Improving Risk Management and Analysis of Structured Notes through Path Dependence, Greeks, and Machine Learning

Mohammed AHNOUCH[1,2], Lotfi ELAACHAK[2], Erwan LESAOUT[1], and Abderrahim GHADI[2]

[1]Université Paris 1 Panthéon Sorbonne, Paris, France
[2]C3S,FSTT, Tangier, Morocco

January 15, 2025

Abstract

This study provides a detailed look at structured financial products, specifically focusing on their price sensitivities and hedging strategies, with an emphasis on factors like path dependence and volatility. The analysis shows that these products are affected by the sequence of market movements, which influences their value. In terms of risk management, the study examines Vega hedging and the profiles of Vanna and Volga, using past market events such as the Uridashi and KOSPI-linked autocallable crises to support these findings. Additionally, the use of hybrid machine learning models, combining Multi-Layer Perceptrons (MLP) and gradient-boosted trees, significantly improves predictive accuracy compared to using individual models. These hybrid models also show strong generalization capabilities, making them more effective for financial modeling. Overall, the results enhance the understanding of sensitivities and hedging strategies for structured notes, while demonstrating the potential of machine learning to improve their valuation and risk management.

Keywords: Structured Financial Products, Risk Management, Sensitivity Analysis, Machine Learning Models, Financial Derivatives

1 Introduction

1.1 Background

The evolution of structured products, particularly autocallable and yield enhancement notes, has significantly reshaped investment landscapes, offering tailored market exposure and enhanced yield in low-rate environments [Hull, 2018]. Autocallables, with their early redemption features contingent on underlying asset performance, embody a complex interaction of barrier options and contingent coupon payments, requiring sophisticated pricing models that account for path dependency [Glasserman, 2013]. While early work focused on adapting barrier option theory [Rubinstein, 1987], more recent research emphasizes the need for advanced simulation techniques and efficient numerical methods to capture the complex dynamics of these products [Benhamou, 2010]. The impact of these products on market dynamics, particularly through

hedging activities, has also garnered attention, with studies exploring their potential to amplify market movements [Cont and Tankov, 2005]. Yield enhancement notes, often structured as short put options combined with coupons, offer a distinct risk-return profile, transferring downside risk for enhanced yield [Černý, 2009]. The increasing prevalence of these instruments necessitates rigorous analytical frameworks addressing both pricing and systemic risk implications.

1.2 Contribution

While literature specifically addressing autocorrelation's impact on autocallable options is scarce, established principles of path dependency and related research on barrier options and stochastic volatility models highlight its relevance [Fouque et al., 2003, Bates, 1996]. Studies on GARCH models further underscore the influence of autocorrelation in asset returns and volatility on derivative pricing [Bollerslev, 1986]. The present study rigorously investigates the isolated effect of autocorrelation on autocallable prices and Greeks, including second-order Greeks like Vanna and Volga, aiming for a balance between mathematical and machine learning rigor and practical applicability for practitioners. Explicitly modeling autocorrelation, particularly within a semi-analytical framework (e.g., relying on multivariate Gaussian CDF integration), offers a computationally efficient alternative to Monte Carlo simulation for pricing path-dependent derivatives. This approach defines a mapping $f : \mathbb{R}^n \to \mathbb{R}$ from observables and model parameters (including the autocorrelation coefficient) to the derivative price, representing a highly non-linear function.

Even with this semi-analytical simplification, the function f remains highly complex and non-linear. This complexity justifies the recourse to machine learning (ML) and deep learning (DL) techniques to learn this map. ML/DL models can approximate complex non-linear functions with high accuracy, effectively learning the relationship between the input features and the derivative price without requiring explicit knowledge of the underlying analytical form of f.

Assessing the complexity of a vectorial function f, crucial for accurate approximation, can be achieved through functional analysis tools like Sobolev spaces, which quantify a function's smoothness and regularity via its derivatives [Adams and Fournier, 2003]. High variability or discontinuities in these derivatives (low regularity) render traditional numerical methods less effective, motivating the use of machine learning (ML), particularly for complex financial derivatives pricing. As highlighted by statistical learning theory, a model's ability to generalize from finite data is linked to the complexity of its function class [Vapnik, 1998]; thus, complex functions, common in finance, necessitate high-capacity ML models like deep neural networks (DNNs). Theoretical advancements, including analyses of DNN approximation capabilities in high dimensions [Poggio et al., 2017] and the Universal Approximation Theorem [Hornik et al., 1989], support DNNs' potential to mitigate the curse of dimensionality. This is especially relevant when dealing with analytically intractable functions, typical of complex financial instruments, where ML/DL offers an efficient mapping from market data and model parameters to derivative prices, circumventing computationally intensive simulations.

2 Literature Review

Our approach is twofold. Firstly, we examine recent advancements in the study of autocallable products, focusing on pricing methodologies, risk management techniques, and empirical analyses. This review establishes the current state-of-the-art in traditional autocallable modeling and highlights the need for new approaches to address existing limitations, such as the inability

to capture certain market dynamics. Secondly, recognizing the inherent non-linearity and complexity of autocallable pricing functions, we explore the potential of machine learning (ML) and deep learning (DL) techniques to approximate these functions. Specifically, we investigate the use of Multi-Layer Perceptrons (MLPs), XGBoost, CatBoost, and hybrid ensemble learning methods combining MLPs and CatBoost. This exploration is motivated by the universal approximation theorem, which states that neural networks with a single hidden layer can approximate any continuous function to arbitrary accuracy, provided sufficient neurons are available [Hornik et al., 1989]. Given that the pricing function of autocallables maps a set of observable market variables and model parameters to a scalar price, this theorem suggests the suitability of neural networks for this task. Moreover, because our dataset is inherently tabular, we also review recent developments in machine learning specifically tailored for tabular data, including gradient boosting methods like XGBoost and CatBoost, which have shown remarkable performance in various applications [Grinsztajn et al., 2022]. The combination of these two strands of literature—traditional autocallable modeling and ML for tabular data—provides the foundation for our proposed model, which aims to leverage the strengths of both approaches to achieve more accurate and efficient pricing of these complex financial instruments. This approach is further justified by the increasing use of ML in finance as a whole as seen in [Dixon et al., 2020].

2.1 Autocallable notes' pricing and risk management

Autocallable structured products are complex financial instruments that require advanced methodologies for pricing and hedging. Cui et al. [2024] propose a Markov chain approximation framework for pricing and hedging autocallable products, offering a computationally efficient solution that adapts to market dynamics. This is complemented by the work of Kim and Yoon [2019a], who incorporate stochastic volatility models, providing a robust approach to handle the nuances of market uncertainty. These contributions are further supported by Fries [2011], who explore a Monte Carlo-based analytic pricing scheme that effectively manages complex payoff structures, broadening the computational strategies available to practitioners.

The empirical analysis of autocallables sheds light on their performance and risk-return characteristics in real markets. Deng et al. [2015] analyze the risk-return profiles of these products, emphasizing their practical implications for investors. Similarly, Albuquerque et al. [2015] examine the trade-offs between risk and return, offering a strategic perspective for portfolio construction. Real-world insights are further enhanced by Klotzle and Pinto [2012], who provide a detailed case study on autocallables, demonstrating how these instruments behave under specific market conditions.

In the realm of hedging, significant advancements are made by Sharma and Nadkarni [2024], who introduce a distributional reinforcement learning framework for managing portfolios containing autocallables. This innovative approach leverages machine learning to address the challenges of dynamic risk management. Paletta and Tunaru [2022] contribute a Bayesian perspective, offering statistical techniques that incorporate uncertainty into pricing and hedging strategies. These works are complemented by Carver [2018], who explore the implications of market dislocations on model calibration, underscoring the importance of adaptability in risk management.

Computational efficiency is a critical concern in pricing and hedging autocallables. Zeron et al. [2023] address the computational challenges posed by regulatory frameworks such as FRTB-IMA, providing insights into the implementation of equity autocallable models. Huang and Wang [2019] propose a simple yet effective numerical method for pricing discretely monitored options, offering solutions that are both practical and scalable. These contributions ensure that pricing and hedging methodologies remain feasible in real-time trading environments.

Finally, studies on the performance and replication of autocallables offer practical insights into their implementation. Lee et al. [2012] provide a comprehensive analysis of the pricing and performance of these products, highlighting their behavior across different market scenarios. Kim and Yoon [2019b] focus on static replication methods, presenting efficient strategies for replicating the payoffs of autocallables. Together, these articles form a cohesive body of knowledge, bridging theoretical advancements with practical applications to address the multifaceted challenges of pricing and hedging autocallable products.

2.2 Implications of Autocorrelation

Autocorrelation serves as the simplest form of path dependency, encapsulating how the entire trajectory of an underlying asset impacts financial derivative instruments. In essence, if an asset's past returns influence its future returns, the current derivative price depends not only on the current asset price but also on the path it has taken to reach that price. This concept is crucial for accurate derivative pricing and risk management, as models that disregard such dependencies can lead to significant mispricings and inadequate hedging strategies. For instance, the use of affine diffusion processes, as detailed by Cheridito et al. [2005], provides a general framework for modeling asset dynamics where factors such as stochastic volatility or interest rates can introduce dependencies in the asset's returns, effectively capturing a form of autocorrelation. Similarly, jump-diffusion models, as reviewed by Aït-Sahalia [2007], incorporate discontinuous jumps in asset prices, which can induce dependencies in subsequent returns. If a large jump occurs, it can alter the likelihood of future jumps or volatility, creating a form of path dependency. The use of Ornstein-Uhlenbeck (OU) processes and their generalizations, as discussed by Barndorff-Nielsen and Shephard [2001] and rigorously analyzed by Brockwell [2001], provides a powerful tool for modeling autocorrelated factors that influence asset prices. Because the OU process has memory, it directly models autocorrelated variables. Finally, the application of more complex Lévy processes, such as the generalized hyperbolic Lévy motions explored by Eberlein [2001], offers another avenue for capturing path dependency. These processes, with their ability to model skewness and heavy tails often associated with autocorrelated returns, provide a more nuanced description of asset dynamics and their impact on derivative pricing. Therefore, addressing autocorrelation in the underlying asset's dynamics is essential for accurate derivative valuation and effective risk management.

To the best of our knwoledge, there is not an explicit study directly isolating the impact of autocorrelation on autocallable pricing, (as most studies embed it within more complex models), its influence, though, can be inferred from research on related topics. For example, studies on stochastic volatility models, such as those discussed by Fouque et al. [2003], often incorporate mean-reverting volatility processes. This mean reversion can be seen as a form of autocorrelation in volatility, which, in turn, impacts the distribution of the underlying asset's returns and thus the pricing of path-dependent options like autocallables.

As a further point, research on discrete-time models like GARCH, as introduced by Bollerslev [1986], demonstrates how autocorrelation in returns can lead to volatility clustering. This clustering effect can significantly affect the probability of hitting barriers in autocallables, especially when the observation dates are close together. Models that fail to account for this clustering can underestimate the probability of early redemption or barrier breaches, leading to mispricing of the autocallable.

2.3 Role of Machine Learning

Traditional Monte Carlo (MC) methods, while theoretically appealing with their $\mathcal{O}(N^{-1/2})$ convergence rate, falter in practice as the dimensionality d of the underlying asset space increases.

This is particularly pertinent for autocallable notes, which often depend on multiple underlying assets and various barrier conditions that must be monitored over numerous observation dates. The curse of dimensionality manifests through exponentially growing computational costs and sample sparsity, making accurate MC simulations computationally prohibitive [Györfi et al., 2002]. However, recent advancements in Machine Learning (ML) and Deep Learning (DL) offer innovative strategies to mitigate these challenges. Techniques such as dimensionality reduction via Principal Component Analysis (PCA) [Jolliffe, 2002], neural network-based function approximation [Hornik et al., 1989], and generative models like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) [Goodfellow et al., 2014, Kingma and Welling, 2013] can effectively capture the intricate dependencies and reduce the effective dimensionality of the problem. By integrating these ML/DL methodologies, it becomes feasible to perform more efficient and accurate Monte Carlo simulations for pricing autocallable notes, thereby enhancing both computational efficiency and model precision.

