Paper Title: Using PDE in Financial Modeling

Introduced by: PDE Profiteers team

2nd Year Comm. | Fall 2024

Team members:

NAME	SECTION	ID
Hussein Moustafa Amin	2	9220261
Dalia Mohamed Soliman	2	9220285
Serag Khaled Ahmed	2	9221268
Ammar Ayed Abdon	2	9221448
Omar Ahmed Ragab	3	9220513
Omar Ahmed Mohamed	3	9220518
Omar Ayman Amin	3	9220528
Mohamed Ahmed Abd-El Hakam	3	9220647
Mohand Hassan Aly	4	9221293

Instructor: Dr. Samah El-Shafiey

Abstract

According to CNBC, options trading activity reached a new high in 2021 with an average of 39 million daily traded option contracts, up 35% from the previous year. Despite this growth, one of the biggest challenges in the option market industry is predicting whether an option's price will rise or fall. Accurate and scientifically-backed predictions are crucial for investors, and mathematical modelling can provide a solution. In our study, we compared three methods for predicting call option prices: analytical and numerical solutions for the Black-Scholes equation, and an Artificial Intelligence model. Using data collected from YAHOO Finance for four of the biggest companies in the stock market, we aimed to find a model with an acceptable Mean Absolute Error (MAE). Our results showed that the AI model was the most reliable, with the lowest MAE of the three approaches. While the numerical solution was also reliable, it did not match the accuracy of the AI model. We then integrated the models into a user-friendly web application to make it easier for people to use.

Table of Contents

Table of Equations

Table of Figures

List of Tables

I. Introduction

According to a report by Statistica [1], the global financial market, which comprises various asset types, was estimated to be worth approximately $294 trillion in 2020. The Options Clearing Corporation (OCC) [2] reported that in 2020, the trading volume of options in the US alone exceeded 7 billion contracts. As of Q2 2021, the global hedge fund industry was managing assets worth about $3.6 trillion. To achieve their investment goals, many hedge funds use options. But what are "options"?

Options are financial derivatives that give the holder the right, but not the obligation, to buy or sell an underlying asset at a predetermined price (strike price) before or at the option's expiration date. The options market is widely used by individuals and institutions alike. Investing in options provides companies with additional income, which encourages more users to invest in the company. This way of financing companies in the option market is considered a way for users to contribute and help companies build a good reputation, which is an effective branding strategy to increase awareness about their activities.

Forecasting or modeling the options markets could have a significant impact on the economic growth of states. By having a robust option market, states can attract investors to invest in their stocks, which will in turn boost the economy of the state and create new opportunities for businesses to flourish. This, in turn, will lead to an increase in the GDP and employment rate. Egypt, in particular, faces a shortage of foreign currency and investors, so developing a strong stock market could play a crucial role in increasing foreign currency reserves.

It is not safe to say that the price market is 100% beneficial all the time. Prices frequently change, and this can seem quite random. Despite these fluctuations, many people still choose to invest in this field, hoping to profit from the changes and take advantage of the high rewards that come with this risky decision. Since the creation of option markets and banking, Data Scientists and Economists have developed models to predict option market flow and identify patterns. These models are designed to be powerful tools for making informed decisions in the options market. Our project is to develop a tool that will help people calculate their risks and make highly accurate predictions about price changes over a selected period. This project will be beneficial for both investors and companies, as it will help them make informed decisions about whether to buy or sell their options at the appropriate time.

II. Problem definition

Working in the options market requires accurate predictions of the option market's future movements and global financial trends. To maximize the benefits, you must make well-informed decisions regarding buying and selling options. However, the complexity of this process often leads to financial losses for many individuals in the market. Our project provides investors and business owners with a strong scientific and historical background for successful stock market engagement.

The behavior of market prices can be considered as a stochastic process known as "Brownian Motion" or Weiner Process with drift. Utilizing a stochastic process provides the benefit of accommodating the random probability distribution for predicting future values, which can be quite challenging. Given these reasons, the option market can be categorized as a Stochastic Process that meets the requirements of a Stochastic Differential Equation (SDE).

• The Black-Scholes Model

There are many used models in prediction of the option prices, one of the most known and used is the Black-Scholes Model which was developed by Economists Fischer Black and Myron Scholes in 1968. this model is a mathematical model (PDE) for the dynamics of a financial market containing derivative investment instruments using various underlying assumptions. The PDE can be solved, and a formula is deduced to give the theoretical estimate of the price of European-style options.

In the work of Black and Scholes the following seven assumptions are made:

- The asset price has properties of a Brownian motion with μ and σ as constants.
- There are no transaction costs or taxes.
- All securities are perfectly divisible.
- There is no dividend during the life of the derivatives.
- There are no riskless arbitrage opportunities.
- The security trading is continuous.
- The option is exercised at the time of maturity for both call and put options.

Indeed, the Black-Scholes model can be very efficient, but it suffers a few drawbacks. These drawbacks are naturally emanating from its strict design assumptions. Among them is the assumption that financial assets' returns follow a normal distribution. In the earlier days, the classical Black-Scholes approach was regarded as a very efficient and robust approach, and as such, it received enormous applications in pricing several financial assets and contracts.

Many professionals utilize a modified version of the Black-Scholes model to evaluate and safeguard European-style options. Regrettably, the model remains somewhat intricate. However, it can be simplified by negating the impact of the time value of money on the exercise price and emphasizing the insurance aspect of options.

III. Literature Review

Upon the mentioned case study, we have come to the conclusion that using modern technologies to develop new techniques for evaluating the Black-Scholes mathematical model can revolutionize the field of the options market. This is especially true when it comes to the dynamics of a financial market containing derivative investment instruments using various underlying assumptions. By doing so, it becomes possible to predict accurate and precise options price movements. As we will see in the following examples, the different models that have been developed before got impressive results, but there's always room for improvement!

1. Predicting the stock price of frontier markets using machine learning and modified Black–Scholes Option pricing model [3].

In this paper, the authors presented the analysis of the prediction of option prices using modified Black-Scholes Option Pricing Model (BSOPM), machine learning models and representation of the model with a quantum physics perspective.

Firstly, the authors discussed the mathematical model of the standard black-Scholes equation and how R.C. Merton later extended the model to account for other parameters. This Merton black-Scholes model can be described with the following equation:

Call option (c) formula:

$$c = Se^{(-qT)}N(d1) - Xe^{(-rT)}\,N(d2) \qquad \textit{Equation 1}$$

Put option (p) formula:

$$p = Xe^{(-rT)}N(-d2) - Se^{(-qT)}N(-d1) \qquad \textit{Equation 2}$$

Where S = Stock Price, X = Strike Price, r = Risk free interest rate, T = Time to expiration in years, σ = Volatility of the relative price change of the underlying stock price, N (x) = Cumulative normal distribution function

$$d1, d2 = \frac{\left(\ln\left(\frac{S}{X}\right) + \left(r - q \pm \frac{\sigma^2}{2}\right)T\right)}{\sigma\sqrt{T}} \qquad \textit{Equation 3}$$

Then, the authors explain the key elements of the Black-Scholes model and how the model parameters are represented in the equation and how can we give input for certain parameters. The authors outlined the need to calculate certain key parameters for the model as those parameters are varying with other parameters, and the absence of standardization as the model seems to be adapted to certain cases.

After that, they discussed their machine learning approach in the prediction of the option prices. They used two machine learning algorithms: decision tree and neural networks. These two algorithms have shown a big improvement in prediction over the standard black-Scholes model. Also, to increase the accuracy and the performance of the model ensemble learning is used. It involves combination of multiple machine learning algorithms to work together.

Finally, the paper introduces the modification of Black-Scholes model from quantum physics perspective. They made four main modifications in the model which are: random component, change in time, implied volatility, and wick rotation. A random noise is added to the model which implies the randomness of the market. Time is no longer considered continuous and constant, but it is a function of many parameters like strike price. implied volatility is now a variable varying with time and varying at each point. The wick rotation adds the concept of imaginary time as the time might not be real and continuous.

$$\pi = \pi(\tau, x), \; i\frac{\partial \pi}{\partial \tau} = -\frac{1}{2}\sigma\hat{}2\frac{\partial^2 \pi}{\partial x^2}$$ Equation 4

where $\pi = \pi(\tau, x)$ represents the option price as a function of time and asset price [3].

2. Performance of the Heston's Stochastic Volatility Model: A Study in Indian Index Options Market [4].

This paper compares the black sholes Stochastic Volatility model and the Heston Stochastic Volatility model. This paper is distinguished on a one-day ahead out-of-sample using data from CNX Nifty index, which represents the highest 50 Indian companies according to the National Stock Exchange. This study showed a higher quality of analysis using SVH (stochastic volatility of Hoston) than BS (Black Scholes model). This procedure will be explained in the following paragraph:

SVH is the first stochastic volatility model to work with a semi-closed form solution, which is an important factor in the popularity of this method. This model considers some issues in the BS model which are: constant interest rates and neglecting extraordinary jumps to the return process. The model added features to overcome these disadvantages of the BS model, but it came with a higher complexity than the BS model. The model is based on the two following PDEs:

$$dS(t) = \mu S(t)dt + \sqrt{v(t)}S(t)dW1(t)$$
$$dv(t) = \kappa(\theta - v(t))dt + \sigma\sqrt{v(t)}dW2(t)$$ Equation 5

Where: S(t) Spot price at time t, μ is the expected return, v(t) is the stochastic volatility, $W1$ (t) and $W2(t)$ Wiener processes, κ Strike Price, θ Long run variance, Volatility of variance, ρ Correlation parameter, μ Drift of the underlying and λ Volatility risk.

