**Erasmus University Rotterdam**

**Erasmus School of Economics**

Master Thesis: Quantitative Finance

# Predicting Individual Equity Options' Implied Volatility With Machine Learning

Name student: Leonardo Knol

Student ID number: 513082

Supervisor: Dr. M. Grith

Second assessor: Dr. A. Tetereva

Date final version: 06-08-2023

The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

## Abstract

*This study conducts a comparison between machine learning (ML) models for forecasting implied volatility (IV). Using monthly option data over the course of 25 years, I examine the predictive performance of different machine learning models in the US equity options market. My main finding is that an ensemble model (Ens) which consists of 5 neural network models exceeds the other models in terms of prediction accuracy. Additionally, I find that support vector regression (SVR) and XGBoost (XGB) demonstrate strong predictive power, suggesting their ability to discern nonlinear relationships and complex patterns in the data. On the other hand, linear models such as ordinary least squares (OLS) and elastic net (Enet) result in inferior predictions. Ultimately, the predictive performance cannot be attributed to any specific predictors, as the various models give ambiguous results concerning the driving forces of the predictions.*



## Keywords

Machine Learning, Options, Implied Volatility Prediction, Cross-Section of Equity Options, Implied Volatility Surface, Ridge Regression, Lasso Regression, Ordinary Least Squares, Elastic Net, Support Vector Regression, Random Forest, Neural Networks, Ensembles, Variable Importance, Fintech

# Contents

# 1 Introduction

The price of an option is commonly measured by its implied volatility, which can be considered as a one-to-one mapping between the option price and the market's expectation of the underlying asset's future volatility. The collection of implied volatilities of all available options on a particular asset, plotted against their moneyness and tenor, is called the implied volatility surface (IVS), which conveys important information about the market's expectations of the underlying asset's future return distribution at different horizons. As a result, accurately modeling and predicting the IVS is essential for investors, traders, and risk managers, as it can help in managing option positions and evaluating option pricing models. Nevertheless, this study will focus on predicting the implied volatility (IV) of individual equity options on a monthly basis using monthly observations for the cross-section of equities. Furthermore, the current econometric literature can be expanded by investigating different machine learning models to predict single equity options' IV, as the current literature focuses mostly on the IVS of S&P 500 index options (Chen and Zhang (2019), Zeng and Klabjan (2019), Aït-Sahalia et al. (2021), and Almeida et al. (2022))

## 1.1 Relevant Literature

Although several studies have focused on predicting the implied volatility (surface) in index options, less work is done on predicting the implied volatility (surface) in individual equity options. For example, Bernales and Guidolin (2014) found that a vector autoregressive (VAR) model that accounted for the dynamic linkages between the IVS of equity and S&P 500 index options outperformed several benchmarks in forecasting the IVS. Zeng and Klabjan (2019) developed an online adaptive primal support vector regression (SVR) to model the IVS using intraday tick data from the E-mini S&P options[1] market. Furthermore, Christoffersen et al. (2018) demonstrated that equity IVS levels, skews, and term structures exhibit a strong factor structure and are highly correlated with those same features of the S&P 500 IVS.

Moreover, recent advances in machine learning techniques have opened up new possibilities for modeling and predicting the IVS of individual equity options. For example, Gu et al. (2020) propose a nonparametric machine learning approach to model the cross-section of stock returns. Moreover, Ackerer et al. (2020)

[1] E-mini is an electronically traded futures contract that is one-fifth of the value of the standardized S&P 500 futures contract.

present a neural network (NN) approach to fit and predict the IVS. Thereby, the authors of the latter paper propose an approach that penalizes the loss using soft constraints in order to guarantee the absence of arbitrage opportunities. Furthermore, their method can be combined with standard IVS models, providing an NN-based correction when such models fail at replicating observed market prices. Almeida et al. (2022) propose a 2-step approach for predicting the IVS, wherein the first step the implied volatility is expressed as a parametric function, and the second step consists of applying a feedforward neural network to the errors of the regression. In this paper, I extend the work of Gu et al. (2020) to predict the IV of individual equity options, using similar machine-learning techniques. Furthermore, I find evidence of modeling non-linearities being very useful to predict the IVS.

The complexity of machine learning algorithms contains implications for their usage in the financial industry, namely, the intuition behind their predictions and decision-making is somewhat hidden behind a "black box." Li et al. (2020) propose a framework for interpreting machine learning models. Daul et al. (2022) attribute the performance of stock portfolios built using machine learning methods to linear, marginal non-linear, and interaction effects. In this research, I implement performance attribution measures from the aforementioned authors to discover the driving predictors for the IVS of individual equity options for the various models.

## 2 Data

I obtain daily individual equity implied volatilities from OptionMetrics for 10 US equities[2]. Table 1, depicts the specific equities that are considered. In order to avoid biasing the machine learning models towards a specific sector, I select equities across a range of 6 sectors, which is given in Table 1 along with the company names. This allows for the model to be more robust across different sectors.[3] The corresponding predictors include the option- and equity-specific features which are described in Table 2, which are also retrieved from OptionMetrics, excluding the intraday VIX data which is retrieved from Yahoo Finance. Furthermore, I use

[2]Ideally, the cross-section of options would contain a larger variety of equities. However, given the computational limitations, I have opted to select a cross-section consisting of ten equities.

[3]Poon and Granger (2003) compare 93 papers published on forecasting volatility and find that the predictability of the options is largely attributed to the sector and sampling frequency.

the implied volatilities of all the options on the third Friday[4] of each month to account for monthly implied volatility. Generally, the expiration date of listed US equity options is on the third Friday of the month.[5] This may lead to more accurate estimates of monthly implied volatility as this day tends to have more liquidity available. Furthermore, the third Friday of the month aligns with the expiration of many futures contracts, where predicting the implied volatility more accurately on this day can be beneficial for practitioners for hedging purposes and trading opportunities[6].

Similarly to Almeida et al. (2022), I select the out-of-the-money (OTM) options with moneyness between 0.8 and 1.2, as they can pose as better indicators for the market's expectations due to higher liquidity and tighter spreads. Moreover, the authors state that in-the-money (ITM) options can be discarded with no loss of information. Therefore, I drop all options that have no trading volume and all deep ITM options[7]. Finally, I only consider options with an expiration date of no less than 30 days and no longer than 240 days.[8]
My sample begins in January 1996[9] and ends in December 2021, allowing for 25 years. The panel is unbalanced due to the different assets containing numerous options.[10]

Table 1: List of Companies Taken Into Account in the Cross-Sectional Analysis

| **Company Name** | **Ticker** | **Sector** |
|---|---|---|
| **Apple Inc** | AAPL | Tech |
| **Boeing Co** | BA | Industrials |
| **Bristol-Myers Squibb** | BMY | Healthcare |
| **Citigroup Inc** | C | Financials |
| **FedEx Corp** | FDX | Consumer discretionary |
| **Johnson & Johnson** | JJ | Healthcare |
| **Coca-Cola Co** | KO | Consumer discretionary |
| **McDonald's Corp** | MCD | Consumer discretionary |
| **Microsoft Corp** | MSFT | Tech |
| **Exxon Mobil Corp** | XOM | Energy |

[4]On months that the third Friday of the month falls on a holiday, I use the first available trading day prior to the third Friday of the month

[5]Information on expiration dates are taken from CBOE options expiration calendar

[6]When investors can more accurately predict option volatility or hedge their positions better through a better estimate of volatility, this can allow them to have an edge when "betting on volatility" while keeping their portfolio relatively delta neutral. That is keeping their portfolio as unsusceptible as possible to changes in the price of the underlying while profiting from a change in volatility. For more information on trading volatility see e.g. Shover (2012) and Liu et al. (2021)

[7]Note that the ATM options are still kept in the dataset.

[8]Other research such as Almeida et al. (2022) consider options between 20 and 240 days. However, due to my research focussing on monthly predictions, I choose not to select options expiring within one month as this may lead to greater implied volatilities as compared to the other options, which can negatively bias the dataset.

[9]The earliest available option data on OptionMetrics is 04-01-1996

[10]Other papers such as Gu et al. (2020), also utilize an unbalanced panel in their research, which is common in cross-sectional prediction.

Table 2: The Specific Covariates Taken Into Account for the Various Models.

| **Predictor Variable** | **Symbol** |
|---|---|
| **Underlying stock price** | S |
| **Underlying stock price range**[*] | *S_spread* |
| **Underlying return percentage**[†] | Return % |
| **Option mid price** | Midprice |
| **Delta** | $\Delta$ |
| **Theta** | $\Theta$ |
| **Gamma** | $\Gamma$ |
| **Vega** | Vega |
| **Call or Put indicator** | *Cp_flag* |
| **Open Interest** | *Open_interest* |
| **Option traded volume** | Volume |
| **Time to expiration in days** | *Days_to_expiration* |
| **Moneyness**[‡] | Moneyness |
| **Strike price** | K |
| **VIX** | VIX |
| **VIX high minus low for the previous timestep** | VIX Range |

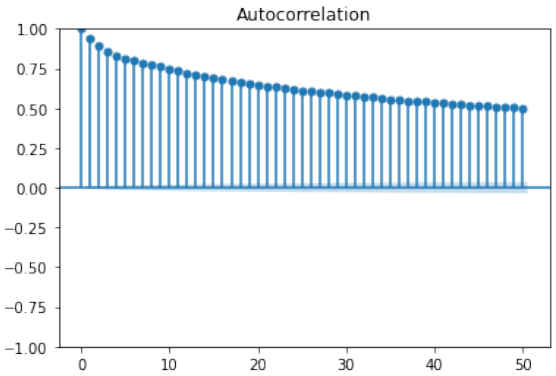
[*] The underlying price range describes the high−low of the price of the underlying stock for the given day.
[†] The return percentage is calculated as the % change in the underlying asset with respect to the previous timestep.
[‡] Moneyness is defined as the Strike price (K) divided by the current asset price (S)

Figure 1a depicts the autocorrelation present in the target variable throughout the first 50 lags. Hereby, it is evident from this figure that the autocorrelation slowly decays, however, it remains persistently above 0.50. This implies that the error term for the regression on the target variable needs to be corrected for autocorrelation when performing calculations that are susceptible to autocorrelation error terms due to the significant autocorrelation present.[11] Moreover, Figure 1b zooms in upon the first five lags of the partial autocorrelation function (pacf). This function is used for identifying the order of an autoregressive model, which in my case shows significant correlation during the first two lags.[12]

