PUC

Projeto Final de Graduação
Ciência da Computação

Predição e Prescrição de Portfólio Financeiro utilizando Aprendizado de Máquina e Otimização de Portfólio

Orientador: **Hélio Cortez Vieira Lopes**
Aluno: **Flávio Sérgio da Silva**

Departamento de Informática

Rio de Janeiro, dezembro de 2021

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Sumário

Agradecimentos

A minha mãe Maria da Glória Pinheiro, que também acumulou a função de pai, ao qual eu dedico este trabalho, pelos valores que carrego comigo até hoje, e que me tornou uma pessoa melhor através do seu amor, e da sua perseverança em ver o seu filho se formar na mesma faculdade que os filhos da patroa dela, quando ela era empregada doméstica em uma casa de família ao lado da PUC. Aos meus irmãos Sílvio, Fábio e Fabiana pela trajetória de vida e pelo amor fraterno e verdadeiro. Ao meu irmão mais novo João Vitor por quem vale a pena construir um mundo melhor.

Ao meu ex-chefe e grande amigo Antônio de Barros Pinheiro pelo grandioso ensinamento ético e moral, sem as orientações dele eu não teria chegado nem na metade desta trajetória. Ao meu grande amigo, de todas as horas, Guaracy Mendes que também acumula a minha grande admiração profissional.

Em ordem temporal, ao professor Markus Endler que não negou esforço quando eu mais precisei, no qual eu sou eternamente grato. Ao professor Hélio Lopes pela orientação e transferência de conhecimento em *Estocástico*. A professora Karla Figueredo pelo seu dom de ensino gentil e humanizado, e pela coorientação em *Inteligência Artificial*. Ao professor Davi Valladão pela incrível receptividade, atenção, e coorientação em *Otimização de Portfólio*. Ao professor Sinésio Pesco que é um dos melhores educadores que eu já conheci, não só pela sua grande capacidade técnica e didática, mas também pelo domínio da arte de obter o respeito dos alunos respeitando os alunos, foi uma honra receber as coorientações em *Matemática* deste grande ser humano.

Aos meus colegas do início de curso, que se tornaram amigos, Pedro Coutinho, Guilherme Guia, Michael Portela, e Douglas Pires, que apesar da diferença de idade, me ensinaram muito, principalmente como ser jovem novamente. E aos meus novos amigos de curso, principalmente ao Victor Fróes.

Aos muitos outros amigos, familiares, e professores que me ajudaram direta ou indiretamente na minha graduação e na conclusão desta *P&D*, e por sempre terem acreditado em mim, e que estão no meu coração.

Resumo

Esta pesquisa propõe um modelo híbrido, para a predição e prescrição de carteira de títulos financeiros, baseados em *Aprendizado de Máquina* e *Otimização de Portfólio*, utilizando respectivamente os modelos *Tabnet* e *Markowitz*. A partir do *Tabnet* é possível identificar quais *features* mais influenciaram no modelo de predição, e o *Markowitz* é utilizado para reduzir o risco da carteira. Eu realizo simulações a partir da combinação de várias configurações dos modelos, e sinalizo qual foi o cenário com menor erro médio. Ao final, eu reexecuto o modelo utilizando o melhor cenário e *prescrevo* (proponho) a configuração (*pesos*) ideal da carteira.

Palavras-chave

Predição, Prescrição, Preço, Volume, Risco, Média, Variância, Covariância, Volatilidade, Carteira, Portfólio, Ações, Opções, Moeda, Digital, Forex, Bitcoin, Criptomoeda, Mercado Financeiro, Tabnet, Markowitz, Estocástico, Aprendizado de Máquina, Inteligência Artificial, Otimização de Portfólio, QuantRio.

Abstract

This research proposes a hybrid model for the *prediction* and *prescription* of financial *portfolio*, based on *Machine Learning* and *Optimization*, using respectively the *Tabnet* and *Markowitz* models. From the *Tabnet* it is possible to identify which *features* most influenced the *predictor* model, and the *Markowitz* is used to reduce the *portfolio´s* risk. I perform simulations by combining various model configurations, and I signal which was the scenario with the lower average error. At the end, I re-run the model using the best scenario and *prescribing* (propose) the ideal *portfolio´s* configuration (weights).

Keywords

Prediction, Prescription, Price, Volume, Risk, Average, Mean, Variance, Covariance, Volatility, Portfolio, Portfolio, Stocks, Options, Currency, Digital, Forex, Bitcoin, Cryptocurrency, Money Market, Tabnet, Markowitz, Stochastic, Machine Learning, Intelligence Artificial, Optimization, QuantRio.

Significância/Contribuição

A relevância científica deste projeto é a combinação entre modelos de *Aprendizado de Máquina* e *Otimização de Portfólio*, para servir como "*baseline*" (ponto de partida) acadêmico de graduação, para futuros trabalhos científicos nesta área.

Responsabilidade

Esta P&D é puramente acadêmica, e visa ilustrar e comparar prováveis modelos para predição de preços de "*asset*' (ativo financeiro). Contudo, este projeto não deve ser utilizado em operações reais sem as devidas modificações nos modelos apresentados, por não garantirem bons resultados em operações reais. Todos os riscos ficam a cargo de quem utilizar os modelos acadêmicos aqui apresentados.

Divulgação

Eu, como autor, afirmo não existir nenhum conflito de interesse sobre este trabalho acadêmico.

Aprendizado

Esta *P&D* sumariza o meu aprendizado nas áreas de predição e prescrição adquiridos na minha graduação, cujo o qual eu pude aplicar os conhecimentos de *Aprendizado de Máquina* e *Otimização de Portfólio*.

1 Introdução

Considerando a necessidade de predição do comportamento de *asset*, faz-se necessário a utilização de mecanismos para a redução das incertezas envolvidas neste processo de tomada de decisão, maximizando o potencial de ganho.

Pesquisar sobre predição de *asset* é desafiador por se tratar de um sistema de comportamento complexo, instável, onde o mercado financeiro é influenciado por vários fatores conhecidos e desconhecidos. Dentre as complexidades das *séries temporais* financeiras temos, não "*estacionariedade*" (*média*, *variância* e *autocorrelação* não mudam com o tempo), não linearidade, não trivialidade de "*escalamento*" (*normalização*), e a "*heterocedasticidade*" (*variância* condicionada ao tempo).

Pequeníssimos avanços, na ordem de $10^{-2}\%$, nesta área podem ser convertidos em grandes benefícios, na ordem de $10^{+7}\$$. Ou seja, um *portfólio* (carteira) de 10 Bilhões (em qualquer moeda) utilizando um bom modelo que supere em 0,10% (zero vírgula dez por cento) da média de resultados de outros modelos concorrentes, resultará em um *retorno* (lucro) de 100 Milhões a mais em relação aos modelos concorrentes.

2 Situação Atual

Atualmente são utilizados inúmeros mecanismos para amparar a tomada de decisão, no mercado financeiro, como as apresentadas abaixo:

- ***Métodos de Análise Técnica –*** A "*Análise Técnica*" faz uso de vários indicadores (tendência, osciladores e volume), tais como, MACD, CCI, RSI, Bollinger Bands, Heiken-Ashi, entre outros. Estes métodos são largamente utilizados e facilmente encontrados em ferramentas de análise técnica e/ou "*home broker*" (corretor residencial). O investidor doméstico se sente mais confortável em utilizar estes métodos.
- ***Métodos Clássicos*** – Uso customizado de *Probabilidade*, *Estatística* e *Otimização de Portfólio*. Estes métodos possuem muita teoria matemática, o que os tornam menos populares para um investidor comum sem muito conteúdo matemático acumulado. Estes métodos são mais utilizados por empresas gestoras de riqueza.
- ***Métodos Quantitativos –*** Ou simplesmente "*Métodos Quant*", são mecanismos que propõem utilizar tudo o que estiver disponível e seja factível, através de automatização computacional para amparar a tomada de decisão. O *Aprendizado de Máquina* é largamente utilizado e a *Otimização de Portfólio* é um pouco menos. Estes métodos são tidos como mais fáceis que os *Métodos Clássicos*, e tendem a se tornar cada vez mais populares devido a prospecção na área de *Aprendizado de Máquina* e também por não necessitarem de grandes conhecimentos de codificação de algoritmo, muito menos de grandes conhecimentos matemáticos, para fazer um modelo *ingênuo*. Atualmente, existe um movimento na indústria financeira onde tanto os investidores domésticos como as gestoras de riqueza estão utilizando *Métodos Quant* para amparar a tomada de decisão. Os investidores domésticos têm aprendido *Métodos Quant* e os grandes bancos têm adquirido "*StartUps*" (empresa recém-criada) com especialização em *Métodos Quant*. Esta *P&D* está mais próxima de *Métodos Quant*.

Os mecanismos de predição acima são construídos de maneira distinta a depender do objetivo, conforme possibilidades abaixo:

- ***Trade System (Painel de Sinais) –*** Consiste em um "*dashboard*" (painel) com sinalizações, sonora e/ou visual, de subida ou descida do valor do *asset* que se está monitorando.
- ***API (Biblioteca) –*** Artefato computacional sobre a forma de "*library*" (biblioteca) a ser incorporada em outros artefatos, onde o primeiro disponibiliza a predição e o segundo faz uso da informação predita.
- ***Auto Trader (Robô) –*** Artefato computacional capaz de predizer e operacionalizar uma ação de compra ou venda de um *asset*.

Os artefatos computacionais acima são construídos através de linguagens de programação convencionais como *C/C++*, *Java*, *Python*, entre outras. O *C/C++* é comumente utilizado para modelar operações de "*high frequency*" (alta frequência), como é o caso de operações com moedas ou operações que utilizam a estratégia de "*scalp*" (operações de curtíssima duração, poucos segundos). O *Python* é o mais utilizado devido à facilidade de prototipação. Esta P&D foi desenvolvida em *Python*, contudo foi necessário recompilar alguns *pacotes* (artefato computacional) que foram construídas em *C/C++*.

3 Proposta

3.1 Objetivo

A proposta é realizar uma *P&D* no domínio de "*Métodos Quant*" baseado em um mecanismo computacional híbrido que ampare a tomada de decisão, através da combinação de um modelo de predição e um modelo de prescrição, de tal forma que reduza o *risco* (volatilidade, ou perda) de um *portfólio* de investimento, com maior foco nos mecanismos e meia etapas, e menos foco no refino dos resultados finais.

Esta *P&D* visa combinar, os modelos de "*knowing*", *theory-driven* (orientados a teoria), com os modelos de "*learning*", data-driven (orientados a dados). Esta *P&D* não tem a pretensão de futurologia (acerto sobre o futuro), mas sim identificar e pôr em prática alguns mecanismos matemáticos e computacionais disponíveis para predição e prescrição de *asset* financeiro, e principalmente identificar o que a academia tem pesquisado para tentar superar os "*benchmarks*" (resultado referência) da atualidade.

3.2 Modelagem

3.2.1 Comum

Eu optei por utilizar uma metodologia híbrida, baseado nos ensinamentos durante a graduação e nas pesquisas realizadas. Maiores detalhes sobre a metodologia de pesquisa e de desenvolvimento podem ser vistos nos itens "*Apêndice – Metodologia de Pesquisa*" e "*Apêndice – Metodologia de Desenvolvimento*", respectivamente.

A estratégia utilizada será de "*day-trade*" (trades diários), com isso, os *asset* precisam ter "*liquidez*" (facilidade para ser comprado ou vendido a qualquer momento) principalmente para conseguir sair do mercado (com *lucro* ou *prejuízo*) em caso seja desejado. Com isso, esta *P&D* não possui foco em "*passive investing*" (trades longos que podem durar meses ou anos) nem "*high-frequency*" (trades curtos que podem durar segundos ou minutos), eu preferi um escopo intermediário. Maiores detalhes sobre o escopo desta *P&D* podem ser vistos no item "*Apêndice – Escopo*".

Esta *P&D* não tem foco cambial, inflacionário nem de correção monetária, conforme definido no item "*Apêndice - Escopo*". Com isso, não é relevante o nome da moeda utilizada no *portfólio*, mas é importante definir que todos os *asset* utilizam a mesma "*moeda*" e não existe a necessidade de realizar *câmbio* (conversões).

Para ser possível reproduzir os resultados obtidos, eu fixei a *seed* (semente para inicialização da randomização) com o valor de "*2021*" e inicializei a randomização dos principais *pacotes* com este mesmo valor. A reprodução deste experimento foi detalhada no item "*Apêndice – Reprodução*".

Eu utilizarei as métricas *MAE*, *MAPE*, *MSE* (Mean Squared Error, Error Médio Quadrado), *RMSE* (Root Mean Square Error, Erro de Raiz Quadrada Média), *RMSLE* (Root Mean Squared Logaritmic Error, Erro Logarítmico Médio Quadrático), e a *R2* (R-Squared, Error Quadrado) para analisar os erros.

3.2.1.1 Dataset

O "*input*" (entrada) de informações no modelo ocorre através de um "*dataset*" (conjunto de dados) típico do mercado financeiro, sobre a forma de "*série temporal*" (dados temporais) contendo os valores dos *preços* e *volumes* de um conjunto de *assets*, conforme tabela abaixo (*tabela 3.2.1.1*). A *série temporal* possui mais de 2 décadas de registros, com "*timeframe*" (fatia de tempo) diário. Inicialmente, o *dataset* utilizado possui uma média de *5506* instâncias, podendo variar de *asset* para *asset*, que pode ser entendido como sendo a quantidade de dias úteis em que o pregão da bolsa efetivamente funcionou entre os dias *01/Jan/2000* e *30/Nov/2021*.

Tabela 3.2.1.1

Colunas no Dataset Original		
#	**Título**	**Descrição**
01	Open	Valor de abertura do *pregão*, normalmente é valor da primeira operação.
02	Close	Valor da última operação no pregão, normalmente é o valor da última operação.
03	High	Maior valor operacionalizado durante todo o *pregão*.
04	Low	Menor valor, operacionalizado durante todo o *pregão*.
05	Adj Close	Valor de fechamento após reajustes.
06	Volume	Volume total operacionalizado no *timeframe* (fatia de tempo) em questão.

Fonte: Elaborado pelo Autor

Eu escolhi um *timeframe* diário, por ser mais simples a obtenção dos dados. Após pesquisas, eu decidi por obter os dados pelo "*Yahoo Finance*", finance.yahoo.com, por conta da confiabilidade da instituição.

3.2.1.2 Portfólio

O *portfólio* é composto pelos *assets* e *pesos* descritos na tabela abaixo (*tabela 3.2.1.2*). Em teoria, como as empresas não são do mesmo setor industrial, os *assets* possuirão baixíssima *covariância* (correlação entre dois ou mais *assets*). Contudo, durante a execução e análise dos modelos de prescrição, é importantíssimo verificar se todos os *assets* que compõem o *portfólio* realmente possuem baixa *covariância*, entre eles.

Tabela 3.2.1.2

Portfólio				
#	**Asset**	**Empresa**	**Setor**	**Peso (%)**
01	ABEV3.SA	Ambev	Bebidas	10
02	ELET3.SA	Eletrobrás	Eletricidade	10
03	EMBR3.SA	Embraer	Aeronáveis	10
04	GGBR4.SA	Gerdau	Siderurgica	10
05	ITUB3.SA	Itaú	Banco	10
06	PETR3.SA	Petrobrás	Petrolífera	10
07	RADL3.SA	Raia	Drogaria	10
08	RANI3.SA	Irani	Celulose	10
09	VALE3.SA	Vale	Mineradora	10
10	VIVT3.SA	Vivo	Telecom	10
Total				100

Fonte: Elaborada pelo autor.

Para evitar problemas de baixa *liquidez*, eu priorizarei os *assets* com maior liquidez que são comumente conhecidos como "*blue chips*" (*assets* importantes), que inclusive fazem parte do índice da "*BOVESPA*" (Bolsa de São Paulo), mais conhecido como "*IBOV*" (índice BOVESPA), onde eles são negociados.

3.2.2 Aprendizado de Máquina

As macro etapas abaixo são particulares ao modelo de *Aprendizado de Máquina,* e serão melhores detalhados no item "*5.2 Desenvolvimento*".

