Connecting Mathematics and Data Science

Emre Anıl Polat

Course Coordinator: Deniz Yılmaz

Course: Math 490

Semester: Fall 2025

Contents

1 Introduction

Financial markets are uncertain systems in which asset prices evolve under the influence of random shocks and changing volatility. Modeling and forecasting this uncertainty is a fundamental problem in both mathematics and data science. Classical approaches in mathematical finance rely on stochastic models that explicitly describe randomness through probability distributions, while modern data-driven methods aim to learn predictive patterns directly from historical data. This divide highlights two important perspectives: one grounded in theoretical assumptions about market behavior, and the other one driven by empirical observations and computational learning.

Models such as Geometric Brownian Motion (GBM) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) are grounded in stochastic calculus and time series theory. These models provide analytical structures that allow us to model volatility clustering and long-term behavior using well-defined probabilistic frameworks. In contrast, statistical and machine learning models such as Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) networks focus on forecasting performance by capturing temporal dependencies, often without explicitly modeling uncertainty. Such methods have gained popularity with the growth of data availability and computational power, which enables models to detect non-linear patterns and adapt to complex market dynamics.

This project connects mathematics and data science by comparing these two modeling styles in the context of stock return forecasting. By examining both traditional stochastic models and modern predictive approaches side by side, this project aims to assess their strengths, weaknesses, and practical usability in real life. At the end, we will use daily log-returns of the S&P 500 tracking ETF to implement GBM, GARCH, ARIMA, and LSTM models and evaluate their predictive performance. Through this comparison, we aim to provide meaningful insights about the applicability of different forecasting models in financial markets and understand which models perform better under market conditions. The results of this study are expected to highlight how mathematical assumptions and data-driven learning contribute to forecast accuracy.

2 Geometric Brownian Motion

2.1 Introduction

Geometric Brownian Motion (GBM) is one of the most fundamental stochastic models used in mathematical finance to describe stock price dynamics. It assumes that the relative change in a stock price consists of a deterministic growth component and a random component expressed by Brownian Motion. Under this model, the stock price process $\{S_t\}$ satisfies the differential equation:

$$dS_t = \mu S_t \, dt + \sigma S_t \, dW_t,$$

where

S_t : stock price at time t,

μ : drift parameter/the expected growth rate,

σ : volatility parameter/return variability,

W_t : standard Brownian motion.

Definition 2.1 (Brownian motion)**.** *Brownian motion $\{W_t\}$ is a continuous-time stochastic process with independent, normally distributed increments and zero mean. [(1), Chapter 2.1, Definition 1.1]*

Brownian motion serves as a mathematical model for random and unpredictable fluctuations. It is widely used to represent the cumulative effects of many independent shocks in financial markets. In the context of this model, dW_t captures the randomness arising from news, trading activity, and other sources of market uncertainty that cannot be predicted deterministically.

The structure of the GBM equation uses two key modeling ideas. The first term ($\mu S_t \, dt$) represents deterministic growth and it describes how the asset price would evolve on average when there is no uncertainty. The second term ($\sigma S_t \, dW_t$) represents random fluctuations. Both terms are multiplied by S_t, which implies that changes in the asset price are proportional to its current level. This assumption models relative returns rather than absolute price changes and ensures that the price process remains strictly positive.

This multiplicative structure leads to log-normally distributed prices and makes GBM

particularly suitable for modeling financial assets in continuous time. Also, the simplicity of the model makes GBM a natural baseline for comparison with more complex stochastic and data-driven models.

2.2 Solution of the GBM Equation

To use GBM with real life data, we need to obtain an explicit solution. This is obtained by applying *Itô's Lemma*, which provides a method for computing the differential of a function of a stochastic process.

Lemma 2.2 (Itô's Lemma)**.** *Let X_t be a stochastic process satisfying*

$$dX_t = a_t\,dt + b_t\,dW_t,$$

and let $f(X_t)$ be a twice continuously differentiable function. Then the differential of $f(X_t)$ is given by

$$df(X_t) = f'(X_t)\,dX_t + \frac{1}{2}f''(X_t)\,(dX_t)^2.$$

[(1)]

We apply Itô's Lemma to the logarithm of the price process. Let

$$X_t = \ln(S_t).$$

Recall that under GBM the stock price equation is

$$dS_t = \mu S_t\,dt + \sigma S_t\,dW_t.$$

Using Itô's Lemma with $f(s) = \ln(s)$, we compute the derivatives as:

$$f'(s) = \frac{1}{s}, \qquad f''(s) = -\frac{1}{s^2}.$$

So using these in the Ito's Lemma we have

$$dX_t = d(\ln S_t) = f'(S_t)\,dS_t + \frac{1}{2}f''(S_t)\,(dS_t)^2 = \frac{1}{S_t}dS_t - \frac{1}{2}\frac{1}{S_t^2}(dS_t)^2.$$