Furthermore, descriptive statistics of the implied volatility data are given in Table 3, with the corresponding boxplot in Figure 2. The columns of Panel A show the descriptive statistics for the shorter-term options in my dataset, grouped by moneyness (K/S), while Panel B depicts the same statistics over the same groups for further dated options (120-240 days to expiry). It is evident that the number of options trading for the short-dated options in Panel A is greater than the long-dated options in Panel B, which can be seen in the row "count."[13] Furthermore, when looking at the max implied volatility denoted by the row "max", we can see

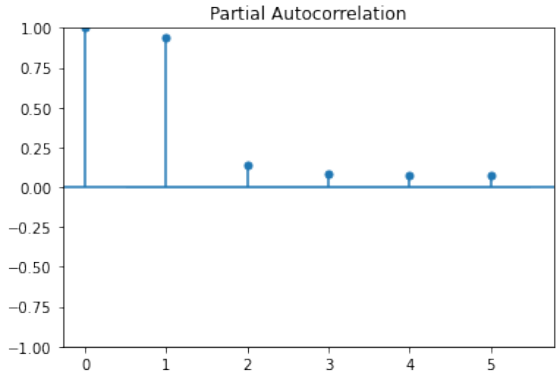
[11] For more information regarding the heteroscedastic and autocorrelation consistent (HAC) error terms implemented in the Diebold-Mariano (1995) test, see section 3.7.
[12] The partial autocorrelation function tells us that an AR model with at least two lags would be appropriate to model the target variable. In machine learning models it is also possible to incorporate the time lags of the target variable as a predictor.
[13] Note that the number of options here does not imply the trading frequency of options but rather the number of options available in the market with a nonzero volume. Thus, there may be a specific set of options that are much more frequently

Figure 1: Autocorrelation Analysis of Implied Volatility Through ACF and PACF




(a) ACF Plot of the Implied Volatility Including the First 50 Lags of At-the-Money Options

Note: This figure depicts the ACF of the implied volatility for the first 50 lags. Where the lags are given on the x-axis and the y-axis shows the level of autocorrelation.




(b) PACF of the Implied Volatility Including the first 5 Lags

Note: This Figure depicts the PACF of the implied volatility for the first 5 lags. From this Figure, it is evident that the order of the AR process appropriate would be 2 due to the first two lags being significantly greater than the others.

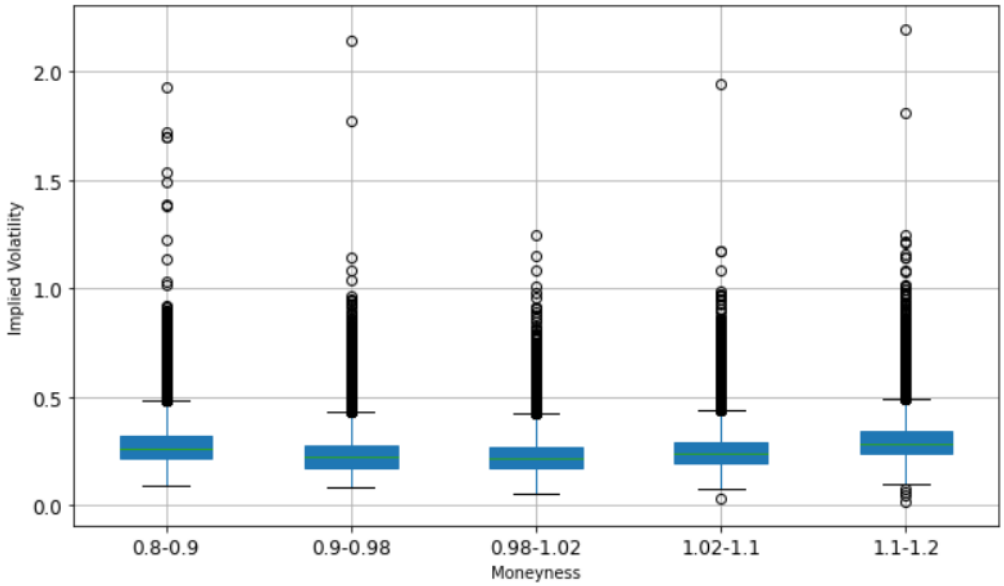
that Panel A portrays a maximum that is much greater than the maximum in Panel B, while the standard deviation for both panels are similar, implying that the short-dated options are much more susceptible to outliers. The implications for the research entail the predictability for the longer-dated options being higher due to the increased stability of the IV in these options.

Table 3: Summary Statistics of Implied Volatilities Across Different Groups of Moneyness (K/S) and Time-to-Maturity.

| Panel A: Descriptive Statistics (Short-term: 30-120 days) | | | | | |
|---|---|---|---|---|---|
| | $0.8 \leq K/S < 0.9$ | $0.9 \leq K/S < 0.98$ | $0.98 \leq K/S < 1.02$ | $1.02 \leq K/S < 1.1$ | $1.1 \leq K/S < 1.2$ |
| Count | 13986 | 24314 | 16517 | 23461 | 15085 |
| Mean | 28.95 | 23.67 | 22.82 | 25.40 | 30.91 |
| Std. dev | 10.14 | 9.25 | 8.82 | 8.98 | 9.65 |
| Min | 8.63 | 8.10 | 5.52 | 3.17 | 1.53 |
| 25% | 22.40 | 17.39 | 16.72 | 19.36 | 24.77 |
| 50% | 27.04 | 21.94 | 21.26 | 23.83 | 29.03 |
| 75% | 33.20 | 27.59 | 26.66 | 29.13 | 34.79 |
| Max | 192.27 | 213.73 | 124.57 | 194.18 | 219.62 |
| Panel B: Descriptive Statistics (Long-term: 120-240 days) | | | | | |
| | $0.8 \leq K/S < 0.9$ | $0.9 \leq K/S < 0.98$ | $0.98 \leq K/S < 1.02$ | $1.02 \leq K/S < 1.1$ | $1.1 \leq K/S < 1.2$ |
| Count | 9959 | 11835 | 6610 | 10794 | 9091 |
| Mean | 26.51 | 24.29 | 24.56 | 25.82 | 28.69 |
| Std. dev | 9.86 | 9.57 | 9.16 | 9.08 | 9.24 |
| Min | 8.84 | 9.32 | 9.61 | 9.04 | 6.35 |
| 25% | 20.07 | 17.64 | 18.01 | 19.53 | 22.73 |
| 50% | 24.35 | 22.36 | 22.79 | 24.06 | 26.65 |
| 75% | 30.78 | 28.23 | 28.47 | 29.58 | 32.34 |
| Max | 169.95 | 176.79 | 95.57 | 96.60 | 124.58 |

traded which is not directly accounted for in this statistic.

Figure 2: Boxplot of the Implied Volatility (y-axis) Grouped by Moneyness (x-axis)




Note: This figure demonstrates the IV theoretical quantiles in terms of a box plot. Hereby, it is also visible that there are a few extreme outliers for all the moneyness except ATM options. Furthermore, it is no coincidence that these outliers correspond to the huge IV spike during the financial crisis in 2009.

# 3 Methodology

In this chapter, I present a comprehensive description of the statistical and machine learning models used to conduct data analysis and evaluate the predictive performance of the models.

The approach used in this study involves a multistep process that includes model construction, model evaluation, and feature importance analysis. Thereby, a variety of regression models are constructed including linear regression, generalized linear model, elastic net regression, support vector regression, random forests, gradient-boosted regression trees, and neural networks. Subsequently, evaluation metrics are used to compare the performance of these models, including out-of-sample R-squared, and Root Mean Squared Error. The feature importance analysis is also carried out to determine which predictors contribute the most to the performance of the various models.

Overall, this methodology aims to provide a rigorous and comprehensive approach to analyzing the data and evaluating the performance of different models. Through this approach, we are able to gain insights into the relationships between variables and make accurate predictions for the dependent variable.

First, we begin by assuming that conditional volatilities are a flexible function of the covariates. Therefore,

the conditional implied volatilities can be written as :

$$E_t(\sigma_{i,t+1}) = g^*(x_{i,t}), \tag{1}$$

where the flexible function ($g^*(x_{i,t})$) is elaborated for each corresponding statistical model in their respective sections.

### 3.1 Linear Model: Ordinary Least Squares

The first benchmark model considered is OLS, which is one of the most commonly used methods in statistical regression analysis. This is a technique that fits a straight line through a set of data points to find the relationship between two variables. The OLS method estimates the parameters of the linear regression model by minimizing the sum of the squared errors between the predicted values and the actual values. Therefore, the straight line that is fitted through the set of data points aims to minimize the distance between the predicted values and the actual values. Whereby, the corresponding loss function is given as follows:

$$\mathcal{L}(\beta) = \frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{j=1}^{T}(\sigma_{i,t+1} - g^*(z_{i,t};\beta))^2, \tag{2}$$

The OLS loss function ($\mathcal{L}(\beta)$) is obtained by minimizing the sum of the squared errors, which consist of the difference between the realized volatility ($\sigma_{i,t+1}$) and an estimated linear function ($g^*(z_{i,t};\beta)$). Hereby, the estimated linear function given in equation 2, can be written as:

$$g^*(x_{i,t};\beta) = x'_{i,t}\beta, \tag{3}$$

where $g^*(x_{i,t};\beta)$ can be seen as a linear transformation of the covariates ($x'_{i,t}$) by a coefficient ($\beta$). Note that the most important aspects of the linear model are the accuracy of the predictions and the interpretability of the model. However, OLS is known to perform poorly in predictive tasks that involve complex nonlinear data due to its linear decision boundary. Nevertheless, the greatest benefit of using OLS is its simplicity in terms of calculation and interpretability.

### 3.2 Generalised Linear Model: $2^{nd}$ Order Polynomial

The generalized linear model (GLM) is an advanced version of linear regression that can accommodate non-normal error structures and non-constant variance. In linear regression, it is assumed that the response variable follows a normal distribution, which may not always hold true. Furthermore, GLMs are more adaptable and can handle a wider range of data types (Gill (2001)). When it comes to predicting option volatility, GLMs can incorporate non-linearities, which can possibly capture the underlying relationships between option prices and their determinants, such as underlying asset returns, moneyness, and time to maturity. GLMs can also handle heteroskedasticity, observed in financial data when the response variable's variance changes as a function of the predictors.