Pré-Processamento

- Obtenção dos Dados, em *feeder* (provedores de séries temporais) populares.
- Seleção dos atributos a serem utilizados.
- "C*lean-up*" (limpeza) dos dados.
- Transformação dos dados.
- "*Feature Engineering*" (criar, remover ou modificar *features*).
- Definição do tamanho da "*backtime windows*" (quantidade de dias passados para utilizar os dados como feature), para compor os dados que serão enviados para o modelo.
- Tratamento de *estacionariedade*.
- *Normalização* dos dados.
- "*Split*" (divisão) em conjuntos "*train*" (treino), "*valid*" (validação), "*test*" (teste), "*X*" (*features*), "*y*" (*resultado real*).

Processamento

- Execução de vários cenários combinando várias configurações do modelo em questão, gerando resultados de *métricas de erros*.
- Execução do melhor cenário, baseado na configuração do cenário que obteve o menor erro médio.

Pós-Processamento

- Visualização dos Resultados, através da geração de tabelas e gráficos.
- Análise quantitativa e qualitativa dos resultados.

O desenvolvimento foi feito na linguagem *Python*, utilizando as boas práticas de orientação a objeto e baixo acoplamento entre os artefatos computacionais. Ao final de casa passo eu gero ao menos um arquivo *"csv"* (Comma Separeted Value) de resultado da etapa em questão. Em alguns passos eu gero alguns arquivos *bin* (binários) que serão utilizados em passos posteriores, com os conteúdos abaixo.

- Algoritmo de *normalização.*
- Lista contendo o nome das colunas.
- Matriz de *mask* (mascara).
- Matriz de *explainability* (pode ser explicado)
- Importância das features

- *Plot* das *mask* (*png*)

3.2.3 Otimização de Portfólio

Eu poderia utilizar um *portfólio* simplista contendo somente *1* (um) único asset, onde este *asset* teria uma proporção para entrar no mercado e outra proporção para não entrar no mercado. A intuição deste cenário é análoga a uma carteira com *2* (dois) *assets* com os nomes fictícios de "PETROBRA-Comprado" e "PETROBRA-NãoComprado", onde o primeiro teria o *peso* (proporção) para utilizar o dinheiro comprando ações da "PETROBRAS" e o segundo teria o *peso* (proporção) para "*Mattress Money*" (guardar o dinheiro no colchão), figuradamente falando, onde o dinheiro não é gasto e continua na *carteira* sem ser utilizado. A soma dos *pesos* pode ser menor que "*100%*", por conta de resíduos fracionários e de arredondamento, mas não maior que "*100%*" do valor nominal da carteira.

Contudo, eu optei por utilizar um *portfólio* contendo *10 assets*, conforme descrito no item "*3.2.1.2 Portfólio*", que em teoria possuem baixíssima *covariância*, para verificar a teoria "*MPT*" (Modern Portfolio Theory) criada por H. Markowitz (1952) e que é utilizada até hoje.

O modelo é inicializado considerando *pesos* iguais para todos os *assets* do *portfólio*, conforme definido no item "*3.2.1.2 Portfólio*", o que não é considerado como ruim, pelo menos nesta fase incipiente do modelo. O *portfólio* é inicializado com um saldo fictício de *10000* (dez mil dinheiros), não sendo relevante a identificação da moeda.

As macro etapas abaixo são particulares ao modelo de *Otimização de Portfólio, e* serão melhores detalhadas no item "*5.2 Desenvolvimento*".

Pré-Processamento

- Identificação de baixa *covariância*, entre os *assets* que compõem o *portfólio*.
- Obtenção dos Dados, em *feeder* populares.
- Seleção dos atributos a serem utilizados.
- C*lean-up* dos dados.
- Transformação dos dados.
- Tratamento de *estacionariedade*.
- *Normalização* dos dados, transformando-os para uma faixa de valor que normalmente é entre 0 e 1.

Processamento

- Execução de vários cenários combinando várias configurações do modelo em questão, gerando resultados de *métricas de erros*.
- Execução do melhor cenário, baseado na configuração do cenário que obteve o menor erro médio.

Pós-Processamento

- Visualização dos Resultados, através da geração de tabelas e gráficos.
- Análise quantitativa e qualitativa dos resultados.

4 Revisão Literária

4.1 Aprendizado de Máquina

A predição de *asset* financeiro começou a ganhar popularidade com "*ASM*" (Artificial Stock Market), Arthur et al (1997), contudo foi com o "*LSTM*" (Long-Short Term Memory), S. Hochreiter & J. Schmidhuber (1997), que este tema avançou. O *LSTM* propunha evitar o problema de "*VGP*" (Vanishing Gradiente Problem) onde o gradiente descendente zerava. O *LSTM* resolve o "*VGP*" inicializando o "*forget gate*" com uma "*bias*" (tendência). A "*GRU*" (Gated Recurrent Unit), Cho et al (2014), é uma variação da *LSTM*, e também propõe resolver o problema de dissipação do gradiente descendente onde o gradiente reduz ao longo do tempo e passa a não contribuir para o aprendizado ou até paralisam o aprendizado. A *GRU* resolve o problema da dissipação utilizando dois "*gates*" (portões), um de "*reset*" (reinicialização) e outro de "*upgrade*" (atualização) que decidem que informações serão passadas adiante. Muitas outras variações arquiteturais da *LSTM* foram criadas, como as populares "*bi-directional*" (bidirecionais) e as "*stacked*" (empilhadas).

4.2 Otimização de Portfólio

A *Otimização de Portfólio* tem o propósito de maximizar o *retorno* e minimizar o *risco*, provendo os *pesos* (proporção) ideais para cada *asset* do *portfólio*. Os métodos de *Otimização de Portfólio* possibilitam o balanceamento do "*trade-off*" (avaliação) entre o *retorno* e o *risco*. Os *pesos* são tidos como ideais baseados em um conjunto de critérios que são *imputados* no modelo de *Otimização de Portfólio*, principalmente o tempo de execução do modelo e a quantidade de amostras.

De acordo com H. Markowitz (1952), ganhador do prêmio "*Nobel de Economia*" (1990), e criador da "*MPT*" (Modern Portfolio Theory), o *risco* está na *covariância* e não no desvio padrão. Por este motivo, Markowitz afirma que o *portfólio* deve ser composto por *asset* com baixa *covariância*, preferencialmente negativa. Empresas que atuam em diferentes áreas tendem a possuir baixa *correlação estatística* uma com a outra, resultando na redução do *risco* financeiro e na maximização do *retorno*. Segundo Markowitz, o investidor é averso ao *risco*, e por isso o *risco* é uma variável importantíssima neste contexto.

As pesquisas com modelo "*Robust*" mostram que este modelo é promissor, principalmente por terem foco em incerteza, que é exatamente o ponto central do mercado financeiro.

5 Atividades Realizadas

5.1 Pesquisa

5.1.1 Comum

Durante as minhas pesquisas, eu encontrei várias abordagens passíveis de serem utilizadas nesta *P&D*, como o "*Runge-Kutta*" que precisaria utilizar uma "*EDP*" (Equação Diferencial Parcial), para precificar os *assets*, podendo ser a "*Black-Scholes*" ou a "*NLSE*" (Non-Linear Schroding Equation). Outro modelo factível para predição, com foco na *variância* é o "*GARCH*" (Generalized Auto-regressive Conditional Heteroskedasticity), principalmente por prover bons resultados utilizando dados *heterocedásticos* e inclusive com *agrupamento de variância*, que é exatamente o caso do mercado financeiro. Contudo, por restrição de tempo não foi possível avançar com estas abordagens. Maiores detalhes sobre estes e outros modelos podem ser vistos nos itens "*Apêndice – Pesquisa – Aprendizado de Máquina*", "*Apêndice – Pesquisa – Otimização de Portfólio*", e "*Apêndice – Pesquisa – Outros Modelos*".

Maiores detalhes sobre a metodologia e o roteiro da pesquisa pode ser visto no item "*Apêndice – Metodologia de Pesquisa*".

5.1.2 Aprendizado de Máquina

Os modelos de predição que mais se aproximaram desta *P&D* foram a *LSTM* e a *GRU*. Contudo, à medida que a minha pesquisa avançou, estes dois modelos perderam prioridade e saíram do escopo desta *P&D*, principalmente por se tratarem de uma caixa-preta, onde não sabemos o que exatamente foi feito para se chegar a uma determinada predição. Maiores detalhes sobre o escopo desta *P&D* podem ser vistos no item "*Apêndice – Escopo*". Eu preferi utilizar o *Tabnet* por conta da "*explainability*" (pode ser explicado) e da "*interpretability*" (pode ser interpretado) oferecida por este modelo, diferente dos problemas comuns de caixa-preta encontrados nas *RNN*. Existem muitas perguntas sem respostas na predição de *asset*, principalmente quais *features* possuem os maiores *pesos*, maior influência. Com isso, eu estou priorizando a identificação das melhores *features* em detrimento de melhores resultados.

O *Tabnet*, criado na *Google Cloud AI*, por SO Arık, T Pfister (2019, 2020), propõe a *explainability* e a *interpretability*. O *Tabnet* utiliza a abordagem de "*stackable*" (mecanismo de *ensemble* que empilha diferentes tipos de algoritmos, gerando resultados para a segunda camada).

O *Tabnet* é muito rápido, conforme pode ser visto no item "*Apêndice – Binário do Modelo*", por conta do mecanismo de "*attention*" (mecanismo de aumento da performance) utilizado por este modelo.

Entretanto, o *Tabnet* foi originalmente construído para modelos de *classificação*, e a predição de *asset* é um problema clássico de *regressão*. Durante as minhas pesquisas, eu encontrei uma implementação do *Tabnet* para *regressão* chamada "*pytorch-tabnet*" criada por Sebastien Fishman (2021).

5.1.3 Otimização de Portfólio

Para a obtenção do "*expected return*" (retorno esperado) e da "*mean-variance*" (variância média) é necessário quantificar o *risco*. Uma maneira *ingênua* de estimar o *risco* é através da *matriz de covariância*, que retrata a *volatilidades* e as suas *correlações,* e esta é a *normalização da covariância*. O desafio consiste em obter a matriz de *covariância* e o *expected return*. Uma alternativa é realizar estimativas baseado em valores passados.

Durante as minhas pesquisas, eu identifiquei que as abordagens utilizando "*Factors*", "*Multi-Factors*" e o "*Adaptative Robust*" se mostraram muito promissoras. Contudo, por restrições de tempo não foi possível avançar com o estudo destes modelos.

5.2 Desenvolvimento

Embora esta *P&D* não tenha foco na descoberta e correção de "*gaps*" (saltos ou buracos) no *dataset*, esta tarefa é extremamente importante, principalmente quando utilizamos mais de um *asset* no modelo. É muito comum que cada *asset* possua uma ou mais datas com problemas nos dados, e ao juntar vários *assets* teremos várias datas distintas com problemas nos dados. Com isso, em um mesmo intervalo de datas, quanto maior a quantidade de *asset* envolvidos, maior será o problema nos dados.

A *estacionarização* e a *feature scaling* são fundamentais para o aumento da eficiência de modelos com base em *séries temporais*. A *estacionarização* traz estabilidade para o modelo através a fixação da *média*, *variância* e *estruturas de autocorrelação*, ao longo do tempo. A *feature scalin*g aumenta a eficiência dos modelos com base em *Otimização de Portfólio*, transformando os dados para o domínio de valores com faixas similares, através das técnicas de "*standardization*" (Padronização) e *normalization*. A *standardization* transformar os valores para se aproximarem a uma distribuição normal padrão (mi=0, sigma=1). A *normalization* transforma os valores (*preços*) para a mesma escala, que normalmente *MinMax(0,1),* que varia 0 e 1. Eu realizo simulações para identificar quais são as melhores feature scaling para cada distribuição

Na tarefa de "*feature scaling*" eu realizei o "*fit*" (aprendizado) somente no conjunto de dados "*X*" (dado real) de *train*, evitando um erro operacional muito comum de realizar o "fit" novamente no conjunto de dados "*X*" de *valid*, evitando assim o "*overfitting*" (aprendizado viciado em um input). Com exceção do sub-conjunto de dados "*X*" de *train*, todos os outros sub-conjuntos de dados foram transformados utilizando o aprendizado realizado no "*X*" de *train*. Os conjuntos de dados são os de *train*, *valid*, e o *test*. Os sub-conjuntos de dados são os "*X*" e o "*y*" (dado previsto). As operações de "inverse" (reverter para o original) sobre os subconjuntos de dados também foram realizadas com base na transformação obtida com o conjunto de dados do "*X*" de *train*.

Eu dividi o processamento em *16* passos para facilitar a análise sobre os resultados parciais de cada um dos passos. Em cada passo, eu gero arquivos *csv* do *"file system"* (Sistema de Arquivo no HD) para preservar o resultado individual de cada passo. Outra grande vantagem na preservação (persistência) dos resultados parciais e que as etapas anteriores que já estão estáveis e coerente não precisam serem reprocessadas.

De acordo com Hyndman and Koehler (2006), sobre *métricas de erros* em *séries temporais*, se os dados estão na mesma escala é preferível utilizar o "*MAE*" (Mean Absolute Error, Erro Absoluto Médio), além de ser mais simples de ser explicado.

Considerando que os valores dos preços não possuem grande diferença na ordem de grande, a métrica de erro *MSLE* são é tão útil.

Se os dados são positivos e muito maiores que zero é preferível utilizar o "*MAPE*" (Mean Absolute Percentage Erro, Erro de Porcentagem Média Absoluta), também por razões de simplicidade. O MAPE é qualificado como sendo muito bom para valores < 10%, Bom quando está entre 10% e 20%, Ok quando está entre 20 e 50%, e não OK para error maiores que 50%. Se os dados possuem escalas muito diferentes inclusive com valores próximos de zero ou negativos, ele sugere que o "*MASE*" (Mean Absolute Scaled Error, Erro Médio Absoluto Escalado) é a melhor métrica de erro para a acurácia na *predição*., contudo não justificaria utilizar o MASE porque não temos muitos gaps nem outliers. Se a distribuição de erro for gaussiana, o *RMSE* é adequado.

Eu utilizarei uma análise *ingênua* sobre os error, onde em cada *métrica* (coluna de resultado), eu atribuo um quantitativo de estrelas (pontos) para os melhores cenários (simulações). O cenário com melhor resultado, em cada *métrica* (coluna de resultado) recebe *5* estrelas para cada uma das *métricas de error*. O segundo melhor cenário recebe *4* estrelas, e assim sucessivamente até o quinto melhor cenário que recebe somente *1* estrela. Ao final existe uma coluna com o total de estrelas para cada cenário. O cenário que possuir mais estrela será o cenário ganhador teórico, conforme tabelas abaixo (*tabela 5.2a , tabela 5.2.b e tabela 5.2.c*)

A tabela abaixo mostra as primeiras colunas do arquivo de resultado (*csv*), sendo elas da esquerda para a direita: id do cenário, *asset*, método de *estacionariedade*, método de *normalização*, e as respectivas *métricas de erro*.

Tabela 5.2a

id	Asset	Stat	Scaler	MAE	MAPE	MSE	RMSE	RMSLE	R2
9	ITUB3.SA	Div	MinMaxScaler	0.04240	0.04216	0.00211			-8.23088
46	PETR3.SA	Div	MinMaxScaler	0.06193	0.06145	0.00483			-6.80452
6	ABEV3.SA	Div	MinMaxScaler	0.01257	0.01260	0.00028			-0.37114
30	GGBR4.SA	Div	MinMaxScaler	0.04298	0.04255	0.00257			-3.83125
78	VIVT3.SA	Div	MinMaxScaler	0.02055	0.02041	0.00064			-1.52653
54	RADL3.SA	Div	MinMaxScaler	0.41280	0.41300	0.17077			-491.43211
22	EMBR3.SA	Div	MinMaxScaler	0.03254	0.03226	0.00183			-2.83750
70	VALE3.SA	Div	MinMaxScaler	0.02316	0.02312	0.00093			-0.82064
14	ELET3.SA	Div	MinMaxScaler	0.03776	0.03718	0.00301			-0.64634

Fonte: Elaborada pelo autor.

