CAPITALIZANDO NAS NÃO LINEARIDADES DAS CARACTERÍSTICAS DAS FIRMAS EM PORTFÓLIOS PARAMÉTRICOS

Gabriel dos Santos Vieira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Carlos Eduardo Pedreira
Marcelo Cunha Medeiros

Rio de Janeiro
Julho 2023

CAPITALIZANDO NAS NÃO LINEARIDADES DAS CARACTERÍSTICAS DAS FIRMAS EM PORTFÓLIOS PARAMÉTRICOS

Gabriel dos Santos Vieira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Carlos Eduardo Pedreira
Marcelo Cunha Medeiros

Aprovada por: Prof. Carlos Eduardo Pedreira
Prof. Marcelo Cunha Medeiros
Prof. Geraldo Bonorino Xexéo
Prof. Carolina Gil Marcelino

.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2023

Agradecimentos

A Deus por me proporcionar esta oportunidade, além de me capacitar e dar saúde para que pudesse superar os desafios impostos ao longo dessa jornada. Aos meus pais, Elinete e Marcelo, que estiveram comigo em toda a trajetória, sempre me dando forças e me incentivando a estudar mais e mais, dizendo-me para ter o conhecimento como se fosse um prato de comida quente e eu estivesse sempre com fome. À minha família, que me ajudou a estudar para que eu pudesse hoje realizar este trabalho. À minha companheira Aimêe, que está sempre ao meu lado nos momentos difíceis da jornada. Aos meus orientadores Marcelo Cunha Medeiros e Carlos Eduardo Pedreira por aceitarem a proposta de trabalho e encararem o desafio comigo. Agradeço a todos que direta ou indiretamente contribuíram para minha educação e para que este trabalho fosse finalizado. Por fim, agradeço ao CNPq pelo fomento à pesquisa e pelo apoio financeiro que foi essencial para minha permanência e conclusão do mestrado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CAPITALIZANDO NAS NÃO LINEARIDADES DAS CARACTERÍSTICAS DAS FIRMAS EM PORTFÓLIOS PARAMÉTRICOS

Gabriel dos Santos Vieira

Julho/2023

Orientadores: Carlos Eduardo Pedreira
Marcelo Cunha Medeiros

Programa: Engenharia de Sistemas e Computação

A otimização de portfólios de ativos financeiros é um problema complexo, que possui uma variedade de métodos que podem ser explorados. Entre esses métodos, é possível destacar o método de otimização utilizando as características financeiras das empresas a fim de construir o portfólio com maior retorno possível. Essa abordagem foi proposta pelos pesquisadores Brandt, Santa-Clara e Valkanov em seu estudo *Parametric Portfolio Policies: Exploiting Characteristics in the Cross-Section of Equity Returns*, no qual utilizaram uma modelagem linear para a composição dessas características. Entretanto, diversos outros estudos, inclusive a pesquisa dos autores previamente mencionados, concluem que essas características na realidade possuem relações não lineares entre si. Por conta dessa hipótese, este trabalho focou em explorar e comparar os portfólios construídos com os mapeamentos linear e não linear, utilizando restrições financeiras para que os resultados possam ser os mais próximos possíveis da realidade. Além disso, utilizou-se redes neurais para realizar o mapeamento das características, visto que estas redes são bons algoritmos para generalizar funções não lineares. Por fim, os resultados obtidos com as redes neurais foram promissores, principalmente se comparados à contraparte linear. As restrições financeiras ainda permitem que o retorno seja expressivo e conclui-se que, para modelos mais complexos de redes neurais, é necessário um conjunto maior de dados para possibilitar seu treinamento.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

CAPITALIZING ON THE NONLINEARITIES OF FIRM CHARACTERISTICS IN PARAMETRIC PORTFOLIOS

Gabriel dos Santos Vieira

July/2023

Advisors: Carlos Eduardo Pedreira
Marcelo Cunha Medeiros

Department: Systems Engineering and Computer Science

The optimization of financial asset portfolios is a complex problem, which has a variety of methods that can be explored. Among these methods, it is possible to highlight the optimization method using the financial characteristics of companies in order to build the portfolio with the highest possible return. This approach was proposed by researchers Brandt, Santa-Clara and Valkanov in their study *"Parametric Portfolio Policies: Exploiting Characteristics in the Cross-Section of Equity Returns"*, where they used linear modeling for the composition of these characteristics. However, in several other studies, including the study of the aforementioned researchers, it is mentioned that these characteristics actually have nonlinear relationships with each other. Because of this hypothesis, this work focused on exploring and comparing portfolios constructed with linear and nonlinear mapping, using financial constraints so that the results can be as close to reality as possible. In addition, neural networks were used to perform this mapping of characteristics since these networks are a good algorithm for generalizing nonlinear functions. Finally, the results achieved with neural networks were promising, even more so compared to the linear counterpart, financial constraints still allow for a significant return, and for more complex neural network models, a larger dataset is necessary for training.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

1 Introdução

Já há vários anos diversos temas na área de finanças vêm sendo explorados e pesquisados através do uso de métodos quantitativos, dentre estes o problema de seleção de portfólios. Um portfólio é um conjunto de ativos negociáveis (*securities*), como opções e ações.

No âmbito de seleção de portfólio, alguns estudos específicos podem ser encontrados na literatura: (i) a seleção de portfólio baseada no modelo de média-variância de Markowitz [1] e (ii) o modelo de análise de portfólio baseado em três fatores, ou características financeiras das empresas, de Fama-French [2]. Este último foi bastante importante para demonstrar que o retorno do portfólio pode ser explicado como uma função das características financeiras dos ativos que o compõem.

Entretanto, com o modelo proposto por Markowitz, utilizar a informação descoberta por Fama-French não é trivial. Brandt, Santa-Clara e Valkanov [3] resolveram esse problema introduzindo um modelo que usa funções de utilidade para o retorno do portfólio com essas características. Assim, eles simplificaram esse problema, permitindo que a otimização do portfólio baseado nas características das empresas pudesse ocorrer.

No modelo apresentado por esses autores, assumiu-se que as dependências entre as características financeiras eram lineares. Entretanto, como os próprios autores pontuam e outros estudos também indicam, as características possuem dependências não lineares umas com as outras. [4][5][6][7][8]

Por conta destes estudos que indicam que as características são não lineares, o objetivo desta dissertação de mestrado tornou-se explorar estas não linearidades e gerar um portfólio otimizado assumindo essa particularidade. Para isso, generalizou-se o modelo criado por Brandt, Santa-Clara e Valkanov e foram mapeadas essas características de maneira não linear.

Ressalta-se, ainda, que existe um estudo similar ao que é proposto nesta dissertação onde se utiliza uma função não linear para a modelagem das características [9], porém seus autores utilizaram *splines* penalizadas [10] como função não linear das características. Nesta dissertação serão utilizadas redes neurais artificiais [11] para a tarefa de modelagem, permitindo, assim, encontrar uma boa função genérica que modele eficientemente as características de maneira não linear.

Foi observado, ao utilizar as redes neurais, que esta é uma técnica que se adequa bem para modelar as características de forma não linear. Os resultados obtidos mostram um um retorno esperado 3 vezes maior que no modelo linear, reafirmando que as características realmente possuem uma dependência não linear entre si. Além disso, utilizamos uma base de dados própria para avaliar a versão linear e não linear dos modelos pois os dados do artigo de Brandt, Santa-Clara e Valkanov eram disponíveis apenas para algumas universidades em parceria com o serviço de dados *Wharton Research Data Service* - WRDS.

Após este resultado levantou-se alguns pontos a serem abordados no futuro, como a obtenção de mais dados para experimentar algoritmos mais complexos de redes neurais assim como redes focadas em séries temporais como o *LSTM* [37]. Além de mais outros dois pontos, como utilizar mais características financeiras em adição às três pontuadas

anteriormente e testar a modelagem criada neste trabalho com dados financeiros dos EUA para critério de comparação com o modelo de Brandt, Santa-Clara e Valkanov.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral desta dissertação é mapear, de modo não linear, características financeiras de ações da bolsa de valores brasileira (B3) generalizando o modelo criado por Brandt, Santa-Clara e Valkanov e utilizando, para esse mapeamento, redes neurais artificiais.

1.1.2 Objetivos específicos

Em adição ao objetivo geral este trabalho também possui os seguintes objetivos específicos:

- adicionar restrições financeiras empíricas após a otimização, como custo de transação e restrição de alavancagem, a fim de obter um resultado mais realista;
- explorar diferentes arquiteturas de redes neurais e compará-las;
- comparar e discutir os resultados do retorno com e sem a restrição de alavancagem;
- explorar maneiras de fazer validação em séries temporais.

1.2 Resumo das contribuições

Listam-se, a seguir, as contribuições deste estudo:

- concepção e utilização de redes neurais para modelagem das não linearidades das características das firmas;
- comparação dos resultados com diferentes arquiteturas de redes neurais (seção 5.2);
- análise e discussão dos resultados com e sem a restrição de alavancagem (seção 5.3);
- implementação do modelo proposto em uma base real de dados utilizando o método de validação apropriado para séries temporais;
- utilização do custo de transação baseado no volume negociado de modo a fazer o retorno ser mais similar ao da realidade.

1.3 Organização do documento

No segundo capítulo são apresentados alguns conceitos financeiros, serão apresentados conceitos que permitem o leitor entender melhor o contexto de finanças que foram utilizados ao longo do trabalho, e além disso, serão apresentados alguns conceitos que permitiram que o resultado tivesse uma boa performance fora da amostra.

No terceiro capítulo há uma descrição da literatura que fundamenta o trabalho, bem como a apresentação de diversos conceitos, a saber: a função do retorno a ser otimizada, as redes neurais e o processo do treinamento utilizado.

No quarto capítulo, introduzem-se informações sobre os dados utilizados neste trabalho assim como informações sobre a otimização utilizando os algoritmos descritos no capítulo 3. Além disso, expõe-se como foi realizada a escolha e utilização do método de validação nesse conjunto de dados.

No quinto capítulo, descrevem-se os resultados alcançados com as abordagens explicitadas nos capítulos anteriores e inicia-se uma discussão sobre os resultados com e sem restrições de alavancagem.

Por fim, no sexto e último capítulo, apresentam-se a conclusão deste trabalho e as considerações sobre as pesquisas a serem exploradas futuramente.

Capítulo 2

2 Conceitos de finanças

Neste capítulo, explica-se a aplicação de diversos conceitos financeiros que serão utilizados no decorrer deste trabalho a fim de familiarizar o leitor com as diversas técnicas utilizadas. Na seção 2.1

2.1 Portfólio de ações

Primeiramente, é necessário explicar o que é um portfólio de ações. Para entender o conceito de um portfólio de ações, podemos começar com a fórmula básica para o retorno de um investimento na Equação 2.1. [39]

$$\text{Retorno} = \frac{(Valor\ final\ do\ investimento - Valor\ inicial\ do\ investimento)}{Valor\ inicial\ do\ investimento}. \quad (2.1)$$

Para um investidor com um único investimento em ações, o retorno pode ser calculado facilmente usando essa fórmula. No entanto, quando um investidor possui um portfólio de ações, o retorno total do portfólio é uma combinação ponderada dos retornos individuais de cada ação no portfólio. A fórmula para o retorno de um portfólio ponderado é dada por 2.2.

$$\text{Retorno do portfólio} = (w_1 . r_1) + (w_2 . r_2) + \ldots + (w_n . r_n) . \quad (2.2)$$

Na qual:

$w_1, w_2, \ldots, w_n$ são os pesos das ações no portfólio (ou seja, a porcentagem do valor total do portfólio que é investido em cada ação).
$r_1, r_2, \ldots, r_n$ são os retornos das ações individuais no portfólio.

Nisto, o retorno do portfólio é a soma ponderada dos retornos individuais de cada ação no portfólio

Entretanto, os pesos deste portfólio podem variar ao longo do tempo à medida que o investidor ajusta a escolha de suas ações, consequentemente alterando o retorno final encontrado. Assim, a equação do retorno do portfólio [3] $r_{p,t+1}$ após um período específico t é dada pela Equação 2.3.

$$r_{p,t+1} = \sum_{i=1}^{N} w_{i,t}\, r_{i,t+1} \quad (2.3)$$

Na Equação 2.3 temos as seguintes variáveis:

N é a quantidade de ações avaliadas

i é a ação atual calculada no somatório
t é o período de tempo antecessor ao que está sendo calculado o retorno
$t + 1$ refere-se a um passo de tempo após t , neste caso, referindo-se ao retorno que foi encontrado após investir na ação i

Entretanto, é necessário levar em consideração todos os períodos de tempo analisados, de modo que é preciso calcular o retorno esperado do portfólio em todos os períodos de tempo ***t***. Logo, a equação do **retorno esperado do portfólio** $\underline{r_p}$ é escrita da seguinte maneira segundo a Equação 2.4. Sendo esse retorno médio $\underline{r_p}$ o **valor esperado** E_t da Equação 2.3.

$$\underline{r_p} = E_t \left[\sum_{i=1}^{N} w_{i,t}\, r_{i,t+1} \right] . \qquad 2.4$$

Após definir a equação média do retorno, podemos verificar que ao longo do tempo os pesos são variáveis. Por conta disso, é possível tentar encontrar os pesos $w_{i,t}$ das ações i em cada tempo t que maximiza o retorno esperado. Assim, um problema de otimização pode ser modelado, dado pela Equação 2.5.

$$\max_{W} \underline{r_p} = E_t \left[\sum_{i=1}^{N} w_{i,t}\, r_{i,t+1} \right] . \qquad 2.5$$

2.2 Sharpe Ratio

Neste trabalho há uma medida que necessita ser mencionada, ela é chamada de *Sharpe Ratio* [40]. Ela é uma medida que calcula a performance de um portfólio levando em consideração a sua volatilidade.

O cálculo do *Sharpe Ratio* é realizado tomando o retorno médio do ativo dividido pelo seu desvio padrão, como na Equação 2.6.

$$sharpe = \frac{E[r]}{\sigma_r} . \qquad 2.6$$

Essa medida visa identificar se os retornos presentes em um ativo são muito voláteis e, consequentemente, mais arriscados. Dessa forma, quanto maior a volatilidade ou, quanto maior o desvio padrão do retorno calculado, menor será o sharpe ratio dado dois retornos médios iguais.

2.3 Pesos de Benchmark

Outro conceito muito importante é a escolha dos pesos de **benchmark** para a geração dos pesos do portfólio. Os pesos de benchmark $\underline{w_{i,t}}$ podem ser escolhidos de formas diferentes. Portanto, citam-se duas possibilidades: a **geração baseada em valor**, que utiliza o valor de mercado (*Market Cap)* das ações, e a **geração igualmente (uniformemente) distribuída** de pesos do portfólio. Na Equação 2.7 é possível ver como é calculado o peso de cada ação para um portfólio igualmente distribuído em cada período de tempo , enquanto na Equação 2.8 é demonstrado os pesos gerados para um portfólio baseado em valor.

$$\underline{w_{i,t}} = \frac{1}{N} \quad . \tag{2.7}$$

$$\underline{w_{i,t}} = \frac{M_{i,t}}{\sum_{i=1}^{N} M_{i,t}} \quad . \tag{2.8}$$

Na qual:

N é a quantidade de ações do portfólio,

$M_{i,t}$ é o valor de mercado da ação i no tempo t,

$\sum_{i=1}^{N} M_{i,t}$ é o somatório do valor de mercado de todas as ações i no tempo t

Apesar de a distribuição igual de pesos ter uma performance melhor em alguns mercados [28], isso não vale para todos os outros. Essa divisão igualitária de pesos para cada ação acaba dando a mesma importância de ações de empresas pequenas e em crescimento às ações de empresas grandes e consolidadas.

Enquanto em alguns mercados essa abordagem gera pouca desvantagem, no mercado brasileiro, por conta da baixa liquidez se comparado a outros mercados, ela não se torna favorável. Isso se deve ao alto risco associado à dificuldade de vender esses ativos no rebalanceamento da carteira. Além disso, os pesos igualmente distribuídos possuem um alto custo de transação se comparados aos baseados em valor. Isso se deve ao alto *turnover* dos pesos igualmente distribuídos, ocasionado por conta do rebalanceamento frequente do portfólio.