As well, the theoretical underpinnings of ML/DL, such as universal approximation theorems [Hornik et al., 1989] and low-rank tensor decompositions [Oseledets, 2011], provide a robust foundation for modeling the complex payoff structures and conditional expectations inherent in autocallable notes. Adaptive sampling techniques informed by ML/DL models can identify and focus computational resources on the most significant regions of the integration domain [Hinton and Salakhutdinov, 2006], thereby reducing variance and improving convergence rates. Additionally, surrogate models trained through supervised learning can emulate expensive payoff evaluations, significantly decreasing the overall computational burden [Hinton and Salakhutdinov, 2006]. These surrogate models enable real-time pricing and risk management by providing rapid approximations of high-dimensional integrals required for autocallable notes. Moreover, advanced variance reduction techniques, enhanced by learned importance sampling distributions, concentrate sampling efforts on high-probability regions of the underlying asset space [Goodfellow et al., 2014], further mitigating the curse of dimensionality. Collectively, these ML/DL approaches not only address the traditional limitations imposed by high dimensionality in Monte Carlo methods but also pave the way for more sophisticated and scalable financial engineering solutions in the realm of autocallable notes.

2.3.1 Is recourse to Black Scholes pertinent?

The Mixtures of Black-Scholes Representation Theorem demonstrates that any arbitrage-free option price under a stochastic volatility model can be expressed as a weighted integral of Black-Scholes prices, as given in Equation (1) [Lorig et al., 2013, Fouque et al., 2011]:

$$V(t, S_t) = \int_0^\infty BS(t, S_t; \sqrt{v})\mu_t(dv), \tag{1}$$

where μ_t represents a measure over instantaneous volatilities, and $BS(t, S_t; \sqrt{v})$ is the Black-Scholes price with volatility $\sqrt{v}$. This result underscores the foundational nature of Black-Scholes pricing, establishing it as the basis for constructing any arbitrage-free pricing model. Consequently, training neural networks on Black-Scholes features not only aids in accurate option pricing under the Black-Scholes model but also inherently equips the network to generalize across broader, more complex stochastic volatility frameworks. This pertinence of learning maps between features and prices even under the Black-Scholes model stems from the fact that the Black-Scholes formula, while analytical, still represents a non-linear relationship between inputs (spot price, strike, time to maturity, volatility, interest rate) and the option price. By learning this mapping, ML models can effectively interpolate and extrapolate, providing fast and accurate pricing even for complex or exotic options whose analytical solutions are not readily available.

2.3.2 Broader Implications and Efficiency

Training neural networks using Black-Scholes features eliminates the need to approximate the full volatility process during training. This reduces computational complexity and avoids potential overfitting to spurious patterns in simulated stochastic paths. Empirical studies have demonstrated that models trained on Black-Scholes-derived features generalize well across a variety of volatility regimes, as they are rooted in universal principles of arbitrage-free pricing [Cont and da Fonseca, 2010, Andreasen and Huge, 2015]. Consequently, the pertinence of learning the functional mapping between Black-Scholes features and prices lies in its ability to capture the essential structures of financial option pricing without requiring exhaustive model-specific training. This is particularly useful in situations where calibration of complex models is computationally intensive or when real-time pricing is required. Learning the Black Scholes map also allows to use the model as a control variate in monte carlo simulations as it learns the smooth part of the function and the monte carlo only has to learn the residual. This also justifies the use of ML even when the underlying process is known.

2.3.3 Performant ML methods for tabular data

Several Machine Learning (ML) and Deep Learning (DL) methods are employed for handling tabular data, each with its strengths and weaknesses. However, tree-based methods, particularly Gradient Boosting Machines (GBMs) such as XGBoost [Chen and Guestrin, 2016a], LightGBM [Ke et al., 2017], and CatBoost [Prokhorenkova et al., 2018], have consistently demonstrated exceptional performance on tabular datasets across various domains. This superiority stems from several key advantages. Tree-based models can naturally handle mixed data types, a common characteristic of tabular data, which often comprises both numerical and categorical features. Unlike methods like linear regression or neural networks, which can require extensive preprocessing such as one-hot encoding for categorical variables, potentially leading to high-dimensional and sparse feature spaces, tree-based models handle these different data types without such transformations. CatBoost, in particular, excels at handling categorical features directly using ordered target statistics, mitigating target leakage issues [Dorogush et al., 2018]. In addition to this, tree-based models exhibit robustness to outliers and feature scaling. Their splits are based on feature order rather than absolute values, reducing the need for extensive data cleaning and preprocessing compared to methods sensitive to outliers like linear regression or neural networks. The ability to capture complex non-linear relationships between features and the target variable through recursive partitioning of the feature space is another key advantage of tree-based models, making them well-suited for the intricate non-linear relationships often found in financial data. Moreover, tree-based models provide measures of feature importance, offering valuable insights into the underlying data generating process. This interpretability is a significant advantage over "black-box" models like deep neural networks, especially in regulated domains like finance, where model explainability is often a requirement. Finally, while training can be computationally intensive for large datasets, especially for deep trees or large ensembles, prediction with tree-based models is very fast, making them suitable for real-time applications where quick predictions are essential. While Deep Neural Networks (DNNs) can also achieve good performance on tabular data, they often require careful architecture design, extensive hyperparameter tuning, and large amounts of data to generalize well. What's more, DNNs can be more prone to overfitting and require techniques like dropout [Srivastava et al., 2014] and batch normalization [Ioffe and Szegedy, 2015] to mitigate this issue. Recent studies have shown that carefully tuned tree-based models often outperform DNNs on tabular data, especially when the dataset size is limited [Grinsztajn et al., 2022, Shwartz-Ziv and Armon, 2021, Kadra et al., 2021]. This relative advantage of tree-based methods for tabular data makes them highly pertinent for pricing autocallables,

where the relationship between input features and prices is complex and non-linear, and where interpretability and computational efficiency are important considerations.

3 Methodology

3.1 Modified Black-Scholes Model

3.1.1 Model Setup

For observation times $\{t_1, \ldots, t_N\}$, the modified Black-Scholes model incorporates an AR(1) process to capture temporal dependence in volatility:

$$S_{t_i} = \exp\left(\left(r - \frac{\sigma^2}{2}\right) t_i + \sigma\sqrt{t_i} X_{t_i}\right) \tag{2}$$

where X_{t_i} follows an AR(1) process:

$$X_{t_i} = \rho X_{t_{i-1}} + \sqrt{1-\rho^2}\epsilon_i, \quad X_{t_0}, \epsilon_i \sim \mathcal{N}(0,1) \text{ iid} \tag{3}$$

3.1.2 Survival Probabilities

The survival probability up to time t_N is given by:

$$\mathbb{P}(\text{survival up to } t_N) = \mathbb{P}(S_{t_j} < B_a, \forall j \leq N) = \Phi_N(\mathbf{k}; \Sigma)$$

where:

- $\mathbf{k} = [k(t_1), \ldots, k(t_N)]^\top$ with $k(t_i) = \frac{\ln(B_a/S_0) - (r - \frac{\sigma^2}{2})t_i}{\sigma\sqrt{t_i}}$
- $\Sigma_{ij} = \rho^{|i-j|}$ (AR(1) covariance matrix)

3.1.3 Temporal Correlation Comparison

We further analyse the temporal correlation structures of the AR(1) and Brownian models, illustrating their decay behaviours and implications for autocallable pricing.

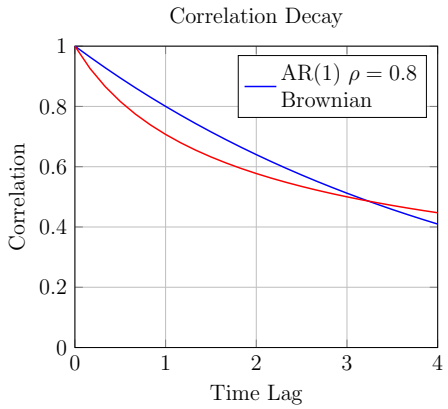


Figure 1: Correlation Decay Comparison between AR(1) and Brownian Models

3.1.4 Computational Methods

Evaluating the high-dimensional multivariate normal cumulative distribution function (CDF) presents a significant computational challenge. This study employs Gaussian quadrature for dimensions $d \leq 10$, leveraging its high accuracy ($\sim 10^{-8}$) and fast convergence, particularly with adaptive schemes [Golub and Welsch, 1969]. For higher dimensions ($d > 10$), quasi-Monte Carlo methods using Sobol sequences are utilized, offering superior uniform coverage and convergence compared to standard Monte Carlo approaches [Niederreiter, 1992].

3.1.5 Numerical Integration Framework

The pricing requires the computation of high-dimensional normal cumulative distribution functions.

The multivariate normal CDF can be expressed as:

$$\Phi_n(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{u}-\boldsymbol{\mu})\right) d\mathbf{u} \tag{4}$$

The Gauss-Hermite quadrature leverages the relationship between this integral and the Hermite polynomials. The n-th Hermite polynomial is defined by:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}(e^{-x^2}) \tag{5}$$

This orthogonality property of Hermite polynomials leads to the quadrature formula:

$$\int_{-\infty}^{\infty} f(x)e^{-x^2}dx \approx \sum_{i=1}^{m} w_i f(x_i) \tag{6}$$

where $\{x_i\}_{i=1}^m$ are the roots of $H_m(x)$ and the weights are given by:

$$w_i = \frac{2^{m-1}m!\sqrt{\pi}}{m^2[H_{m-1}(x_i)]^2} \tag{7}$$

3.1.6 Greek Computation

The computation of Greeks, which are sensitivities of the option price to various parameters, is essential for effective hedging strategies. These sensitivities, such as Delta (sensitivity to the underlying asset price), Vega (sensitivity to volatility), Vanna (sensitivity of Delta to volatility), and Volga (sensitivity of Vega to volatility), provide crucial information for managing risk and constructing hedging portfolios. Traditional methods for computing Greeks, such as finite difference approximations, can be computationally expensive and prone to numerical instability, especially for complex derivatives or when higher-order Greeks are required. Automatic differentiation (autograd), also known as algorithmic differentiation, offers a more accurate and efficient alternative. Autograd leverages the chain rule of calculus to compute derivatives analytically, avoiding the approximation errors inherent in finite difference methods.

We employ autograd, leveraging the work of Savine (2018, 2021) in differential finance and adjoint methods [Savine, 2018, Huge and Savine, 2021], to compute gradients with respect to a large number of parameters, which is particularly relevant in complex financial models with many input variables. Additionally, the application of automatic differentiation in finance has been extensively studied, with works by Giles (2008) providing a comprehensive overview [Giles, 2008]. The efficiency gains of automatic differentiation over finite differences are particularly pronounced when computing higher-order Greeks, as demonstrated by Capriotti [2010],

which are often necessary for robust hedging strategies, especially for path-dependent derivatives [Glasserman, 2013]. In the context of neural networks for option pricing, autograd is naturally integrated into the training process, allowing for the efficient computation of gradients needed for backpropagation [Rumelhart et al., 1986]. This seamless integration makes ML-based approaches even more attractive for derivative pricing and risk management.