The research uses tow methods to measure the performance of each model which are:

Liquidity-weighted Mean Percentage Error (MPE)

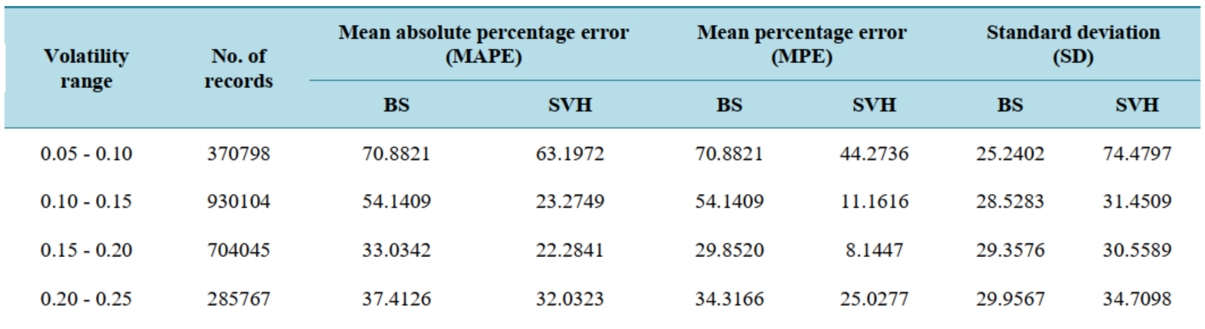
$$\text{MPE} = 100 * \frac{\sum_{i=1}^{N} Q_i * (A_i - C_i)/A_i}{\sum_{i=1}^{N} Q_i}$$ Equation 6

Liquidity-weighted Mean Absolute Percentage Error (MAPE)

$$MAPE = 100 * \frac{\sum_{i=1}^{N} Q_i * (|A_i - C_i| / A_i)}{\sum_{i=1}^{N} Q_i}$$ *Equation 7*

Where: the C_i is the calculates prices and A_i is the actual prices of the i th option, Q_i is the the traded quantity for i th option and N is the total options we have.

As shown in Table 1, the two measures MPE and MAPE with the standard deviation of each of them in mispricing across volatility range. These results showed the higher performance of SVH and the less percentage of error in all volatility ranges which proves the demonstration of SVH model on BS in Indian stock market.



Volatility range	No. of records	Mean absolute percentage error (MAPE)		Mean percentage error (MPE)		Standard deviation (SD)	
		BS	SVH	BS	SVH	BS	SVH
0.05 - 0.10	370798	70.8821	63.1972	70.8821	44.2736	25.2402	74.4797
0.10 - 0.15	930104	54.1409	23.2749	54.1409	11.1616	28.5283	31.4509
0.15 - 0.20	704045	33.0342	22.2841	29.8520	8.1447	29.3576	30.5589
0.20 - 0.25	285767	37.4126	32.0323	34.3166	25.0277	29.9567	34.7098

Table 1 shows Liquidity-weighted performance metrics for mispricing across volatility range for call options [4]

The methods of measuring that the paper uses verify the higher performance of SVH than the BS model which is shown in lower percent of error in Liquidity-weighted Mean Percentage Error (MPE) and Liquidity-weighted Mean Absolute Percentage Error (MAPE). The advantages of SVH over BS are shown in some subgroups moneyness, volatility, and time-to-expiration. These subgroup analysis results where distinctly true and very matched in normal volatility regimes but when testing the low volatility and deep OTM subgroups the BS was far enough from accurate results and SVH was much better. For these reasons, the paper concludes that the SVH is much better way in dealing with and analyze the pricing dynamics [4].

3. Stock Price Model-Based Stochastic Pantograph Differential Equation [5].

This paper aims to improve the evaluation process and prediction accuracy of option prices by introducing a new model that utilizes a stochastic pantograph differential equation (SPDE) with a variable delay. The proposed method is expected to provide a more precise prediction of option prices. This model is a dynamic model which is based on non-constant volatility, which leads to better option prices prediction, unlike the Black-Scholes model which is based on constant volatility.

SPDE was used because of its properties like past dependence and having a variable delay time. Using variable delay time enhanced the model as the data in the recent past are more effective in predicting prices than the oldest one.

Numerical experiments utilizing real financial data as it shown in Figure 1 & Figure 2 have revealed that the proposed model outperforms both non-linear models with constant delay and the classical Black and Scholes model in terms of prediction accuracy [5].

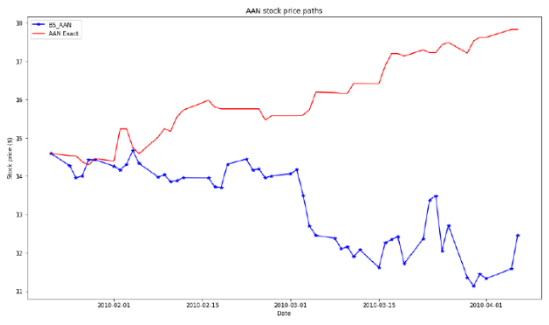


Figure 1 shows the BS model over 50 days [5]

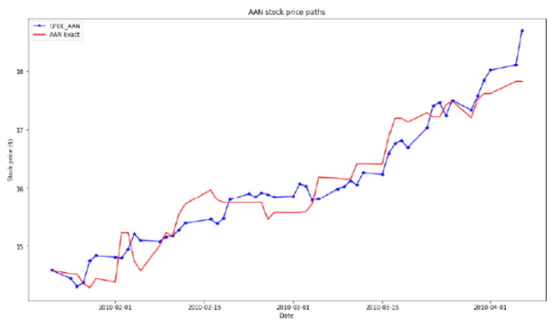


Figure 2 shows the SPDE model over 50 days [5]

4. Estimation of stock prices using Black-Scholes partial differential equation for put option [6].

This paper mainly tackles one of the numerical solutions to the Black-Scholes partial differential equation, the Crack-Nicholson finite difference method for the variation of the European put option.

In the process of attempting to solve the PDE, quite a few assumptions were made. Some of which are: The stock prices follow a geometric Brownian motion, the security trading is continuous, and that there are no riskless arbitrage opportunities.

The analytic formula for the prices of European put can be found as shown in equation 2. And according to Rinalini (2006), the derivation of BS PDE is based on ItÔ process with an assumption that the stock prices follow a geometric Brownian motion, i.e.

$$\partial S = \mu S \partial t + \sigma S \partial x \qquad \textit{Equation 8}$$

Where S is the stock price, µ is the drift, σ is the volatility of underlying asset and dx is a wiener process.

After applying the crack-Nicholson model 1:

$$a_i V_{i-1}^{j} + (1 + b_i) V_i^{j} - c_i V_{i+1}^{j} = a_i V_{i-1}^{j+1} + (1 - b_i) V_i^{j+1} + c_i V_{i+1}^{j+1} \qquad \textit{Equation 9}$$

Where

$$a_i = \frac{\Delta t}{4}(\sigma^2 S_i^2 - rS_i), b_i = -\frac{\Delta t}{2}(\sigma^2 S_i^2 - r) \; and \; c_i = \frac{\Delta t}{4}(\sigma^2 S_i^2 + rS_i) \qquad \textit{Equation 10}$$

a_i, b_i, c_i are random variables, i=0,1,……,M

The CN discretization outcome in tridiagonal matrix which is solvable at each time step.

The Crack-Nicholson model shown in equations 9 and 10 is applied to the final PDE which was derived after applying all the assumptions, and the results were as follows:

- The model yielded some considerably accurate results when the σ* parameter was set to lower values $\sigma < 0.5$.
- The model yielded more inaccurate results as σ increased, making the error much more noticeable and pronounced.

To conclude, The Crack-Nicholson model can be used to estimate the option prices in the right circumstances i.e., an application with as close properties as the assumptions made and σ cannot be greater than 0.5 [6].

5. Numerical solving of the generalized Black-Scholes differential equation using Laguerre neural network [7].

In this paper, the Laguerre neural network was proposed as a novel numerical algorithm with three layers of neurons for solving Black-Scholes (BS) equations, with the option price as the only output layer. Laguerre functions are used as the activation function in the hidden layer, and the BS equation and boundary conditions are set as a penalty function. The training points are uniformly selected in the domain, and the improved extreme learning machine algorithm is used to optimize the network connection weights.

The Laguerre neural network as shown in Figure 3 is established as a special case of a single hidden layer feedforward neural network model, with two input neurons for stock price and expiration time. When the option contains multiple stocks, the information can be processed separately by adding neurons to the input layer. The only output layer neuron displays the numerical solution of the BSM equation.

Extreme learning machine algorithm:

The single hidden layer feedforward neural network (SLFN) model used in this study is a popular neural network structure with a single hidden layer, making it easier for training. However, back-propagation algorithms can't avoid gradient explosion problems, dispersion, and overfitting. Huang proposed an extreme learning machine (ELM) algorithm for training SLFN without iteration, which was perfected by Huang et al. for classification and regression tasks.

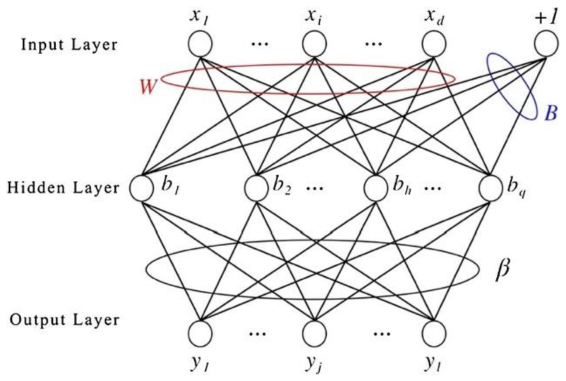


Figure 3 shows the topology structure of a single hidden layer feedforward neural network [7]

At the end, three numerical experiments were conducted to calculate the numerical solutions of BS equations for European options and generalized option pricing models and were compared to existing algorithms like the finite element method and radial basis function neural network. The Laguerre neural network obtained higher accuracy and smaller errors, demonstrating its feasibility and superiority [7].

To summarize, this section has reviewed recent research on how the Black-Scholes equation can be applied to predicting option prices. As shown in Table 2, some studies have attempted to modify the equation directly, while others have focused on comparing commonly used approaches or testing new numerical techniques. In our study, we solved the equation analytically and then numerically using finite difference methods. In addition, we developed a new AI model that uses a neural network to solve the equation, as discussed in the next section. Our objective is to evaluate the three alternatives and determine which is the most effective.

Points of comparison	*Predicting the stock price of frontier markets using machine learning and modified Black–Scholes Option pricing model*	*Performance of the Heston's Stochastic Volatility Model: A Study in Indian Index Options Market*	*Stock Price Model-Based Stochastic Pantograph Differential Equation*	*Estimation of stock prices using Black-Scholes partial differential equation for put option*	*Numerical solving of the generalized Black-Scholes differential equation using Laguerre neural network*
Modify BS	Yes	NO	NO	NO	NO
Use ML	Yes	NO	NO	NO	YES
Compare different methods	NO	YES	YES	NO	YES
Find new method for solving	NO	NO	NO	YES	YES

Table 2 shows comparison between different studies.

IV. Mathematical Modeling

The Black-Scholes Equation is as follows.

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}=rV-rS\frac{\partial V}{\partial S}$$ *Equation 11*

V: The price of the option as a function of time (t) and the price of the underlying asset (S).

T: Time to maturity (expiration) of the option.