Por esses motivos, neste trabalho escolheu-se como benchmark o portfólio baseado em valor, assim dando mais importância às ações de empresas com maior valor de mercado.

2.4 Custos de Transação

Outro conceito que é necessário levar em consideração é o **custo de transação** ao rebalancear o portfólio. Apesar de esse custo poder ser computado de diversas maneiras, vamos exemplificar apenas um neste trabalho.

O custo de transação pode ser computado utilizando o volume de transações de um determinado ativo na bolsa. Dessa maneira, ativos com mais liquidez terão um custo de transação menor, enquanto ativos difíceis de vender terão um custo maior. Essa ideia segue o conceito de que o risco para ações com baixa liquidez é maior que o risco para ações com alta liquidez. A equação que descreve o cálculo do custo de transação de um ativo é dada pela Equação 2.9.

$$TC_{i,t} = \frac{Vol_{i,t} - \min Vol_t}{maxVol_t - \min Vol_t} * (mintax - maxtax) \quad . \qquad 2.9$$

Na Equação 2.9, o custo de transação é normalizado baseado no volume das ações de um período de tempo t a fim de padronizar os valores entre uma taxa mínima e máxima definida pela modelagem do problema.

Os custos de transação são computados e inseridos após encontrar o retorno de cada ação com os pesos otimizados, visto que o custo de transação incide no retorno encontrado por conta da compra/venda do ativo naquele período de tempo.

2.5 Índices de comparação

Foram utilizados dois índices muito importantes na economia brasileira, para fins de comparação, são eles o índice Bovespa e a taxa Selic. A taxa Selic é a taxa básica de juros brasileira, ela é a taxa definida pelo Comitê de política monetária do Banco Central do Brasil. Essa taxa é referenciada como o retorno do investimento mais seguro e menos volátil que é possível obter com os títulos de investimento emitidos pelo governo brasileiro. Sendo assim, todo e qualquer investimento de renda variável precisa ter um retorno médio acima desta taxa, caso contrário, será mais vantajoso ao investidor escolher um ativo que esteja atrelado a esta taxa pois assim ele obtém o que é chamado na literatura de retorno livre de riscos ou *risk-free return.*

Enquanto isso, o índice Bovespa é o índice que representa as maiores empresas do mercado de ações da bolsa brasileira. Este índice é utilizado para calcular a performance do retorno das ações brasileiras de uma forma geral. Para gerar este índice, é criado um portfólio de ações na qual os pesos das ações são gerados utilizando o volume de transações daquelas empresas, assim como outros indicadores, sendo reavaliado a cada 4 meses. Dessa forma, esse índice é utilizado para comparar o quão bem o portfólio de um investidor está se comportando em relação ao retorno obtido das maiores ações do mercado da bolsa brasileira.

2.6 Restrição de Alavancagem

Por fim, a última característica a ser abordada é a **restrição de alavancagem**. A alavancagem permite que o portfólio compre ativos além da receita original do investimento pegando empréstimos de outros investidores. Na prática, a alavancagem nada mais é do que pesos negativos nas ações do portfólio, com os quais o portfólio consegue ter mais de 100% da receita original alocada em alguns ativos. Além disso, os pesos negativos também são chamados de *short selling,* onde é esperado que o preço da ação diminua ao longo do tempo.

Entretanto, quando não há nenhuma restrição na quantidade de alavancagem do portfólio, é possível que o modelo de otimização encontre portfólios com altíssima alavancagem. Dito isso, o empréstimo do valor do investimento para tal portfólio existir se torna inviável. Ademais, mesmo que o valor desse empréstimo possa vir a existir, a volatilidade do portfólio criado faz com que o investimento seja muito arriscado, podendo levar à falência o investidor do portfólio.

Por esses motivos, é necessária a utilização de uma restrição na alavancagem do portfólio. Ela pode ser computada pela soma dos pesos negativos em cada instante de tempo. Tal restrição é aplicada em cada instante de tempo, para que não haja períodos com alta alavancagem e baixa alavancagem, que na média se anulariam. A equação da restrição pode ser vista na Equação 2.10.3. Nesta Equação, τ^+ é a taxa de alavancagem máxima dos pesos positivos e τ^- é a taxa de alavancagem máxima dos pesos negativos.

$$w^+{}_{i,t} = \frac{W^+{}_{i,t}}{\sum_{i=1}^{N} W^+{}_{i,t}} \tau^+ \mid \tau^+ > 1 \ \wedge \ \tau^+ + \tau^- = 1 \quad , \qquad 2.10.1$$

$$w^-{}_{i,t} = \frac{W^-{}_{i,t}}{\sum_{i=1}^{N} W^-{}_{i,t}} \tau^- \mid \tau^- < 0 \ \wedge \ \tau^+ + \tau^- = 1 \quad , \qquad 2.10.2$$

$$w_{i,t} = (w_{i,t} < 0) * w^-{}_{i,t} + (w_{i,t} > 0) * w^+{}_{i,t} \quad . \qquad 2.10.3$$

Essa restrição é realizada empiricamente após a otimização dos pesos do portfólio. Ela é realizada desta forma pois mesmo sendo realizada após a otimização ela ainda gera bons resultados e caso essas restrições fossem incorporadas a otimização, ela se tornaria extremamente complexa.

Capítulo 3

3 Revisão bibliográfica

Neste capítulo, apresenta-se uma revisão dos conceitos teóricos utilizados no desenvolvimento deste trabalho. Na seção 3.1 são mencionados os trabalhos relacionados a este, relatando as abordagens que ambos utilizaram. Já na seção 3.2 é explicado o que é um portfólio de ações, assim como o que é e para o que serve a otimização do mesmo. A seção 3.3 explica a função de otimização a ser resolvida pelo problema de otimização de portfólio com características financeiras. Na seção 3.4, fala-se brevemente sobre o algoritmo de otimização escolhido a fim de ser comparado aos modelos não lineares. Na seção 3.5, as redes neurais escolhidas como modelos não lineares são explicadas, assim como todo o processo que a permeia. Por fim, na última seção deste capítulo, explica-se a aplicação de alguns conceitos financeiros no problema deste trabalho.

3.1 Trabalhos Relacionados

O estudo pioneiro em relação a parametrização de informações financeiras na otimização de portfólios e o qual foi embasado este trabalho é o estudo de Brandt, Santa-Clara e Valkanov [3]. Neste artigo, foi explicada a modelagem escolhida para parametrizar as características financeiras na equação do portfólio. Os autores assumiram uma dependência linear entre as características em sua modelagem.

Entretanto, eles também mencionam em seu próprio estudo que as características financeiras possuem uma dependência não linear segundo outros pesquisadores e que essa informação deveria ser explorada em trabalhos futuros.

Com isso, o trabalho de Caldeira, Santos e Torrent [9] resolveu explorar estas não linearidades das características, modelando-as com uma função não linear chamada *spline* penalizada. Com este trabalho eles confirmaram que as não linearidades impactam positivamente no resultado da otimização do portfólio.

Por fim, esse trabalho propõe encontrar uma função não linear genérica que modela bem as características financeiras realizando a otimização do portfólio. Para essa finalidade serão utilizadas as redes neurais.

3.2 Função de otimização

Neste problema de otimização de portfólios, o objetivo é encontrar os melhores pesos que geram os maiores retornos. Como mencionado na introdução, para solucionar este problema, adotou-se uma simplificação na qual é utilizada uma função utilidade $\boldsymbol{u}$, que será explicada em breve, auxiliando na otimização do retorno do portfólio. Após a adição dessa função, a equação 2.4 passou a ser representada pela Equação 3.1.

$$E_t\left[u\left(r_{p,t+1}\right)\right] = E_t\left[u\left(\sum_{i=1}^{N} w_{i,t}\, r_{i,t+1}\right)\right] . \quad 3.1$$

Por fim, como esse problema visa a encontrar os melhores pesos com o maior retorno possível, a equação final – considerando a **otimização** – é dada pela Equação 3.2.

$$\max_{W} E_t\left[u\left(r_{p,t+1}\right)\right] = E_t\left[u\left(\sum_{i=1}^{N} w_{i,t}\, r_{i,t+1}\right)\right] . \quad 3.2$$

Nela, W é uma matriz de pesos de dimensão $i\ x\ t$ na qual i corresponde às ações, e t, aos períodos de tempo analisados. Assim, maximizando o retorno esperado a partir da escolha dos pesos W.

A Equação 3.2 é aquela necessária para a realização da otimização neste trabalho. Entretanto, faz-se necessário exemplificar o objetivo da função utilidade ***u*** em tal contexto, assim como o que ela é. Por meio dela, sendo **N** a quantidade de ações, em vez de precisar modelar $(N^2+N)/2$ segundos momentos dos retornos (como no modelo tradicional de Markowitz), é necessário modelar apenas **N** momentos, simplificando o problema [3]. Para o presente trabalho, assumiu-se que o investidor terá uma preferência de aversão ao risco, de modo que foi usada uma função utilidade chamada Constant Relative Risk Aversion (CRRA) [3], seguindo a mesma função utilidade que o artigo este na qual este trabalho foi baseado, esta função é dada pela Equação 3.3.

$$u\left(r_{p,t+1}\right) = \frac{\left(1+r_{p,t+1}\right)^{1-\gamma}}{1-\gamma} . \quad 3.3$$

Em que, γ é a constante que determina o grau de aversão ao risco do investidor. Quanto maior o valor de γ, maior é a aversão ao risco.

Além disso, retomando a equação 3.2, os pesos $w_{i,j}$ das ações em cada período de tempo precisam ser baseados nas características das ações. Portanto, $w_{i,j}$ pode ser escrito como demonstrado na equação 3.4.

$$w_{i,t} = f(x_{i,t};\theta) . \quad 3.4$$

Nela, $x_{i,t}$ são as características da ação ***i*** no tempo ***t***, e θ corresponde aos pesos para cada uma das características. No trabalho original de Brandt, Santa-Clara e Valkanov [3], os autores definem uma modelagem linear para os pesos $w_{i,j}$ das ações, dada pela Equação 3.5.

$$w_{i,t} = \underline{w}_{i,t} + \frac{1}{N} \theta^T \hat{x}_{i,t} \quad . \tag{3.5}$$

Na qual, $\underline{w}_{i,t}$ representa o **peso de benchmark**, que pode se basear em uma política ***value weighted,*** em que, quanto maior o valor de mercado, maior o peso de cada ação; N é a **quantidade de ações** no tempo ***t***; θ^T indica os **pesos das características**; e $\hat{x}_{i,t}$ corresponde às **características normalizadas** no *cross-section,* ou seja, em cada período de tempo, para obter média zero e desvio padrão unitário.

Com isso, a Equação 3.2 pode ser reescrita em função das equações 3.4 e 3.5, pelo modelo de otimização disposto em 3.6.3.

$$\max_{W} E_t \left[u\left(r_{p,t+1}\right) \right] = E_t \left[u\left(\sum_{i=1}^{N} w_{i,t}\, r_{i,t+1} \right) \right] \quad , \tag{3.6.1}$$

$$\max_{\theta} E_t \left[u\left(r_{p,t+1}\right) \right] = E_t \left[u\left(\sum_{i=1}^{N} f(x_{i,t};\theta)\, r_{i,t+1} \right) \right] \quad , \tag{3.6.2}$$

$$\max_{\theta} E_t \left[u\left(r_{p,t+1}\right) \right] = E_t \left[u\left(\sum_{i=1}^{N} \left(\underline{w}_{i,t} + \frac{1}{N} \theta^T \hat{x}_{i,t} \right) r_{i,t+1} \right) \right] \quad . \tag{3.6.3}$$

Sendo assim, o objetivo é encontrar θ que maximize o retorno esperado a partir da função utilidade. Por fim, relacionado à modelagem dos pesos das ações, está o tema deste trabalho. Brandt, Santa-Clara e Valkanov [3] mencionam que a relação linear entre o peso das características θ e seus valores $\hat{x}_{i,t}$ é, de fato, não linear e utilizaram apenas uma simplificação para o propósito de apresentar sua abordagem. Essa afirmação sobre a não linearidade é corroborada por uma série de autores [4][5][6][7][8] e, com isso, é possível definir que a função de peso das ações $w_{i,j}$ é dada genericamente pela equação 3.7.

$$w_{i,t} = \underline{w}_{i,t} + g\left(\hat{x}_{i,t};\theta\right) \quad . \tag{3.7}$$

Nela, $g(\cdot;\cdot)$ é uma função não linear genérica. Como já estabelecido, o trabalho objetiva encontrar essa função não linear genérica ou uma boa aproximação dela com o intuito de produzir, a partir da função utilidade, bons pesos θ das características que maximizem o retorno esperado.

3.3 Algoritmo de otimização utilizado

O algoritmo utilizado para a otimização da versão linear deste trabalho se chama *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) [11]. Trata-se de uma aproximação do método de Newton [12] pela qual é possível utilizar a Hessiana [13] para funções que são duplamente diferenciáveis, usufruindo do cálculo da aproximação dessa Hessiana. Por conta disso, aproveitou-se o fato de a função de otimização ser duplamente diferenciada para usufruir desse algoritmo. A equação para a atualização de iteração pelo método de Newton em uma dimensão utilizando a inversa da Hessiana é dada pela Equação 3.8.

$$x_{k+1} = x_k - H(x_k)^{-1} \Delta f(x_k) . \quad 3.8$$

À priori a otimização foi realizada utilizando o algoritmo do gradiente descendente desenvolvido do zero, entretanto, observou-se que o algoritmo desenvolvido não tinha uma performance tão boa, isso gerou a busca por algoritmos com uma performance melhor para poder realizar o experimento. Sendo assim, observou-se que a utilização de algoritmos que utilizam a segunda derivada da função de otimização tem uma performance melhor e o BFGS possuía uma implementação fechada disponível em uma biblioteca de Python bem conhecida como o Scipy.

3.4 Redes Neurais

As Redes Neurais Artificiais (RNAs) são uma classe de modelos de aprendizado de máquina inspirados na estrutura e função das redes neurais biológicas no cérebro humano. Elas consistem em nós interconectados, ou "neurônios", organizados em camadas: uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída. Cada conexão entre neurônios está associada a um peso, que é ajustado durante o treinamento para otimizar o desempenho da rede em uma tarefa específica. As RNAs são usadas para tarefas como reconhecimento de padrões, classificação, regressão e tarefas mais complexas como reconhecimento de imagens e fala. [38]

Esse algoritmo é utilizado para encontrar uma aproximação de uma função a partir de um conjunto de dados. Para este trabalho, foi decidido que essa função deve gerar os pesos das características que maximizam a função de otimização destacada na seção 3.3. As redes neurais são ferramentas capazes de realizar aprendizado de máquina, pois, além de modelar funções lineares, tem a capacidade de modelar funções não lineares a depender de como é criada.

Adicionalmente, uma rede neural é uma função matemática *F* que é definida por uma composição de funções multivariadas: f_1, f_2, … , f_k, *g.* Esta função *F* pode ser definida pela Equação 3.13.

$$
\begin{aligned}
F: R^I \rightarrow R^O \ , \\
F = g \circ f_k \circ f_{k-1} \circ \ldots \circ f_2 \circ f_1(x) \ , \\
f_i : R^{i-1} \rightarrow R^i \ , \\
f_i(x) = a(w_i x + b_i) \ .
\end{aligned}
\qquad 3.13
$$

Na Equação 3.13, os valores de I e O nos números reais R representam a dimensão dos dados de entrada e saída da função F respectivamente. Assim, nas funções intermediárias f_i , a dimensão dos dados de entrada é definido pela função anterior $i-1$ e a de saída é dada pela função atual i, sendo definidos pela arquitetura escolhida. Cada função $f_i(x)$ é uma combinação linear dos coeficientes w_i com os valores de entrada x mais um viés b_i . Após esta combinação linear, é aplicada uma função de ativação a não linear que será comum para todas as funções em i, sendo esta a responsável pela transformação dos dados de entrada no espaço.

