UNIVERSIDADE DE CABO VERDE
FACULDADE DE CIÊNCIAS E TECNOLOGIA
MESTRADO em MATEMÁTICA e APLICAÇÕES

Gestão de Portfólio

Otimização e Previsão

João Dantas Gomes Vaz
Orientador | Evgeny Lakshtanov

Com o apoio da Fundação Calouste Gulbenkian

Praia, Abril de 2022

UNIVERSIDADE DE CABO VERDE
FACULDADE DE CIÊNCIAS E TECNOLOGIA
MESTRADO em MATEMÁTICA e APLICAÇÕES

Gestão de Portfólio

Otimização e Previsão

João Dantas Gomes Vaz
Orientador | Evgeny Lakshtanov

Praia, Abril de 2022

O Júri:

Presidente _____

Arguente _____

Orientador _____

Evgeny Lakshtanov, Departamento de Matemática,
Universidade de Aveiro 3810-193, Portugal.

Dedicatória

Aos Meus Pais
Maria Helena Vaz e Eugénio Vaz
(in memoriam)

Agradecimentos

Este processo de crescimento pessoal e acadêmico não teria sido possível sem a influência positiva de diversos intervenientes. Assim, não poderia deixar de destacar todos aqueles que, direta ou indiretamente, contribuíram para a conclusão desta Dissertação.

Em primeiro lugar, quero expressar o meu mais profundo apreço aos meus amores *Suely Vasconcelos* e *Jayla Eveline*, eternamente grato pelo apoio e amor incondicional.

Agradecimentos ao Orientador *Dr. Evgeny Lakshtanov* pelos conselhos construtivos e conversas agradáveis.

Apresento os meus agradecimentos ao Professor *Dr. Uwe Kaehler* pelas sugestões engenhosas e conselhos valiosos.

Muito obrigado à todos os professores deste Mestrado pelas contribuições úteis e práticas.

Agradecimentos aos colegas do Mestrado, em especial, *José Moreira*, *Kelton Garcia* e *Alexsandro Baptista* pelos desabafos, apoios e amizade constante durante a jornada.

Gostaria de estender os meus sinceros agradecimentos à Direção da Escola Secundária Abílio Duarte, em nome da Diretora, *Sra. Ângela Martins*, pelo suporte implacável e crença profunda no meu trabalho.

Aos amigos e familiares um muito obrigado pelo carinho e pelos apoios.

A todos os que estiveram do meu lado, fazendo-me sentir que há percursos que vale a pena serem realizados - “ Many Thanks”!

Resumo

O modelo de otimização de portfólio de *média-variância*, modelo introduzido por Markowitz em 1952, e que fornece as bases do que é conhecida como *teoria moderna do portfólio*, é um modelo aplicável ao setor financeiro e que fornece respostas fundamentais para o problema de gestão de portfólios.

Este modelo visa determinar como distribuir o capital empregue entre ativos num portfólio de modo a que, para um nível de risco fixado à partida, possamos maximizar o retorno esperado e, simultaneamente, para um determinado nível de retorno, o risco associado possa ser minimizado. Para isso, procura-se um portfólio adequado na *fronteira eficiente*, que combine os melhores "trade-offs" entre os dois objetivos conflituosos, *maximizar o retorno* e *minimizar o risco*.

Neste estudo, propomos o uso de simulações de Monte Carlo, bem como de algoritmos de *aprendizado de máquina* (Machine Learning), que permitam otimizar os retornos de portfólio, e adicionalmente permitam também estimar a evolução dos preços dos ativos em tempo futuro.

Para testar estes algoritmos, criámos um problema modelo. Por um lado, os resultados confirmaram que uso dos algoritmos de aprendizado de máquina permite aos investidores otimizar e reestruturar o portfólio de modo eficiente, bem como prever o desempenho do portfólio nos mercados de ações. Por outro lado, o uso de *simulações de Monte Carlo*, baseado na hipótese de que os preços das ações seguem um *movimento Browniano geométrico*, evidenciou que estas têm um forte potencial para competir favoravelmente com os algoritmos de aprendizado de máquina propostos.

Palavras-chave: média-variância; teoria moderna do portfólio; retorno; risco; fronteira eficiente; simulação de Monte Carlo; aprendizado de máquina; movimento Browniano geométrico.

Abstract

Mean-variance analysis is an optimization model introduced by Markowitz in 1952, which provides the basis for what is known as *modern portfolio theory*. This is a model applicable to the financial sector which provides fundamental answers to the problem of portfolio management.

This model aims to determine how to distribute the capital between different assets in a portfolio, such that, for a fixed risk level, one can maximize the expected return while, at the same time, one minimizes the associated risk for a given level of return. This is done by looking for an appropriate portfolio at the efficient frontier, i.e. the ideal curve that combines the best trade-offs between the two conflicting objectives *maximizing return* versus *minimizing risk*.

In this work we propose the usage of Monte Carlo simulation techniques, as well as of algorithms of *Machine Learning*, which allow for an optimization of the portfolio's return, and can be also used for estimations of the prices of the assets in the future.

To test these algorithms we presented a model problem. On the one hand, the results confirm that the use of machine learning algorithms allows the investor to efficiently optimize and restructure its portfolio, as well as predict the performance of said portfolio in the stock market. On the other hand, the usage of Monte Carlo simulations (based on the hypothesis that prices follow a Geometric Brownian motion) show potential to compete favourably with the used Machine Learning algorithms.

Keywords: mean-variance; modern portfolio theory; return; risk; efficient frontier; Monte Carlo simulation; Machine Learning; geometric Brownian motion.

Conteúdos

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

X_0	Montante inicial
X_{t_0}	Quantia investida em t_0
X_{t_1}	Quantia recebida em t_1
$T = t_1 - t_0$	Período de retenção do ativo
R_i	Retorno total do ativo i
R	Retorno total do portfólio
r_i	Taxa de retorno do ativo i
r_p	Taxa do retorno do portfólio
r_f	Taxa anual de retorno livre de risco

$\mu_p = E[r_p] = E(r_p)$	Retorno esperado do portfólio
μ_i	Retorno esperado no ativo i
w_i	Montante investido no ativo i
κ	Parâmetro de aversão ao risco

Σ	Matriz de covariância dos retornos
$Var(r_p) = \sigma_p^2$	Variância do retorno do portfólio
σ_p	Desvio padrão do retorno do portfólio
$Cov(r_i,r_j) = \rho_{ij}\sigma_i\sigma_j$	Covariância de r_i e r_j
$\sigma_i\sigma_j$	Desvio padrão dos ativos i e j
ρ_{ij}	Correlação entre ativos i e j
$L(\mathbf{w}, \lambda_1, \lambda_2)$	Lagrangiano, onde λ_1 e λ_2 são os multiplicadores de Lagrange
$Z \sim N(0,1)$	Z segue uma distribuição normal padrão (média 0 e variância 1).

$$\mathbf{r} = \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}, \ \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}, \quad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \ \Sigma = \begin{bmatrix} \sigma_{11} & \dots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \dots & \sigma_{nn} \end{bmatrix}$$

Lista de Abreviaturas

TMP	Teoria Moderna do Portfólio
ML	Machine Learning
AM	Aprendizado de Máquina
SR	Sharpe Ratio (índice de Sharpe)
PQ	Programa Quadrático
PPQC	Problema de Programação Quadrática Convexa
SMC	Simulação de Monte Carlo
MC	Monte Carlo
MBG	Movimento Browniano Geométrico
HC	Hill Climb
SA	Simulated Annealing
AG	Algoritmo Genético
MAE	(em inglês, Mean Absolute Error) erro médio absoluto
MAPE	(em inglês, Mean Absolute Percentage Error) erro médio absoluto percentual

Capítulo 1

Introdução

1.1 Contextualização

O problema do investimento é comum às famílias e às empresas. Muitas vezes, as famílias visam simplesmente proteger as suas economias da inflação, ou ganhar dinheiro adicional, sem colocar em risco as suas poupanças. Já as empresas enfrentam estratégias de investimento mais complexas, procurando oportunidades de retorno elevado com risco aceitável. Ao longo dos anos, o número de oportunidades de investimento aumentou drasticamente, dada a globalização dos mercados financeiros, e também motivado pela criação de novos instrumentos de investimento. Assistimos atualmente a uma dinâmica de volatilidade no mercado. Aliás, os mercados financeiros são ambientes totalmente caóticos, o que contribui significativamente para a dinâmica de volatilidade dos mesmos [10]. Por conseguinte, a teoria financeira continua a ser um dos principais campos económicos onde a tomada de decisão, sob um ambiente de incerteza, desempenha um papel crucial [51]. Daí a necessidade de reforçar os investimentos, criando portfólios financeiros. Um portfólio que é um conjunto de ativos cuja seleção se baseia em objetivos e restrições económicas específicas, e visa gerar lucros para o investidor. Portanto, a correta gestão de portfólios financeiros é de extrema importância, no sentido de encontrar uma combinação de ativos que melhor satisfaça as necessidades e exigências do investidor.

Isto inclui alguns aspetos, tais como a análise da atitude do investidor em relação ao risco e o retorno esperado [41]. É geralmente aceite que um portfólio é projetado com base numa tolerância ao risco, por parte do investidor, com prazos e metas de investimento. A atribuição de pesos aos ativos no portfólio, referido como *alocação de ativos*, influencia a relação risco-retorno do portfólio. Os ativos específicos num portfólio e o peso de cada ativo são projetados para maximizar o retorno esperado, ao mesmo tempo que visam minimizar o risco [22]. No entanto, o problema de criar portfólio financeiros não é trivial. De facto, um dos maiores problemas atuais no setor financeiro é a criação e gestão de um portfólio de investimentos eficiente no ambiente complexo da sociedade globalizada, onde a concorrência aumenta rapidamente e se expandem as mudanças económicas a nível nacional e internacional [16]. A principal questão, na gestão

de portfólios, é de decidir sobre ativos e pesos para um melhor e mais eficiente investimento [42, 43]. Notar que nada é livre no mercado, ou seja, num mercado competitivo, os retornos acima das expectativas têm um preço: a necessidade, e capacidade, de suportar investimentos de maior risco (ver [6]).

Uma resposta fundamental para o problema de gestão de portfólio foi dada pelo modelo de média-variância [42, 43], e que deu origem à teoria moderna do portfólio. Esta é uma teoria aplicada no setor financeiro que evoluiu desde a década de cinquenta, e que descreve de forma rigorosa, e com base em conceitos matemáticos, a construção de um portfólio ótimo baseado em parâmetros de risco-retorno. Este modelo envolve selecionar ativos com base na correlação dos seus históricos de retorno, por forma a funcionarem para diversificar o portfólio [22]. Nesta teoria, o modelo de média-variância originalmente introduzido por Markowitz em 1952, e que lhe valeu o prémio nobel da economia, ganhou ampla aceitação como uma das ferramentas práticas para otimização de portfólios. O trabalho pioneiro de Markowitz revolucionou a forma como as pessoas pensam sobre o portfólio e inúmeros estudos sobre a otimização de portfólio têm sido realizados com base neste [30]. É amplamente considerado como uma das teorias fundamentais da economia financeira [31].

É uma teoria de um período único na escolha de pesos que proporcionam o melhor equilíbrio entre a média (como medida de retorno) e a variância (como medida de risco) do portfólio. A ideia principal subjacente da teoria é que, ajustando os pesos dos ativos individuais, podemos construir portfólios ótimos que oferecem o máximo retorno esperado possível para um nível de risco pré-determinado. Em consequência, o processo de criação e a gestão de portfólios de ativos tem-se desenvolvido significativamente nas últimas décadas [16].

O investidor investe em busca de um retorno futuro previsto, mas este retorno raramente pode ser previsto com precisão. Há (quase) sempre riscos associados aos investimentos. O retorno efetivo ou realizado quase sempre se desvia do retorno esperado [6]. Na presença de incerteza, a previsão e a otimização do portfólio utilizando o *método de aprendizado da máquina* (em inglês, *Machine Learning*) está a tornar-se cada vez mais popular, uma vez que os investidores podem não só prever o desempenho dos mercados e dos seus investimentos, como também podem usar os algoritmos de Machine Learning (ML) para otimizar os seus retornos [53]. De acordo com Tadlaoui [58], construir resultados ótimos do portfólio a partir de um problema de otimização - onde muitos parâmetros desconhecidos, como por exemplo, os retornos esperados e a sua correlação, têm de ser estimados - implica uma elevada sensibilidade e precisão das metodologias de estimativa.

1.2 Metodologias Para Construção de Portfólios Ótimos

Pretendemos construir portfólios ótimos a partir de um certo problema de otimização. Esta ideia é melhor estruturada no livro [40], sendo que a seleção de ativos não é apenas um problema de encontrar investimentos atrativos [17]. O projetar de um portfólio não pode ser feito apenas baseado na intuição humana, sob pena de ter um risco muito elevado para o investidor. Tal requer uma rigorosa teoria matemática, programas eficientes, em rápido tempo, e fiáveis. Tais programas são chamados *otimizadores* e, de acordo com Raja [53], que reagem às variações do ambiente circundante (o mercado financeiro). No mundo da Inteligência Artificial, a presença de métodos que possam lidar com incerteza pode ajudar ao objetivo primordial de maximização dos retornos. É neste contexto que surgem os problemas que abordaremos nesta dissertação e segundo o modelo média-variância:

1. Como ajustar os pesos dos ativos do portfólio de forma eficiente?
2. Quais os algoritmos que melhor ajudam o investidor a prever o desempenho do mercado (e, consequentemente, a planear os seus investimentos)?
3. Como podem os algoritmos de aprendizado da máquina ajudar o investidor no problema anterior?
4. Como comparar diferentes algoritmos (Monte Carlo versus aprendizado da máquina)?

Há vários métodos e técnicas que podem ser utilizadas para construir um portfólio ótimo. Para ajudar a responder às questões acima, propomos as seguintes metodologias (apoiadas em raciocínio matemático):

- Ajustar de pesos em portfólios, o que pode ser feito
  - recorrendo à simulação de Monte Carlo;
  - usando algoritmos de otimização para esse ajuste;

  e sua análise por meio da *índice Sharpe* para análise do desempenho dos retornos;
- Identificar portfólios ótimos usando o conceito de *fronteira eficiente*;
- Análise das previsões e simulação de preços futuros, com base em
  - uso de simulação de Monte Carlo (assumindo que o mercado cumpre um movimento Browniano);
  - ou em implementação de algoritmos de Machine Learning;

  e comparação, e avaliação, do desempenho das previsões obtidas.

1.3 Estrutura da Dissertação

Esta dissertação encontra-se estruturada em 6 capítulos, e 2 apêndices. Após este capítulo introdutório, seguem-se os capítulos centrais, cujo conteúdo é sumarizado em seguida.

O segundo capítulo fornece o enquadramento teórico que suporta a dissertação, abordando o estado da arte, sumarizando a teoria moderna do portfólio, isto é, o conceito de média-variância e de alguns outros conceitos fundamentais na teoria de portfólios. Nomeadamente, introduzimos os conceitos de retorno e de risco, de diversificação, de fronteira eficiente e de índice Sharpe.

No terceiro capítulo começamos por descrever o problema geral de otimização, concentrando-nos depois no problema de otimização financeira. Apresentamos então o esquema de programação quadrática convexa e sua aplicação ao problema de otimização do portfólio, com uma descrição detalhada deste.

No quarto capítulo começaremos por rever os objetivos que pretendemos numa gestão do portfólio. Descreveremos sucintamente um problema modelo que ilustra o modelo média-variância. Introduzimos à linguagem de programação que iremos usar, bem como a descrição da base de dados utilizada para testar os algoritmos. Também introduzimos os métodos matemáticos que vamos usar (curta descrição dos métodos de Monte Carlo e dos algoritmos de Aprendizado de Máquina). Terminamos este capítulo com a indicação das métricas de desempenho a usar neste trabalho.

No quinto capítulo mostramos, com vários exemplos, a análise dos resultados obtidos por aplicação dos métodos descritos no capítulo anterior.

No sexto capítulo apresentamos as considerações finais, e comentários às limitações do método descrito neste trabalho, e terminamos com uma perspetiva para trabalho futuro.
Seguem-se então as referências bibliográficas utilizadas na dissertação, bem como os apêndices nos quais estão incluídos os materiais adicionais necessários ao bom entendimento.

Capítulo 2

Revisão da Literatura

Neste capítulo apresentamos o enquadramento teórico que suporta a dissertação, abordando o estado da arte, sumarizando a teoria moderna do portfólio, isto é, o conceito de média-variância e de alguns outros conceitos fundamentais na teoria de portfólios. Nomeadamente, introduzimos os conceitos de retorno e de risco, de diversificação, de fronteira eficiente e de índice Sharpe.

2.1 Estado da Arte

A teoria do portfólio foi desenvolvida por Harry Markowitz na década de cinquenta e fornece a base para a chamada *teoria moderna de portfólio*. O modelo pioneiro, da sua autoria, fornece um esquema para encontrar o equilíbrio ideal entre o risco e o retorno de dado investimento [42, 43]. Desde então, este tema tem sido abordado e estendido por diversos profissionais e académicos.

O modelo de Markowitz revelou-se, com o tempo, demasiado limitado, tendo em conta os pressupostos iniciais. Estes incluem uma elevada sensibilidade ao histórico dos dados e não permitem incluir as opiniões dos investidores. Assim, foram criados outros modelos - ainda baseados no de Markowitz - mas que permitem ultrapassar estas dificuldades. Uma dessas extensões é o modelo Black-Litterman, desenvolvido nos anos noventa, e uma introdução a este modelo pode ser encontrada em [23].

Ainda mais tarde, o modelo de Markowitz foi modificado para incluir características mais realistas como, por exemplo, a *função de utilidade* que mede a preferência relativa do investidor por diferentes níveis de retorno e risco [28]. Outros modelos ainda, introduzem restrições de cardinalidade que impõem um limite pré-determinado no número de ativos [7, 57] e no custo de transação [25, 39, 45]; modelos mais sofisticados que utilizam extensões multiperiódicas ou dinâmicas (veja-se por exemplo, [33, 36, 37]) ou ainda medidas de risco para portfólios durante eventos extremos (ver [8]).

Entretanto, assiste-se atualmente a uma crescente inclinação para o uso de aprendizado das máquinas (*Machine Learning*). Desenvolvimentos recentes em aprendizado de máquina (AM)

trouxeram oportunidades significativas para incorporar teoria de previsão na gestão de portfólios [11]. É de notar que os algoritmos de aprendizado de máquina inspirados pela natureza têm ganho uma atenção considerável da comunidade científica, que se reflete no seu uso em várias áreas, veja-se, por exemplo, [3, 4, 12, 13, 24, 27, 29, 50, 52, 60, 61].

Além disso, um aprendizado de máquina pode ser especificamente desenvolvido e utilizado para otimizar portfólios que atendam às metas de desempenho de modo mais preciso, para efeitos de tomada de decisões de alocação de ativos [38]. Métodos de aprendizado de máquina são algoritmos proeminentes que estão a ser utilizados por várias instituições bancárias para prever o comportamento de séries temporais (usadas em finanças) e também para otimizar os retornos de certos investimentos [53, 58].

No modelo pioneiro de Markowitz, a média refere-se ao retorno esperado, sendo a variância considerada uma medida de risco, e que deu origem ao nome de *modelo de otimização do portfólio de média-variância*.

Embora outros modelos tenham sido propostos na literatura, o modelo de média-variância continua a ser a espinha dorsal da moderna teoria do portfólio, e é o modelo comumente aplicado e citado [45].

2.2 Teoria Moderna do Portfólio

A Teoria Moderna do Portfólio (TMP) é uma teoria financeira que tenta, por um lado, maximizar o retorno esperado de um portfólio, para um nível de risco fixado e, por outro lado, minimizar o risco para um certo nível de retorno considerado.

Esta teoria é habitual nos planos de reforma: quanto mais jovem o investidor e menor o montante investido, maior a vontade de correr riscos, que possam trazer maior retorno. À medida que o investidor se aproxima da idade de reforma e o valor total do portfólio aumenta, o mais provável é que assuma menos riscos, para garantir o investimento base do portfólio [22].

A TMP fornece um modelo de investimento matemático diversificado, com o objetivo de selecionar um conjunto de investimentos que tenham um risco combinado inferior ao de qualquer ativo individual. Portanto, a ideia básica por trás da TMP é de se ter uma diversificação de ativos, isto é, os ativos num portfólio não devem ser selecionados com base unicamente no seu desempenho individual.

Vamos agora introduzir os conceitos necessários para o bom prosseguimento deste trabalho.

2.2.1 Conceitos Necessários

Definição 2.2.1 (Portfólio)**.** *Chamamos portfólio a todo o conjunto de ativos negociáveis.*

O termo portfólio refere-se assim a qualquer combinação de ativos financeiros, sejam estes obrigações, ações ou títulos detidos por um ou vários investidores. É de todo o interesse do investidor que tenha diversificação na combinação dos ativos. O termo *diversificação* é bem

conhecido na comunidade financeira pois uma combinação de diferentes tipos de ativos é menos arriscada do que ter apenas um tipo (ver [43]). Diversificação significa a manutenção de vários ativos no portfólio de forma a limitar a exposição ao risco de um ativo específico [6]. A ideia é criar um portfólio que inclua vários investimentos, para reduzir o risco. No sentido mais geral, pode resumir-se no velho ditado: "Não coloque todos os ovos no mesmo cesto".

Definição 2.2.2 (Pesos)**.** *As proporções de investimentos atribuídas aos ativos no portfólio são chamadas pesos dos ativos.*

Por outras palavras, os pesos indicam a quantidade de capital investido em cada ativo que compõe um dado portfólio.

Para uma correta atribuição de pesos aos ativos, há necessidade de "medir" os retorno e risco associados. Duas das métricas mais importantes para esse efeito, e comumente usadas quando se referem aos preços dos ativos[1], são o *retorno* e a *volatilidade* do ativo.

Por retorno, geralmente queremos dizer a taxa de retorno, de que as formas mais comuns são a *taxa de retorno simples* e o *retorno logarítmico*[2]. Já por volatilidade, referimo-nos ao *desvio padrão do retorno*, ou sua *variância*. A variância é então uma medida de quão espalhado é o conjunto dos dados.

No caso mais simples, os retornos r_i de um ativo i podem ser descritos com a distribuição normal; o valor esperado (ou média) desse retorno é denotada por $E(r_i)$, e sua variância, por σ_i^2, (em literatura financeira, é usada a raíz quadrada, σ_i, referida como volatilidade). Estes captam toda a informação sobre o resultado esperado e a probabilidade e amplitude de desvios em relação a ela [41].

Para o nosso modelo simplificado, há apenas dois períodos de tempo: o tempo inicial t_0 e o tempo final t_1. Cada ativo i tem um valor inicial X_{t_0}, e um valor final X_{t_1}. Chamamos ao período de tempo $T = t_1 - t_0$ o *período de retenção* do ativo. Na ausência de fluxos de caixa intermediários (por exemplo, de dividendos a tempos intermédios), o retorno total de um investimento é definido como

$$\textbf{retorno total} = \frac{montante\ recebido}{montante\ investido}$$

Ou, se X_{t_0} e X_{t_1} são, respetivamente, as quantias de dinheiro investido e recebido, e R é o retorno total, então

$$R = \frac{X_{t_1}}{X_{t_0}}.$$

Muitas vezes, para simplificar, o termo *retorno* é usado para retorno total.

[1] Um instrumento financeiro que pode ser comprado e/ou vendido é muitas vezes chamado de *ativo*.

[2] É muitas vezes mais conveniente usar o retorno logarítmico $\boxed{r_{\log} = log(1+r)}$, porque se encaixa melhor em pressupostos de muitos modelos de preços estocásticos.