S: Price of the underlying asset (e.g., stock).

σ: Volatility of the underlying asset's return.

r: Risk-free interest rate.

t: time.

Payoff function ($K(S_T)$): K is a function that takes the stock price at the expiration date and returns its payoff.

Forward VS Backward PDEs

The most frequent type of PDE in financial problems is the parabolic equation. A parabolic equation for a function V (S, t) is a specific relationship between V and its partial derivatives with respect to the independent variables S (which is second order) and t (first order).

To understand how we got the boundary conditions for our approach, first, let's take an example the heat equation which is equivalent to Black sholes with some differences.

The heat equation describes the change of temperature with respect to space and time which is equivalent to the motion that is described by black sholes.

The heat equation, which is first order in t and second order in x, we need one condition in t and two conditions in x. We have u (x, 0) which is the temperature at t = 0, on any point on the rod and we need to know two other conditions u (0, t) and u (L, t) where L is the rod length. All these conditions are required to get a unique solution for our equation.

For Black-Scholes PDE, which is a parabolic partial differential equation, two boundary conditions and one initial condition are needed to extract an exact solution for the PDE. However, the Black-Scholes PDE can be defined as a backward parabolic partial differential equation: A parabolic PDE is considered backward when the terms $\frac{\partial V}{\partial t}$ and $\frac{\partial^2 V}{\partial S^2}$ are on the same side of the equation with the same signs e.g.,

$$\frac{\partial V}{\partial t}+\frac{\partial^2 V}{\partial S^2}=0$$ *Equation 12*

When attempting to solve for Black-Scholes, we encounter an issue. Black-Scholes is a backward parabolic equation, whereas heat is a forward parabolic equation. This presents a challenge in obtaining boundary conditions for Black-Scholes. Instead of an initial condition V(S,0), a final (terminal) condition V(S, T) must be obtained and the equation must be solved backward for t<T.

Boundary Conditions

For the moment, we focus our attention on a vanilla European call c (S, t) with a strike price K and expiry date T.

By definition, S follows a stochastic Brownian motion; hence, if *S* is ever zero, then *dS* is also zero, and therefore *S* can never change. Since if *S* = 0 at expiry, the payoff will be zero. Thus, the call option is worthless on *S* = 0 even if there is a long time to expiry. Hence in *S* = 0, we have

$$c(0,t) = 0 \text{, for all } t \geq 0 \qquad \textit{Equation 13}$$

As S → ∞, it becomes more likely that the option will be exercised, and the magnitude of the exercise price becomes less and less important. Thus, as S → ∞ the value of the option becomes that of the asset minus the exercise price we need to pay to exchange for the asset. Hence, we have for all *t* > 0.

$$c(S,t) \sim S - Ke^{-(rT-t)} \text{, as S } \to \infty \qquad \textit{Equation 14}$$

We take it to mean that,

$$\lim_{S\to\infty} \frac{c(S,t)}{S-Ke^{-r(T-t)}} = 1$$

The final condition of a call is just its payoff at T:

$$c(S,T) = \max(S-K,0) \text{, for all S} \geq 0 \qquad \textit{Equation 15}$$

dS is the infinitesimal change in *S(t)* found in the Brownian motion equation:

$$dS(t) = \mu S(t)dt + \sigma S(t)dX(t) \qquad \textit{Equation 16}$$

By using the Black-Scholes equation as in equation 11 along with boundary conditions in equations *12 & 14*, we can get the exact value of European call option.

For Put option call conditions will be stated as follows:
The terminal condition:

$p(S,T) = \max(K - S, 0)$ for all S ≥ 0

For the boundary conditions, it's already mentioned that if S is zero it must remain zero. In this case, the final payoff is E also known as (K(S_{T}))

So, for S=0 $p(0,t) = Ee^{-rt}$, for all t ≥ 0

And as S → ∞ for t > 0 $p(S,t) \to 0$, as S → ∞

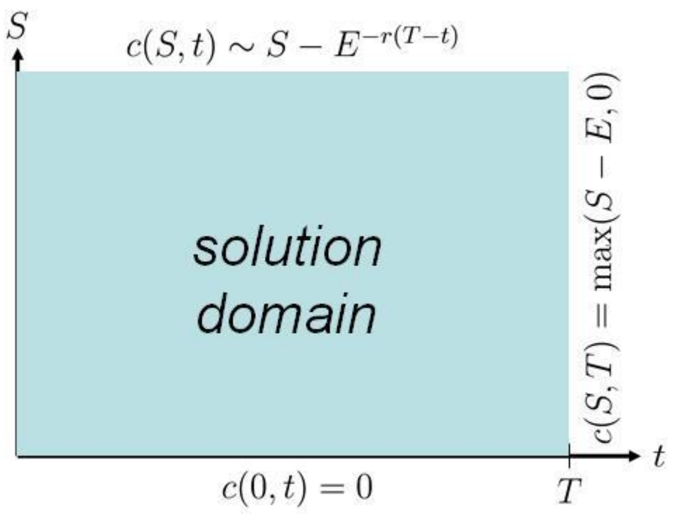


Figure 4 shows the solution domain for S

1) Analytical Solution for the Black-Scholes PDE

Before we begin, we need to be familiar with certain concepts and conditions for our solution. Firstly, the Black-Scholes equation is usually solved using numerical methods like the finite difference method. However, in case of the European call option, it is possible to solve for a specific formula. Secondly, we need to understand what the standard convolution method is. It is a mathematical technique that is used to solve partial differential equations (PDEs), such as the heat (diffusion) equation. The heat equation describes how a quantity, such as temperature or concentration, changes over time and space due to diffusion.

Since the stock motion follows a geometric Brownian motion similar to the particles in heat transfer, we can use the heat equation to solve it using the convolution method to get an exact solution. However, we will encounter a problem, which is that the heat equation is a forward parabolic equation while Black-Scholes is a backward parabolic equation. Therefore, our approach is to convert our equation to a forward parabolic heat equation and solve it using convolution method to get the exact solution.

The solution of the PDE gives the value of the option at any earlier time, $\mathbb{E}[\max\{S - K, 0\}]$. To solve the PDE we recognize that it is a Cauchy–Euler equation which can be transformed into a diffusion equation by introducing the change-of-variable transformation.

$$\tau = T - t$$

$$u = Ce^{r\tau}$$

$$x = \ln\left(\frac{S}{K}\right) + \left(r - \frac{1}{2}\sigma^2\right)\tau$$

Equation 17

Then, the Black-Scholes equation becomes a diffusion equation:

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sigma^2 \frac{\partial^2 u}{\partial x^2}$$ *Equation 18*

Now, the equation became a forward parabolic PDE i.e., a heat equation, so we have to convert the terminal condition into initial condition.

Therefore, the terminal condition $c(S,T) = \max\{S - T, 0\}$ becomes as shown in equation 19:

$$u(x,0) = K\left(e^{\max\{x,0\}} - 1\right) = K(e^x - 1)H(x)$$ *Equation 19*

Where H(x) is the Heaviside Step function. It is used to enforce the boundary condition in the S, t coordinate system that requires when t = T,

$$C(S,T) = 0, \ \forall \, S < K$$ *Equation 20*

Using the standard convolution method for solving heat (diffusion) equation given an initial value function u(x,0), we have

$$u(x,\tau) = \frac{1}{\sigma\sqrt{2\pi\tau}} \int_{-\infty}^{\infty} u_0(y) e^{\left(-\frac{(x-y)^2}{2\sigma^2\tau}\right)} dy$$ *Equation 21*

which, after some manipulation, yields:

$$u(x,\tau) = K \, e^{\left(x+\frac{1}{2}\sigma^2\tau\right)} N(d_+) - KN(d_-)$$

Equation 22

Where N() is the standard normal cumulative distribution function and

$$d_+ = \frac{1}{\sigma\sqrt{\tau}}\left[\left(x + \frac{1}{2}\sigma\tau\right) + \frac{1}{2}\sigma^2\tau\right]$$

$$d_- = \frac{1}{\sigma\sqrt{\tau}}\left[\left(x + \frac{1}{2}\sigma\tau\right) - \frac{1}{2}\sigma^2\tau\right]$$ *Equation 23*

Reverting u, x, τ to the original set of variables yields the above stated solution to the Black–Scholes equation.

2) Numerical Solution for the Black-Scholes PDE

Solving Black Scholes using finite difference method:

$$\frac{\partial V(t,s)}{\partial t} + rs\frac{\partial V(\dot{t},s)}{\partial s} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 V(t,s)}{\partial s^2} - rV(t,s) = 0.$$

Equation 24

The previous is the well-known Black Scholes equation we decided to use a fully implicit method, because it is an efficient and stable method and can also be used to do the pricing of different types of derivatives (barrier options, digital options, American options, etc) with little to no modifications.

To simplify the computations by passing the log variable $x = \log{(s)}$. So:

$$s\frac{\partial}{\partial s} = \frac{\partial}{\partial x}\ , \qquad s^2\frac{\partial^2}{\partial s^2} = \frac{\partial^2}{\partial x^2} - \frac{\partial}{\partial x}.$$

Equation 25

For an option with strike K and maturity T, the boundary conditions are:

CALL:

- Terminal:

$$V(T,x) = \max(e^x - K, 0),$$

Equation 26

- Lateral:

$$V(t,x) \underset{x\to-\infty}{=} 0 \text{ and } V(t,x) \underset{x\to\infty}{\sim} e^x - Ke^{-r(T-t)}.$$

Equation 27

We developed our solution to predict the call price and the user will decide if this company is worth investing in or not.

Derivative approximation

Finite difference methods are a technique for obtaining numerical solutions of PDEs. The idea underlying finite-difference methods is to replace the partial derivatives occurring in the PDE by finite difference approximations. If we assume that V is a smooth function, we can use the Taylor series expansion near the point of interest. For a $\Delta t > 0$ we can write

$$V(t+\Delta t, x) \approx V(t,x) + \frac{\partial V(t,x)}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 V(t,x)}{\partial t^2}\Delta t^2 + \mathcal{O}(\Delta t^3).$$
$$V(t-\Delta t, x) \approx V(t,x) - \frac{\partial V(t,x)}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 V(t,x)}{\partial t^2}\Delta t^2 + \mathcal{O}(\Delta t^3)$$

Equation 28

An analogous approximation can be done for $V(t, x+\Delta x)$ with $\Delta x > 0$. If we want to approximate the partial derivative with respect to time, we obtain the following finite difference approximation.

$$\frac{\partial V(t,x)}{\partial t} \approx \frac{V(t+\Delta t, x) - V(t,x)}{\Delta t} + \mathcal{O}(\Delta t)$$

Equation 29

Also called forward difference since the differencing is in the forward t direction. We can also consider the backward difference.

$$\frac{\partial V(t,x)}{\partial t} \approx \frac{V(t,x) - V(t-\Delta t, x)}{\Delta t} + \mathcal{O}(\Delta t)$$

Equation 30

and the central difference

$$\frac{\partial V(t,x)}{\partial t} \approx \frac{V(t+\Delta t, x) - V(t-\Delta t, x)}{2\Delta t} + \mathcal{O}(\Delta t^2)$$

Equation 31

The use of the forward and backward difference approximation leads to the explicit and implicit finite difference schemes respectively. The central difference is not used for the time variable because it leads to bad numerical schemes. But it is common to use it for the space variable.

