only consider monthly returns and set $t = 1$. Therefore, the correlation between firm p and q is given by:

$$\operatorname{corr}_t(r^{MJD}_{t+1,p}, r^{MJD}_{t+1,q}) = \frac{\operatorname{Cov}_t\left[r^{MJD}_{t+1,p}, r^{MJD}_{t+1,q}\right]}{\sqrt{\operatorname{Var}_t[r_{t+1,p}]\operatorname{Var}_t[r_{t+1,q}]}} \tag{G8}$$

$$= \frac{\sigma_{t,p}\sigma_{t,p}}{\sqrt{\operatorname{Var}_t[r_{t+1,p}]\operatorname{Var}_t[r_{t+1,q}]}} \operatorname{Cov}_t[z_{t+1,p}, z_{t+1,q}] \tag{G9}$$

$$= \frac{\sigma_{t,p}\sigma_{t,p}}{\operatorname{Var}_t[r_{t+1,p}]\operatorname{Var}_t[r_{t+1,q}]} \sum_k \beta_{k,p}\beta_{k,q} \operatorname{Var}_t[f_{k,t+1}] \tag{G10}$$

$$= \frac{\sigma_{t,p}\sigma_{t,p}}{\sqrt{\operatorname{Var}_t[r_{t+1,p}]\operatorname{Var}_t[r_{t+1,q}]}} \operatorname{corr}^{fact}_t(r_{t+1,p}, r_{t+1,q}). \tag{G11}$$

Hence, the correlations we obtain from our simulation procedure are scaled versions of the correlations implied by the factor model. Using $\sigma^2_{t,n} = \operatorname{Var}_t[r_{t+1,n}] - \lambda_{t,n}\delta^2_{t,n} - \lambda_{t,n}\alpha^2_{t,n}$, we see that the correlations among the simulated returns under the MJD-type stock dynamic are smaller than the correlations implied by the factor model. However, our simulation procedure yields the same correlations among the continuous components of the stocks:

$$\operatorname{corr}_t(r^{cont}_{t+1,p}, r^{cont}_{t+1,q}) = \frac{\operatorname{Cov}_t\left[r^{cont}_{t+1,p}, r^{cont}_{t+1,q}\right]}{\sqrt{\operatorname{Var}_t[r^{cont}_p]\operatorname{Var}_t[r^{cont}_q]}} \tag{G12}$$

$$= \frac{\sigma_{t,p}\sigma_{t,q}}{\sqrt{\operatorname{Var}_t[r^{cont}_{t+1,p}]\operatorname{Var}_t[r^{cont}_{t+1,q}]}} \operatorname{Cov}_t[z_{t+1,p}, z_{t+1,q}] \tag{G13}$$

$$= \frac{1}{\sqrt{\operatorname{Var}_t[r_{t+1,p}]\operatorname{Var}_t[r_{t+1,q}]}} \sum_k \beta_{k,p}\beta_{k,q} \operatorname{Var}_t[f_{k,t+1}] \tag{G14}$$

$$= \operatorname{corr}^{fact}_t(r_{t+1,p}, r_{t+1,q}). \tag{G15}$$

In Equation (G14) we used $\sqrt{\operatorname{Var}_t[r^{cont}_{t+1,n}]} = \sqrt{\operatorname{Var}_t[\sigma_{t,n}z_{t+1,n}]} = \sigma_{t,n}$.

Appendix I: Expected R^2 under MJD

In this Section, we show how to calculate the expected R^2 for the MJD prediction model. In Section 3.3 we show that:

$$\mathbb{E}[R^2] = 1 - \frac{\sum_{(t,n)}(\mathbb{E}_t\left[R^2_{t+1,n}\right] - \mathbb{E}_t\left[R_{t+1,n}\right]^2)}{\sum_{(t,n)}\left(\mathbb{E}_t\left[R^2_{t+1,n}\right] - 2\bar{R}\mathbb{E}_t\left[R_{t+1,n}\right] + \bar{R}^2\right)}. \tag{I1}$$