A **taxa de retorno** é dada por

$$\textbf{taxa de retorno} = \frac{montante\ recebido - montante\ investido}{montante\ investido}$$

Mais uma vez, se X_{t_0} e X_{t_1} são, respectivamente, as quantias de dinheiro investido e recebido, e r é a taxa de retorno, então

$$r = \frac{X_{t_1} - X_{t_0}}{X_{t_0}} \tag{2.1}$$

Distinguimos as duas definições usando letras maiúsculas ou minúsculas, R e r, respectivamente, para o retorno total e taxa de retorno; geralmente o contexto torna as notações claras e permite que usemos a expressão "retorno" em ambos os casos, sendo claro do contexto.

É claro que as duas noções estão relacionadas por

$$R = 1 + r,$$

e a equação (2.1) pode ser reescrita como

$$X_{t_1} = (1 + r)X_{t_0}.$$

Isto mostra que uma taxa de retorno age como uma taxa de juro.

De notar também que o período de retenção pode ser em qualquer unidade de tempo de calendário: dias, semanas, meses, anos ou qualquer unidade adequada para este assunto.
No caso de ações, se o período de retenção T for de um dia de negociação, e X_{t_1} for o preço do fecho de hoje, então X_{t_0} denotará o preço de fecho de mercado no dia imediatamente anterior - ontem e então (2.1) representa o retorno diário das ações.

Antes de passarmos à descrição do modelo, iremos descrever o que se pretende com ele nas duas subseções seguintes.

2.2.2 Retorno do Portfólio

Suponha-se agora que n ativos diferentes estão disponíveis para formar um portfólio. Suponha ainda que isto seja feito distribuindo o investimento (uma quantidade X_0) entre os n ativos. Cada investimento individual no ativo i será designado por X_{0_i}. Assim os valores X_{0_i}, $i = 1,\ldots,n$, são tais que $\sum_{i=1}^{n} X_{0_i} = X_0$, sendo X_{0_i} o montante investido no i-ésimo ativo. Supomos adicionalmente que não são permitidas *vendas a descoberto* - venda de ativos que não foram ainda comprados (em inglês, *short selling*) - e isto para evitar que alguns dos $X'_{0_i}s$ possam tomar valores negativos.

Os montantes investidos no ativo i podem ser expressos em frações w_i do investimento total X_0, isto é,

$$X_{0_i} = w_i X_0, \quad i = 1,\ldots,n,$$

Dizemos que w_i é o *peso* ou *fração* desse ativo no portfólio. Claramente,

$$\sum_{i=1}^{n} w_i = 1.$$

Seja agora R_i o retorno total do ativo i. Assim, a quantidade de dinheiro gerada no final do período de retenção T pelo i-ésimo ativo é dado por $R_i X_{0_i} = R_i w_i X_0$. O valor total gerado por este portfólio no final do período T é, portanto, $\sum_{i=1}^{n} R_i w_i X_0$. Daí encontramos que o retorno total do portfólio é

$$R = \frac{\sum_{i=1}^{n} R_i w_i X_0}{X_0} = \sum_{i=1}^{n} w_i R_i$$

Além disso, temos que a taxa de retorno do ativo i é $r_i = R_i - 1$, $i = 1, 2, \ldots, n$. Daí que a taxa de retorno do portfólio seja

$$r_p = R - 1 = \left(\sum_{i=1}^{n} w_i R_i\right) - \left(\sum_{i=1}^{n} w_i\right) = \sum_{i=1}^{n} w_i (R_i - 1) = \sum_{i=1}^{n} w_i r_i.$$

Repetindo o procedimento para o número de dias de negociação T_y num ano (por exemplo, $T_y = 252$ num total dos 365 dias) e $r_1, r_2, \ldots, r_n$ representarem os retornos diários dos ativos, então o valor do retorno anual do portfólio (r_{ap}) seria dado por:

$$r_{ap} = \sum_{i=1}^{n} w_i r_i T_y.$$

Uma vez que o valor de um ativo em tempo futuro é uma variável aleatória r_i, então o valor esperado (média) do retorno total do portfólio é dado por

$$E(r_p) = \mathbf{w}^\intercal \boldsymbol{\mu},$$

onde $\mu_i = E(r_i)$ é o retorno esperado de cada ativo individual, $\mathbf{w} = (w_1, w_2, \ldots, w_n)^\intercal$ é o vetor dos pesos, e $\mathbf{r} = (r_1, r_2, \ldots, r_n)^\intercal$, o vetor retorno do portfólio.

Uma vez que os retornos individuais geralmente não são independentes, a variância do portfólio (σ_p^2) dependerá da covariância entre os retornos dos diferentes ativos.

2.2.3 Risco do Portfólio

Como já dizemos atrás, o risco de um dado portfólio é medido pela variância σ_p^2 deste.

Definição 2.2.3 (Risco do Portfólio)**.** *O risco de um portfólio p, designado por σ_p, desvio padrão, isto é:*

$$\sigma_p = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j Cov(r_i, r_j)} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \rho_{ij} \sigma_i \sigma_j}, \tag{2.2}$$

onde ρ_{ij} representa a correlação entre os retornos dos ativos i e j, e cada σ_i, o desvio padrão do retorno do ativo i, com i,j = $1, \ldots, n$.

Mais detalhes, podem ser encontrados em, por exemplo, [40, 48].

Sendo conhecida a covariância $Cov(r_i,r_j)$ entre os retornos diários dos ativos i e j, e o período T_y de dias de negociação num ano, então o risco anual do portfólio σ_{ap} é agora:

$$\sigma_{ap} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j Cov(r_i,r_j) T_y}.$$

Estas covariâncias surgem, pois, as flutuações nos preços dos diversos ativos não são (em geral) independentes entre si, pois estão expostos a fontes comuns de risco. Assim, tornam-se ativos correlacionados. Para Pai [48] um portfólio é normalmente exposto a dois tipos de riscos, nomeadamente, o *risco sistemático* e *risco idiossincrático*. Infelizmente, só podemos mitigar o risco idiossincrático. Para mais detalhes sobre riscos, ver [46].

Vamos agora abordar a razão por que a variância é uma boa medida para o risco do portfólio, e constitui o ponto de partida para o conhecido modelo de otimização do portfólio média-variância [17, 45].

2.3 Fundamento Matemático Subjacente ao Modelo

A TMP assume os retornos de um investimento individual como uma distribuição normal. Para um portfólio de n ativos a taxa de retorno do i-ésimo ativo é uma variável aleatória r_i com média $E(r_i) = \mu_i$, $i = 1, \ldots, n$, pesos $\mathbf{w} = (w_1, w_2, \ldots, w_n)^\intercal$, e retornos $\mathbf{r} = (r_1, r_2, \ldots, r_n)^\intercal$, com retorno do portfólio dado por

$$r_p = \mathbf{w}^\intercal \mathbf{r}. \tag{2.3}$$

Considere-se agora o retorno esperado dos ativos $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)^\intercal$. Então o retorno esperado do portfólio é dado por

$$E(r_p) = E\left(\sum_{i=1}^{n} w_i r_i\right) = \sum_{i=1}^{n} w_i E(r_i) = \sum_{i=1}^{n} w_i \mu_i = \mathbf{w}^\intercal \boldsymbol{\mu}.$$

Por outro lado, a variância é dada por:

$$\sigma_p^2 = E\left((r_p - E(r_p))^2\right),$$

e mede as flutuações da variável r_p em torno de sua média $E(r_p)$. Deste modo, valores altos para a variância indicam investimentos mais arriscados, enquanto que valores mais baixos indicam um risco menor.

2.3.1 Espaço de Probabilidades Associado

Vamos agora formalizar este modelo, seguindo a ideia em [45]. Defina-se um espaço de probabilidades $(\Omega,\mathcal{F},\mathbb{P})$, onde Ω é um espaço amostral, $\mathcal{F}$ é um espaço mensurável em Ω e $\mathbb{P}$ é uma medida de probabilidade em $\mathcal{F}$.

Considere-se, de seguida, $\mathcal{L}^2(\Omega)$ como o espaço L^2 dos retornos aleatórios do portfólio. Sejam $r_p, Y \in \mathcal{L}^2(\Omega)$, onde Y representa uma variável aleatória constante nesse espaço $\mathcal{L}^2(\Omega)$, ou seja $E(Y) = Y$.

Teorema 2.3.1. *Seja f um funcional definido por*

$$f(r_p,Y) = E((r_p - Y)^2).$$

Então f atinge o mínimo em $Y = E(r_p)$ e esse mínimo é o valor da variância de r_p.

Demonstração. Primeiro, observe-se que, porque Y representa variável aleatória constante, temos

$$f(r_p,Y) = E(r_p^2) - 2E(r_p)Y + Y^2.$$

Portanto, derivando f e igualando a zero, obtemos $Y = E(r_p)$, isto é, o mínimo é atingido na média do retorno do portfólio, e assume o valor

$$E((r_p - E(r_p))^2) = \sigma_p^2,$$

isto é, o valor da variância de r_p. □

Assim, o Teorema 2.3.1 mostra que a variância estabelece uma distância mínima (no sentido do quadrado médio) de uma variável aleatória relativamente ao seu valor esperado. Um portfólio p com variância mínima é, portanto, aquele que dá a menor distância (no sentido de quadrado médio) entre o retorno do portfólio r_p e o seu retorno esperado $E(r_p)$, e consequentemente, que apresenta menor risco.

2.3.2 Fronteira Eficiente

Vamos agora representar o espaço amostral Ω das variáveis retornos aleatórios dos portfólios, r_p, num gráfico onde o eixo Ox representa a *volatilidade* destas, e o eixo Oy, os seus retornos. Assim, o gráfico obtido mostrará a posição de cada portfólio em função dos seus volatilidade-retorno.

Chamamos *fronteira eficiente* à linha ideal que delimita este gráfico à esquerda, com inclinação ascendente. Esta indicando uma curva delimitadora do gráfico que expressa a relação risco-retorno e que mostra o conjunto de portfólios ótimos que oferecem o maior retorno esperado para um determinado nível de risco, ou o menor risco para um determinado nível de retorno esperado. Portanto, a fronteira eficiente é a curva ideal que contém / conecta os portfólios ótimos.

O modelo de otimização do portfólio, busca encontrar os portfólios que estão nesta fronteira eficiente (ver [42]).

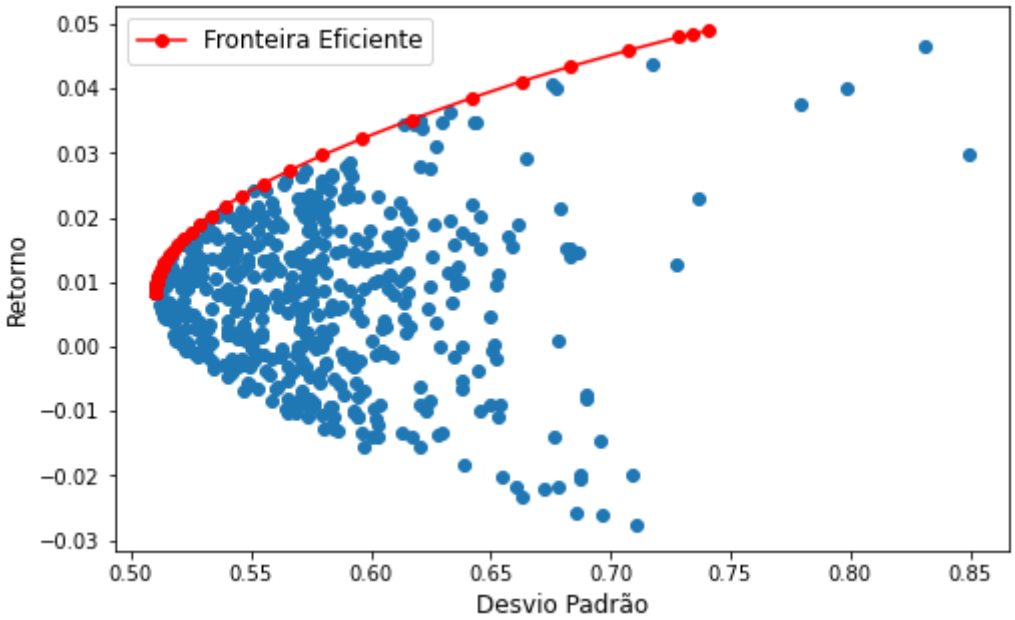


Figura 2.1: Fronteira eficiente

2.3.3 Índice de Sharpe

O índice de Sharpe (ou simplesmente Sharpe Ratio (SR)) fornece uma medida do cálculo do retorno ajustado ao risco. É definido como o quociente entre o retorno esperado do portfólio $\mathbf{w}^\intercal \boldsymbol{\mu}$ relativamente a taxa livre de risco r_f, por unidade de volatilidade σ_p (dita, desvio padrão do portfólio), ou seja.

$$SR = \frac{\mathbf{w}^\intercal \boldsymbol{\mu} - r_f}{\sigma_p}. \tag{2.4}$$

Esta é a medida de desempenho mais usada na literatura financeira. A sua contribuição teórica surge como uma das medidas pioneiras em estudos de avaliação de portfólios, com o trabalho seminal da Sharpe (1964) [56]. Este índice é particularmente importante porque mede o excesso de retorno sobre a taxa livre de risco em relação ao seu desvio padrão. Assim, na prática, ao invés de tentarmos minimizar a volatilidade para um determinado retorno, muitas vezes faz mais sentido apenas encontrar o portfólio que maximiza o índice de Sharpe, isto é, implementamos a busca do portfólio ótimo como uma busca do máximo deste índice

$$\max \left(\frac{\mathbf{w}^\intercal \boldsymbol{\mu} - r_f}{\sigma_p} \right)$$

sobre a fronteira eficiente [53]. Quanto maior for o valor do índice Sharpe, maior os retornos esperados, e indica que os ativos (mesmo que expostos a volatilidade extra) têm baixo risco. O índice de Sharpe perto de zero indica obviamente ativos sem retornos em excesso. Em resumo, um portfólio de pouco interesse para o investidor [48].

Capítulo 3

Otimização

Neste capítulo começamos por descrever o problema geral de otimização, concentrando-nos depois no problema de otimização financeira. Apresentamos então o esquema de programação quadrática convexa e sua aplicação ao problema de otimização do portfólio, com uma descrição detalhada deste.

3.1 Otimização

Dados uma função $f : \mathbb{R}^n \to \mathbb{R}$ (dita *função objetivo*), e um conjunto fechado $S \subseteq \mathbb{R}^n$ (*conjunto de restrições*), pretende-se encontrar o ponto $x^\star \in S$ onde f atinge o mínimo global, ou seja,

$$f(x^\star) \leq f(x), \quad \text{para todo o } x \in S.$$

O ponto $x^\star$ é chamado minimizador de f, correspondente a um mínimo $\big(x^\star, f(x^\star)\big)$.

O problema pode ser adaptado ao cálculo de máximo global. Todavia, basta considerar uma minimização, pois o máximo de f é atingido no mínimo de $(-f)$. A função objetivo f podendo ser linear ou não linear. No que se segue, usaremos funções convexas, isto é, para quaisquer dois pontos $x_1, x_2 \in \mathbb{R}^n$ e qualquer $t \in [0,1]$ temos

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

Normalmente, o conjunto de restrições S é definido por um sistema de equações ou inequações que podem também ser lineares ou não lineares. Se $S = \mathbb{R}^n$ o problema diz-se que não tem restrições. Os pontos do conjunto S são chamados pontos praticáveis.

Assim, formularemos o nosso problema de otimização como:

$$\min f(x) \;\text{ sujeito a }\; g(x) = 0 \;\text{ e }\; h(x) \leq 0$$

em que $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ e $h : \mathbb{R}^n \to \mathbb{R}^p$.
Falaremos de *programação linear* se f, g e h são todas lineares. De modo semelhante, teremos

programação não linear se pelo menos uma das funções f, g e h é não linear. Para mais detalhes ver, por exemplo, [9, 21]

3.2 Otimização em Finanças - Objetivos

A otimização de portfólios, ou seleção de portfólio ótimo, está principalmente preocupada em encontrar a melhor combinação de ativos que satisfaça as necessidades dos investidores. Cada ativo individual tem caraterísticas e história próprias, e o objetivo da otimização do portfólio é alocar capital entre esses ativos da forma mais eficiente (ver [48]).

Na realidade, a taxa de retorno esperada, que é obtida anexando uma probabilidade para cada taxa de retorno possível, é utilizada para calcular o retorno do portfólio. Como vimos atrás, o retorno do portfólio é dado pela soma ponderada dos retornos esperados, ou seja:

$$E(r_p) = \sum_{i=1}^{n} w_i \mu_i,$$

com um risco dado pelo desvio padrão na Definição 2.2.3.
O objetivo (ou objetivos gémeos) da otimização do portfólio acabam por ser:

Maximizar o retorno do portfólio / minimizar o risco do portfólio.

Assim, a otimização do portfólio é uma metodologia que serve para controlar o risco e retorno do portfólio. No entanto, modelos complexos de otimização de portfólio podem envolver múltiplas funções objetivos. Deste modo, um problema de otimização de portfólio em geral acaba sendo um problema de otimização com dois objetivos, ou mais complexa, de multi-objetivo [48].

3.3 Problema de Programação Quadrática Convexa (PPQC)

Nesta seção, as notações e definições seguem os livros [32, 47].

Definição 3.3.1. *Um programa quadrático (PQ) é um problema de otimização com função objetivo quadrática e restrições lineares*

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^\intercal \boldsymbol{q}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\intercal G\boldsymbol{x} + \boldsymbol{x}^\intercal \boldsymbol{c} \\ s.a \quad & \boldsymbol{a}_i^\intercal \boldsymbol{x} = b_i, \ \ i \in \mathcal{E}, \\ & \boldsymbol{a}_i^\intercal \boldsymbol{x} \geq b_i, \quad i \in \mathcal{I}, \end{aligned} \tag{3.1}$$

onde G é uma matriz simétrica de dimensão $n \times n$, $\mathcal{E}$ e $\mathcal{I}$ são conjuntos finitos disjuntos de índices e $\boldsymbol{c}, \boldsymbol{x}$, $\boldsymbol{a}_i^\intercal = (a_{i1}, a_{i2}, \ldots a_{in}) \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, com $i \in \mathcal{E} \cup \mathcal{I}$.

Os programas quadráticos sempre podem ser resolvidos em tempo finito de computação, mas o esforço necessário para encontrar uma solução depende fortemente das caraterísticas da função objetivo e do número de restrições de desigualdade. Se a matriz G é semidefinida positiva, dizemos que (3.1) é um *programa quadrático convexo*, e neste caso o problema é geralmente semelhante em dificuldade a um programa linear. Todavia, a dificuldade deste problema muda drasticamente se G não é semidefinida positiva.
A otimização de portfólios é apontada como uma das aplicações mais famosas do PPQC.

Problema 3.3.1. *Gostaríamos de investir o nosso dinheiro em n ativos num período fixo. Os retornos dos ativos geralmente não são conhecidos antecipadamente e são assumidos como variáveis aleatórias que seguem uma distribuição normal.*

Seja $\mathbf{r} \in \mathbb{R}^n$ o vetor aleatório de todos os retornos, de que assumimos conhecer a média $\boldsymbol{\mu} \in \mathbb{R}^n$ e covariância Σ. Sendo w_i a parte do nosso dinheiro investido no ativo i, então o retorno esperado do nosso portfólio seria

$$E[(\mathbf{w}^\intercal \mathbf{r})] = \mathbf{w}^\intercal \boldsymbol{\mu},$$

com variância

$$\begin{aligned} E[(\mathbf{w}^\intercal \mathbf{r} - (\mathbf{w}^\intercal \boldsymbol{\mu}))^2] =& E[(\mathbf{w}^\intercal (\mathbf{r} - \boldsymbol{\mu}))^2] = E[\mathbf{w}^\intercal (\mathbf{r} - \boldsymbol{\mu})(\mathbf{r} - \boldsymbol{\mu})^\intercal \mathbf{w}] \\ =& \mathbf{w}^\intercal E[(\mathbf{r} - \boldsymbol{\mu})(\mathbf{r} - \boldsymbol{\mu})^\intercal]\mathbf{w} \\ =& \mathbf{w}^\intercal \Sigma \mathbf{w}, \end{aligned} \tag{3.2}$$

e onde Σ é a matriz semidefinida positiva e simétrica $n \times n$ das covariâncias, definida por

$$\Sigma = [Cov(r_i, r_j)]_{i,j=1,\ldots,n} = [\rho_{ij}\sigma_i\sigma_j]_{i,j=1,\ldots,n}.$$

Os retornos não são, em geral, independentes entre si, e podemos definir as correlações entre pares de retornos da seguinte forma:

$$\rho_{ij} = \frac{E[(r_i - \mu_i)](r_j - \mu_j)}{\sigma_i \sigma_j}, \quad \text{para } i,j = 1,2,\ldots,n.$$

Obviamente, a correlação mede a tendência do retorno dos investimentos nos ativos i e j se moverem na mesma direção. Dois investimentos cujos retornos tendem a subir, ou descer, juntamente têm uma correlação positiva; por outro lado, investimentos cujos retornos tendem a mover-se em direções opostas têm uma correlação negativa. Quanto mais próximo ρ_{ij} estiver de 1, mais correlacionados estarão os dois investimentos (ambos crescem, ou ambos decrescem).

Na prática, $\boldsymbol{\mu}$ e $\Sigma = [Cov(r_i, r_j)]_{i,j=1,\ldots,n}$ podem ser (e são) estimados a partir dos histórico dos dados e substituímos, nos modelos, os valores exatos (desconhecidos) por o de estas estimativas.

Idealmente, gostaríamos de encontrar um portfólio para o qual o retorno esperado $\mathbf{w}^\intercal \boldsymbol{\mu}$ seja grande com variância $\mathbf{w}^\intercal \Sigma \mathbf{w}$ pequena. No modelo proposto por Markowitz [42], combinamos

estes dois objetivos numa única função objetivo com a ajuda de um "parâmetro de tolerância ao risco", denotado por κ, e resolvemos o seguinte problema para encontrar o portfólio ótimo:

$$\begin{aligned} \max_{\mathbf{w}} \quad & \mathbf{w}^\intercal \boldsymbol{\mu} - \kappa \mathbf{w}^\intercal \Sigma \mathbf{w} \\ \text{s.a} \quad & \sum_{i=1}^{n} w_i = 1, \text{ com } w_i \geq 0, i = 1, \ldots, n. \end{aligned} \tag{3.3}$$

O valor escolhido para o parâmetro $\kappa > 0$ depende das preferências individuais do investidor. Investidores conservadores, que dão mais ênfase à minimização de riscos no seu portfólio, escolheriam um grande valor de κ para aumentar o peso da variância medida na função objetivo. Os investidores mais ousados, preparados para assumir maior risco na esperança de um retorno esperado maior, escolheriam um valor menor de κ.

Em algumas variantes do problema, a restrição $w_i \geq 0$ é removida - (venda a descoberta (Short selling)[1] é permitida).

3.4 Modelação Matemática de um Portfólio

Apresentamos agora a modelação matemática do modelo média-variância proposto por Markowitz para a otimização de portfólios. Uma introdução semelhante à aqui apresentada pode ser encontrada em [6, 11, 16, 31, 41, 45, 58] ou em [42, 43].

Vamos considerar um portfólio p com n ativos, e onde w_i é o peso do ativo i no portfólio, com a restrição $\sum_{i=1}^{n} w_i = 1$. Sendo r_i a variável aleatória que representa a taxa de retorno do ativo i (assumido como normalmente distribuído), tome-se a sua média $\mu_i = E(r_i)$. Temos então,

$$\mathbf{r} = \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}, \; E[\mathbf{r}] = \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}, \; \Sigma = \begin{bmatrix} \sigma_{11} & \ldots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \ldots & \sigma_{nn} \end{bmatrix}$$

onde Σ é a matriz de covariância dos retornos dos ativos, com $\sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$, com i,$j = 1, \ldots, n$. Relembra-se que a matriz Σ é a matriz das covariâncias entre os retornos dos diferentes ativos. Assim, $\sigma_{ii} = \sigma_i^2$ (variância do ativo i) e $\sigma_{ij} = \sigma_{ji}$ (para $i \neq j$), isto é, a matriz de covariância Σ é semidefinida positiva e simétrica, ou equivalentemente, todos os valores-próprios são não-negativos.