For second order derivatives, such as $\partial^2 V(t,x)/\partial x^2$, we can use the symmetric central difference approximation for a $\Delta x > 0$:

$$\frac{\partial^2 V(t,x)}{\partial x^2} \approx \frac{V(t, x+\Delta x) + V(t, x-\Delta x) - 2V(t,x)}{\Delta x^2} + \mathcal{O}(\Delta x^2)$$

Equation 32

Implicit discretization:

First, we must restrict the theoretical infinite domain to the finite region $[t_0, T] \times [A_1, A_2]$, with $A_1 < A_2$.

The next step is to replace $[t_0, T] \times [A_1, A_2]it$ by a discrete grid:

- For $n = 0,1, \dots N \in \mathbb{N}$, define the discrete time step $\Delta t = \frac{T - t_0}{N}$ such that $t_n = t_0 + n\Delta t$.
- For $i = 0,1, \dots M \in \mathbb{N}$, define the discrete space step $\Delta x = \frac{A_2 - A_1}{M}$ such that $x_i = A_1 + i\Delta x$.

The grid is divided into equally spaced nodes of distance Δxon the x-axis, and of distance Δton the t-axis. The mesh points have the form $(t_0 + n\Delta t, A_1 + i\Delta x)$. At this point, we concern ourselves only with the values of $V(t, x)$ the mesh nodes.

$$V(t_0 + n\Delta t, A_1 + i\Delta x) = V_i^n$$

Equation 33

We apply the backward discretization (*implicit scheme*) for the time derivative and a central discretization for the first-order space derivative. We are interested in the value of V at time t_0. We know the values V^N corresponding to the terminal conditions. The algorithm consists of finding the values V^n given the knowledge of the values V^{n+1}.

The discretized equation becomes:

$$\frac{V_i^{n+1} - V_i^n}{\Delta t} + \left(r - \frac{1}{2}\sigma^2\right)\frac{V_{i+1}^n - V_{i-1}^n}{2\Delta x} + \frac{1}{2}\sigma^2\frac{V_{i+1}^n + V_{i-1}^n - 2V_i^n}{\Delta x^2} - rV_i^n = 0.$$

Equation 34

Rearranging the terms:

$$\begin{aligned} V_i^{n+1} = {} & V_i^n\left(1 + r\Delta t + \sigma^2\frac{\Delta t}{\Delta x^2}\right) \\ & + V_{i+1}^n\left(-\left(r - \frac{1}{2}\sigma^2\right)\frac{\Delta t}{2\Delta x} - \frac{1}{2}\sigma^2\frac{\Delta t}{\Delta x^2}\right) \\ & + V_{i-1}^n\left(\left(r - \frac{1}{2}\sigma^2\right)\frac{\Delta t}{2\Delta x} - \frac{1}{2}\sigma^2\frac{\Delta t}{\Delta x^2}\right). \end{aligned}$$

Equation 35

We can rename the coefficients such that:

$$V_i^{n+1} = aV_{i-1}^n + bV_i^n + cV_{i+1}^n,$$

Equation 36

and write it in matrix form:

$$\begin{pmatrix} V_1^{n+1} \\ V_2^{n+1} \\ \vdots \\ V_{M-2}^{n+1} \\ V_{M-1}^{n+1} \end{pmatrix} = \underbrace{\begin{pmatrix} b & c & 0 & \cdots & 0 \\ a & b & c & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & a & b & c \\ 0 & 0 & 0 & a & b \end{pmatrix}}_{\mathcal{D}} \cdot \begin{pmatrix} V_1^{n} \\ V_2^{n} \\ \vdots \\ V_{M-2}^{n} \\ V_{M-1}^{n} \end{pmatrix} + \underbrace{\begin{pmatrix} aV_0^n \\ 0 \\ \vdots \\ cV_M^n \end{pmatrix}}_{\text{B (boundary terms)}}$$

The system

$$V^{n+1} = \mathcal{D}V^n + B$$

Equation 37

can be solved easily V^n by inverting the matrix $\mathcal{D}$, and then we iterate through time steps to get values for each time and space.

3) AI Solution for the Black-Scholes PDE

The solution presented in this section is based on building a deep-learning model. This model is trained with data from many inputs and corresponding solutions to those inputs. Those solutions are obtained by the analytical method of solution for the black Scholes equation mentioned in Analytical Solution for the Black-Scholes PDE so that this model can then predict the solution on its own.

This model is a sequential model that allows you to build a neural network by stacking layers on top of each other.

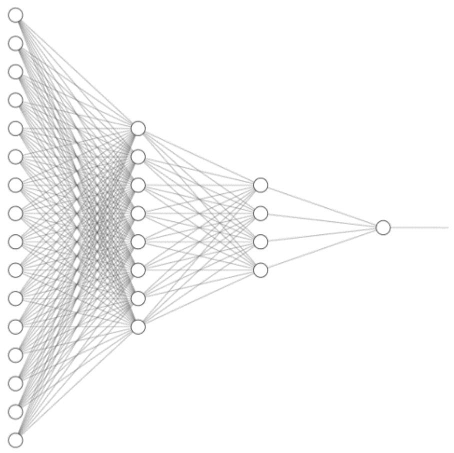


Figure 5 shows the hidden layers in the AI model.

The structure of the neural network used in this work has five layers of neurons four hidden layers and one output layer as shown in Figure 2.

1- Hidden layer 1

Input: adds a fully connected (Dense) layer to the model. It has nodes number of neurons it expects input data.

The first layer having an exact of 128 nodes.

Activation: adds the Rectified Linear Unit (ReLU) activation function to the layer.

Rectified Linear Unit (ReLU) function is a commonly used activation function in neural networks. It is a simple and computationally efficient non-linear function that introduces non-linearity into the network.

The ReLU function is defined as follows:

$$f(x) = \max(0, x)$$

In other words, the ReLU function returns the input value if it is positive or zero, and it returns zero for any negative input. Mathematically, this can be expressed as:

$$f(x) = \{x, \quad if \quad x > 0$$

$$0 \quad if \quad x \leq 0\}$$

2- Hidden layer 2, 3, and 4

Like Layer 1, three more fully connected layers with nodes neurons are added to the model, along with ReLU activation and dropout layers.

These layers would have 64,32,8 nodes respectively.

3- Output layer

This layer is responsible for producing the final output of the model.

Activation: adds the Linear Unit (LU) activation function to the output layer.

Model Compilation: It specifies the loss function (mse stands for mean squared error) used for training and evaluation, the optimizer (which can be specified as an argument to the function), and the metrics to be evaluated during training and testing (in this case, accuracy).

The **MSE** loss can be mathematically represented as follows:

$$mse = \left(\frac{1}{N}\right) * \sum (y\_true - y\_pred)^2$$

where:

N is the number of samples or data points.

y_true represents the true or target values.

y_pred represents the predicted value.

4) User-Friendly Web Application

Combining the three solutions, we designed a web application integrating the three solutions in a user-friendly way to make our theoretical work tangible to the user. We used HTML programming language to code the main structure of the website interface and applied CSS to style it. Our goal was to design a user interface that is not only easy to use but also visually appealing and interactive, following the latest UI/UX principles as shown in Figure 6.

To input data, we added input fields, and to visualize the output, we used a line graph that displays detailed data on each point. This feature enables users to track any point on the graph easily.

To further enhance the functionality of our application, we have made the graph grid incredibly precise by displaying more detailed results in a shorter time frame. For instance, we can display data in days instead of years, which enables users to monitor changes and trends with even greater precision.

We have ensured the security of our users by providing personal accounts to track their history search, projections they may have considered, and many more features.

We have also added several interactive features, such as the ability to zoom in on specific time periods or data points and to add more data points to display more nuanced changes and trends. Additionally, we have incorporated hover-over tooltips to provide users with more detailed information on specific data points. These features make our application incredibly powerful, providing users with a more thorough understanding of the data they are analyzing.

In the future, we plan to partner with established mobile applications in the options and stock markets such as Thunder and Binance. Also, we will add 3D visualization for the output grids in case of considering more than one variable like volatility changes due to sudden changes like wars and pandemics.

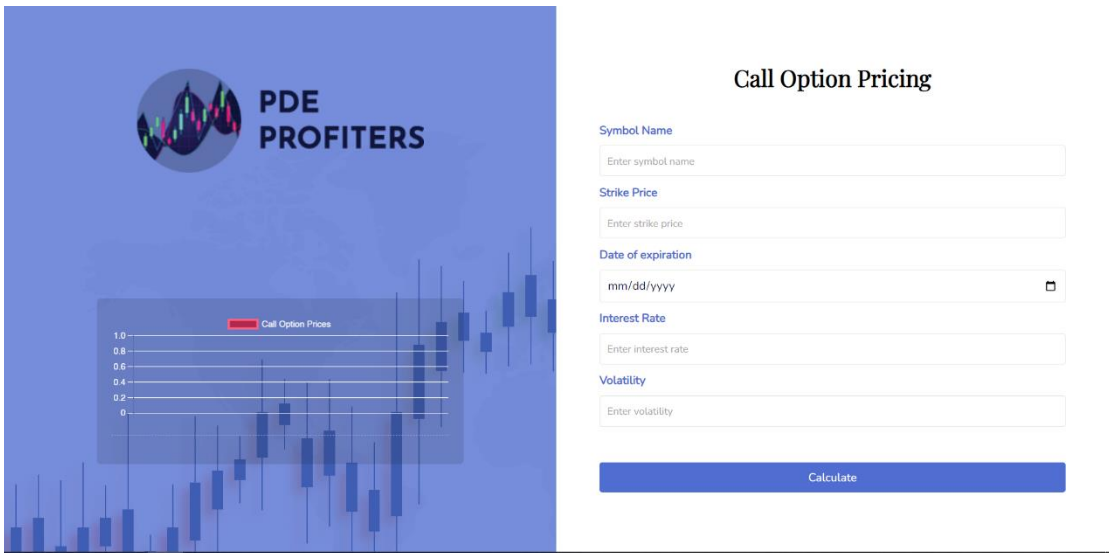


Figure 6 shows the web application user interface.