In this case, $R_{t+1,n}$ refers to the hold-until-maturity return of an option under the MJD model. The conditional expected option return, $\mathbb{E}_t\left[R_{t+1,n}\right]$, can be computed using the MJD pricing formula in Equation (43). The conditional expected value of the squared return can be derived from:

$$\mathbb{E}_t\left[R^2_{t+1,n}\right] = \mathbb{E}_t\left[R_{t+1,n}\right]^2 + \mathrm{Var}_t[R_{t+1,n}]. \tag{I2}$$

The stock price process under the MJD model is driven by two stochastic components, namely the jump process and the Geometric Brownian Motion. Hence, the variance of the hold-until-maturity option return must be calculated using the law of total variance:

$$\mathrm{Var}_t[R_{t+1,n}] = \mathbb{E}\left[\mathrm{Var}_t\left[R_{t+1,n} \mid s_n = s\right]\right] + \mathrm{Var}\left[\mathbb{E}_t\left[R_{t+1,n} \mid s_n = s\right]\right]. \tag{I3}$$

We calculate the first term as follows:

$$\mathbb{E}_t\left[\mathrm{Var}_t\left[R_{t+1,n} \mid s_n = s\right]\right] = \frac{\mathbb{E}_t\left[\mathrm{Var}_t\left[(S^{MJD}_{t+1,n} - K)^+ \mid s_n = s\right]\right]}{\left(C^{MJD}_{t+\tau,n}\right)^2} \tag{I4}$$

$$= \frac{1}{\left(C^{MJD}_{t+\tau,n}\right)^2}\sum_{s=0}^{\infty}\mathbb{P}(s_n = s)\mathrm{Var}_t\left[(S^{BS}_{t+1,n} - K)^+ \mid s_n = s\right] \tag{I5}$$

$$= \frac{1}{\left(C^{MJD}_{t+\tau,n}\right)^2}\sum_{s=0}^{\infty}\frac{e^{-\lambda'\tau}\left(\lambda'\tau\right)^s}{s!}\mathrm{Var}_t\left[(S^{BS}_{t+1,n} - K)^+ \mid s_n = s\right], \tag{I6}$$

where $S^{MJD}_{t+1,n}$ denotes the price of the stock at maturity under the MJD model, and $S^{BS}_{t+1,n}$ the log-normally distributed price under the BS model. The parameter λ' is defined as in the MJD option pricing model. The call price under the MJD model, $C^{MJD}_{t+\tau,n}$, is calculated

using Equation (43). Further,

$$\mathrm{Var}_t\left[(S^{BS}_{t+1,n}-K)^+ \mid s_n=s\right] = \mathbb{E}_t\left[(S^{BS}_{t+1,n}-K)^+)^2 \mid s_n=s\right] - C^{BS}_{t,n}(\sigma_s,r_s)^2 \tag{I7}$$

$$= C^{BS}_{t,n}(\sigma_s,r_s)^2\mathbb{E}_t\left[(R^{BS}_{t+1,n})^2 \mid s_n=s\right] - C^{BS}_{t,n}(\sigma_s,r_s)^2. \tag{I8}$$

Conditional on that there are s jumps, the Black-Scholes call price $C^{BS}_{t,n}(\sigma_s,r_s)$ can be calculated by using the modified parameters σ_s and r_s defined in Section 6. The conditional expected return $\mathbb{E}_t\left[(R^{BS}_{t+1,n})^2 \mid s_n=s\right]$ is obtained by implementing the closed-form solution provided in Boyer and Vorkink (2014) with the modified parameters σ_s and r_s .

The second term in Equation (I3) can be computed as follows:

$$\mathrm{Var}\left[\mathbb{E}_t\left[R_{t+1,n} \mid s_n=s\right]\right] = \mathbb{E}\left[\left(\mathbb{E}_t\left[R_{t+1,n} \mid s_n=s\right] - \mathbb{E}\left[\mathbb{E}_t\left[R_{t+1,n} \mid s_n=s\right]\right]\right)^2\right] \tag{I9}$$

$$= \sum_{s=0}^{\infty} \frac{e^{-\lambda'\tau}\left(\lambda'\tau\right)^s}{s!}\left(\mathbb{E}_t\left[R^{BS}_{t+1,n}(\sigma_s,r_s)\right] - \mathbb{E}_t\left[R^{MJD}_{t+1,n}\right]\right)^2. \tag{I10}$$

The expected Black-Scholes return conditional on that there are s jumps, $\mathbb{E}_t\left[R^{BS}_{t+1,n}(\sigma_s,r_s)\right]$, can be calculated using the modified parameters σ_s and r_s and applying Equation (11). The holding return under MJD, $\mathbb{E}_t\left[R^{MJD}_{t+1,n}\right]$, is calculated using Equation (49).