A tabela abaixo mostra mais algumas colunas contendo o quantitativo de estrelas (pontos) atribuídas aos cenários de cada *asset* individualmente, imediatamente ao final do processamento dos cenários de cada *asset*. O título contém um único asterisco (*) e um único símbolo de *hash* (#), que representam os resultados individuais de um único *asset*.

Tabela 5.2b

id	MAE*	MAPE*	MSE*	MAE*	RMSLE*	R2*	MAE#	MAPE#	MSE#	RMSE#	RMSLE#	R2#	Total#
9	****	*****	****	****			4	5	4				13
46	**	****	**	**		*	2	4	2			1	9
6	****	*****	****	****			4	5	4				13
30	*****	*****	*****	*****			5	5	5				15
78	***	*****	***	***			3	5	3				11
54		***				****		3				4	7
22	***	****	***	***		*	3	4	3			1	11
70	****	*****	****	****			4	5	4				13
14	****	*****	***	****			4	5	3				12

Fonte: Elaborada pelo autor.

A tabela abaixo mostra o restante das colunas contendo o quantitativo de estrelas (pontos) atribuídas de forma global ao final da execução de todos os cenários, de todos os *asset*. A última coluna será o indicador teórico, global, dos melhores cenários. Eu analisarei esta coluna para identificar qual foi o cenário com a melhor parametrização global. O título contém dois asteriscos (**) e dois símbolos de *hash* (##), que representam os resultados globais de todos os *asset*.

Tabela 5.2c

id	MAE**	MAPE**	MSE**	RMSE**	RMSLE**	R2**	MAE##	MAPE##	MSE##	RMSE##	RMSLE##	R2##	Total##
9	****	*****	****			****	4	5	4			4	17
46	****	*****	****			****	4	5	4			4	17
6	****	*****	****			****	4	5	4			4	17
30	****	*****	****			****	4	5	4			4	17
78	****	*****	****			****	4	5	4			4	17
54	****	*****	****			****	4	5	4			4	17
22	****	*****	****			****	4	5	4			4	17
70	****	*****	****			****	4	5	4			4	17
14	****	*****	****			****	4	5	4			4	17

Fonte: Elaborada pelo autor.

Maiores detalhes sobre a metodologia de desenvolvimento podem ser vistos no item "*Apêndice – Metodologia de Desenvolvimento*".

5.2.1 Obtenção de Dados para Predição ("b_01_ds_fetch.py")

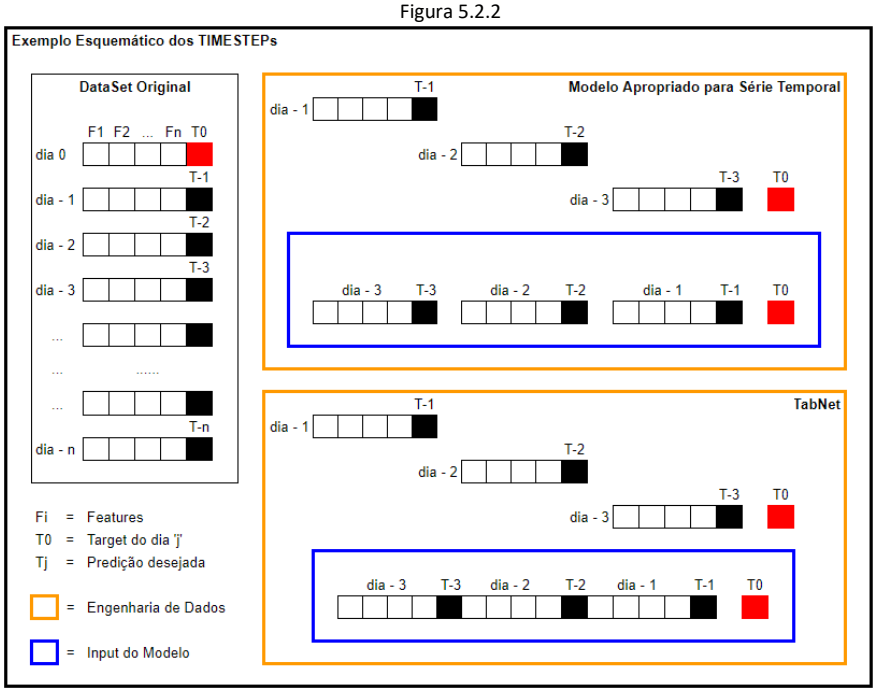
Nesta etapa, eu pego os *dataset*, de forma individual a cada *asset*, no *feeder* chamado *Yahoo Finance*. Os valores de *volume* das transações não foram considerados como uma *feature* nesta *P&D*. Contudo, ela deve ser considerada diante de cenários mais realísticos. Maiores detalhes sobre o *layout* do *dataset* pode ser visto no item "*3.2.1.1 Dataset*".

A detecção de *outliers* pode ser feita analiticamente com uma função apropriada, como é o caso da *detecção gaussiana*, ou pode ser feita olhando um gráfico de *BoxPlot*. Neste pode ser visto a distribuição e principalmente se existe algum evento além dos "*bigodes do BoxPlot*", representado por uma semireta inferior e outra superior no gráfico *BoxPlot*, área inferior (*Q1-1.5*IQR*) e superior (*Q3+1.5*IQR*) do gráfico *BoxPlot*, demarcada por uma semi-reta horizontal). Contudo, a identificação e tratamento de *outliers* não é o foco desta *P&D*. Maiores detalhes sobre o escopo desta *P&D* podem ser vistos no item "*Apêndice – Escopo*".

5.2.2 Adicionar "Features" ("b_02_feature_add.py")

As "*future engineering*" (incluir, deletar ou modificar features) são úteis aos modelos de *Aprendizado de Máquina*. Conforme mencionado no item "*5.1.2 Aprendizado de Máquina*", o *Tabnet* não foi originalmente feito para problemas de *regressão*, muito menos para problemas com base em *séries temporais*. O *pacote* "*pytorch-tabnet*" possui um componente para *regressão*, entretanto este componente não está preparado, nativamente, para consumir dados oriundos de *séries temporais*, sem as devidas transformações. Por este motivo, eu precisei fazer uma *future engineering* sobre os dados, conforme diagrama abaixo (*figura 5.2.2*).

Figura 5.2.2



Fonte: Elaborada pelo autor.

Na tabela abaixo (*tabela 5.2.2*), eu mostro um cenário utilizando uma *backtime window* de 15 dias passados. A última coluna a direita ("*Real Close*") possui os valores reais (verdadeiros) sem qualquer tipo de "*shifting*" (deslocamento). As outras colunas mais à esquerda contém os valores com *shifting*, onde o título da coluna faz menção ao nome da *feature* original, conforme item "*5.2.1 Obtenção de Dados*", acrescido do *shifting*. Este *shifting* faz menção a quantos dias existe entre a coluna corrente e a coluna "*Real Close*". Analisando a linha (penúltima) e a coluna (última) hachurada com a cor beje, a célula composta pelo cruzamento entre a linha e a coluna, com o número em vermelho, é o "*preço real*" que desejamos realizar a predição. Todos os valores da linha, a esquerda do "*Real Close*", são as novas *features* que estamos adicionando como *input* para o modelo *preditor*. O valor do dia anterior "*3.93*", na cor azul, pode ser visto do lado esquerdo do "*Real Close*" como uma nova *feature* a ser entregue para o modelo, ou imediatamente acima do "*Real Close*" como sendo um valor real de fechamento do dia anterior. Analo-

gamente, para os valores com a cor verde, e também com a cor amarela, e sucessivamente.

Tabela 5.2.2

Close 15	Close 14	Close 13	Close 12	Close 11	Close 10	Close 9	Close 8	Close 7	Close 6	Close 5	Close 4	Close 3	Close 2	Close 1	Low 1	…	High 1	…	Open 1	…	Real Close
																					4.59
														4.59							4.34
													4.59	4.34							4.39
												4.59	4.34	4.39							4.36
											4.59	4.34	4.39	4.36							4.39
										4.59	4.34	4.39	4.36	4.39							4.52
									4.59	4.34	4.39	4.36	4.39	4.52							4.37
								4.59	4.34	4.39	4.36	4.39	4.52	4.37							4.22
							4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22							4.22
						4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	…	…		…		…	4.25
					4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	…	…		…		…	4.22
				4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	4.22	…	…		…		…	4.22
			4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	4.22	4.22	…	…		…		…	4.18
		4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	4.22	4.22	4.18	…	…	…	…	…	…	4.04
	4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	4.22	4.22	4.18	4.04	…	…	…	…	…	…	3.93
4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	4.22	4.22	4.18	4.04	3.93	…	…	…	…	…	…	3.95
4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	4.22	4.22	4.18	4.04	3.93	3.95	…	…	…	…	…	…	3.91

Fonte: Elaborada pelo autor.

A tabela acima (*tabela 5.2.2*) é uma representação reduzida da realidade. O *dataset* original contém mais colunas relativas ao "*shifting*" (deslocamento) da *features* originais, conforme lista de *features* originais descritas no item "*5.2.1 Obtenção de Dados*".

5.2.3 Limpeza após Adicionar "*Features*" ("b_03_feature_del.py")

Após a *future engineering*, é comum realizar *clean-up* no *dataset* para deixar os dados prontos para as próximas etapas. Este *clean-up* pode ser realizado dentro da mesma etapa de *future engineering*. Contudo eu optei por não fazer o *clean-up* junto com a f*uture engeneering* para isolar, preservar e analisar o resultado da etapa de *future engeneering* sobre o ponto de vista da corretude funcional e técnica. A análise do resultado da *future engineering* é possível utilizando o *dataframe* no *workbook* e o arquivo *csv* do *file system* gerado durante a *future engineering* antes de qualquer tipo de *clean-up*.

No modelo *Tabnet,* durante a inclusão das *features* realizado no passo anterior (*5.2.2 Adicionar Features*), é adicionado várias linhas que não devem ser mantidas no *dataset*. Estas linhas foram úteis no passo anterior, mas devem ser retiradas antes do passo seguinte (*próximo item*). Neste passo, eu retiro as linhas contendo "*NaN*" (não é um número) que foram adicionadas durante o "*shift*" (deslocamento) da *feature* ao longo do *backtime windows*, conforme "*tabela 5.2.2"*. As *15* primeiras linhas, acima da linha hachurada com a cor bege, são efetivamente retiradas do *dataset* antes de serem enviadas para o passo seguinte. Este número *15* tem relação com o b*acktime window* que neste caso é de exatamente *15* dias.

Neste passo, eu faço um "*dump*" (escrita de bytes no HD) da lista final de colunas contidas no *dataset* após todas as manipulações de *features,* conforme *path* relativo abaixo. Este *dump* será lido por um passo mais a diante chamado "5.2.12 Resultado da Predição". Maiores informações sobre o *dump* podem ser vistas no item "*Apêndice – Metodologia de Desenvolvimento*".

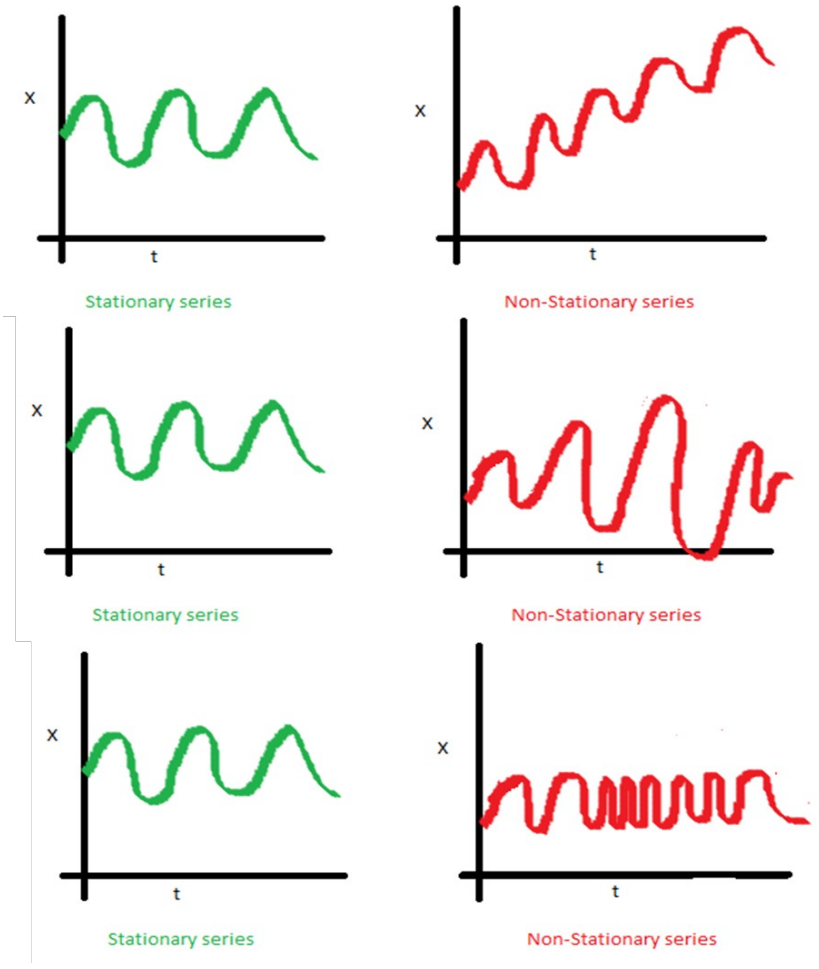
./dataset/03_feature_del/FinalColumnList.bin

5.2.4 Estacionarizar (“b_04_stationarity.py”)

Neta *P&D*, eu utilizo modelos de predição e prescrição. Independentemente do tipo do modelo, é necessário realizar a *estacionarização*. A necessidade de *estacionarização* devesse ao fato que os dados são *heterocedásticos.*

Para ilustrar, os gráficos vermelhos abaixo, *figura 5.2.4a*, não são *estacionários*, ao longo do tempo. No primeiro, embora a *variância* seja constante, a média é crescente. No segundo, a *variância* muda. No terceiro, o *spread* (movimento ou propagação) são comprimidos fazendo com que a *covariância* entre os termos não são em função do tempo, implicando em uma *variância* não constante com o tempo.

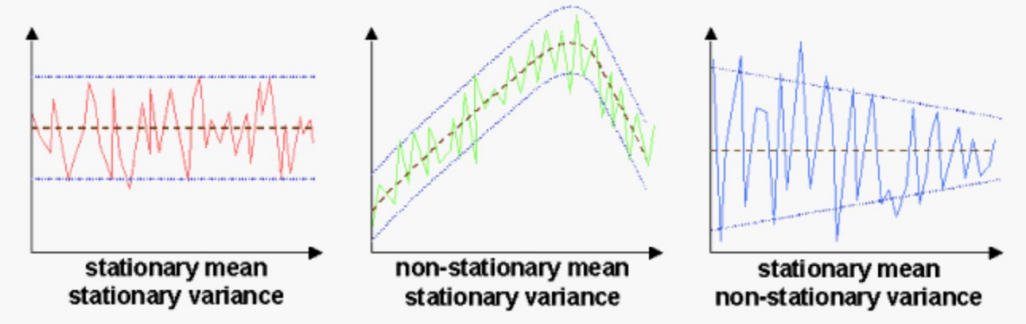
Figura 5.2.4a



Fonte: http://www.seanabu.com/2016/03/22/time-series-seasonal-ARIMA-model-in-python/ - Acesso em Nov/2021

A figura abaixo mostra 3 casos clássicos do mercado financeiro. O primeiro gráfico da esquerda, vemos o *mercado de lado*, muito comum na hora do almoço, com pouca movimentação. No segundo gráfico, vemos um classico zig-zag do mercado, onde mesmo tendo uma variação constante entre as linhas azuis, a média sobe e depois desce. No terceiro gráfico a direita, é ocorre no final do pregão nas Sextas-feiras, onde a linha pontinha representa a média constante e as linhas superiores e inferiores em azul estão afunilando, alterando a variância.