I initialize the GLM as a $2^{nd}$ order polynomial, whereby the flexible function in equation 1 takes the following form:

$$g^*(x_{i,t};\beta) = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \ldots + \beta_i x_{i,t} + \beta_{i+1} x_{1,t}^2 + \beta_{i+2} x_{2,t}^2 + \ldots + \beta_{i+m} x_{i,t}^2 + \epsilon, \tag{4}$$

where $x_{1,t}$, $x_{2,t}$, ..., $x_{i,t}$ represent the predictors, and $x_{1,t}^2$, $x_{2,t}^2$, ..., $x_{i,t}^2$ represent the squared terms of the predictors. The coefficients $\beta_0$, $\beta_1$, $\beta_2$, ..., $\beta_i$ correspond to the linear terms, while $\beta_{i+1}$, $\beta_{i+2}$, ..., $\beta_{i+m}$ represent the coefficients for the squared terms.

Furthermore, the loss function for the GLM ($\mathcal{L}(\beta)$) remains the same as the OLS loss function given in section 3.1, where the goal is to minimize the sum of squared errors. Moreover, introducing a second-order polynomial allows us to establish whether the response variable is predicted to vary non-linearly. I expect this method to produce a significant improvement with respect to linear models as the implied volatility given as a function of moneyness is highly non-linear according to previous literature.[14]

### 3.3 Regularized Linear Model: Elastic Net

In the context of options, regularization methods can be used to promote sparsity in option pricing models while also ensuring that the resulting options prices are smooth and continuous with respect to the underlying

[14] See e.g. Derman and Kani (1994) and Wilmott et al. (1995) for an explanation of the nonlinearity in option IV. Specifically, the authors show that the IV as a function of strike price (which is proportional to moneyness) displays a "skew" or "smile" attribute which can not be captured by linear models.

asset price. One way to achieve this is to use a form of regularization that allows for a trade-off between sparsity and allowing for some form of interaction between the predictors. Zou and Hastie (2005) impose an elastic net and compare this with the lasso regularization. The authors show that lasso is often outperformed by elastic net while benefiting from an analogous sparsity of representation.[15] Furthermore, the elastic net enjoys the benefit of being able to deal with multicollinearity such as the ridge Regression. Thus the elastic net regularized regression can handle high-dimensional data with potential multicollinearity issues while also promoting sparse solutions. Therefore, it is ideal to make use of an elastic net regression when it is believed that the predictors may be correlated. The importance of utilizing this model is to attempt to enhance the predictive power of the linear models in volatility forecasting for equity options. Moreover, this model still gives valuable insights into the predictive power of a regularized linear model for the prediction of implied volatility. Finally, the mathematical notation of the elastic net model consists of the loss function, which is given by the following equation:

$$\mathcal{L}(\beta;\cdot) = \mathcal{L}(\beta) + \phi(\beta;\cdot), \tag{5}$$

where the loss function is composed of the Ordinary Least Squares (OLS) loss function ($\mathcal{L}(\beta)$) given in section 3.1, and added to a penalty function ($\phi(\beta;\cdot)$). The corresponding penalty function is given as follows:

$$\phi(\alpha;\theta,\beta) = \theta(1-\beta)\sum_{i=1}^{K}|\alpha_i| + \frac{\theta\beta}{2}\sum_{i=1}^{K}\alpha_i^2. \tag{6}$$

where the penalty function consists of two penalty terms which are also known as lasso and ridge respectively. Lasso imposes sparsity in the model by forcing some of the coefficients to be precisely zero, while the ridge term shrinks coefficients simultaneously towards zero which is beneficial among highly multicollinear variables. Therefore, the elastic net benefits from the ability to deal with multicollinearity through the ridge term and benefit from the variable selection process of the lasso term.

[15]The elastic net regression originates from a combination of ridge regression and lasso regression. Moreover, this paper does not go into detail about the origin of elastic net regression. For a more elaborate explanation including the geometry of the elastic net algorithm, I refer you to Zou and Hastie (2005).

## 3.4 Tree Based Models: Random Forests & Gradient Boosting

### 3.4.1 Decision Tree Basics

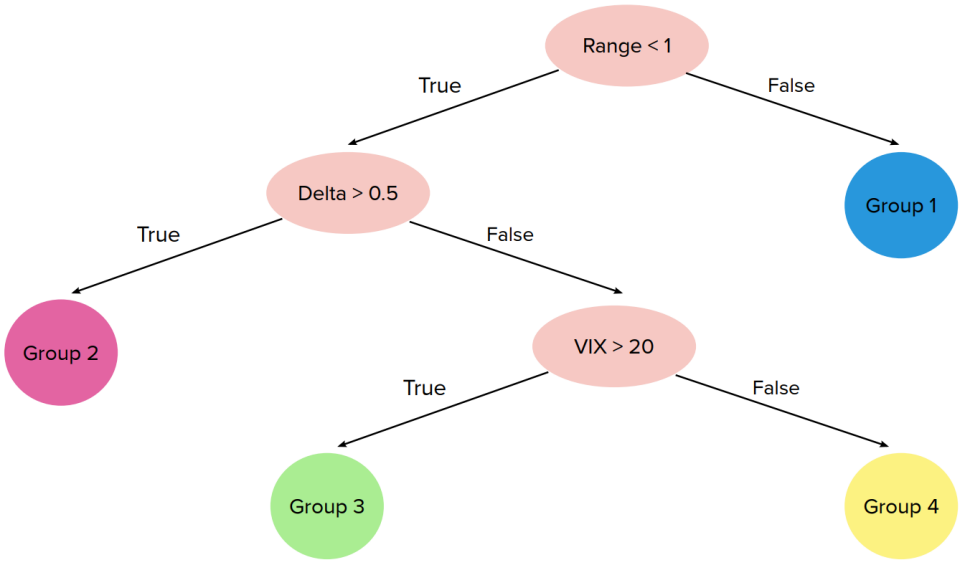
Decision trees are a fully nonparametric model that can incorporate interactions in the predictors through multivariate functions of predictors. Thereby, the basic principle of a decision tree is to identify groups of data that behave in a similar manner with respect to each predictor. A tree is formed by selecting a characteristic variable that separates the data regarding its behavior with respect to this characteristic. Therefore, this characteristic acts as a threshold or a decision boundary, whereby the data is segregated into two buckets depending on whether the current response variable identifies with this characteristic in a true or false way. Subsequently, at each new step, the data is then sorted into new bins based on a new characteristic chosen, whether it be at random or not. This way of sequential sorting allows for the predictors to split the data into rectangular partitions. Furthermore, I define the prediction of a tree, with depth (L) and (K) terminal nodes as the mean value of the response variable within each partition (Gu et al. (2020)):

$$g\left(x_{i,t};\theta,K,L\right)=\sum_{k=1}^{K}\theta_k\mathbf{1}_{\{x_{i,t}\in C_k(L)\}}, \tag{7}$$

where one of the (K) partitions of the data is equivalent to $C_k(L)$. Furthermore, it is evident from equation 7 that each partition (k) with corresponding constant $\theta_k$, is a product of a maximum of L indicator functions of the covariates. Similarly to Gu et al. (2020), I focus on binary recursive trees due to their relative simplicity.

Figure 3 depicts an example of the decision tree process for the regression task in this research, based on three covariates, "Range", "Delta", and "VIX". We may notice that the observations are first sorted based on the range of the option price, with the option prices that portray a range greater than one, ending up in the first bin, and the options that have a smaller range than 1, end up in a new "node", where the observations are subsequently sorted on a new predictor. In the next node, the options with a smaller range than 1 are sorted based on the delta of the underlying option, where the remaining observations now either end up in group two or are sorted once more through the third covariate (VIX). Moreover, the final goal of the decision tree is to minimize the prediction error, and thus to find groups that discriminate the observations in the best possible way. In machine learning jargon, the loss connected with the prediction error of a branch C is

Figure 3: Example of the Regression Tree Using Three Characteristics(Range, Delta, and VIX) to Make the Decisions on the Predictions




Note: This figure depicts the diagram of a regression tree in the space of three characteristics (Range of the option price, Delta, and VIX). The sample of the equity options is divided into four groups based on the characteristics. These four groups can be seen as the colored circles in the figure, which are also known as terminal nodes or leaf nodes.

often referred to as "impurity." For this research, similarly to Gu et al. (2020), I opt for the most common impurity measure for each branch of the tree, also known as the "l2" impurity, which is defined as:

$$H(\theta, C) = \frac{1}{|C|} \sum_{x_{i,t} \in C} \left(\sigma_{i,t+1} - \theta\right)^2, \tag{8}$$

whereby |C| is the number of observations in the set C. Furthermore, to minimize the prediction error is therefore to find the branch C that minimizes the function $H(\theta, C)$. This non-parametric approach to predicting the regression outcomes allows for a great deal of flexibility within the model which is convenient for accommodating non-linearities and interactions among predictors. Ambiguously, the model flexibility is nevertheless the impediment to the predictive performance, due to this leaving the model susceptible to overfitting. Therefore, according to Gu et al. (2020), the decision tree process must be heavily regularized. I follow the same procedures as in Gu et al. (2020) to mitigate the effects of overfitting by introducing two ensemble tree regularizers, which aggregate the prediction of multiple trees into one prediction.[16]

[16]Note that these are not the only procedures to mitigate the effect of overfitting. However, due to the scalability of ensemble methods to large datasets and more reliable performance, it makes for the optimal regularization method for tree-based models (Gu et al. (2020)).

### 3.4.2 Random Forests

Random forests are a type of machine learning algorithm that is an ensemble learning method based on the construction of multiple decision trees.[17] Hereby, the idea is to build a forest of trees, where each tree is built by selecting a random subset of features and a random subset of data. Furthermore, the random subsets are independently chosen for each tree in the forest, which helps to include variability and prevent overfitting. Subsequently, when the random subsets are chosen, the decision trees are grown through a recursive binary splitting process as explained in section 3.4.1, where the current tree is split at each node based on the feature that best separates the data into two groups according to a specified splitting criterion. Ultimately, this process is repeated until some stopping criterion is met, such as a minimum number of observations per leaf[18] or a maximum depth of the tree, which include, but are not limited to the parameters that need to be selected a priori. I refer you to Appendix A for a full definition of all the parameters that need to be determined in advance.