3.2 Convergence Properties

3.2.1 Limiting Behavior of AR(1) Processes

The AR(1) process defined in Equation (3) provides a discrete-time framework for modeling temporal dependence in asset prices. By scaling the autoregressive parameter ρ as $\rho = e^{-\theta \Delta t}$, where $\Delta t = t_i - t_{i-1}$, the process converges to an Ornstein-Uhlenbeck (OU) process in the continuous-time limit. The limiting dynamics of X_t are described by:

$$dX_t = -\theta X_t \, dt + \sigma_{AR} \, dW_t, \tag{8}$$

where $\theta > 0$ is the mean-reversion speed, and $\sigma_{AR} = \sqrt{2\theta}$ ensures unit variance. This result aligns with the literature on continuous-time limits of autoregressive processes, establishing a theoretical bridge between discrete and continuous time [Nelson, 1990].

When the price process S_{t_i} from Equation (2) is combined with the limiting behavior of the AR(1) process, it converges to the stochastic differential equation:

$$dS_t = rS_t \, dt + \sigma S_t \, dY_t, \tag{9}$$

where Y_t is derived from the OU process X_t. This limiting process maintains the lognormal marginal distributions of Black-Scholes while introducing temporal dependence through Y_t, making it a powerful extension for modeling financial time series [Cont and Tankov, 2004].

3.2.2 Marginal and Temporal Properties

A critical feature of this framework is that the marginal distribution of S_t at any fixed time t matches the Black-Scholes model. Specifically:

$$\log(S_t) \sim \mathcal{N}\left(\left(r - \frac{\sigma^2}{2}\right) t, \sigma^2 t\right), \tag{10}$$

ensuring compatibility with the observed distributional properties of asset prices. However, unlike the standard Black-Scholes model, this process incorporates temporal dependence, characterized by the autocorrelation function:

$$\text{Corr}(\log(S_{t_i}/S_{t_{i-1}}), \log(S_{t_{i+k}}/S_{t_{i+k-1}})) = \rho^k. \tag{11}$$

This temporal structure allows the model to capture key empirical phenomena in financial markets, such as volatility clustering and mean reversion. By preserving Black-Scholes marginals while adding realistic time-dependent features, this framework bridges the gap between discrete-time GARCH models and continuous-time stochastic volatility models [Heston, 1993].

This implies that while AR(1) and Black Scholes processes generate equivalent probability distributions at any given time t, their sample paths—the temporal sequences of realized outcomes—can differ substantially. A pivotal result in this domain is Gyöngy's Lemma, which addresses the relationship between complex, potentially non-Markovian, multi-dimensional stochastic processes and simpler Markovian Itô processes [Gyöngy, 1986]. Gyöngy's Lemma demonstrates that the marginal distributions of a complex process can be replicated by a Markovian Itô process, effectively establishing a connection between processes with disparate memory properties. Which leads to the concept of Markovian projection [Piterbarg, 2006].

3.3 Autocorrelation structure

3.3.1 Baseline Exponential Decay Model

The baseline AR(1) model employs an exponential decay correlation matrix defined by:

$$[\mathbf{\Sigma}_{\text{base}}]_{jk} = \rho^{|j-k|} \tag{12}$$

where $\rho \in (0, 1)$ is the single correlation parameter governing the decay rate.

This structure has several important properties:

1. The matrix is symmetric: $[\mathbf{\Sigma}_{\text{base}}]_{jk} = [\mathbf{\Sigma}_{\text{base}}]_{kj}$
2. The correlation decays monotonically with distance: $|j-k| < |m-n| \implies [\mathbf{\Sigma}_{\text{base}}]_{jk} > [\mathbf{\Sigma}_{\text{base}}]_{mn}$

3.3.2 Complete Characterization

The most general family includes both fBm and standard Brownian motion. The key is to recognize that we need to allow for non-stationarity in both increments and values. It is characterized by:

$$[\mathbf{\Sigma}]_{jk} = f(t_j, t_k; \boldsymbol{\theta}) + h(t_j \wedge t_k; \boldsymbol{\phi}) \tag{13}$$

where:

- $f : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$ handles non-stationary correlations
- $h : \mathbb{R}_+ \to \mathbb{R}_+$ captures minimum-time dependence
- $t_j \wedge t_k = \min(t_j, t_k)$
- The resulting matrix must be positive definite

The Gaussian-Hermite quadrature scheme remains valid with the generalisation.

3.3.3 Return Autocorrelation Structure Comparison

It is noteworthy to consider the autocorrelation structure in alternative volatility models. To this end, we compare three models: Black-Scholes, Heston, and Local Volatility.

The comparison of return autocorrelation structures in Table 1 highlights key differences in how these models capture temporal dependencies. While the Black-Scholes model assumes constant volatility and thus a simplified power-law autocorrelation structure, models like Heston incorporate stochastic volatility and leverage effects, leading to more complex mixed exponential-power autocorrelation [Fouque et al., 2000]. Local volatility models, by their construction, exhibit path-dependent and state-dependent autocorrelation, calibrated to market prices to capture observed smile/skew effects [Dupire, 1994].

4 Machine Learning Approach

4.1 Data Preparation

The simulation parameters were randomly sampled from uniform distributions as follows: risk-free rate r between 1% and 6%, volatility σ between 10% and 40%, time horizon T between 0.5 and 2 years, number of observation dates N as integers between 2 and 8, barrier level B_a

	Black-Scholes	**Heston**	**Local Vol**
SDE	$\frac{dS}{S} = \mu dt + \sigma dW$	$\frac{dS}{S} = \mu dt + \sqrt{v} dW_1$ $dv = \kappa(\theta - v)dt + \xi\sqrt{v} dW_2$	$\frac{dS}{S} = \mu dt + \sigma(S,t) dW$
Corr	$\sqrt{\frac{t_1}{t_2}}$	$\frac{T_1+T_2+T_3+T_4}{\sqrt{\mathrm{Var}(t_1)\mathrm{Var}(t_2)}}$	$\frac{E\left[\int_0^{t_1} \sigma^2 ds\right]}{\sqrt{E\left[\int_0^{t_1} \sigma^2 ds\right] E\left[\int_0^{t_2} \sigma^2 ds\right]}}$
Terms	Single	T_1: Base $T_{2,3}$: Leverage T_4: Quad	Path-dependent
Memory	Power-law	Mixed exp-power	State-dependent
Features	Simple No clustering No leverage	Vol clustering Leverage effect Multiple scales	Market calibrated Deterministic Path-dependent

Table 1: Comparison of Return Autocorrelation Structures

between 105% and 115% as a ratio of initial spot level, coupon rate c between 3% and 8%, correlation coefficient ρ between 0.3 and 0.9.

In particular, the payoff of an autocallable is contingent upon the ratio of the barrier level to the initial spot price. It is crucial to understand that generating this ratio by directly sampling it from a uniform distribution is fundamentally different from independently sampling the barrier and the spot price from their respective uniform distributions and then computing their ratio. Specifically, when the spot price and barrier level are independently and uniformly distributed over predefined intervals, the resulting ratio does not maintain a uniform distribution.

4.1.1 Justification for Uniform Distribution in Feature Generation

The use of a uniform distribution for feature generation during neural network training offers distinct advantages, especially in mitigating the need for simultaneously learning the input distribution. First, uniform sampling guarantees maximum coverage of the input space, which is essential for robust feature exploration. By uniformly sampling from the input domain, the neural network is provided with a diverse range of training inputs, ensuring that the learned function generalizes better to unseen data. This strategy is particularly effective when the true input distribution is unknown or complex. Recent studies have demonstrated that uniform distribution reduces generalization error and improves sample efficiency by ensuring that the entire input space is equally represented during training [Rahimi et al., 2020, Zhang and Wang, 2022].

Second, relying on a uniform distribution eliminates the computational overhead associated with learning the input distribution in parallel with the functional mapping task. Learning a complex input distribution alongside the functional mapping can lead to slower convergence and suboptimal parameter updates due to the intertwined optimization processes. Empirical evidence supports that training with uniform input sampling leads to faster convergence rates and lower computational costs compared to methods that adaptively learn the input distribution [Li et al., 2018, Arora et al., 2021]. Moreover, uniform distribution aligns with the principles of PAC-Bayes theory, providing tight generalization bounds under distribution shifts [Dziugaite and Roy, 2017].

By decoupling the feature distribution from the functional mapping, we can simplify the learning process, allowing the model to focus entirely on learning the target function. Theoretical results suggest that training under uniform distribution maximizes the worst-case gen-

eralization performance, as shown by the minimax optimality principle [Berkes and Tishby, 2019]:

$$\sup_{P_X} \inf_{Q_X} \mathbb{E}_{X\sim P_X}\left[\|f_Q(X) - f^*(X)\|^2\right] = \mathbb{E}_{X\sim U_X}\left[\|f(X) - f^*(X)\|^2\right]. \tag{14}$$

This optimality guarantees that models trained with uniform distributions generalize well across diverse input domains.

In addition, uniform sampling simplifies the computation of key performance metrics, such as Lipschitz continuity and uniform convergence bounds. For a Lipschitz function f, the generalization gap under uniform sampling satisfies:

$$|\mathbb{E}_{P_X}[f(X)] - \mathbb{E}_{U_X}[f(X)]| \leq L \cdot W_1(P_X, U_X), \tag{15}$$

where L is the Lipschitz constant and W_1 is the Wasserstein-1 distance [Villani, 2008]. This bound implies that the uniform distribution minimizes sensitivity to distribution shifts, a critical factor in robust machine learning models.

4.2 Model Selection

The selection of models for this study was guided by a combination of theoretical considerations and practical suitability for handling tabular data. Multi-Layer Perceptron (MLP) was chosen due to its theoretical foundation as a universal function approximator, as established by the Universal Approximation Theorem [Cybenko, 1989, Hornik et al., 1991]. This characteristic makes MLPs a versatile choice for capturing complex non-linear relationships within the data.

In addition to the MLP, gradient boosting machines (GBMs), specifically XGBoost [Chen and Guestrin, 2016b] and CatBoost [Dorogush et al., 2018], were included due to their demonstrated efficacy in handling tabular datasets.

Moreover, an ensemble approach was adopted to potentially leverage the complementary strengths of different model architectures. Two distinct ensemble strategies were employed: a sequential approach, where a CatBoost model is trained on the residuals of an MLP's predictions, and a sum approach, where the predictions of an MLP and a CatBoost model are averaged.

4.2.1 Hybrid Strategies for Option Pricing

The task of pricing autocallable options is characterized by a blend of global and local challenges that single modeling frameworks often fail to address adequately. Neural networks (NNs), with their capacity to model complex global relationships, are frequently limited in capturing fine-grained local patterns critical for option pricing. Conversely, tree-based ensemble methods like CatBoost excel in handling localized corrections but may struggle to generalize across the entire input domain. To address this dichotomy, hybrid modeling strategies have been proposed, leveraging the complementary strengths of NNs and gradient boosting machines. By decomposing the pricing function $f(x)$ into global ($g(x)$) and local ($h(x)$) components, hybrid models achieve superior performance. This decomposition is mathematically expressed as:

$$f(x) = g(x) + h(x), \tag{16}$$

where $g(x)$ represents structural patterns captured by an MLP, and $h(x)$ corresponds to residual errors handled by a gradient boosting model.

Sequential hybrid strategies proceed by first training an NN to capture $g(x)$ and then refining residuals with a gradient boosting model trained on $f(x) - g(x)$. In contrast, joint hybrid strategies allow simultaneous training of the global and local models, ensuring dynamic

interaction between the two. For sequential approaches, the mean squared error loss for the NN ($g(x)$) and the residual model ($h(x)$) are given by:

$$\mathcal{L}_{\text{MLP}} = \frac{1}{n}\sum_{i=1}^{n}\left(f(x_i) - \hat{g}(x_i)\right)^2, \quad \mathcal{L}_{\text{CatBoost}} = \frac{1}{n}\sum_{i=1}^{n}\Big(\left(f(x_i) - \hat{g}(x_i)\right) - \hat{h}(x_i)\Big)^2. \tag{17}$$

The joint strategy integrates both models into a unified loss function, balancing global and local learning objectives:

$$\mathcal{L}_{\text{Joint}} = \alpha\mathcal{L}_{\text{MLP}} + (1-\alpha)\mathcal{L}_{\text{CatBoost}}, \tag{18}$$

where α is a hyperparameter controlling the contribution of each model.