Gráfico 5.2.4.b



Fonte: https://vermasimmi1609.medium.com/ - Acesso em Nov/2021

A *estacionariedade* é uma propriedade estatística muito importante, e ela ocorre quando a *média* e a *variância* permanecem constante ao longo do tempo na *série temporal*. A intuição é que se os dados forem *estacionários*, existirá uma alta probabilidade que eles continuem *estacionários*. As ferramentas teóricas sobre os dados *estacionário* são mais simples e mais maduros se comparado com as teorias sobre os dados *não-estacionários*. Dados com uma *auto-covariância* (função de *covariância*) constante também são considerados *estacionários*.

Nem sempre é possível identificar visualmente, plotando o gráfico, a *não-estacionariedade* da *série temporal*. Nestes casos, é necessário utilizar abordagens analíticas como a "Dickey-Fuller Test" para realizar testes de checagem de *estacionariedade*. Em se tratando de mercado financeiro, não é necessário realizar checagem de *estacionariedade*, pois a *estacionariedade* é uma da principal característica estatística do mercado financeiro.

Comumente, a *estacionarização* dos dados é realizada por uma função que utiliza o preço do dia atual com o preço do dia anterior, conforme fórmula abaixo (*fórmula 5.2.4*).

Fórmula 5.2.4

$$R_d = f_s (P_d , P_{d-1})$$

Onde:
R_d = Resultado da função de estacionarização.
f_s = Função de estacionarização.
P_d = Preço do dia corrente.
P_{d-1} = Preço do dia anterior.

Fonte: Elaborada pelo autor.

Eu utilizei os mecanismos de *estacionarização* abaixo. O "*NoOne*" é o caso onde não é realizado nenhum tipo de *estacionarização*. Pode parecer um contra senso não *estacionarizar*, contudo eu julgo importante tem um cenário ingênuo para efeitos comparativos. Além da comparação, pode ocorrer algum comportamento anômalo, com os dados não *estacionarizados*, nas etapas futuras e que mereça atenção. A tabela abaixo (*tabela 5.2.4*) mostra os mecanismos de *estacionarização* que foram utilizados.

Tabela 5.2.4

Mecanismos de Estacionarização		
#	**Nome**	**Descrição**
01	NoOne	Não é realizado nenhum tipo de *estacionarização*.
02	Pct	Percentual de variação entre o preço atual e o preço anterior
03	Div	Divisão entre o preço atual e o preço anterior.
06	LogDiv	Log sobre a divisão entre o preço atual e o preço anterior.

Fonte: Elaborada pelo autor.

A métrica de erro "*MSLE*" (Mean Squared Logarithmic Error) não pode ser utilizada com os dados (*preço*) sendo *estacionarizados* pelos métodos de "*Percentage Change*" (Percentual de *returno*), "*Division*" (Divisão do atual pelo anterior) e "*Log Division*" (Log da Divisão do atual pelo anterior), pois ambos os métodos resultam direta ou indiretamente em valores negativo, o que impossibilitaria o uso da função "*log*" que não é definida como tendo números negativos no seu domínio. O *MSLE* possui os termos "*log(y_real + 1)*" e "*log(y_pred + 1)*" que impossibilita o seu uso para valores (*preço*) de "*y_real <= 1*" e "*y_pred <= 1*". Esta impossibilidade ocorrer em "*Percentage Change*", "*((atual-anterior)/anterior)*100*", quando "*atual < anterior", em "Division", "(atual / anterior)", quando "(atual / anterior) <= 1/10 ",* e em "*Log Division*", "*log(atual / anterior)*", quando "*(atual / anterior) <= 1/10 "*.

5.2.5 Limpeza após Estacionarizar (“b_05_stationarity_del.py”)

Conforme descrito no item anterior (*5.2.4 Tratar Estacionariedade*), e *fórmula 5.2.4*, o procedimento de *estacionarização* gera uma linha em branco ao início do *dataset*. Isso se deve ao fato que a “*função estacionarizadora*” (função responsável por *estacionarizar* a *série temporal*) necessita do preço do dia anterior, e como não existe dia anterior para a primeira linha, esta primeira linha não poderá ter um resultado da “*função estacionarizadora*”. Por isso, o tratamento de *NaN* é fundamental após a operação de *estacionarizar*, pois esta operação utiliza valores passados. Neste caso sempre ocorrerá pelo menos uma ocorrência de *NaN* ao início (primeiro registro) da *série temporal*, por não possuir valores anteriores ao primeiro registro (evento).

Pelos mesmos motivos já expostos no item “5.2.3 Limpeza após Adiconar Features”, eu optei por não fazer o *clean-up* do *dataset* dentro do item anterior (*5.2.4 Tratar Estacionariedade*). Desta forma, eu isolo e preservo os dados recém *estacionarizados* para realizar as devidas conferências. Somente a primeira linha é efetivamente retirada do *dataset* antes de serem enviadas para o passo seguinte.

5.2.6 Particionar em “Train”, “Valid” e “Test” (“b_06_slice_tvt.py”)

O funcionamento dos modelos de Aprendizado de Máquina consiste basicamente em “*Preparação dos Dados*”, “*Treinar o Modelo*”, e “*Realizar a predição*”. O “*Treino do Modelo*” utiliza duas fatias do *dataset* que comumente chamamos de *train* e *valid*. O “*Treino do Modelo*” utiliza uma terceira fatia do *dataset* comumente chamada de *test*. Normalmente reservamos entre *80%* e *90%* do *dataset* para o *train* e o restante dividimos em partes iguais para o *valid* e para o *test*. Contudo, estes valores não devem ser rígidos e o ideal é que se realize testes sobre a distribuição existente de modo a descobrir a proporcionalidade ideal para a distribuição em questão. A busca pela proporcionalidade ideal não é o foco desta *P&D*.

Nesta etapa de “*Preparação dos Dados*”, para os processamentos desta *P&D*, eu utilizei a proporção de *80, 10,* e *10* para o *train, valid* e *test.* Havendo necessidade, esta proporção pode ser modificada nos parâmetros na “*dataclass*” (classe somente de dados) chamada “*ComSliceCfg*” no arquivo de configuração (*./a_com/com_0_config.py*), conforme exemplo do código abaixo (*código 5.2.6*).

Código 5.2.6

```
@dataclass
class ComSliceCfg():
 """ Dataset splitting. """
 # % over the original dataset, to be used to train, valid and test.
 F_PERCENTUAL_TRAIN_DATASET = 0.80
 F_PERCENTUAL_VALID_DATASET = 0.10
 F_PERCENTUAL_TEST_DATASET  = 0.10
```

Fonte: Elaborada pelo autor.

5.2.7 Remover a Data (“b_07_date_del.py”)

Conforme descrito no item “*Apêndice – Escopo*”, esta *P&D* não foca em qualquer análise sobre a data, ou *feature engineering* sobre as datas. Contudo, em um modelo menos *ingênuo*, é importante utilizar à distância de datas como uma nova *feature* para o modelo predição. Esta nova *feature* conteria a quantidade de dias que o dia corrente estaria de alguma data importante como feriado, final de semana, final de mês, publicação de indicadores (juros, desemprego, câmbio, aceleração industrial, exportação, entre outros). Esta distância teria valores negativo e positivos, se a data fosse uma data passada ou futura, com um limite absoluto de *15* dias por entender que o mercado é influenciado por dias recentes, onde o último dia tem maior peso de influência.

Antes de realizar o passo seguinte, de *normalização*, é necessário retirar a coluna “*Date*” (data). Esta retirada é importante evitar a inclusão de “*noise*” (ruídos) nas *features* do modelo predição. Eu poderia *normalizar* a data com valores “*float*” (ponto flutuante) ou “*int*”, na mesma faixa dos valores dos preços dos *assets*, contudo até mesmo os *asset* já possuem valores com *escala* (ordem de grandeza) diferente. Como é muito fácil colocar *noise* e muito difícil *retirar*, e a data sem o cálculo da distância relativa não traria benefício, eu preferir não utilizar a data *normalizada*.

Contudo, é importante preservar esta coluna “*date*” pois ela será necessária em todos os outros passos após a execução do modelo predição, principalmente para plotar gráficos contendo as datas. Neste passo, eu gravo dois arquivos no diretório “./dataset/*07_date_del*”, um contendo somente a coluna *date* para futuro uso, e o outro arquivo contendo as *features* sobre a forma de valore dos preços a serem *normalizados* no próximo passo.

Em relação ao *dataset* original, houve uma perda de *3,66%* dos dados por conta de “*gaps*” (ausência de dado), *NaN*, *estacionarização*, e “*feature scaling*” (escalamento/transformação para faixa de valores parecidas).

5.2.8 Normalizar (“b_08_scale.py”)

Os modelos de Aprendizado de Máquina costumam performar melhor com *inputs* dentro de uma faixa de valores, preferencialmente pequenos. Por este motivo, eu gerei novos *dataset* contendo *normalizações* distintas para serem consumido pelos próximos passos, conforme descrito na tabela abaixo (*tabela 5.2.8*).

Tabela 5.2.8

Mecanismos de Normalização		
#	**Nome**	**Descrição**
01	NoOne	Não é realizado nenhum tipo de *normalização*.
02	MinMax	Os valores são convertidos para resultados entre 0 (zero) e 1 (um), também conhecido como Min-Max(0,1).
03	MaxAbs	Maior valor operacionalizado durante todo o pregão.
04	Standard	Menor valor, operacionalizado durante todo o pregão.
05	Robust	Valor de fechamento após a realização de reajustes. Os reajustes podem ser por motivos variados.
04	Normalizer	Menor valor, operacionalizado durante todo o pregão.
05	Quantile Transformer	Valor de fechando após a realização de reajustes. Os reajustes podem ser por motivos variados.
06	Power Transformer	Volume total operacionalizado no *timeframe* em questão.

Fonte: Elaborada pelo autor.

5.2.9 Particionar em “X” e “y” (“b_09_slice_xy.py”)

Conforme já descrito no item “*5.2.6 Particionar em Train, Valid e Test*”, os modelos de Aprendizado de Máquina realizam *train* e *valid*, em ambos os casos o modelo realiza checagem dos valores “*reais*” (verdadeiros), descritos como sendo “*Real Close*” na tabela abaixo (*tabela 5.2.9*), com os valores encontrados durante a predição. Por isso é necessário informar ao modelo quais são as colunas de valores a serem computados, conhecidos como *features*, e qual é a coluna com o valor *real*. As colunas de *features* são comumente chamadas de conjunto “*X*”, em maiúsculo, e a coluna contendo os valores *reais* são chamados de conjunto “y”, em minúsculo.

Nesta *P&D*, eu utilizo o valor de *close* como sendo a informação que o modelo deve realizar a predição, que na tabela abaixo (*tabela 5.2.9*) é a última coluna com o título “*Real Close*”. Ou seja, o modelo tem a meta de tentar predizer valor o mais próximo possível do “*Real Close*”. Todas as outras colunas são tratadas como *features* a serem tratadas como *input* para o modelo *preditor*.

Tabela 5.2.9

Close 15	Close 14	Close 13	Close 12	Close 11	Close 10	Close 9	Close 8	Close 7	Close 6	Close 5	Close 4	Close 3	Close 2	Close 1	Low 1	…	High 1	…	Open 1	…	Real Close
4.59	4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	4.22	4.22	4.18	4.04	3.93	…	…	…	…	…	…	3.95
4.34	4.39	4.36	4.39	4.52	4.37	4.22	4.22	4.25	4.22	4.22	4.18	4.04	3.93	3.95	…	…	…	…	…	…	3.91

Fonte: Elaborada pelo autor.

5.2.10 Predição Combinada (“c_10_pred_tabnet_all.py”)

Neste passo eu combino vários cenários baseados nas possíveis configurações particulares ao meu modelo e também ao modelo *Tabnet*, conforme tabela abaixo (*tabela 5.2.10*).

Tabela 5.2.10

Parâmetros do Modelo Tabnet			
Nome	**Descrição**	**Domínio**	**Qt**
V_RAW_ASSET_PRD	Lista de *assets* financeiros.	['ABEV3.SA', 'ELET3.SA', 'EMBR3.SA', 'GGBR4.SA', 'ITUB3.SA', 'PETR3.SA', 'RADL3.SA', 'RANI3.SA', 'VALE3.SA', 'VIVT3.SA']	10
PriceStationarity	Mecanismos de *estacionarização*.	[NoOne, Pct, Div, LogDiv]	4
ScalerModel	Mecanismos de *normalização*.	[NoOne, MinMaxScaller, MaxAbsScaler, StandardScaler, RobustScaler, Normalizer, QuantileTransformer, PowerTransformer]	8
S_DT_BEGIN	Data Inicial da Série Histórica	03/Jan/2000	n/a
S_DT_END	Data Final da Série Histórica	30/Nov/2021	n/a
F_PERCENTUAL_TRAIN_DATASET	Percentual de *train* sobre o *dataset*.	0.80	n/a
F_PERCENTUAL_VALID_DATASET	Percentual de *valid* sobre o *dataset*.	0.10	n/a
F_PERCENTUAL_TEST_DATASET	Percentual de *test* sobre o *dataset*.	0.10	n/a
V_I_MAX_EPOCH	Número máximo de *epoch*.	[10000]	1
V_I_BATCH_SIZE	Tamanho do *batch* de treino	[2048]	1
V_I_VIRTUAL_BATCH_SIZE	Tamanho do *batch* para o “*Ghost Batch Normalization*”.	[256]	1
V_I_BACKTIME_WINDOW	Janela de tempos passados.	[15]	1
Total de Cenários			320

Fonte: Elaborada pelo autor.

O “*Ghost Batch Normalization*” é uma variação do *Batch Normalization*. O primeiro coloca ruído em uma pequena parcela dos dados. O segundo faz a *normalization*, que normalmente é a *MinMax(0,1)*.

As opções de *PriceStationarity* são melhores detalhadas no item “*5.2.4 Estacionarizar*”, as opções de *ScalerModel* são melhores detalhadas no item “*5.2.7 Normalizar*”.

A configuração “V_I_BACKTIME_WINDOWS” é particular ao meu modelo onde é possível modificar de forma 100% automática, através da modificação deste parâmetro, como as *features* serão geradas. As configurações “*V_I_MAX_EPOCH*”, “*V_I_BATCH_SIZE*”, e “*V_I_VIRTUAL_BATCH_SIZE*” são particulares ao modelo Tabnet. Todos estes quatro parâmetros são do tipo “*list*” (vetor) para ser possível testar uma grande variedade de *tests*. Esta grande variedade de *test* eu chamo de “*test mode*” (execução de teste) e acaba processando muito mais cenários que o “*debug mode*” (execução de depuração) e o “*production mode*” (execução do melhor cenário). O “*debug*

mode" testa pequenos cenários com focos específicos. O "*production mode*" testa o melhor cenário baseado nos resultados das *métricas de erros* obtidas no "*test mode*".

Durante os testes, para trabalhar com menos volume de dados e também menos cenários a serem processados, eu configurei a variável de "*B_DEBUG*" como sendo "*True*", conforme descrito no item "*Apêndice – Reprodução*". Outros detalhes sobre configuração e reprodução também podem ser melhor entendidos no mesmo item "*Apêndice – Reprodução*".

Neste passo, eu não faço um *dump* do algoritmo utilizado, por ser muito grande. Contudo, eu obtenho do tamanho do algoritmo para ser possível tirar a média do tamanho, conforme mencionado no item "*Apêndice – Binário do Modelo*".

5.2.11 Predição Ajustada ("c_11_pred_tabnet_best.py")

Para a execução deste passo, é necessário informar as configurações ideais para o modelo. Eu poderia setar (configurar) estas configurações de forma automática, através da extração dos resultados do passo anterior, que foram gravamos em um arquivo do tipo *csv,* conforme definido ao início do item "*5.2 Desenvolvimento*". Entretanto, eu preferi não automatizar este passo por entender que para iniciar o passo atual é necessário um "*human reasoning*" (raciocínio humano) sobre os resultados das *métricas de erros*.