Por último, há a função dos dados de saída g. Ela é a responsável pela transformação final dos dados para o tipo de problema a ser resolvido. Esta função pode definir o problema como uma classificação de múltiplas classes, como no caso da função *softmax* exemplificada na equação 3.14, ou também pode ser definida como uma classificação binária, **otimização**, ou regressão.

$$
g(z) = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}}, para\ j = 1, \ldots, N \ . \qquad 3.14
$$

Após definir a rede e seus componentes é necessário metrificar o quão bom são os resultados encontrados. Para isso é utilizada a função custo, ela tem o objetivo de calcular o erro entre o resultado encontrado e o valor de referência, no caso de problemas de classificação, ou no caso de problemas de otimização, calcular o valor da função a ser otimizada.

Entretanto, para garantir que a generalização dessas funções será adequada, é preciso levar em consideração diversos passos. Dentre destacam-se três: (i) a escolha do **algoritmo de otimização** da rede, que minimiza a função custo e geralmente define a precisão e rapidez com que o treinamento da rede ocorre; (ii) a **validação**, que garante que os resultados estão de acordo dentro e fora da amostra; e (iii) a **regularização**, que permite regular a importância dos dados dentro da amostra para obter uma generalização fora da amostra.

Por fim, o algoritmo que possibilita o treinamento e a generalização dessa rede neural *feedforward* é chamado *backpropagation* [14], um algoritmo de programação dinâmica

utilizado para ajustar os pesos das redes neurais *feedforward* baseando-se nos erros encontrados durante o treinamento.

Esses erros são dados pela função custo e se caracterizam, normalmente, pela diferença entre o valor da saída da rede neural e seu valor de referência dentro da amostra. Para diminuí-los, usualmente se calcula o gradiente da função custo e, a partir dele, atualiza-se o valor dos pesos da rede na direção contrária.

No caso deste trabalho, não há valor de referência a ser utilizado pois ele é o que se está tentando encontrar. Sendo assim, ao invés de utilizar o erro entre o valor de saída e de referência, será utilizada a função de utilidade da equação de otimização mencionada no início deste capítulo como o valor de saída, e a partir deste valor será realizada a otimização/atualização dos pesos da rede neural utilizando o gradiente desta equação de otimização.

3.4.1 Algoritmo de otimização da rede

Os algoritmos de otimização das redes neurais são responsáveis por minimizar a função de custo do problema a ser modelado. Existem diversos algoritmos de otimização utilizados na literatura especializada, como o gradiente descendente estocástico (conhecido como **SGD**), o **RMSprop** e o **Adam**.

O SGD [15] é um método iterativo que otimiza uma função objetivo diferenciável. Esse algoritmo pode ser considerado uma aproximação estocástica do gradiente descendente, pois substitui o gradiente (calculado com o conjunto de dados inteiro) por uma estimação dele (calculada a partir de um subconjunto de dados selecionado de maneira aleatória).

Já o RMSprop [16] foi introduzido nas aulas *on-line* de redes neurais lecionadas por Geoffrey Hinton, da Universidade de Toronto. Hinton não publicou o RMSprop em um artigo formal, e, ainda assim, tornou-se um dos métodos mais populares de otimização para aprendizado profundo. O RMSprop tenta resolver o problema da alta variação de magnitude dos gradientes.

Por fim, o Adam [17] pode ser visto como uma combinação do RMSprop e o SGD com momentum [18]. Assim como o RMSprop, ele utiliza o gradiente ao quadrado para escalar a taxa de aprendizado e também tira vantagem do momentum por utilizar a média móvel do gradiente em vez de apenas o gradiente, como ocorre com o SGD com momentum. O Adam é um algoritmo amplamente utilizado na literatura e é comumente adotado em diversas aplicações do mundo real. Por isso, ele se mostra um ótimo algoritmo a ser utilizado em razão dos inúmeros estudos desenvolvidos em cima dele. Dessa maneira, este foi o algoritmo escolhido neste trabalho para ser utilizado como o otimizador da rede neural artificial.

3.4.2 Validação

Validação no escopo de redes neurais, como o nome sugere, é o processo que valida o modelo da rede escolhido, garantindo que o resultado será generalizado. Assim, o resultado esperado fora da amostra será próximo ao encontrado. Para isso, utilizam-se métodos estatísticos que visam a testar o modelo fazendo uso dos dados de diferentes maneiras. Uma dessas maneiras é a técnica chamada ***validação cruzada*** [19].

O *cross-validation* é um método que usa diferentes partes do conjunto de dados para teste e treinamento de maneira iterativa a fim de garantir que o modelo utilizado generalize bem esse conjunto de dados. Entre as maneiras de utilizar o *cross-validation*, é possível citar o ***leave-one-out*** e o ***k-fold***.

O *leave-one-out* consiste em uma divisão no conjunto de dados em que apenas um exemplo é utilizado como teste, e todo o restante dos dados é utilizado como treinamento. Esse tipo de validação é muito utilizado em especial quando há poucos dados, pois permite que o modelo seja treinado com a maior quantidade de dados possível enquanto ainda possui a capacidade de testar sua generalização fora da amostra.

O *k-fold*, por sua vez, consiste na divisão do conjunto de dados em k partes iguais, sendo que são utilizadas k-1 partes para treinamento e uma parte para teste. Essa parte de teste varia a cada iteração, assim como as k-1 partes de treinamento, como demonstrado na figura 3.1. Com esse tipo de divisão, é possível garantir de maneira mais acurada que a generalização ocorrerá. Entretanto, só é válido recorrer a ela quando há uma maior quantidade de dados.

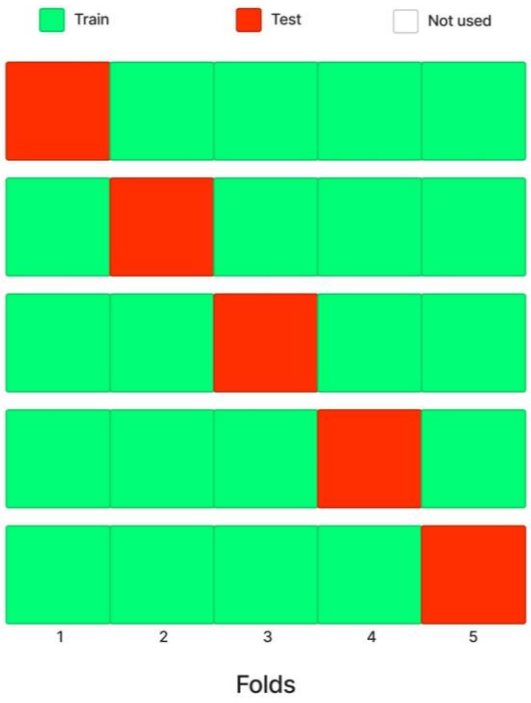

Figura 3.1 Exemplo de K-*fold* com 5 *folds* em suas 5 iterações.

Quando comparado ao *leave-one-out*, o *k-fold* apresentou uma performance pior quando há poucos dados, pois, como há menos dados para treinamento, a validação do resultado dentro da amostra mostrou-se inferior à do *leave-one-out*. Porém, quando há mais dados, é preferível escolher o *k-fold*, já que pode garantir que o resultado obtido se estenderá a dados que não foram vistos anteriormente.

Existem também formas diferentes de divisão de dados dependendo do tipo de dado com que se está trabalhando. Por exemplo, em séries temporais, é comum dividir os dados utilizando uma janela deslizante, ou ***sliding window***. Por conta da dependência temporal nessas séries, dividir os dados utilizando o *k-fold* ou o *leave-one-out* pode fazer com que tal dependência se perca. Logo, muitas vezes é preferível utilizar uma *sliding window*, pois consegue manter a ordem dos dados no tempo.

Uma *sliding window* nada mais é que uma divisão do conjunto de dados em n partes iguais. Diferentemente do *k-fold*, selecionam-se a primeira e a segunda parte, respectivamente, como treinamento e teste. Após isso, escolhem-se iterativamente a segunda e a terceira parte, repetindo esse processo até o fim das iterações, quando serão selecionadas a n-1 e n-ésima parte como treinamento e teste, como pode ser visto na figura 3.2.

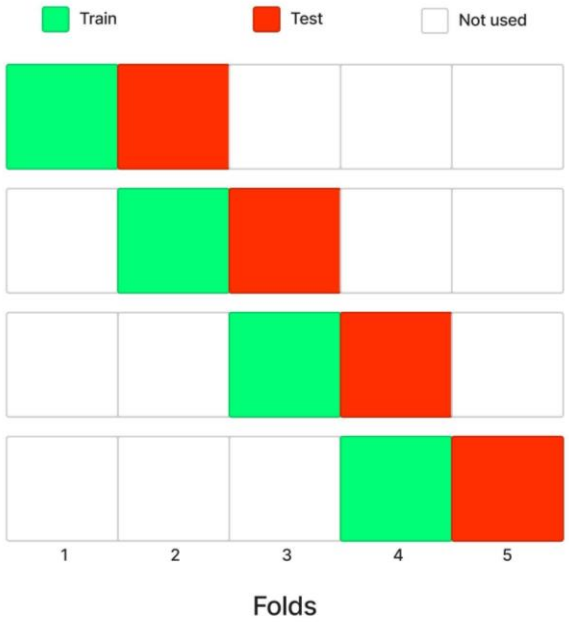

Figura 3.2 Exemplo de *sliding window* com 5 *folds* em suas 4 iterações.

Além da *sliding window*, existe outro método: a janela crescente, ou ***growing window (Preq-grow)***. Nele, ocorre uma divisão dos dados similar à da *sliding window.* Entretanto, a cada iteração, o conjunto de treinamento adiciona uma parte dessas divisões enquanto o teste se mantém apenas com uma parte ao longo das iterações. É possível ver com mais detalhes a divisão *growing window* na Figura 3.3.

Figura 3.3 Exemplo de *growing window* com 5 folds em suas 4 iterações.

Por fim, para basear o melhor método de validação a ser utilizado no contexto de séries temporais, Cerqueira, Torgo e Mozetic [20] estudaram e testaram diversos métodos de validação e construíram uma árvore de decisão baseada nas estatísticas da estrutura das séries temporais que permite calcular a melhor forma de validação baseada nos dados utilizados.

3.4.3 Regularização

Enquanto a validação é o processo em que se calcula o desempenho do modelo corretamente, a regularização ajusta o modelo para que ele tenha uma boa performance fora da amostra. A regularização, resumidamente, seria uma espécie de relaxamento do treinamento para que o modelo não sofra o ***overfitting*** [21]. Para isso, na regularização, utilizam-se diversas técnicas que têm abordagens diferentes, mas que possuem a mesma finalidade. Entre essas técnicas, citam-se o ***Dropout*** [22], o ***Early Stopping*** [23], o ***Weight Decay*** [24], entre outras.

Em maiores detalhes, uma regularização seria o processo que deixa de utilizar algumas informações presentes nos dados e que atrapalham a generalização do modelo. Essas informações, ou *features*, podem adicionar um ruído ao treinamento do modelo, gerando uma hiperespecialização nesses dados. Com isso, caso os dados sejam utilizados sem regularização, será observado que o modelo gerado não apresentará boa performance em relação a dados nunca vistos antes.

Embora essa seja a definição mais simples de regularização, algumas abordagens citadas previamente não a seguem, como no caso do *Early Stopping.* Nele, a ideia principal é calcular o erro gerado pelo modelo utilizando os dados que foram divididos no conjunto de treinamento e de validação. O ideal é que, enquanto o treinamento ocorre, tanto o erro do conjunto de validação quanto o de treinamento diminuam ao longo do tempo. Quando o erro presente na amostra de validação começa a divergir do erro de treinamento, indica que o modelo está ficando enviesado em relação aos dados de treinamento. Portanto, no *Early Stopping* o processo de treinamento é interrompido assim que esse comportamento é

identificado ou poucos instantes depois, impedindo o enviesamento e, assim, contribuindo para a generalização.

Já o *Dropout*, diferente do *Early Stopping,* é apenas utilizado em redes neurais. Ele consiste em uma ideia simples que é a de remover aleatoriamente neurônios já treinados de uma rede neural. Utilizando essa ideia de remoção, é possível diminuir o *overfitting* e melhorar a performance da rede. No artigo do *Dropout* é explicado com mais detalhes o porquê dessa remoção de neurônios apresentar bom funcionamento na melhora da generalização no contexto de redes neurais.

Por fim, o *Weight Decay*, como o nome sugere, introduz um decaimento nos pesos calculados. Esse decaimento ocorre exponencialmente a cada iteração de atualização dos pesos. A equação mais comum para o *Weight Decay* é dada em 3.15:

$$w_{t+1} = (1 - \lambda)w_t - \alpha \nabla f_t(w_t) \quad . \tag{3.15}$$

λ é a taxa de decaimento dos pesos, e α é a taxa de aprendizado, que é multiplicada pelo gradiente do *batch* na época *t* . Esse decaimento permite que a complexidade do modelo diminua e também diminui a chance de *overfitting* do modelo. Além disso, as técnicas de regularização utilizadas neste trabalho foram o Early Stopping e o Weight Decay, pois elas contribuíram bem para a generalização do problema resolvido, enquanto o Dropout provavelmente ocasionaria um underfitting.

3.4.4 Otimização de hiperparâmetros

Os hiperparâmetros são parâmetros inerentes ao processo de treinamento e geralmente são definidos a priori. Enquanto o processo de treinamento é responsável por encontrar parâmetros que otimizam a função custo do modelo utilizado, os hiperparâmetros são escolhidos previamente para que esse treinamento ocorra. Alguns exemplos de hiperparâmetros são: a quantidade de iterações de treinamento, a taxa de aprendizado, a tolerância de erro de alguns algoritmos numéricos, entre outros.

Entretanto, também é possível encontrar os melhores hiperparâmetros otimizando essa escolha por meio de um processo automatizado, que, por sua vez, é muito importante [25], visto que, ao otimizar os hiperparâmetros de um modelo, também melhorará os resultados que o modelo será capaz de gerar. Existem diversas formas de automatizar esse processo [26], como o uso de técnicas tais quais o ***Grid Search***, o ***Random Search*** e a **Otimização Bayesiana**.

Para utilizar o *Grid Search*, é necessário definir um conjunto de valores para os hiperparâmetros que serão testados. Para cada hiperparâmetro é definido um conjunto, e todos os conjuntos têm o mesmo tamanho. Além disso, utiliza-se uma porção dos dados,

(comumente o conjunto de validação) para realizar a avaliação dos hiperparâmetros e escolher os que produzem o melhor resultado.

Por exemplo, no caso do otimizador Adam [17], temos os hiperparâmetros α, b1, b2. α é a taxa de aprendizado, b1 é o decaimento exponencial da estimativa do primeiro momento e b2 é o decaimento exponencial da estimativa do segundo momento. Assim, a definição de conjuntos de três valores para cada um de seus hiperparâmetros pode ser vista na Equação 3.16.

$$\begin{aligned} \underline{\alpha} &= (\alpha_1, \alpha_2, \alpha_3)\ , \\ \underline{b1} &= (b1_1, b1_2, b1_3)\ , \\ \underline{b2} &= (b2_1, b2_2, b2_3)\ . \end{aligned} \tag{3.16}$$

Na Equação 3.16, temos os conjuntos $\underline{\alpha}$, $\underline{b1}$, $\underline{b2}$ com três possíveis valores para cada hiperparâmetro. No *Grid Search*, serão testadas as triplas $(\alpha_1, b1_1, b2_1)$, $(\alpha_2, b1_2, b2_2)$ e $(\alpha_3, b1_3, b2_3)$com o modelo escolhido nos dados de validação. Por fim, ao encontrar a melhor tripla de hiperparâmetros baseada na performance no conjunto de validação, eles serão utilizados ao longo de todo o treinamento e teste do modelo.