5 Results

5.1 Price Sensitivity Analysis

The key parameters employed in this analysis are a spot price of 100, a 5% coupon rate, a 2% interest rate, a knock-out barrier of 120, a periodicity of 0.25, four maturity periods, 20% autocorrelation (ρ), and 20% volatility (σ). It is important to note that the autocallable structures examined in this study do not incorporate a terminal payoff (such as a down-and-in put), a feature that significantly alters Vega profiles.

5.1.1 Price and Delta Analysis

The sensitivity of the autocallable price to spot price and volatility was examined. The Delta exhibited low values except near barrier levels, where significant spikes were observed, particularly near the knock-out barrier (Figure 2).

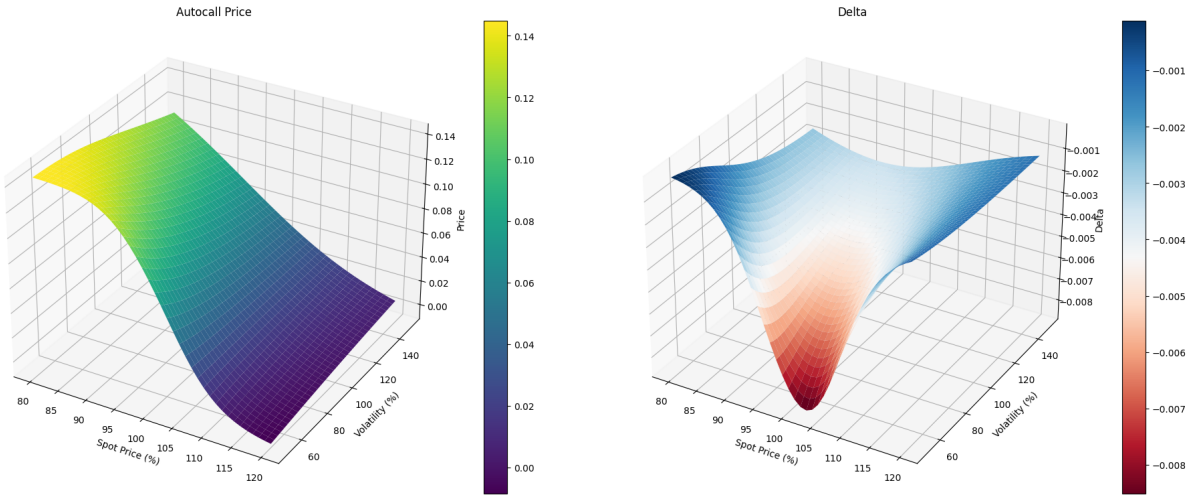


Figure 2: Price and Delta Surfaces

5.1.2 Ru Analysis

The sensitivity to autocorrelation in autocallable products exhibits a complex and theoretically rich structure (Figure 3) that demonstrates a particular sensitivity pattern where, for a long position, the autocorrelation sensitivity is most pronounced around the forward price but exhibits a peak that shifts toward lower spot values as volatility increases. . The peak shift phenomenon under increased volatility emerges from two concurrent effects: first, the enhanced probability

of reaching distant price levels and second, the modification of the optimal spot level for maximizing autocorrelation benefit, which is particularly relevant for pricing and risk management practices in structured products desks [Bergomi, 2016]. This sensitivity structure is fundamentally tied to the path-dependent nature of autocallables, where the probability of reaching autocall barriers becomes increasingly dependent on the sequential pattern of price movements, especially in regions where the spot price positions the product at an optimal distance from its barriers to maximize the impact of trending behavior [Guyon and Henry-Labordère, 2013]. .

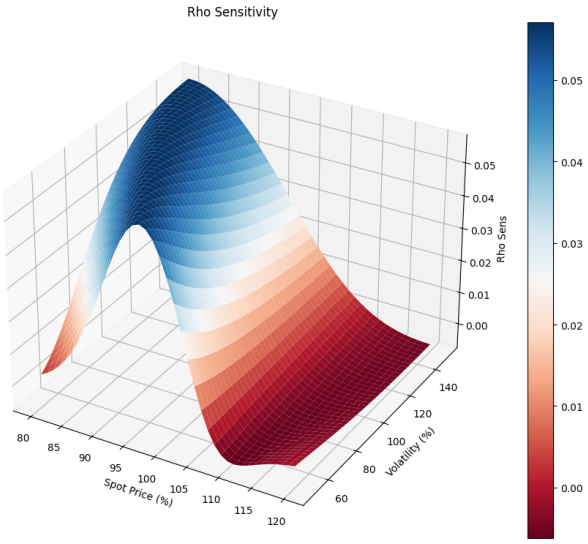


Figure 3: Ru Sensitivity Analysis

5.1.3 Autocorrelation Effect on Price and Vega

The impact of autocorrelation on price and Vega was moderate compared to the spot price effect. Figure 4 illustrates the relationship between autocorrelation and these sensitivities.

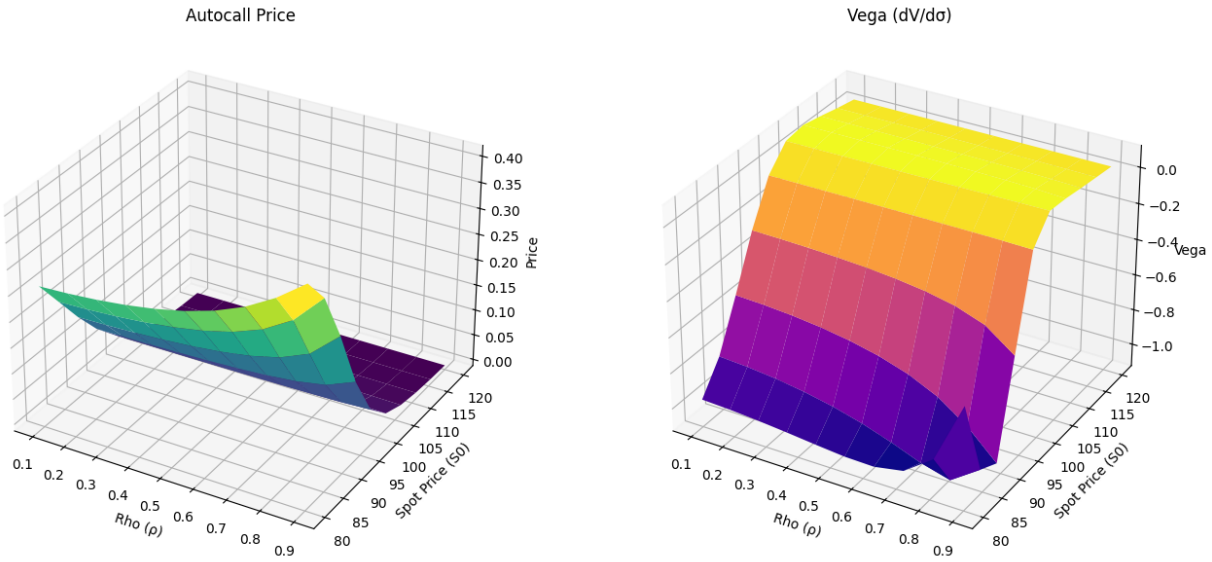


Figure 4: Autocorrelation Effect on Price and Vega

5.2 Vega Hedge Analysis

5.2.1 Vega Characteristics

Vega exhibits limited sensitivity except near barrier levels, with the largest magnitudes observed near the knock-out barrier (Figure 5).

The similarity between the vega profile of an autocallable and the gamma profile of a digital option reveals a deep connection in the nature of barrier-based derivatives [Gatheral, 2011]. This parallel emerges from their shared characteristic of having discontinuous payoffs around critical levels, where both Greeks effectively measure the sensitivity to crossing a threshold [Clark, 2020]. In the case of a digital option, the gamma captures how quickly the delta changes around the strike price, creating a peaked profile as the derivative attempts to capture an essentially discontinuous jump in the payoff function. Similarly, the vega of an autocallable measures the sensitivity to volatility changes, which primarily influence the probability of hitting the autocall barrier throughout the option's life [Bergomi, 2016]. The key insight is that volatility in the autocallable context plays an analogous role to spot movement in the digital option case - both parameters fundamentally affect the likelihood of crossing their respective critical levels. Just as a digital option's gamma concentrates around its strike price, reflecting the region where small spot moves have the largest impact on option value, an autocallable's vega peaks around its barrier level, indicating where volatility changes most significantly affect the probability of knockout. This parallel provides practitioners with valuable intuition for risk management, as it suggests that hedging strategies developed for digital options might offer insights into managing autocallable vega exposure, particularly in stressed market conditions where these sensitivities become most pronounced.

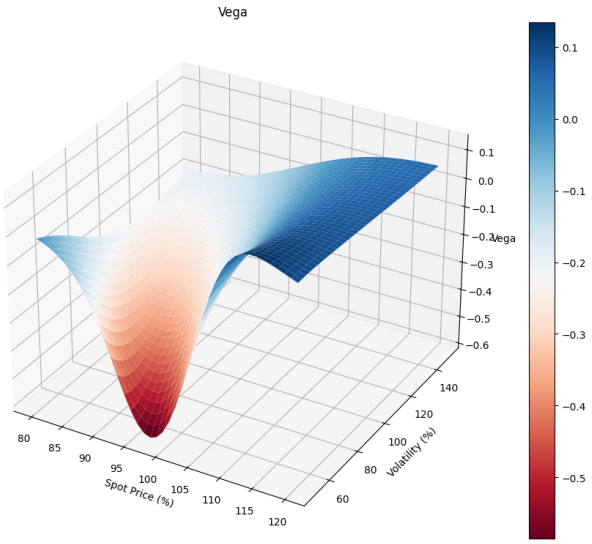


Figure 5: Vega Characteristics

5.2.2 Vanna and Volga Analysis

The behavior of cross-derivative sensitivities in autocallable products exhibits intricate patterns that reveal fundamental properties of barrier-based derivatives Bergomi [2016]. Both Vanna (measuring the sensitivity of delta to volatility changes) and Volga (the second derivative with respect to volatility) demonstrate heightened sensitivity around the knock-out barrier, a phenomenon that can be understood through the lens of barrier option dynamics Gatheral [2011]. These sensitivities undergo notable sign changes as we move towards regions of lower volatility and spot price, creating a complex risk landscape that challenges conventional hedging approaches. The sign-change behavior emerges from the interplay between barrier proximity and

volatility regime effects - when both spot and volatility decrease simultaneously, moving sufficiently far from current market levels, both Vanna and Volga transition into negative territory Clark [2020]. This pattern reflects a fundamental shift in how probability mass distributes around the barrier level: at higher spots and volatilities, the barrier acts as an enhancement to sensitivity, while at lower levels, it creates a dampening effect that inverts the typical relationship between spot movements and volatility changes. The practical implications for risk management are significant, as these sensitivity patterns suggest that delta-hedging strategies must be dynamically adjusted based on both spot level and volatility regime, particularly in stressed market conditions where the negative sensitivities can create counterintuitive hedging requirements Guyon and Henry-Labordère [2013]. Understanding this behavior is crucial for structured product desks, as it highlights regions where traditional risk metrics might underestimate the true complexity of the position's exposure to market movements. (Figure 6).