V. Hypothesis

How to predict the call price?

Usually, the Black-Scholes equation is solved using numerical methods but, in some cases, an analytical solution can be obtained by using the standard convolution method. $V(S,t)$, representing a solution to a parabolic partial differential equation (PDE), establishes a specific relationship between (V) and its partial derivatives with respect to the independent variables (S) and (t). In the context of the Black-Scholes PDE, which is a backward parabolic equation, the boundary conditions are crucial for obtaining an exact solution. The terminal condition for a European call option, $c(S,T)$, is given as the maximum of $S-K$ and zero, and as S approaches infinity, the option value approaches the difference between the asset value and the exercise price. Additionally, for a put option, the terminal condition is $p(S,T) = max(K-S,0)$, with corresponding boundary conditions ensuring the option value remains zero when S is zero and approaches zero as S tends to infinity. The hypothesis emphasizes the importance of these boundary and terminal conditions in determining unique solutions for parabolic PDEs, particularly in the financial context of option pricing.

To solve the Black-Scholes equation using the finite difference method, a fully implicit approach is chosen for its efficiency and stability in pricing various derivatives. The boundary conditions for European call and put options, with strike K and maturity T, are established. Derivative approximations using finite difference methods, including forward, backward, and central differences, are applied to replace partial derivatives in the equation. Implicit discretization involves restricting the theoretical infinite domain to a finite region and replacing it with a discrete grid. The discretized equation is then rearranged to find the values of V at the next time step $(n+1)$ in terms of the values at the current time step (n), using a backward scheme for the time derivative and central discretization for the space derivative. The resulting discretized equation with renamed coefficients $(a, b,$ and $c)$ represents the iterative process of solving for V^n given the knowledge of V^{n+1}. This comprehensive approach underscores the systematic application of finite difference methods and implicit discretization to obtain numerical solutions for the Black-Scholes equation within a discrete grid framework.

Our third approach is to use AI to solve Black-Scholes; this method is based on developing a deep learning model. This method works by representing data from numerous inputs and the appropriate solutions to those inputs are used to train this model. These solutions are acquired by the analytical approach to solving Black-Scholes so that this model can independently predict the solution.

VI. Experimental Work

In our comprehensive analysis, we explore three distinct methods for solving the given problem: analytically, numerically, and using artificial intelligence (AI)-based approach. For the analytical method, we utilize PDE solving techniques and formulas to derive solutions, providing a theoretical foundation for our problem. The numerical method involves employing algorithms and computational techniques, specifically finite difference technique, to approximate solutions, offering a practical and flexible approach. Additionally, we integrate AI methodologies, thoroughly testing each method with appropriate datasets customized to their respective input and output requirements. This ensures a comprehensive understanding of the strengths, limitations, and applicability of each solution methodology in addressing the specific challenges posed by the problem at hand.

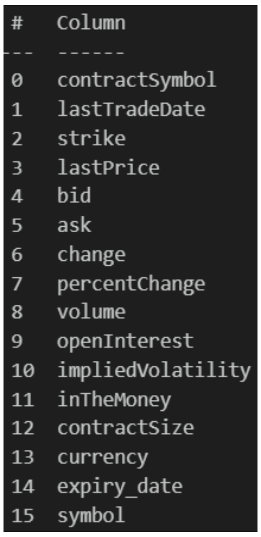
Data Collection

The analysis was conducted using data obtained from the Yahoo Finance API, focusing on four companies: Microsoft, Apple, Amazon, and Tesla.

The investigation involved call options with various expiry dates ranging from December 8, 2023, to January 16, 2026.

The specific expiry dates considered were "2023-12-08", "2023-12-15", '2023-12-22', "2023-12-29", "2024-01-05", "2024-01-12", "2024-01-19", "2024-02-16", "2024-03-15", "2024-04-19", "2024-06-21","2024-09-20","2025-01-17", "2025-06-20", "2025-12-19", "2026-01-16".

The dataset featured various columns, providing detailed insights into the relevant financial metrics and parameters for further analysis.



```
#   Column
--- ------
0   contractSymbol
1   lastTradeDate
2   strike
3   lastPrice
4   bid
5   ask
6   change
7   percentChange
8   volume
9   openInterest
10  impliedVolatility
11  inTheMoney
12  contractSize
13  currency
14  expiry_date
15  symbol
```

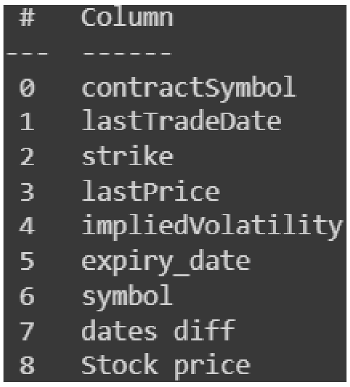
Figure 7 columns of collected data.

Data processing

Our primary focus centers on specific columns, namely 'contractSymbol,' 'lastTradeDate,' 'strike,' 'lastPrice,' 'impliedVolatility,' 'expiry_date,' and 'symbol.'

These columns contain critical information, with each element representing a key aspect: contract symbol, initial trading date, strike price, call option price, volatility, expiration date, and the associated symbol.

The dataset has undergone thorough cleaning, ensuring data accuracy and normalization of formats for consistency.



```
#   Column
--- ------
0   contractSymbol
1   lastTradeDate
2   strike
3   lastPrice
4   impliedVolatility
5   expiry_date
6   symbol
7   dates diff
8   Stock price
```

Figure 8 shows the final dataset shape.

Additionally, a new column has been introduced to capture the time difference between the initial trade date and the expiry date, enhancing temporal insights. Furthermore, the stock price information has been integrated into the initial dataset through the Yahoo API, establishing a valuable link between the financial data and real-time stock prices for a more comprehensive analysis.

Real-Market Data Analysis

Using the collected data, we could plot the data distribution of the stock price at trading times and the call price of different companies and analyzed them as shown below.

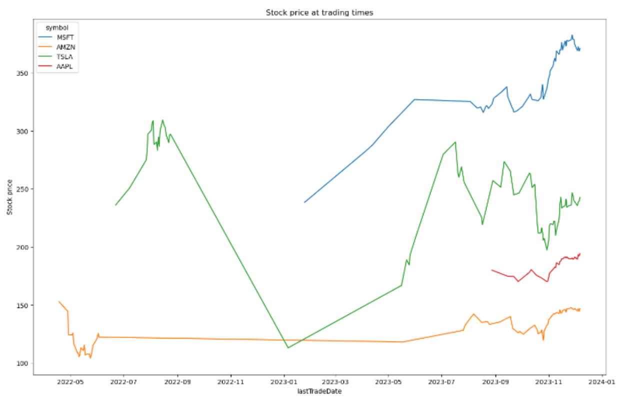


Figure 10 represents the stock prices at last trade dates.

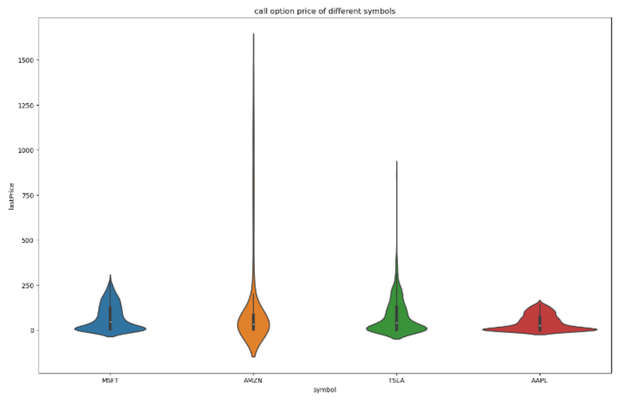


Figure 9 represents the concentration of call option prices of different companies.

The above figures represent the distribution of data at given intervals. Figure 9 represents the deviation that happened in the stock price with respect to time. Each company is represented with lines according to intervals we could get from yahoo finance. For example, TSLA has a great deviation from every time to another so having an option price of these companies makes you more aware to the deviations happened to get no or a little amount of loss.

Figure 10 represents the number of call option of each company and its relationship with last price. There are some outliers in every company which aren't used in our work because they aren't constant and happen without any desire. For example, amazon has most of the call options with nearly 125 last prices, so this indicator gives us the average price and the up and down price boundaries.

The descriptive statistics for our collected data for Microsoft, Apple, Amazon and Tesla is shown below.

	strike	lastPrice	impliedVolatility	dates diff	Stock price
count	35727.00000	35727.00000	35727.000000	35727.00000	35727.00000
mean	163.558230	33.963996	0.165873	187.674280	115.886382
std	380.722103	100.049276	0.697095	234.796544	135.571782
min	0.350000	0.010000	0.000000	2.000000	0.860300
25%	20.000000	0.440000	0.000010	23.000000	19.219999
50%	50.000000	4.400000	0.000010	65.000000	49.340000
75%	155.000000	20.150000	0.125009	262.000000	148.839996
max	5400.000000	1499.750000	39.875004	967.000000	504.045685

Table 3 represents the statical data for the analytical

Our models

1) Black-Scholes Analytical Solution

Inputs: stock price S, strike price K, time to expiration T, risk-free rate r, volatility σ and option type $call\ or\ put$.