Definindo o retorno do portfólio como uma variável aleatória normal r_p, o retorno esperado

[1] Às vezes é possível vender um ativo que não possuímos. Isto é chamado de venda a descoberto (short selling).

do portfólio $E(r_p)$, e a sua variância $\sigma_{r_p}^2$, então temos:

$$r_p = w_1 r_1 + \ldots + w_n r_n = \sum_{i=1}^{n} w_i r_i = \mathbf{w}^\intercal \mathbf{r} \tag{3.4}$$

$$E(r_p) = E(w_1 r_1 + \ldots + w_n r_n) = \sum_{i=1}^{n} E(w_i r_i) = \sum_{i=1}^{n} w_i E(r_i) = \sum_{i=1}^{n} w_i \mu_i = \mathbf{w}^\intercal \boldsymbol{\mu} \tag{3.5}$$

$$\sigma_{r_p}^2 = E\left[(r_p - E(r_p))^2\right] = \mathbf{w}^\intercal \Sigma \mathbf{w} \tag{3.6}$$

A Equação (3.5) fornece a fórmula para o retorno esperado do portfólio, dados os pesos dos ativos individuais. Portanto, o retorno esperado do portfólio é uma média ponderada dos retornos esperados individuais e a variância σ_p^2 do portfólio (3.6) é uma função quadrática do vetor de peso.

Markowitz caracteriza *portfólio eficiente* pelos vetores de peso que resolvem os problemas de otimização:

- Maximizar o retorno para um determinado nível de risco σ_0.

$$\begin{aligned} \max_{w} \quad & \mathbf{w}^\intercal \boldsymbol{\mu} \\ \text{s.a}: \quad & \mathbf{w}^\intercal \Sigma \mathbf{w} = \sigma_0 \\ & \mathbf{w}^\intercal \mathbf{1} = 1 \end{aligned} \tag{3.7}$$

- Minimizar o risco para um determinado nível de retorno μ_0.

$$\begin{aligned} \min_{w} \quad & \frac{1}{2} \mathbf{w}^\intercal \boldsymbol{\Sigma} \mathbf{w} \\ \text{s.a}: \quad & \mathbf{w}^\intercal \boldsymbol{\mu} = \mu_0 \\ & \mathbf{w}^\intercal \mathbf{1} = 1 \end{aligned} \tag{3.8}$$

- Maximizar o retorno esperado / minimizar o risco usando um fator de aversão ao risco.

$$\begin{aligned} \max_{\mathbf{w}} \quad & \mathbf{w}^\intercal \boldsymbol{\mu} - \kappa \mathbf{w}^\intercal \Sigma \mathbf{w} \\ \text{s.a} \quad & \sum_{i=1}^{n} w_i = 1, \end{aligned} \tag{3.9}$$

e onde, $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^n$, e κ é chamado de parâmetro de aversão ao risco.

Portanto o portfólio ótimo para o modelo média-variância de Markowitz é a solução dos problemas deste tipo seguindo o esquema de programação quadrática convexa descrito na seção 3.3.

Iremos resolver o problema de otimização (3.8) usando multiplicadores de Lagrange.

Aplicando o método dos multiplicadores de Lagrange a este problema de minimização convexa sujeito a restrições lineares temos:

- O Lagrangiano definido por

$$L(\mathbf{w}, \lambda_1, \lambda_2) = \frac{1}{2}\mathbf{w}^\intercal \Sigma \mathbf{w} + \lambda_1(\mu_0 - \mathbf{w}^\intercal \boldsymbol{\mu}) + \lambda_2(1 - \mathbf{w}^\intercal \mathbf{1});$$

- As condições de primeira ordem:

$$\frac{\partial L}{\partial w} = \mathbf{0} = \Sigma \mathbf{w} - \lambda_1 \boldsymbol{\mu} - \lambda_2 \mathbf{1}, \tag{3.10}$$

$$\frac{\partial L}{\partial \lambda_1} = \mathbf{0} = \mu_0 - \mathbf{w}^\intercal \boldsymbol{\mu}, \tag{3.11}$$

$$\frac{\partial L}{\partial \lambda_2} = \mathbf{0} = 1 - \mathbf{w}^\intercal \mathbf{1}. \tag{3.12}$$

A partir do Teorema Espetral, resulta que Σ é uma matriz invertível pois tem determinante não nulo.

- Resolve-se $\mathbf{w}$ em ordem a λ_1 e λ_2:

$$\mathbf{w} = \Sigma^{-1}(\lambda_1 \boldsymbol{\mu} + \lambda_2 \mathbf{1}) = \lambda_1 \Sigma^{-1} \boldsymbol{\mu} + \lambda_2 \Sigma^{-1} \mathbf{1}. \tag{3.13}$$

Os multiplicadores de Lagrange podem ser encontrados através das suas restrições e, deste modo, reescrevendo (3.11) e (3.12) usando a fórmula obtida em (3.13), temos:

$$\begin{aligned} \mathbf{w}^\intercal \boldsymbol{\mu} = \mu_0 \Leftrightarrow & \lambda_1 \boldsymbol{\mu}^\intercal \Sigma^{-1} \boldsymbol{\mu} + \lambda_2 \mathbf{1}^\intercal \Sigma^{-1} \boldsymbol{\mu} = \mu_0 \\ \mathbf{w}^\intercal \mathbf{1} = 1 \Leftrightarrow & \lambda_1 \boldsymbol{\mu}^\intercal \Sigma^{-1} \mathbf{1} + \lambda_2 \mathbf{1}^\intercal \Sigma^{-1} \mathbf{1} = 1. \end{aligned} \tag{3.14}$$

Claro que

$$\mathbf{1}^\intercal \Sigma^{-1} \boldsymbol{\mu} = \boldsymbol{\mu}^\intercal \Sigma^{-1} \mathbf{1}.$$

- Obtém-se λ_1, λ_2 por substituição de $\mathbf{w}$:

$$\Rightarrow \begin{bmatrix} \mu_0 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix},$$

com

$$a = (\boldsymbol{\mu}^\intercal \Sigma^{-1} \boldsymbol{\mu}),\ b = (\boldsymbol{\mu}^\intercal \Sigma^{-1} \mathbf{1}) \text{ e } c = (\mathbf{1}^\intercal \Sigma^{-1} \mathbf{1}).$$

Este sistema admite solução se:

$$\begin{vmatrix} a & b \\ b & c \end{vmatrix} = ac - b^2 \neq 0.$$

Isto é verdade quando $\boldsymbol{\mu}$ não é proporcional a $\mathbf{1}$. Assumindo isto, ou seja, que os retornos de ativos não são todos iguais - resolvemos as equações do sistema em λ_1 e λ_2, obtemos:

$$\lambda_1 = \frac{b - c\mu_0}{b^2 - ac},$$

$$\lambda_2 = \frac{\mu_0 b - a}{b^2 - ac}.$$

- Substituindo em (3.13) obtemos a solução desejada

$$\begin{aligned}\mathbf{w} &= \lambda_1\Sigma^{-1}\boldsymbol{\mu} + \lambda_2\Sigma^{-1}\mathbf{1} \\ &= \frac{b - c\mu_0}{b^2 - ac}\Sigma^{-1}\boldsymbol{\mu} + \frac{\mu_0 b - a}{b^2 - ac}\Sigma^{-1}\mathbf{1} \\ &= \frac{b\Sigma^{-1}\boldsymbol{\mu} - c\Sigma^{-1}\boldsymbol{\mu}\mu_0}{b^2 - ac} + \frac{b\Sigma^{-1}\mathbf{1}\mu_0 - a\Sigma^{-1}\mathbf{1}}{b^2 - ac} \\ &= \frac{b\Sigma^{-1}\boldsymbol{\mu} - a\Sigma^{-1}\mathbf{1}}{b^2 - ac} + \frac{b\Sigma^{-1}\mathbf{1} - c\Sigma^{-1}\boldsymbol{\mu}}{b^2 - ac}\mu_0 \\ &= g + h\mu_0,\end{aligned} \tag{3.15}$$

onde g e h são vetores e μ_0 é o escalar que representa o retorno esperado do portfólio. Note que $\mathbf{w}$ é uma função linear de μ_0.

- A variância do portfólio otimizado de média-variância é:

$$\begin{aligned}\sigma_p^2 &= \mathbf{w}^\intercal\Sigma\mathbf{w} \\ &= \mathbf{w}^\intercal\Sigma\Sigma^{-1}(\lambda_1\boldsymbol{\mu} + \lambda_2\mathbf{1}) \\ &= \lambda_1\mathbf{w}^\intercal\boldsymbol{\mu} + \lambda_2\mathbf{w}^\intercal\mathbf{1}.\end{aligned} \tag{3.16}$$

Usando (3.11) e (3.12) a variância do portfólio vem dada por:

$$\sigma_p^2 = \lambda_1\mu_0 + \lambda_2 = \frac{b - c\mu_0}{b^2 - ac}\mu_0 + \frac{\mu_0 b - a}{b^2 - ca} = \frac{1}{ac - b^2}(c\mu_0^2 - 2b\mu_0 + a).$$

Deste modo, obtemos que a variância é uma função quadrática do retorno esperado.

Os três primeiros problemas enunciados (3.7)-(3.9) são matematicamente equivalentes entre si e as suas soluções são chamadas de *médias-variâncias* eficientes. Os pontos eficientes no gráfico risco-retorno são os pontos da *fronteira eficiente*. Assim, nos gráficos que relacionam o desvio padrão e o retorno, a fronteira é uma hipérbole, enquanto que nos gráficos que relacionam a variância e o retorno, esta tem a forma de uma parábola.

Capítulo 4

Implementação Numérica

Neste capítulo, começaremos por rever os objetivos que pretendemos numa gestão do portfólio. Descreveremos sucintamente um problema modelo que ilustra o modelo média-variância. Introduzimos à linguagem de programação que iremos usar na implementação numérica do modelo descrito nos capítulos anteriores, bem como a descrição da base de dados utilizada para testar os algoritmos aqui apresentados. Também introduzimos os métodos matemáticos que vamos usar (curta descrição dos métodos de Monte Carlo e dos algoritmos de Aprendizado de Máquina). Terminamos este capítulo com a indicação das métricas de desempenho a usar neste trabalho.

4.1 Filosofia e Ferramentas ao Dispor

Com o modelo de média-variância de Markowitz pretende-se encontrar os melhores pesos para cada ativo do portfólio a construir por forma a otimizar o desempenho desse portfólio. Isto tem também a consequência de que a implementação deste modelo pode ser usada também para simular a evolução futura dos preços dos ativos do portfólio. Para minimizar o risco, o investidor precisa de informação, como referência para a tomada de decisão de quais ativos deve comprar, vender ou manter. Então, previsões de preços dos ativos no portfólio é de extrema importância.

Assim, o trabalho será desenvolvido em duas fases. A primeira é dedicada à construção e análise de algoritmos que determinam os melhores pesos para um portfólio ótimo. Esta otimização é aqui feita com base em algoritmos de Aprendizado de Máquina e da simulação de Monte Carlo (SMC). Numa segunda fase, propomos a implementação de algoritmos de Aprendizado de Máquina e da simulação de Monte Carlo (assumindo um movimento Browniano geométrico (MBG)) para estimar a evolução dos preços futuros. Terminaremos com uma curta comparação entre os resultados obtidos por estes diferentes métodos.

4.2 Linguagem de Programação e Dados Usados

É extremamente importante garantir que o código do `script` seja executado numa linguagem de programação de alto desempenho, e que sua construção e sintaxe esteja escrita da forma mais eficiente possível. Para este projeto, a linguagem `Python`[1] foi escolhida para processar dados financeiros porque, para além de ser uma linguagem de alto nível, tem velocidade de execução muito alta.

Acrescenta-se ainda que os recursos gráficos do Python facilitam a análise exploratória dos dados, fornecendo assim ao analista uma ferramenta poderosa.

Todos os resultados foram obtidos implementando o código em Python (executado no Google Colab). Os códigos para estas rotinas são apresentados no apêndice [B].

Para base de dados, serão utilizados dados do mercado, em observações diárias. Vamos, através de dados do Yahoo Finanças, construir um portfólio com os preços de fecho das ações de quatro empresas, negociadas nos mercados internacionais.

A amostra escolhida corresponde ao período de 01 de Janeiro de 2015, a 01 de Janeiro de 2020 perfazendo um total de 1,258 dias negociáveis. As empresas selecionadas estão sintetizadas na Tabela 4.1.

Empresa	**Símbolo**	**Setor**
Amazon.com	*AMZN*	Tecnologia com foco em e-commerce
Alphabet Inc.	*GOOG*	Tecnologias
Netflix	*NFLX*	Produção e serviço de streaming por assinatura
Apple	*AAPL*	Tecnologias

Tabela 4.1: 4 Empresas selecionadas aleatoriamente

Porque os mercados financeiros são complexos, a criação de um modelo matemático (aproximação ao sistema real) requer pressupostos e algumas simplificações. Portanto, neste trabalho assumimos os seguintes pressupostos:

- O modelo de otimização é um modelo de período único;
- Assumimos que apenas são permitidas posições longas (ou seja, apenas comprar ações, não vendê-las);
- Assumimos também que temos um mercado perfeito, isto é, um mercado em que não há comissões ou custos de transação;
- O portfólio será formado apenas por ações.

Claro que não existe mercado perfeito no mundo real, mas esta suposição tornará a análise consideravelmente mais simples [54].

[1] É uma linguagem de programação multifuncional de alto nível que é usada em uma ampla gama de domínios e campos técnicos.

Para comparar melhor os retornos, vamos assumir um investimento inicial de 5,000 dólares.

Todos os conceitos teóricos importantes para a compreensão do nosso trabalho foram apresentados nos capítulos anteriores. Vamos agora abordar a primeira fase, da construção e análise de algoritmos para determinação dos melhores pesos.

- **Passo I - Alocação e tratamento de dados**
  Saber distribuir o investimento é importante, porque tem uma influência direta nos retornos. Isto implica que ter algoritmos precisos para esse efeito é decisivo.

  Há vários algoritmos já disponíveis e em open source, que são comumente empregues. Estão organizados em bibliotecas e pré-programados em linguagem Python. São exemplo: `Pandas, yfinance, numpy, matplotlib, mlrose e scipy`.

  Todavia, não podemos introduzir os dados de forma aleatória. São necessários procedimentos de *importação dos dados*, de *normalização dos preços de fecho*, dito pré-processamento de dados.

  Com os dados já pré-processados, tem lugar o cálculo do vetor dos retornos, a determinação da matriz de covariância e cálculo do vetor da volatilidade.

- **Passo II - Otimização de Portfólio**

  Neste segundo passo, introduzimos uma otimização aleatória usando inicialmente uma simulação de Monte Carlo para determinar os portfólios de menor risco e maior retorno (ou seja, máximo Sharpe Ratio), e representar-los graficamente (fronteira eficiente).

Na sequência, recorremos à biblioteca `mlrose`[2] para implementar três algoritmos para determinação de pesos ótimos de portfólio, e consequentemente o máximo Sharpe Ratio. São estes algoritmos o Hill Climb, Simulated Annealing e Algoritmo Genético. Com base nos resultados, faremos as devidas análises, considerado um certo valor para o investimento inicial. Para uma análise detalhada da biblioteca `mlrose` e seus algoritmos implementados consultar, por exemplo, [19].

Na segunda fase, a de previsão da evolução dos preços de mercado, propomos começar por usar simulação de Monte Carlo (assumindo um movimento Browniano geométrico). Outras abordagens são usar o modelo ARIMA, ou o algoritmo Prophet, baseados em Aprendizado de Máquina. Vamos estimar os preços futuros num horizonte de 90 dias. Para comparar os diferentes métodos utilizados, recorremos ao erro médio absoluto e ao erro médio absoluto percentual.

[2]mlrose é uma biblioteca Python para aplicar alguns dos algoritmos de otimização e busca aleatórios mais comuns para uma variedade de diferentes problemas de otimização, tanto em espaços de parâmetros discretos quanto contínuos.

Estas métricas estão já incluídas no módulo `sklearn.metrics`, que é uma biblioteca de Aprendizado de Máquina para Python.

4.3 Fase 1 - Alocação e Otimização de Portfólio

4.3.1 Passo I

Preparemos o ambiente Python com os pacotes necessários:

```
!pip install yfinance
```

```
import pandas as pd
import numpy as np
from pandas_datareader import data
import matplotlib.pyplot as plt
from scipy.stats import norm
import yfinance as yf
```

Construímos, de seguida, o portfólio com as ações das empresas selecionadas: AMZN, GOOG, NFLX, AAPL.

```
Portfólio = ['AMZN', 'GOOG', 'NFLX', 'AAPL']
```

Consideramos um período de 2015-01-01 à 2020-01-01, num total de 1,258 dias negociáveis.

```
preços = pd.DataFrame()
for ação in Portfólio:
  preços[Portfólio] = yf.download(Portfólio, start='2015-01-01',
  end='2020-01-01')['Close']
```

```
preços #  O dólar americano é a moeda utilizada.
```

	AMZN	GOOG	NFLX	AAPL
Date				
2015-01-02	27.332500	308.519989	523.373108	49.848572
2015-01-05	26.562500	302.190002	512.463013	47.311428
2015-01-06	26.565001	295.290009	500.585632	46.501431
2015-01-07	26.937500	298.420013	499.727997	46.742859
2015-01-08	27.972500	300.459991	501.303680	47.779999
...	...	...	...	...

1258 rows × 4 columns

Tabela 4.2: Os primeiros preços de fechamento

Para uma melhor comparação dos preços, efectua-se uma normalização destes, apresentada na Tabela 4.3 seguinte.

	AMZN	GOOG	NFLX	AAPL
Date				
2015-01-02	1.000000	1.000000	1.000000	1.000000
2015-01-05	0.971828	0.979483	0.979154	0.949103
2015-01-06	0.971920	0.957118	0.956460	0.932854
2015-01-07	0.985548	0.967263	0.954822	0.937697
2015-01-08	1.023415	0.973875	0.957832	0.958503
...	...	...	...	...

1258 rows × 4 columns

Tabela 4.3: Preços normalizados

Mostramos a evolução dos preços em diferentes frequências de tempo na Figura 4.1.

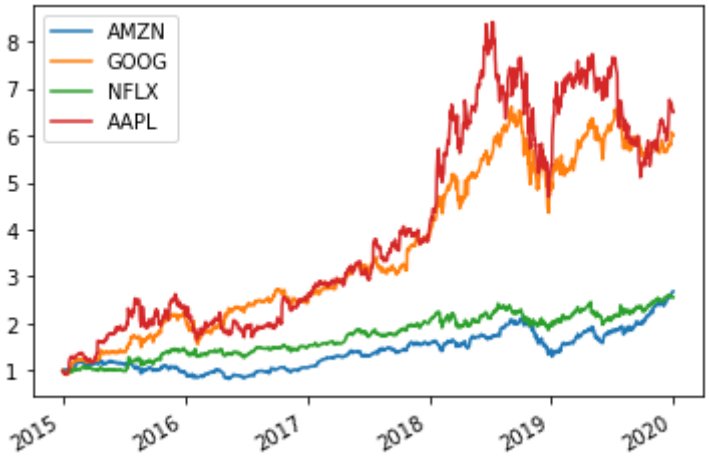


Figura 4.1: Evolução dos preços

Com base nestes dados pré-processados, temos para o nosso portfólio uma taxa de retorno diário representada na Tabela 4.4.

	AMZN	GOOG	NFLX	AAPL
Date				
2015-01-02	NaN	NaN	NaN	NaN
2015-01-05	-0.028172	-0.020517	-0.020846	-0.050897
2015-01-06	0.000094	-0.022833	-0.023177	-0.017121
2015-01-07	0.014022	0.010600	-0.001713	0.005192
...	...	...	...	...

1258 rows × 4 columns

Tabela 4.4: Taxa de retornos diário

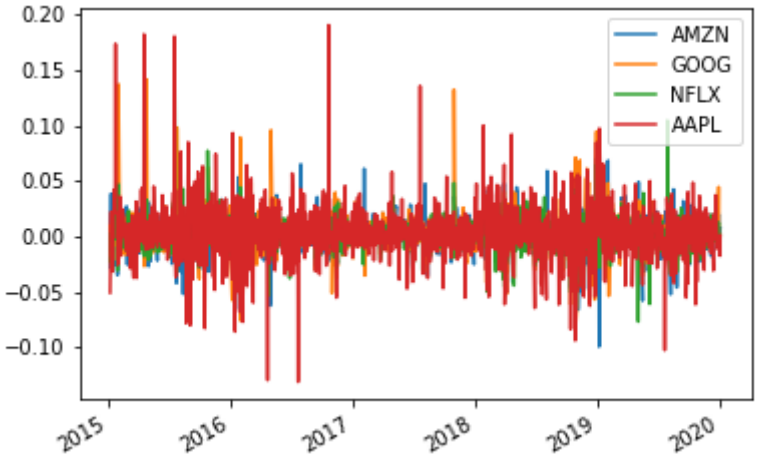


Figura 4.2: Taxa de retornos diário

A forma como os retornos se relacionam é dada pela matriz de covariância.

	AMZN	**GOOG**	**NFLX**	**AAPL**
AMZN	0.000245	0.000142	0.000123	0.000151
GOOG	0.000142	0.000340	0.000179	0.000233
NFLX	0.000123	0.000179	0.000229	0.000185
AAPL	0.000151	0.000233	0.000185	0.000689

Tabela 4.5: Matriz de covariância dos retornos

e correspondente matriz de correlação.

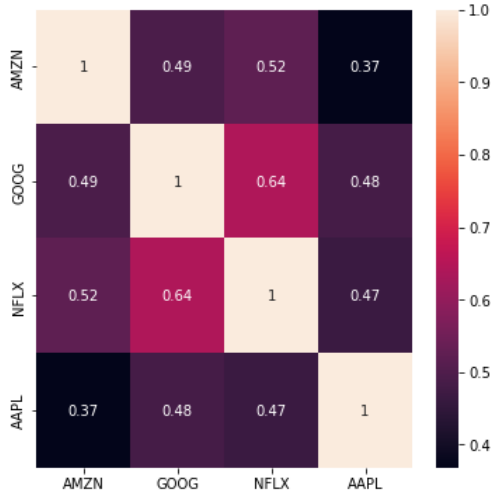


Figura 4.3: Correlação entre os retornos

Sendo o nosso portfólio composto por quatro ativos, de pesos w_1, w_2, w_3, w_4, e retornos

r_1, r_2, r_3, r_4, então o risco do nosso portfólio é dado pela variância dos seus retornos, ou seja:

$$\sigma_p^2 = \sum_{i=1}^{4}\sum_{j=1}^{4} w_i w_j \sigma_{ij},$$

onde σ_{ij} é a covariância entre os retornos dos ativos i e j.

O risco anual do portfólio é então dado por:

$$\sigma_{ap}^2 = \sum_{i=1}^{4}\sum_{j=1}^{4} w_i w_j \sigma_{ij} \times 252,$$

assumindo 252 dias negociáveis por ano.

4.3.2 Passo II

Vamos agora proceder à otimização do portfólio via simulação de Monte Carlo.