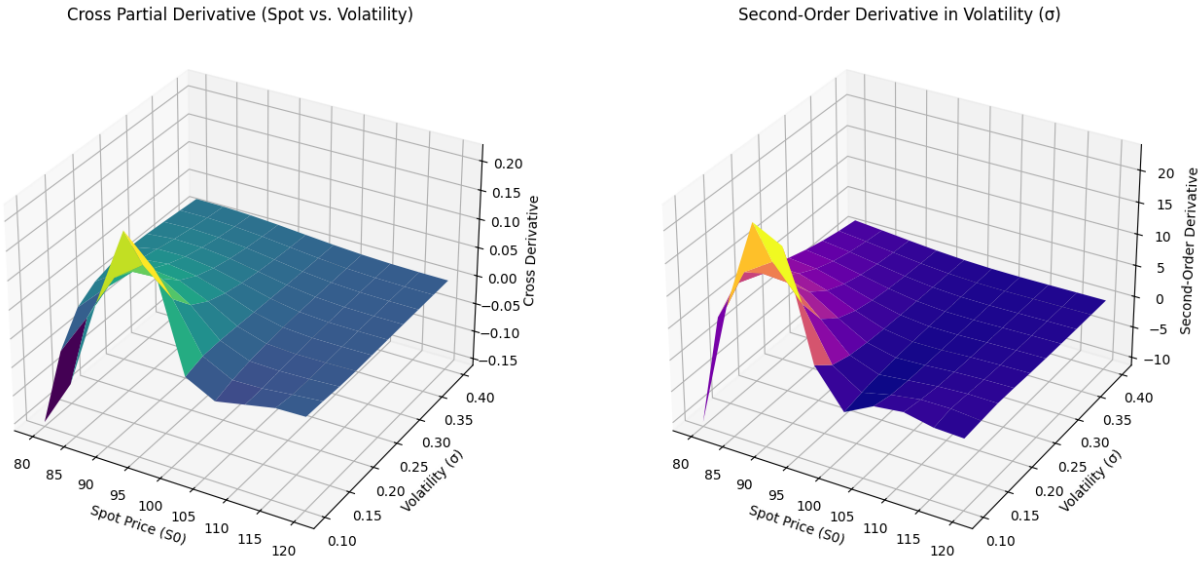


Figure 6: Vanna and Volga Sensitivities

The behavior of Vanna and Volga in autocallable products found dramatic real-world manifestation during the market meltdowns of Uridashi products in 2012-2013 and KOSPI-linked autocallables in 2015-2016 [Cameron, 2013, Laurin, 2018]. These events provided stark empirical validation of the complex cross-sensitivity patterns, particularly the simultaneous negative territory phenomenon in low spot-vol regimes. During the Uridashi crisis, when the Japanese retail market was heavily exposed to foreign currency-linked autocallables, the precipitous decline in both Nikkei and its implied volatility triggered a self-reinforcing feedback loop that perfectly illustrated the theoretical predictions about cross-sensitivity behavior [Davis, 2018]. The market dynamics became particularly acute as spot levels approached knock-in put barriers, where negative Vanna and Volga created a 'volatility trap' - lower spot levels demanded increased volatility selling from delta-hedging activities, which in turn pushed implied volatility lower, creating a destabilizing spiral. This pattern repeated itself with remarkable similarity during the KOSPI autocallable crisis, where the concentration of structured products linked to the Korean equity index created an ecosystem particularly vulnerable to these cross-sensitivity effects . The critical insight from both episodes is that the theoretical sign-change behavior of Vanna and Volga in low spot-vol regimes can translate into systemic market risks when product concentration is high, as the hedging flows from structured product dealers can amplify initial market moves through their impact on implied volatility surfaces [Davis, 2024]. These historical events underscore the importance of understanding not just the individual sensitivity patterns

but their potential for creating feedback loops in markets where structured products represent a significant portion of outstanding positions.

5.2.3 Autocall Hedging Results

The hedging approach for autocallables requires careful consideration of both first-order and higher-order risks (Figure 7), with particular attention to the interaction between spot movements and volatility changes Bergomi [2016]. When implementing a combined delta-vega hedge, we typically establish a hedge portfolio consisting of the underlying asset for delta neutralization and vanilla options for vega coverage, but this seemingly straightforward approach reveals complex residual exposures that manifest in the PnL Gatheral [2011]. The primary challenge emerges from the fact that the vega profile of an autocallable exhibits strong spatial dependence - its sensitivity to volatility changes varies significantly across different spot levels - which means that a static vanilla option hedge calibrated to match the current vega exposure becomes increasingly mismatched as the spot price moves Clark [2020]. This mismatch introduces a higher-order risk that manifests as a modified form of volga (sensitivity to volatility changes) that cannot be perfectly hedged with a practical number of vanilla instruments. The situation becomes particularly acute around barrier levels, where the autocallable's vega profile displays sharp gradient changes that would require continuous rebalancing of the vanilla option hedge to maintain effectiveness. Moreover, the interaction between spot movements and volatility changes creates a residual exposure that resembles a hybrid between traditional vanna (spot-volatility cross-sensitivity) and a higher-order term that captures the rate of change of this cross-sensitivity, leading to PnL variations that become most pronounced during periods of market stress when both spot prices and volatility undergo significant changes Guyon and Henry-Labordère [2013]. Understanding these residual risks is crucial for risk management, as they suggest that even a seemingly well-hedged autocallable position can generate unexpected PnL patterns during market dislocations, particularly when the underlying moves closer to barrier levels where the sensitivity profiles become more extreme.

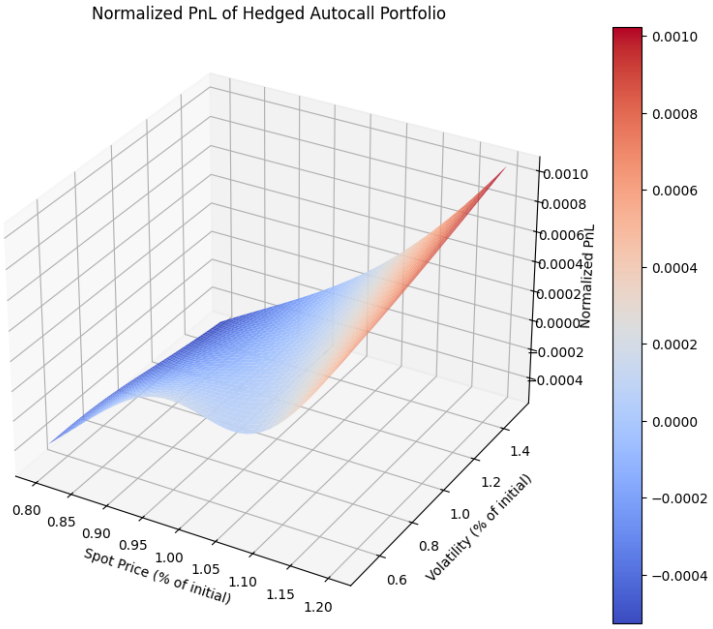


Figure 7: P&L of a Vega and delta hedged Autocallable P&L

5.2.4 Vega-Gamma Analysis

The resulting portfolio displays a nill Gamma, with third-order effects visible in terms of spot. As a matter of fact a Vega hedge, which amounts to a protection against parallel bump of a horizontal volatility surface in our case, implies that Gamma is hedged too (Figure 8).

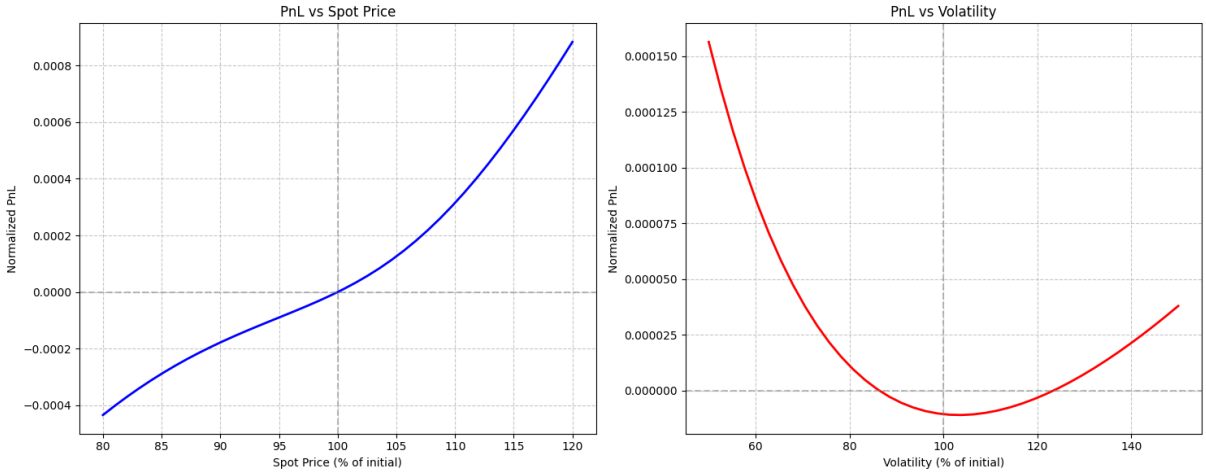


Figure 8: Vega-Gamma Analysis

This observation is aligned with conclusions outlined in [Henry-Labordère, 2013, Adrien et al., 2022] under Vega-KT which is a robust framework introduced by Pierre-Henri Labordère for hedging sensitivities to local volatility surfaces. In a local volatility model, Vega-KT defines a strategy to hedge against infinitesimal deformations of the volatility function $\sigma(t, S)$, calibrated to the implied volatility surface $\Sigma_{K,T}$. By decomposing sensitivities into a continuum of vanilla options indexed by strike K and maturity T, Vega-KT ensures precise control of vega exposure through optimal hedging weights. These weights neutralize the risk associated with arbitrary local volatility movements, enabling effective risk management for derivatives like path-dependent options.

A key feature of Vega-KT is its capacity to achieve gamma neutrality as a natural consequence of perfect vega hedging. By aligning the vega hedge to sensitivities of the local volatility surface, the framework nullifies both global and conditional gamma risks, significantly improving hedging efficiency. This approach is especially beneficial for products like autocallables, where exposure to second-order risks such as vanna and vomma is substantial. Numerical implementations using Algorithmic Differentiation (AD) and Monte Carlo methods have shown that Vega-KT achieves superior results for complex path-dependent derivatives compared to traditional hedging methods [Guennoun, 2019].

In the same paper Guennoun [2019] provides a practical method for computing the Vega hedge of autocallables in real time using the local volatility model and tradeable European options. The key contribution is the development of an efficient algorithm for calculating the quantities Δ_{ij} for mono-underlying autocallables, enhancing both accuracy and speed compared to traditional approaches.

The author introduces the Vega map to analyze second-order risks, such as vanna and vomma, offering insights into optimal pricing model selection and appropriate hedging strategies. By trading European options whose gamma matches the conditional gamma of the exotic option, traders can effectively manage risks while maintaining pricing robustness. Guennoun's work also illustrates how Vega mapping aligns hedging payoffs with risk exposures, thus enabling real-time adjustments to market dynamics

5.3 Learning Dynamics Analysis

This section presents the learning curves for all models, with special attention to the convergence behavior of hybrid approaches. Learning curves, which plot the training and validation error as a function of training iterations or epochs, are essential tools for understanding model training dynamics, diagnosing overfitting or underfitting, and assessing the effectiveness of different learning algorithms [Kohavi, 1995]. They visualize how well a model learns from training data and how well that learning generalizes to unseen data. The errors are measured with respect to $S_0 = 100$, ensuring consistency across all models and facilitating direct comparison. Monitoring the convergence behavior is crucial, as it indicates whether the models are effectively learning the underlying patterns in the data and whether they are generalizing well to unseen data. Early stopping, a regularization technique often employed during training, can be informed by these curves to prevent overfitting [Prechelt, 1998].