Now substitute $dS_t = \mu S_t\,dt + \sigma S_t\,dW_t$ into the first term:

$$\frac{1}{S_t}dS_t = \frac{1}{S_t}\left(\mu S_t\,dt + \sigma S_t\,dW_t\right) = \mu\,dt + \sigma\,dW_t.$$

Then compute $(dS_t)^2$ for the second term:

$$(dS_t)^2 = \left(\mu S_t\,dt + \sigma S_t\,dW_t\right)^2 = \mu^2 S_t^2 (dt)^2 + 2\mu\sigma S_t^2\,dt\,dW_t + \sigma^2 S_t^2 (dW_t)^2.$$

Using the Itô rules $(dt)^2 = 0$, $dt\,dW_t = 0$, and $(dW_t)^2 = dt$, we find

$$(dS_t)^2 = \sigma^2 S_t^2\,dt.$$

Finally, substituting these terms back into the Itô expansion gives

$$dX_t = \left(\mu\,dt + \sigma\,dW_t\right) - \frac{1}{2}\frac{1}{S_t^2}\left(\sigma^2 S_t^2\,dt\right) = \left(\mu - \frac{1}{2}\sigma^2\right)dt + \sigma\,dW_t.$$

Therefore, the differential of X_t is given by this equation

$$dX_t = \left(\mu - \frac{1}{2}\sigma^2\right)dt + \sigma\,dW_t.$$

This transformation converts the multiplicative stochastic differential equation into a linear differential equation.

Now integrating both sides from 0 to t gives us

$$X_t = X_0 + \left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t.$$

Since $X_t = \ln(S_t)$ and $X_0 = \ln(S_0)$, we take exponents of the both sides and find the closed form solution of the GBM process:

$$S_t = S_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t\right).$$

This solution shows that S_t follows a log-normal distribution. We can forecast stock prices using this equation just by finding the μ and σ from real life data. The parameter μ controls the expected growth of the process, and σ determines the uncertainty of future prices. GBM serves as a mathematical baseline for modeling stock prices and

for comparison with more complex machine learning models.

3 Generalized Autoregressive Conditional Heteroskedasticity

3.1 Introduction

An important feature of financial return series is that volatility is not constant over time. Instead, markets exhibit volatility clustering: Large price movements tend to be followed by large movements, and calm periods tend to be followed by calm periods. Although the mean of daily returns is often close to constant, their variability changes across time. Classical models such as GBM assume a constant volatility parameter, which makes them unable to represent this time-varying uncertainty.

3.2 GARCH Model

The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model addresses this limitation as it allows the conditional variance of returns to evolve over time as a function of past shocks and past volatility. [(2), Chapter 1, Page 307]

Definition 3.1 (Return value)**.** *In this model, the return at time t is defined as*

$$r_t = \mu + \epsilon_t,$$

where μ denotes the average return and ϵ_t is the shock value.

Definition 3.2 (Random shock value)**.** *The shock is defined as*

$$\epsilon_t = \sigma_t z_t, \qquad z_t \sim \mathcal{N}(0,1),$$

as σ_t is the time-varying standard deviation and z_t represents a random disturbance.

The key component of the model is the recursion defining the conditional variance. In the GARCH$(1,1)$ model, volatility changes over time by

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2,$$

where $\omega > 0$ determines the baseline variance level, $\alpha \geq 0$ measures the reaction of volatility to new information (squared shocks), and $\beta \geq 0$ measures the persistence of past volatility, and that is why it is multiplied by past volatility in the equation. This structure allows the model to reproduce volatility clustering and persistent variance dynamics that occur in real financial data. To use this model in a real life data, we need to find the parameters α and β, and then find the volatility over time by using the recursive equation above. Therefore, to find the properties of these parameters, we will first find the expected value of volatility and then the maximum likelihood function of these parameters.