Já a técnica *Random Search* tem um conceito um pouco diferente. Em vez de escolher as triplas baseadas na sua posição, no *Random Search* é utilizado um limite superior e inferior para cada hiperparâmetro. Dentro dessa faixa de possíveis valores, adota-se uma amostragem aleatória do valor de cada hiperparâmetro. Por exemplo, seguindo o exemplo apresentado para o *Grid Search* e definindo a posição 1 como o limite inferior e a posição 3 como o limite superior para cada hiperparâmetro, é possível obter os seguintes conjuntos dados pela Equação 3.17.

$$\begin{aligned} \hat{\alpha} &= random(\alpha_1\ , \alpha_3)\ , \\ \widehat{b1} &= random(b1_1\ , b1_3)\ , \\ \widehat{b2} &= random(b2_1\ , b2_3)\ , \end{aligned} \tag{3.17}$$

$$\begin{aligned} \underline{\alpha} &= (\hat{\alpha}, \hat{\alpha}, \hat{\alpha}\ )\ , \\ \underline{b1} &= \left(\widehat{b1}, \widehat{b1}, \widehat{b1}\right)\ , \\ \underline{b2} &= \left(\widehat{b2}, \widehat{b2}, \widehat{b2}\right)\ . \end{aligned}$$

Na equação 3.17, os hiperparâmetros do exemplo anterior, mas agora também existem os valores $\hat{\alpha}$, $\widehat{b1}$ *e* $\widehat{b2}$, que são aleatórios e são amostrados dentro do limite superior e inferior. Por fim, o conjunto de valores dos hiperparâmetros a ser testado será dado, nesse caso, por três amostragens aleatórias independentes para cada um deles. Esses valores serão avaliados e utilizados da mesma maneira mencionada no exemplo do *Grid Search.*

Por último, há a otimização de hiperparâmetros utilizando a otimização bayesiana [27]. Diferentemente do *Grid Search* e do *Random Search*, a otimização bayesiana utiliza a avaliação de cada teste dos hiperparâmetros para escolher o próximo conjunto de valores a serem testados. Essa próxima escolha feita a partir da informação da avaliação permite que a otimização encontre rapidamente bons valores para os hiperparâmetros.

Essa técnica cria um modelo probabilístico substituto para os hiperparâmetros chamado *Surrogate Model/Function*, que visa estimar a função objetivo da otimização dos hiperparâmetros. Essa função, geralmente, é muito complexa para ter seu modelo probabilístico real criado, de modo que muitas vezes é o resultado de um modelo de aprendizado de máquina usando esses hiperparâmetros. Por esse motivo, gera-se essa estimativa do modelo para que seja possível predizer o resultado da escolha dos hiperparâmetros nesse espaço.

Além desse modelo, é necessário eleger uma política de escolha dos próximos hiperparâmetros que serão testados. Essa política recebe o nome de *Acquisition Function* e há diversas opções. Dependendo da *Acquisition function* escolhida, a *surrogate function* será otimizada de maneira diferente, escolhendo, assim, o melhor valor para os hiperparâmetros daquela *surrogate function* naquele momento.

Sendo assim, as iterações desse algoritmo são dadas por estes passos:

1. criar a *Surrogate Function;*
2. utilizar a *Acquisition Function* e escolher os hiperparâmetros;
3. avaliar os hiperparâmetros na função objetivo real;
4. atualizar a *Surrogate Function* com os valores obtidos;
5. repetir o processo dos passos 2 ao 4 até um número máximo de iterações previamente definido.

Dessa maneira é possível obter uma boa estimativa dos melhores hiperparâmetros dado o modelo a ser utilizado. Para mais detalhes sobre otimização, é possível checar o artigo [27], que explica esse método mais profundamente. Neste trabalho utilizou-se a otimização bayesiana para encontrar os melhores hiperparâmetros, pois além de poder encontrar bons resultados com ela, havia recurso computacional disponível para utilizá-la.

Capítulo 4

4 Metodologia

Este capítulo abordará a maneira como o experimento foi realizado. Na seção 4.1, apresenta-se o conjunto de dados utilizado, assim como suas particularidades e seu pré-processamento. Logo após, na seção 4.2, descrevem-se as ferramentas utilizadas, tais quais a especificação do computador e a linguagem de programação e as bibliotecas utilizadas. Por fim, a seção 4.3 expõe o treinamento dos algoritmos escolhidos para otimizar a função objetivo mencionada no capítulo anterior.

4.1 Dados utilizados

Neste trabalho, foram utilizados dados de 250 ações que fazem parte do mercado de ações brasileiro (B3). Esses dados iniciais foram cedidos pelo professor coorientador deste trabalho, Marcelo Cunha Medeiros. O conjunto de dados final foi feito processando e selecionando os dados originais a fim de criar as informações financeiras de cada um dos ativos, como seu retorno, valor de mercado, valor da empresa, dívida bruta e volume de transações do período de janeiro de 1995 a dezembro de 2011. Além disso, está sendo considerado o valor mensal de cada uma dessas informações. Com essas informações, são geradas três características financeiras que serão utilizadas na otimização (como na seção 3.3), sendo elas o *lagged return*, o *market cap* e o *book-to-market ratio*. No fim, os dados foram trabalhados no formato de uma matriz de dimensão 250x203x3 onde 203 são o número de meses trabalhados, 3 é o número de características financeiras e 250 são o número de ações.

O *lagged return* é a expectativa de que o retorno que ocorreu no mês anterior possa acontecer no mês seguinte. Com isso, para calcular essa característica, é necessário apenas substituir a posição do retorno que foi obtido no tempo t-1 para o tempo t. O *market cap* é o valor de mercado do ativo em questão. No caso das ações, trata-se do número de ações existentes no mercado multiplicado pelo preço da ação. Com essa característica, é possível definir o tamanho da empresa baseado no valor de mercado com o qual se está lidando.

Por fim, o *book-to-market ratio* é o valor dos ativos da empresa subtraído pela dívida que ela possui; esse resultado, então, é dividido pelo seu valor de mercado. A ideia dessa característica é verificar se o mercado está precificando o ativo como se valesse mais do que a empresa pode oferecer, caso decida vender todas as suas posses e liquidar as suas dívidas. Essas três características são utilizadas no artigo original de Brandt, Santa-Clara e Valkanov [3], mas os autores originalmente usam dados da bolsa americana (CRSP – Compustat merged data set). Aqui, como mencionado, serão usados dados da bolsa brasileira, pois os dados do artigo de Brandt, Santa-Clara e Valkanov eram disponíveis apenas para algumas universidades em parceria com o serviço de dados *Wharton Research Data Service* - WRDS.

4.1.1 Pré-processamento

Os dados do trabalho também passaram por um pré-processamento antes de serem utilizados para a otimização. A primeira alteração realizada nos dados diz respeito aos **dados faltantes**. Em cada característica, se algum ativo tiver dados faltantes, o valor faltante será substituído pelo valor do mês seguinte. Na prática, todo valor faltante no período *t-1* será substituído por algum valor de *t*, se este for um valor válido. Essa iteração começa no período *t* e decresce de um em um até chegar a *t=2*, sendo *t=1* o período inicial. Se o valor da característica do ativo tiver apenas valores faltantes, então será substituído por 0 para todos os períodos de tempo.

Além dessa primeira alteração, é necessário **aplicar o log** em alguns dos valores das características para não trabalhar com dados de magnitudes muito distintas. Por exemplo, enquanto o retorno geralmente chega a valores da ordem de 10^3 a 10^4, o *market cap* pode ter valores da ordem de 10^9. Essa diferença, quando inserida em algoritmos de otimização, acaba por inviabilizar sua utilização, pois o algoritmo irá pender para os valores de maior magnitude, cujos valores não necessariamente acrescentam informação à solução final, divergindo do resultado ótimo esperado.

As características que foram transformadas pela utilização do log foram o *market cap* e o *book-to-market ratio*. A necessidade do primeiro foi evidenciada no parágrafo anterior, enquanto o segundo é utilizado por conta da sua construção no artigo de Brandt, Santa-Clara e Valkanov. Por último, há o **pré-processamento dos dados no *cross-section*.** Esse pré-processamento subtrai o valor de cada ação pela média de cada característica no *cross-section*.

Para exemplificar melhor a transformação no *cross-section*, é possível pensar que se trata de uma transformação em cada período de tempo. Isso permite que os dados de períodos distintos não se contaminem, impedindo ruídos ocasionados por conta do processamento. Esses ruídos, caso não fossem levados em consideração, poderiam gerar resultados falsos.

Seguindo o exemplo, como há três características, então serão calculadas três médias. Os cálculos tomam os valores de todas as ações em um período de tempo *t* para cada uma das características. Depois, cada valor no período *t* terá de subtrair a média calculada. Podemos ver essa transformação com mais detalhes na equação 4.1, em que $\underline{x_t}$ é a média da característica no tempo *t* e $x_{i,i}$ é o valor da característica da ação *i* no tempo *t.*

$$\begin{aligned} \underline{x_t} &= \frac{1}{N}\sum_{i=1}^{N} x_{i,t} \quad , \\ x_{i,i} &= x_{i,t} - \underline{x_t} \quad . \end{aligned} \tag{4.1}$$

4.1.2 Definição de limites financeiros

Outro aspecto relevante em relação aos dados utilizados são os limites impostos em algumas das informações. Por exemplo, o **custo de transação** dos ativos foi limitado à faixa

de valores de 0.02% no mínimo e 0.1% no máximo do retorno do ativo. Esses valores se dão por conta do tipo de pesos de *benchmark* utilizados anteriormente. Na literatura especializada, portfólios construídos com base em valor têm um custo de transação que oscila entre os valores mencionados.

Além disso, como mencionado na seção 2.4, o custo de transação está sendo calculado com base no volume de transações do ativo. Sendo assim, ativos com maior liquidez terão um custo de transação menor, aproximando-se de 0.02%, enquanto ativos com pouca saída no mercado terão um custo maior e perto de 0.1%.

Por fim, outra limitação financeira empregada neste trabalho é a da **alavancagem**. Na seção 2.6 foi mencionado o que é a restrição de alavancagem. Dentro do objetivo deste estudo, adotou-se uma restrição de no máximo 30% de alavancagem para cada período de tempo do portfólio. Isso possibilita a utilização desse portfólio, pois ele terá resultados mais próximos da realidade. Ambas as restrições mencionadas previamente foram adicionadas após a otimização de forma empírica a fim de melhorar o resultado obtido.

4.1.3 Períodos dos dados em cada iteração

Além dos ajustes nos dados mencionados nos dois subcapítulos anteriores, também serão descritos os períodos utilizados em cada iteração da validação. De acordo com a árvore de decisão do artigo de Vitor Cerqueira, Luis Torgo e Igor Mozetic [20] sobre técnicas de validação em séries temporais, constatou-se calculando a aceleração, o 5º e 95º percentil e a assimetria das séries temporais das características financeiras que a melhor técnica de validação seria a de ***growing window (Preq-grow).***

Como ilustrado na seção 3.4.2, o conjunto de dados é dividido em partes de treinamento e validação onde, a cada iteração, aumentou-se o tamanho dessas partes. Por conta dessa divisão, para que a avaliação do resultado final não seja enviesada, também foi utilizada outra parte dos dados para o teste do modelo obtido. Dito isso, na primeira iteração, obteve-se 20% dos dados para treinamento, 20% dos dados para validação e 20% dos dados para testes. Na segunda iteração, 40%, 20% e 20% para treinamento, validação e teste respectivamente. Por fim, temos a divisão de 60%, 20% e 20% na última iteração dessa validação. Os períodos dos dados que compreendem essa validação em cada iteração são indicados pela lista a seguir:

1ª Iteração:
- Treinamento: janeiro de 1995 a abril de 1998.
- Validação: maio de 1998 a setembro de 2001.
- Teste: outubro de 2001 a janeiro de 2005.

2ª Iteração:
- Treinamento: janeiro de 1995 a setembro de 2001.
- Validação: outubro de 2001 a janeiro de 2005.
- Teste: fevereiro de 2005 a junho de 2008.

3ª Iteração:

- Treinamento: janeiro de 1995 a janeiro de 2005.
- Validação: fevereiro de 2005 a junho de 2008.
- Teste: junho de 2008 a julho de 2008.

4.2 Especificação de softwares e hardwares utilizados

Uma informação que também é pertinente a este trabalho diz respeito aos requisitos de software e hardware para obter os resultados. Requisitos mínimos necessários do computador para conseguir executar o experimento: um processador Intel i5-4440 3.1GHz e 16GB RAM; o sistema operacional utilizado foi o Linux ubuntu 20.04LTS. Essas foram as configurações necessárias para que o experimento pudesse ocorrer. A quantidade de memória é muito importante, visto que a biblioteca do algoritmo que faz a otimização de hiperparâmetros consome bastantes recursos.

Outro ponto importante a ser mencionado é a linguagem de programação utilizada. Usou-se o Python na sua versão 3.8.5. Além disso, utilizou-se, dentre as bibliotecas de python mais importantes deste trabalho, o Pytorch na versão 1.9.0 e o Scipy na versão 1.7.0 para o desenvolvimento dos modelos linear e não linear utilizados. Em cada um destas bibliotecas, foi necessária uma customização no código na criação dos modelos gerados para calcular o erro utilizando a equação de utilidade mencionada previamente neste trabalho.

Também é bom destacar as outras bibliotecas de códigos, mas, por conta de a lista ser bem extensa, é mais apropriado apresentá-las ao indicar o código do trabalho em um link do Github[1]. Por último, o algoritmo 1 demonstra em alto nível o pseudo código da aplicação completa deste trabalho.

Entrada: Conjunto de observações $X_i = \{x_{i_1}, x_{i_2}, \cdots, x_{i_n}\}$ onde i é a característica utilizada e n é o período de tempo, neste caso mêses; Conjunto de porcentagens $P = \{p_i, p_2, \cdots, p_k\}$ utilizado para a divisão dos dados em cada iteração da validação onde k é o número de divisões.
Saída: Gráficos de comparação do modelo linear com o não linear no conjunto de testes.
Definir as divisões X_d dos dados de treinamento, validação e teste para realizar as iterações da validação.
Para cada divisão $X_d \in X_i$ **faça:**

Encontrar os melhores hiperparâmetros H com os dados de treinamento $x_t \in X_d$ avaliando o resultado no conjunto de validação $x_v \in X_d \mid x_v \neq x_d$.
Encontrar o modelo linear M_l realizando o treinamento com os dados de treinamento x_t avaliando-o com x_v

[1] O link para o trabalho completo no Github é dada pela url https://github.com/gabrielvieira37/master_thesis, e a lista completa das bibliotecas encontra-se em https://github.com/gabrielvieira37/master_thesis/blob/main/requirements.txt.

Encontrar o modelo não linear M_{nl} utilizando os melhores hiperparâmetros H com o treinamento utilizando os dados x_t e avaliando-o com x_v em cada iteração e acionando o *early stopping* caso necessário.

Utilizar os modelos M_l e M_{nl} no conjunto de teste $x_o \in X_d \mid x_o \neq x_t \, e \, x_v$ e avaliar a performance em um conjunto de dados fora da amostra.

Calcular e adicionar em uma lista a média do retorno mensal para cada modelo e benchmarks com o resultado no conjunto de teste.

Fim

Calcular a média do retorno mensal de cada modelo em relação a todas as divisões calculadas.

Criar um gráfico em barra para comparação dos modelos entre si e os benchmarks.

Algoritmo 1: Pseudocódigo da comparação dos modelos avaliados

4.3 Treinamento dos modelos

Após a descrição dos dados neste capítulo e também a descrição dos modelos de otimização no capítulo 2, torna-se possível explicar como se deram os experimentos deste trabalho. Foram utilizados dois algoritmos para critério de comparação: o primeiro é o BFGS, que foi abordado na seção 3.3; e o segundo são as redes neurais, que são comentadas na seção 3.4. Além disso, o BFGS será utilizado como a versão de dependência linear das características na função de otimização, enquanto as redes neurais serão utilizadas como a versão não linear.

Em cada um dos treinamentos, serão computadas as métricas dentro do conjunto de treinamento e validação, bem como as métricas com custo de transação e restrição de alavancagem mencionadas anteriormente. A seguir, será demonstrado o processo de treinamento de cada um desses algoritmos.

4.3.1 BFGS

Na otimização com dependência linear do BFGS, obtiveram-se resultados interessantes em todas as iterações de validação. Também é possível observar que, mesmo com a restrição de alavancagem, o retorno dos dados dentro da amostra é considerável. Na figura 4.1, veem-se mais detalhes do treinamento e resultados finais da métrica de retorno médio mensal.