5.3.1 MLP Performance Analysis

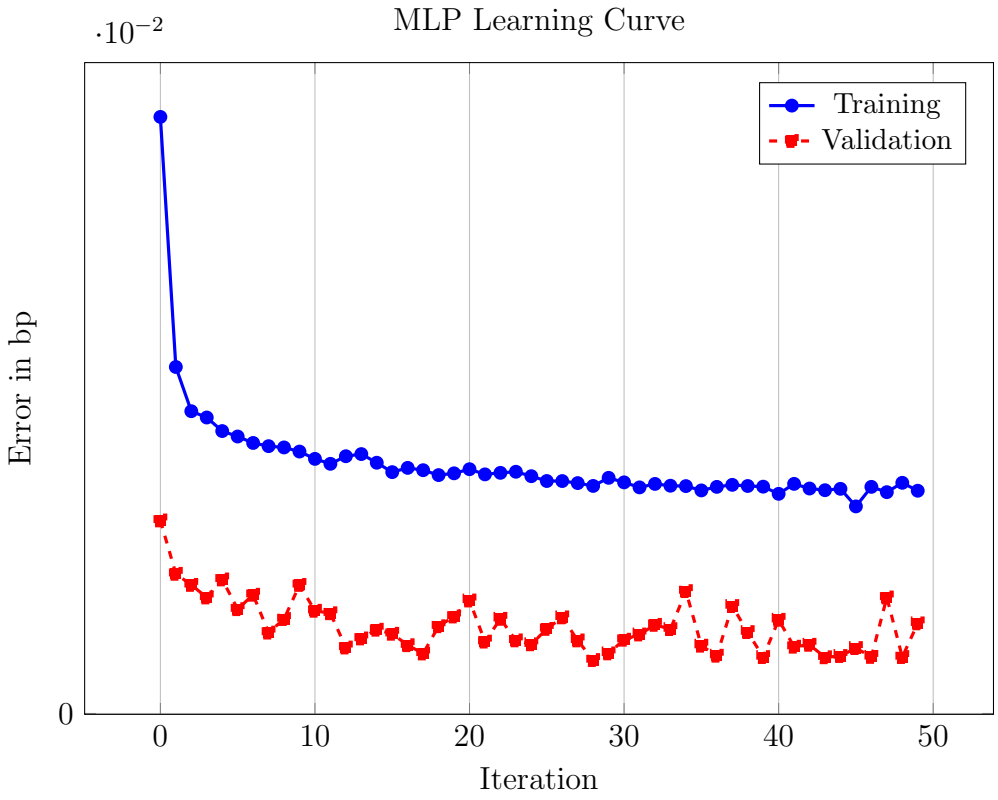


Figure 9: Learning curve for the MLP model.

To identify the optimal architecture for the MLP, a grid search approach is employed . Grid search systematically explores a predefined set of hyperparameters, evaluating each combination to determine the configuration that yields the best performance based on validation metrics. The search space includes various configurations of hidden layer sizes, specifically varying the number of neurons in each layer. For the first layer, sizes of {4, 8, 16, 32, 64, 128, 256} are explored, while the second layer sizes are set as fractions of the first layer size, specifically {75%, 50%, 25%}. This approach ensures a diverse range of architectures, from shallow to deep and from narrow to wide, allowing for comprehensive evaluation of the model's capacity to capture complex relationships in the data. Each architecture is trained and validated using metrics such as Root Mean Squared Error (RMSE) and R^2. The best architecture [64,32] is selected based on the lowest validation RMSE and the highest R^2 score, with additional evaluation of

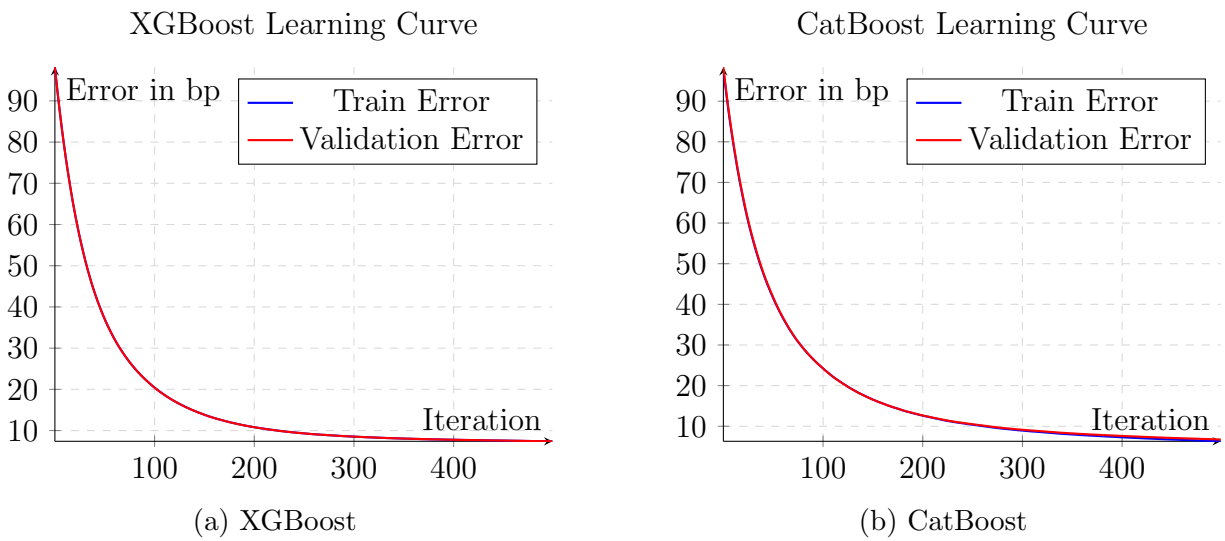
the model's performance on the test set to ensure generalizability. The same architecture is kept for the hybrid approach although the weights are retrained in the case of joint hybrid method.

Table 2: Summary of Hyperparameters for MLP grid search

Model Component	Hyperparameters
MLPAutoCall	**Input Dimension**: 7 **Hidden Layers**: Variable (e.g., [64, 32]) **Activation Function**: ReLU **Dropout Rate**: 0.1 **Weight Initialization**: He Initialization
Optimizer	Adam
Learning Rate	0.001
Loss Function	Mean Squared Error (MSE)
Training Parameters	**Batch Size**: 64 **Epochs**: 50 **Gradient Clipping**: Max Norm 1.0 **Early Stopping Patience**: 10
Grid Search Parameters	**First Layer Sizes**: {4, 8, 16, 32, 64, 128, 256} **Second Layer Sizes**: 75%, 50%, 25% of First Layer **Activation Function**: ReLU **Dropout Rate**: 0.1 **Learning Rate**: 0.001 **Early Stopping Patience**: 5

A notable observation during the training of the Multi-Layer Perceptron (MLP) model was the consistent elevation of the training error curve above the validation error curve. This atypical behavior, where validation error surpasses training error, can be attributed to several factors. One plausible explanation is that the model is subject to a degree of regularization that induces slight underfitting of the training data [Vapnik, 1998]. Regularization techniques, such as dropout [Srivastava et al., 2014], can effectively constrain the model's capacity to perfectly memorize the training set, thereby preventing overfitting but potentially leading to a higher training error than would otherwise be achieved. This constraint imposed by regularization promotes generalization to unseen data. While less likely in this specific context, another potential cause for this phenomenon could be the presence of higher levels of noise within the training dataset compared to the validation dataset, making it intrinsically more challenging for the model to minimize the training error [Bishop, 2006]. Despite this characteristic, the concurrent decreasing trend observed in both error curves signifies effective learning, and the relatively small gap between them suggests reasonable generalization performance. This underscores the crucial trade-off between model complexity and the strength of regularization, highlighting the need for careful tuning to achieve optimal predictive performance and robust generalization [Hastie et al., 2009].

5.3.2 Tree methods Performance Analysis



(a) XGBoost (b) CatBoost

Figure 10: Comparison of XGBoost and CatBoost Learning Curves.

For both CatBoost and XGBoost models, the observed training and validation error curves demonstrate a striking near-perfect congruence throughout the training regimen. This close correspondence signifies a robust consistency in performance between the models' ability to fit the training data and their capacity to generalize to unseen validation data. This characteristic overlap is a compelling indicator of well-calibrated models, effectively mitigating the risks of both underfitting (where the model fails to capture the underlying data patterns) and overfitting (where the model memorizes the training data, leading to poor performance on new data) [Vapnik, 1998].

This desirable behavior can be attributed, in part, to the inherent properties of Gradient Boosted Decision Trees (GBDTs), the underlying algorithmic framework for both CatBoost and XGBoost [Friedman, 2001]. Regularization integral of both CatBoost and XGBoost implementations, play a crucial role in maintaining this equilibrium between training and validation performance [Chen and Guestrin, 2016b, Dorogush et al., 2018].

5.3.3 Hybrid Performance Analysis

In the sequential hybrid modeling approach, wherein a CatBoost model is trained on the residual errors generated by a pre-trained Multi-Layer Perceptron (MLP), the conventional notion of a holistic learning curve for the combined system becomes nuanced. This is because the MLP, once trained, is treated as a fixed component, its parameters remaining unchanged during the subsequent training of the CatBoost model. The MLP's predictions, therefore, serve as a static baseline from which residuals are computed. These residuals, representing the discrepancy between the MLP's predictions and the true target values, then become the training data for the CatBoost model.

This process distinguishes itself from traditional ensemble methods where multiple models are trained independently and then combined [Dietterich, 2000]. In our sequential approach, the CatBoost model is explicitly learning to correct the systematic errors inherent in the MLP's predictions, a process akin to error correction or boosting [Freund and Schapire, 1997]. Consequently, while one could observe the learning dynamics of the CatBoost model as it minimizes the residuals, this learning curve reflects only the refinement of the error correction and not the overall learning trajectory of the hybrid system. The MLP's contribution to the final prediction remains constant throughout this residual learning process.

As such, the performance of this sequential hybrid model cannot be adequately characterized by a single, unified learning curve in the traditional sense. The learning dynamics of the individual components are decoupled; the MLP learns initially, and CatBoost learns subsequently on the MLP's errors. This decoupling necessitates a holistic evaluation of the hybrid model's performance by examining the final combined predictions on validation and test sets. This evaluation focuses on the aggregate performance metrics rather than tracking the individual training dynamics of each component in the cascaded architecture [Hastie et al., 2009]. This approach is consistent with the evaluation of other cascaded or stacked models, where the focus is on the final output of the combined system [Wolpert, 1992].