Moreover, once all the decision trees are grown, the final prediction for each observation is constructed by aggregating the prediction of all the decision trees within the forest. For classification purposes, this can be done by taking the majority vote among all the predictions, while for regression one can take the mean of all the prediction outcomes.

### 3.4.3 Gradient Boosting

Gradient Boosting is another tree-based ensemble learning method that aims to improve the performance of decision trees.[19] Unlike random forests which build multiple trees independently and combine their results by averaging, gradient boosting builds trees sequentially where each new tree is fitted to the residuals of the previous tree.

The idea behind gradient boosting is to iteratively fit weak learners such as shallow decision trees to the residuals of the previous trees. The predicted values from each new tree are then added to the predictions

[17] Random forests are originally described by Breiman (2001). This algorithm is a variation of a general procedure known as "bagging."

[18] The minimum number of observations per leaf is often referred to as minimum samples (per) leaf in random forests algorithms

[19] One popular implementation of gradient boosting is the Extreme Gradient Boosting (XGBoost) algorithm (Chen and Guestrin (2016)), which I will implement as my boosting algorithm. This form of boosting is widely used in industry and research due to its efficiency and effectiveness (Zhang et al. (2022), Christensen et al. (2021), and Chen et al. (2021)).

of the previous trees, thereby reducing the residuals in each iteration. In this manner, gradient boosting gradually improves the predictions by iteratively focusing on the areas where the previous trees performed poorly. Furthermore, gradient boosting stops when a predefined number of iterations is attained or the loss function reaches a certain threshold.

Similar to random forests, gradient boosting can handle both regression and classification problems and can be used with different loss functions. In this research, I use gradient boosting with the mean squared error loss function, which is appropriate for regression problems.

One potential issue with gradient boosting is overfitting, especially when the number of iterations is large. To avoid overfitting, several regularization techniques can be used, such as early stopping, learning rate reduction, and tree depth limitation.[20]

Overall, gradient boosting is a powerful and flexible method that can achieve high predictive performance, especially when combined with other advanced techniques such as feature engineering and hyperparameter tuning.

### 3.5 Support Vector Regression

Support vector machines (SVMs) are a popular class of machine learning algorithms that are widely employed for regression and classification (Bishop (2006), Cortes and Vapnik (1995), and Hastie et al. (2009)). SVMs are based on the idea of finding a hyperplane that separates data points in a high-dimensional feature space. Thereby, the goal is to find a hyperplane that maximizes the margin between the two classes, where the margin is defined as the distance between the hyperplane and the closest data points of each class. Analogously, Support vector regression (SVR) is a variant of SVM that is used for regression purposes. SVR follows a similar concept as SVM, but instead of locating a hyperplane that divides the data, its goal is to identify a hyperplane that crosses the center of the data. Thereby, the main objective is to locate a hyperplane that has the greatest margin on both sides, with a predetermined margin width. A great benefit of SVR includes its ability to generalize well, with a high prediction accuracy (Awad and Khanna (2015)). Furthermore, the authors state that SVR's computational complexity is independent of the dimensionality of the input space.

[20] In this research I have implemented regularization, tree depth limitation, and learning rate reduction to mitigate the effects of overfitting. See Appendix A for more details regarding the parameters utilized in the algorithm

Moreover, Bettencourt et al. (2023)[21] find that SVR outperforms both linear and machine learning-based models in an environment of limited target variables. For these reasons, I propose to make use of SVR for the prediction of implied volatilities of individual equities. In this section, I discuss the theoretical foundations of SVR and the practical aspects of implementing SVR for regression purposes.

I begin by assuming the conditional return of the volatility as a flexible function of a kernel ($K(x_i, x)$), that maps the vector of covariates ($x_i$) into a higher dimensional space. The corresponding flexible function ($g^*(\cdot)$) is given by:

$$g^*(\cdot) = \sum_{i \in SV} (\alpha_i - \alpha_i^*) K(x_i, x_j) + \psi. \tag{9}$$

Subsequently, the loss function of SVR following Smola and Schölkopf (2004)[22] is defines as:

$$\mathcal{L}(\cdot) = \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} (\epsilon_i + \epsilon_i'), \tag{10}$$

where $||w||^2$ is the squared norm of the weight vector $w$, $C$ is a hyperparameter that controls the trade-off between the margin and the errors, and $\epsilon_i$ and $\epsilon_i'$ are the slack variables that represent the deviation of each training instance from the decision boundary. Moreover, the minimization problem is subjected to the following constraints:

$$y_i - w^T \Phi(x_i) - b \leq \epsilon + \epsilon_i, \tag{11}$$

$$w^T \Phi(x_i) + b - y_i \leq \epsilon + \epsilon_i', \tag{12}$$

$$\epsilon_i, \epsilon_i' \geq 0, \tag{13}$$

where $y_i$ is the target value of the $i^{th}$ training data point, $\Phi(x_i)$ represents the covariates of the $i^{th}$ example, and $b$ is the bias term.

The goal is to find the values of $w$, $b$, and the slack variables $\epsilon_i$ and $\epsilon_i'$ that minimize the objective func-

[21] Although the authors focus on forecasting returns, they attribute the findings to non-linearities and interaction effects which are also expected to play a role in the nonlinear dynamics of the volatility smile.

[22] Initially the foundation for SVR was laid out by Drucker et al. (1996), whereby Smola and Schölkopf (2004) extended and refined the algorithm.

tion ($\mathcal{L}(\cdot)$) while satisfying the constraints. To achieve this, various optimization techniques like quadratic programming or gradient descent can be employed. My algorithm will follow scikit-learn's implementation (Pedregosa et al. (2011)), which makes use of the Sequential Minimal Optimization (SMO) algorithm to solve the dual problem. SMO is a specialized algorithm for solving the quadratic programming problem that arises in the dual problem of SVR minimization.

Furthermore, the dual problem that follows from the SVR formulation consists of introducing Lagrange multipliers for each of the inequality constraints in the primal problem. The resulting dual problem can therefore be expressed as:

$$\max: \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j (y_i - y_j)^2, \tag{14}$$

$$\text{subject to:} \quad 0 \leq \alpha_i, \alpha_i' \leq C, \tag{15}$$

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i') = 0, \tag{16}$$

where $\alpha_i$ and $\alpha_i'$ are the Lagrange multipliers corresponding to the two inequality constraints in the primal problem, and $C$ corresponds to the upper bound on the multipliers. The dual problem includes maximizing a concave quadratic function which is subject to linear equality and inequality constraints, which are given by equation 11. Subsequently, the optimal values of $\alpha$ can be used to compute the weight vector $w$ and the bias term $b$ in terms of the training examples and their corresponding Lagrange multipliers.

In order to capture the non-linear relationships between the covariates and the target variables, I propose to make use of the Radial Basis Function (RBF) kernel. The radial basis function evaluates the likeness of two instances by measuring the Euclidean distance between them in the characteristic space, leaving no relevant details behind. Due to its known flexibility and effectiveness, the RBF is a well-known choice for handling non-linear problems(Hsu et al. (2010) and Schölkopf and Smola (2002)). To utilize the RBF kernel in the SVR algorithm, the inner product of the feature vectors in the primal problem is replaced with the kernel function. This modification allows for the following expression of the dual problem with the RBF

kernel:

$$\max: \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j (y_i - y_j) K(x_i, x_j), \tag{17}$$

$$\text{subject to:} \quad 0 \leq \alpha_i, \alpha_i' \leq C, \tag{18}$$

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i') = 0, \tag{19}$$

where $K(x_i, x_j)$ is the kernel function. By utilizing the kernel function, it becomes possible to calculate the dot product in the feature space of a higher dimension without having to explicitly map the data to that space. This helps avoid the computational expenses that come with high-dimensional spaces. Therefore, the kernel trick can efficiently compute the dot product when using the RBF kernel.

### 3.6 Neural Networks

The ultimate machine learning algorithm I propose is the feedforward neural network. Neural networks are a type of machine learning algorithm that can be used for both regression and classification tasks. They are inspired by the structure of the human brain, with layers of interconnected "neurons"[23] that process and transmit information.[24] Furthermore, one of the most common types of neural networks is the feedforward neural network (Bishop (1995), Goodfellow et al. (2016), and Nielsen (2015)), which consists of an input layer, one or more hidden layers, and an output layer. Each layer contains a set of neurons that receive input from the previous layer and output to the next layer.[25] Therefore, the information is always fed forward and does not include any loops in the network, which makes it easier for understanding the models. However, if we were to increase the number of layers, enabling the modeling to weave many layers of nonlinear interactions, the model complexity would increase accordingly and thus making the model less transparent.[26] Returning to

[23]Note that in machine learning jargon, "neurons" are referred to as "nodes" in modern terms, while it originally was referred to as "perceptrons" in the late 1950s and early 1960s (Rosenblatt (1958) and Joseph (1961)). However, for simplicity and to maintain the focus on the econometric understanding, I maintain the use of the word neurons in the aspiration of improving the reader's understanding of the neural network process.

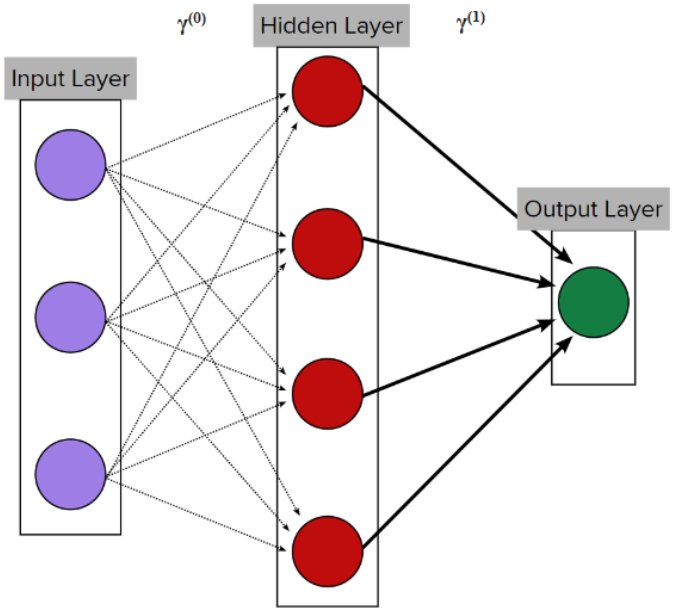
[24]The origin of neural networks dates back to the beginning of the 19th century (Legendre (1805) and Gauss (1809). Initially, these neural networks were practically alternatives for linear-regression methods.