Para a simulação de Monte Carlo usaremos, 10,000 simulações e uma taxa livre de risco (r_f) de 0.01. Para estes, vamos encontrar os parâmetros: retorno, volatilidade, índice Sharpe e pesos. Relembre-se que o índice de Sharpe (Sharpe Ratio) mede a qualidade do retorno ajustado ao risco, pela fórmula:

$$SR = \frac{\boldsymbol{\mu}_p - r_f}{\sigma_p} \tag{4.1}$$

onde $\boldsymbol{\mu}_p = \mathbf{w}^\intercal \boldsymbol{\mu}$ é o retorno esperado do portfólio, e $\sigma_p = \sqrt{\sigma_p^2}$ é o seu desvio padrão. Porque pretendemos encontrar o portfólio que maximiza o índice Sharpe, estamos efetivamente à procura do

$$\max\left(\frac{\boldsymbol{\mu}_p - r_f}{\sigma_p}\right)$$

na classe de fronteira eficiente.

```
mc_retorno=[]
mc_volatilidade=[]
mc_pesos=[]
mc_sharpe=[]
num_simul=10000
rf = 0.01
```

	Retorno	Volatilidade	Sharpe Ratio	AMZN-Pesos	GOOG-Pesos	NFLX-Pesos	AAPL-Pesos
0	0.333250	0.239506	1.349652	0.184182	0.281132	0.279196	0.255491
1	0.373478	0.264088	1.376353	0.022088	0.425739	0.233058	0.319115
2	0.335760	0.254370	1.280652	0.357054	0.105198	0.147779	0.389969
...	...	...	...	...	...	...	...

10000 rows × 7 columns

Tabela 4.6: Distribuição aleatória dos pesos

Acessando os dados da Tabela 4.6 podemos encontrar o portfólio com o menor risco, através da linha de comando.

```
volatilidade_minima_por=portfolio_mc.iloc[portfolio_mc['Volatilidade'].idxmin()]
volatilidade_minima_por
```

```
Retorno         0.240367
Volatilidade    0.211913
Sharpe Ratio    1.087086
AMZN-Pesos      0.446447
GOOG-Pesos      0.086153
NFLX-Pesos      0.458136
AAPL-Pesos      0.009263
Name: 4858, dtype: float64
```

Como podemos observar no output acima a volatilidade mínima está no portfólio onde os pesos de AMZN, GOOG, NFLX, AAPL são 44.64%, 8.62%, 45.81%, 0.93% respetivamente.

Da mesma forma, podemos aceder ao portfólio ótimo (máximo retorno com menor risco) acedendo ao máximo Sharpe Ratio.

```
max_sharpe_por=portfolio_mc.iloc[portfólio_mc['Sharpe Ratio'].idxmax()]
max_sharpe_por
```

```
Retorno         0.385621
Volatilidade    0.261464
Sharpe Ratio    1.436608
AMZN-Pesos      0.160074
GOOG-Pesos      0.633669
NFLX-Pesos      0.001728
AAPL-Pesos      0.204529
Name: 9372, dtype: float64
```

e que nos dá o portfólio com os pesos para AMZN, GOOG, NFLX, AAPL de 16.01%, 63.37%, 0.17%, 20.45% respetivamente.

Note-se que, embora a diferença de risco entre o portfólio de volatilidade mínima e o portfólio de risco ideal seja apenas de 4.96%, já a diferença de retornos é de 14.53%.

Podemos obter o conjunto de todos os portfólios ótimos e visualizar o comportamento dos portfólios através da fronteira eficiente e identificar o portfólio de menor risco e maior índice Sharpe (Figura 4.4).

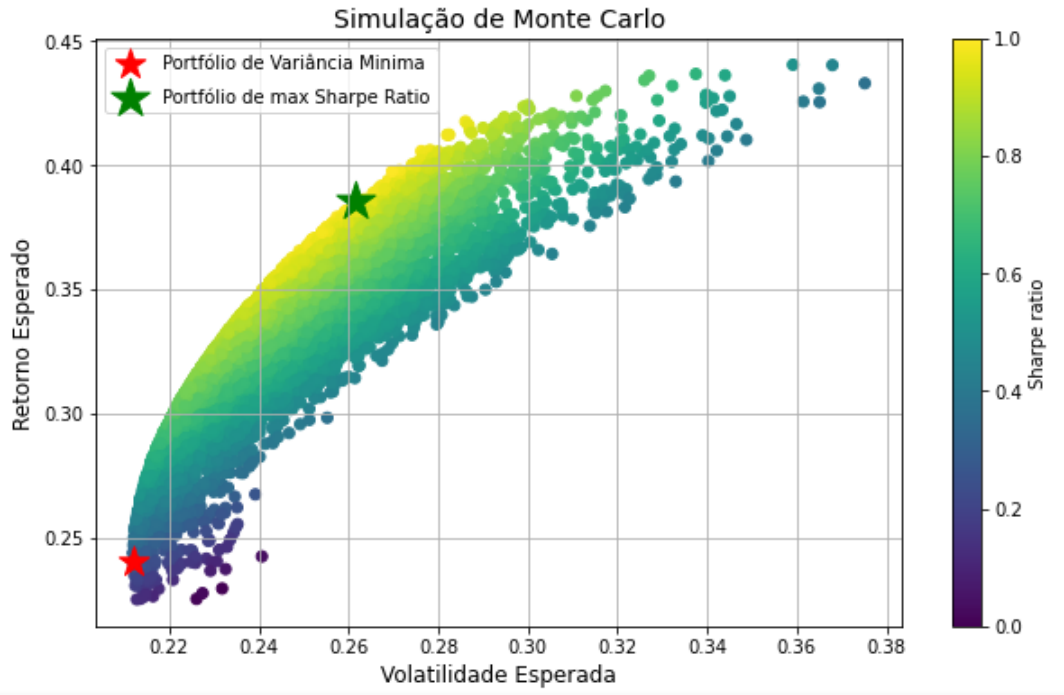


Figura 4.4: Fronteira eficiente

No entanto, este procedimento (otimização do portfólio) pode também ser feito por algoritmos de otimização. Os algoritmos de otimização são de grande importância na forma como nos ajudam a resolver certos tipos de problemas em que os métodos comuns não são suficientes. Estes algoritmos tentam encontrar a melhor solução possível para o problema no período de tempo mais razoável. A solução obtida pode não ser a melhor absoluta (mínimo/máximo global), mas é boa o suficiente, tendo em conta o tempo atribuído.

Encontrar soluções ótimas para muitos problemas não é uma tarefa fácil, porque, dependendo da sua complexidade, pode ser impossível analisar em tempo eficaz todas as soluções possíveis. Em consequência, e porque soluções não globais podem ainda assim ser suficientes, o uso destes algoritmos de busca está a tornar-se cada vez mais popular. De entre estes, destacamos as meta-heurísticas, especialmente aquelas inspiradas na natureza, e que têm ganho uma atenção considerável da comunidade científica, sendo usadas em vários domínios de problemas. Neste trabalho propomos a implementação dos algoritmos inseridos na biblioteca `mlrose` do Python, nomeadamente o Hill Climb, Simulated Annealing e o Algoritmo Genético.

O desempenho de tais algoritmos pode ser avaliado assintoticamente, quer através de resultados de convergência, quer por comparação com outros algoritmos.

A seguir, damos uma noção básica destes algoritmos, discussões e detalhes podem ser encontrados em [3, 4, 12, 13, 24, 27, 50, 52, 59, 60].

Hill Climb (HC)

O HC é um algoritmo clássico para otimização, sendo bastante eficiente na tarefa de encontrar máximos ou mínimos locais. O HC é um método de busca heurístico, utilizado para problemas de otimização matemática na área de Inteligência Artificial. Dado um grande conjunto de entradas e uma boa função heurística, tenta encontrar uma solução suficientemente boa para o problema. Esta solução pode não ser o global ótimo. O algoritmo guarda apenas o estado atual e a sua avaliação, e apenas faz modificações que melhoram o estado atual [24, 27, 50, 52].

Simulated Annealing (SA)

O SA é um algoritmo probabilístico meta-heurístico que pode ser usado em um grande espaço de busca para alcançar um ótimo, com uma certa probabilidade de ser ótimo global.

O SA é inspirado no processo térmico para o aprimoramento de aços, onde o material é aquecido a altas temperaturas, fazendo com que os átomos se movam livremente e depois é arrefecido gradualmente para que as moléculas se encaixem numa melhor posição. O SA tem demonstrado eficiência na resolução de muitos problemas práticos. O SA trabalha do seguinte modo:

(i) Começar com uma solução inicial gerada aleatoriamente para um dado problema;

(ii) Selecionar uma sucessora aleatória por processos estocásticos (que será uma nova solução, e dentro de certos limites dependentes do problema);

(iii) Avaliar a nova solução: se a nova solução for melhor que a solução inicial, então aceitar esta como a nova solução. Caso contrário, a nova solução é aceite, mas com uma probabilidade decrescente (que diminui com o número de iterações efetuadas);

(iv) Terminar a busca assim que a condição de paragem for satisfeita; caso contrário, repetir o passo (ii).

Em conclusão, o SA pode fazer modificações que pioram temporariamente a solução, para que acabam por ser a melhor solução num tempo futuro. A principal vantagem do SA sobre os outros métodos de busca local é a sua flexibilidade e capacidade de se aproximar do ótimo global.

No entanto, a maioria das aplicações da meta-heurística SA é feita para problemas de otimização combinatória, e a sua aplicabilidade aos problemas de seleção de portfólio ainda é fruto de estudo [4, 12, 13, 60].

Algoritmo Genético (AG)

O AG é uma ferramenta muito popular para resolver problemas de otimização em aplicações do mundo real [52]. Os algoritmos genéticos são geralmente utilizados para gerar soluções de alta qualidade para problemas de busca e otimização [3]. É um método heurístico de busca e otimização baseado no processo de seleção natural e busca convergir, de uma população inicial

aleatória, para uma solução de boa qualidade para o problema. Cada indivíduo da população é também chamado de cromossoma e representa uma solução no espaço de busca do problema. O algoritmo avalia a qualidade dos cromossomas utilizando a função `fitness` - no nosso caso, o índice Sharpe - que mede como cada um se adapta (ou não) ao problema. Em seguida, aos indivíduos com maior aptidão é associada uma maior probabilidade de sobreviverem e se reproduzir. Esta seleção pode ser feita aleatoriamente. A geração de novos indivíduos ocorre a partir da aplicação de operadores genéticos cruzamento (`crossover`) e mutação, a qual modifica a estrutura dos genes dos cromossomas, produzindo uma população nova e diferente. Este processo repete-se até que um critério de paragem seja alcançado [3, 44, 52].

Implementação dos algoritmos de otimização

Após prepararmos os ambientes necessários,

```
import mlrose
```

```
preços # Base de Dados
dinheiro_total = 5000 # Investimento Inicial
rf=0.01 # A taxa livre de risco de 1%
```

Obtemos os seguintes resultados.

- Para o algoritmo Hill Climb

```
Ações  Pesos (%)
AMZN  16.199552
GOOG  61.232911
NFLX  0.000000
AAPL  22.567537
--------------------------------
Sharpe Ratio  1.4378606499964712
--------------------------------
Soma_valor  27837.207333661852
```

- Para o algoritmo Simulated Annealing

```
Ações  Pesos (%)
AMZN  23.076923
GOOG  53.846154
NFLX  0.000000
AAPL  23.076923
--------------------------------
Sharpe Ratio  1.43369057146118922
--------------------------------
Soma_valor  26714.02768741193
```

- Para o Algoritmo Genético

```
Ações  Pesos (%)
AMZN  19.462915
GOOG  52.604309
NFLX  6.396743
AAPL  21.536033
--------------------------------
Sharpe Ratio  1.4266447532046367
--------------------------------
Soma_valor  26173.752574423324
```

Soma_valor é o valor gerado pelo portfólio na data do vencimento (ou seja, na última data do portfólio), considerando um investimento inicial de 5,000 dólares.

4.4 Fase 2 - Previsão do Mercado de Ações

A análise e previsão das séries temporárias é uma área de investigação muito ativa nos últimos anos. A precisão da previsão é fundamental para as decisões e, portanto, a pesquisa para melhorar a eficácia dos modelos de previsão mantém-se como objeto de grande interesse [62].

Todavia, prever o mercado de ações é uma tarefa muito complexa e difícil, devido à natureza da alta volatilidade dos preços de mercado. Qualquer erro de decisão resultará em prejuízo para o investidor. Assim, é fundamental ter informação de referência para podermos minimizar o alto risco que o investidor assume quando toma uma decisão sobre quais ações para comprar, vender ou manter.

Neste contexto, diferentes modelos têm sido utilizados. Entre eles, a simulação de Monte Carlo (assumindo um movimento Browniano geométrico (MBG)) [2, 14, 49, 55], o modelo ARIMA [5, 15, 62] e, mais recentemente, o Prophet [26, 34, 59, 63], são modelos muito usados e que vamos utilizar aqui.

No nosso caso, vamos usar a nossa base de dados de histórico de preços do período de 2015-01-01 a 2020-01-01 para prever os próximos 90 dias, de por exemplo, a ação Goog. Os valores previstos (simulados) são comparados com os preços reais (conhecidos) e terminaremos com o estudo da precisão desta previsão.

Acompanharemos esta simulação de previsão com os gráficos que exibem as curvas dos valores reais e previstos.

4.4.1 Simulação de Monte Carlo

De acordo com Healy [20] o modelo de simulação de Monte Carlo baseia-se na hipótese de que os preços das ações seguem um movimento Browniano. O princípio fundamental do método de Monte Carlo consiste em executar um número elevado de simulações que representam o processo para o qual se deseja calcular o valor esperado, de tal maneira que, pela lei dos grandes números, o valor médio dos resultados das simulações convergirá para o valor correto da expressão. Naturalmente, quanto maior o número de simulações, maior será a precisão atingida. Uma simulação de Monte Carlo efetua um elevado número de simulações com diferentes números aleatórios gerados a partir de uma distribuição subjacente para as variáveis. As inúmeras trajetórias possíveis dos preços das ações, dadas por meio de simulação de Monte Carlo, permitem construir a distribuição do preço da ação, e o seu valor à data de vencimento é estimada com base nesta distribuição.

Consideramos um ativo S que segue um movimento Browniano geométrico. Se $S \sim MBG(\mu,\sigma^2)$ e se S tiver o valor inicial $S(0)$, então

$$S(t) = S(0)exp\left(\left[\mu - \frac{1}{2}\sigma^2\right]t + \sigma W(t)\right). \tag{4.2}$$

onde $W(t)$ é um movimento Browniano. Mais geralmente, se $u < t$ então

$$S(t) = S(u)exp\left(\left[\mu - \frac{1}{2}\sigma^2\right](t-u) + \sigma(W(t) - W(u))\right), \tag{4.3}$$

e onde os incrementos de W são independentes e normalmente distribuídos. Isso fornece um procedimento recursivo simples para simular valores de S em diferentes passos temporais $0 = t_0 < t_1 < \ldots < t_n$:

$$S(t_{i+1}) = S(t_i)exp\left(\left[\mu - \frac{1}{2}\sigma^2\right](t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i}Z_{i+1}\right), \tag{4.4}$$

onde as amostras aleatórias $Z_1, Z_2, \ldots, Z_n$ seguem uma distribuição normal independente para cada $i = 0, 1, \ldots, n-1$. Para detalhes sobre o movimento Browniano, consultar o apêndice [A] ou as referências [1, 2, 14, 18].

4.4.2 Previsão de Preços Usando Monte Carlo

Na previsão do preço das ações (assumindo um movimento Browniano geométrico), o algoritmo começa com o cálculo do valor de retorno, seguido de uma estimativa da volatilidade e do valor de *drift*, obtendo então uma previsão do preço das ações (ver, [2]).

O movimento Browniano tem duas componentes principais:

- Drift - que é a direção que as taxas de retorno tomaram no passado.
- Volatilidade a volatilidade histórica multiplicada por uma variável normal padrão aleatória.

Para a previsão de preços da ação GOOG, utilizando a simulação de Monte Carlo baseada no movimento Browniano geométrico, vamos prever preços desta ação num horizonte de 90 dias, a partir da nossa base de dados (de 2015-01-01 até 2020-01-01). Os procedimentos encontram-se resumidos na Tabela 4.7.

Início
↓
Dados Históricos de 2015-01-01 até 2020-01-01
↓
Cálculo do retorno histórico Cálculo da volatilidade histórica Cálculo do drift Cálculo dos retornos diários previstos
↓
Previsão dos preços de 2020-01-02 até 2020-05-12
↓
Fim

Tabela 4.7: Procedimentos de Monte Carlo

Para projetar uma possível trajetória de preços, utilizamos os dados históricos para gerar uma série de retornos diários periódicos usando o logaritmo natural:

$$r_{log} = \log(1 + r)$$

```
retorno_log = np.log(1 + data.pct_change())
retorno_log
```

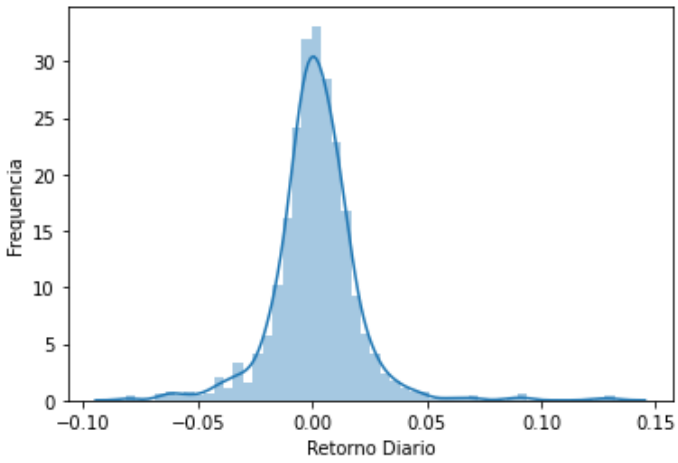


Figura 4.5: Retornos diários da ação GOOG

Começamos então por calcular estes diferentes parâmetros.

```
mu = retorno_log.mean()
var = retorno_log.var()
drift = mu - (0.5*var)
```

Previsão dos retornos diários

```
std = retorno_log.std()
dias_f = 91 # Na primeira data passamos o último preço (real).
n_sim = 1000
Z = norm.ppf(np.random.rand(dias_f, n_sim))
retorno_diario = np.exp(drift.values+std.values*Z)
```

Previsão dos preços futuros

```
for t in range(1, dias_f):
  preços_fut[t] = preços_fut[t-1]*retorno_diario[t]
```

A Figura 4.6 apresenta os resultados obtidos na simulação de 90 dias futuros usando 1,000 caminhos de Monte Carlo.

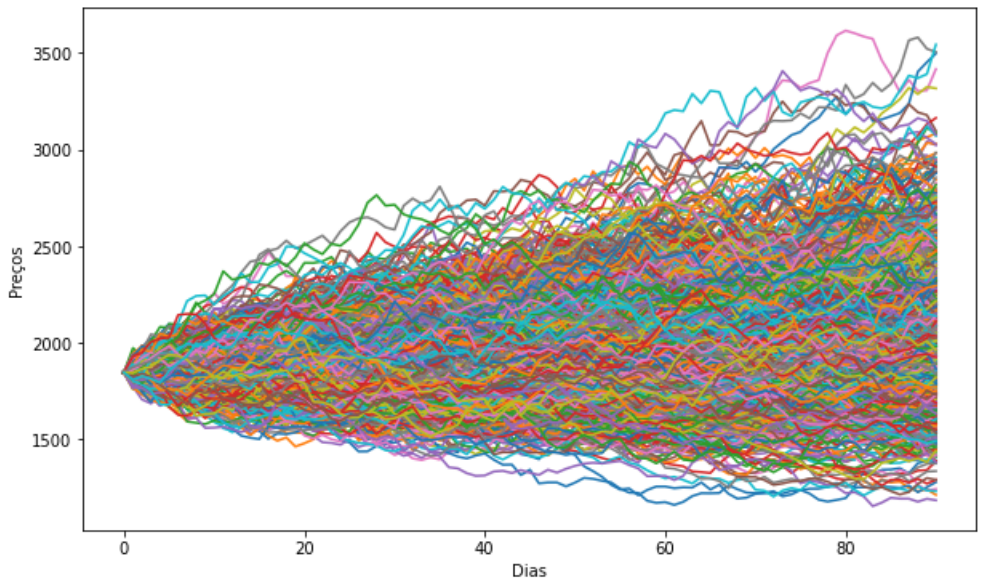


Figura 4.6: Simulação de Monte Carlo (1,000 simulações)

Na Tabela 4.8 escolhemos mostrar apenas cinco primeiros e os últimos preços da previsão versus preços reais para a ação GOOG.

	Date	Previsão MBG & MC	Preços Reais
0	2020-01-02	1859.710929	1898.010010
1	2020-01-03	1847.731820	1874.969971
2	2020-01-06	1868.057923	1902.880005
3	2020-01-07	1899.587018	1906.859985
4	2020-01-08	1917.320397	1891.969971
...	...	...	...
85	2020-05-05	2266.416853	2317.800049
86	2020-05-06	2258.047848	2351.260010
87	2020-05-07	2284.775258	2367.610107
88	2020-05-08	2286.207293	2379.610107
89	2020-05-11	2269.441065	2409.000000

90 rows × 2 columns

Tabela 4.8: Os primeiros 5 e os últimos 5 preços obtidos com a SMC

A Figura 4.7 apresenta a melhor simulação obtida com Monte Carlo.

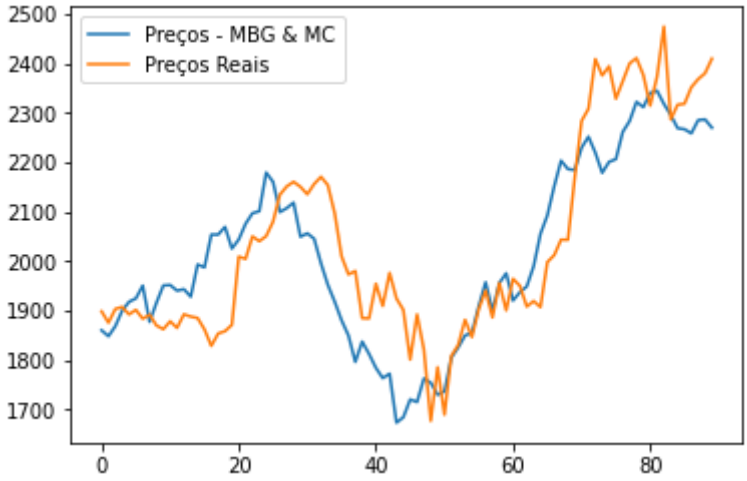


Figura 4.7: Melhor previsão obtida com a SMC

4.4.3 Previsão de Preços Usando Algoritmos de Machine Learning

O sucesso da construção do portfólio depende do seu desempenho futuro nos mercados. O Aprendizado de Máquina, que deixa (sob certo controlo) a *máquina* analisar um elevado número de desempenhos de ativos em curto tempo, ajuda a construir um portfólio com base em extrapolações dos desempenhos passados. Desenvolvimentos recentes em aprendizado de máquina trouxeram oportunidades significativas para incorporar teoria de previsão na seleção de portfólio [11], que permite, com indicações de dados históricos, estimar com uma certa margem

de erro os retornos futuros.

Muitos modelos têm sido explorados na literatura para previsão de séries temporais e, de acordo com Lewinson [34] uma abordagem amplamente utilizada para modelagem é o modelo autoregressivo integrado da média móvel (modelos de classe ARIMA), ou também uma nova abordagem utilizando o modelo aditivo da biblioteca do Facebook Prophet.

Para avaliar o desempenho da previsão dos diferentes modelos, dividimos a nossa base de dados em duas amostras: amostra de treino e amostra de teste, fornecidas na Tabela 4.9. O conjunto de dados de treino é utilizado exclusivamente para o desenvolvimento do modelo e, depois, a amostra de teste é usada para avaliar o sucesso do modelo estabelecido.