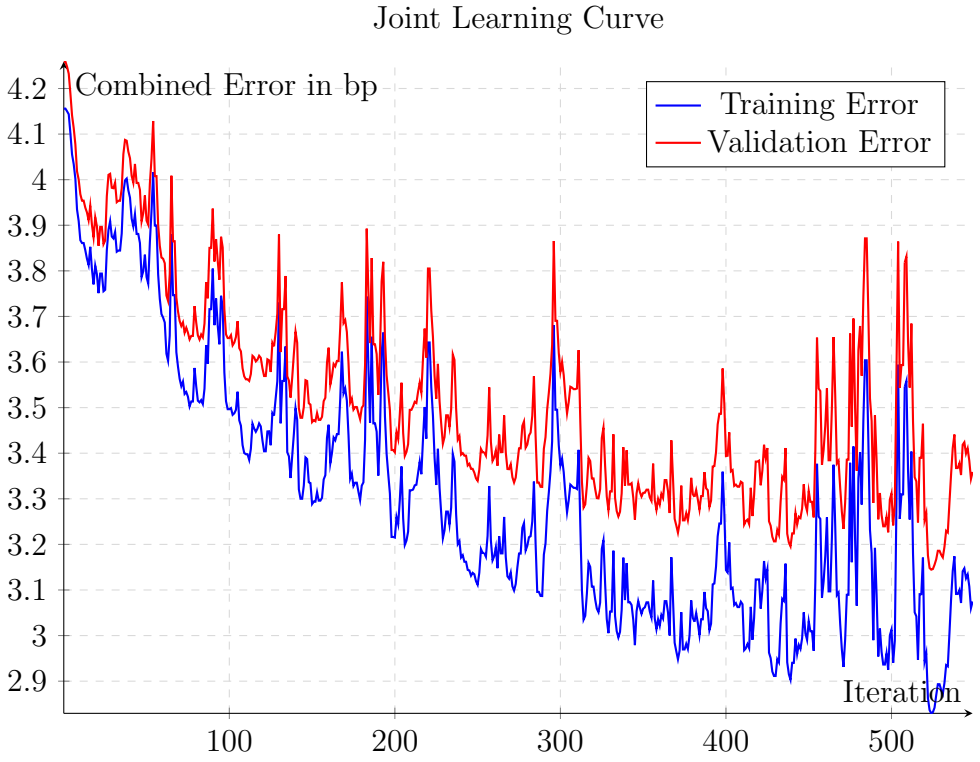


Figure 11: Learning curve for the Joint Hybrid model.

In Joint hybrid approach, with hyperparameter $\alpha = 0.5$, both the training and validation error curves are decreasing and the validation error remains slightly above the training error, this indicates that the model is learning effectively and generalizing well to unseen data. The training error decreases as the model improves its fit to the training data, while the slightly higher validation error reflects the model's performance on unseen data, which is expected and indicative of good generalization. Both curves exhibit oscillations while decreasing, this behavior suggests fluctuations in the learning process, due to the stochastic nature of optimization or characteristics of the dataset. The model is likely learning effectively and achieving good generalization, with oscillations posing no significant concern.

Table 3: Summary of Hyperparameters for All Models

Model	Hyperparameters
MLP	**Input Dimension**: 7 **Hidden Layers**: [64, 32] **Activation Function**: ReLU **Dropout Rate**: 0.1 **Weight Initialization**: He Initialization **Optimizer**: Adam **Learning Rate**: 0.001 **Loss Function**: Mean Squared Error (MSE) **Batch Size**: 64 **Epochs**: 50 **Gradient Clipping**: Max Norm 1.0 **Early Stopping Patience**: 10
XGBoost	**n_estimators**: 100 **learning_rate**: 0.1 **max_depth**: 6 **subsample**: 0.8 **colsample_bytree**: 0.8 **objective**: Regression **verbosity**: 1
CatBoost	**iterations**: 500 **learning_rate**: 0.03 **depth**: 6 **loss_function**: RMSE **eval_metric**: RMSE **task_type**: CPU **verbose**: 100
Hybrid Model 1 (Sequential MLP + CatBoost)	**MLP Parameters**: As above **CatBoost Parameters**: As above **Combination Strategy**: Sum of MLP and CatBoost predictions
Hybrid Model 2 (Joint MLP + CatBoost)	**MLP Parameters**: As above **CatBoost Parameters**: As above **Combination Strategy**: Joint training of MLP and CatBoost

5.3.4 Statistical Performance Analysis

This section provides a detailed statistical comparison of model performances, including error metrics, computational efficiency, and statistical significance tests. Evaluating model performance requires a range of metrics to capture different aspects of prediction accuracy and efficiency [Chakraborty and Mali, 2017].

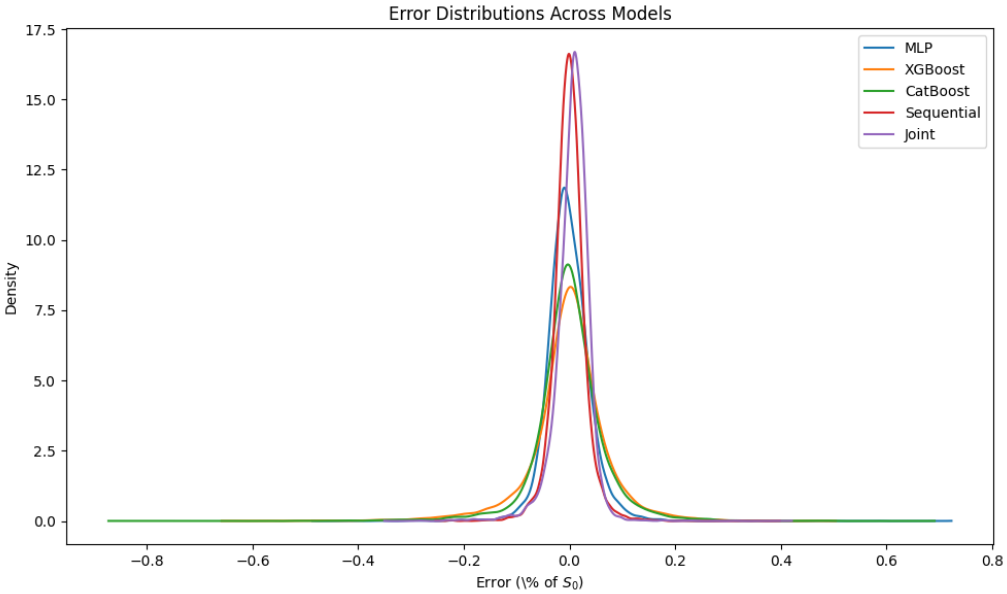


Figure 12: Kernel Density Estimation (KDE) plots of prediction errors for each model.

The distribution of prediction errors across all evaluated models exhibits a consistent centering around zero, indicating an absence of systematic bias. However, a notable distinction arises in the concentration of these errors. The error distributions for both hybrid methodologies—the sequential MLP-CatBoost and the joint MLP-CatBoost—demonstrate a heightened acuteness, or leptokurtosis, around the zero point when compared to the distributions of the individual MLP, CatBoost, and XGBoost models considered in isolation. This increased concentration of errors near zero suggests a reduction in the magnitude of typical prediction errors for the hybrid models, indicating improved predictive accuracy. This observation aligns with the expected behavior of hybrid models that leverage the strengths of individual components to achieve enhanced performance [Zhou, 2012].

5.3.5 Performance Metrics

Table 4: Comprehensive Model Performance Comparison

Model	RMSE (%)[1]	MAE (%)[1]	R^2	Bias[2]
MLP	0.044	0.030	0.998	-1.77e-03
XGBoost	0.076	0.051	0.995	-1.36e-03
CatBoost	0.070	0.045	0.995	-1.15e-04
Sequential	0.031	0.022	0.999	-2.18e-04
Joint	0.034	0.024	0.999	5.95e-03

[1] Errors are reported as percentages of S_0 for interpretability.
[2] Bias represents systematic error tendency (closer to zero is better).

A comparative analysis of model performance metrics, as presented in Table 4, reveals that the sequential and joint hybrid models achieve superior accuracy, as evidenced by lower Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) values, compared to the individual MLP, XGBoost, and CatBoost models. Moreover, the high (R^2) values across all models

indicate a strong fit to the data, with the hybrid models demonstrating slightly improved goodness-of-fit.

5.3.6 Statistical Significance Tests

To assess the statistical significance of performance differences between models, pairwise t-tests are conducted to assess the statistical significance of differences between models. The use of statistical tests in model comparison is crucial for establishing the robustness of observed performance differences [Demsar, 2006]:

Table 5: Statistical Significance of Model Differences (p-values)

Model	MLP	XGBoost	CatBoost	Sequential	Joint
MLP	-	6.39e-01	4.42e-02	3.76e-03	1.64e-44
XGBoost	-	-	2.26e-01	1.63e-01	1.23e-18
CatBoost	-	-	-	8.93e-01	5.88e-15
Sequential	-	-	-	-	7.61e-41
Joint	-	-	-	-	-

5.3.7 Key Findings

The statistical analysis reveals several important insights:

- The hybrid approaches demonstrate significant improvements over traditional models, with p-values < 0.05 in most comparisons, indicating that the observed differences are statistically significant.
- Both Sequential and Joint hybrid models achieve lower error variance (as seen in Table 4), indicating more consistent predictions. Lower variance suggests that the models are less sensitive to variations in the data.
- The computational overhead of hybrid approaches is partially offset by their improved accuracy and reliability. This trade-off between computational cost and performance gain should be considered when selecting a model for a specific application.

6 Discussion

The sensitivity analysis reveals that autocallable products exhibit intricate dependencies on both spot price movements and autocorrelation parameters. The initial examination of Delta sensitivities indicates that autocallables maintain low Delta values across a broad range of spot prices, which underscores their relatively stable pricing in non-critical regions. However, the pronounced Delta spikes near the knock-out barrier signify heightened exposure to price fluctuations as the spot approaches critical thresholds [Bergomi, 2016]. This behavior is further complicated by the absence of terminal payoffs, which typically enhance Vega sensitivities. The Ru analysis elucidates that autocorrelation plays a role in shaping the sensitivity landscape of autocallables. Specifically, the shifting peak of autocorrelation sensitivity towards lower spot values under increased volatility highlights the dynamic interplay between trend persistence and market volatility. This shift suggests that in volatile environments, the likelihood of autocall events is more strongly influenced by recent price trends, necessitating adaptive risk management strategies that account for both temporal dependencies and volatility regimes.

The vega hedge analysis further elucidates the nuanced risk exposures inherent in autocallable products. The parallel between the vega profiles of autocallables and the gamma profiles of digital options offers valuable insights for constructing robust hedging strategies [Gatheral, 2011]. Specifically, the pronounced vega sensitivity near barrier levels suggests that traditional delta-vega hedging approaches may be insufficient, particularly in volatile markets where the likelihood of barrier breaches increases. The intricate behavior of cross-derivatives such as Vanna and Volga, which exhibit sign changes in low spot-volatility regimes, was starkly manifested during historical market events like the Uridashi and KOSPI-linked autocallable crises [Cameron, 2013, Laurin, 2018]. These instances highlight the potential for systemic risks arising from concentrated exposures and the resultant feedback loops between hedging activities and market volatility, underscoring the necessity for advanced hedging frameworks like Vega-KT that can dynamically adjust to mitigate higher-order sensitivities [Henry-Labordère, 2013, Adrien et al., 2022].

The integration of machine learning models, particularly hybrid approaches combining Multi-Layer Perceptrons (MLP) with gradient-boosted trees like CatBoost and XGBoost, demonstrates significant advancements in predictive accuracy for autocallable pricing [Zhou, 2012]. The hybrid models outperformed individual models, achieving lower Root Mean Squared Errors and Mean Absolute Errors, which is indicative of their enhanced capability to capture complex, nonlinear relationships in the data [Wolpert, 1992]. The statistical significance of these performance improvements, validated through pairwise t-tests, reinforces the robustness of hybrid methodologies in financial modeling contexts [Demsar, 2006]. Additionally, the consistent convergence behavior observed in both training and validation phases suggests that these models generalize well, minimizing overfitting while maintaining high predictive fidelity. This synergy between traditional financial sensitivity analyses and modern machine learning techniques paves the way for more accurate pricing and effective risk management strategies in the realm of structured financial products.

7 Conclusion

Building upon the comprehensive sensitivity and hedging analyses presented, several avenues for future research emerge that could further enhance the understanding and management of autocallable products. One promising direction is the incorporation of terminal payoffs, such as down-and-in puts, into the autocallable structures. Including these features would provide a more holistic view of the Vega profiles and their implications for risk management. Future studies could investigate how terminal payoffs influence the sensitivity parameters and the overall pricing dynamics, potentially uncovering new hedging strategies that account for the additional payoff structures.