[25]As far as I am aware, the first deep learning (DL) models of the feedforward multi-layer perceptron type date back to the 1960s (Ivakhnenko and Lapa (1965), Ivakhnenko et al. (1967), and Ivakhnenko (1968)), whereas the single-layer perceptron was introduced by Rosenblatt (1958), with the goal of eventually understanding the capability of higher organisms for perceptual recognition, thought process, and more.

[26]The phenomenon where we add many layers is known in the machine learning community as "deep learning." In this research, we will stick to "shallow learning" similar to Gu et al. (2020), due to the high predictive accuracy and more importantly the better interpretability than their deep learning counterparts.

the analogy of neurons in a human brain, I explain the fundamentals behind the feedforward neural network with the help of Figure 4.

Figure 4: Feedforward Neural Networks Diagram With Three Predictors As Input and Five Nodes in the Single Hidden Layer




Note: This figure depicts the diagram of a neural network with one hidden layer. The purple circles represent the neurons of the input layer, the dark red circles represent the hidden layer, and the dark green circle denotes the output layer.

From this figure, it is evident that the layers of the networks delineate groups of neurons whereby every layer can transmit signals to neurons in subsequent layers. Furthermore, the number of neurons in the input layer is equivalent to the dimension of the covariate set, which is three in Figure 4 ($x_1$, $x_2$, and $x_3$). Additionally, each neuron in the input layer sends information to the neurons of the hidden layer in a linear manner according to a parameter vector $\gamma^i$, where the superscript $i$ denotes the layer starting from 0. Thereafter, each neuron in the following layer(s) (in the case of Figure 4, it is one hidden layer) applies a nonlinear activation function $f$ to the corresponding signal it has received. Finally, the results from the hidden layer are linearly aggregated for each individual neuron similar to the information transformation from the input layer to the hidden layer, such that the output (prediction) is obtained. The aggregation procedure with the corresponding prediction according to the flexible functional form of the example in Figure 4 is given by the

following equation:

$$g^*(x;\gamma) = \gamma_0^{(1)} + \sum_{i=1}^{5} x_i^{(1)} \gamma_i^{(1)}, \tag{20}$$

where $g^*(x;\gamma)$ is the output prediction, $\gamma_0^{(1)}$ the intercept of the amplification vector $\gamma$, and $\sum_{i=1}^{5} x_i^{(1)} \gamma_i^{(1)}$ the aggregated weighted signals from the hidden layer to the output.

For the prediction of IV among the individual equity options, I consider architectures of two and three hidden layers. Furthermore, I ensure that all the architectures are connected such that every neuron has a connection with a subsequent neuron.[27] Gu et al. (2020) and Bianchi et al. (2020) have shown that in high dimensional spaces where they predict asset risk premia and bond risk premia respectively in the cross-section, two and three layers are the most prominent neural networks. Ultimately, I have tested the predictive accuracy of 1 and 4 hidden layers to verify this statement in the context of predicting implied volatility. See Appendix A for the complete overview of all the neural networks and their corresponding parameters to be hypertuned. An additional choice that needs to be made for the hidden layers regards the number of neurons to incorporate. Nevertheless, it is not realistic to search through all the possibilities (Stathakis (2009)). The issue is not that the number of parameters is very large but rather that the necessary time to evaluate each combination is tedious. It is also known that ceteris paribus, different results are often produced due to alternative initialization conditions (Yao (1993)). To stabilize the neural networks results, I propose to use the approach of Bianchi et al. (2020), where the authors implement an ensemble of various neural networks. This procedure ensures that the variance of the output of the neural networks decreases. Nevertheless, this also implies multiple neural networks need to be collectively tuned and hence the computational time per model increases.

Next, it is imperative to establish which nonlinear activation function $f$ I employ in the hidden layers mentioned earlier in this section. It is also possible to include the activation function as a hyperparameter to optimize. However, this implies that the procedure of tuning the neural networks will demand a longer time. Therefore, I anticipate which two activation functions can be of great benefit for this research and tune the

[27] In the case that the architecture is not fully connected, that would imply that the neuron with no subsequent connection will not add value to the predicted output. The relevant predictors have already been selected and therefore, it is not necessary to exclude certain neurons from the final prediction.

models according to the best predictive performance. Note that I apply the same activation function across all the hidden layers for the sake of model simplicity. Among the various[28] activation functions available, I make use of one of the most well-known activation functions in the literature, namely: Rectified linear unit (ReLU) (Gu et al. (2020) Agarap (2018), and Nair and Hinton (2010)). The corresponding activation function is defined as follows:

$$f(x) = \max(0, x). \tag{21}$$

Evidently, the ReLU function promotes sparsity by ensuring that the signal can not be negative but at the minimum 0. Nevertheless, nonlinearity, nonconvexity, and strong parametrization often present in neural networks tend to make it difficult to optimize the objective function and thus making it computationally intensive (Gu et al. (2020)). A solution for this is to utilize stochastic gradient descent (SGD) to train the neural network. Instead of computing the gradient of the objective function using the entire training dataset, SGD computes the gradient using a small randomly selected subset of the data, known as a mini-batch. This approach is computationally more efficient, as it reduces the amount of data that needs to be processed at each iteration. SGD works by iteratively updating the parameters in the direction of the negative gradient of the loss function. The corresponding size of the update is controlled by the learning rate, which determines how much of a step is taken in each direction. By taking small steps in the direction of the gradient, SGD slowly converges towards a minimum of the loss function, where the parameters are optimal for the given dataset. While SGD is effective in optimizing the parameters of a neural network, it can still suffer from issues such as getting stuck in local minima or saddle points. In order to address these issues, I implement an adaptive learning rate and a maximum amount of iterations. The adaptive learning rate keeps the learning rate constant to the initial learning rate that is implemented as long as the training loss remains decreasing. Every time two epochs[29] fail to decrease the training loss consecutively by at least some tolerance level ($\tau$), or if two epochs do not succeed at increasing the validation score by $\tau$, the learning rate will be decreased by

[28] See for example Sharma et al. (2017), where the authors describe the different activation functions.

[29] An epoch implies training the neural network where all the data is used exactly once. For example, in a neural network that uses 10 epochs and contains 1000 data points, assuming that one batch consists of the entire dataset, the model will use the 1000 data points 10 times. It is often confused with iterations, whereas iterations are the number of batches required to train an entire epoch. For example, a batch containing 500 data points will require 2 iterations per epoch.

a factor of five. Furthermore, the maximum amount of iterations ensures that a solution is obtained either when the stochastic gradient descent algorithm has converged or the maximum number of iterations has been achieved.[30]

## 3.7 Evaluation Metrics

To evaluate the performance of the different models, I will make use of an out-of-sample R-squared measure ($R^2_{os}$) (Gu et al. (2020)) and root mean squared error (RMSE) (Almeida et al. (2022)). First, I describe the process pertaining to the $R^2_{os}$. Thereafter, I describe the process regarding the RMSE. Finally, I describe the Diebold and Mariano (1995) (DM) test to inspect the significance of prediction differences between models. In contrast to Gu et al. (2020), I will take the well-known $R^2_{os}$, which is equivalent to

$$R^2_{os} = 1 - \frac{\sum_{t=1}^{T} (\sigma_t - \widehat{\sigma}_t)^2}{\sum_{t=1}^{T} (\sigma_t - \bar{\sigma}_t)^2}, \tag{22}$$

where my ($R^2_{os}$) accounts for the historical means, as these are of importance when looking at volatilities, due to them being significantly different from zero.

Furthermore, the second evaluation metric is based on Almeida et al. (2022), where the authors use an RMSE as a measure. This is a very common prediction accuracy measure among measuring option volatility predictions, while even more so in econometric literature. The benefit of utilizing both measures allows for comparison across different models and different metrics which are more elaborate than a single measure. The RMSE of a predictor ($y$) is calculated by taking the square root of the mean squared error (MSE) of this predictor, where the MSE represents the expectation of the squared difference between the forecasted values (y) and observed values ($\hat{y}$). The corresponding mathematical equation is given as follows:

$$\text{RMSE}(\hat{y}) = \sqrt{\text{MSE}(\hat{y})} = \sqrt{\text{E}\left((\hat{y} - y)^2\right)}. \tag{23}$$

Moreover, I utilize the DM test, to identify whether the predictions made from a certain model are significantly different from another model. Additionally, I ensure that the error dependence is accounted

[30] It is important to note that the maximum iterations determine the number of epochs and not the number of gradient steps.

for by using Newey-West standard errors (Newey and West (1987)). Nevertheless, this idea stems from Gu et al. (2020), where the authors justify the use of a heteroscedastic and autocorrelation consistent covariance matrix due to potential cross-sectional return error correlation. However, in the case of equity volatilities, we can also expect a correlation among the error terms due to the underlying asset returns being correlated. Furthermore, the test statistic between two models (a) and (b) is thus given by the following equation:

$$DM_{ab} = \frac{\bar{\mu}_{ab}}{\hat{\sigma}_{\bar{\mu}_{ab}}}, \tag{24}$$

where the mean and Newey-West standard error are respectively given as $\bar{\mu}_{ab}$ and $\hat{\sigma}_{\bar{\mu}_{ab}}$. Furthermore, in the following equation I define the Diebold Mariano test based on a single time series, stripped from the cross-section similarly to Gu et al. (2020)[31].

$$\mu_{ab,t+1} = \frac{1}{n_{3,t+1}} \sum_{i=1}^{n_{3,t+1}} \left( \left( \widehat{e}_{i,t+1}^{(1)} \right)^2 - \left( \widehat{e}_{i,t+1}^{(2)} \right)^2 \right), \tag{25}$$

where $n_{3,t+1}$ is the number of equity options in the test sample and $\widehat{e}_{i,t}^{(k)}$ is the prediction error corrected for heteroskedasticity and autocorrelation of model k on the implied volatility of security option i.