3.3 Expected Value of Volatility

In the GARCH$(1, 1)$ recursion,

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2,$$

we start by taking expectations on both sides:

$$E[\sigma_t^2] \;=\; E[\omega] \;+\; \alpha\, E[\epsilon_{t-1}^2] \;+\; \beta\, E[\sigma_{t-1}^2].$$

Since ω is a constant parameter, its expectation is

$$E[\omega] = \omega.$$

And also let's denote

$$E[\sigma_t^2] = E[\sigma_{t-1}^2] = \bar{\sigma}^2.$$

So now in our main equation we have

$$\bar{\sigma}^2 = \omega + \alpha\, E[\epsilon_{t-1}^2] + \beta\, \bar{\sigma}^2.$$

Also remember the definition of random shock:

$$\epsilon_{t-1} = \sigma_{t-1} z_{t-1}, \qquad z_{t-1} \sim \mathcal{N}(0, 1).$$

Then,

$$E[\epsilon_{t-1}^2] = E[(\sigma_{t-1} z_{t-1})^2] = E[\sigma_{t-1}^2 z_{t-1}^2].$$

Assuming z_{t-1} is independent of σ_{t-1}, we can separate this expected value as:

$$E[\sigma_{t-1}^2 z_{t-1}^2] = E[\sigma_{t-1}^2]\; E[z_{t-1}^2].$$

And since $z_{t-1} \sim \mathcal{N}(0,1)$, we know

$$E[z_{t-1}^2] = 1.$$

Also, by definition $E[\sigma_{t-1}^2] = \bar{\sigma}^2$. Therefore, we have

$$E[\epsilon_{t-1}^2] = \bar{\sigma}^2 \times 1 = \bar{\sigma}^2.$$

Now put this all together in the main equation

$$\bar{\sigma}^2 = \omega + \alpha\, E[\epsilon_{t-1}^2] + \beta\, \bar{\sigma}^2$$

gives

$$\bar{\sigma}^2 = \omega + \alpha\bar{\sigma}^2 + \beta\bar{\sigma}^2.$$

Now collect the $\bar{\sigma}^2$ terms on the left-hand side:

$$\bar{\sigma}^2 - \alpha\bar{\sigma}^2 - \beta\bar{\sigma}^2 = \omega,$$

$$\bar{\sigma}^2(1 - \alpha - \beta) = \omega.$$

Finally, solving for $\bar{\sigma}^2$ gives the expected variance:

$$\boxed{E[\sigma_t^2] = \bar{\sigma}^2 = \frac{\omega}{1 - \alpha - \beta}}.$$

This equation shows the importance of $\alpha+\beta$. As $\alpha+\beta \to 1$, the denominator $1-\alpha-\beta$ becomes very small, which implies that $E[\sigma_t^2]$ becomes very large. This is a big problem for the performance of the model because in this case, volatility is highly persistent and this means that each new volatility value depends almost entirely on the past. But we

don't want our model to entirely rely on past data because in real life new shock values also affect prices. Therefore, from this foundation we generalize a rule as $\alpha + \beta$ must be smaller than 1 in GARCH.

3.4 Calculating the Maximum Likelihood of Parameters

From the definition of random shock, we find the distribution of return price as:

$$\epsilon_t \mid \mathcal{F}_{t-1} \sim \mathcal{N}(0, \sigma_t^2) \quad \Longrightarrow \quad r_t \mid \mathcal{F}_{t-1} \sim \mathcal{N}(\mu, \sigma_t^2),$$

where $\mathcal{F}_{t-1}$ denotes the information up to time $t-1$, and θ denotes the vector of model parameters.

Then from the formula of the density function with normal distribution, we find the density of r_t given past information as:

$$f(r_t \mid \mathcal{F}_{t-1}; \theta) = \frac{1}{\sqrt{2\pi\sigma_t^2}} \exp\left(-\frac{(r_t - \mu)^2}{2\sigma_t^2}\right).$$

Now assume conditional independence across time, so the likelihood function can be constructed by multiplication of density functions:

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} f(r_t \mid \mathcal{F}_{t-1}; \theta).$$

Taking the logarithm of the likelihood gives the log-likelihood function

$$\ell(\theta) = \sum_{t=1}^{T} \ln f(r_t \mid \mathcal{F}_{t-1}; \theta).$$

Substituting the conditional density we found into this equation gives

$$\ell(\theta) = \sum_{t=1}^{T} \left[-\tfrac{1}{2}\ln(2\pi) - \tfrac{1}{2}\ln(\sigma_t^2) - \frac{(r_t - \mu)^2}{2\sigma_t^2}\right].$$

The log-likelihood is a sum of individual contributions,

$$\ell(\theta) = \sum_{t=1}^{T} \ell_t(\theta),$$

where

$$\ell_t(\theta) = -\tfrac{1}{2}\left[\ln(2\pi) + \ln(\sigma_t^2) + \frac{(r_t-\mu)^2}{\sigma_t^2}\right].$$