Ações	Dados da Amostra	Dados de Treino	Dados de Teste
GOOG	De: 2015-01-01 Até: 2020-05-12 (1,348 dias$^{\Psi}$)	De: 2015-01-01 Até: 2020-01-01 (1,258 dias$^{\Psi}$)	De: 2020-01-02 Até: 2020-05-12 (90 dias$^{\Psi}$)

$^{\Psi}$ *Dias de negociações*

Tabela 4.9: Composição da amostra

4.4.4 Previsão de Preços com ARIMA

O modelo autoregressivo integrado da média móvel, também conhecido como o modelo ARIMA, é um dos algoritmos de Machine Learning mais fáceis e eficazes para realizar previsões. Tem a sua origem na combinação de Regressão Automática (modelo autoregressivo AR) e no modelo da Média Móvel (MA) [15]. A parte de auto-regressão (AR) é um modelo de série temporal que usa observações de etapas de tempo anteriores como entradas para a equação de regressão por forma a prever o valor na próxima etapa de tempo, isto é, realiza regressão no passo de tempo anterior $(t-1)$ para prever no passo de tempo t. Já o modelo da Média Móvel (MA) calcula a média simples em determinados períodos de tempo e divide pelo número total dos períodos tomados. Detalhes sobre o método ARIMA podem ser encontrados em [5, 15, 62]. Neste trabalho, iremos usar o Auto-ARIMA.

Mostramos na Tabela 4.10 os cinco primeiros e os últimos preços da previsão versus preços reais para a ação GOOG.

	Previsão Arima	Preços Reais
Date		
2020-01-02	1849.064564	1898.010010
2020-01-03	1850.289162	1874.969971
2020-01-06	1851.513761	1902.880005
2020-01-07	1852.738359	1906.859985
2020-01-08	1853.962957	1891.969971
...	...	...
2020-05-05	1953.155414	2317.800049
2020-05-06	1954.380012	2351.260010
2020-05-07	1955.604610	2367.610107
2020-05-08	1956.829208	2379.610107
2020-05-11	1958.053807	2409.000000

90 rows × 2 columns

Tabela 4.10: Os primeiros 5 e últimos 5 preços obtidos com ARIMA

A Figura 4.8 mostra os resultados obtidos pelo modelo ARIMA.

Figura 4.8: Previsões de 90 dias com ARIMA

4.4.5 Previsão de Preços com Fbprophet

O Fbprophet, ou simplesmente Prophet, é um procedimento desenvolvido para a previsão de dados de séries temporais com base num modelo aditivo onde tendências não lineares são ajustadas com sazonalidade anual, semanal e diária, bem como os efeitos dos feriados. Funciona melhor com séries temporais que têm fortes efeitos sazonais e múltiplas temporadas registadas no histórico de dados. Prophet é robusto na ausência de dados e a mudanças na tendência. Normalmente, lida bem com valores discrepantes [59, 63].

O Prophet é uma ferramenta recente, baseado na técnica de ajuste de curva no modelo Bayesi-

ano, e tem mostrado uma melhoria de desempenho em termos de precisão de previsão de séries temporais [26]. Não requer muitos dados para fazer previsão. Esta técnica é mais adequada quando os dados da série temporal têm fortes atributos sazonais como fatores influenciadores.

Os dados de treino são fornecidos à entrada para o modelo usando `fit()`. Quando o modelo está pronto (isto é, treinado), a previsão é feita usando o comando `predict()` [26].

Os primeiros cinco e os últimos cinco preços da previsão e preços reais da ação GOOG estão representados na tabela 4.11.

	Preços Prophet	Preços Reais
Date		
2020-01-02	1776.729863	1898.010010
2020-01-03	1779.399434	1874.969971
2020-01-06	1779.583359	1902.880005
2020-01-07	1754.999241	1906.859985
2020-01-08	1758.396760	1891.969971
...	...	...
2020-05-05	1893.591038	2317.800049
2020-05-06	1890.768488	2351.260010
2020-05-07	1862.833732	2367.610107
2020-05-08	1862.597923	2379.610107
2020-05-11	1889.091499	2409.000000

90 rows × 2 columns

Tabela 4.11: Os primeiros 5 e os últimos 5 preços obtidos com Prophet

A Figura 4.9 ilustra os resultados das previsões obtidas pelo modelo prophet.

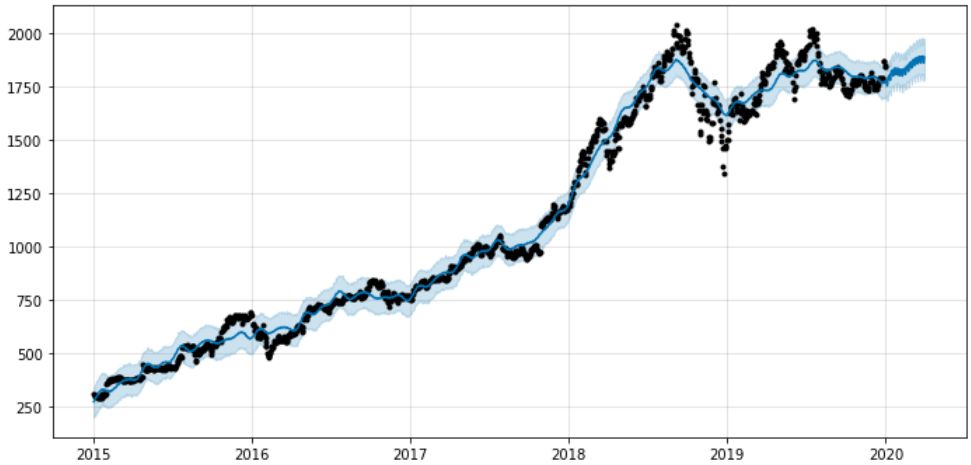


Figura 4.9: Previsão obtida com Prophet

4.4.6 Métricas de Desempenho

Para avaliar o nível de precisão e qualidade das previsões obtidas, há necessidade de introduzir métricas de desempenho. Duas métricas de desempenho são essencialmente usadas:

- erro médio absoluto, MAE (da sigla em inglês, Mean Absolute Error) - para ver o quanto em média obtemos de erro absoluto entre a previsão e o valor real;
- erro médio absoluto percentual, MAPE (da sigla em inglês, Mean Absolute Percentage Error) para quantificar a precisão geral.

O erro médio absoluto (MAE) é dado como:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|,$$

e é usado para medir a precisão do erro entre os dados previstos e os reais. Já o erro médio absoluto percentual (MAPE), calculado como:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \times 100,$$

é usado para quantificar a precisão geral da estrutura de previsão e calcular o nível de confiança. Aqui, Y_i são os valores reais dos dados, $\hat{Y}_i$ são os valores da previsão e n indica o número de observações.

Estas métricas são as mais usadas devido à sua simplicidade e interpretação muito intuitiva. As métricas de desempenho indicadas medem a precisão dos modelos, e desempenham um papel crucial na aceitação da estrutura do portfólio com base na qualidade da previsão de preços. Outras métricas possíveis de usar são as métricas de erro médio quadrático (MSE) e raiz do erro médio quadrático (RMSE).

Para interpretar o valor do MAPE, vamos utilizar os valores na Tabela 4.12 e avaliar a precisão das previsões.

MAPE-Value	Accurace Forecast
Less than 10%	Highly accurate forecasting
10% to 20%	Good forecasting
20% to 50%	Reasonable forecasting
More than 50%	Inaccurate forecasting

Tabela 4.12: Fonte: Lewis [35] - Valor do MAPE - Interpretação

Na Tabela 4.13 apresentamos a avaliação das previsões usando MAE e MAPE.

	Monte Carlo	arima	Prophet
MAE	88.95	161.08	206.63
MAPE	4.37	7.34	9.50

Tabela 4.13: Avaliação das previsões

Capítulo 5

Resultados e Análises

Neste capítulo vamos analisar os resultados obtidos no capítulo anterior, relativamente à construção do portfólio exemplo:

- investimento inicial no valor de 5,000 dólares;
- portfólio com as ações das empresas selecionadas: AMZN, GOOG, NFLX, AAPL;
- base de dados de 2015-01-01 à 2020-01-01, num total de 1,258 dias negociáveis.
- período temporal de previsão a 90 dias.

e resultantes dos algoritmos anteriormente discutidos.

5.1 Otimização de Portfólio

Nesta seção vamos apresentar e analisar os resultados decorrentes da primeira fase - Otimização de Portfólios.

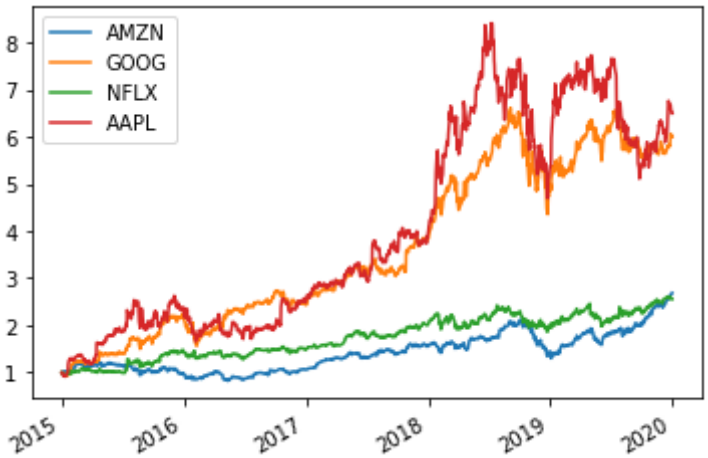


Figura 5.1: Evolução dos preços

É importante notar que a Figura 5.1 nos dá já uma projeção futura do desenvolvimento dos preços, e podemos observar que empresas GOOG e AAPL são as que prometem retornos mais elevados. A confirmação deste facto será feita posteriormente, à base das medidas de desempenho e após o processo de otimização ter sido feito.

Usando o nosso exemplo, com o investimento inicial de 5,000 dólares, teremos o valor na maturidade (isto é, ao fim de 1,258 dias de negociações) indicado por Soma-valor.

Podemos verificar que a simulação de Monte Carlo é sempre uma alternativa viável para otimização do portfólio, e que o desempenho do portfólio aumenta com o número de simulações. Considerando, por exemplo, 1,000 simulações de Monte Carlo, podemos obter um portfólio ótimo em que o investimento inicial será distribuído da seguinte forma: 20.00% em AMZN, 54.29% em GOOG, 1.28% em NFLX e 24.42% em AAPL e, na maturidade teríamos um Soma-valor de 27,034.99 dólares. Isto é visível nos dados da Tabela 5.1.

	Otimização de Portfólio com Monte Carlo Número de Simulações			
	10	100	1,000	10,000
AMZN_Pesos	0.085678	0.030636	0.200063	0.160074
GOOG_Pesos	0.551602	0.720910	0.542885	0.633669
NFLX_Pesos	0.175638	0.080306	0.012812	0.001728
AAPL_Pesos	0.187082	0.168147	0.244241	0.204529
Volatilidade	0.252616	0.269562	0.257381	0.261464
Retorno	0.365271	0.391233	0.379052	0.385621
Sharpe Ratio	1.406369	1.414269	1.433872	1.436608
Soma-valor	25,984.60	26,173.75	27,034.99	27,786.22

Tabela 5.1: Otimização de portfólio com Monte Carlo

Esta simulação garante o maior retorno de investimento nas duas empresas acima mencionadas. É evidente que ter-se considerado uma taxa livre de risco de 0.01 tem grande influência nestes resultados.

Avaliamos e indicamos na Figura 5.2 as combinações obtidas por otimização via Monte Carlo (considerando 10,000 simulações). Encontramos assim o portfólio de variância mínima, que é o portfólio que apresenta o menor risco (21.19%) para um nível de retorno de 24.04%. Analisando o índice Sharpe, identificamos também o portfólio que garante maior rentabilidade (38.56%) para um nível de risco superior ao do portfólio de variância mínima (26.15%)(também indicado na Figura 5.2). Embora a diferença de risco entre o portfólio de volatilidade mínima e o portfólio maior rentabilidade seja apenas de 4.96%, já a diferença de retornos é de 14.53%. Note-se que a fronteira eficiente é formada pelos portfólios que possuem o menor nível de risco / maior retorno.

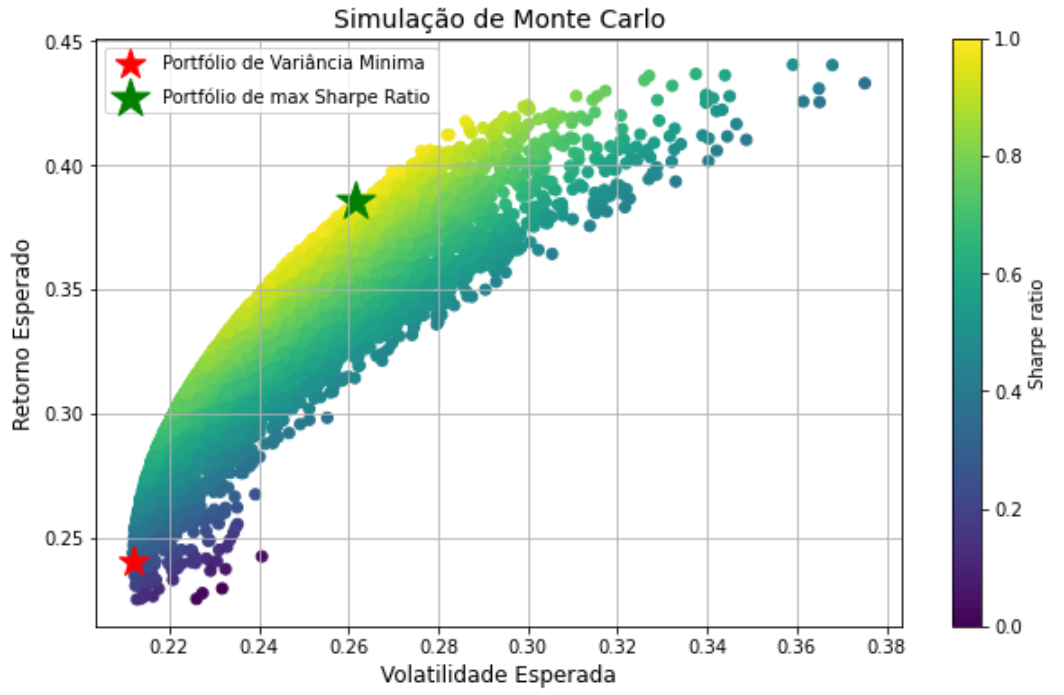


Figura 5.2: Fronteira eficiente

Quando usamos os algoritmos de Aprendizado de Máquina para otimizar o portfólio, com vista a obter a melhor distribuição dos pesos com o maior índice Sharpe, obtemos os resultados listados na Tabela 5.2.

	Otimização de Portfólio com Algoritmos		
	Hill Climb	Simulated Annealing	Algoritmo Genético
AMZN_Pesos	0.161996	0.230769	0.194629
GOOG_Pesos	0.612329	0.538462	0.526043
NFLX_Pesos	0.0	0.0	0.063967
AAPL_Pesos	0.225675	0.230769	0.215360
Volatilidade	0.262104	0.254816	0.251512
Retorno	0.386869	0.375328	0.368818
Sharpe Ratio	1.437861	1.433691	1.426645
Soma-valor	27,837.21	26,714.03	26,173.75

Tabela 5.2: Otimização de portfólios com algoritmos

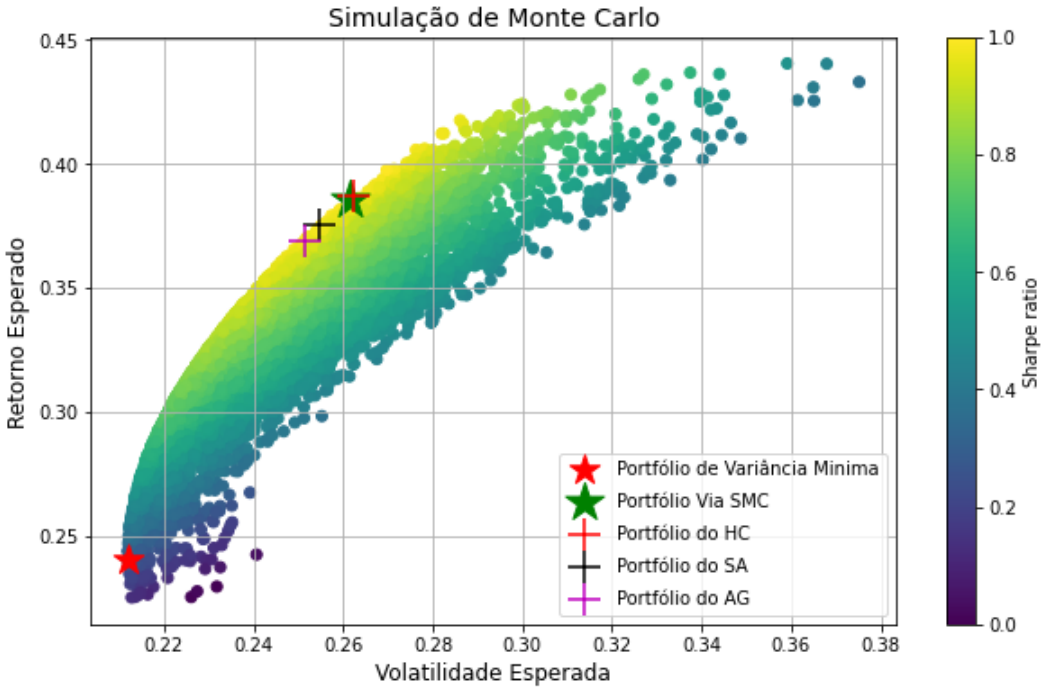


Figura 5.3: Fronteira eficiente - Desempenho dos algoritmos

A simulação de Monte Carlo, comparativamente aos resultados por Aprendizado de Máquina (AM), parece apresentar iguais ou melhores resultados (dada a flexibilidade de aumentarmos o número de simulações). Com efeito, e apoiados na Figura 5.3 notamos que os algoritmos (AM) utilizados provaram ser técnicas poderosas, mostrando um bom desempenho, visível no facto de que os portfólios selecionados por cada um dos algoritmos se situam todos na fronteira eficiente.

Analisando agora os valores da Soma-valor, temos um potencial valor de 27,837.21 (dólares) para o HC, enquanto que para SA e AG, os valores são próximos uns dos outro. É importante notar que tanto a simulação de Monte Carlo como os algoritmos AM apontam para as mesmas empresas como merecedores de maior investimento.

5.2 Previsão de Preços

Nesta seção vamos apresentar, analisar e comparar os resultados decorrentes das simulações e previsões dos nossos modelos, relativamente à ação GOOG do portfólio.

Começamos por analisar os primeiros e últimos valores previstos pelos modelos para o período considerado (Tabela 5.3). A primeira coluna refere-se aos valores reais dos preços da ação, as restantes colunas indicam as previsões mediante os algoritmos AM indicados e ao método de Monte Carlo MC (lembramos, mais uma vez, que o método MC é feito assumindo um movimento Browniano geométrico).

	Date	Preços Reais	Previsão MGB & MC	Previsão Arima	Previsão Prophet
0	2020-01-02	1898.010010	1859.710929	1849.064564	1776.729863
1	2020-01-03	1874.969971	1847.731820	1850.289162	1779.399434
2	2020-01-06	1902.880005	1868.057923	1851.513761	1779.583359
3	2020-01-07	1906.859985	1899.587018	1852.738359	1754.999241
4	2020-01-08	1891.969971	1917.320397	1853.962957	1758.396760
...	...	...	...	...	...
85	2020-05-05	2317.800049	2266.416853	1953.155414	1893.591038
86	2020-05-06	2351.260010	2258.047848	1954.380012	1890.768488
87	2020-05-07	2367.610107	2284.775258	1955.604610	1862.833732
88	2020-05-08	2379.610107	2286.207293	1956.829208	1862.597923
89	2020-05-11	2409.000000	2269.441065	1958.053807	1889.091499

90 rows × 4 columns

Tabela 5.3: Preços reais e previstos

De acordo com os valores obtidos na Tabela 5.3, constata-se que os valores reais e a estimativa de Monte Carlo são muito próximos uns dos outros, o que é visível na Figura 5.4.

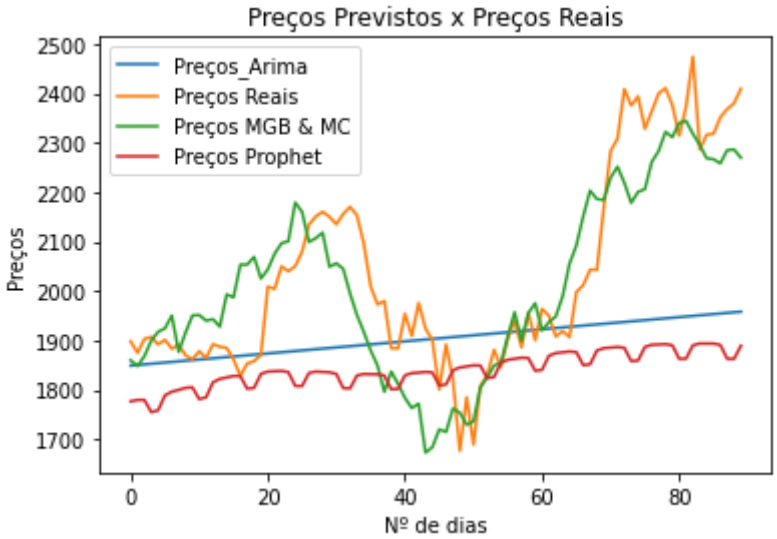


Figura 5.4: Previsão dos modelos

Também é visível que a previsão do comportamento a 90 dias da ação, dada pelos modelos ARIMA e Prophet, não é uma previsão credível. Tal pode ser devido, ou dados de treino, ou o horizonte temporal considerado. As medidas de desempenho ajudam aqui a classificar o bom (ou não tão bom) desempenho do algoritmo. A Tabela 5.4 indica os valores para as diferentes métricas de desempenho.

	MAE	MAPE
Monte Carlo	88.95	4.37
Arima	161.08	7.34
Prophet	206.63	9.50

Tabela 5.4: Avaliação das previsões

Olhando para os dados na Tabela 5.4, observa-se que a simulação de Monte Carlo (assumindo o movimento Browniano geométrico), supera os algoritmos AM. Por outro lado, é importante notar que o Prophet apresenta um desempenho pior, quando medido segundo MAE, porque este algoritmo é criado para obter melhor resultado, em face de dados em falta. Na nossa situação (não há dados em falta) tal afeta negativamente o seu desempenho. Ainda assim, e de acordo com a interpretação dos valores do MAPE (Tabela 4.12), as previsões feitas pelos algoritmos AM são consideradas altamente precisas, apresentando um valor inferior a 10%.

Note-se que o período selecionado para fazer as previsões coincide com o período de Pandemia, uma altura em que os mercados estavam em alta volatilidade. Assim, é expetável que o MC (baseado em movimento Browniano geométrico) apresente melhores resultados do que os métodos AM. Ainda assim, o facto de estas previsões serem ainda altamente precisas reforça a ideia de podermos utilizar estes algoritmos de uma forma considerada segura para o investidor.

Capítulo 6

Considerações Finais

Neste capítulo apresentamos as nossas considerações finais, trazendo algumas contribuições, indicando as limitações dos métodos usados, e terminando com uma perspetiva para trabalho futuro.

6.1 Considerações Finais

A teoria do portfólio foi desenvolvida pela primeira vez por Harry Markowitz na década de cinquenta. O seu trabalho, que foi expandido por vários investigadores, fornece as bases para a chamada teoria moderna do portfólio. Este introduziu o conceito de diversificação de portfólio, e mostrou a importância de se investir num portfólio diversificado. O seu modelo pioneiro visou encontrar o trade-off ótimo entre o risco e o retorno dos diferentes investimentos, procurando encontrar uma combinação de ativos que atendesse, da melhor forma, às necessidades e exigências do investidor.