Outputs: The price of the type of option you provide in inputs value

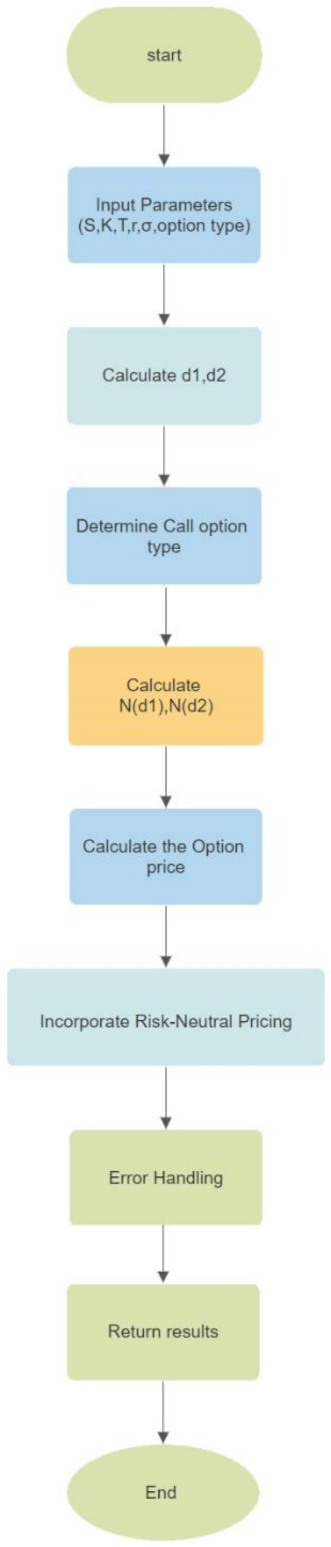

start
Input Parameters (S,K,T,r,σ,option type)
Calculate d1,d2
Determine Call option type
Calculate N(d1),N(d2)
Calculate the Option price
Incorporate Risk-Neutral Pricing
Error Handling
Return results
End

Figure 11 shows the analytical method flow chart.

Algorithm:

1.Receive input parameters for the Black-Scholes formula: stock price S, strike price K, time to expiration T, risk-free rate r, volatility σ and option type $call\ or\ put$.

2. Use the Black-Scholes formulas to calculate $d1$ and $d2$ based on the provided input parameters.

3. Determine Option Type is a call or put.

4.Utilize the cumulative distribution function from a standard normal distribution for $N(d1)$ and $N(d2)$.

5.Calculate the call option price based on the Black-Scholes formula for calls.

6.Calculate the put option price based on the Black-Scholes formula for puts.

6. Incorporate risk-neutral pricing concepts, accounting for the time value of money and the risk-free rate.

7. Error Handling: Check for valid option types and handle errors appropriately (e.g., raise a Value Error for an invalid option type).

8. Return Result: Provide the calculated option price as the output of the function.

2) Black-Scholes Numerical Solution

Inputs the risk-free rate (r), volatility (σ), strike price (K), and maturity (T).

Outputs: Grid of call option as a function of call option and price.

Algorithm:

1. Initialize parameters: Set values for the risk-free rate (r), volatility (σ), strike price (K), and maturity (T).

2. Discretize time and space: Define discrete time steps (Δt) and space steps (Δx), creating a grid for computation.

3. Change of variables: Transition to the log variable x=log(s) for computational simplicity.

4. Establish boundary conditions: Define terminal and lateral conditions for call and put options based on strike (K) and maturity (T).

5. Derivative approximation: Use finite difference methods to approximate partial derivatives in time and space.

6. Create auxiliary variables: Introduce variables for simplification in the implicit discretization step.

7. Implicit discretization: Apply implicit discretization for the time derivative in the Black-Scholes equation.

8. Create matrix equation: Represent the discretized equation in matrix form.

9. Inversion and solution: Invert the matrix D to solve for V^n at each time step.

10. Iterate through time steps: Compute option values for each time step, progressing from initial to final time.

11. Output: Obtain the final option values for various time steps and spatial locations.

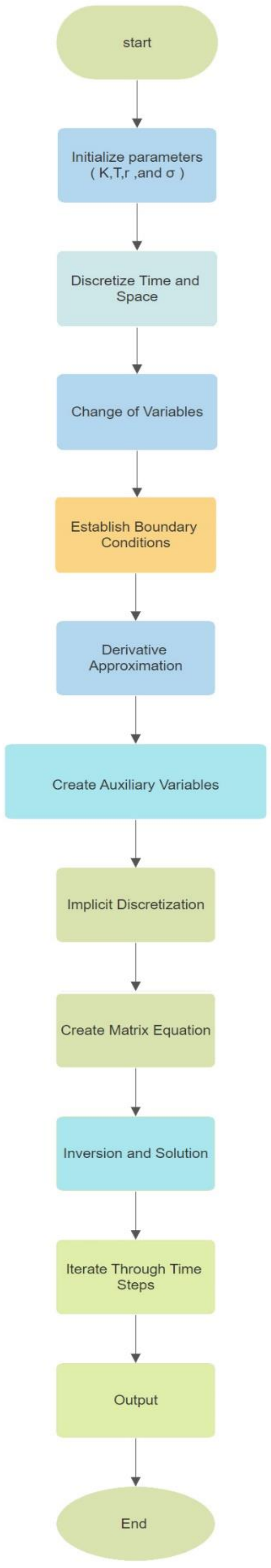

start
Initialize parameters (K,T,r ,and σ)
Discretize Time and Space
Change of Variables
Establish Boundary Conditions
Derivative Approximation
Create Auxiliary Variables
Implicit Discretization
Create Matrix Equation
Inversion and Solution
Iterate Through Time Steps
Output
End

Figure 12 shows the numerical solution flow chart.

3) Black-Scholes AI Solution

Inputs: stock price S, strike price K, time to expiration T, risk-free rate r, volatility σ and option type $call\ or\ put$.

Outputs: The type of option you provide in inputs value.

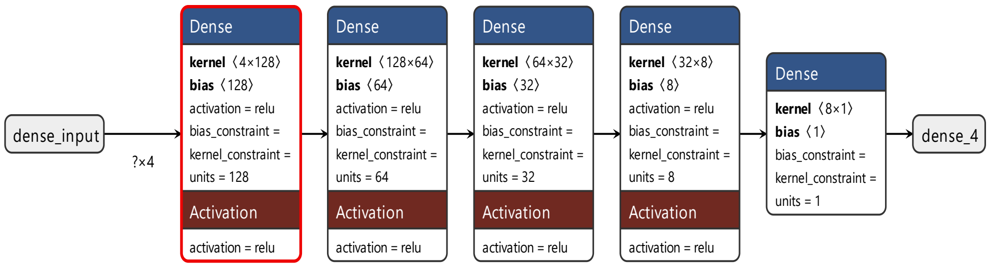
Algorithm:

1. Data Splitting: Split the dataset into training and testing sets using `train_test_split` with a test size of 20% and a random seed of 40.

2. Model Initialization: Initialize a neural network model using TensorFlow's Sequential API.

3. Add Layers:

- Dense layer with 128 neurons and ReLU activation.
- Dense layer with 64 neurons and ReLU activation.
- Dense layer with 32 neurons and ReLU activation.
- Dense layer with 8 neurons and ReLU activation.
- Output layer with 1 neuron and linear activation.



4. Model Compilation: Compile the model using the Adam optimizer and mean squared error (MSE) as the loss function.

5. Callback Setup: Set up an early stopping callback to monitor validation loss and stop training if it does not improve for 10 consecutive epochs.

6. Model Training:

- Train the model on the training data (X_train, y_train) for 1000 epochs.
- Use a batch size of 32 and include a validation split of 30%.
- Incorporate the early stopping callback to prevent overfitting.

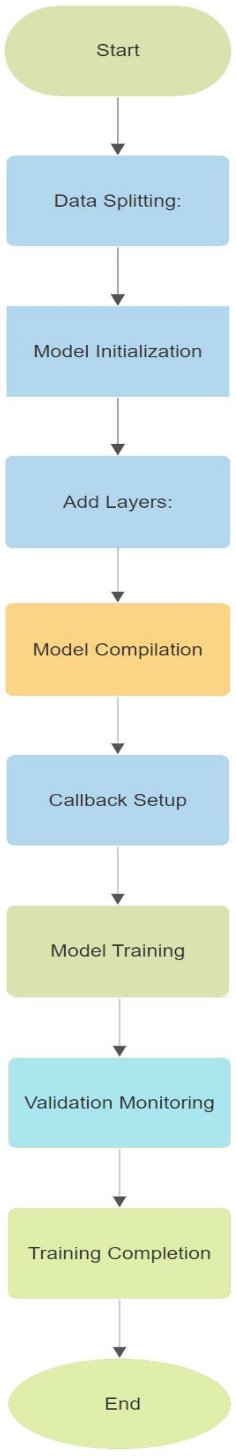


Figure 13 shows the AI solution flow chart.

VII. Results

We have developed three models that enable users to make informed decisions about investing in companies. Our model provides users with a call option at regular intervals, allowing them to determine whether a company is profitable or not. By knowing the call option at a future interval, users can be confident that they will make a gain around the given number and won't make a hasty decision to sell the call option due to a temporary dip in price.

To develop the model, we split our dataset, consisting of approximately 34480 values, into two datasets: a randomly-distributed 20% for testing the analytical and AI model, and a customized 80% for training the AI.

1) Black-Scholes Analytical Solution

We began by evaluating the Black-Scholes solution using the random 20% of the entire dataset, which represents around 6896 values from total of around 34480. The graph below indicates that the BS (Analytical) solution is an accurate predictor of the call option price. Additionally, the Mean Absolute Error (MAE) is 14.026966, which suggests that the model provided a reliable estimate.

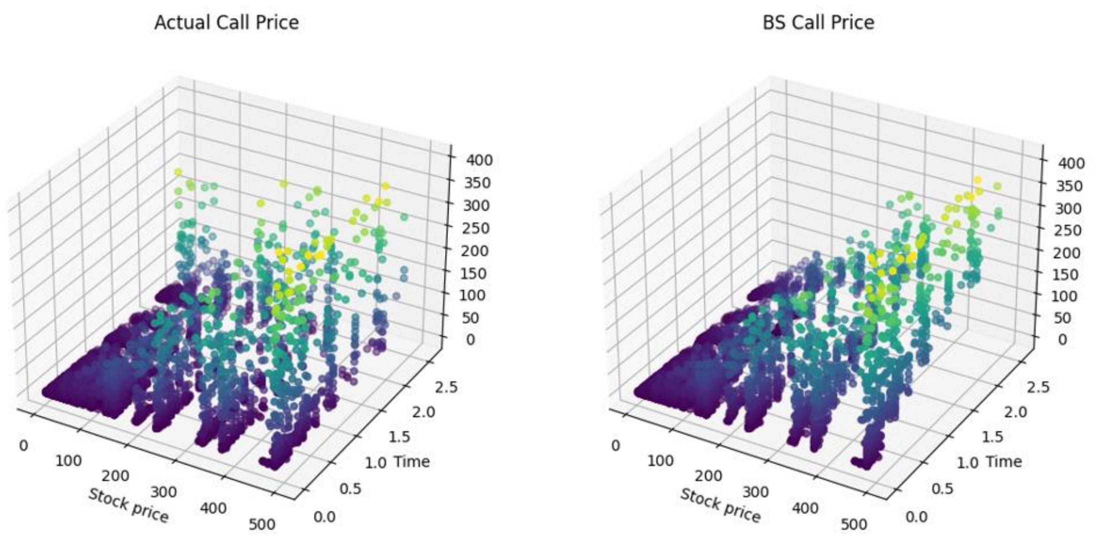


Figure 14 shows the results of analytical model with 20% of the entire dataset.