Finally, the maximum log-likelihood function to estimate the parameters is:

$$\boxed{\ell(\theta) = -\frac{1}{2}\sum_{t=1}^{T}\left[\ln(2\pi) + \ln(\sigma_t^2) + \frac{(r_t-\mu)^2}{\sigma_t^2}\right].}$$

This is the final log-likelihood function we use to estimate the model parameters. By evaluating this function for different parameter values, we can measure how well a given parameter set explains the observed data, and parameter estimation is performed by finding the values that maximize the log-likelihood. So actually we can compare different parameter groups and find out which one is better for our data. Hence, this function does not give the best parameters directly, but it helps us to evaluate our parameters performance. It is almost impossible to find the best parameters by calculating manually because it takes a lot of time to test each possible parameter's likelihood value. But there is a Python package called ARCH that gives the best parameters for our data since it can calculate likelihood of each pair in a short time. Using that we can find the best parameters for our data, and when we have the parameters, we can easily calculate the volatility at each time step, since it is a recursive function. [(3), Chapter 12.3, Page 419] And then we can use those results to calculate the return value. This concludes how we use the GARCH model in forecasting.

4 Long Short Term Memory

4.1 Introduction

Classical stochastic models such as Geometric Brownian Motion (GBM) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) describe uncertainty through explicit probabilistic structures. However, real financial time series often exhibit nonlinear patterns, structural breaks, and long-term dependencies that are difficult to capture with closed-form mathematical models. To address these limitations, modern data-driven approaches such as Long Short-Term Memory (LSTM) networks have become widely used in financial forecasting.

LSTMs are a specialized type of recurrent neural network designed to model sequential data while overcoming the vanishing gradient problem, which prevents classical Recurrent Neural Networks (RNNs) from learning long-range dependencies. By using gating mechanisms that regulate how information is stored, updated, and forgotten, LSTMs can learn complex temporal relationships present in stock prices and returns. [(4), Chapter 3, Section 3.10] This makes them particularly well-suited for forecasting tasks in which patterns are subtle, nonlinear, or persist across extended periods of time.

4.2 Limitations of Classical RNNs: The Vanishing Gradient Problem

RNNs model sequential data by updating a hidden state h_t at each time step. The hidden state serves as a memory vector that stores information about all past inputs up to time t. A standard RNN updates this state using the recurrence

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b),$$

where

- x_t is the input at time t (for example, the price at time t),
- h_{t-1} is the previous hidden state,
- W_h is the weight matrix that determines how past information influences the present,
- W_x is the weight matrix that connects the current input to the hidden state,
- b is a bias vector applied at each step,
- $\tanh(\cdot)$ is an activation function ensuring nonlinearity.

During training, RNNs use backpropagation-through-time to compute gradients. However, since the derivative of the tanh activation satisfies $|\tanh'(x)| \leq 1$, the repeated multiplication of these derivatives across many time steps causes gradients to decrease exponentially. This phenomenon is called the *vanishing gradient problem.*

As a result, classical RNNs struggle to learn long-term dependencies: information that occurred many time steps earlier has almost no influence on parameter updates.

In the context of financial modeling, this means that RNNs cannot reliably capture long-lasting patterns such as persistent volatility regimes, trends, or cyclical behaviors. These limitations motivate the use of Long Short-Term Memory (LSTM) networks, which explicitly address the vanishing gradient problem through a gated architecture.

4.3 LSTM Architecture: Mathematical Formulation

LSTM networks extend classical RNNs by introducing a separate *cell state* C_t, which serves as a long-term memory pathway designed to preserve information across many time steps. Unlike the hidden state h_t, which stores short-term information, the cell state is updated through controlled mechanisms called *gates*. These gates determine what information is kept, updated, or discarded and this allows the network to overcome the vanishing gradient problem.

At each time step t, an LSTM computes four main components: the forget gate, the input gate, the cell state update, and the output gate. Given the previous hidden state h_{t-1} and the current input x_t, the gates are defined as follows.

Forget Gate. The forget gate determines which parts of the previous cell state should be remembered:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f),$$

where $\sigma(\cdot)$ is the logistic sigmoid activation. Values close to 1 preserve information, while values close to 0 cause the network to forget it.