Appendix J: Boundary Conditions of the MJD Parameters

In this Section, we derive the boundary conditions for the MJD parameters $\lambda_{t,n}$ and $\alpha_{t,n}$ we present in Section 6.2. These conditions constraint the interval on which we sample the parameters. First, we derive the upper boundary for $\lambda_{t,n}$. Rearranging Equation (51), we obtain:

$$\delta_{t,n}^2 = \left(\frac{\text{Skew}_t[r_{t+1,n}]\text{Var}_t[r_{t+1,n}]^{3/2}}{\lambda_{t,n}} - \alpha_{t,n}^3 \right) \frac{1}{3\alpha_{t,n}}. \tag{J1}$$

We require $\delta_{t,n}^2 > 0$. Hence:

$$\lambda_{t,n} < \frac{\text{Skew}_t[r_{t+1,n}]\text{Var}_t[r_{t+1,n}]^{3/2}}{\alpha_{t,n}^3}. \tag{J2}$$

Next, we rearrange Equation (50) to obtain:

$$\sigma_{t,n}^2 = \text{Var}_t[r_{t+1,n}] - \lambda_{t,n}\delta_{t,n}^2 - \lambda_{t,n}\alpha_{t,n}^2. \tag{J3}$$

$$= \text{Var}_t[r_{t+1,n}] - \lambda_{t,n}\left(\left(\frac{\text{Skew}_t[r_{t+1,n}]\text{Var}_t[r_{t+1,n}]^{3/2}}{\lambda_{t,n}} - \alpha_{t,n}^3 \right) \frac{1}{3\alpha_{t,n}} \right) - \lambda_{t,n}\alpha_{t,n}^2. \tag{J4}$$

We require $\sigma_{t,n}^2$. Rearranging Equation (J4) yields:

$$\lambda_{t,n} < \frac{3\alpha_{t,n} \cdot \text{Var}_t[r_{t+1,n}] - \text{Skew}_t[r_{t+1,n}] \cdot \text{Var}_t[r_{t+1,n}]^{3/2}}{2\alpha_{t,n}^3}. \tag{J5}$$

Lastly, we require that $\lambda_{t,n} > 0$. We obtain:

$$0 < \frac{\text{Var}_t[r_{t+1,n}] - \text{Skew}_t[r_{t+1,n}] \cdot \text{Var}_t[r_{t+1,n}]^{3/2}}{2\alpha_{t,n}^3}. \tag{J6}$$

Rearranging yields:

$$\begin{aligned} \alpha_{t,n} &> \frac{\text{Skew}_t[r_{t+1,n}]\sqrt{\text{Var}_t[r_{t+1,n}]}}{3} && \text{, if } \text{Skew}_t[r_{t+1,n}] \geq 0, \\ &< \frac{\text{Skew}_t[r_{t+1,n}]\sqrt{\text{Var}_t[r_{t+1,n}]}}{3} && \text{, else.} \end{aligned} \tag{J7}$$

The boundary conditions are derived. From Equation (J7) we see that $\alpha_{t,n}$ has the same sign as $\text{Skew}_t[r_{t+1,n}]$. Therefore, boundary condition (J2) never violates $\lambda_{t,n} > 0$.

Appendix K: Distribution of Excess Kurtosis under MJD

This Figure shows the empirical distribution of excess-return kurtosis values obtained from historical stock data and the distribution of the kurtosis values produced as residual parameters by the simulation. The simulated distribution is based on a simulation with $T, N = 1000$. The average kurtosis values of both distributions are are also presented.