Eu utilizei a configuração do melhor resultado do passo anterior, "*5.2.10 Predição Combinada*". Abaixo, na "*tabela 5.2.11*", pode ser visto o que foi efetivamente modificado para realizar o passo atual.

Tabela 5.2.11

Melhores Parâmetros do Modelo Tabnet			
Nome	**Descrição**	**Domínio**	**Qt**
PriceStationarity	Mecanismos de *estacionarização*.	LogDiv	1
ScalerModel	Mecanismos de *normalização*.	NoOne	1

Fonte: Elaborada pelo autor.

Neste passo, eu faço um *dump* do resultado das melhores *features* que foram identificadas pelo Tabnet. Este *dump* será lido pelo passo seguinte, "*5.2.12 Resultado da Predição*". Maiores informações sobre o *dump* podem ser vistas no item "*Apêndice – Metodologia de Desenvolvimento*".

5.2.12 Resultado da Predição (“c_12_pred_tabnet_result.py”)

Como era de se esperar, cada *asset* obteve resultados de *métricas de erros* distintos, muito provavelmente por se tratar de distribuições diferentes. Mesmo analisando um *asset* isoladamente, já observamos que este possui distribuições diferente ao longo do tempo devido a característica de “*heterocedasticidade*”.

O *Tabnet* é muito rápido, conforme pode ser visto no item “*Apêndice – Binário do Modelo*”, por conta do mecanismo de “*attention*” (mecanismo de aumento da performance) utilizado por este modelo. Conforme já mencionado no item “*5.1.2 Aprendizado de Máquina*”.

Dado a importância do tempo de execução e o tamanho do binário, por conta dos serviços “*lambda*” (em *cloud*, na *AWS*, é um serviço atômico que se assemelha a um evento) comumente utilizados em ambiente “*cloud*” (Ambiente para hospedagem com especialização em infraestrutura de *HW* e *SW*), eu detalhei o tempo e tamanho dos binários no item “*Apêndice – Binário do Modelo*”.

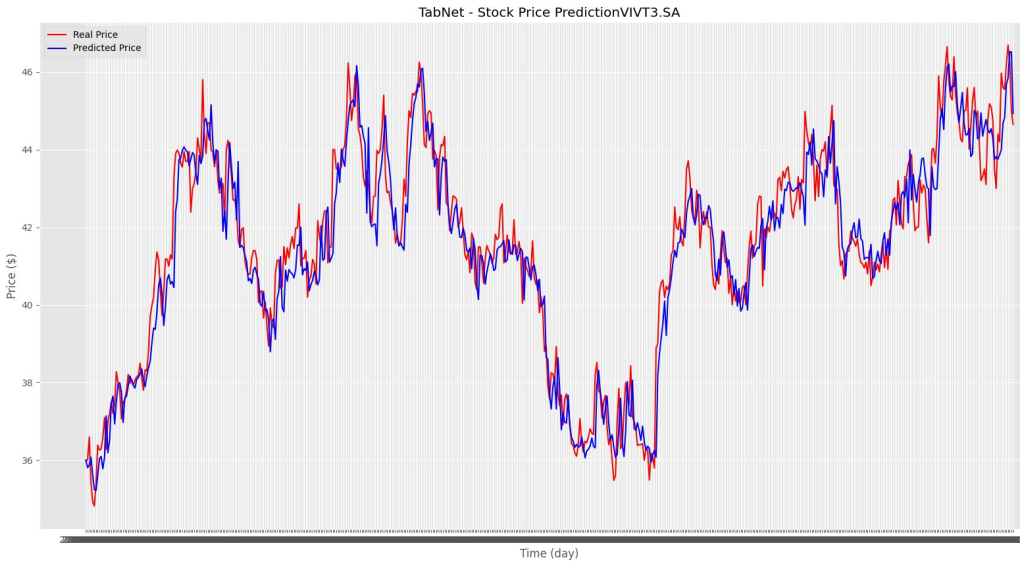
- Algoritmo de *normalização*
- Lista contendo o nome das colunas
- Matriz de mask
- Matriz de *explainability*
- Importância das features
- *Plot* das *mask* (*png*)
- Plot do gráfico de linhas contendo os valores real e os *preditos*.

Os gráficos foram gerados utilizando os dados de *test*, dos últimos *2* (dois) anos, com um *backtime windows* de *15* (quinze) dias. Com isso, o que inicialmente eram somente *5* (cinco) *features*, *open*, *close*, *high*, *low*, e *adj_close*, passaram a ser 5 x 15 (cinco vezes cinco), totalizando *75* (setenta e cinco) *features*, conforme já definidor no item “*5.2.2 Adicionar Features*”.

Analisando os gráficos gerados pelo *TabNet*, eu puder confirmar que o modelo não funciona bem para todos com os mesmos parâmetros. Cada *asset* possui a sua própria distribuição e os modelos de *Aprendizado de Máquina* precisam se especializar em cada contexto, com os devidos cuidados para não ocorrer *overfitting*. Considerando a pandemia é esperado que o modelo erre muito pois ele não tive a oportunidade de aprender com este tipo de fato histórico que gera movimentos atípicos e imprevistos adicionais aos que o mercado já possui.

No gráfico abaixo, *gráfico 5.2.12a*, vemos que o modelo conseguiu acompanhar os movimento do *asset* VIVT3 no mercado.

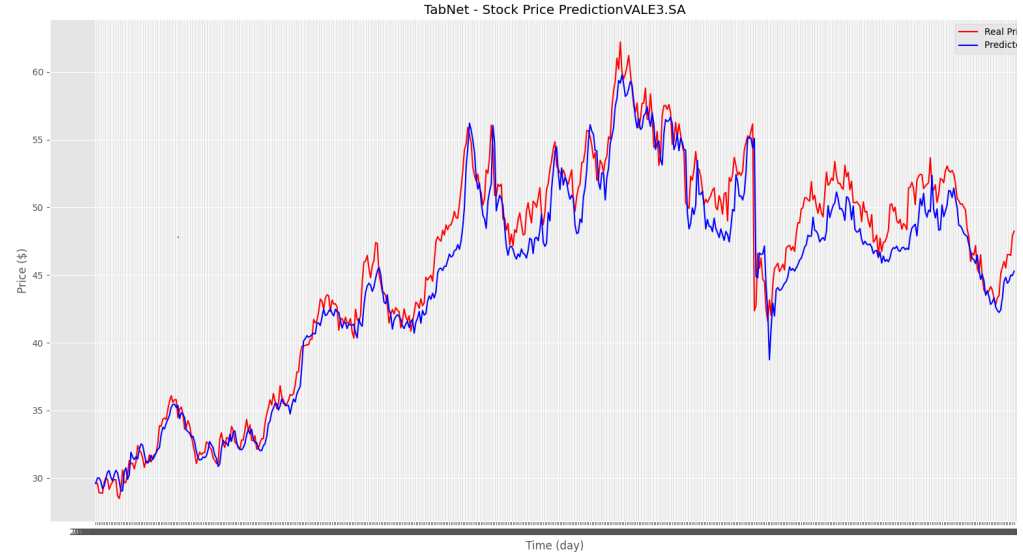
Gráfico 5.2.12a



Fonte: Elaborada pelo autor.

No gráfico abaixo, *gráfico 5.2.12b*, vemos que o modelo acompanhou mas teve dificuldade de manter os valores próximo do *asset* VALE3. Esta dificuldade já começou em ¼ do período em questão. No meio do período, o modelo voltou a se aproxima, mas logo depois se distanciou novamente.

Gráfico 5.2.12b

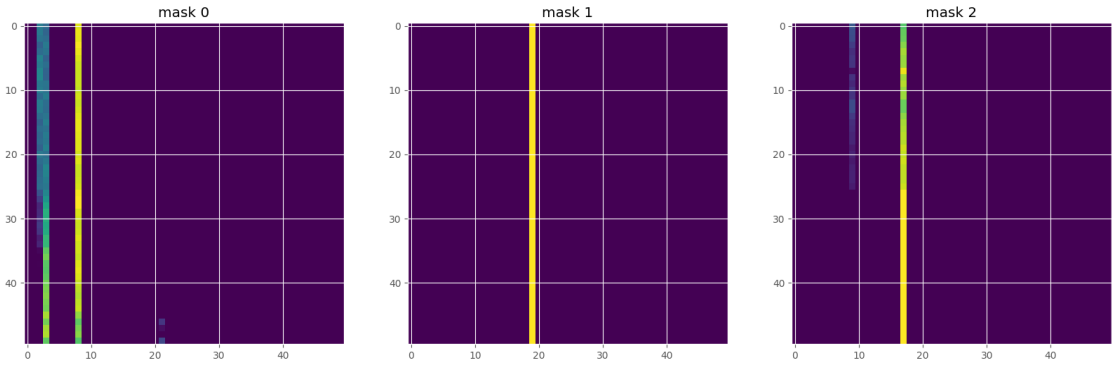


Fonte: Elaborada pelo autor.

Abaixo, gráfico 5.2.12c, vemos *3* "*mask*" (mascara) de importância das *features*. A partir delas é possível identificar qual *feature* pesou mais durante o aprendizado. Este artifício é fundamental para desenvolver modelos mais eficientes.

Na *mask* da esquerda, vemos que as *features* do dia anterior, que são de 0 até 4, pesaram no aprendizado, o que era de se esperar pois sabemos que o mercado financeiro se influencia por informações recentes.

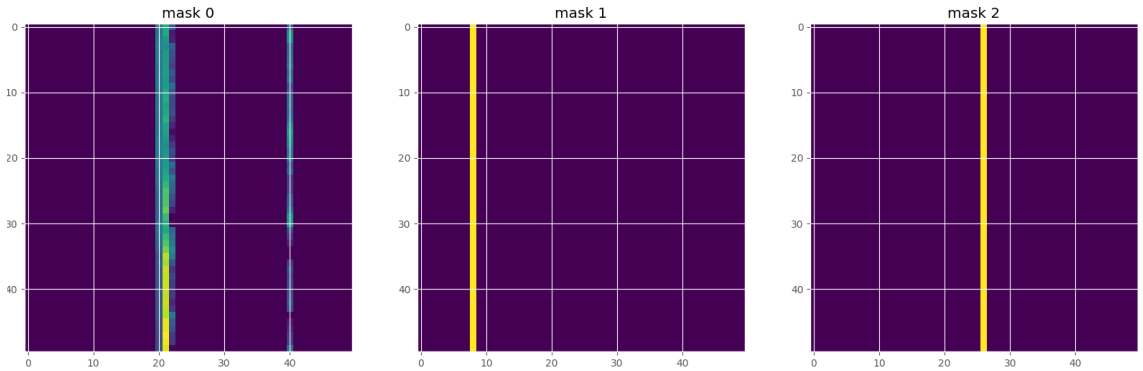
Gráfico 5.2.12c



Fonte: Elaborada pelo autor.

Na *mask* da esquerda, vemos que as *features* de número 20, 21 e 22 pesaram no aprendizado, com ruído, diferente das máscaras do meio e da direita onde as corres estão fortes e contínuas.

Gráfico 5.2.12d



Fonte: Elaborada pelo autor.

5.2.13 Obtenção de Dados para Prescrição ("d_13_ds_concat.py")

Os dados que serão utilizados do processo de prescrição serão obtidos e tratados neste passo. Este passo é análogo ao item "*5.2.1 Obtenção de Dados para Predição*". Neste passo, os dados já estão prontos para serem consumidos, pois já foram *estacionarizados* e normalizados nos passos anteriores, ainda assim este passo é importante que o *dataset* de *input* do processo de prescrição estão de acordo com o esperado.

5.2.14 Prescrição Combinada ("e_14_pres_markowitz_all.py")

Neste passo eu combino vários cenários baseados nas possíveis configurações particulares ao meu modelo e também a abordagem de *Markowitz*, conforme tabela abaixo, *tabela 5.2.14.*

Tabela 5.2.14

Parâmetros do Modelo Markowitz			
Nome	**Descrição**	**Domínio**	**Qt**
V_RAW_ASSET_PRD	Lista de *assets* financeiros.	*[ABEV3, ELET3,EMBR3,GGBR4,ITUB3,PETR3,RADL3,RANI3,VALE3,VIVT3]*	10
PriceStationarity	Mecanismos de *estacionarização*.	*[NoOne, Pct, Div, LogDiv]*	4
ScalerModel	Mecanismos de *normalização*.	*[NoOne, MinMaxScaller, MaxAbsScaler, StandardScaler, RobustScaler, Normalizer, QuantileTransformer, PowerTransformer]*	8
S_DT_BEGIN	Data Inicial da Série Histórica	*03/Jan/2000*	n/a
S_DT_END	Data Final da Série Histórica	*30/Nov/2021*	n/a
F_PERCENTUAL_TRAIN_DATASET	Percentual de *train* sobre o *dataset*.	*0.80*	n/a
F_PERCENTUAL_VALID_DATASET	Percentual de *valid* sobre o *dataset*.	*0.10*	n/a
F_PERCENTUAL_TEST_DATASET	Percentual de *test* sobre o *dataset*.	*0.10*	n/a
OPT_METHOD	Métodos de optimização de *portfólio*.	*[MVO-Mean-Variance Optimization,HRP-Hierarchical-Risk-Parity]*	3
RISK_METHOD	Métodos para calcular o *risco*.	*[sample_cov,semicovariance,exp_cov,ledoit_wolf,ledoit_wolf_constant_variance,ledoit_wolf_single_factor,ledoit_wolf_constant_correlation,oracle_approximating]*	8
RETURN_METHOD	Método para calcular o *retorno*.	*[mean_historical_return,ema_historical_return,capm_return]*	3
EFFICIENT_FRONTIER	Tipo de função objetivo.	*[max_sharpe,min_volatility]*	2
Total de Cenários			46080

Fonte: Elaborada pelo autor.

A comunidade financeira realiza *Otimizações de Portfólio* com projeções anuais para o *risco* e *retorno*. Os principais *pacotes* refletem este padrão de anualizar os modelos. Como esta *P&D* tem foco no "*day trade*" (operações diárias), eu utilizei uma abordagem ingênua de dividir os resultados (*retorno*) pela quantidade média padrão de "*business-day*" (dias comerciais), que em média é de *252* dias. Entretanto, para uma *P&D* com foco no "*tunning*" (ajuste fino) sobre a *retorno*, é necessário utilizar uma abordagem mais acurada como o "*Bootstrap*" (método para estimativa de amostragem).

Embora eu não tenha utilizado o "*solver*" (resolvedor de modelos de *Otimização*) "*Gurobi*" (um dos melhores *solvers* da atualidade), os *pacotes* que eu utilizei fazem uso do *Gurobi* direta ou indiretamente (através do *facade* sobre outros *pacotes*).

O modelo é inicializado considerando *pesos* iguais para todos os *asset* do *portfólio*, o que não é considerado como ruim, pelo menos nesta fase incipiente do modelo.

Para a obtenção do "*Log(Return)*" (log do retorno), não é possível utilizar o método "*pct_change( )*" (porcentagem sobre a mudança de preço), que é nativo ao "*pandas*" (*pacote* de *dataset*), porque este resulta em valores negativos quando os preços caem, o que inviabiliza de aplicar a função "*log( )*" sobre estes retornos negativos. Por este motivo, eu considerei o *retorno* como sendo o preço atual dividido pelo preço anterior, para evitar que resulte em "*- inf* " (menos infinito) quando o preço não variar, e resulte em "*NaN*" (*Not a Number*, não é um número) quanto o preço cair.