Input Gate. The input gate controls how much new information enters the cell state:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i),$$

and a candidate update is computed as

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C).$$

Cell State Update. The cell state combines old information scaled by the forget gate with new information scaled by the input gate:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t,$$

where $\odot$ denotes element-wise multiplication. This equation is the main part of the LSTM: the forget gate preserves needed long-term information, and the input gate introduces new information only when relevant. So it creates a balance between them. Since C_t is modified through addition rather than repeated multiplication, gradients propagate through time more easily.

Output Gate. Finally, the output gate determines which parts of the updated cell state form the new hidden state:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o),$$

$$h_t = o_t \odot \tanh(C_t).$$

Summary

- The **forget gate** f_t removes irrelevant past information so that it prevents old patterns from dominating new ones.
- The **input gate** i_t selectively writes important new information into memory.
- The **cell state** C_t acts as a controlled memory that preserves key information over long horizons.
- The **output gate** o_t filters the cell state to produce the hidden state h_t used for predictions.

Through these mechanisms, LSTM can maintain important information for long periods of time. It can adaptively forget what is no longer needed. This makes LSTMs effective for financial time series, where trends, volatility phases, and market may persist for many time steps and cannot be easily captured using classical RNNs or mathematical models such as GBM and GARCH.

4.4 Parameter Estimation in LSTM Networks

The parameters of an LSTM network consist of all weight matrices and bias vectors associated with the forget, input, cell, and output gates:

$$\theta = \{W_f, W_i, W_C, W_o, b_f, b_i, b_C, b_o, W^{(\text{out})}, b^{(\text{out})}\}.$$

These parameters are learned directly from the data through an optimization procedure known as *training*.

Given a dataset of input–output pairs

$$(X^{(i)}, y^{(i)}), \qquad i = 1, \ldots, N,$$

where $X^{(i)}$ contains a sequence of past returns and $y^{(i)}$ is the next return, the LSTM produces a prediction of the form

$$\hat{y}^{(i)} = f_\theta(X^{(i)})$$

To find prediction error, a loss function such as the mean squared error (MSE) is defined:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \big(y^{(i)} - \hat{y}^{(i)}\big)^2$$

This measures how wrong the predictions are. So, the goal of training is to find the parameter vector θ that minimizes the loss:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta).$$

Since no closed-form solution exists, for this optimization problem, gradient-based methods are used. The gradients of the loss with respect to each parameter are computed using backpropagation-through-time, which applies the chain rule across all time steps of the LSTM. Actually it computes how much each weight contributed to the error, across all time steps. The parameters are updated iteratively in the direction that makes the error smaller.

$$W \leftarrow W - \eta \frac{\partial \mathcal{L}}{\partial W}$$

where η denotes the learning rate, which is chosen manually, in financial forecasting the 0.001 value is used most of the time.

The training process is repeated over many iterations until convergence. This means until the loss stops getting significantly better. In this way, the LSTM learns the weight and biases that best capture the temporal patterns in the financial data.

4.5 Using LSTM for Stock Price Forecasting

To apply LSTM networks in financial forecasting, we treat historical stock prices or returns as a sequential dataset and train the network to learn patterns across time. Unlike GBM or GARCH, LSTM learns directly from the data by adjusting its internal weights through backpropagation. This allows the model to capture nonlinear dependencies, long-term effects, and structural patterns that may not be expressible in closed-form mathematical models.

Data Preparation. Let $r_1, r_2, \ldots, r_T$ denote a sequence of historical returns. We choose a window size k and construct input–output pairs of the form

$$X^{(i)} = [r_i, r_{i+1}, \ldots, r_{i+k-1}], \qquad y^{(i)} = r_{i+k},$$

where $X^{(i)}$ is inserted into the LSTM and $y^{(i)}$ is the target value.

How the LSTM Produces a Forecast. Once the LSTM has computed the hidden state h_t, this hidden state is inserted into a final output layer that converts the internal representation of the sequence into a numerical prediction. Mathematically, the forecasted value is obtained as

$$\hat{y}_{t+1} = W_o^{(\text{out})} h_t + b^{(\text{out})},$$

where $W_o^{(\text{out})}$ and $b^{(\text{out})}$ are the weights and bias of the output layer. Depending on the modeling choice, $\hat{y}_{t+1}$ may represent the next-period return, log-return, or price.

Therefore, the LSTM forecasting mechanism can be summarized as

$$(x_{t-k+1}, \ldots, x_t) \xrightarrow{\text{LSTM}} h_t \xrightarrow{\text{output layer}} \hat{y}_{t+1},$$

which shows that the LSTM first processes the sequence to extract temporal dependencies, and then applies a linear transformation to map this internal representation into a numerical prediction.