2) Black-Scholes AI Solution

The comprehensive training of our model on the 80% of the dataset has culminated in exceptional outcomes in testing the remaining 20%. After undergoing meticulous evaluation, it has demonstrated the ability to generate predictions that are impressive as shown below.

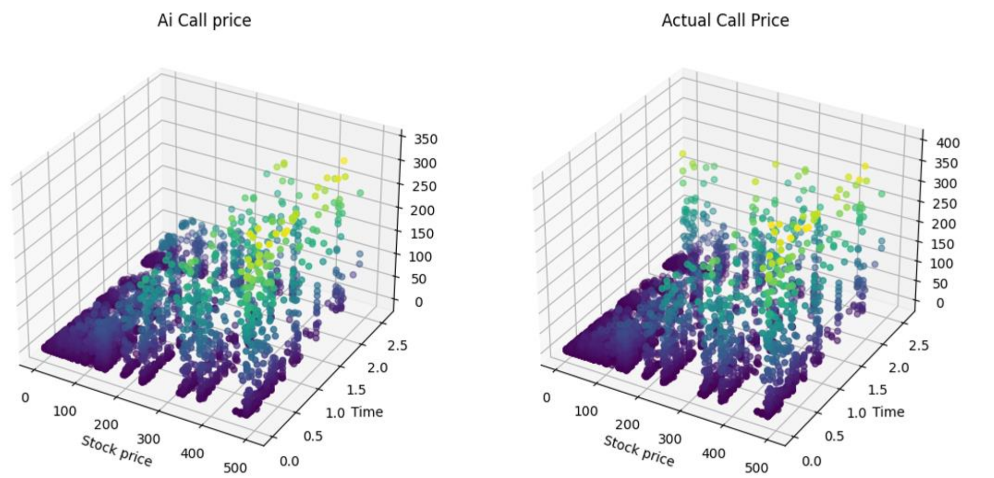


Figure 15 represents the output data from AI model and the actual price on 20% of the entire dataset.

3) Black-Scholes Numerical Solution

The numerical solution generates a grid for every input given (stock price – strike - time to expire - volatility) showing inputs at different times and stock prices, while keeping all other parameters constant. So, its execution time is high and depends on the grid size as for (4000,5000) it takes about 2 seconds for every input to get an output.

As such, we tried to use a smaller grid that maintains balance between execution time and accuracy. This grid size is (2000, 250) with time-execution 0.002 for every input.

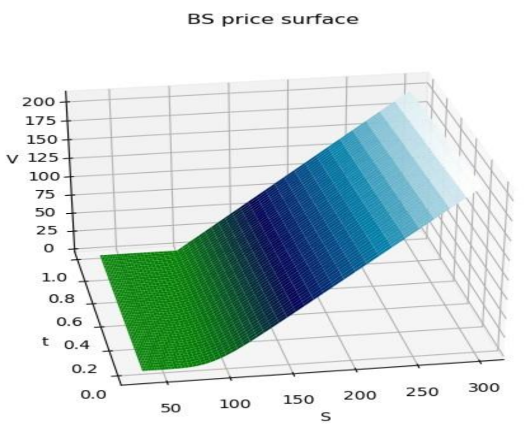


Figure 16 shows the grid formed at specific inputs of stock price = 150, strike price=100 and expiration date= one year and volatility=10.

As before, we began our evaluation by using random 20% of the entire dataset and then evaluating the model of the entire dataset as whole.

The graph shown indicates that the numerical solution is an accurate predictor of the call option price. Additionally, the Mean Absolute Error (MAE) is 32.310085, which suggests that the model provided a good estimate; however, it fails when it comes to outliers.

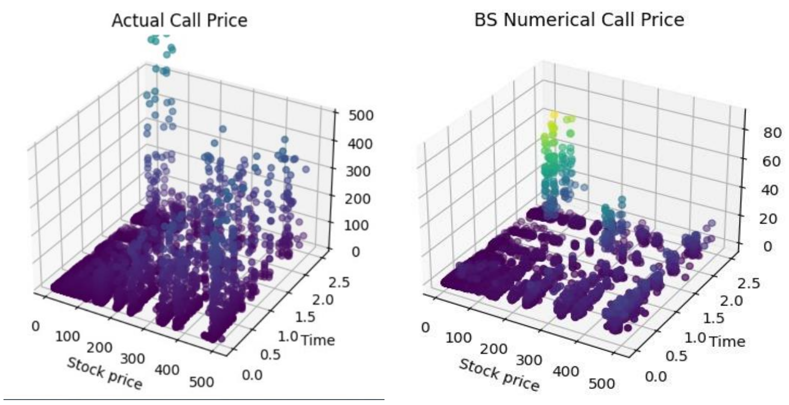


Figure 17 represents the output data from numerical model and the actual price on 20% of the entire dataset.

Comparison between the three solutions

Comparing the three solutions on the 20% of the entire dataset, we also calculated their Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) as shown below.

METRIC	Analytical Solution	AI Solution	Numerical Solution
Mean Absolute Error (MAE)	13.667211	4.688871	32.310085
Mean Absolute Percentage Error (MAPE)	4.475546	11.067686	18.584168

Table 4 shows the MAE and MAPE comparing the analytical, the numerical, the AI solutions on 20% of the dataset.

Then, we conducted a comparison between the three solutions using the entire dataset across all strike prices. The results indicate that while the Black-Scholes (BS) analytical solution failed to predict the data outliers, the AI model was able to provide approximate predictions as shown.

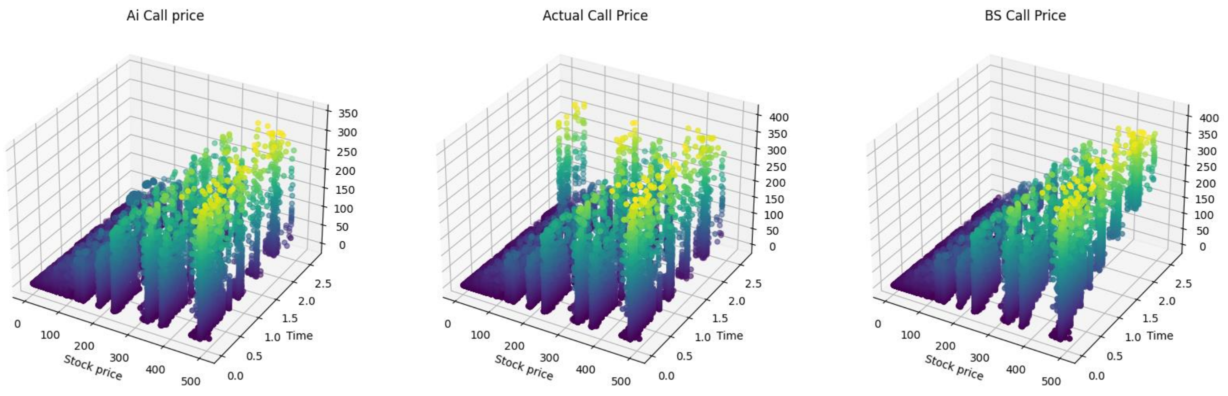


Figure 18 represents the normalized full dataset and compared with the outputs of the analytical model, AI model, and numerical model.

Comparing the three solutions, we also calculated their Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) as shown below.

METRIC	Analytical Solution	AI Solution	Numerical Solution
Mean Absolute Error (MAE)	14.304378	4.7451	32.62239
Mean Absolute Percentage Error (MAPE)	13.884891	11.8834	19.90709

Table 5 shows the MAE and MAPE comparing the analytical, the numerical, the AI solutions on the entire dataset.

VIII. Conclusion

The global financial market is a vast and growing market, with the options market being one of the most widely used markets worldwide. This market allows both individuals and institutions to trade, providing companies with additional funds and investors with a share of the companies' income. However, the market prices are frequently changing and challenging to predict, making it difficult to determine whether the market is entirely beneficial.

We aim to model the option market using the Black-Scholes Model, as it has a significant impact on economic growth. Our analysis explored three different methodologies to solve the model: analytical, numerical, and artificial intelligence (AI)-based approaches. We used data from the Yahoo Finance API, focusing on four companies: Microsoft, Apple, Amazon, and Tesla.

Our results indicated that the AI model provided excellent approximations with low MAE at 4.688871 when tested on 20% of the entire data, while the analytical solution scored an MAE of 13.667211 and the numerical solution scored 32.310085. The results clearly state the AI scored the best results. Indeed, the AI solution, trained on the entire dataset, demonstrated exceptional predictive capabilities. It surpassed the analytical solution and the numerical solution by providing predictions at MAE of 4.7451, while the analytical solution scored MAE of 14.304378and the numerical solution scored 32.62239 and was also able to predict data outliers, while the numerical solution could not – disadvantages of numerical solutions.

The numerical solution generated a grid of inputs at different times and stock prices, but it requires keeping other parameters constant. This may be a weakness of the numerical solution as it may not always be possible to keep these parameters constant.

In summary, the option market is one of the biggest markets globally, with significant economic impacts that require accuracy in predicting option prices. Our analysis provides a comprehensive understanding of the problem and showcases the strengths and limitations of each methodology. The AI solution provides the most efficient solution for the Black-Scholes Model, delivering impressive approximations, even for data outliers. The numerical solution also provides an excellent approximation but requires keeping other parameters constant.

IX. Future Work

In the future, we are looking forward to adding more variable parameters like volatility. Variations in volatility frequently occur in real-time scenarios. For example, during recent escalations in Gaza, many individuals decided to boycott any company dealing with Israel. Therefore, we intend to make our project capable of anticipating how the call price and the stock market will respond to sudden changes. We also plan to consider pandemics such as COVID-19. Furthermore, we will use extra time to train our AI model with more data, to increase its reliability and accuracy.