### 3.8 Feature Importance

Complex machine learning models are discouraged by some investors and academics who criticize the ability to understand the economic intuition behind the decision-making process of the models. Therefore, recent studies have tackled the problem of unlocking the black box of some machine learning models (Daul et al. (2022) and Li et al. (2020)). I will pursue the same approach as the authors to understand the driving importance of the covariate set. Thereby, I will use a decrease in the training sample's ($R^2$) as the evaluating measure. Moreover, I will analyze the relationship between each of the most important predictors and the corresponding importance in terms of predicting the implied volatilities.

[31] The authors state the single time series is more inclined to hold onto asymptotic normality, which allows for the test to be applied to inference in the significance of the forecasts.

# 4 Case Study: The Cross-Section of Individual Equity Options

## 4.1 Machine Learning Model Results

Table 4 presents the results of the models in terms of implied volatility RMSE (IV RMSE), including the benchmark model, Random Walk (RW). I compare ten models, including OLS, Enet, SVR, GLM, RF, XGB, NN(2), NN(3), NN Ensemble, and the RW. Moreover, in Appendix B I report the results in terms of $R^2_{oos}$ due to the performance corresponding exactly with the results of the IV RMSE, and therefore can be interpreted analogously.[32]

Due to the nature of the volatility smile present in the implied volatility as discussed earlier, I expect the nonlinear models to outperform the linear models as well as the Random Walk, as a linear model would not be able to capture such an effect. The columns of the nonlinear models all exhibit a lower RMSE of the implied volatility compared to the linear regression (OLS) and Random Walk benchmark, confirming the nature of the volatility surface dynamics being highly nonlinear. Moreover, the best results are given by the NN models, where the flexible function can approximate the implied volatility most accurately. In contrast to Gu et al. (2020), where the authors have found that characteristics are fundamentally noisy signals and partially redundant, I find that all the characteristics play a role synchronously.[33] One reason for the difference being the prediction task is aimed at implied volatilities for my research whereas Gu et al. (2020) predicts the equity risk premia. Thus, the implied volatility estimate might be described by more complex interactions between the covariates. Nevertheless, this does not imply that all the covariates are highly informative and never redundant. We will see in the following section more specifically how the relative importance of each predictor plays a role in the overall prediction of the cross-section.

The OLS model which is the first benchmark model produced an RMSE of 5.67, amounting to the second least accurate prediction in Table 4, whereas the RW model produced an RMSE of 6.03. The volatility smile dynamics that are well known among the option literature imply that a linear fit to the volatility surface will not lead to an optimal prediction, which is confirmed by the inferior prediction accuracy.

The reduction in the RMSE from 5.67 to 3.03 implies that the application of the GLM significantly

[32] I have tested various sizes of reduced dimensions to include, as well as select hand-picked predictors and other predictors that were not reported such as Fed Funds Rate and change in Fed Funds Rate. Nevertheless, this has always led to a deterioration in performance. Therefore, for clarity, I have only reported the results of the prominent models and predictors

[33] Note that the number of characteristics considered in Gu et al. (2020) is significantly higher than in my research.

improves the predictability of the model compared to the OLS model. This improvement indicates that incorporating a polynomial expansion in the linear model, without including interactions, yields valuable insights. Moreover, the results of the Diebold-Mariano test as presented in Table 5, support the statistical superiority of the generalized linear method over the other linear models. The statistical significance observed in the test further reinforces the credibility of the findings. The inclusion of a polynomial expansion in the linear model offers an important implication. It allows for capturing and modeling non-linear relationships between variables as mentioned in section 3.2.

Moreover, the SVR model substantially improves the RMSE of the benchmark model, with an RMSE of 2.96. One of the reasons why this model does well is that it can manage non-i.i.d data and model non-linear relationships. We have seen from section 2, that the data is autocorrelated and thus the implied volatility is not an i.i.d. random variable. Furthermore, the model is prone to noisy data (See e.g. Sabzekar and Hasheminejad (2021)), which is the limiting factor when trying to fit a hyperplane[34] that separates the data the best.

The elastic net model gives similar results to the OLS model where the RMSE is given as 5.46 in Table 4. However, the improvement is not statistically significant as can be seen from the DM test results in Table 5. The similarity between the results of the EN model and the OLS model shows that performing a regularized regression does not enhance the predictive performance. This implies that the penalization of the predictors by shrinking them jointly towards 0 to promote sparsity and selecting relevant predictors does not add much value over performing a regression on the full predictor set.

The tree-based models also significantly improve the results of the benchmark model. With random forests producing an RMSE of 4.27 and XGB producing an RMSE of 2.83, including interaction effects greatly improve the predictive ability of the implied volatility. In contrast to other econometric literature[35], I find that random forests produce better results with deeper trees (on average 16-36 layers) rather than shallow trees. One reason possibly is the signal-to-noise ratio being much higher among IV prediction tasks as compared to return prediction, thus enabling deeper trees to more accurately predict the IV without

[34] In a three-dimensional problem, SVR sets a hyperplane that spans the two dimensions of the problem which best separates the data. In higher dimensions it is not possible to visualize in what manner the "hyperplane" will look like that separates the data

[35] See e.g. Gu et al. (2020), where the authors show that shallow trees of one to five layers on average produce better results.

overfitting. Moreover, the gradient-boosted trees are superior in terms of predictive results and include an ensemble of many 'weak' learners which are shallow trees with a maximum depth of 4 layers. Despite the fact that Random Forest and XGBoost are similar, in the sense that they are both decision tree-based algorithms, the performance of the two models varies significantly. This result can be attributed to several factors, whereby one being the fact that Boosting algorithms possibly fit the data more closely than Bagging algorithms for certain datasets and vice-versa, as econometric literature show ambiguous results pertaining to gradient boosting and random forest algorithms.

NN produces optimal results for 2 and 3 hidden layers which are consistent with the literature[36]. Respectively the RMSE is given as 2.03 and 2.11. The results in Table 4 do not include other architectures with 1-5 layers to keep the report concise. Once again, a model which incorporates interactions among the predictors statistically outperforms the benchmark according to Table 5. Furthermore, the results demonstrate significant predictive power in terms of monthly implied volatility when using shallow learning.[37] However, the NN models exhibit a notable level of variance in their predictive outcomes. This indicates that despite maintaining consistent model parameters, the predictions generated by the NN models tend to differ significantly. Therefore, the results are not as reliable as the ensemble model where I combine the 5 best NN models with 2 and 3 hidden layers. Despite the predictive accuracy reported being higher than the 2 and 3 hidden layer models, with an RMSE of 1.78, the ensemble model is not always the most accurate.[38]

Table 4: Monthly Out-of-sample Predictive Performance of All Models in Terms of IV RMSE

| | OLS | SVR | GLM | Enet | RF | XGB | NN(2) | NN(3) | Ens | RW |
|---|---|---|---|---|---|---|---|---|---|---|
| 16 features‡ | 5.67 | 2.96 | 5.46 | 5.45 | 4.27 | 2.83 | 2.03 | 2.11 | 1.78 | 6.03 |

Note: The table delineates the performance of all the models in terms of implied volatility root mean squared error (IV RMSE) for a one-step-ahead prediction. Furthermore, it is evident that the random walk (RW) benchmark model is outperformed by all the other models.
‡ The 16 features are the stock- and option-specific predictors described in section 2

[36]See e.g. Almeida et al. (2022) and Gu et al. (2020)
[37]In recent literature deep learning has been a prominent topic when it comes to predicting the implied volatility surface (See e.g. Bloch and Böök (2020) and Medvedev and Wang (2022)).
[38]I have purposely not chosen to report the best 2 and 3 hidden layer results as this would be unrealistic to expect these to consistently outperform the ensemble model.

Table 5: Diebold-Mariano (1995) test statistics for the outperformance of one model with respect to another.

| | **OLS** | **GLM** | **EN** | **SVR** | **RF** | **XGB** | **NN(2)** | **NN(3)** | **Ens** |
|---|---|---|---|---|---|---|---|---|---|
| **OLS** | x | 6.83 | **-3.68** | 29.22 | 15.64 | 37.25 | 42.71 | 41.35 | 43.69 |
| **GLM** | | x | **-5.12** | 23.36 | 12.72 | 26.18 | 28.16 | 27.93 | 29.05 |
| **EN** | | | x | 26.47 | 18.76 | 36.54 | 37.61 | 36.70 | 38.51 |
| **SVR** | | | | x | **-12.78** | 6.51 | 19.35 | 17.84 | 22.07 |
| **RF** | | | | | x | 28.10 | 21.97 | 21.16 | 23.08 |
| **XGB** | | | | | | x | 10.68 | 9.42 | 13.02 |
| **NN(2)** | | | | | | | x | **-3.46** | 17.2 |
| **NN(3)** | | | | | | | | x | 14.15 |
| **Ens** | | | | | | | | | x |

Note: The negative bold numbers imply that the model in the row outperforms the model in the column and a positive number implies that the column outperforms the row. Furthermore, the lower triangular elements are kept empty due to symmetry.