5 Autoregressive Integrated Moving Average

5.1 Introduction

Autoregressive Integrated Moving Average (ARIMA) models provide a classical, statistics based approach to time series forecasting. Unlike GBM and GARCH, which are built on stochastic calculus and conditional variance, ARIMA models are formulated directly in terms of past observations and past forecast errors. They are suited for modeling time series that exhibit temporal dependence and can be made stationary through differencing.

In ARIMA, the future value of a time series is expressed as a linear combination of its own past values (autoregressive part), past error terms (moving average part), and possibly differenced values to remove trends or non-stationarities (integrated part). [(5), Chapter 7, Section 3] An ARIMA(p, d, q) model is defined by:

- p: the order of the autoregressive component,
- d: the degree of differencing needed to achieve stationarity,
- q: the order of the moving average component.

In the context of financial forecasting, ARIMA models are typically applied to log-returns or differenced price series. This allows us to capture short-term linear dependencies and mean-reversion effects in the data. In this project, ARIMA serves as a classical time series benchmark against which the performance of GBM, GARCH, and LSTM can be compared.

5.2 AR and MA Processes

ARIMA models are constructed from two fundamental linear time series structures: autoregressive (AR) processes and moving average (MA) processes. It is needed to understand these components before introducing the full ARIMA(p, d, q) model.

Autoregressive (AR) Processes. An autoregressive process of order p, denoted AR(p), shows the current value of the series as a linear combination of its previous p values plus a random noise term:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t,$$

where $\phi_1, \ldots, \phi_p$ are the autoregressive coefficients and $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ is random noise. Autoregressive coefficients measure how strongly past values influence the current value. A large positive ϕ_1 indicates that X_{t-1} contributes to X_t strongly. And a negative coefficient implies reversing the direction. If the coefficient is zero, this means that there is no influence from yesterday's return, so that it doesn't have an effect in the linear combination above. The AR component captures the idea that past values influence the present through linear dependence.

Moving Average (MA) Processes. A moving average process of order q, denoted MA(q), models the current value as a linear combination of the present and past q error terms:

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q},$$

where $\theta_1, \ldots, \theta_q$ are the moving average coefficients. They capture how past shocks affect the current observation. Each ϵ_t is random noise and it represents new unexpected information. The MA component captures the impact of past shocks or disturbances on the current value.

ARMA Processes. The AR and MA components are combined in the ARMA(p, q) model:

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}.$$

To express ARMA models compactly, the *backshift operator* B is used. The operator

shifts a time series one step backwards:

$$BX_t = X_{t-1}.$$

By applying the operator k times, we can find the k-lag value:

$$B^k X_t = X_{t-k}.$$

The backshift operator allows autoregressive and moving–average models to be written in a compact form. For example, an AR(1) model

$$X_t = \phi X_{t-1} + \epsilon_t$$

can be expressed as

$$(1 - \phi B)X_t = \epsilon_t.$$

Similarly, an MA(1) model

$$X_t = \epsilon_t + \theta \epsilon_{t-1}$$

becomes

$$X_t = (1 + \theta B)\epsilon_t.$$

This notation simplifies derivations and makes it easier to manipulate ARMA and ARIMA models using polynomial expressions in B.

Also with this notation, the AR and MA polynomials become,

$$\phi(B)X_t = (1 - \phi_1 B - \cdots - \phi_p B^p)X_t, \qquad \theta(B)\epsilon_t = (1 + \theta_1 B + \cdots + \theta_q B^q)\epsilon_t,$$

allowing the ARMA(p, q) model to be written concisely as

$$\phi(B)X_t = \theta(B)\epsilon_t.$$

These structures form the ARIMA model, which extends ARMA(p, q) by introducing differencing to handle non-stationary time series.

5.3 Selecting the Parameters p and q

In an ARIMA(p, d, q) model, the parameters p and q are the orders of the autoregressive and moving–average components respectively. The method for selecting these parameters is based on the behavior of the Autocorrelation Function (ACF) and the Partial Autocorrelation Function (PACF).

In time–series analysis, lag is defined as how many time steps in the past we look when to relate the current value X_t to its previous observations. So, the value X_{t-k} is called the lag-k value of the series. For example, lag 2 compares X_t with X_{t-2}.