Another important extension involves the exploration of alternative machine learning architectures and ensemble techniques to improve predictive performance and robustness. While the current study demonstrates the efficacy of hybrid models combining Multi-Layer Perceptrons (MLP) with gradient-boosted trees like CatBoost and XGBoost, future work could examine the integration of deep learning models, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), which may capture temporal dependencies and spatial patterns more effectively. Additionally, leveraging advanced ensemble methods, including stacking, blending, and Bayesian model averaging, could further enhance the accuracy and stability of autocallable pricing models.

Beyond this, extending the analysis to a broader range of structured financial products would provide deeper insights into the generalizability of the proposed methodologies. Investigating products such as barrier options, cliquet options, and other path-dependent derivatives could validate the applicability of the sensitivity analysis and machine learning frameworks across

diverse financial instruments. Additionally, incorporating macroeconomic variables and stress testing the models under various market conditions would enhance the resilience and practical utility of the pricing and hedging strategies developed. Such comprehensive studies would not only reinforce the theoretical findings but also offer actionable strategies for practitioners in dynamic and volatile markets.

References

Robert A Adams and John JFH Fournier. *Sobolev spaces*, volume 140. Academic press, 2003.

J. Adrien, A. Conze, P. Henry-Labordère, R. Louzir, R. Mahi, L. Mathieu, M. Messaoud, F. Monciaud, C. Muller, and A. Reghai. Vega knock-in times for local volatility models: An algorithmic differentiation approach. *SSRN*, 2022. URL `https://ssrn.com/abstract=4107770`.

Yacine Aït-Sahalia. Modeling jump diffusions for financial assets. *Annual Review of Financial Economics*, 1:379–410, 2007.

Rui Albuquerque, Pedro Pereira, and Bruno Ribeiro. Autocallable securities: Risk-return trade-offs and portfolio implications. *Journal of Banking & Finance*, 50:485–499, 2015.

Jesper Andreasen and Brian Huge. Volatility interpolation and the vix. *Applied Mathematical Finance*, 22(1):1–24, 2015.

S. Arora, S. Du, and R. Salakhutdinov. Uniform sampling and optimization: Theory meets practice. In *Proceedings of the Conference on Learning Theory (COLT)*, 2021.

Ole E Barndorff-Nielsen and Neil Shephard. Non-gaussian ornstein-uhlenbeck-based models and some of their uses in financial modeling. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):167–241, 2001.

David S Bates. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *Review of financial studies*, 9(1):69–107, 1996.

Eric Benhamou. *Monte Carlo Methods in Finance*. CRC Press, 2010.

Lorenzo Bergomi. *Stochastic Volatility Modeling*. Chapman and Hall/CRC, Boca Raton, 2016.

Peter Berkes and Naftali Tishby. Uniform convergence and generalization in neural networks. *Journal of Machine Learning Research*, 2019.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

Peter J Brockwell. The ornstein-uhlenbeck process. *Handbook of statistics*, 19:249–286, 2001.

Matt Cameron. Equity derivatives - dashing the uridashi dream. *Risk: Managing Risk in the World's Financial Markets*, 26(3):22–24, 2013. URL `https://www.risk.net/derivatives/structured-products/2256182/uridashi-losses-put-500-million-after-nikkei-rebounds`.

Lorenzo Capriotti. Fast greeks by algorithmic differentiation. *Risk Magazine*, 2010.

Michael Carver. Autocallable pricing and hedging in the presence of market dislocations. *Available at SSRN 3280808*, 2018.

Aleš Černý. *Mathematics of Financial Markets*. Springer Science & Business Media, 2009.

Chandan Chakraborty and Kalpana Mali. Performance evaluation of machine learning algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, 6(1):263–270, 2017.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. pages 785–794, 2016a.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. pages 785–794, 2016b.

Patrick Cheridito, Damir Filipović, and Marc Yor. Affine diffusion processes and applications in finance. *Finance and stochastics*, 9(3):337–378, 2005.

Ian Clark. Path-dependent volatility. *Risk Magazine*, 33(1), 2020.

Rama Cont and José da Fonseca. Modeling term structures of option prices. *Mathematical Finance*, 20(1):117–152, 2010.

Rama Cont and Peter Tankov. *Financial Modelling with Jump Processes*. Chapman and Hall/CRC, 2004.

Rama Cont and Peter Tankov. Model uncertainty and its impact on the pricing of derivative instruments. *Mathematical Finance*, 15(1):1–23, 2005.

Xiaofan Cui, Meng Wu, and Mariano Zeron. Pricing and hedging autocallable products via markov chain approximations. *arXiv preprint arXiv:2401.00895*, 2024.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals, and systems*, 2(4):303–314, 1989.

Chris Davis. Nikkei sell-off puts japanese autocall dealers on alert. 2018. URL `https://www.risk.net/derivatives/5441781/nikkei-sell-off-puts-japanese-autocall-dealers-on-alert?ref=search`.

Chris Davis. Regulatory crackdown puts korea autocalls in deep freeze. 2024. URL `https://www.risk.net/markets/7959196/regulatory-crackdown-puts-korea-autocalls-in-deep-freeze?ref=search`.

Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

Xiaotie Deng, Shuhao Li, and Zhaojun Li. Empirical analysis of autocallable products in the chinese market. *Emerging Markets Finance and Trade*, 51(sup1):S128–S140, 2015.

Thomas G Dietterich. Ensemble methods in machine learning. pages 1–15, 2000.

Matthew F Dixon, Diego Klabjan, and Zhiguang Bang. Machine learning in finance: From theory to practice. *Applied Stochastic Models in Business and Industry*, 36(1):3–27, 2020.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: unbiased boosting with categorical features. 2018.

Bruno Dupire. Pricing with a smile. *Risk*, 7(1):18–20, 1994.

G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep networks with uniform priors. *arXiv preprint*, 2017.

Ernst Eberlein. Application of generalized hyperbolic lévy motions to finance. pages 559–584, 2001.

Jean-Pierre Fouque, George Papanicolaou, and K Ronnie Sircar. *Derivatives in financial markets with stochastic volatility*. Cambridge university press, 2000.

Jean-Pierre Fouque, George Papanicolaou, and Ronnie Sircar. Multiscale stochastic volatility asymptotics. *Multiscale Modeling & Simulation*, 2(1):22–42, 2003.

Jean-Pierre Fouque, George Papanicolaou, K Ronnie Sircar, and Knut Solna. *Multiscale stochastic volatility for equity, interest-rate, and credit derivatives*. Cambridge university press, 2011.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Christian P Fries. Conditional monte carlo for barrier options and path-dependent derivatives. *arXiv preprint arXiv:1108.4069*, 2011.

Jim Gatheral. *The Volatility Surface: A Practitioner's Guide*. Wiley, 2011.

Mike Giles. Calculating sensitivities with algorithmic differentiation. pages 249–271, 2008.

Paul Glasserman. *Monte Carlo methods in financial engineering*. Springer Science & Business Media, 2013.

Gene H Golub and John H Welsch. Calculation of gauss quadrature rules. *Mathematics of computation*, 23(106):221–230, 1969.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Léonard Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Tree-based ensembles for tabular data: A survey. *arXiv preprint arXiv:2203.05552*, 2022.

H. Guennoun. Understanding autocalls: Real time vega map. *SSRN*, 2019. URL `https://ssrn.com/abstract=3387810`.

Julien Guyon and Pierre Henry-Labordère. Nonlinear option pricing. *Chapman and Hall/CRC Financial Mathematics Series*, 2013.

István Gyöngy. Mimicking the one-dimensional marginal distributions of processes having an itô differential. *Probability theory and related fields*, 71(4):501–516, 1986.

László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2002.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Pierre-Henri Henry-Labordère. Vega decomposition of exotics on vanillas: A monte-carlo approach. *SSRN*, 2013. URL `https://ssrn.com/abstract=2229990`.

Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6(2):327–343, 1993.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Chao Huang and Xiaolin Wang. A simple numerical method for pricing discretely monitored barrier options under local volatility models. *Applied Mathematics and Computation*, 348: 681–692, 2019.

Brian Huge and Antoine Savine. Differential machine learning. *ArXiv preprint ArXiv:2005.02347*, 2021.

John C Hull. *Options, futures, and other derivatives*. Pearson Education Limited, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. pages 448–456, 2015.

Ian T Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2 edition, 2002.

Anargyros Kadra, Szymon Grabowski, and Jaroslaw Jozefczyk. Comparing deep neural networks and gradient boosting machines in tabular data classification. *Applied Sciences*, 11 (15):6978, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

Young Shin Kim and Jinho Yoon. Pricing and hedging autocallable securities under stochastic volatility models. *Journal of Futures Markets*, 39(1):5–27, 2019a.

Young Shin Kim and Jinho Yoon. Recursive static replication of autocallable products. *Asia-Pacific Financial Markets*, 26:33–53, 2019b.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Marcelo C Klotzle and Ana CG Pinto. Autocallable structures: The vale s.a. case. *Revista de Administração de Empresas*, 52:554–565, 2012.

Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. 14(2):1137–1145, 1995.

Amélie Laurin. Natixis paie cher son offensive dans les dérivés actions. 2018. URL `https://www.agefi.fr/news/banque-assurance/natixis-paie-cher-son-offensive-dans-les-derives-actions`.

Chi-Ming Lee, Yu-Cheng Lin, and Chia-Hsun Shih. Pricing and performance analysis of autocallable structured products. *Review of Securities and Futures Markets*, 24(1):1–33, 2012.

Q. Li, S. Arora, and S. Du. On the benefits of uniform input distributions for neural network training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Martin Lorig, Marek Musiela, and Marek Rutkowski. Multi-asset option pricing under stochastic volatility: A spectral approach. *SIAM Journal on Financial Mathematics*, 4(1):121–165, 2013.

Daniel B Nelson. Arch models as diffusion approximations. *Journal of Econometrics*, 45(1-2): 7–38, 1990.

Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*, volume 63. SIAM, 1992.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5): 2295–2317, 2011.

T. Paletta and R. Tunaru. A bayesian view on autocallable pricing and risk management. *Journal of Derivatives Accounting*, 29(5):40–59, 2022.

Vladimir Piterbarg. Markovian projection method for volatility calibration, 2006.

Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, and Jascha Mitrovic. Theory of deep learning iii: The key role of depth. *arXiv preprint arXiv:1703.05075*, 2017.

Lutz Prechelt. Early stopping—but when? pages 55–72, 1998.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. 31:6638–6648, 2018.

A. Rahimi, B. Recht, S. Arora, and T. Zhang. Uniform sampling in high-dimensional spaces: Theory and applications. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

Mark Rubinstein. Pay now, choose later. *Journal of Financial and Quantitative analysis*, 22 (4):459–472, 1987.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Antoine Savine. Adjoint algorithmic differentiation for option pricing. *Journal of Computational Finance*, 21(4):1–33, 2018.

Aniket Sharma and Shreyas Nadkarni. Hedging autocallable products using distributional reinforcement learning. *arXiv preprint arXiv:2401.08207*, 2024.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *ArXiv [Cs.LG]*, 2021. URL `http://arxiv.org/abs/2106.03253`.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Vladimir N Vapnik. *Statistical learning theory*. Wiley, 1998.

Cédric Villani. *Optimal Transport: Old and New*. Springer Science & Business Media, 2008.

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

Mariano Zeron, Meng Wu, and Ignacio Ruiz. The frtb-ima computational challenge for equity autocallables. *ArXiv [q-Fin.RM]*, 2023. URL `http://arxiv.org/abs/2305.06215`.

X. Zhang and Y. Wang. Improved generalization with uniform sampling: A pac-bayes perspective. *Journal of Machine Learning Research*, 23(12):1341–1379, 2022.

Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.