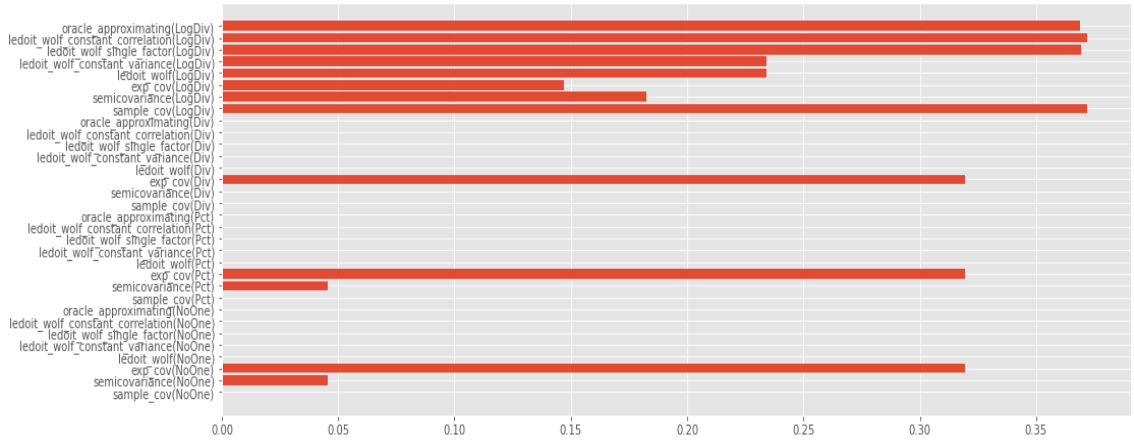
Além do modelo clássico, que utilizada os valores históricos, existem outros modelos mais eficientes para *Otimização de Portfólio* como o *Black Litterman e o Factor Model,* que por restrições de tempo não foi possível avançar com estes modelos. Além da *MaxSharpe* e da *MinVolatility*, também podemos citar a *MaxRet* e a Utility como função objetivo que eu não utilizei nesta *P&D*.

Em alguns cenários, temos o método *semi-covariance* zerando a matriz de covariância, fazendo com que o *SharpeRatio* resulte em " *- inf* " (menos infinito). Estes valores ocorrem nos cenários abaixo.

- Valor = [*Log* (*Preço*), *Log*(retorno)]
- Método de *Risco* = [*semi-covariance*]
- Método de *Retorno* = [*mean historical return, semi-covariance, ema historical return, capm return*]

Na tabela abaixo (*tabela 5.2.14a*) temos o resultado das simulações dos métodos para estimar o *risco* do *portfólio*, combinados com o método de *estacionarização*.

Gráfico 5.2.14a



Fonte: Elaborada pelo autor.

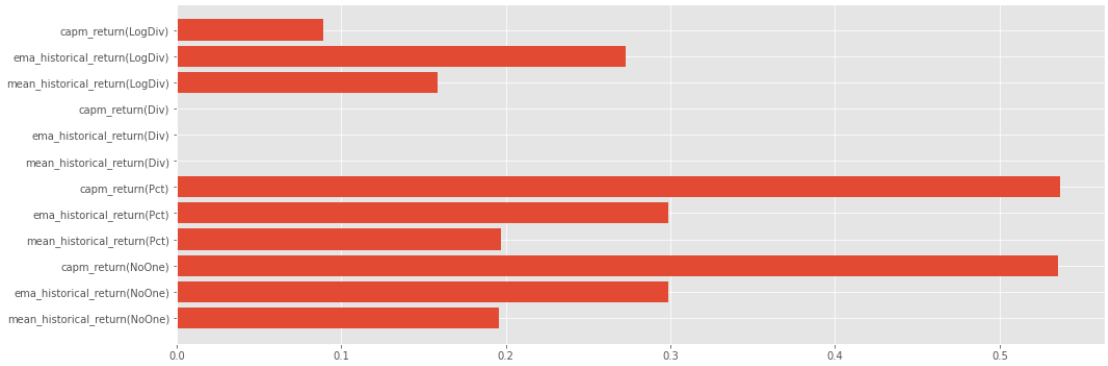
Os valores na mesma origem do gráfico são 810.5801003508005, 0.045700810390097234, 0.31930128551738507, 809.7904265484501, 809.7904265484501, 809.925773905582, 810.5801003508005, 809.7904265484501, 810.5799129873581, 0.045570227055563496, 0.3193072151007251, 809.7907578321974, 809.7907578321974, 809.9255873214886, 810.5799129873581, 809.7907578321972, 810.5797044840521, 0.0, 0.31930161693587145, 810.9567665755706, 810.9567665755706, 811.0414654591974, 810.5797044840518, 810.9567665755704, 0.3720703624996363, 0.18234285179433055, 0.1469448960097794, 0.23405863647907052, 0.23405863647907052, 0.3693528764093082, 0.3720703624996363, 0.3689349889906748.

Nas simulações acima, temos o *semi-coavariance* como método ganhador para estimar o *risco* do *portfólio*, em conjunto com o método *PriceStationarity.Pct* de *estacionarização*. Este método calcula o percentual de variação do *preço atual* com o *preço anterior*. Abaixo, temos detalhes do cenário ganhador.

- Error (*MAE*) = *0.045570227055563496*
- Método de *Risco* = *semi-coavariance*
- Método de Estacionarização = *PriceStationarity.Pct*

Na tabela abaixo (*tabela 5.2.14b*) temos o resultado das simulações dos métodos para estimar o *retorno* do *portfólio*, combinados com o método de *estacionarização*.

Gráfico 5.2.14b



Fonte: Elaborada pelo autor.

Os valores na mesma origem do gráfico são 0.195714560967189, 0.2987281290788976, 0.5352099802655819, 0.19719237661647612, 0.29893628670897954, 0.5368486426607836, 0.0, 9.582770734338586e+74, 0.0, 0.1586566053904325, 0.2730783629523935, 0.08898578817295373.

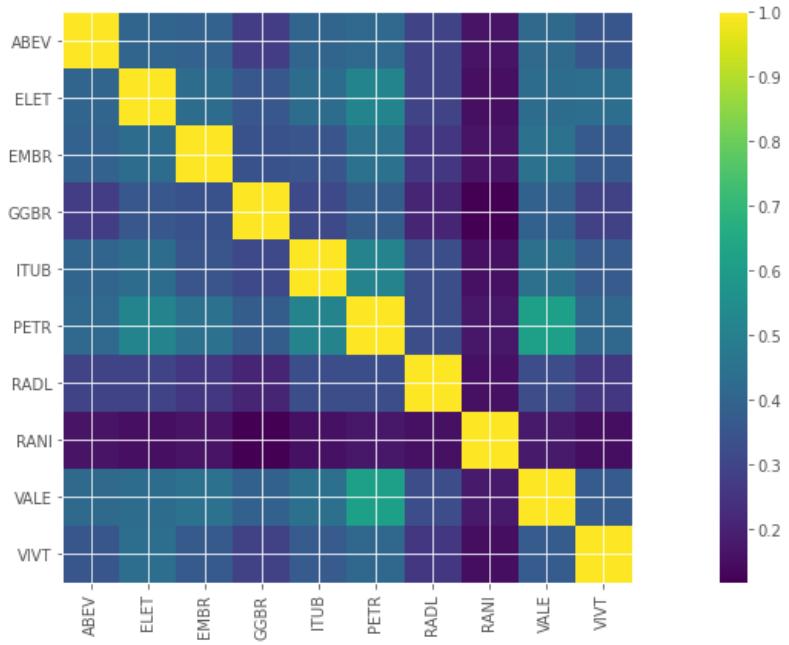
Nas simulações acima, temos o *capm_return* como método ganhador para estimar o *retorno* do *portfólio*, em conjunto com o método *PriceStationarity.LogDiv* de *estacionarização*. Este método calcula o *log( )* da divisão do *preço atual* com o *preço anterior*. Abaixo, temos detalhes do cenário ganhador.

Abaixo, temos o cenário que obteve melhores resultados nas simulações acima.

- Error (*MAE*) = *0.08898578817295373*
- Método de *Risco* = *capm_return*
- Método de Estacionarização = *PriceStationarity.LogDiv*

Abaixo, *gráfico 5.2.14c*, temos a matriz de covariância, gerada com os *inputs* de *train* , onde podemos notar a baixa *correlação* da empresa "*Celulose IRANI*" com as outras empresas, o que é racional considerando que nenhuma outra empresa no *portfólio* é do ramo de commodities agrícolas. De acordo com Markowitz, este cenário é o ideal para mitigar o *risco* do *portfólio*, onde as empresas do *portfólio* possuem baixa *correlação*.

Gráfico 5.2.14c



Fonte: Elaborada pelo autor.

5.2.15 Prescrição Ajustada (“e_15_pres_markowitz_best”)

Eu utilizei a configuração do melhor resultado do passo anterior, “*5.2.14 Prescrição Combinada*”. Abaixo, na “*tabela 5.2.15*”, pode ser visto o que foi efetivamente modificado para realizar o passo atual.

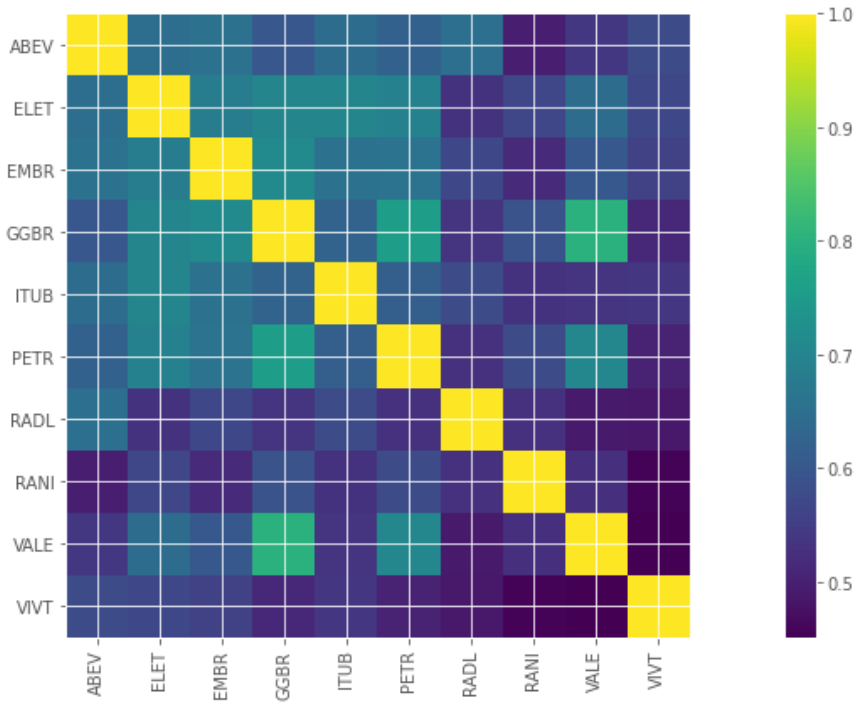
Tabela 5.2.15

Parâmetros do Modelo Markowitz			
Nome	**Descrição**	**Domínio**	**Qt**
PriceStationarity	Mecanismos de *estacionarização*.	*Pct*	1
RISK_METHOD	Métodos para calcular o *risco*.	*semicovariance*	1

Fonte: Elaborada pelo autor.

Abaixo temos a matriz de covariância, gerada com os *inputs* de *test,* onde podemos notar o aumento da correlação da empresa “*Celulose IRANI*” com as outras empresas, e também o aumento da correlação entre as empresas no canto superior esquerdo.

Gráfico 5.2.15



Fonte: Elaborada pelo autor.

5.2.16 Resultado da Prescrição (“e_16_pres_markowitz_result.py”)

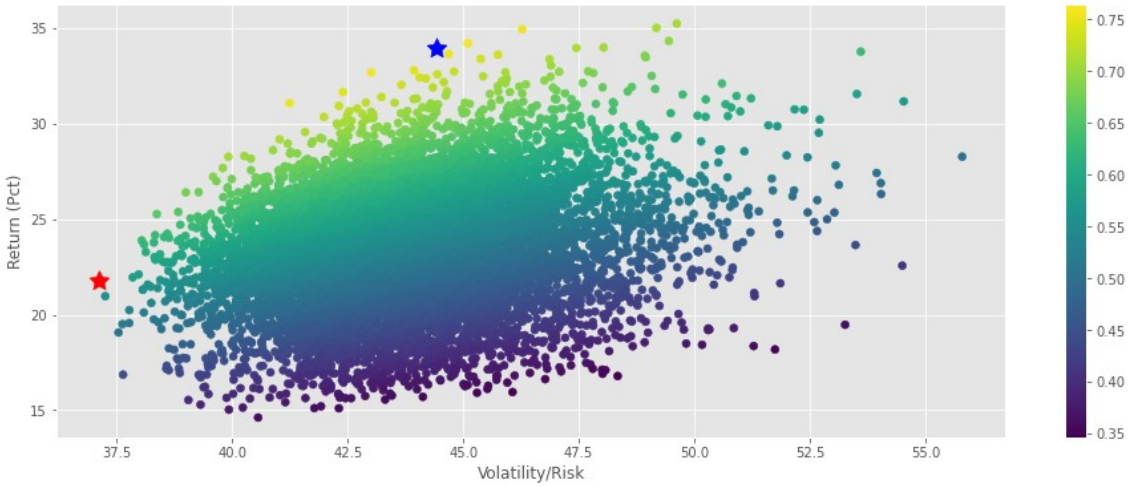
Como era de se esperar o modelo que performou melhor com os *inputs* de *train* não performou tão bem com os inputs de *test.* Isso ocorre porque a *média* e a *variância* da distribuição não são constantes.

Durante a escolha das empresas para compor o *portfólio*, eu selecionei empresas de indústrias distintas. Contudo isso não foi suficiente para obter um *portfólio* com baixa *correlação*. O preço também sofre influência por variáveis *exógenas* como índice de consumo, industrialização, entre outros. Mesmo que consigamos identificar *correlação* entre empresa, é muito difícil identificar a *causalidade*.

Quando optamos por utilizar um modelo baseado no *retorno* e não obtemos bons resultados, uma alternativa é utilizar modelos com base no *risco* como é o caso do “*Risk Parity*”, pois os modelos baseado em *risco* são mais estável que os baseados em *retorno*.

No gráfico abaixo, *gráfico 5.2.16a*, vemos a *Efficient Frontier* (Fronteira Eficiente) desenvolvida por Markowitz, que tem o conceito de diversificar o *portfólio* para reduzir o *risco*. O gráfico consiste em um *scatter plot* (gráfico de pontinhos), onde eu plotei com *10000* (dez mil) pontinhos, onde cada pontinho consiste em uma configuração (pesos) do *portfólio*. A estrela azul mostra a configuração máxima de *MaxSharpeRatio*, deste *portfólio*, que é um índice que busca equilibrar o retorno e o *risco*. Um investidor moderado, nem arrojado nem conservador, optaria pelo *sharpe ratio*. A estrela vermelha é de uma configuração do portfólio em questão para um investidor conservador que não quer correr risco, pois este indicador é para risco mínimo, *MinRisk*. Cada pontinho, que equivale a uma configura do *portfólio*, consiste em uma *chave-valor* contendo *asset-peso*, por exemplo PETR4-40%, o que seria uma ***prescrição*** para o *investidor* compra 40% do dinheiro do portfólio em ações da Petrobrás.

Gráfico 5.2.16a



Fonte: Elaborada pelo autor.

No gráfico abaixo, temos o mesmo *portifólio*, com os mesmos *10* (dez) *assets*, onde o modelo fez *2* (duas) ***prescrições*** para o *investidor*. A prescrição *MinRisk* e a *MaxSharpe-Ratio*.