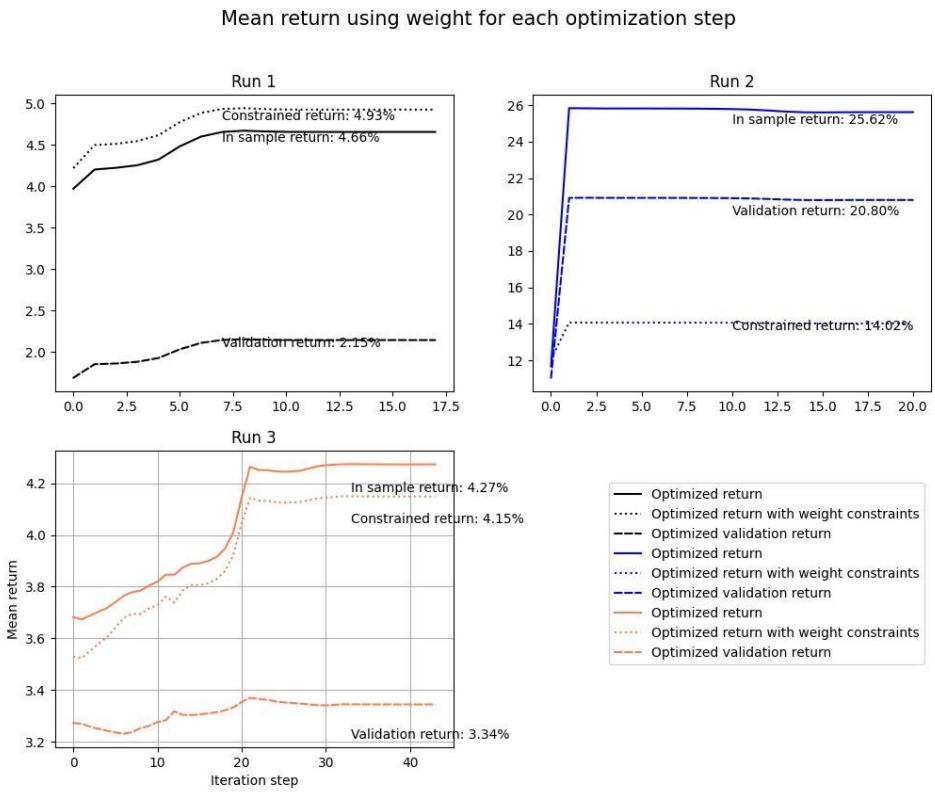

Figura 4.1: Variação do retorno médio mensal ao longo das iterações da otimização. Cada cor representa uma das iterações da validação.

Cada uma das cores do gráfico são diferentes iterações de validação. Ao final de cada otimização, são apresentados os valores dos retornos médios mensais obtidos. Os retornos In sample são os retornos utilizando dados de treinamento. Assim, nos retornos In sample validation, utilizaram-se dados de validação. Por fim, os resultados do In sample constrained return são os resultados a partir dos dados de treinamento com restrição de alavancagem. Não há uma mensagem para o retorno utilizando o custo de transação junto das outras restrições, pois o retorno da restrição de alavancagem se aproxima muito do valor obtido utilizando todas as restrições.

4.3.1 Redes neurais

As redes neurais foram adotadas como modelo de otimização não linear. Assim, é necessário informar uma série de configurações utilizadas em sua construção. Uma delas é a própria arquitetura a ser construída.

Nessa arquitetura, foram utilizados três neurônios para a primeira camada, representando a quantidade de características presentes na pesquisa. Portanto, caso fossem utilizadas mais características financeiras, seria necessário aumentar proporcionalmente o número de neurônios. A equação que define esta arquitetura pode ser vista na equação 4.2.

$$F: R^3 \rightarrow R^1 F = g\big(a\,(wx + b)\big) g: R^3 \rightarrow R^1 \quad 4.2$$

Em que g é a função de otimização definida no subcapítulo 3.4 e a função de custo é a minimização da saída desta rede negativada, ou seja, a maximização da função de otimização. Adicionalmente, x é um vetor de 3 posições assim como w e b. Cada uma das posições do vetor de entrada x é uma das características que foram mencionadas anteriormente. A função de custo da rede utilizando a função g negativada pode ser vista na Equação 4.3 a seguir.

$$\min_{\theta} -E_t \left[u \left(\sum_{i=1}^{N} f(x_{i,t}\,; \theta)\, r_{i,t} \right) \right] \quad 4.3$$

Além disso, a função Leaky ReLU [29] foi usada como função de ativação a nessa rede. A princípio, seria a função ReLU [29] por conta de seu bom desempenho na maioria das redes neurais; entretanto, por conta da característica do problema apresentado, a função Leaky ReLU seria mais adequada. Essa escolha se deve ao fato de que essa função de ativação permite o mapeamento dos valores a valores negativos, diferentemente da função ReLU, que transforma valores negativos em 0. Na equação 4.4 é possível analisar o mapeamento da Leaky ReLU.

$$f(x) = \begin{Bmatrix} x_i \Leftrightarrow x_i \geq 0 \\ \alpha x_i \Leftrightarrow x_i < 0 \end{Bmatrix} \quad 4.4$$

No mapeamento na equação 4.4, destaca-se α, o hiperparâmetro que define o coeficiente angular da função quando há valores de x_i menores que zero. Neste trabalho foi utilizado α=0.01. Além da função de ativação da rede, é importante mencionar a função de otimização utilizada para otimizar a função de custo do problema. Utilizou-se a função de otimização Adam – mencionada nos subcapítulos 3.4.1 e 3.4.4 – por conta da sua boa performance em diversos modelos.

Seus hiperparâmetros α, b1 e b2 foram otimizados por meio de uma otimização bayesiana a fim de descobrir os melhores valores para o problema aqui proposto. Ao utilizar tal otimização, também foram escolhidos, além dos parâmetros do Adam, os seguintes hiperparâmetros com seus respectivos limites superior e inferior.

Quantidade de épocas: Uniforme (500, 3000)
Taxa de aprendizado: Uniforme (1e-7, 1e-2)
b1: Uniforme (0.85, 0.95)
b2: Uniforme (0.99, 0.999)
α: Uniforme (1e-7, 1e-2)

Paciência: Uniforme (2, 10)

A quantidade de épocas e a taxa de aprendizado controlam o número de iterações de treinamento e o tamanho do passo dado na direção da otimização, respectivamente. Por último, a paciência controla o número de iterações que ocorrerão logo após identificar a ocorrência do *early stopping*.

Adicionalmente, a distribuição Uniforme corresponde à escolha dos valores iniciais para realizar as iterações da otimização bayesiana. Ao total, são 25 amostras iniciais selecionadas para cada um dos hiperparâmetros e 10 iterações realizadas, como explicado na seção 3.4.4. Por fim, após essas iterações, são escolhidos os hiperparâmetros que melhor performaram no conjunto de validação.

Os melhores hiperparâmetros encontrados nessa rede neural de uma camada em cada uma das três iterações de validação são indicados na tabela 4.1. Consistentemente, em todas as rodadas a otimização encontrou os mesmos valores para cada um dos hiperparâmetros. Feito isso, foi possível iniciar o treinamento de fato, utilizando-os para produzir os resultados daquele modelo.

Tabela 4.1: Melhores hiperparâmetros encontrados em cada rodada utilizando a otimização bayesiana na rede neural de uma camada.

Run	learning_rate	l2_regularization	epochs_size	adam_betha1	adam_betha2	patience
1	0,0156	0,0001	2.030	0,8854	0,9909	4
2	0,0156	0,0001	2.030	0,8854	0,9909	4
3	0,0156	0,0001	2.030	0,8854	0,9909	4

O **output** que foi utilizado desta rede neural é o valor dos pesos w encontrados na otimização desta rede, estes pesos w foram utilizados como os pesos das características financeiras θ. Assim, esses pesos serão utilizados na equação descrita na subseção 3.2, gerando o valor da função utilidade. Dessa forma, será realizado um **aprendizado não supervisionado**, pois estaremos utilizando apenas o valor da função utilidade para basear a escolha da direção do passo do algoritmo de otimização. Atualizaremos assim, os pesos w a cada iteração utilizando o gradiente da equação de otimização da função utilidade, utilizando o Adam como o otimizador base desta rede neural.

Além da função objetivo, é interessante observar o retorno médio utilizando os pesos de cada ação encontrados ao longo de cada interação do treinamento, pois, com essa informação, é possível analisar se a otimização está aumentando o retorno médio esperado. A Figura 4.2 exibe os valores do retorno esperado ao longo das iterações (*Epochs*) do treinamento. Nela, há quatro tipos de retorno para cada uma das três rodadas de validação. Há o retorno *In Sample* (dentro da amostra, ou apenas *NN return*), que é o retorno utilizando o conjunto de treinamento; o *Validation Return*, que ilustra o retorno utilizando o conjunto de validação; e, finalmente, há dois tipos de retorno que usam os conjuntos mencionados

anteriormente, porém com restrição (*constrained return*) de alavancagem, como mencionado no subcapítulo 2.6.

Nota-se que, ao longo do treinamento, todos os retornos cresceram em todas as rodadas de validação. Destaca-se o fato de que os retornos no conjunto de validação são menores que o retorno no conjunto de treinamento, como esperado, assim como os retornos com as restrições de alavancagem são menores que os retornos sem essas restrições.

Paralelamente, na figura 4.3 se observa a minimização do erro da função objetivo ao longo do treinamento e em cada iteração da validação. Ao analisar o gráfico, é notável que o erro de todos os modelos, seja no conjunto de validação, seja no de treinamento em todas as iterações da validação, diminuiu em comparação ao início do treinamento.

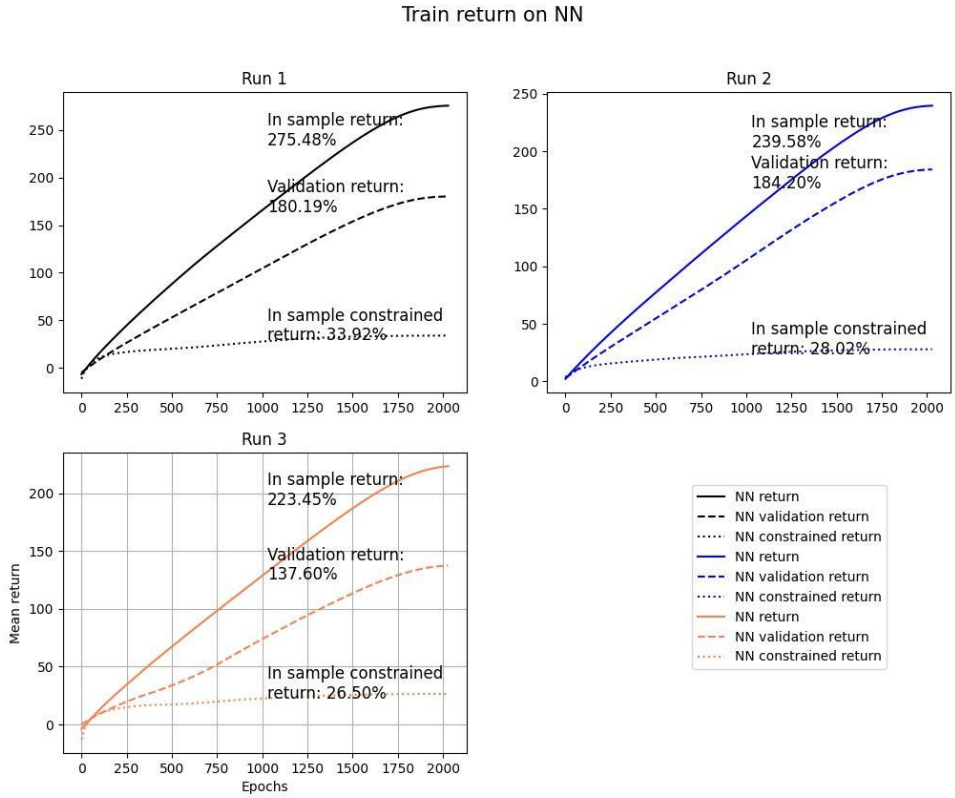

Figura 4.2: Variação do retorno médio ao longo das iterações da otimização. Cada cor representa uma das iterações da validação. Rede neural de uma camada.

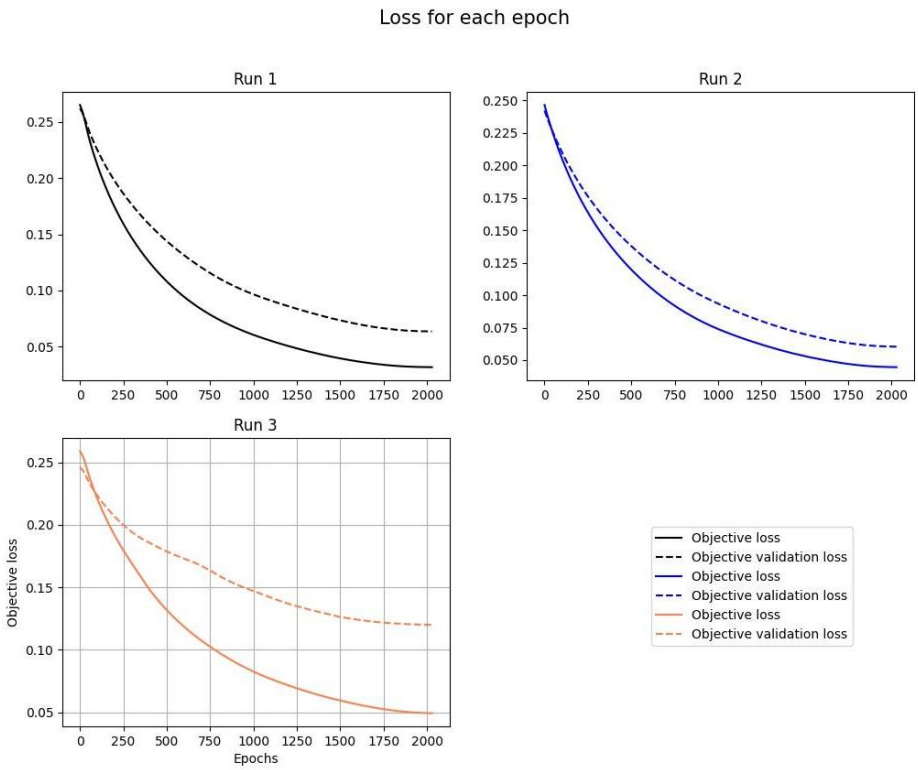

Figura 4.3: Variação do erro da função objetivo ao longo das iterações da otimização. Cada cor representa uma das iterações da validação. Rede neural de uma camada.

Além da rede neural de uma camada, também foi proposta a ideia de explorar, na pesquisa, um modelo mais complexo. Para isso, selecionou-se a rede neural novamente, mas com um número maior de camadas. Quanto maior o número de camadas, mais complexa é a função que pode ser modelada pela rede neural, e muitas vezes isso requer um número maior de dados.

Para este modelo mais complexo será utilizada uma arquitetura de duas camadas demonstrada pela equação 4.5. Nesta arquitetura os pesos e vieses da primeira assim como os da segunda camada possuem 3 dimensões, compativel com os dados x utilizados.

$$F: R^3 \rightarrow R^1 \ ,$$

$$F = g\big(\, a(w_2 a\,(\, w_1 x + b_1) + b_2)\big) \ , \qquad 4.5$$

$$g: R^3 \rightarrow R^1 \ .$$

Todo o processo realizado na rede de uma camada se repete nessa nova arquitetura, porém agora o cálculo da função objetivo ao longo treinamento utiliza o vetor encontrado após o processamento da segunda camada.

Na tabela 4.2, destacam-se os resultados da otimização de hiperparâmetros em cada uma das rodadas de validação. Diferentemente do modelo de apenas uma camada, cada rodada de validação encontra diferentes valores para os hiperparâmetros.

Tabela 4.2: Melhores hiperparâmetros encontrados em cada rodada utilizando a otimização bayesiana na rede neural de duas camadas.