## 4.2 Relative Predictor Importance

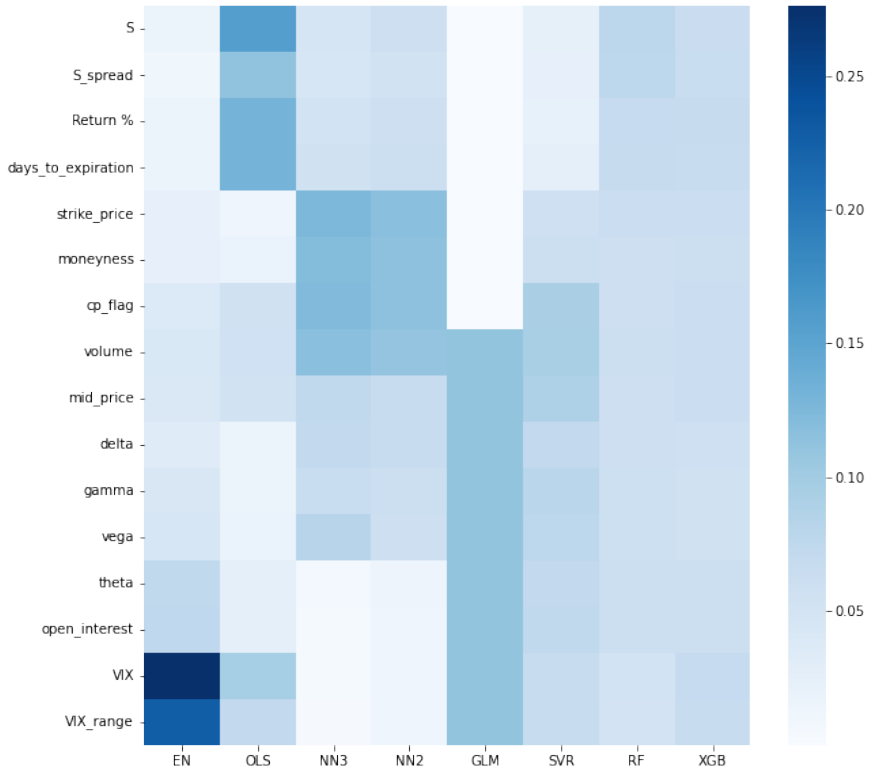
Figure 6 depicts the relative predictor importance for all the models. Note that the ensemble model is a combination of NN2 and NN3 and therefore, will resemble similar variable importance as the two aforementioned models. Furthermore, I have scaled the decreases in the training sample $R^2$ to sum up to one such that the comparison of the relevance among different models are comparable. Moreover, Figure 5 displays the relative predictor importance in the form of ranks that are visualized through a heatmap. The ranks are merely the score that corresponds to the relative predictor importance given in Figure 6 visualized in a more reader-friendly manner. Thus the purpose of this heatmap is to visualize and compare the importance in a 'smoother' manner. Therefore, in this section, I will focus on Figure 5, whereas Figure 6 can be used to pinpoint more detailed comparisons for the interested reader. The first and most interesting characteristic of Figure 5 which is noticeable is that there is no predictor that consistently is the most significant across all models. Thereby, we see that the hierarchical models such as RF and XGB display a uniform pattern such that the relative importance of all the variables is more or less identical. This ambiguous performance of predictors across different models also supports the argument for the need for the full predictor set made in the previous section and the multicollinearity present within the variables. However, it is also evident from the Figure that the linear models (OLS and EN) favor the importance of the 'VIX' and 'VIX range.' This can be attributed to the linear effect present in predicting the implied volatility of separate options and the market's expectations for the relative strength of near-term price changes in the S&P500 (VIX). It is possible that the

nonlinear models such as neural networks, SVR, and tree-based models make better use of the non-linearities and interactions present within the predictors, where the VIX is more of a 'linear' indicator in that sense and therefore does not add much value to the non-linear models. Additionally, an interesting observation is that of the GLM model. From Figures 5 it is evident that GLM identifies roughly half the predictors as important, where more specifically all these variables are considered equally important. We can see that the GLM combines the use of the more 'linear' predictors which are identified in the elastic net model, while also benefiting from new features such as the greeks[39], open interest, and volume. This interesting combination may be attributed to the nature of our $2^{nd}$ order polynomial allowing for linear effects with the predictors as well as quadratic terms, which together allow for the GLM model to identify this unique set of predictors as equally important.

Furthermore, the dynamics where most of the non-linear models arise from are quite similar in the sense that they make use of most of the information available by incorporating a similar importance ranking among all the predictors. Overall, there is no select amount of features among all the models that are the most important when predicting the IV. The implications for this research entail that the machine learning models are susceptible to which predictors are selected and in what way they are utilized such that the models differ in the manner that information is utilized from the predictors in order to make an estimate of the IV.

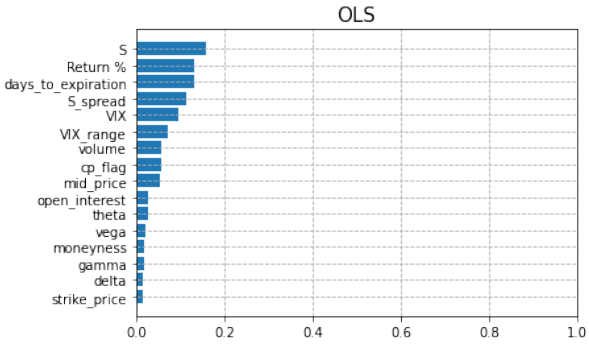
[39]The greeks in option theory are quantities that measure the sensitivity of an option price with respect to several factors. The greeks used in this study are $\Delta, Vega, \Gamma, and \Theta$.

Figure 5: Feature Ranks Across All Models, Where the Ranks Correspond to the Variable Importance Scores Calculated in Figure 6

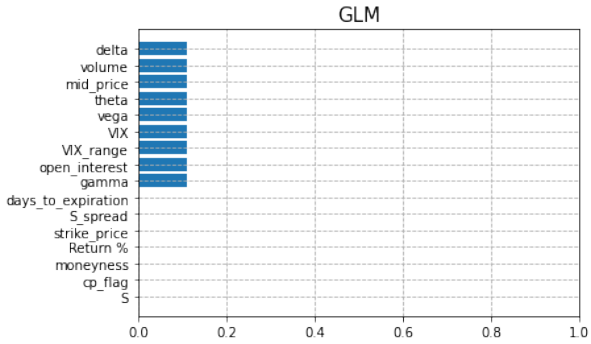



Note: The relative ranking per feature per model is the strongest when the colors are dark(er) and the weakest when the colors are light(er).

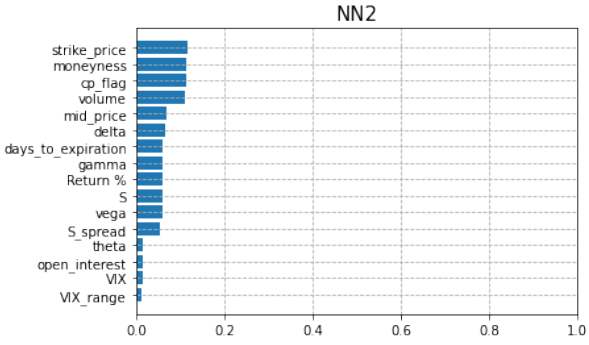
Figure 6: Variable Importance Measure for All the Different Models



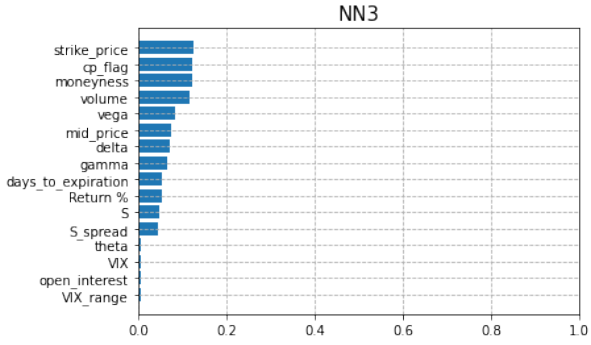

(a) Variable Importance: OLS



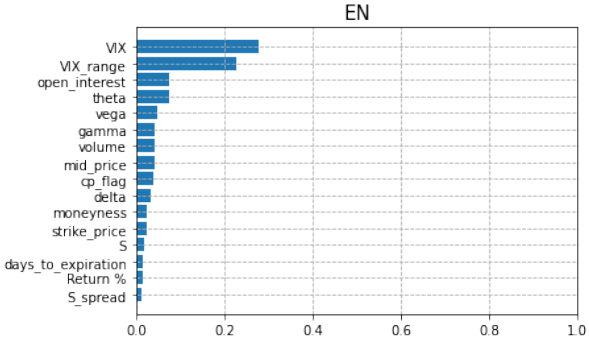

(b) Variable Importance: GLM



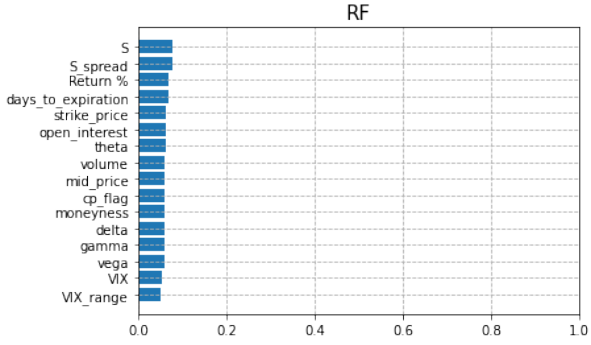

(c) Variable Importance: NN2



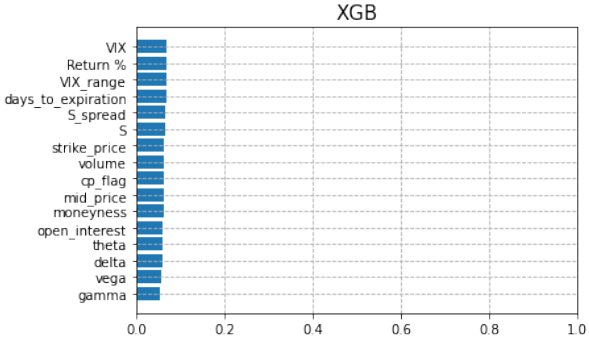

(d) Variable Importance: NN3



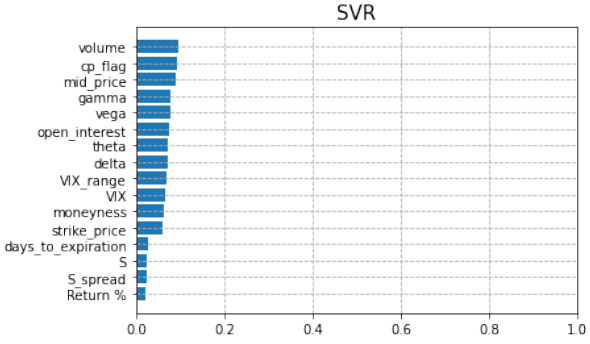

(e) Variable Importance: EN




(f) Variable Importance: RF




(g) Variable Importance: XGB




(h) Variable Importance: SVR

# 5 Conclusion

In my research, I carried out an extensive analysis of various machine learning models designed for predicting implied volatility (IV) in options markets. By taking into account a myriad of stock- and option-specific indicators, my objective was to pinpoint those models which showcased exceptional predictive capabilities and thus could offer valuable knowledge for those working in finance or carrying out research.

The outcomes of this study underscored the comparative performance levels of different models by measuring them against root mean square error and out-of-sample R-squared metrics. Among all the models appraised, it was found that the ensemble model (Ens) consistently surpassed the others proving its adeptness at capturing intricate IV dynamics. This insinuates that within neural network modeling processes, amalgamating multiple model types can pave the way toward more precise and dependable projections - as ensemble modeling harnesses individual strengths while curbing their weaknesses.