- Opção MinRisk:

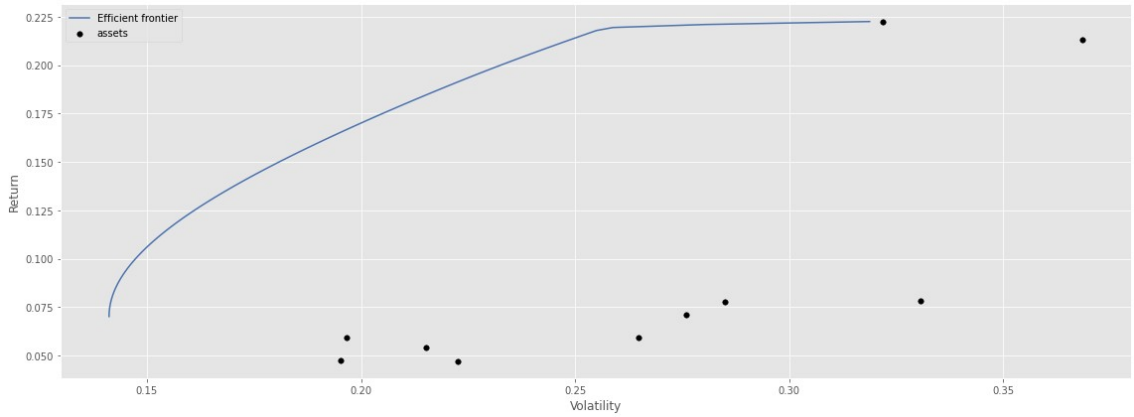
```
ABEV = 0.00
ELET = 0.00
EMBR = 0.00
GGBR = 0.3988881092079215
ITUB = 0.00
PETR = 0.00
RADL = 0.0238245469178762
RANI = 0.5772873438742026
VALE = 0.00
VIVT = 0.00
```

- Opção MinRisk:

```
ABEV = 0.2539905171133548
ELET = 0.00
EMBR = 0.0399640984210908
GGBR = 0.0009756663355737
ITUB = 0.1419048755059486
PETR = 0.0
RADL = 0.300096476296019
RANI = 0.100530062648538
VALE = 0.0
VIVT = 0.1625383036794751
```

De acordo com Markowitz, existe uma linha segura, onde o risco é muito pequeno, conforme gráfico abaixo, *gráfico 5.2.16b*.

Gráfico 5.2.16b



Fonte: Elaborada pelo autor.

Neste cenário, temos poucos cenários na *simulação*. Eu coloquei poucos cenários para clarificar que o *investidor* deve se manter na linha para mitigar o *risco* do *portfólio*. E a medida que se deseja aumentar o *retorno* (lucro) indo para cima, sobre a linha, o *risco* também aumenta sempre que escolhemos posições mais à direita do eixo *x*.

6 Conclusão

Os modelos híbridos se mostraram mais efetivos, e as pesquisas sugerem que métodos como o de *Análise Técnica*, *Aprendizado de Máquina*, *Estocásticos* e de *Física Quântica* (por conta da teoria com base na aleatoriedade) contribuir sobremaneira nesta mescla de domínios científicos.

Devido a natureza *heterocedasticidade*, é mais prudente executar os modelos periodicamente, pois pode ocorrer mudança de comportamento no mercado que se façam necessário o reajuste imediato do modelo utilizado.

Os modelos baseados em *risco* são mais estáveis, e mais fácil de serem estimados , com maior confiança, e geram melhores resultado, que modelo baseado em *retorno*, pois a incerteza sobre os *retornos* é maior que a incerteza sobre o *retorno*, é muito difícil estimar o *retorno*. E isso piora quando utilizamos simplesmente o *preço* sem qualquer tipo de *estacionariedade*, porque os modelos com dados *heterocedásticos*, como é o caso dos *preços*, são instáveis e não apresentam bons resultado, porque as propriedades estatísticas mudam com o tempo, como é caso da *média* e da *variância*.

A depender da distribuição, pode ocorrer que um modelo baseado em *risco* seja mais eficiente que um modelo baseado no *retorno*, e vice-versa, por isso é importante realizar as devidas simulações.

É muito difícil realizar a predição por conta do alto grau de imprevisibilidade do mercado financeiro. A maioria dos modelos de predição perdem para o caso *ingênuo* mais conhecido de *predição*, onde se utiliza o preço do dia anterior como *benchmark*.

Também é muito difícil realizar a prescrição porque os modelos de prescrição dependem de boas *predições* como *input*. A maioria dos modelos de prescrição perdem para o caso *ingênuo* mais conhecido de prescrição, onde se utiliza os *pesos* iguais para todos os *asset* como *benchmark*.

Após alguns testes, eu constatei que o modelo aumenta o erro médio para *backtime window* maiores que *20* (vinte) dias.

7 Trabalhos Futuros

Segue abaixo, o que eu desejaria contemplar nesta P&D, para efeitos de teste (entendimento/aprendizado), e que por falta de tempo eu não consegui implementar.

Comum

- Considerações utilizando variáveis "*exógenas*" (de fora do contexto), tidas como explicativas da realidade econômica como "*PIB*" (Produto Interno Bruto), Câmbio de Moedas (*dólar*), Sazonalidade, entre outros.
- Análise de erros utilizando métodos diferentes dos especificados no item de "*Apêndice - Escopo*".
- Avaliação de outros métodos de predição, que não são o foco desta *P&D*, para aumentar o peso de confirmação da tendência.
- Identificação e análise de *assets correlacionados*, onde um exerce influência ou segue o outro.
- Utilização de diferentes modelos a partir da *volatilidade* momentânea, sejam baixa, média e alta.
- Considerações sobre os custos operacionais e tributários sobre cada operação.
- Considerações sobre o "*time constraint*" (limitação de tempo), onde o tempo de cada operação seria de *6* (seis) horas (*16:00 – 10:00*) ou até *2* (dois) dias, que seria o caso típico de *dormir comprado*.
- Utilização de *timeframe* dinâmico a partir da *volatilidade*, onde em momentos mais voláteis poderia ser utilizado de um *timeframe* menor.
- Combinação de *2* (dois) *timeframes* distintos e ao mesmo tempo, uma para identificação de tendência (utilizando vários métodos, inclusive a *volatilidade*) e outro para a análise para identificar o melhor momento para a entrada e saída do mercado. Por exemplo, o de *tendência* em *15* minutos e o de *operação* em *5* minutos.
- Identificação, análise e tratamento de tendência, para se manter em uma posição por mais tempo.
- Identificação, análise e tratamento de sazonalidade e/ou ciclos, sejam eles anuais, mensais ou semanais. Por exemplo, é sabido que o mercado financeiro não costuma ficar aquecido antes e após as datas festivas (natal, ano novo, carnaval), onde a intuição é que os "*traders*" (operadores do mercado financeiro) estão preocupados em gastar dinheiro e não em ganhar.
- Identificação, análise e tratamento de "*white noise*" (variações nos dados).

Aprendizado de Máquina

- Experimentar modelos de "*econophysics*" (aplicação de teorias e métodos físicos para resolver problemas econômicos) como "*Quantum*" (teoria da física quântica), "*Black-Scholes*" (equação diferencial vastamente utilizada para precificar preços de *assets*), "*NLSE*" (Non-Linear Schroding Equation, é uma equação diferencial parcial não linear), "*Kinetic Theory of Gas*" (modelos termodinâmicos do comportamento dos gases), entre outros.
- Identificação de correlação entre novas *features*, diferentes das encontradas originalmente no *dataset* e que foram incluídas no *input* através de *feature engineering*".
- Combinar um mecanismo *preditor* linear com um não linear, para confirmação de resultados.
- Utilizar "*features measured*" (métricas de *randomness*, *long-term memory* and *scaling properties*, sobre as *features*) para aumentar a *acuracidade*.
- Teste utilizando dois *timeframes* distintos, um para sinalizar a tendência e outro para sinalizar o momento de entrada ou saída do mercado.
- Inclui outras *features* para representar o distanciamento do final de semana, para tentar captura alguma *tendência* ou *volatilidade* no mercado. A intuição por trás da *volatilidade* são os intemperes sistêmicos e *sazonalidade* como por exemplo: se o *timeframe* for minuto-a-minuto, é de se esperar que o mercado fique mais calmo durante o horário de almoço; se o *timeframe* for diário, é de se esperar que o mercado fique mais calmo ao final de cada mês; se o *timeframe* for mensal, o mês de Dezembro é atípico, assim como o de fevereiro no Brasil por conta de períodos festivos.

Otimização de Portfólio

- Utilizar o "*Robust Portfolio Management Model*" proposto Zhu and Fukushima (2009). O "*Robusto*" parece ser interessante, principalmente por ser um modelo de *Otimização de Portfólio* baseado em incertezas, como é o caso de *preços* de *asset*.
- Utilizar o "*Multi-Stage Stochastic Financial Planning*" proposto por Birge and Francois (2011).
- Utilizar mecanismos com foco em "*day trade*" (*operações* diárias), diference do que é mais comumente utilizado no mercado que são as otimizações anuais. A abordagem "*Bootstrap*" (método para estimativa de amostragem) parece ser um bom ponto de partida para otimizar o *portfólio* para o "*day-by-day*" (dia a dia).
- Identificação de diferentes tipos de *riscos*, como os *riscos sistêmicos* e os *não-sistêmicos*.
- Como aplicar o "*Robusto*" (modelo de *Otimização de Portfólio*) contendo "*constraint*" (critério para o modelo) de tempo, como é o caso de cenários de "*alta-frequência*" (entre segundos e minutos) de *operações*.

- Como utilizar o "*Robusto*" nas novas abordagens de modelos baseados em "*Factor*" e "*Mult-Factor*".

8 Referências Bibliográficas

CAJAS, Dany. Riskfolio: riskfolio-lib (2.0.0). 2021. Disponível em: https://github.com/dcajasn/Riskfolio-Lib. Acesso em: 07 nov. 2021.

Martin, R. A., (2021). PyPortfolioOpt: portfolio optimization in Python. Journal of Open Source Software, 6(61), 3066, https://doi.org/10.21105/joss.03066

Birge, John R, Francois Louveaux. 2011. Introduction to stochastic programming. Springer.

Zhu, Shushang, and Masao Fukushima. 2009. Worst-case conditional value-at-risk with application to robust portfolio management. Operations Research 57(5) 1155-1168.

Harry Markowitz. Portfolio selection. The Journal of Finance, 7(1):77–91, 1952. URL: http://www.jstor.org/stable/2975974.

Dany Cajas. Entropic portfolio optimization: a disciplined convex programming framework. 02 2021. doi:10.2139/ssrn.3792520.

Vaughn Gambeta and Roy Kwon. Risk return trade-off in relaxed risk parity portfolio optimization. Journal of Risk and Financial Management, 2020. URL: https://www.mdpi.com/1911-8074/13/10/237, doi:10.3390/jrfm13100237

Cajas, Dany, Entropic Portfolio Optimization: A Disciplined Convex Programming Framework (February 24, 2021). Available at SSRN: https://ssrn.com/abstract=3792520 or http://dx.doi.org/10.2139/ssrn.3792520

Arık, Sercan O., and Tomas Pfister. "Tabnet: Attentive interpretable tabular learning." arXiv (2020).

9 Glossário

Abstract	Breve resumo de um artigo de pesquisa.
Acatamento da Ordem	Efetivação da ordem de compra ou venda de um asset.
ACF	Auto Correlation Function, é uma função para identificação de autocorrelação.
Activation Function	Função que prover a não linearidade dos dados, permitindo que os modelos aprendam.
ADAM	Adaptive Moment estimation, é um algorítmico de otimização.
AIC	Alaike Information Criteria
Alta-frequência	Tempo variando entre segundos e minutos.
ANN	Artifical Neural Network.
API	Interface de Programação de Aplicações.
APT	É um tipo de "factor investing".
AR	Autoregressive Mode, é um modelo que modifica o modelo de regresão linear de multi-variáveis.
ARCH	Autoregressive Conditional Heteroskedasticity (Heteroscedasticidade Condicional Auto-regressiva).
ARIMA	Autoregressive Integrated Moving Average (Modelo Autoregressivo Integrado de Médias Móveis).
Artificial Intelligence	Inteligência Artifical
Asset	Ativo financeiro utilizado para investimento ou especulação.
Attention	Mecanismo para aumentar a performance de encoder-decoder nos modelos de 'Machine Learning'.
Autocovariance	Auto-covariância, função para obtenção da covariância.
AutoML	Aprendizado de máquina automático.
AWS	Empresa lider no mercado de infraestrutura de HW e SW para ambientes cloud.
B3	Bolsa de valores brasileira.
Back Propagation	Propagação para trás, da rede neural.
Backend	Camada de retaguarda da solução.
Background	Experiência anterior.
Backtest	É um teste de modelos, utilizando dados históricos.
Backtime Window	Quantidade de dias, para compor a janela temporal de dados a ser passado para o modelo.
Backtime Windows	Ou "Timestep Back", é a janela de tempo passado que o modelo recebera como input para predizer o futuro.
Baseline	Ponto de partida, linha base ou trabalho base.
Bechmark	Resultado referência.
BELRFS	Brain Emotional learning-Based Recurrent Fuzzy System (Sistema Fuzzy recorrente baseado na aprendizagem emocional do cérebro).
Bias	Tendência, relativa a uma ou mais camadas, na rede neural.
BIC	Bayesian Information Criteria
Black Box	Caixa preta. Modelo que não disponibiliza explicação ou interpretação de como chegou no resultado.

Black-Scholes	É uma equação diferencial vastamente utilizada para precificar preços de assets.
Bootstrapping	Bootstrapping é um método que estima a distribuição de amostragem que coleta várias amostras
Box-Cox Transform	Data transform method that supports both square root and log transform.
Business-days	Quantidade de dias comerciais em um ano fiscal.
Bussword	Palavra da moda.
Capital	Quantia em dinheiro disponível para investir ou especular.
CAPM	Capital-Asset Pricing Model. É um método para estimar o "retorno".
Carteira	É um conjunto de *assets* a serem utilizados em investimentos, também conhecido como *portfólio*.
CLA	Critical Line Algorithm. É um modelo de otimização.
Clean-up	Limpeza de dados.
Cloud	Ambiente para hospedagem com especialização em infraestrutura de HW e SW.
CML	Capital Market Line, é a tangente que intercepta a "Efficient Frontier" no seu ponto ótimo de maximização do "Sharpe Ratio".
Constraint	São critérios a serem avaliados durante a execução de modelos de otimização.
Cost Function	É uma função de perda (custo).
Covariância	Corelação ou influência entre asset, ou variáveis aleatórias.
CPU	Central Processing Unit (Unidade Central de Processamento).
Criptocurrence	Criptomoedas, ou moedas digitais.
Cross validation	Validação cruzada, dos dados.
CRRA	Constant Relative Risk Aversion, é uma técnica utilizada em modelos de otimização.
CSV	Comma Separeted Value (Valores Separados por Vírgula), é a extensão de arquivo-texto destinado ao intercambiamento de informações tabulares, como planilhas eletrônicas e 'dataframes'.
Dashboard	Painel de visualização sumarizada.
Data-Driven	Orientado a dados.
DataClass	Classe destinada a ser utilizada para dados, sem a existência de métodos.
Dataset	Conjunto de dados.
Day Trade	Operações diárias, normalmente associadas a especulação.
Day-by-Day	Dia a dia, um dia seguido do outro.
Debug	Depuração, de código.
Debug Mode	Execução para efeitos de depuração do código fonte.
Default	Padrão.
Denoise	Retiradas de ruídos
Dickey-Fuller Test	É um teste estatístico para verificação de stacionariedade dos dados.
Dinheiro	Moeda agnóstica para esta *P&D*, podendo ser a mesma moeda da s*érie histórica* (dataset).
Dividend yeld	Rendimentos de dividendos.
DNN	Deep Neural Network (Redes Neurais Profundas)
Dropout	É um mecanismo para evitar overfitting.
Dump	Gravação de um conjunto de informações binárias no HD.