Run	learning_rate	l2_regularization	epochs_size	adam_betha1	adam_betha2	patience
1	0,0998	1.53E-05	851	0,8938	0,9915	6
2	0,0156	0,0001	2.030	0,8854	0,9909	4
3	0,0261	1,00E-07	2.029	0,8802	0,9907	4

Nas figuras 4.4 e 4.5, constam, respectivamente, as informações sobre o retorno médio e a variação do erro da função objetivo nessa rede de duas camadas. É correto afirmar que o erro no conjunto de treinamento rapidamente chegou próximo de zero com poucas iterações, porém, ao analisar o retorno médio no início do treinamento, é possível ver que o modelo encontrou retornos negativos na primeira rodada de validação. Além disso, os erros dentro do conjunto de validação são maiores que aqueles obtidos no conjunto de treinamento, evidenciando uma dificuldade de generalização. Assim, isso indica uma necessidade de mais dados para um modelo mais complexo.

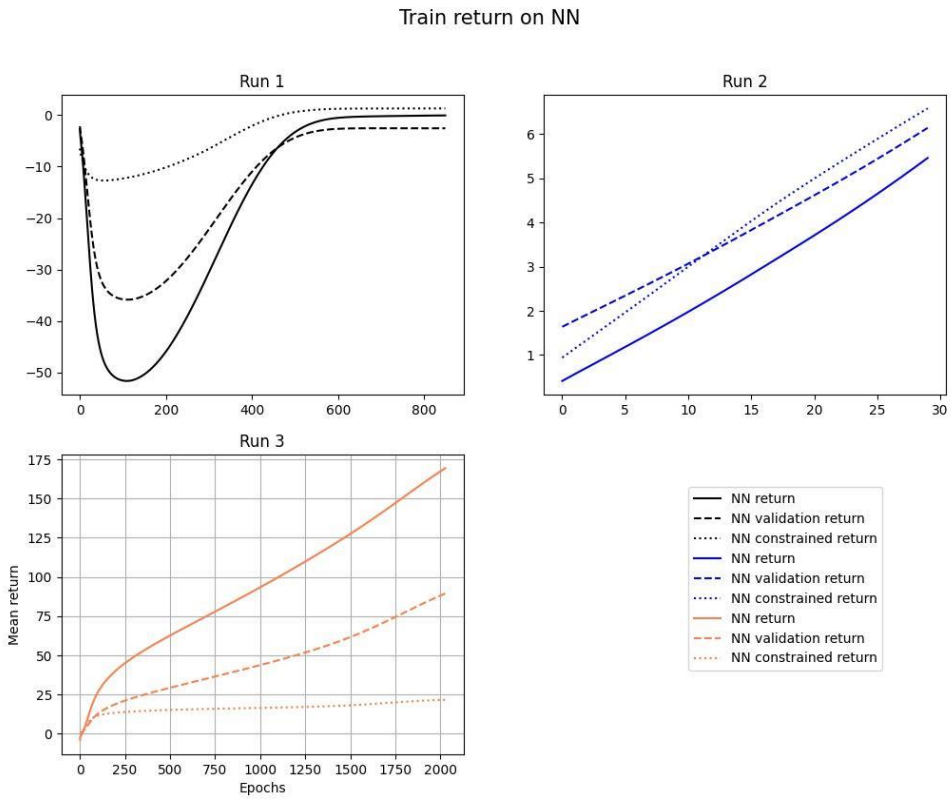

Figura 4.4: Variação do retorno médio ao longo das iterações da otimização. Cada cor representa uma das iterações da validação. Rede neural de duas camadas.

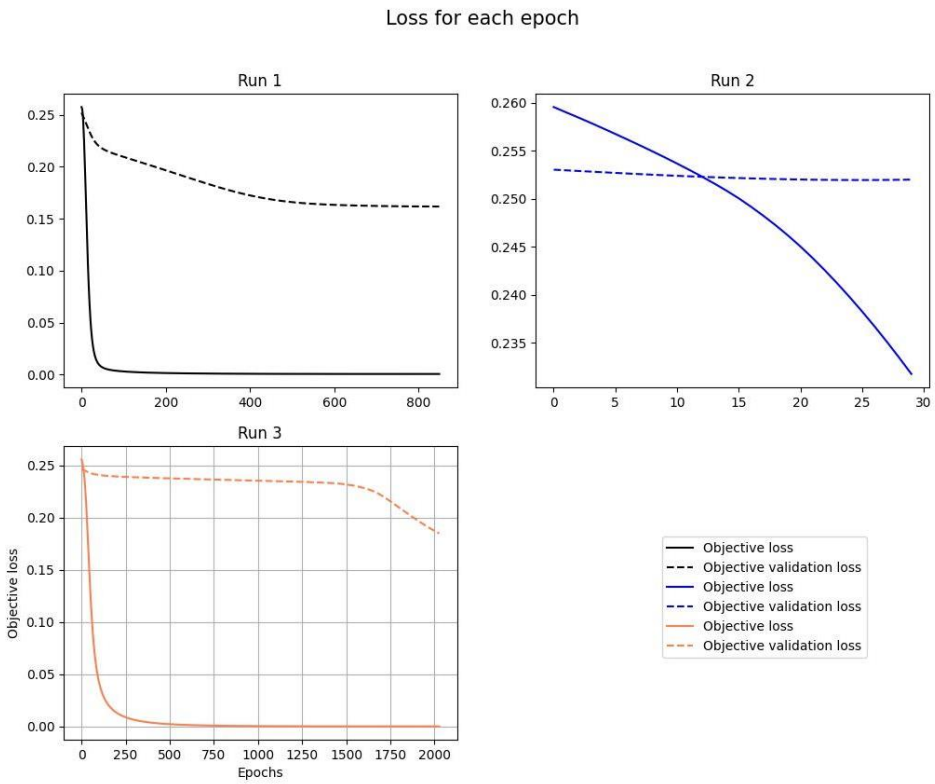

Figura 4.5: Variação do erro da função objetivo ao longo das iterações da otimização. Cada cor representa uma das iterações da validação. Rede neural de duas camadas.

Com isso é finalizado o treinamento dos modelos que serão comparados no capítulo de resultados. Neste próximo capítulo, serão abordados os resultados obtidos no conjunto de testes utilizando esses modelos, além de compará-los.

Capítulo 5

5 Resultados

No capítulo 5, apresentam-se resultados obtidos utilizando os modelos previamente mencionados. A subseção 5.1 levanta as perguntas que foram realizadas por esta pesquisa e que foram respondidas com os resultados alcançados .Na subseção 5.2, detalham-se os retornos médios, o *sharpe ratio* e as estatísticas relacionadas à distribuição de retorno dentro do conjunto de dados no período de teste. Em seguida, na subseção 5.3, há discussões acerca das diferenças nos resultados dos modelos utilizando restrições nos pesos do portfólio e da necessidade dessas restrições.

Os experimentos foram realizados buscando responder às seguintes perguntas:

- **Pergunta 1**: Redes neurais são um bom modelo não linear no contexto de portfólios parametrizados com características financeiras?
- **Pergunta 2**: É possível utilizar modelos de redes neurais mais complexas neste trabalho?
- **Pergunta 3**: As restrições financeiras impostas no problema ainda permitem um bom resultado?

A **pergunta 1** foca em verificar se o modelo escolhido para este trabalho consegue gerar uma função não linear genérica que modela bem o problema a ser resolvido. Já para obter a resposta da **pergunta 2** será preciso realizar o mesmo processo da primeira pergunta, porém com modelos mais robustos. Por último, para responder a **pergunta 3** será necessário analisar os resultados com e sem as restrições financeiras e verificar o quanto elas impactam na performance final.

5.1 Avaliando resultados no período de teste

Utilizando o método de validação *Preq-Grow,* obtiveram-se os retornos médios mensais de cada uma de suas rodadas no conjunto de teste. A Figura 5.1 exibe esses retornos, assim como seus desvios padrões. Nesta figura, utilizou-se a rede neural com a arquitetura de apenas uma camada.

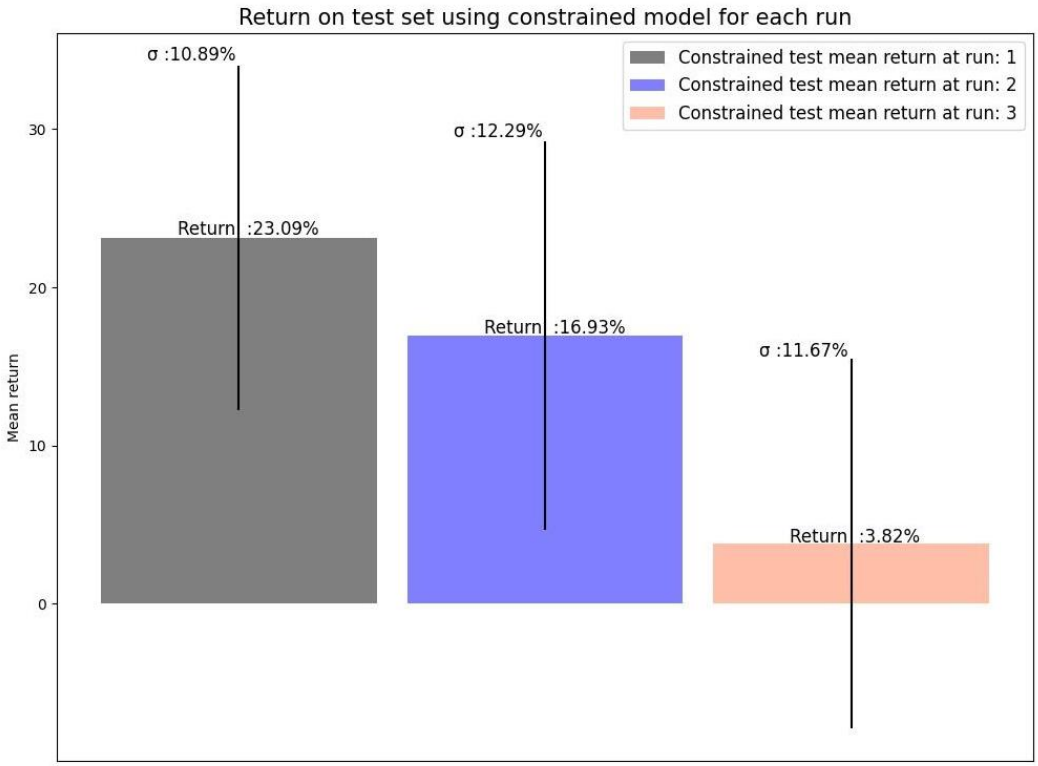

Figura 5.1: Retorno médio no conjunto de teste e sua variância em cada rodada do modelo restrito com rede neural de uma camada.

Vê-se o retorno médio mensal do período de teste em cada uma das rodadas de validação. Os períodos compreendidos são outubro de 2001 a janeiro de 2005, fevereiro de 2005 a junho de 2008 e julho de 2008 a novembro de 2011, respectivamente.

A primeira rodada possui o maior retorno médio mensal se comparada às outras rodadas, o que pode ser contraintuitivo, assumindo que a segunda e terceira rodadas utilizaram um período de dados maior em seu treinamento. Isso se deve ao fato de que, entre outubro de 2001 e janeiro de 2005, houve oportunidades de investimento melhores que os períodos das rodadas seguintes. Por conta do fato previamente mencionado, foi possível observar que, apesar de ter um período de treinamento pequeno em comparação às outras rodadas, o modelo conseguiu um bom resultado.

Com o retorno médio mensal de cada uma das rodadas, é possível criar uma visualização utilizando a média dos valores encontrados. Além do retorno da rede neural de uma camada, também podemos utilizar o retorno do modelo linear, do *benchmark*, da Selic/CDI e do IBovespa em cada uma dessas rodadas e fazer a média desses valores. A Figura 5.2 demonstra os retornos médios das rodadas de cada um desses modelos.

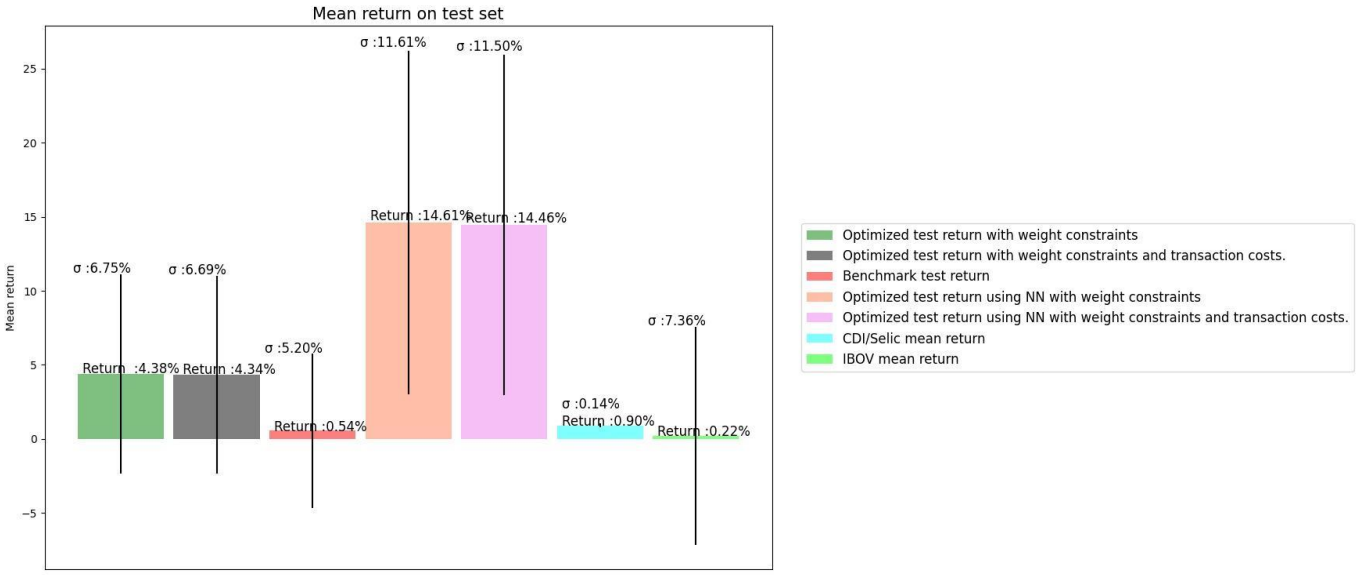

Figura 5.2: Retorno médio de todas as rodadas com o modelo linear e com o modelo da rede neural com uma camada.

Comparando todos os retornos médios obtidos, vê-se que o maior retorno é o da rede neural de apenas uma camada. Mesmo com os custos de transação inseridos no modelo final, houve um retorno extraordinário de 14% a.m. nesse modelo.

Além disso, tendo esse retorno um valor grande, esperava-se uma variância de tamanho correspondente, mas não é o que acontece. Ao analisar o *sharpe ratio* entre esse modelo e o modelo linear, verificou-se que o último tem uma taxa menor, como demonstrado na Figura 5.3.

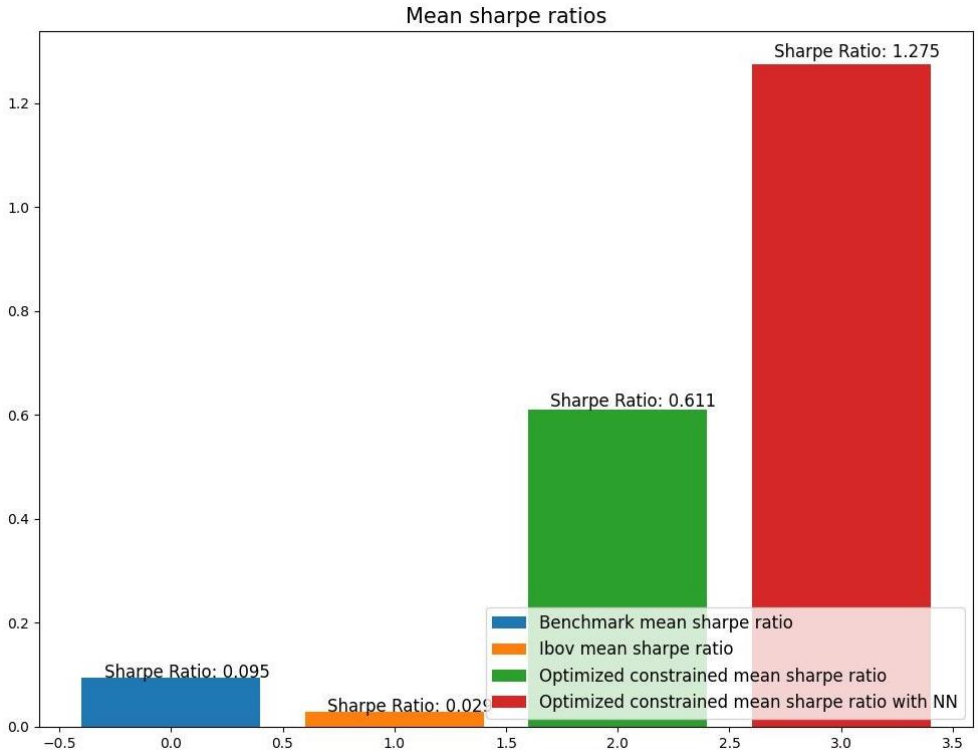

Figura 5.3: Sharpe ratio médio de todas as rodadas do modelo linear, do modelo com rede neural de uma camada, do Benchmark e do IBOV.