O modelo proposto é operacionalizado por técnicas de programação quadrática, cujo objetivo visa uma otimização de portfólio com base na média, variância e covariância dos retornos esperados dos ativos. Estes parâmetros são estimados com base no histórico de informações (que se presume conhecido, mesmo que de forma incompleta), e tendo em conta o vetor de médias e a matriz de covariância desses retornos.

Neste trabalho propomo-nos analisar a teoria original do portfólio desenvolvida por Markowitz. Propomos o uso de simulações de Monte Carlo, bem como de algoritmos de Machine Learning, que permitam, por um lado, determinar e visualizar o portfólio ótimo através da determinação dos melhores pesos e, por outro lado, simular os preços futuros dos ativos do portfólio.

Resumidamente, uma primeira fase lidámos com a otimização do portfólio, ajustando pesos de melhor forma, e de modo a obter portfólios ótimos, usamos o índice Sharpe como medida de desempenho. Exploramos a simulação de Monte Carlo e os algoritmos de otimização (a sa-

ber, HC, SA e AG) para obter o portfólio ótimo. De entre os algoritmos utilizados, e para o nosso exemplo modelo, podemos dizer que o HC apresentou um desempenho ligeiramente melhor quando comparado com o SA ou AG. Já o MC permitiu obter bons resultados quando considerado um elevado número de simulações. Estes resultados vão de encontro aos trabalhos de de [27] (que aponta o HC como eficaz na otimização de portfólio), de [4, 12] (onde se considera o SA como uma abordagem promissora para problemas de otimização) e também de [3] (onde se concluiu que o AG gera soluções de alta qualidade para problemas de busca e otimização). No entanto, o MC surge como uma abordagem alternativa promissora para os problemas de otimização de portfólio. Estes métodos realçam, acima de tudo, a facilidade de implementação.

Na segunda fase do trabalho, utilizámos a simulação de Monte Carlo (assumindo movimento Browniano geométrico), bem como os modelos ARIMA e Prophet para estimar a evolução dos preços em tempo futuro. Para efeitos de comparação de desempenho, os resultados foram avaliados utilizando as métricas de desempenho MAE e MAPE. De acordo com os resultados, o MC ajustou-se melhor aos nossos dados, apresentando um melhor valor de MAPE. No entanto, e de acordo com o critério em [35], podemos classificar todas estas previsões como sendo altamente precisas, dado terem um valor do MAPE $< 10\%$. Estes resultados estão em acordo com os trabalhos realizados por [1, 2, 49, 55] onde consideraram o MC como uma técnica poderosa na previsão, com alta precisão. Notem-se também os estudos de [5] e [15], onde se concluiu que o modelo ARIMA tem um grande potencial, produzindo previsões confiáveis. Ainda de focar o trabalho de [26], onde está estudado o método Prophet, que é uma das ferramentas recentes que está a mostrar um bom desempenho.

De um modo geral, acreditamos que a otimização e previsão utilizando técnicas de AM aumenta o poder de decisão dos investidores em alguns aspetos.

6.2 Limitações e Recomendações

À semelhança de outros modelos financeiros (por exemplo, Black-Scholes), a TMP está fortemente dependente de pressupostos que podem ser relevantes para as questões de seleção de portfólio. Note-se que o nosso problema modelo parte de pressupostos irrealistas (mercado perfeito, não há opção de venda, ou ainda custo de transição, etc.) e os avanços na teoria financeira já propõe novos e mais eficientes modelos que lidam com alguns - mas não todas - destes pressupostos.

Há também que ter em consideração outras limitações, como sejam os ativos considerados, em que quantidade, a taxa de risco considerada (no nosso exemplo modelo, tomámos uma taxa livre de risco de 0.01), ou inclusive, o horizonte temporal.

Este estudo destina-se a introduzir os investidores e gestores no básico da TMP, indicando alguns dos algoritmos já existentes no mercado e que podem garantir uma obtenção de portfólio ótimo (retorno elevado a baixo risco).

Tem também a vantagem de permitir a tomada de decisões políticas para desenvolvimento de estratégias de macro-estabilização com base na construção de portfólio ótimo. Chama-se aqui a atenção para o facto de que, independentemente dos meios e métodos disponíveis para o investidor, não existe um modelo genérico de otimização que responda a todas as necessidades do investidor individual. Alguns fatores, como sejam o horizonte temporal, a tolerância ao risco, os objetivos de investimento, os recursos financeiros e o nível de experiência de investimento pessoal, desempenham um papel fundamental na seleção de portfólios.

Dada a volatilidade dos mercados, é frequente que o retorno esperado não seja aquele que necessariamente o investidor acreditava ir receber, nem o mais provável. É, pelo contrário, o resultado de um processo estatístico, em que assumimos que alguns resultados são mais prováveis que outros.

Num mercado imperfeito, é da maior conveniência que os investidores sigam a lição de Markowitz e cuidem não só do retorno, mas também do risco.

6.3 Trabalho Futuro

Espera-se que esta investigação possa influenciar a melhoria das estratégias de previsão e otimização do portfólio.
Acreditamos que o presente trabalho pode ser continuado e aprimorado de diversas maneiras. Por exemplo, introduzir no modelo diferentes aspetos / parâmetros, como sejam as decisões individuais dos investidores, o seu receio em investir em produto de alto risco, bem como a incerteza do comportamento de mercado face a alterações inesperadas a nível mundial. Assim, seria de todo pertinente considerar outras métricas de desempenho possíveis, para além da MAE e MAPE e outras métricas que captem uma noção de risco mais vasta. Consideramos importante a introdução de caraterísticas reais como os custos de transação, preferências dos investidores.
Pretendemos também, em futuras investigações, considerar outros tipos de ativos dentro do portfólio e efetuar diferentes combinações dos modelos de previsão.

Referências Bibliográficas

[1] Abidin, S. N. Z., and Jaffar, M. M. A review on geometric brownian motion in forecasting the share prices in bursa malaysia. *World Applied Sciences Journal 17*, 1 (2012), 82–93.

[2] Agustini, W. F., Affianti, I. R., and Putri, E. R. Stock price prediction using geometric brownian motion. In *Journal of physics: conference series* (2018), vol. 974, IOP Publishing, p. 012047.

[3] Albadr, M. A., Tiun, S., Ayob, M., and Al-Dhief, F. Genetic algorithm based on natural selection theory for optimization problems. *Symmetry 12*, 11 (2020), 1758.

[4] Alkhateeb, F., and Abed-Alguni, B. H. A hybrid cuckoo search and simulated annealing algorithm. *Journal of Intelligent Systems 28*, 4 (2019), 683–698.

[5] Ariyo, A. A., Adewumi, A. O., and Ayo, C. K. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation* (2014), IEEE, pp. 106–112.

[6] Bodie, Z., Kane, A., and Marcus, A. *ISE Essentials of Investments*, 12 ed. McGraw-Hill Education, 2021.

[7] Borchers, B., and Mitchell, J. E. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research 21*, 4 (1994), 359–367.

[8] Bouyé, E. Multivariate extremes at work for portfolio risk measurement. *FINANCE-PARIS- 23*, 2 (2002), 125–144.

[9] Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

[10] Chakraborty, M., and Subramaniam, S. Asymmetric relationship of investor sentiment with stock return and volatility: evidence from india. *Review of Behavioral Finance* (2020).

[11] Chen, W., Zhang, H., Mehlawat, M. K., and Jia, L. Mean–variance portfolio optimization using machine learning-based stock price prediction. *Applied Soft Computing 100* (2021), 106943.

[12] Crama, Y., and Schyns, M. Simulated annealing for complex portfolio selection problems. *European Journal of operational research 150*, 3 (2003), 546–571.

[13] Delahaye, D., Chaimatanan, S., and Mongeau, M. Simulated annealing: From basics to applications. In *Handbook of metaheuristics*. Springer, 2019, pp. 1–35.

[14] Dewi, P. S., Widyanto, J. K., and Tick, A. Monte carlo simulation of stock movements–an indonesian case. In *2021 IEEE 19th International Symposium on Intelligent Systems and Informatics (SISY)* (2021), IEEE, pp. 101–106.

[15] Ediger, V. Ş., and Akar, S. Arima forecasting of primary energy demand by fuel in turkey. *Energy policy 35*, 3 (2007), 1701–1708.

[16] Elissaios Sarmas, Panos Xidonas, H. D. *Multicriteria Portfolio Construction with Python*, 1st ed. ed. Springer Optimization and Its Applications 163. Springer International Publishing;Springer, 2020.

[17] Fernando, K. Practical portfolio optimization. *The Numerical Algorithms Group, Ltd White Paper* (2000).

[18] Glasserman, P. *Monte Carlo methods in financial engineering*, vol. 53. Springer Science & Business Media, 2013.

[19] Hayes, G. mlrose: Machine Learning, Randomized Optimization and SEarch package for Python. https://github.com/gkhayes/mlrose, 2019.

[20] Healy, J. Variance reduction of ordinary monte-carlo estimates with the brownian-bridge path construction. *Wilmott 2018*, 96 (2018), 54–57.

[21] Heath, M. T. *Scientific Computing: An Introductory Survey*, revised 2nd edition ed. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2018.

[22] Heydt, M. *Mastering pandas for Finance: Master pandas, an open source Python Data Analysis Library, for financial data analysis*. Packt Publishing, 2015.

[23] Idzorek, T. A step-by-step guide to the black-litterman model: Incorporating user-specified confidence levels. In *Forecasting expected returns in the financial markets*. Elsevier, 2007, pp. 17–38.

[24] Jacobson, S. H., and Yücesan, E. Analyzing the performance of generalized hill climbing algorithms. *Journal of Heuristics 10*, 4 (2004), 387–405.

[25] Jensen, M. C. Some anomalous evidence regarding market efficiency. *Journal of financial economics 6*, 2/3 (1978), 95–101.

[26] Jha, B. K., and Pande, S. Time series forecasting model for supermarket sales using fb-prophet. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)* (2021), IEEE, pp. 547–554.

[27] John, C. High speed hill climbing algorithm for portfolio optimization. *Tanzania Journal of Science 47*, 3 (2021), 1236–1242.

[28] John Von Neumann, O. M. *Theory of Games and Economic Behavior*, 3rd ed. Princeton University Press, 1966.

[29] Kalayci, C. B., Ertenlice, Ö., Akyer, H., and Aygören, H. A review on the current applications of genetic algorithms in mean-variance portfolio optimization.

[30] Lai, K. K., Yu, L., and Wang, S. Mean-variance-skewness-kurtosis-based portfolio optimization. In *First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)* (2006), vol. 2, IEEE, pp. 292–297.

[31] Lai, T. L., Xing, H., and Chen, Z. Mean–variance portfolio optimization when means and covariances are unknown. *The Annals of Applied Statistics 5*, 2A (2011), 798–823.

[32] Laurent, M., and Vallentin, F. Semidefinite optimization. *Lecture Notes, available at http://page. mi. fu-berlin. de/fmario/sdp/laurentv. pdf* (2012).

[33] Leippold, M., Trojani, F., and Vanini, P. A geometric approach to multiperiod mean variance optimization of assets and liabilities. *Journal of Economic Dynamics and Control 28*, 6 (2004), 1079–1113.

[34] Lewinson, E. *Python for Finance Cookbook: Over 50 recipes for applying modern Python libraries to financial data analysis. Code.* Packt Publishing, 2020.

[35] Lewis, C. D. *Demand Forecasting and Inventory Control. A computer aided learning approach.* Woodhead Pub. in association with the Institute of Operations Management, 2000.

[36] Li, C., and Li, Z. Multi-period portfolio optimization for asset–liability management with bankrupt control. *Applied Mathematics and Computation 218*, 22 (2012), 11196–11208.

[37] Li, D., and Ng, W. L. Optimal dynamic portfolio selection: Multiperiod mean-variance formulation. *Mathematical finance 10*, 3 (2000), 387–406.

[38] Liang, Z., Chen, H., Zhu, J., Jiang, K., and Li, Y. Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940* (2018).

[39] Lobo, M. S., Fazel, M., and Boyd, S. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research 152*, 1 (2007), 341–365.

[40] Luenberger, D. G. *Investment science*, illustrated edition ed. Oxford University Press, 1998.

[41] Maringer, D. G. *Portfolio Management with Heuristic Optimization*, 1st edition. ed. Advances in Computational Management Science. Springer, 2010.

[42] Markowitz, H. M. *Portfolio Selection.* 1952.

[43] Markowitz, H. M. *Portfolio Selection: Efficient Diversification of Investments.* Cowles Foundation Monograph: No. 16. Yale University Press, 1968.

[44] Marsland, S. *Machine learning: an algorithmic perspective*, 1 ed. CRC Press, 2009.

[45] Mayambala, F. *Mean-Variance Portfolio Optimization: Eigendecomposition-Based Methods.* PhD thesis, Linköping University Electronic Press, 2015.

[46] McNeil, A. J. *Quantitative risk management : concepts, techniques and tools*, revised ed. Princeton Series in Finance. Princeton University Press, 2015.

[47] Nocedal, Jorge;Wright, S. J. *Numerical optimization*, 2. ed ed. Springer series in operations research and financial engineering. Springer, 2006.

[48] Pai, G. A. V. *Metaheuristics for Portfolio Optimization: An Introduction using MATLAB*, 1 ed. Metaheuristics Set. Wiley-ISTE, 2018.

[49] Parungrojrat, N., and Kidsom, A. Stock price forecasting: Geometric brownian motion and monte carlo simulation techniques. *MUT Journal of Business Administration 16*, 1 (2019), 90–103.

[50] Patrick, K. Comparison of simulated annealing and hill climbing in the course timetabling problem. *African Journal of Mathematics and Computer Science Research 5*, 11 (2012), 176–178.

[51] Prigent, J.-L. *Portfolio Optimization and Performance Analysis.* Chapman & Hall/CRC financial mathematics series. Chapman & Hall/CRC, 2007.

[52] Prügel-Bennett, A. When a genetic algorithm outperforms hill-climbing. *Theoretical Computer Science 320*, 1 (2004), 135–153.

[53] Raja, H. H. *Returns Optimization and Returns Prediction in Presence of Fear Sentiments Using Machine Learning Algorithms.* PhD thesis, CAPITAL UNIVERSITY, 2021.

[54] Roman, S. *Introduction to the Mathematics of Finance: From Risk Management to Options Pricing*, 1 ed. Undergraduate Texts in Mathematics. Springer, 2004.

[55] Seifoddini, J. Stock option pricing by augmented monte-carlo simulation models. *Advances in Mathematical Finance and Applications 6*, 4 (2021), 1–22.

[56] Sharpe, W. F. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance 19*, 3 (1964), 425–442.

[57] Shaw, D. X., Liu, S., and Kopman, L. Lagrangian relaxation procedure for cardinality-constrained portfolio optimization. *Optimisation Methods & Software 23*, 3 (2008), 411–420.

[58] Tadlaoui, G. Intelligent portfolio construction: Machine-learning enabled mean-variance optimization.

[59] Taylor, S. J., and Letham, B. Forecasting at scale. *The American Statistician 72*, 1 (2018), 37–45.

[60] Van Laarhoven, P. J., and Aarts, E. H. Simulated annealing. In *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.

[61] Yi, H., and Yang, J. Multi-objective portfolio optimization based on fuzzy genetic algorithm. In *2013 Ninth International Conference on Computational Intelligence and Security* (2013), IEEE, pp. 90–94.

[62] Zhang, G. P. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing 50* (2003), 159–175.

[63] Zunic, E., Korjenic, K., Hodzic, K., and Donko, D. Application of facebook's prophet algorithm for successful sales forecasting based on real-world data. *arXiv preprint arXiv:2005.07575* (2020).

Apêndice A

Gerando Caminhos de Amostra

A.1 Movimento Browniano a uma Dimensão

Por um movimento Browniano (MB) unidimensional padrão em $[0, T]$, queremos dizer um processo estocástico $W(t),\ 0 \leq t \leq T$ com as seguintes propriedades:

(i) $W(0) = 0$;

(ii) o mapeamento $t \to W(t)$ é, com probabilidade 1, uma função contínua em $[0, T]$;

(iii) os incrementos $W(t_1) - W(t_0), W(t_2) - W(t_1),\ \ldots, W(t_k) - W(t_{k-1})$ são independentes para qualquer k e quaisquer $0 \leq t_0 < t_1 < \ldots < t_k \leq T$;

(iv) $W(t) - W(s) \sim N(0, t - s)$ para quaisquer $0 \leq s < t \leq T$.

Em (iv) seria suficiente exigir que $W(t) - W(s)$ tenha média 0 e variância $t - s$; que a sua distribuição é, de facto, uma distribuição normal, é consequência da continuidade dos caminhos de amostra em (ii) e da propriedade de incrementos independentes (iii). A inclusão da condição de normalidade em (iv) é central nesta discussão. Uma segunda consequência de (i) e (iv) é que

$$W(t) \sim N(0, t), \tag{A.1}$$

para $0 < t \leq T$.

Para constantes μ e $\sigma > 0$, chamamos um processo $X(t)$ de movimento Browniano com drift μ e coeficiente de difusão σ^2 (abreviado $X \sim MB(\mu,\sigma^2)$ se

$$\frac{X(t) - \mu t}{\sigma}$$

é um MB padrão. Assim, podemos construir X a partir de um MB padrão W definindo

$$X(t) = \mu t + \sigma W(t).$$

Segue então que $X(t) \sim N(\mu t, \sigma^2 t)$ (ver (A.1)). Além disso, X resolve a equação diferencial estocástico (EDE)

$$dX(t) = \mu dt + \sigma dW(t).$$

A suposição de que $X(0) = 0$ é uma normalização natural, mas podemos construir um MB com parâmetros μ e σ^2 e valor inicial x_0 simplesmente adicionando x_0 a cada $X(t)$.
Para $\mu(t)$ e $\sigma(t) > 0$ determinísticos, mas variantes no tempo, podemos definir um Movimento Browniano com drift μ e coeficiente de difusão σ^2 através da EDE

$$dX(t) = \mu(t)dt + \sigma(t)dW(t);$$

ou seja,

$$X(t) = X(0) + \int_0^t \mu(s)ds + \int_0^t \sigma(s)dW(s),$$

onde $X(0)$ é uma constante arbitrária. O processo X tem caminhos de amostra contínuos e incrementos independentes. Cada incremento $X(t) - X(s)$ é normalmente distribuído com média

$$E\left[X(t) - X(s)\right] = \int_s^t \mu(u)du$$

e variância

$$Var\left[X(t) - X(s)\right] = Var\left[\int_s^t \sigma(u)dW(u)\right] = \int_s^t \sigma^2(u)du.$$

A.2 Construção de Caminhos Aleatórios

Ao discutir a simulação do movimento Browniano, focamos principalmente na simulação dos valores $(W(t_1), \ldots, W(t_n))$ ou $(X(t_1), \ldots, X(t_n))$ num conjunto fixo de pontos $0 < t_1 < \ldots < t_n$. Como o movimento Browniano tem distribuição normal de incrementos independentes, simular o $W(t_i)$ (ou $X(t_i)$) a partir dos seus incrementos é relativamente simples. Sejam $Z_1, \ldots, Z_n$ variáveis aleatórias normais independentes gerados por um dos métodos (ver, por exemplo, seção 2.3.2 em [18]). Para um movimento Browniano padrão fixe-se $t_0 = 0$ e $W(0) = 0$. Os valores subsequentes podem ser gerados da seguinte forma:

$$W(t_{i+1}) = W(t_i) + \sqrt{t_{i+1} - t_i}Z_{i+1}, \ i = 0, \ldots n-1. \tag{A.2}$$

Para $X \sim MB(\mu, \sigma^2)$ com μ e σ constantes e $X(0)$ dado, defina-se

$$X(t_{i+1}) = X(t_i) + \mu(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i}Z_{i+1} \ \ i = 0, \ldots n-1. \tag{A.3}$$

Para coeficientes dependentes do tempo, a recursão se torna

$$X(t_{i+1}) = X(t_i) + \int_{t_i}^{t_{i+1}} \mu(s)ds + \sqrt{\int_t^{t_i+1} \sigma^2(u)du}Z_{i+1} \ \ i = 0, \ldots n-1. \tag{A.4}$$

Os métodos em (A.2) - (A.4) são exatos no sentido de que a distribuição conjunta dos valores simulados $(W(t_1), \ldots, W(t_n))$ ou $(X(t_1), \ldots, X(t_n))$ coincide com a distribuição conjunta

do MB correspondente em $t_1, \ldots, t_n$. Claro que isso não diz nada sobre o que acontece entre $t_i's$. Podemos estender os valores simulados a outros valores temporais, por exemplo, por interpolação linear por partes; mas nenhum método de interpolação determinística dará o vetor alargado a distribuição conjunta correta. Os métodos em (A.2) - (A.4) são exatos nos pontos de tempo $t_1, \ldots, t_n$ mas sujeitos a erro de discretização, em comparação com verdadeiro MB, se interpolado deterministicamente para outros pontos de tempo.

Substituindo (A.4) pela aproximação de Euler

$$X(t_{i+1}) = X(t_i) + \mu(t_i)(t_{i+1} - t_i) + \sigma(t_i)\sqrt{t_{i+1} - t_i}Z_{i+1} \;\; i = 0, \ldots n-1, \tag{A.5}$$

geralmente introduziremos erros de discretização mesmo nos pontos $t_1, \ldots, t_n$, porque os incrementos não terão mais a média e a variância corretas.
O vetor $(W(t_1), \ldots, W(t_n))$ é uma transformação linear do vetor de incrementos $(W(t_1), W(t_2) - W(t_1), \ldots, W(t_n) - W(t_{n-1}))$. Uma vez que estes incrementos são independentes e normalmente distribuídos, segue da Propriedade da Transformação Linear que $(W(t_1), \ldots, W(t_n))$ tem uma distribuição normal multivariada. Simular $(W(t_1), \ldots, W(t_n))$ é, portanto, um caso especial do problema geral de gerar vetores normais. Embora a construção de passeio aleatório seja suficiente para a maioria das aplicações, é interessante e às vezes útil considerar métodos de amostragem alternativa.

Para aplicar qualquer um dos métodos, precisamos primeiro de encontrar o vetor médio e a matriz de covariância de $(W(t_1), \ldots, W(t_n))$. Para movimento Browniano padrão, sabemos de (A.1) que $E[W(t_i)] = 0$, donde o vetor médio é o vetor nulo. Para a matriz de covariância, considere-se primeiro quaisquer $0 < s < t < T$; usando a independência dos incrementos encontramos que

$$\begin{aligned} Cov[W(s), W(t)] &= Cov[W(s), W(s) + (W(t) - W(s))] \\ &= Cov\,[W(s), W(s)] + Cov\,[W(s), W(t) - W(s)] \\ &= s + 0 = s. \end{aligned} \tag{A.6}$$

Sendo C a matriz de covariância de $(W(t_1), \ldots, W(t_n))$, assim temos

$$C_{ij} = \min(t_i, t_j). \tag{A.7}$$

A.3 Movimento Browniano Geométrico (MBG)

Um processo estocástico $S(t)$ é um movimento Browniano geométrico se $\log S(t)$ for um movimento Browniano com valor inicial $\log S(0)$; em outras palavras, um movimento browniano geométrico é simplesmente um movimento Browniano exponencial.

Consequentemente, todos os métodos para simular o movimento Browniano tornam-se métodos para simular movimento Browniano geométrico por exponenciação.

O MBG é o modelo mais fundamental do valor de um ativo financeiro. O uso do movimento Browniano geométrico como modelo em finanças teve início década de 1960.