Autocorrelation Function (ACF). The ACF measures the correlation between the time series and its past values:

$$\rho(k) = \text{corr}(X_t, X_{t-k}).$$

If the ACF cuts off, which means it drops to zero after a finite lag, this suggests an MA(q) model. Then we find

$$q = \text{lag at which the ACF cuts off.}$$

Partial Autocorrelation Function (PACF). The PACF measures the correlation between X_t and X_{t-k} after removing the effects of intermediate lags. So while ACF measures total correlation, PACF measures direct effect only. It is defined as:

$$\phi_{kk} = \text{partial correlation between } X_t \text{ and } X_{t-k}.$$

If the PACF cuts off after lag p, this indicates an AR(p) model:

$$p = \text{lag at which the PACF cuts off.}$$

A cutoff in the ACF suggests that shocks have a short-lived impact (MA behavior), whereas a cutoff in the PACF indicates dependence on a finite number of past values (AR behavior). By examining these plots, the orders p and q can be chosen in a principled and interpretable way.

5.4 From ARMA to ARIMA

The ARMA(p, q) model assumes that the underlying time series is stationary, so its mean and variance remain constant over time. However, many real world financial time series such as stock prices, exhibit trends or changing levels and therefore they are non-stationary. To handle such cases we apply differencing, which removes trends and stabilizes the series.

The differencing operator is defined using the backshift operator B:

$$\nabla X_t = X_t - X_{t-1} = (1 - B)X_t.$$

By applying the operator d times we get the dth-order difference:

$$\nabla^d X_t = (1 - B)^d X_t.$$

For example, the second difference is computed as

$$\nabla^2 X_t \;=\; \nabla(\nabla X_t) \;=\; (X_t - X_{t-1}) \;-\; (X_{t-1} - X_{t-2}) \;=\; X_t - 2X_{t-1} + X_{t-2}.$$

If the differenced series $\nabla^d X_t$ becomes stationary after d differences, it can be modeled by using ARMA(p, q) process:

$$\phi(B)\,\nabla^d X_t = \theta(B)\,\epsilon_t,$$

where $\phi(B)$ and $\theta(B)$ are the autoregressive and moving–average polynomials.

This leads to the ARIMA(p, d, q) model, where:

p : order of the autoregressive component,

d : number of differences required for stationarity,

q : order of the moving–average component.

Thus, an ARIMA model extends ARMA by allowing the original series to be non-stationary, while still capturing linear dependence after differencing. It is one of the most used classical models for forecasting financial time series.

6 Comparison of the Models

6.1 Data Setup

To compare the forecasting performance of the models introduced in the previous sections, we use daily data for the S&P 500 tracking ETF (ticker SPY). This ETF follows the S&P 500 index and is commonly used as a standard reference for the U.S. stock market. Because of this, SPY is a suitable and representative dataset for studying how stock returns behave. For this reason, in this project it will be used in all four models.

The dataset is obtained from a publicly available, pre-processed CSV file containing daily prices for SPY. So for this project a ready to use dataset will be used, there will be no changing in the real data. We use the adjusted closing price series and construct daily log-returns by

$$r_t = \log(P_t) - \log(P_{t-1}),$$

where P_t denotes the closing price on day t. Log-returns are preferred because they are approximately additive over time and are standard in both the mathematical finance and time series literature. [(6), Page 230, Section 5.2]

In the analysis, all models are trained on the first 80% of the log-return observations and evaluated on the remaining 20% which we call it the test data.[(3)] Each model is tasked to produce one step ahead forecasts of r_t, so that their performance can be compared on exactly the same prediction problem by using the same data for each model.

Figure 1: Change of Price over Time

6.2 Model Comparison and Interpretation

Each model is evaluated on the same one-step-ahead forecasting task by using the test set of daily log-returns. Table 1 reports the Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and directional accuracy for all models. These metrics allow us

to assess how well each method predicts the magnitude and direction of future returns.

Model	RMSE	MAE	Directional Accuracy
GBM	0.012566	0.008347	0.5512
ARIMA	0.012566	0.008348	0.5512
GARCH	0.012567	0.008335	0.5512
LSTM	0.012381	0.008353	0.5102

Table 1: Performance of all models on forecasting.

The first observation is that all GBM, GARCH and ARIMA models produce almost identical error metrics. Their RMSE and MAE values are very close, and each achieves a directional accuracy of approximately 55%. This suggests that the underlying structure of daily returns is largely unpredictable, with only a small amount of short-term dependence that all three models capture to a similar degree.

The GARCH model provides richer information through its time-varying volatility estimates. Although this does not largely improve return accuracy, it successfully reflects volatility clustering in the data. This behavior is consistent with properties of financial time series.