Observando a Figura 5.3, percebe-se que o *sharpe ratio* do modelo proposto com a rede neural com restrições tem um resultado melhor que os demais retornos. Com isso, garante-se que há robustez nos resultados encontrados, pois, mesmo com um retorno elevado, sua variância não dilui todo o resultado esperado.

Levando em consideração os bons resultados do modelo com a rede neural, experimentou-se uma arquitetura de rede mais complexa a fim de verificar se seriam obtidos resultados igualmente bons ou ainda melhores no mesmo conjunto de dados. Entretanto, a Figura 5.4 mostra que a arquitetura mais complexa de duas camadas não obteve um resultado tão bom quanto o modelo simples.

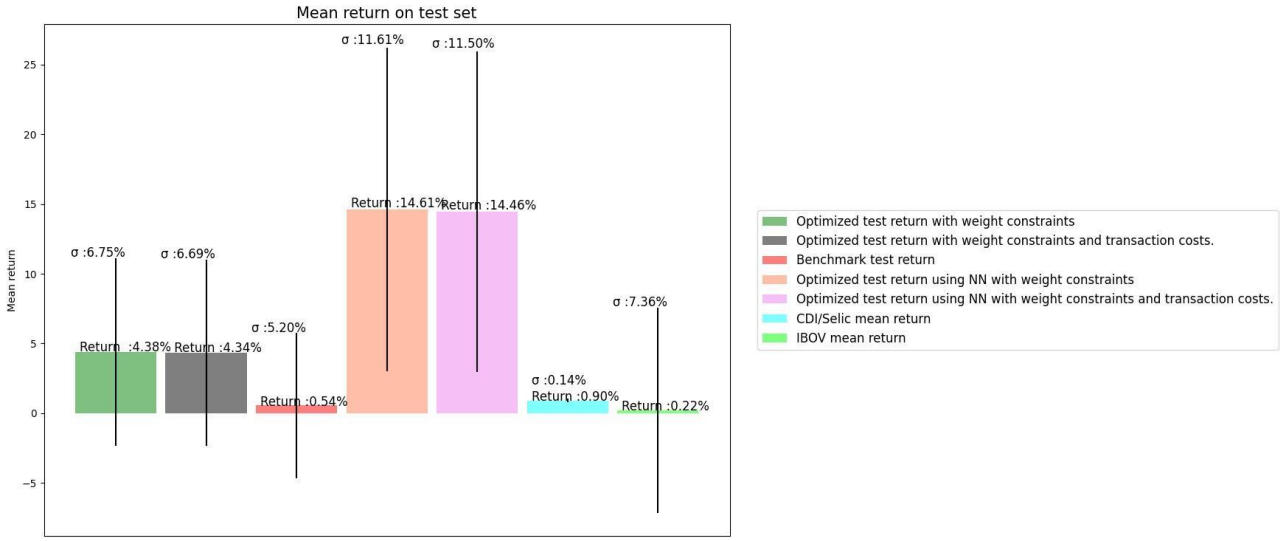

Figura 5.4: Retorno médio de todas as rodadas com
o modelo linear e com o modelo da rede neural com duas camadas.

Além dos retornos, verificou-se que o *sharpe ratio* também diminuiu, em comparação ao resultado do modelo simples. A figura 5.5 mostra que o *sharpe ratio* da rede neural com duas camadas está mais próximo do resultado do modelo linear do que do resultado da rede neural com apenas uma camada.

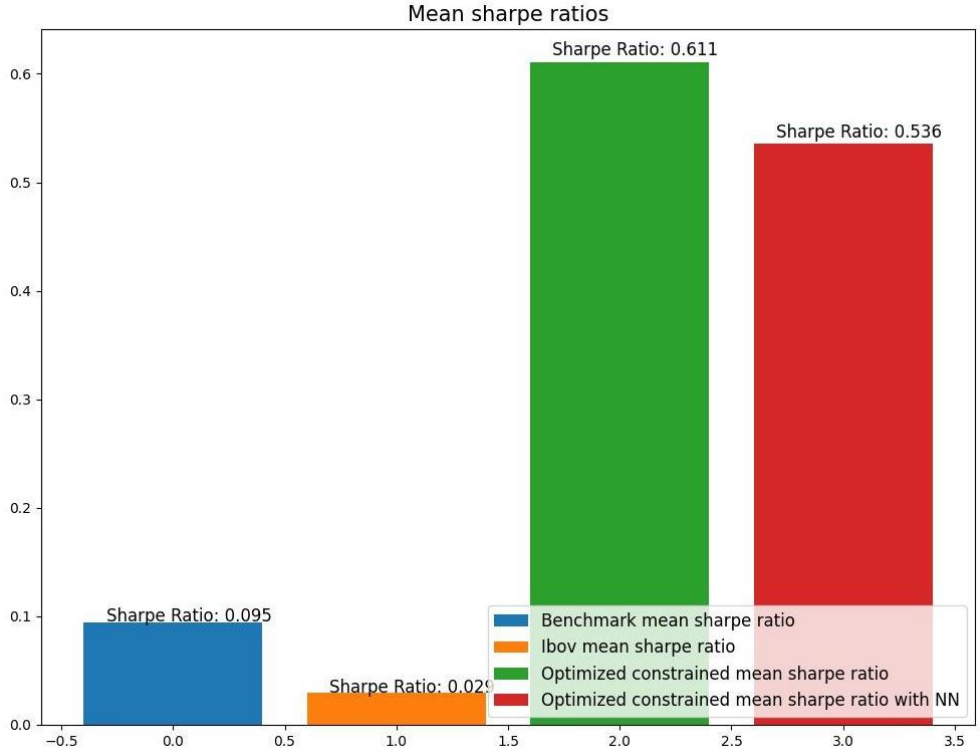

Figura 5.5: Sharpe ratio médio de todas as rodadas do modelo linear,
do modelo com rede neural de duas camadas, do Benchmark e do IBOV.

Inferiu-se, então, que a rede com duas camadas não é o modelo ideal a ser usado no conjunto de dados trabalhados, mas sim o modelo mais simples com apenas uma camada. Contudo, vale ressaltar que provavelmente um conjunto de dados com mais exemplos ou mais características poderia apresentar um resultado diferente ao utilizar uma rede mais complexa.

Por fim, as tabelas 5.1 e 5.2 corroboram a análise das redes de diferentes camadas. Nelas, é possível observar a diferença entre o retorno cumulativo obtido (*Cumulative Return*) entre as redes de uma camada na tabela 5.1 e as de duas camadas na tabela 5.2. É possível notar que tanto o retorno acumulado do modelo de redes sem restrição (*Neural Network*) quanto o dos modelos com restrição (*Neural Network Constrained*) das redes de uma camada superam os resultados das redes de duas camadas com uma boa margem.

Além do retorno acumulado, é possível ver o mesmo padrão na média do retorno mensal após subtrair o retorno sem risco da renda fixa da Selic/CDI (Average Excess Return), no qual a rede neural de uma camada apresentou melhor performance. Também é possível analisar, nestas duas tabelas, os diferentes valores de curtose (*Kurtosis*) [30] e assimetria (*Skewness*) [30] do retorno gerado nas duas arquiteturas.

Nos resultados da rede neural de uma camada, observa-se uma curtose maior que 3, de modo que apresenta uma distribuição Leptocúrtica (*Leptokurtic*) ou distribuição de cauda longa. Além da curtose, ao analisar a assimetria é possível ver que ela é positiva e maior que 1, ou seja, possui uma distribuição de cauda positiva. Essas duas informações indicam que há muitos retornos positivos de valor elevado na cauda dessa distribuição. Com tal característica, verificam-se um retorno médio satisfatório e maiores chances de obter um retorno positivo acima da média. [31]

Enquanto isso, na distribuição de retorno da rede de duas camadas, há diferentes valores para a curtose e assimetria. O valor da curtose dessa arquitetura é bem próximo de 3, podendo ser descrita como uma distribuição Mesocúrtica (*Mesokurtic*), que possui o mesmo tipo de cauda de uma distribuição normal. Por fim, ao analisar a assimetria do retorno desse modelo é notável que o valor se aproxima de 0, indicando uma distribuição quase simétrica e similar a uma distribuição normal. Devido a esses fatos, é possível dizer que a distribuição do retorno dessa arquitetura é semelhante a uma distribuição normal, indicando que os retornos tendem a seguir a média da distribuição, enquanto retornos positivos e negativos elevados são raros. [31]

Na tabela das redes de uma camada também é possível mencionar que os desvios padrões dos retornos do modelo não linear são maiores que o modelo linear. Essa afirmação sozinha poderia trazer uma preocupação quanto ao risco envolvido em investir no modelo não linear devido a sua volatilidade. Entretanto, quando olhamos mais atentamente ao desvio padrão na parte de retornos negativos (*Lower partial Standard Deviation)*, podemos ver que tanto o modelo linear quanto o não linear restritos possuem um desvio padrão muito próximo.

Isso nos mostra que o maior desvio padrão nas redes neurais, na realidade, se deve ao fato de que a rede neural produziu maiores retornos positivos que a modelagem da versão linear.

Por último podemos ver na mesma tabela que o retorno acumulado da rede neural é pelo menos 3 vezes maior que o retorno do modelo linear e que a restrição nas redes neurais proporcionou uma melhor distribuição dos pesos negativos e positivos. Isso nos leva a crer que a modelagem não linear e as restrições de fato geram bons resultados.

Tabela 5.1: Estatísticas do da distribuição de retorno no conjunto de teste utilizando o modelo de redes neurais com uma camada.

Statistic Name	*Neural Network*	*Neural Network Constrained*	*Optimized*	*Optimized Constrained*
Standard Deviation (%)	69,78	11,52	6,17	5,93
Lower Partial Standard Deviation (%)	31,96	5,18	4,75	4,32
Kurtosis	9,33	13,79	1,32	1,28
Skewness	2,33	2,97	-0,79	-0,70
Cumulative Return (%)	5066,91	362,35	105,57	111,69
Average Max. Weight	0,57	0,09	0,11	0,10
Average Min. Weight	-0,29	-0,01	-0,01	-0,01

Tabela 5.2: Estatísticas da distribuição do retorno no conjunto de teste utilizando o modelo de redes neurais com duas camadas.

Statistic Name	*Neural Network*	*Neural Network Constrained*	*Optimized*	*Optimized Constrained*
Standard Deviation (%)	59,29	7,90	6,17	5,93
Lower Partial Standard Deviation (%)	42,16	3,80	4,75	4,32
Kurtosis	2,93	6,72	1,32	1,28
Skewness	0,19	1,78	-0,79	-0,70
Cumulative Return (%)	351,62	244,10	105,57	111,69
Average Max. Weight	0,40	0,07	0,11	0,10
Average Min. Weight	-0,26	-0,01	-0,01	-0,01

5.2 Discussão dos resultados com e sem restrição

Ao analisar as figuras dos retornos na seção anterior podemos atentar ao fato de que alguns retornos estão com o adjetivo *constrained* (ou restrito), pois na seção anterior levou-se em consideração apenas retornos prováveis de acontecer. A restrição em questão é a de

alavancagem, que o portfólio pode ter em cada instante de tempo. Sem esse tipo de restrição, o modelo pode escolher posições *short-selling* muito altas, que por sua vez podem levar o investidor a tomar decisões arriscadas. Essas posições de risco são descritas e explicadas no artigo *Short-Selling Risk* [32].

Sendo assim, com a restrição de alavancagem é possível ter um resultado mais próximo ao que seria produzido na realidade, ou, em outras palavras, mais confiável. Na Figura 5.6, há uma comparação entre os resultados encontrados com e sem a restrição dos modelos linear e de rede neural simples. Nesta subseção, utilizou-se apenas o modelo de rede neural com uma camada para comparação, pois foi o que obteve o melhor resultado.

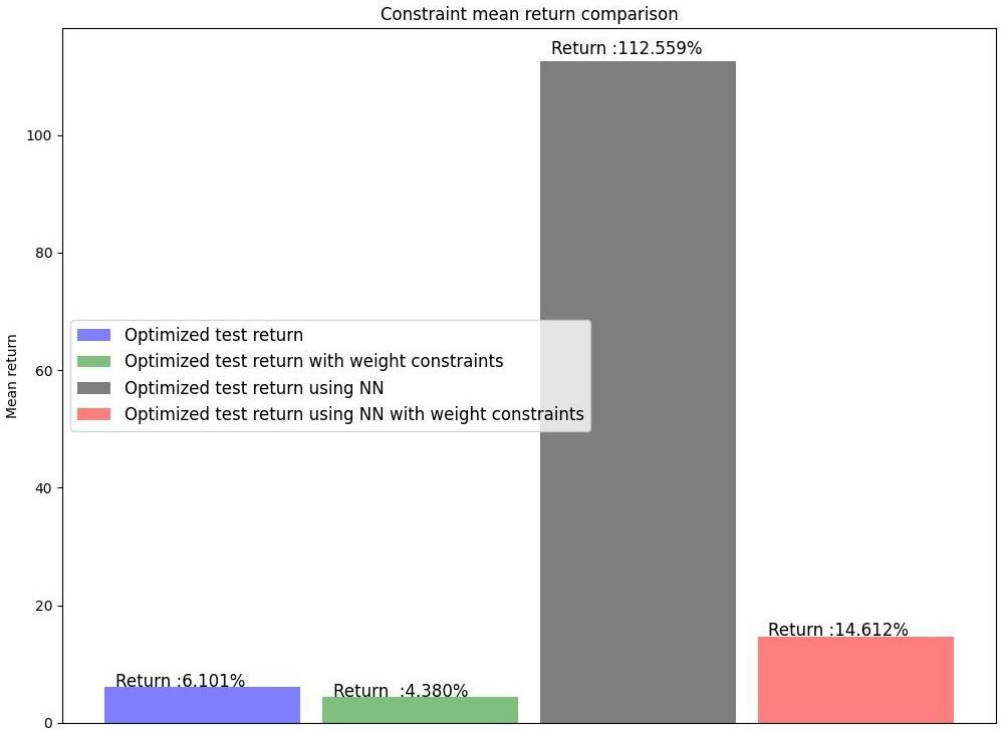

Figura 5.6: Comparação entre os retornos médios da versão com e sem restrição do modelo linear e do modelo de rede neural com duas camadas.

Nota-se que os retornos, no caso do modelo de rede neural, são muito discrepantes entre os resultados com e sem restrição. Isso ocorre porque o modelo de rede neural gera uma alavancagem muito grande e permite retornos altos. Entretanto, quando se calcula o retorno acumulado no período de teste sem restrição, existe um momento em que o portfólio quebra, ou seja, tem retorno negativo de 100% ou mais.