X. References:

1) M. A. Eissa and M. A. Elsayed, “Improve Stock Price Model-Based Stochastic Pantograph Differential Equation,” vol. 14, no. 7, pp. 1358–1358, Jul. 2022, doi: https://doi.org/10.3390/sym14071358.
2) D. S. A. Wokoma, I. U. Amadi, and P. A. Azor, “Estimation of Stock Prices using Black-Scholes Partial Differential Equation for Put Option,” African Journal of Mathematics and Statistics Studies (AJMSS), vol. 3, no. 2, pp. 80–89, Jun. 2020
3) Y. Chen et al., “Numerical solving of the generalized black-scholes differential equation using Laguerre neural network,” Digital Signal Processing, vol. 112, p. 103003, 2021. doi:10.1016/j.dsp.2021.103003
4) (N.d.-b). *African Journal of Mathematics and Statistics Studies*. https://doi.org/10.52589/ajmss
5) Brenner, M., & Subrahmanyam, M. G. (1994). A simple approach to option valuation and hedging in the black-scholes model. Financial Analysts Journal, 50(2), 25–28. https://doi.org/10.2469/faj.v50.n2.25
6) Nasdaq. (n.d.-a). https://nasdaq.com/sites/acquia.prod/files/Best-Identity-Theft-Protection-Services-2023.jpg%3Fquality%3D85
7) Nuugulu, S. M., Gideon, F., & Patidar, K. C. (2021). A robust numerical scheme for a time-fractional black-scholes partial differential equation describing stock exchange dynamics. Chaos, Solitons &amp; Fractals, 145, 110753. https://doi.org/10.1016/j.chaos.2021.110753
8) J. E. Esekon, "A particular solution of a nonlinear Black-Scholes partial differential equation," International Journal of Pure and Applied Mathematics, vol. 81, 2016.
9) F. Black, "The pricing of commodity contracts," Journal of Financial Economics, vol. 3, no. 0304-405X, pp. 167-179, 1976.
10) Md. N. Anwar and L. S. Andallah, “A study on numerical solution of Black-Scholes model,” SCIRP
11) All dataset used, program codes, and documentations are included in: Github Link
12) Website Link

XI. Appendix

1 import import libraries

1.0.1 tensorflow for the AI module

```
[ ]: import tensorflow as tf
```

```
[ ]: import pandas as pd #pandas for data manipulation
     import numpy as np #numpy for mathematical operations
     from sklearn.model_selection import train_test_split #for splitting dataset to␣
      ↪train and test
     import seaborn as sns #seaborn for data visualization
     import matplotlib.pyplot as plt #matplotlib for pyplot
```

2 the black sholes analytical solution

```
[ ]: import numpy as np
     from scipy.stats import norm
     def black_scholes_analytic(S, K, T, r, sigma, option_type='call'):
         d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
         d2 = d1 - sigma * np.sqrt(T)

         if option_type == 'call':
             option_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
         elif option_type == 'put':
             option_price = K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)
         else:
             raise ValueError("Invalid option type. Use 'call' or 'put'.")

         return option_price
```

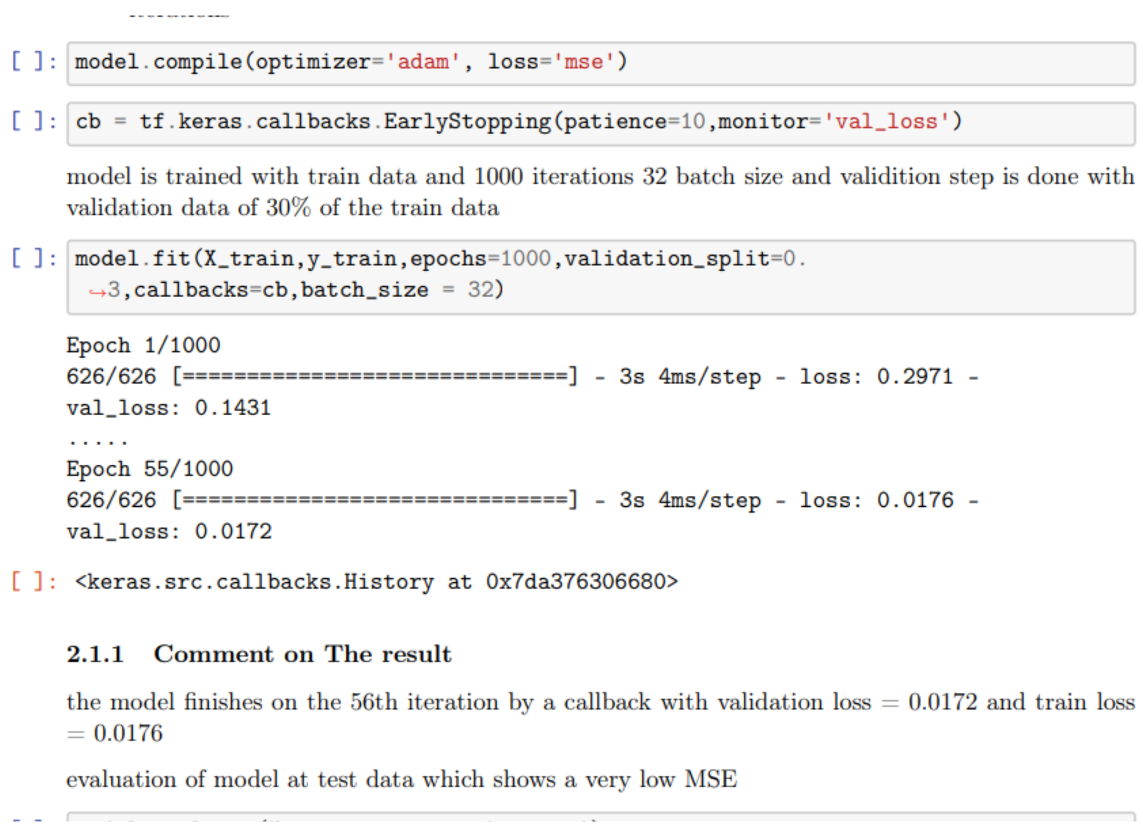
2.1 The AI Model

- the model is ANN Deep learning model
- model consists of 5 dense layers as shown in code
- after the first layer a dropout layer is added to decrease the effect of overfitting
- hidden layers has **relu** activation and nuerons number as shown in code for each layer and output layer has linear activation as it is a regression model

```
[ ]: model = tf.keras.Sequential([

         tf.keras.layers.Dense(128, activation='relu', input_shape=(4,)),
         tf.keras.layers.Dropout(0.5),
         tf.keras.layers.Dense(64, activation='relu'),
         #tf.keras.layers.Dropout(0.5),
         tf.keras.layers.Dense(32, activation='relu'),
         #tf.keras.layers.Dropout(0.5),
         tf.keras.layers.Dense(32, activation='relu'),
         #tf.keras.layers.Dropout(0.5),
         tf.keras.layers.Dense(8, activation='relu'),
         tf.keras.layers.Dense(1, activation='linear')
     ])
```

- model uses ADAM optimizer and MSE error loss
- the model has a callback that stops learning if the validation loss doesn't change after 10 iterations



[ ]:
```
model.compile(optimizer='adam', loss='mse')
```

[ ]:
```
cb = tf.keras.callbacks.EarlyStopping(patience=10,monitor='val_loss')
```

model is trained with train data and 1000 iterations 32 batch size and validition step is done with validation data of 30% of the train data

[ ]:
```
model.fit(X_train,y_train,epochs=1000,validation_split=0.
 ↪3,callbacks=cb,batch_size = 32)
```

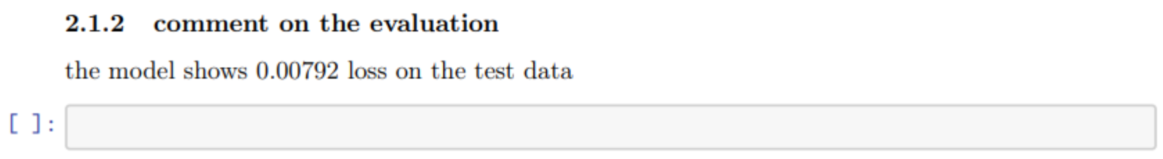
```
Epoch 1/1000
626/626 [==============================] - 3s 4ms/step - loss: 0.2971 -
val_loss: 0.1431
.....
Epoch 55/1000
626/626 [==============================] - 3s 4ms/step - loss: 0.0176 -
val_loss: 0.0172
```

[ ]:
```
<keras.src.callbacks.History at 0x7da376306680>
```

2.1.1 Comment on The result

the model finishes on the 56th iteration by a callback with validation loss = 0.0172 and train loss = 0.0176

evaluation of model at test data which shows a very low MSE



2.1.2 comment on the evaluation

the model shows 0.00792 loss on the test data

[ ]:
```

```

0.1 Numerical solution of the PDE

Ok, let us solve the vector equation derived from the paper

In this case we consider a call option with strike K, maturity T. The stock price S_0 is not relevant for the algorithm. We will use it in the end to compute the value of the option for S_0.

A common practice is to choose the computational region between $3K$ and $K/3$. Then we have $A_1 = \log K/3$ and $A_2 = \log 3K$.

The values of the parameter are:

[8]:
```
r = 0.1
sig = 0.2
S0 = 100
X0 = np.log(S0)
K = 100
Texpir = 1
```

[9]:
```
Nspace = 3000   # M space steps
Ntime = 2000   # N time steps

S_max = 3 * float(K)
S_min = float(K) / 3

x_max = np.log(S_max)   # A2
x_min = np.log(S_min)   # A1
```

[10]:
```
x, dx = np.linspace(x_min, x_max, Nspace, retstep=True)   # space discretization
T, dt = np.linspace(0, Texpir, Ntime, retstep=True)   # time discretization
Payoff = np.maximum(np.exp(x) - K, 0)   # Call payoff
```

[11]:
```
V = np.zeros((Nspace, Ntime))   # grid initialization
offset = np.zeros(Nspace - 2)   # vector to be used for the boundary terms

V[:, -1] = Payoff   # terminal conditions
V[-1, :] = np.exp(x_max) - K * np.exp(-r * T[::-1])   # boundary condition
V[0, :] = 0   # boundary condition
```

[12]:
```
# construction of the tri-diagonal matrix D
sig2 = sig * sig
dxx = dx * dx

a = (dt / 2) * ((r - 0.5 * sig2) / dx - sig2 / dxx)
b = 1 + dt * (sig2 / dxx + r)
c = -(dt / 2) * ((r - 0.5 * sig2) / dx + sig2 / dxx)

D = sparse.diags([a, b, c], [-1, 0, 1], shape=(Nspace - 2, Nspace - 2)).tocsc()
```

[13]:
```
# Backward iteration
for i in range(Ntime - 2, -1, -1):
    offset[0] = a * V[0, i]
    offset[-1] = c * V[-1, i]
    V[1:-1, i] = spsolve(D, (V[1:-1, i + 1] - offset))
```