Considerando que o movimento Browniano comum pode assumir valores negativos - uma característica indesejável num modelo do preço de uma ação ou qualquer outro ativo de responsabilidade limitada - é de conveniência seguir um movimento Browniano geométrico, que é sempre positivo.

Mais concretamente, para o movimento Browniano geométrico, a mudança da percentagem (retornos)

$$\frac{S(t_2) - S(t_1)}{S(t_1)}, \frac{S(t_3) - S(t_2)}{S(t_2)}, \ldots, \frac{S(t_n) - S(t_{n-1})}{S(t_{n-1})} \tag{A.8}$$

é independente para $t_1 < t_2 < \ldots < t_n$, ao invés das mudanças absolutas $S(t_{i+1}) - S(t_i)$ usadas no MB.

Propriedades Básicas

Suponha que W seja um movimento Browniano padrão e X satisfaça

$$dX(t) = \mu dt + \sigma dW(t),$$

de modo que $X \sim MB(\mu, \sigma^2)$. Se definirmos $S(t) = S(0)exp(X(t)) \equiv f(X(t))$, então uma aplicação da fórmula de Itô mostra que

$$\begin{aligned} dS(t) &= f'(X(t))dX(t) + \frac{1}{2}\sigma^2 f''(X(t))d(t) \\ &= S(0)exp(X(t))[\mu dt + \sigma dW(t)] + \frac{1}{2}\sigma^2 S(0)exp(X(t))dt \\ &= S(t)\left(\mu + \frac{1}{2}\sigma^2\right)dt + S(t)\sigma dW(t). \end{aligned} \tag{A.9}$$

Em contraste, um processo geométrico de movimento Browniano é frequentemente especificado por uma equação diferencial estocástica (EDE) da forma

$$\frac{dS(t)}{S(t)} = \mu dt + \sigma W dt, \tag{A.10}$$

uma expressão que sugere um modelo Browniano dos "retornos instantâneos", $dS(t)/S(t)$. A comparação de (A.9) e (A.10) indica que os modelos são incoerentes e revelam uma ambiguidade no papel de "μ". Em (A.9), μ é o drift do movimento Browniano, que exponenciamos para definir $S(t)$ - drift de $logS(t)$. Em (A.10), $S(t)$ tem drift $\mu S(t)$ e (A.10) implica

$$d\log S(t) = \left(\mu + \frac{1}{2}\sigma^2\right)dt + S(t)\sigma dW(t), \tag{A.11}$$

como pode ser verificado pela fórmula de Itô ou comparação com (A.9).

Usaremos a notação $S \sim GBM(\mu, \sigma^2)$ para indicar que S é um processo de o tipo em (A.10). Nós referimo-nos a μ em (A.10) como o *parâmetro de drift*, embora não seja o drift de $S(t)$ ou de $\log S(t)$. Referimo-nos a σ em (A.10) como o *parâmetro de volatilidade* de $S(t)$; o coeficiente

de difusão de $S(t)$ é então $\sigma^2 S^2(t)$.

De (A.11) vemos que se $S \sim MBG(\mu, \sigma^2)$ e se S tem valor inicial $S(0)$, então

$$S(t) = S(0)exp\left(\left[\mu - \frac{1}{2}\sigma^2\right]t + \sigma W(t)\right). \tag{A.12}$$

E mais geralmente, se $u < t$ então

$$S(t) = S(u)exp\left(\left[\mu - \frac{1}{2}\sigma^2\right](t-u) + \sigma(W(t) - W(u))\right), \tag{A.13}$$

a partir do qual fica evidente a independência dos retornos em (A.8). Além disso, como os incrementos de W são independentes e normalmente distribuído, isso fornece um procedimento recursivo simples para simular valores de S em $0 = t_0 < t_1 < \ldots < t_n$:

$$S(t_{i+1}) = S(t_i)exp\left(\left[\mu - \frac{1}{2}\sigma^2\right](t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i}Z_{i+1}\right), \tag{A.14}$$

onde as amostras aleatórias $Z_1, Z_2, \ldots, Z_n$ seguem uma distribuição normal independente para cada $i = 0, 1, \ldots, n-1$. De facto, (A.14) é equivalente para exponenciar ambos os lados de (A.3) com μ substituído por $\mu - \frac{1}{2}\sigma^2$. Este método é exato no sentido de que o vetor $(S(t_1), \ldots, S(t_n))$ produzido tem a distribuição conjunta do processo $S \sim MBG(\mu, \sigma^2)$ em $t_1, \ldots, t_n$ - isto é, o método não envolve nenhum erro de discretização. Os parâmetros dependentes do tempo podem ser incorporados por exponenciar ambos os lados de (A.4).

Apêndice B

Projeto - Otimização e Previsão

B.1 Otimização de Portfólios

```
!pip install yfinance
```

```
import pandas as pd
import numpy as np
from pandas_datareader import data
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import norm
import plotly.express as px
import yfinance as yf
import datetime
```

```
Portfólio = ['AMZN', 'GOOG', 'NFLX', 'AAPL'] # Ações do Portfólio
```

```
preços = pd.DataFrame()
for ação in Portfólio:
 preços[Portfólio]=yf.download(Portfólio,start='2015-01-01',end='2020-01-01')['Close']
```

```
preços.to_csv('dados.csv')  # Guardado como _csv
preços
```

	AMZN	GOOG	NFLX	AAPL
Date				
2015-01-02	27.332500	308.519989	523.373108	49.848572
2015-01-05	26.562500	302.190002	512.463013	47.311428
2015-01-06	26.565001	295.290009	500.585632	46.501431
2015-01-07	26.937500	298.420013	499.727997	46.742859

```
2015-01-08  27.972500 300.459991  501.303680  47.779999
   ...        ...          ...        ...         ...
2019-12-24  71.067497 1789.209961 1343.560059 333.200012
2019-12-26  72.477501 1868.770020 1360.400024 332.630005
2019-12-27  72.449997 1869.800049 1351.890015 329.089996
2019-12-30  72.879997 1846.890015 1336.140015 323.309998
2019-12-31  73.412498 1847.839966 1337.020020 323.570007
```

```
Primeira_data=preços.iloc[0] # Locamos a primeira data para podermos normalizar
Primeira_data
AMZN      27.332500
GOOGL    308.519989
NFLX     529.549988
AAPL      49.848572
Name: 2015-01-02 00:00:00, dtype: float64
```

```
preços_normalizados=preços/Primeira_data # Normalizamos os preços
preços_normalizados
```

```
               AMZN     GOOG        NFLX        AAPL
Date
2015-01-02  1.000000  1.000000  1.000000  1.000000
2015-01-05  0.971828  0.979483  0.979154  0.949103
2015-01-06  0.971920  0.957118  0.956460  0.932854
2015-01-07  0.985548  0.967263  0.954822  0.937697
2015-01-08  1.023415  0.973875  0.957832  0.958503
   ...        ...       ...       ...         ...
2019-12-24  2.600110  5.799332  2.567117  6.684244
2019-12-26  2.651697  6.057209  2.599293  6.672809
2019-12-27  2.650690  6.060548  2.583033  6.601794
2019-12-30  2.666423  5.986290  2.552940  6.485843
2019-12-31  2.685905  5.989369  2.554621  6.491059
```

```
preços_normalizados.to_csv('pd.csv') # Guardamos os preços normalizados
```

```
capital_inicial=5000 # Definimos um valor do investimento inicial
```

```
''' Definimos uma função que  que recebe a nossa base de dados normalizados,
o capital inicial e os melhores pesos, se não, gera pesos aleatorios'''

np.random.seed(2022) # Semente
def soma(preços_normalizados,dinheiro,melhores_pesos=[]):
  preços_normalizados=preços_normalizados.copy()
  if len(melhores_pesos)>0:
```

```
    pesos=melhores_pesos
  else:
    pesos=np.random.random(len(preços_normalizados.columns)-1)
    pesos=pesos/pesos.sum()
  #print(pesos,pesos.sum())
  for i, ação in enumerate(preços_normalizados.columns[1:]):
    preços_normalizados[ação]=preços_normalizados[ação]*pesos[i]*dinheiro
  preços_normalizados['Retorno_total']=preços_normalizados.sum(axis=1)
  ações_pesos1=pd.DataFrame(data={'Ações':preços_normalizados.columns[1:5],
  'Pesos':pesos*100})
  #print(preços_normalizados)
  return preços_normalizados, ações_pesos1,
  preços_normalizados.loc[len(preços_normalizados)-1]['Retorno_total']
```

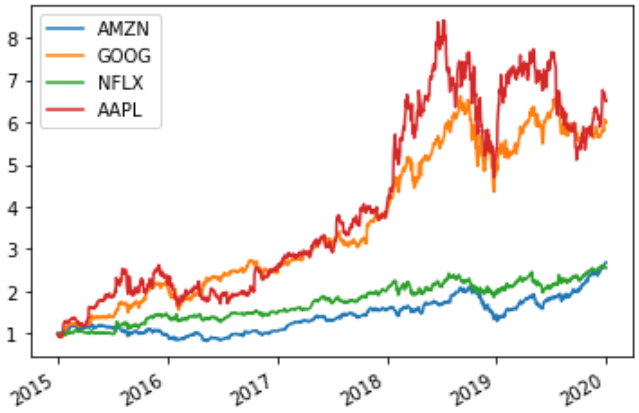
```
dados,a_p,Soma_valor=soma(pd.read_csv('pd.csv'),capital_inicial,melhores_pesos=[])
# Chamamos a função soma
```

```
      Date          AMZN        GOOG          NFLX         AAPL        Soma_valor
0   2015-01-02    69.655955   3714.476239   843.912294   371.955511    5000.000000
1   2015-01-05    67.693635   3638.265343   826.320325   353.024084    4885.303388
2   2015-01-06    67.700008   3555.191752   807.168659   346.980119    4777.040538
3   2015-01-07    68.649310   3592.875952   805.785765   348.781587    4816.092614
4   2015-01-08    71.286972   3617.436597   808.326474   356.520422    4853.570466
...     ...      ...      ...      ...      ...      ...
1253  2019-12-24  181.113119  21541.482314  2166.421688  2486.241359   26375.258480
1254  2019-12-26  184.706466  22499.358489  2193.575269  2481.988130   27359.628355
1255  2019-12-27  184.636373  22511.759694  2179.853314  2455.573619   27331.823000
1256  2019-12-30  185.732215  22235.930637  2154.457247  2412.444953   26988.565051
1257  2019-12-31  187.089276  22247.367727  2155.876210  2414.385070   27004.718283
```

```
a_p
   Ações  Pesos
0 AMZN  1.393119
1 GOOG  74.289525
2 NFLX  16.878246
3 AAPL  7.439110

soma_valor
27004.718282873455
```

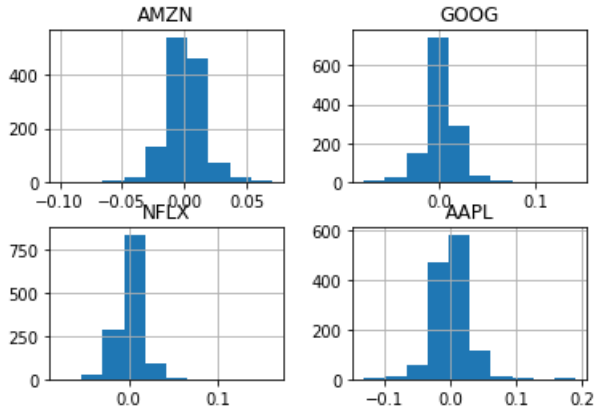
```
preços_normalizados.plot() # Plotamos os preços normalizados
```




```
retorno=preços/preços.shift(1)-1 # Calculamos a taxa de retorno diário
retorno
```

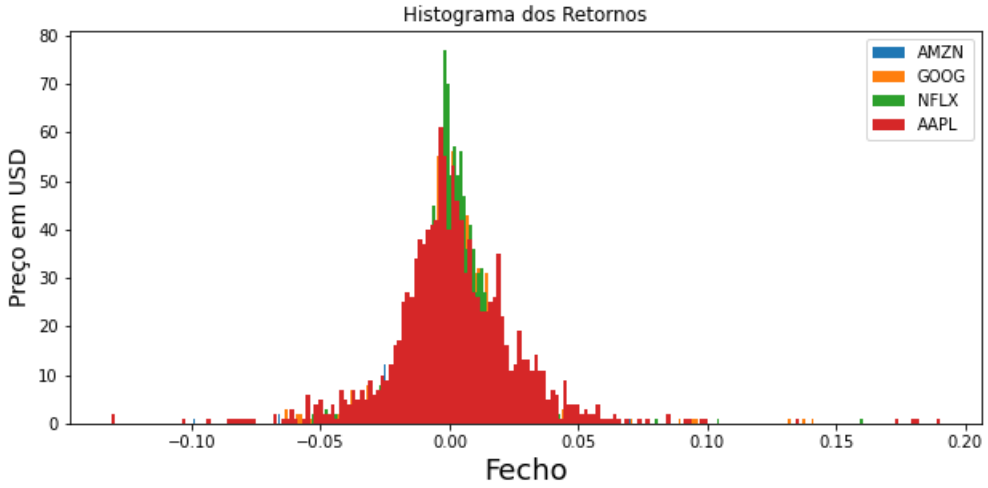
```
            AMZN     GOOG     NFLX     AAPL
Date
2015-01-02       NaN      NaN      NaN      NaN
2015-01-05  -0.028172 -0.020517 -0.020846 -0.050897
2015-01-06  0.000094  -0.022833 -0.023177 -0.017121
2015-01-07  0.014022  0.010600  -0.001713 0.005192
2015-01-08  0.038422  0.006836  0.003153  0.022188
  ...          ...       ...       ...       ...
2019-12-24  0.000951  -0.002114 -0.003914 0.000300
2019-12-26  0.019840  0.044467  0.012534  -0.001711
2019-12-27  -0.000379 0.000551  -0.006256 -0.010642
2019-12-30  0.005935  -0.012253 -0.011650 -0.017564
2019-12-31  0.007307  0.000514  0.000659  0.000804
```

```
retorno.hist() # Histograma
```

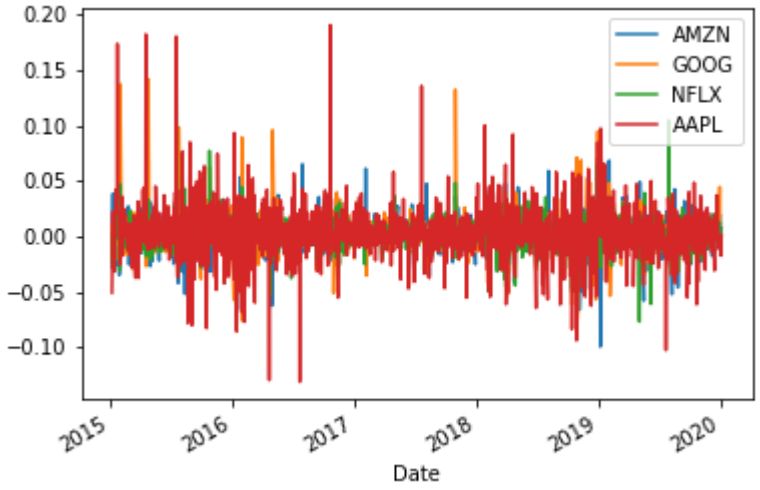



```
plt.figure(figsize=(12.2,4.5))
```

```
for i in retorno.columns.values:
  plt.hist( retorno[i],  label=i, bins = 200)
plt.title('Histograma dos Retornos')
plt.xlabel('Fecho',fontsize=18)
plt.ylabel('Preço em USD',fontsize=18)
plt.legend(retorno.columns.values)
plt.show()
```




```
retorno.plot(); # Plot dos retornos diarios
```




```
retorno_medio=retorno.mean() # Retorno médio diário
retorno_medio
AMZN      0.000909
GOOGL     0.001593
NFLX      0.000850
```

```
AAPL     0.001829
dtype: float64
```

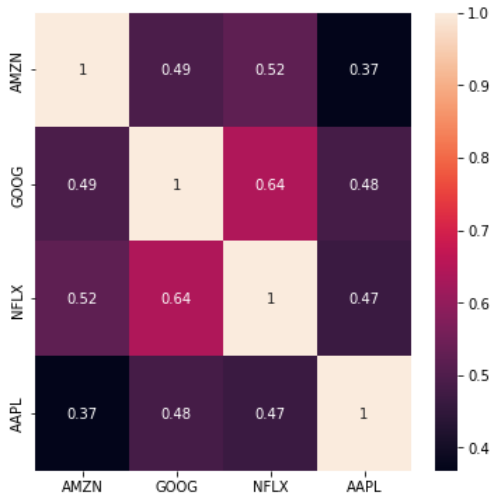
```
covariancia_retorno=retorno.cov() # Matriz de Covariância
covariancia_retorno
```

	AMZN	GOOG	NFLX	AAPL
AMZN	0.000245	0.000142	0.000123	0.000151
GOOG	0.000142	0.000340	0.000179	0.000233
NFLX	0.000123	0.000179	0.000229	0.000185
AAPL	0.000151	0.000233	0.000185	0.000689

```
correlacao_retorno=retorno.corr() # Matriz de Correlação
correlacao_retorno
```

	AMZN	GOOG	NFLX	AAPL
AMZN	1.000000	0.492717	0.521287	0.368179
GOOG	0.492717	1.000000	0.640953	0.482430
NFLX	0.521287	0.640953	1.000000	0.466755
AAPL	0.368179	0.482430	0.466755	1.000000

```
plt.figure(figsize=(6,6))
sns.heatmap(correlacao_retorno, annot=True);
```




```
# Exemplo: Definimos pesos iguais para os ativos e determinamos o retorno e o risco.
pesos=np.array([0.25,0.25,0.25,0.25])
```

```
retorno_portfolio=retorno_medio.dot(pesos)
retorno_portfolio

0.0014635647706924633
```

```
# Retorno médio anual
retorno_anual=retorno_portfolio*252
retorno_anual
0.32698090036859606
```

```
volatilidade_portfolio=np.sqrt(pesos.T.dot(covariancia_retorno.dot(pesos)))
volatilidade_portfolio
0.015843768833273285
```

```
volatilidade_anual =volatilidade_portfolio *np.sqrt(252)
volatilidade_anual
0.25151203297702257
```

B.1.1 Simulação de Monte Carlo - Alocação Aleatória

```
# Definimos as listas vazias
mc_retorno=[]
mc_volatilidade=[]
mc_pesos=[]
mc_sharpe=[]
numero_ativos=len(Portfolio) # Gera pesos de comprimento = 4
num_simul=10000
rf = 0.01 # 1% para a taxa livre de risco
```

```
for i in range(num_simul):
  pesos=np.random.random(numero_ativos)
  pesos/=np.sum(pesos)
  mc_pesos.append(pesos)
  mc_retorno.append(np.sum(retorno_medio*pesos)*252)
  mc_volatilidade.append(np.sqrt(np.dot(pesos.T,np.dot(covariancia_retorno*252,pesos))
  mc_sharpe.append(((np.sum(retorno_medio*pesos)*252)-rf)
  /(np.sqrt(np.dot(pesos.T,np.dot(covariancia_retorno*252,pesos)))))
```

```
Dados={'Retorno':mc_retorno,'Volatilidade':mc_volatilidade,'Sharpe Ratio':mc_sharpe}
# Dados
```

```
for counter, symbol in enumerate(precos.columns.tolist()):
  #print(counter, symbol)
  Dados[symbol+'-Pesos'] = [w[counter] for w in mc_pesos]
```

```
portfolio_mc=pd.DataFrame(Dados)
portfolio_mc
```

```
      Ret     Vol     Sharpe  AMZN-Pesos  GOOG-Pesos  NFLX-Pesos  AAPL-Pesos
0     0.333250  0.239506  1.349652  0.184182  0.281132  0.279196  0.255491
1     0.373478  0.264088  1.376353  0.022088  0.425739  0.233058  0.319115
2     0.335760  0.254370  1.280652  0.357054  0.105198  0.147779  0.389969
3     0.288511  0.219454  1.269108  0.320215  0.347748  0.317042  0.014994
4     0.365271  0.252616  1.406369  0.085678  0.551602  0.175638  0.187082
...     ...       ...         ...       ...         ...       ...       ...
9995  0.282591  0.217080  1.255716  0.321327  0.210433  0.373699  0.094542
9996  0.326671  0.237285  1.334564  0.130242  0.297385  0.353349  0.219025
9997  0.372816  0.254071  1.428011  0.172904  0.598239  0.050616  0.178241
9998  0.314535  0.239925  1.269294  0.061171  0.215756  0.488470  0.234602
9999  0.262006  0.213499  1.180364  0.381693  0.211574  0.400415  0.006318
```

```
volatilidade_minima_por=portfolio_mc.iloc[portfolio_mc['Volatilidade'].idxmin()]
volatilidade_minima_por
```

```
Retorno         0.240367
Volatilidade    0.211913
Sharpe Ratio    1.087086
AMZN-Pesos      0.446447
GOOG-Pesos      0.086153
NFLX-Pesos      0.458136
AAPL-Pesos      0.009263
Name: 4858, dtype: float64
```

```
max_sharpe_por=portfolio_mc.iloc[portfolio_mc['Sharpe Ratio'].idxmax()]
max_sharpe_por

Retorno         0.385621
Volatilidade    0.261464
Sharpe Ratio    1.436608
AMZN-Pesos      0.160074
GOOG-Pesos      0.633669
NFLX-Pesos      0.001728
AAPL-Pesos      0.204529
Name: 9372, dtype: float64
```

B.2 Otimização com Algoritmos

```
!pip install mlrose # Instalamos mlrose
```

```
import mlrose
```

```
dataset_original = pd.read_csv('/content/dados.csv') # Base de dados
```

```
# Definimos a função fitness()
def fitness_function(solucao):
  preços_normalizados1 = preços_normalizados.copy()
  pesos = solucao/solucao.sum()
  #print(pesos,solucao.sum())
  Retorno=(np.sum(retorno_medio*pesos)*252)
  Volatilidade=(np.sqrt(np.dot(pesos.T,np.dot(covariancia_retorno*252,pesos))))
  sharpe_ratio=(Retorno-rf)/(Volatilidade)
  return sharpe_ratio
```

```
# Confirmar se fitness funtion() funciona!
a=np.array(max_sharpe_por[3:])
fitness_function(a)
# Passamos os pesos obtido no portfólio de max sharpe
1.4366077293327097
```

```
fitness = mlrose.CustomFitness(fitness_function) # Passamos a nossa função
```

```
# Definimos o problema de maximização
problema_maximização = mlrose.ContinuousOpt(length=len(Portfolio), fitness_fn=fitness,
maximize = True, min_val = 0, max_val = 1)
```

B.2.1 Hill Climb

```
melhor_solução, melhor_Sharpe=mlrose.hill_climb(problema_maximizacao,random_state = 1)
melhor_solução=melhor_solução/melhor_solução.sum()
melhor_solução, melhor_Sharpe,melhor_solução.sum()
# Obtemos os pesos e o sharpe
(array([0.16199552, 0.61232911, 0.        , 0.22567537]),
1.4378606499964712,1.0)
```

```
_,_,_=soma(pd.read_csv('pd.csv'),capital_inicial,melhores_pesos=melhor_solução)

a_p,soma_valor
(  Ações      Pesos
0  AMZN  16.199552
1  GOOG  61.232911
2  NFLX   0.000000
3  AAPL  22.567537, 27837.207333661852)
```

B.2.2 Simulated Annealing

```
melhor_soluca,melhor_Sharpe=mlrose.simulated_annealing(problema_maximização,
 random_state=1)
melhor_soluca = melhor_soluca / melhor_soluca.sum()
melhor_soluca, melhor_Sharpe,melhor_soluca.sum()
(array([0.23076923, 0.53846154, 0.        , 0.23076923]),
1.4336905714611892,1.0)
```

```
_,soma_valor=soma(pd.read_csv('pd.csv'),capital_inicial,melhores_pesos=melhor_soluca)
a_p,soma_valor
(  Ações      Pesos
0  AMZN  23.076923
1  GOOG  53.846154
2  NFLX   0.000000
3  AAPL  23.076923, 26714.02768741193)
```