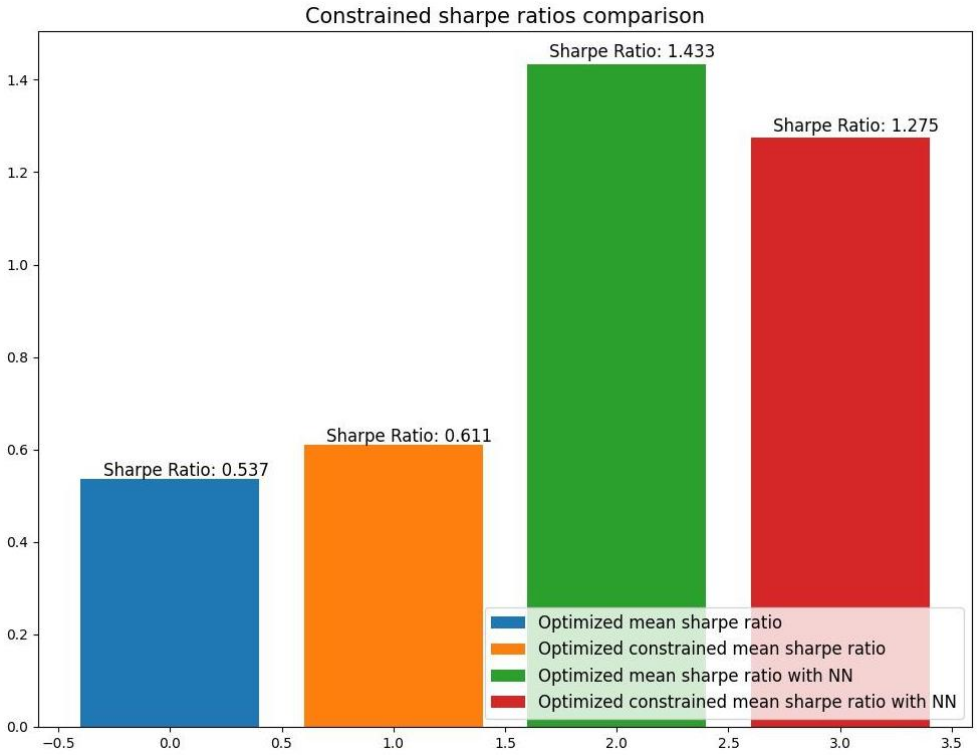

Figura 5.7: Comparação entre os sharpe ratios do modelo linear e do modelo da rede neural de uma camada com e sem restrição.

Ao analisar a figura 5.7, vê-se que o *sharpe ratio* na rede neural com restrição diminui em comparação à sem restrição. Isso mostra que, mesmo com a diminuição do risco e, consequentemente, da volatilidade do retorno, não foi o bastante para que o *sharpe ratio* do modelo com restrição fosse maior. Constata-se que a diminuição do seu retorno foi mais impactante que a diminuição da sua volatilidade. Em contrapartida, no modelo linear, o maior *sharpe ratio* é o da versão com restrição, indicando que a diminuição da volatilidade foi maior que a diminuição no seu retorno final.

Além disso, tanto neste trabalho quanto em outros estudos sobre a utilização de restrições em portfólios indicam que utilizar algum tipo de restrição é melhor que não utilizar nada. Esse fenômeno é descrito no artigo [33]. Nele, os autores escolheram analisar a restrição na qual os pesos precisam necessariamente ser positivos, ou seja, o portfólio não deve ter posições em *short-selling.* Eles chegaram à conclusão prática de que a imposição de restrições permite que o portfólio ainda tenha uma boa performance em razão da diminuição do risco envolvido ao tomar posições *short-selling*.

Com isso, observou-se que, apesar das restrições impostas aos portfólios usados neste trabalho, os resultados encontrados ainda são melhores que os resultados dos *benchmarks* selecionados para comparação. Por fim, compararam-se as distribuições dos pesos dos portfólios gerados com e sem restrição utilizando a divergência Jensen-Shannon [34] como medida de distância para verificar a similaridade das distribuições positivas e negativas em comparação a sua versão com e sem restrição. A divergência Jensen-Shannon é baseada na divergência KL [35], mas difere desta última, pois, ao comparar duas

distribuições usando a Jensen-Shannon, obtiveram-se resultados entre 0 e 1, o que facilita o entendimento do quão próxima ou distante as distribuições estão entre si.

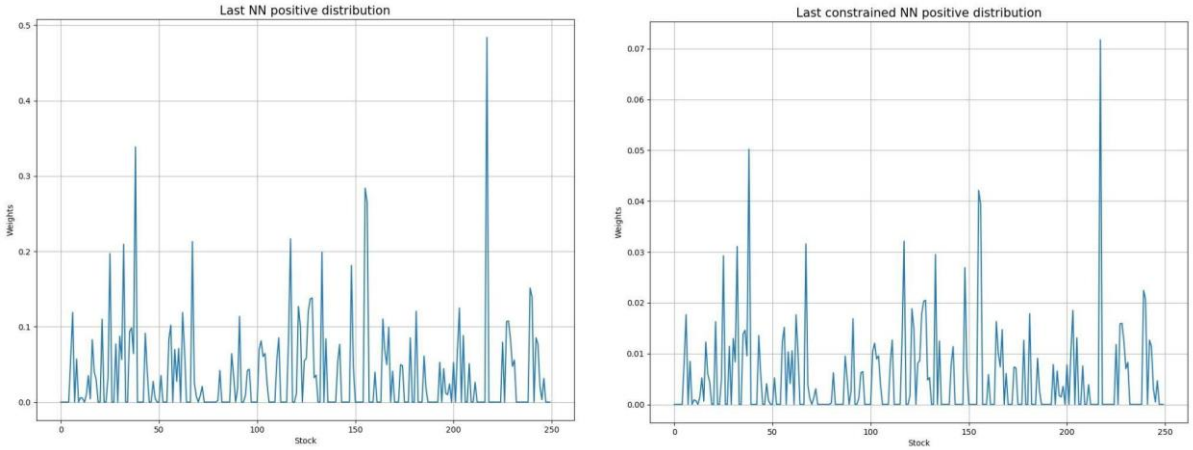

Figura 5.8: Parte positiva da distribuição dos pesos do portfólio com e sem restrição.

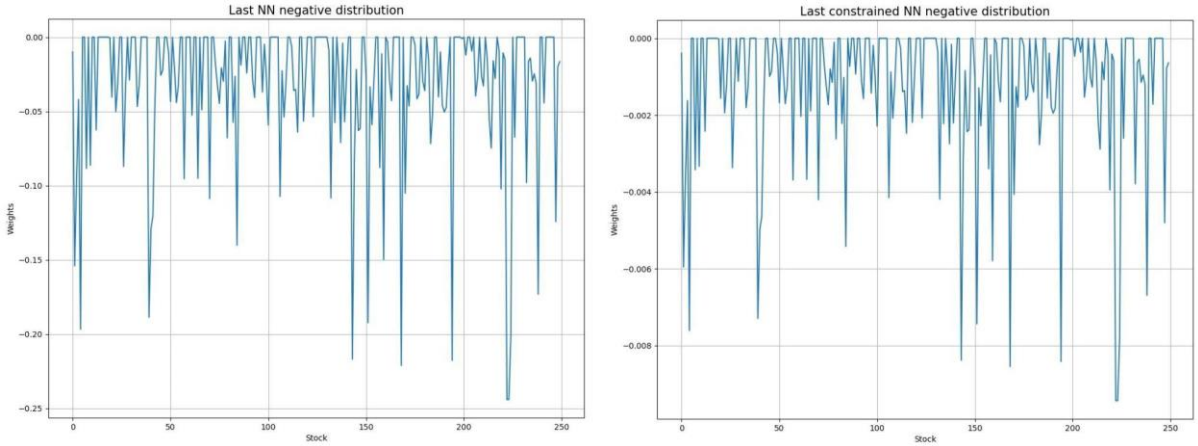

Figura 5.9: Parte negativa da distribuição dos pesos do portfólio com e sem restrição.

As figuras 5.8 e 5.9 exibem as partes positiva e negativa das distribuições com e sem restrição com base no último mês do conjunto de teste do modelo com rede neural. Visualmente, os formatos das distribuições são bem similares, e as diferenças estão apenas na escala dos valores gerados. Para confirmar essa similaridade, utilizou-se a divergência Jensen-Shannon e verificou-se que a distância entre a versão com e sem restrição dessas distribuições é zero. Além disso, a média da divergência para todos os meses calculados com e sem restrição também é zero. Isso reforça a hipótese de que a distribuição com restrição pode ser utilizada sem muitas perdas em comparação à original.

Capítulo 6

6 Conclusão

Com base nos resultados apresentados, pode-se concluir que as estruturas das características financeiras possuem, de fato, uma relação não linear entre si. Essa afirmação se deve aos bons resultados obtidos nas análises em comparação aos dois métodos apresentados. Além disso, este trabalho propõe uma nova abordagem na criação de portfólios de ações ao explorar as não linearidades por meio de redes neurais para essa finalidade. Após utilizar esta nova abordagem, é possível obter resultados mais promissores que a sua modelagem linear.

Verificou-se também o problema da complexidade do modelo utilizado quando experimentada uma arquitetura com mais camadas na rede neural. Esse problema ocorreu devido à pequena quantidade de dados, não permitindo a utilização de modelos mais complexos. Adicionalmente, foi possível observar que as restrições de alavancagem nos pesos das ações mantiveram a estrutura de suas distribuições e, com isso, é notável que os resultados apresentados foram bastante satisfatórios em termos do retorno médio mensal.

Finalmente, compete mencionar que os resultados obtidos no conjunto fora da amostra foram melhores que o esperado. Uma vez que eles excederam as expectativas, é justo concluir que este trabalho abre caminho para futuros testes e experimentações.

6.1 Trabalhos futuros

É possível propor os seguintes itens como experimentações futuras:

- Testar múltiplos inícios aleatórios diferentes e fazer um intervalo de confiança para cada valor do retorno encontrado;
- Adicionar mais dados da bolsa brasileira e tentar novamente o modelo mais complexo de redes neurais;
- Realizar um experimento com mais características financeiras além das três propostas neste trabalho;
- Utilizar dados de diferentes mercados mundiais;
- Experimentar modelos específicos para séries temporais, como o LSTM(Long Short Term Memory) [36] , CNN 1D (Convolutional Neural Network 1 Dimension) [37] etc.;.
- Utilizar coeficientes dinâmicos, permitindo que as características possuam diferentes pesos θ com o passar do tempo

Referências Bibliográficas

[1] Markowitz, Harry M. Portfolio Selection: Efficient Diversification of Investments. Yale University Press, 1959.

[2] Fama, Eugene F., and Kenneth R. French. “The Cross-Section of Expected Stock Returns.” The Journal of Finance 47, no. 2, pp 427–65, 1992.

[3] Michael W. Brandt, Pedro Santa-Clara, Rossen Valkanov, Parametric Portfolio Policies: Exploiting Characteristics in the Cross-Section of Equity Returns. The Review of Financial Studies, vol. 22, no. 9, p.p. 3411–3447,2009.

[4] Bryzgalova, S., M. Pelger, and J. Zhu . Forest through the trees: Building cross-sections of stock returns, 2020.

[5] Chen, L., M. Pelger, and J. Zhu . Deep learning in asset pricing, 2020.

[6] Freyberger, J., A. Neuhierl, and M. Weber. Dissecting characteristics nonparametrically. The Review of Financial Studies, vol. 33, nº 5, pp. 2326-2377, 2020.

[7] Gu, S., B. Kelly, and D. Xiu. Empirical asset pricing via machine learning. The Review of Financial Studies, vol. 33, nº 5, pp. 2223-2273, 2020.

[8] Gu, S., B. Kelly, and D. Xiu. Autoencoder asset pricing models. Journal of Econometrics, vol. 222, nº 1, pp. 429-450, 2021.

[9] Caldeira, João and A. P. Santos, Andre and Torrent, Hudson, Semiparametric Portfolios: Improving Portfolio Performance by Exploiting Non-Linearities in Firm Characteristics , 2022.

[10] Ruppert, D., Wand, M., & Carroll, R. Semiparametric Regression (Cambridge Series in Statistical and Probabilistic Mathematics). Cambridge: Cambridge University Press, 2003.

[11] Nocedal, J. Updating Quasi-Newton Matrices with Limited Storage. Mathematics of Computation, vol. 35, nº 151, pp. 773–782, 1980.

[12] Ypma, Tjalling J. Historical Development of the Newton-Raphson Method. SIAM Rev. 37, p.p 531-55, 1995.

[13] Mykel J. Kochenderfer and Tim A. Wheeler. Algorithms for Optimization. The MIT Press, p.p 21, 2019.

[14] Werbos, Paul J.. “Backpropagation Through Time: What It Does and How to Do It.” Proc. IEEE 78, p.p 1550-1560, 1990.

[15] Bottou, L.; Bousquet, O. The tradeoffs of large scale learning. Advances in Neural Information Processing Systems 20. Curran Associates, Inc. p.p. 161–168, 2008.

[16] Tieleman, T.; Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[17] Kingma, Diederik & Ba, Jimmy. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations, 2014.

[18] Sutskever, I. & Martens, J. & Dahl, G. & Hinton, G.. On the importance of initialization and momentum in deep learning. 30th International Conference on Machine Learning, ICML. p.p 1139-1147, 2013.

[19] Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. IJCAI, 1995.

[20] Cerqueira, V., Torgo, L. & Mozetič, I. Evaluating time series forecasting models: an empirical study on performance estimation methods. Mach Learn 109, p.p. 1997–2028, 2020.

[21] Ying, X. An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series 1168, 2019.

[22] Srivastava, Nitish et al. “Dropout: a simple way to prevent neural networks from overfitting.” J. Mach. Learn. Res. 15, p.p. 1929-1958, 2014.

[23] Prechelt, Lutz. “Early Stopping-But When?” Neural Networks: Tricks of the Trade ,1996.

[24] Krogh, Anders and John A. Hertz. “A Simple Weight Decay Can Improve Generalization.” NIPS, 1991.

[25] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: importance of hyperparameters of machine learning algorithms. J. Mach. Learn. Res. vol. 20, nº 1, p.p 1934-1965, 2019.

[26] Yu, Tong and Hong Zhu. “Hyper-Parameter Optimization: A Review of Algorithms and Applications.” ArXiv abs/2003.05689, 2020.

[27] Snoek, Jasper et al. “Practical Bayesian Optimization of Machine Learning Algorithms.” ArXiv abs/1206.2944, 2012.

[28] Plyakha, Yuliya et al. “Why Does an Equal-Weighted Portfolio Outperform Value- and Price-Weighted Portfolios?” Econometric Modeling: Capital Markets - Portfolio Theory eJournal, 2012.

[29] M. Khalid, J. Baber, M. K. Kasi, M. Bakhtyar, V. Devi and N. Sheikh, "Empirical Evaluation of Activation Functions in Deep Convolution Neural Network for Facial Expression Recognition," 43rd International Conference on Telecommunications and Signal Processing (TSP), pp. 204-207, 2020.

[30] Joanes, D. N., and C. A. Gill. “Comparing Measures of Sample Skewness and Kurtosis.” Journal of the Royal Statistical Society. Series D (The Statistician), vol. 47, no. 1, pp. 183–89, 1998.

[31] Conrad, J.S., Dittmar, R.F., & Ghysels, E. “Ex Ante Skewness and Expected Stock Returns”. Capital Markets: Market Efficiency eJournal. 2009.

[32] Engelberg, Joseph and Reed, Adam V. and Ringgenberg, Matthew C., Short-Selling Risk. Journal of Finance, Forthcoming, UNC Kenan-Flagler, 2017.

[33] Jagannathan, R., & Ma, T. . Risk Reduction in Large Portfolios: Why Imposing the Wrong Constraints Helps. National Bureau of Economic Research, 2002.

[34] J. Lin, "Divergence measures based on the Shannon entropy," in IEEE Transactions on Information Theory, vol. 37, no. 1, pp. 145-151, Jan. 1991.

[35] Joyce, J.M. Kullback-Leibler Divergence. In: Lovric, M. (eds) International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg, 2011.

[36] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. Neural Computation. vol. 9, no. 8, p.p. 1735–1780, 1997.

[37] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D.J. Inman. "1D convolutional neural networks and applications: A survey". Mechanical Systems and Signal Processing, vol. 151, 2021.

[38] Rumelhart, David E., Geoffrey E. Hinton and Ronald J. Williams. “Learning representations by back-propagating errors.” Nature 323. p.p. 533-536,1986

[39] Sharpe, William F. "CAPITAL ASSET PRICES: A THEORY OF MARKET EQUILIBRIUM UNDER CONDITIONS OF RISK*." The Journal of Finance 19, no. 3, p.p. 425-442, 1964.

[40] Sharpe, William F.. “The Sharpe Ratio.” The Journal of Portfolio Management no. 21 (1) p.p 49–58, 1994.