The LSTM model achieves a slightly lower RMSE, and this indicates that it fits the magnitude of returns barely better than the classical models. However, its directional accuracy is close to 50%, which is essentially no better than random choice. This result suggests that the LSTM may be overfitting noise rather than learning stable predictive patterns, despite its flexibility and higher model capacity.

Overall, the experiments show that none of the models can reliably predict one-day-ahead stock returns with substantially better accuracy than the others. This aligns with the weak-form Efficient Market Hypothesis, which states that past price information alone contains little predictive power for short-term returns.

6.3 Why the Models Perform Similarly

Daily stock returns, especially for large indices such as SPY are known to contain very little predictable structure. In finance, it is accepted that daily returns behave almost like noise and shows only weak autocorrelation. As a result, models such as GBM, ARIMA, and GARCH all receive nearly the same information from the data and therefore achieve very similar forecasting accuracy.

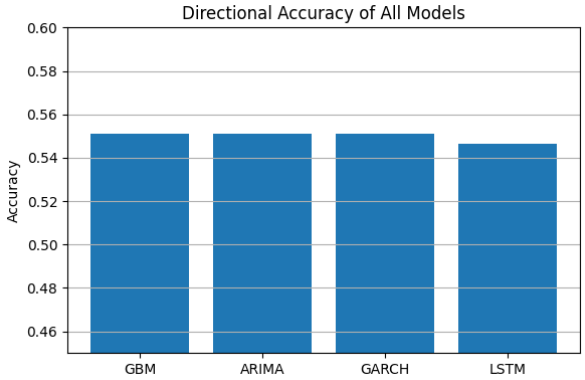

Figure 2: Directional Accuracy Comparison

GARCH improves volatility modeling but does not improve return-level forecasts, since changes in volatility do not translate into predictable changes in the mean of returns. Similarly, more deep models such as LSTMs cannot extract additional patterns from the data, because nonlinear dependencies are extremely limited at the daily frequency. All models are exposed to the same noise dominated structure of real financial markets, where predictable patterns are extremely limited. As a result, differences in model complexity do not translate into meaningful improvements in sample forecasting accuracy. This explains why all models perform nearly the same in our project.

6.4 Why the LSTM Does Not Outperform Classical Models

Although LSTM is a powerful nonlinear sequence model, it do not outperform the simpler statistical methods in this setting. Daily log-returns contain almost no stable or usable patterns but very weak autocorrelation and volatility clustering. When evaluated on real-world financial data, the complexity of the LSTM does not provide an advantage over classical models. Daily SPY log-returns contain little stable structure for the network to learn, and this causes the model to fit noise rather than meaningful patterns. Because of this, an LSTM has no meaningful structure to learn and it tends to overfit noise in the training data. This results in slightly lower RMSE but worse directional accuracy on the test set. The similar performance across all models is therefore not a limitation of the LSTM itself, but a consequence of the unpredictability of daily stock returns. This shows that testing on real market data is important since theoretical model capacity alone does not guarantee better predictive performance.

7 Conclusion

In this project, we constructed, analyzed and compared several models for return forecasting, including GBM, ARIMA, GARCH, and LSTM. Using daily log-returns from the SPY ETF, all models were evaluated on the same one-step-ahead prediction task. The results show that all classical models achieve nearly identical forecasting accuracy, and even the LSTM model does not provide a significant improvement. These findings are consistent with the Efficient Market Hypothesis, which states that past price information offers very limited predictive power for short-term returns.

While more advanced models can capture volatility dynamics or nonlinear structure in other contexts, our results illustrate that for daily returns, predictive performance is constrained by the noise-like behavior of the data. This highlights the inherent difficulty of short-term financial forecasting.

An important aspect of this project is that all models were evaluated using real world financial data rather than idealized or simulated ones. We tested GBM, ARIMA, GARCH, and LSTM on daily log-returns of the SPY ETF, and the analysis shows the practical challenges of forecasting in actual financial markets. The results proves that increased model complexity does not necessarily lead to better predictive performance in real life, where noise and uncertainty dominate short-term price movements.

References

[1] I. Karatzas and S. E. Shreve, *Brownian Motion and Stochastic Calculus*, 2nd ed. Springer, 2014.

[2] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[3] W. McKinney, *Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter*, 3rd ed. O'Reilly Media, 2022.

[4] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PLOS ONE*, vol. 13, no. 3, 2018.

[5] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 3rd ed. OTexts, 2021.

[6] R. Cont, "Empirical properties of asset returns: Stylized facts and statistical issues," *Quantitative Finance*, vol. 1, no. 2, pp. 223–236, 2001.