B.2.3 Algoritmo Genético

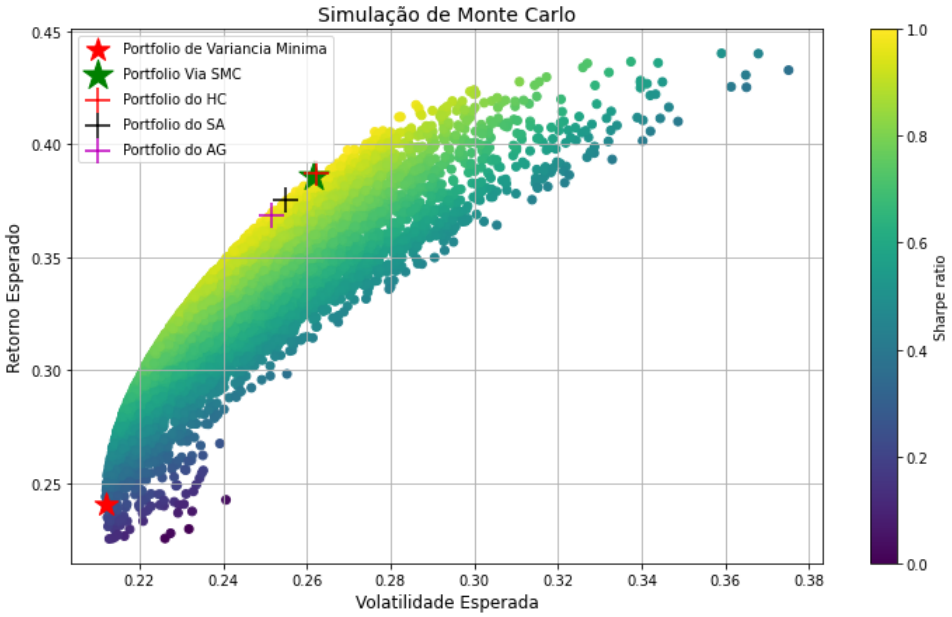
```
problema_maximização_ag = mlrose.ContinuousOpt(length = len(Portfolio),
fitness_fn = fitness, maximize = True, min_val = 0.0, max_val = 1)
```

```
melhor_solução1, melhor_Sharpe= mlrose.genetic_alg(problema_maximização_ag,
 random_state = 1)
melhor_solução1 = melhor_solução1 / melhor_soluçao1.sum()
melhor_solução1, melhor_Sharpe,melhor_solução1.sum()
(array([0.19462915, 0.52604309, 0.06396743, 0.21536033]),
1.4266447532046367,1.0)
```

```
_,_,_=soma(pd.read_csv('pd.csv'),capital_inicial,melhores_pesos=melhor_solucao1)
a_p, soma_valor
(  Ações      Pesos
0  AMZN  19.462915
1  GOOG  52.604309
2  NFLX   6.396743
3  AAPL  21.536033, 26173.752574423324)
```

```
  volatilidade  retorno   sharpe  AMZN-pesos  GOOG-pesos  NFLX-pesos  AAPL-pesos
Alg.
HC  0.262104    0.386869  1.437861  0.161996  0.612329      0.000000  0.225675
SA  0.254816    0.375328  1.433691  0.230769  0.538462      0.000000  0.230769
AG  0.251512    0.368818  1.426645  0.194629  0.526043      0.063967  0.215360
```

```
color=(portfolio_mc['Retorno']-rf)/portfolio_mc['Volatilidade']
plt.figure(figsize=(10,6))
plt.scatter(portfolio_mc['Volatilidade'],portfolio_mc['Retorno'],c=color,marker='o')
plt.scatter(volatilidade_minima_por[1], volatilidade_minima_por[0], color='r',
marker='*', s=300,label='Portfolio de Variancia Minima')
plt.scatter(max_sharpe_por[1], max_sharpe_por[0], color='g', marker='*', s=500,
label='Portfolio Via SMC')
plt.scatter(por_alg['volatilidade'][0],por_alg['retorno'][0], color='b', marker='+',
s=300,label='Portfolio do HC')
plt.scatter(por_alg['volatilidade'][1],por_alg['retorno'][1], color='black',
 marker='+',s=300,label='Portfolio do SA')
plt.scatter(por_alg['volatilidade'][2],por_alg['retorno'][2], color='m', marker='+',
s=300,label='Portfolio do AG')
plt.legend(labelspacing=0.8)
plt.grid()
plt.title('Simulação de Monte Carlo',fontsize=14)
plt.xlabel('Volatilidade Esperada',fontsize=12)
plt.ylabel('Retorno Esperado',fontsize=12)
plt.colorbar(label='Sharpe ratio')
plt.show()
```



B.3 Previsão de Preços

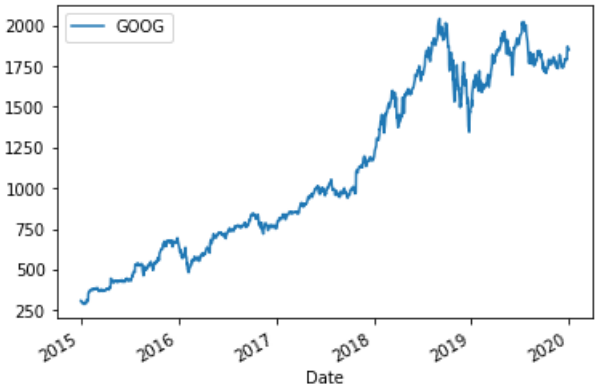
B.3.1 Movimento Browniano Geométrico e Simulação de Monte Carlo

```
import datetime
```

```
data=pd.read_csv('/content/dados.csv')
data # Chamamos a nossa base de dados
```

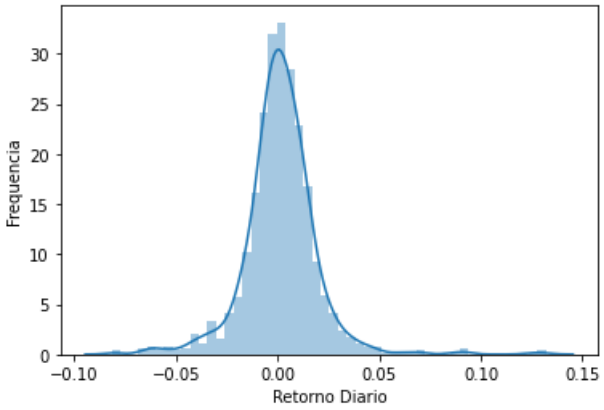
```
dataconvers=lambda  dates:datetime.datetime.strptime(dates,'%Y-%m-%d')
Empresa=pd.read_csv('dados.csv',parse_dates=['Date'],index_col='Date',
date_parser=dataconvers, usecols=['Date','GOOG']) # Convertemos datas em indices
Empresa
```

```
Empresa.plot();
```




```
retorno_log = np.log(1 + Empresa.pct_change())
retorno_log # Calculamos os retornos diarios
```

```
sns.distplot(retorno_log.iloc[1:], xlabel("Retorno,ylabel("Frequência");
```



```
mu = retorno_log.mean() # média dos retornos
var = retorno_log.var() # variância
drift = mu - (0.5*var) # drift
```

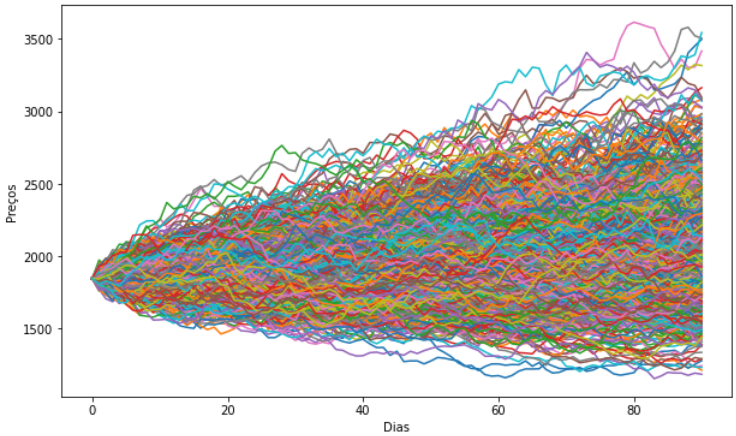
```
np.random.seed(2000) # uma semente aleatória
std = retorno_log.std() # Desvio Padrão
dias_f = 91 # O primeiro valor conincide com o último preço real (não sendo previsão).
n_sim = 1000 # número de simulações
Z = norm.ppf(np.random.rand(dias_f, n_sim)) #normalizar
retorno_diario = np.exp(drift.values + std.values * Z)
retorno_diario
```

```
preços_fut = np.zeros_like(retorno_diario)
preços_fut
```

```
preços_fut [0] = Empresa.iloc[-1]
preços_fut '''passamos o último de base de dados para primeiro da simulação
e depois cada um segue o seu caminho com base nos retornos diarios aleatorios'''
```

```
for t in range(1, dias_f):
preços_fut[t] = preços_fut[t-1]*retorno_diario[t]
preços_fut
```

```
plt.figure(figsize=(10,6))
plt.title("1000 Simulações de MC",loc='center',color='g', fontsize=22)
plt.xlabel('Dias')
plt.ylabel('Preços')
for i in range(len(preços_fut.T)):
  plt.plot(preços_fut.T[i])
```




```
Portfolio=["AMZN","GOOG","NFLX","AAPL"]
```

```
dados_do_mercado=pd.DataFrame()
for i in Portfolio:
  dados_do_mercado[Portfolio]=yf.download(Portfolio, start='2020-01-01',
 end='2020-05-12')['Close']
dados_Goo_merc=pd.DataFrame(dados_do_mercado['GOOG'])
dados_Goo_merc # Chamamos os dados do mercado (Preços Reais)
          GOOG
Date
2020-01-02  1898.010010
2020-01-03  1874.969971
2020-01-06  1902.880005
2020-01-07  1906.859985
2020-01-08  1891.969971
  ...          ...
2020-05-05  2317.800049
2020-05-06  2351.260010
2020-05-07  2367.610107
2020-05-08  2379.610107
2020-05-11  2409.000000
```

```
  # Importamos as métricas
from sklearn.metrics import mean_absolute_error # MAE
from sklearn.metrics import mean_absolute_percentage_error # MAPE
```

```
erros=[]
for i in range(len(preços_fut.T)):
  simulacao1=precos_fut.T[i][0:len(dados_Goo_merc)]
erros.append(mean_absolute_error(dados_Goo_merc,simulação1))
erros=np.array(erros)
```

```
df={"Erros": erros}
err=pd.DataFrame(df)
err
```

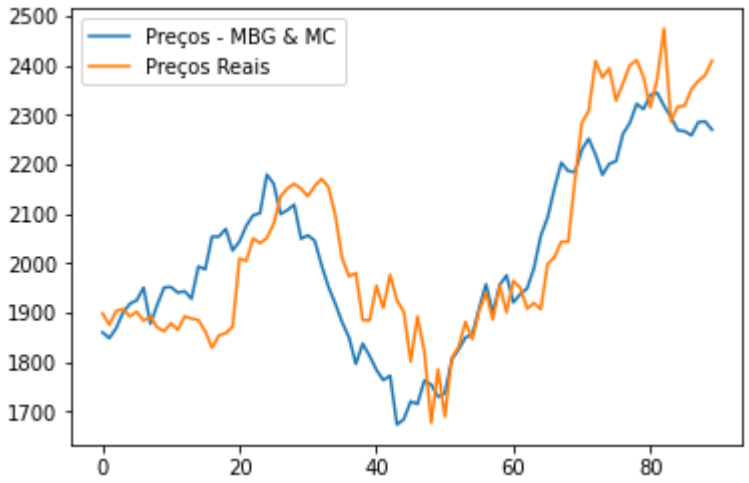
```
menor_erro=err.iloc[err['Erros'].idxmin()]
menor_erro.name
```

```
  # MAE
mean_absolute_error(dados_Goo_merc,preços_fut.T[menor_erro.name][1:])
88.9477785364924
```

```
  # MAPE
mean_absolute_percentage_error(dados_Goo_merc,preços_fut.T[menor_erro.name][1:])
0.04366527883107488
```

```
          Preços - MBG & MC Preços Reais
Date
2020-01-02  1859.710929   1898.010010
2020-01-03  1847.731820   1874.969971
2020-01-06  1868.057923   1902.880005
2020-01-07  1899.587018   1906.859985
2020-01-08  1917.320397   1891.969971
  ...          ...           ...
2020-05-05  2266.416853   2317.800049
2020-05-06  2258.047848   2351.260010
2020-05-07  2284.775258   2367.610107
2020-05-08  2286.207293   2379.610107
2020-05-11  2269.441065   2409.000000
```

```
prev_mbg_mc.plot()
```



B.3.2 Previão com ARIMA

```
!pip install pmdarima
```

```
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima.arima import auto_arima
import pandas as pd
import datetime
```

```
preços=pd.read_csv('/content/dados.csv')
preços
```

```
              AMZN       GOOG         NFLX         AAPL
  Date
  2015-01-02  27.332500 308.519989  523.373108  49.848572
  2015-01-05  26.562500 302.190002  512.463013  47.311428
  2015-01-06  26.565001 295.290009  500.585632  46.501431
  2015-01-07  26.937500 298.420013  499.727997  46.742859
  2015-01-08  27.972500 300.459991  501.303680  47.779999
      ...        ...           ...         ...         ...
  2019-12-24  71.067497 1789.209961 1343.560059 333.200012
  2019-12-26  72.477501 1868.770020 1360.400024 332.630005
  2019-12-27  72.449997 1869.800049 1351.890015 329.089996
  2019-12-30  72.879997 1846.890015 1336.140015 323.309998
  2019-12-31  73.412498 1847.839966 1337.020020 323.570007
```

```
dataconvers=lambda  dates:datetime.datetime.strptime(dates,'%Y-%m-%d')
dados_treino=pd.read_csv('dados.csv',parse_dates=['Date'],index_col='Date',
date_parser=dataconvers, usecols=['Date','GOOG'])
dados_treino
```

```
            GOOG
Date
2015-01-02  308.519989
2015-01-05  302.190002
2015-01-06  295.290009
2015-01-07  298.420013
2015-01-08  300.459991
    ...        ...
2019-12-24  1789.209961
2019-12-26  1868.770020
2019-12-27  1869.800049
2019-12-30  1846.890015
2019-12-31  1847.839966
```

```
dados_Goo_merc # Dados do Mercado
```

```
            GOOG
Date
2020-01-02  1898.010010
2020-01-03  1874.969971
2020-01-06  1902.880005
2020-01-07  1906.859985
2020-01-08  1891.969971
    ...          ...
2020-05-05  2317.800049
```

```
2020-05-06  2351.260010
2020-05-07  2367.610107
2020-05-08  2379.610107
2020-05-11  2409.000000
```

```
dados_treino.plot()
```


```
modelo=auto_arima(dados_treino,suppress_warnings=True,error_action='ignore')
```

```
modelo.order
(0,1,0)
```

```
previsões_arima=modelo.predict(n_periods=90)
previsões_arima
```

```
array([1849.06456405, 1850.28916228, 1851.51376051, 1852.73835875, 1853.96295698,
 1855.18755521, 1856.41215344, 1857.63675167,1858.8613499 , 1860.08594813,
  1861.31054637, 1862.5351446 ,1863.75974283, 1864.98434106, 1866.20893929,
  1867.43353752,1868.65813575, 1869.88273398, 1871.10733222, 1872.33193045,
  1873.55652868, 1874.78112691, 1876.00572514, 1877.23032337,
  1878.4549216 ,
  1879.67951984, 1880.90411807, 1882.1287163 ,1883.35331453, 1884.57791276,
   1885.80251099, 1887.02710922,1888.25170745, 1889.47630569, 1890.70090392,
    1891.92550215,1893.15010038, 1894.37469861, 1895.59929684, 1896.82389507,
    1898.04849331, 1899.27309154, 1900.49768977, 1901.722288 ,1902.94688623,
    1904.17148446,
    1905.39608269, 1906.62068093,1907.84527916, 1909.06987739, 1910.29447562,
    1911.51907385,1912.74367208, 1913.96827031, 1915.19286854, 1916.41746678,
    1917.64206501, 1918.86666324, 1920.09126147, 1921.3158597 ,1922.54045793,
    1923.76505616, 1924.9896544 , 1926.21425263,1927.43885086, 1928.66344909,
    1929.88804732, 1931.11264555,1932.33724378, 1933.56184201, 1934.78644025,
    1936.01103848,1937.23563671,1938.46023494, 1939.68483317, 1940.9094314,
    1942.13402963, 1943.35862787, 1944.5832261,1945.80782433,1947.03242256,
```

```
    1948.25702079, 1949.48161902, 1950.70621725,1951.93081549, 1953.15541372,
    1954.38001195, 1955.60461018,1956.82920841, 1958.05380664])
```

```
previsões_arima=pd.DataFrame(modelo.predict(n_periods=90), index=dados_Goo_merc.index)
previsões_arima.columns=['Preços_Arima']
previsões_arima
```

```
          Preços_Arima
Date
2019-12-31  1849.064564
2020-01-02  1850.289162
2020-01-03  1851.513761
2020-01-06  1852.738359
2020-01-07  1853.962957
    ...         ...
2020-05-04  1953.155414
2020-05-05  1954.380012
2020-05-06  1955.604610
2020-05-07  1956.829208
2020-05-08  1958.053807
```

```
plt.figure(figsize=(12,5))
plt.plot(dados_treino,label ='Treinamento')
plt.plot(dados_Goo_merc, label ="Teste")
plt.plot(previsões_arima,label='Previsões')
plt.grid()
plt.legend();
```


```
mean_absolute_error(dados_Goo_merc,previsões_arima) # MAE
```

```
161.07711416409862
```

```
mean_absolute_percentage_error(dados_Goo_merc,previsões_arima) # MAPE
```

```
0.0733513311660005
```

```
previsões_arima['Preços Reais']=dados_Goo_merc['GOOG']
previsões_arima
```

```
          Preços_Arima   Preços Reais
Date
2019-12-31  1849.064564   1898.010010
2020-01-02  1850.289162   1874.969971
2020-01-03  1851.513761   1902.880005
2020-01-06  1852.738359   1906.859985
2020-01-07  1853.962957   2315.989990
    ...         ...          ...
2020-05-04  1953.155414   2317.800049
2020-05-05  1954.380012   2351.260010
2020-05-06  1955.604610   2367.610107
2020-05-07  1956.829208   2379.610107
2020-05-08  1958.053807   2409.000000
```

B.3.3 Previsão com Prophet

```
!pip install fbprophet
```

```
from fbprophet import Prophet
import pandas as pd
```

```
preços_goog=preços_p[['Date','GOOG']].rename(columns={'Date':'ds','GOOG':'y'})
preços_goog
```

```
      ds            y
0 2015-01-02      308.519989
1 2015-01-05      302.190002
2 2015-01-06      295.290009
3 2015-01-07      298.420013
4 2015-01-08      300.459991
...     ...         ...
1253  2019-12-24  1789.209961
1254  2019-12-26  1868.770020
1255  2019-12-27  1869.800049
1256  2019-12-30  1846.890015
1257  2019-12-31  1847.839966
```

```
model = Prophet()
model.fit(preços_goog)
```

```
futuro=model.make_future_dataframe(periods=90)
previsões_fprof=model.predict(futuro)
```

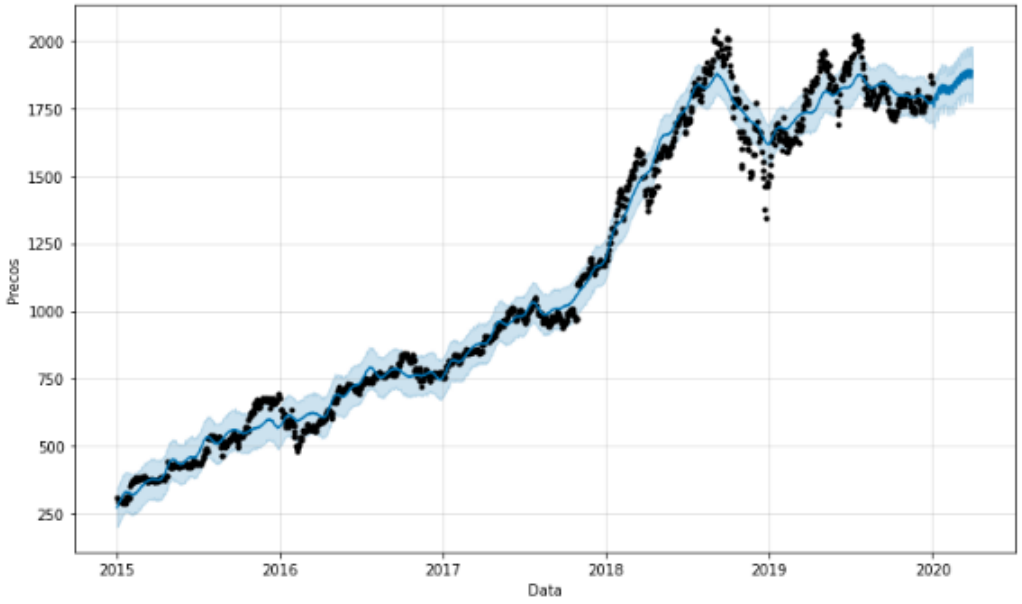
```
previsões_fprof
```

```
ft=previsões_fprof[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
ft.tail(90)
```

```
              ds        yhat       yhat_lower         yhat_upper
1258  2020-01-01    1776.729863    1698.665711    1849.602514
1259  2020-01-02    1779.399434    1703.011665    1850.003584
1260  2020-01-03    1779.583359    1706.383228    1857.431227
1261  2020-01-04    1754.999241    1677.824351    1825.558690
1262  2020-01-05    1758.396760    1684.082254    1832.684652
...          ...            ...            ...         ...
1343  2020-03-26    1893.591038    1804.527920    1980.628035
1344  2020-03-27    1890.768488    1796.444591    1976.134722
1345  2020-03-28    1862.833732    1771.213569    1951.905593
1346  2020-03-29    1862.597923    1777.083714    1946.399848
1347  2020-03-30    1889.091499    1801.890089    1979.032638
```

```
ph=previsões_fprof[['yhat']].tail(90)
```

```
model.plot(previsões_fprof,xlabel='Data',ylabel='Precos')
```



```
mean_absolute_error(dados_Goo_merc,ph)
206.63270006351527
```

```
mean_absolute_percentage_error(dados_Goo_merc,ph)
0.09495547605886821
```

```
previsões_arima['Preços Prophet']=prophet['Preços Prophet']
previsões_arima
```

Date	Preços_Arima	Preços Reais	Preços MGB & MC	Preços Prophet
2020-01-02	1849.064564	1898.010010	1859.710929	1776.729863
2020-01-03	1850.289162	1874.969971	1847.731820	1779.399434
2020-01-06	1851.513761	1902.880005	1868.057923	1779.583359
2020-01-07	1852.738359	1906.859985	1899.587018	1754.999241
2020-01-08	1853.962957	1891.969971	1917.320397	1758.396760
...	...	...	...	...
2020-05-05	1953.155414	2317.800049	2266.416853	1893.591038
2020-05-06	1954.380012	2351.260010	2258.047848	1890.768488
2020-05-07	1955.604610	2367.610107	2284.775258	1862.833732
2020-05-08	1956.829208	2379.610107	2286.207293	1862.597923
2020-05-11	1958.053807	2409.000000	2269.441065	1889.091499