Moreover, two other noteworthy performers were SVR and XGB - these too emerged as strong contenders in the predictive modeling arena indicating they are capable to interpret nonlinear associations and complex patterns within data sets effectively. The findings from this study emphasize NN's potential including SVR and XGB as formidable instruments when dealing with implied volatility predictions.

Contrarily, linear methodologies like ordinary least squares (OLS) and elastic net (Enet) displayed comparatively high RMSE values and lower $R^2_{\text{oos}}$ values when juxtaposed with the ensemble and neural network models. Despite their simplicity and ease of interpretation, these linear models might find it daunting to accurately portray the complex dynamics of IV, which typically showcases non-linearities and multicollinearity among predictors.

To further evaluate one model's superiority over another, I undertook the Diebold-Mariano (DM) test. The outcomes consistently leaned in favor of the ensemble model along with its foundational neural network models, offering empirical proof for its dominance over other competing models. Interestingly, positive DM test statistics were also evident in GLM, SVR, and XGB models underlining their comparable performance against other methods.

Although my primary concentration was on forecasting efficacy, the significance of understanding predictor importance cannot be overlooked as it provides crucial knowledge about the fundamental factors influencing

IV. My research has demonstrated that no specific group of predictors is always deemed more significant across all models. The intricate non-linear and hierarchical models depict a balanced utilization of predictors, considering them to be mostly equally important. On the other hand, straightforward linear models facilitate a simpler interpretation of predictor importance within these frameworks, at the cost of an inferior prediction.

Conclusively, the implications of my findings allow practitioners and researchers to create more precise volatility surface predictions by applying complex machine-learning methods, as well as understanding the underlying drivers of these models. Furthermore, this allows them to make more informed decisions on which machine learning models are suitable for lower timeframe implied volatility predictions. Finally, being able to accurately predict implied volatility can aid with devising option trading strategies and improving option pricing models. Nevertheless, further research is necessary to improve current option pricing models with machine learning.

# References


Ackerer, D., Tagasovska, N., and Vatter, T. (2020). Deep smoothing of the implied volatility surface. *arXiv preprint arXiv:2002.04838*.

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Almeida, C., Fan, J., Freire, G., and Tang, F. (2022). Can a machine correct option pricing models? *Working Paper*.

Awad, M. and Khanna, R. (2015). *Support Vector Regression*, pages 67–80. Apress, Berkeley, CA.

Aït-Sahalia, Y., Li, C., and Li, C. X. (2021). Implied stochastic volatility models. *Review of Financial Studies*, 34(1):394–450.

Bernales, A. and Guidolin, M. (2014). Can we forecast the implied volatility surface dynamics of equity options? predictability and economic value tests. *Journal of Banking and Finance*, 46:326–342.

Bettencourt, L., Hsieh, S., Knol, L., and Rathi, S. (2023). Improving the prediction of cross-sectional government bond returns through machine learning. Unpublished manuscript.

Bianchi, D., Büchner, M., and Tamoni, A. (2020). Bond Risk Premiums with Machine Learning. *The Review of Financial Studies*, 34(2):1046–1089.

Bishop, C. M. (1995). Neural networks for pattern recognition. *Oxford university press*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Bloch, D. A. and Böök, A. (2020). Predicting future implied volatility surface using tdbp-learning.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Chen, S. and Zhang, Z. (2019). Forecasting implied volatility smile surface via deep learning and attention mechanism.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 785–794.

Chen, W., Zhang, H., Mehlawat, M. K., and Jia, L. (2021). Meanvariance portfolio optimization using machine learning-based stock price prediction. *Applied Soft Computing*, 100:106943.

Christensen, K., Siggaard, M., and Veliyev, B. (2021). A machine learning approach to volatility forecasting. *Available at SSRN*.

Christoffersen, P., Fournier, M., and Jacobs, K. (2018). The factor structure in equity options. *Journal of Financial Economics*, 129(2):277–298.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Daul, S., Jaisson, T., and Nagy, A. (2022). Performance attribution of machine learning methods for stock returns prediction. *The Journal of Finance and Data Science*, 8:86–104.

Derman, E. and Kani, I. (1994). Riding on a smile. *Risk*, 7(2):32–39.

Diebold, F. X. and Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business  Economic Statistics*, 13(3):253–263.

Drucker, H., Burges, C. J., and Kaufman, L. (1996). Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161.

Gauss, C. F. (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Typis Dieterichianis.

Gill, J. (2001). *Generalized Linear Models: A Unified Approach*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5):2223–2273.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer, 2 edition.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2010). A practical guide to support vector classification. *Department of Computer Science, National Taiwan University, Taipei 10617, Taiwan*.

Ivakhnenko, A. G. (1968). The group method of data handlinga rival of the method of stochastic approximation. *Soviet Automatic Control*, 13(3):43–55.

Ivakhnenko, A. G. and Lapa, V. (1965). *Cybernetic predicting devices*. CCM Information Corporation.

Ivakhnenko, A. G., Lapa, V., and McDonough, R. (1967). *Cybernetics and forecasting techniques*. American Elsevier.

Joseph, R. D. (1961). *Contributions to perceptron theory*. PhD thesis, Cornell University.

Legendre, A.-M. (1805). *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot.

Li, Y., Turkington, D., and Yazdani, A. (2020). Beyond the black box: An intuitive approach to investment prediction with machine learning. *Journal of Portfolio Management*, 46(7):61–74.

Liu, D., Liang, Y., Zhang, L., Lung, P., and Ullah, R. (2021). Implied volatility forecast and option trading strategy. *International Review of Economics  Finance*, 71:943–954.

Medvedev, N. and Wang, Z. (2022). Multistep forecast of the implied volatility surface using deep learning. *Journal of Futures Markets*, 42:645–667.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *ICML*.

Newey, W. K. and West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3):703–708.

Nielsen, M. A. (2015). *Neural networks and deep learning: A textbook*. Determination Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Poon, S.-H. and Granger, C. W. (2003). Forecasting volatility in financial markets: A review. *Journal of Economic Literature*, 41(2):478–539.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Sabzekar, M. and Hasheminejad, S. M. H. (2021). Robust regression using support vector regressions. *Chaos, Solitons Fractals*, 144:110738.

Schölkopf, B. and Smola, A. J. (2002). *Support Vector Machines and Kernel Algorithms*. Cambridge University Press.

Sharma, S., Sharma, S., and Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Science*, 6(12):310–316.

Shover, L. (2012). *Calendar Spreads: Trading Theta and Vega*, pages 217–225.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Stathakis, D. (2009). How many hidden layers and nodes? *International Journal of Remote Sensing*, 30(8):2133–2147.

Wilmott, P., Howison, S., and Dewynne, J. (1995). *The Mathematics of Financial Derivatives: A Student Introduction*. Cambridge University Press.

Yao, X. (1993). Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4:203–222.

Zeng, Y. and Klabjan, D. (2019). Online adaptive machine learning based algorithm for implied volatility surface modeling. *Knowledge-Based Systems*, 163:376–391.

Zhang, C., Zhang, Y., Cucuringu, M., and Qian, Z. (2022). Volatility forecasting with machine learning and intraday commonality. *Available at SSRN*.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

# Appendix

## Appendix A: Hyperparameters of Machine Learning Models

The following table gives the details pertaining to the specific hyperparameters utilized for tuning the various models. Furthermore, the tuning is done according to the sci-kit learn implementation as mentioned in the report. Moreover, the models that require no hypertuning, have not been mentioned in this table.

Table 6: Hyperparameters for tuning the ML models

| SVR | RF | XGB | NN(2) | NN(3) |
|---|---|---|---|---|
| C $\in (10^{-3}, 10)$ | max features : 'sqrt' | max depth : $2 \sim 5$ | hidden layers size : ** | hidden layers size : * |
| gamma : 'scale' | n_estimators $\in (100, 300)$ | n_estimators $\in (100, 300)$ | solver : 'sgd', 'adam' | solver : 'sgd', 'adam' |
| epsilon : (1, 0.5, 0.4, 0.35) | min samples per leaf: | learning rate : | learning rate init : | learning rate init : |
| kernel : 'rbf' | $\in (1, 4)$ | (0.45; 0.4; 0.3) | $\in (0.001; 0.01)$ | $\in (0.001; 0.01)$ |
| | max depth: | $\alpha : (0.1; 0.5; 0.7; 1)$ | learning rate : | learning rate : |
| | (2, 3, 4, 9, 16, 25, 36) | | {constant, adaptive} | {constant, adaptive} |
| | | | $\alpha \in (0, 0.1)$ | $\alpha \in (0, 0.1)$ |
| | | | max iterations : 1000 | max iterations : 1000 |
| | | | activation : ReLu | activation : ReLu |

Note: The table describes the hyperparameters that are utilized to fine-tune each corresponding model.
** Hidden layers size (NN(2)) : (100,100), (100,50), (50,50), (8, 4), (50, 25), (9, 3)
* Hidden layers size (NN(3)) : (100, 50, 50), (100, 50, 25), (100, 100, 50), (100, 100, 100), (8, 4, 2), (16, 4, 2)

## Appendix B: Model Results Extension

The following Table delineates the performance of the various models in terms of $R^2_{oos}$. The results can be interpreted analogously with the results reported in Table 4 in section 4.1. Therefore, for further elaboration of the interpretability of the models I refer you to section 4.1.

Table 7: Monthly Out-of-sample Predictive Performance on the cross-section of options ($R^2_{\text{oos}}$ )

| | OLS | SVR | GLM | Enet | RF | XGB | NN(2) | NN(3) | Ens |
|---|---|---|---|---|---|---|---|---|---|
| 16 features[‡] | 0.57 | 0.87 | 0.63 | 0.56 | 0.75 | 0.86 | 0.92 | 0.93 | 0.95 |
| 16 features + $FF$ + $ChangeFF(\%)$ | 0 | -0.09 | 0 | 0.02 | -0.69 | -0.78 | 0.15 | 0.21 | 0.18 |

Note: The table reports the predictive performance across different sets of features. With the 16 stock- and option-specific features as a baseline. Where the

[‡] The 16 features are the stock- and option-specific predictors described in section 2