Early stopping	Parada antecipada, em relação a quantidade de ciclos de treinamento.
Econophysics	Aplicação de teorias e métodos físicos para resolver problemas econômicos.
EDP	Equação Diferencial Parcial
Efficient Frontier	Fronteira Eficiente, conceita de diversificação da carteira para reduzir o risco.
Engenharia de Features	Menamismo para tratamento ou criação de "features".
Ensemble	Método que utiliza múltiplos algoritmo de aprendizado.
Enterpretability	
Error Metrics	Mecanismo para medição de erros do modelo, baseado do distanciamento entre os valores reais com os valores preditos.
Estacionária	Série temporal onde a média, variância e estrutura de autocorrelação não mudam no decorrer do tempo.
Estocástico	Ciência Analítica de Predição.
EWMA	Exponentially Weighted Moving Average, é uma estatística para modelar a volatilidade.
Exôgenas	De fora do contexto.
Explainability	O modelo pode ser explicado de maneira que um ser humano entenda.
Facade	É um design pattern com a funcionalidade de prover uma interface (mascara) simples de uma 'package' mais complexa.
Factors Optimization	Modelos baseados em "factor investing".
Feature	Dado de entrada na rede neural.
Feature Engeeneering	Mecanismo para manipulação (criar, remover ou modificar) de features.
Feature Scaling	Escalamento ou transformação dos dados para faixa de valores parecidas.
Features measured	Métricas (randomness, long-term memory e scaling properties) sobre as features para aumentar a acuracidade.
Feed-forward	Antecipação de informação, em rede neural.
Feeder	Provedor do dataset.
Fit	Aprendizado, ou ajudste.
Fix	Correções, de código.
Forex	Foreign Exchange, é um investimento baseado na competição entre duas moedas.
Forget Gate	No LSTM, é o primeiro 'block'.
Frontend	Camada de visualização da solução.
Função Estacionarizadora	Função responsável por estacionarizar a série temporal, utilizando o preço do dia atual e do dia anterior.
Gaps	Saldos ou buracos, nos dados.
GARCH	Generalized Autoregressive Conditional Heteroskedasticity (Heteroscedasticidade Condicional Auto-regressiva Generalizada).
Ghost Batch Normalization	Ghost batch normalization (upper right) is a modified version of batchnorm that normalizes the mean and variance for disjoint sub-batches of the full batch.
GitHub	Repositório público para versionamento de código, hosteado pela Microsoft.
GMRAE	Geometric Mean Relative Absolute Error, Erro médio absoluto

	relativo da média geométrica
GPU	Graphics Processing Unit (Unidade gráfica de processamento).
Gradient	É um mecanismo matemático para minimizar a função de perda (custo).
GRU	Gated Recurrent Unit (Unidade Recorrente Bloqueada).
GUI	Graphical User Interface (Interface Gráfica de Usuário).
Gurobi	Um dos melhores 'solvers' da atualidade.
Heteroscedasticidade	Variância condicionada ao tempo.
Hidden layers	Camadas escondidas, na rede neural.
High-Frequency	Trades com intervalos temporais na ordem entre segundos e minutos.
Hill Climbing	É uma técnica de optimização.
Home broker	Corretor residencial, utilizado remotamente através de um aplicativo web ou mobile.
HRP	Hierarchical Risk Parity (paridade de risco hierárquica), é um método para otimizar a média da variância (risco).
Human Reasoning	Raciocínio humano, normalmente útil para complementar modelos com base em 'inteligência artifical'.
Hyper parameters	Parâmetros, do classificador da rede neural.
IBOVESPA	Índice da B3 (Bolsa de Valores Brasileira).
ICA	Independent Component Analysis (Análise de Componente Independente).
IDE	Integrated Development Environment (Ambiente de desenvolvimento integrado).
In-scope	Dentro do escopo
inf	"infinito" na linguagem python.
Input	Valores de entrada.
Inverse	Operação de inversão/reversão dos dados de "feature scaling" para os valores originais.
IV	Inverse Variance (Variância Inversa)
K-fold	Envelopamento de tamanho "K".
Kelly Optimization	É uma técnica de optimização.
Kinetic Theory of Gas	Modelos termodinâmicos do comportamento dos gases.
Knowing Model	Theory Driven Model.
Label	Rótulo ou categoria.
Lambda	Função Lambda, em Clound, na AWS, é um serviço de computação sem senvidor e orientado a eventos. Existe um conceito mais formal de Lambda no domínio de computabilidade e semântica de linguágem.
Learing Model	Data Driven Model.
Learning rate	Taxa de aprendizagem, da rede neural.
Library	Artefato computacional para reúso de código.
Likelihood	Verossimilhança.
Liquidez	Facilidade para realizar uma operação (compra ou venda) efetiva a qualquer momento.
Log Div	Log da Divisão, onde a divisão ocorre entre o valor (preço) atual pelo anterior. É utilizado como um método para estacionarizar a sérite temporal.
Log Likelihood	LLF (Log Likelihood Function). It measures the goodness of fit of a statistical model to a sample of data for given values of

	the unknown parameters.
Log Return	It s useful when there is a huge difference of magnitude order between some assets.
Lógica Fuzzy	É a forma de lógica multi-valorada, na qual os valores de verdade das variáveis podem ser qualquer número real entre 0 (correspondente ao valor falso) e 1 (correspondente ao valor verdadeiro).
LoLiMot	Locally Linear Model Tree.
Long-Term Momory	Memória de longo prazo.
LP	Linear Predictor, é um preditor linear.
LSTM	Long-Short Term Memory (Memõria de longo prazo), rede neural criar por S. Hochreiter & J. Schmidhuber, em 1997.
MA	Moving Average Model.
Machine Learning	Aprendizado de máquina automático.
MAE	Mean Absolute Error (Erro Absoluto Médio).
MAE	Mean Absolute Error (Erro Absoluto Médio).
MAPE	Mean Absolute Percentage Error (Erro de porcentagem média absoluta)
MASE	Mean Absolute Scaled Error (Erro Médio Absoluto Escalado)
Matriz de Covariancia	Matriz simétrica que representa o quando um conjunto de observações, de cada asset, diferente um do outro.
Mattress Money	Dinheiro de colchão, dinheiro guardado sem a realização de aplicações.
Max	Função de maximização, de uma coleção finita ou infinita de valores.
Maximal Error	Quantidade máxima de error consecutivos, antes de interromper o treinamento.
Maximum Likelihood Estimation	MLE (Maximum Likelihood Estimation), It is a method of estimating the parameters of an assumed probability distribution, given some observed data.
mCVAR	Mean Conditional Value at Risk (valor condicional médio de risco), é um método para otimizar a média da variância (risco).
MdRAE	Median Relative Absolute Error, Erro absoluto relativo mediano
Middleware	Camada intermediária de uma solução.
Min	Função de minimização, de uma coleção finita ou infinita de valores.
Minimal Error	Erro mínimo, para a solução.
ML	Machine Learning (Aprendizado de Máquina).
MLP	Multi-layer perceptron (Perceptron multicamadas), It s the most common form of ANN.
Momentum	Momento ou tração.
MPT	Modern Portfolio Theory, é a teoria moderna para gestão de portfólio.
MSE	Mean Squared Error (Error Médio Quadrado)
MSLE	Mean Squared Logarithmic Error, é uma métrica de erro.
Multistep	Múltiplos saltos, a frente, na previsão de séries temporais.
MV	Minimum Variance, é a variância mínima.
MVO	Mean Variance Optimization. É um método/técnica para optimização dos pesos de cada asset no porrifólio.
Naive	Ingênuo. Abordagens muito básicas e simplistas são conside-

	radas como ingênuas.
NaN	"Not a Number" na linguagem Python. Não é um número.
Neural network	Rede neural.
Neurons	Neurônios, de uma cada na rede neural.
NLSE	Non-Linear Schroding Equation, é uma equação diferencial parcial não linear.
Noising	Ruído nos dados.
Normalization	Reescala os valores entre uma faixa, que normalmente é entre 0 e 1.
On the fly	Durante o curso, da operação.
Operar Comprado	O investidor espera que o ativo valorize
Operar Vendido	O investidor está realizando um investimento em renda variável esperando que esse mesmo ativo caia e desvalorize
Out of sample	Fora da amostra (dataset de treino).
Out-scope	Fora do escopo
Outlier	Datos significativamente diferentes.
Output	Valores de saída.
Output Layer	It connects the model with the outer world.
Overfitting	Ocorre quando o modelo está muito ajustado a um input específico, e menos generarista.
Overhead	Sobrecarga, de entendimento ou de trabalho.
Overtraining	Treinamento excessivo.
Overwrite	Sobrescrever um método ou uma classe.
P&D	Pesquisa e Desenvolvimento.
PACF	Partial Auto Correlation Function, It indicates the correlation between the time séries yt and $yt-k$.
Package	Pacotes de código computacional dedicado a um domínio específico.
Pandas	Package' em python para manipulação de 'dataset'.
Paper	Trabalho científico.
Passive Investing	Trades com intervalos temporais na ordem entre meses e anos.
Payload	Conteúdo a ser entregue ou deployado.
PCR	Principal Components Regression, é uma técnica para estimar coeficientes de regressão desconhecidas em um modelo de regressãio linear padrão.
Percentage Change	Percentual de retuno, calculado entre o valor (preço) anterior e o atual. É utilizado como um método para estacionarizar a sérite temporal.
Peso	Percentual ou proporção que cada *asset* possui em uma carteira.
PIB	Produto Interno Bruto.
PMPT	Post-Modern Portfólio, teoria menos conservadora que a MPT.
Portifólio	Carteira consistida de um conjunto de *assets*.
Power Transform	The square root transform and log transform belong to a class of transforms called power transforms.
Pre-Processamento	Tratamento dos dados, antes da execução do modelo principal.
Precision-Recall	São dois modelos para avaliar métricas.
Pregão	Momento em que é permitido a realização de ordens de compra ou venda, tipicalmente dentro de horários comerciais e em dias comerciais.
Production Mode	Execução final utilizando o melhor cenários obtido pelo mode-

	lo.
Programação Dinâmica	É um método para a construção de algoritmos para a resolução de problemas computacionais, em especial os de otimização combinatória.
Quantum	Teoria da física quântica.
R2	R-Squared, Error Quadrado
RAE	Relative Absolute Error (Erro Absoluto Relativo).
Random Walk	É um processo estocástico/randômico.
Randomness	Randomicidade.
Release	É uma versão com foco em correções e 'bugs' ou pequenas funcionalidades.
Return	Lucro operacional sobre um asset.
Risk	Está associado a volatividade do asset, ou a perda financeira sobre uma provável operação.
Risk Parity Optimization	É uma técnica de optimização.
RMSE	Root Mean Square Error, Erro de raiz quadrada média
RMSE	Root Mean Square Error (Raiz Quadrada do Erro Médio).
RMSLE	Root Mean Squared Logaritmic Error, Erro Logarítmico Médio Quadrático
RNN	Recorrent Neural Network (Redes Neurais Recorrentes)
Road blocks	Barreiras no caminho.
Robust	É um modelo de otimização onde exista incerteza.
RRSE	Root Relative Squared Error (Raiz Quadrada do Erro Relativo).
Saída do Mercado	Enceramento de todas as operações que estejam em curso, ou execução, realizando o lucro ou o prejuízo.
Saldo Atual	Saldo após a realização de uma operação.
Scaling Properties	Propriedade de escalabilidade.
Scalp	Estratégia de 'trading' onde o tempo das operações são de curtíssima duração, podendo ser segundo ou alguns poucos minutos.
Score	Pontuação, de acuracidade da rede neural.
Script	Linguagem de programação que não gera código nativo, mas sim interpretado.
SCS	Splitting Conic Solver, pacote em python, de otimização.
Seed	Sementes, a serem utilizadas em randomizações.
Série Temporal	Conjunto de dados onde cada registro contém uma relação com o tempo.
ServerLess	Serviço em cloud, sem a contratação direta de um servidor.
Sharpe Ratio	It is the ratio between returns and risk.
Shift	Deslocamento do dado, para a direita ou esquerda, ou na linha do tempo.
Slot	Intervalo de dados ou tempo.
SMAPE	Symmetric Mean Absolute Percentage Error, Erro percentual médio absoluto simétrico
Solver	Ambiente computacional especializado em resolver modelos de 'optimization'.
Stacionarization	Estacionarizar. Tormar estacionária.
Stackable	Mecanismo de 'ensemble' de empilhamento de vários/diferentes algorítmo de aprendizado que geram resultados para a segunda camada.

Stardardization	Técnica para padronizar os valores centrando em média 0 e variância 1, se aproximando assim de uma distribuição normal padrão.
Stop Loss	Marco para saída do mercado em caso de perda, realizando o prejuízo.
SVM	Support-Vector Machines (Máquina de Vetores de Suporte).
TabNet	Modelo de classification desenvolvido pelo Google Research.
Take Profit	Marco para saída do mercado em caso de atingimento de meta de lucro, realizando o lucro.
Test	Conjunto de dados para testar o resultado final.
Test Mode	Execução com foco de testagem geral.
Theory-Driven	Orientado a teoria.
Ticket by ticket	Transação por transação.
Time constraint	Limitação de tempo.
Timeframe	Fatia de tempo, frequência, ou a menor fração de tempo da série temporal.
Timeslot	Intervalo de tempo a ser considerada como janela de trabalho ou processamento.
Trade system	Painel de sinalização financeiro .
Trade-off	Avaliação entre o custo e o benefício.
Trader	Operadores do mercado financeiro.
Train	Conjunto de dados para treinar o modelo.
Transformation Function	Função capaz de decompor e descrever ou representar outra função (ou uma série de dados) originalmente descrita no domínio do tempo.
Tuning	Ajuste fino.
Valid	Conjunto de dados para checar se o modelo foi bem treinado, ou validar os resultados do modelo.
VGP	Vanishing Gradient Problem, decoberto por S. Hochreiter, em 1991. Onde para grandes "stime step training", o gradiente pode tender a zero.
Wavelet	Função capaz de decompor e descrever ou representar outra função (ou uma série de dados) originalmente descrita no domínio do tempo.
Wealth	Riqueza
Webhits	Índice de popularidade.
White Noise	São variações nos dados, que não é possível ser explicado por regressões lineares.
Windows Insiders	Versão beta do '*windows*' para experimentação antecipada de novas funcionalidades.
Workflow	Fluxo de trabalho ou tarefas.
WSL	Windows Subsystem for Linux, é um ambiente "linux" fornecido pela Microsoft para rodar dentro do windows.
X	Conjunto de dados reais, ou verdadeiros.
y	Conjunto de dados preditados, ou previstos.

10 Apêndice – Tema Escolhido

Alguns pesquisadores não veem valor em pesquisar sobre predições de *asset* financeiro pela dificuldade de se encontrar bons resultado, principalmente porque os resultados possuem grandes erros médios. Eu aprecio esta área justamente porque o problema continua em aberto e pela dificuldade em se encontrar bons resultado.

No geral, os adeptos pelos *Métodos Clássicos* não sugerem o uso dos *Métodos Quant*. Assim como, os especialistas nos *Métodos Quant* afirmam que não se faz necessário o uso de *Métodos Clássicos* para a obtenção de predições. Durante esta *P&D* eu percebi ambos os domínios servem de ferramental um para o outro.

Eu gostaria de tentar refutar a ideia que a ciência ainda não conseguiu descobrir modelos melhores do que os já publicados, baseado na minha intuição que é possível que o pesquisador possa ter encontrados bons modelos e possa não ter publicado a sua descoberta, ou não tenha publicado na sua plenitude. Enquanto isso, eu espero aumentar a acuracidade nas *predições* e *prescrições*, décimo a décimo, a partir da combinação de diferentes abordagens e áreas de conhecimento aqui apresentadas.

O foco não é um *asset* em específico, pois eu entendo que modelo desenvolvido possa ser configurado para atuar em outros tipos de *asset* financeiro como o de "*forex*" (investimento baseado em moedas) ou "*cripto-currency*" (moedas digitais) e afins, bastando o "*tuning*" (ajuste fino) dos "*hiper-parameters*" (parâmetros do classificador da rede neural) do modelo, para atender as características do *asset* desejado.

Inicialmente, eu havia considerado a *Álgebra* como uma linha de solução para esta P&D. Durante as pesquisas eu concluir que a *Álgebra* é um poderoso ferramental a ser utilizado como meio e não como fim.

Ainda no início, desta P&D, eu havia colocado o *risco* como fora do escopo desta *P&D*. Durante as pesquisas eu tive um melhor entendimento sobre *Otimização de Portfólio* e ficou claro a necessidade de considerar o *risco* como uma variável importantíssima a ser analisada.

11 Apêndice – Metodologia de Pesquisa

Eu utilizei uma metodologia informal e não padronizada para realizar a pesquisa deste trabalho científico, entretanto eu segui passos que eu julguei coerentes para a realização no mesmo, conforme detalhamento que se seguem.

Como pesquisa científica, este trabalho apresenta resultados e reproduções determinísticas e estocásticas. Os resultados consolidam as investigações, tidas como sondagem e pesquisa, por mim realizadas. Esta pesquisa visa avaliar os mecanismos básicos existentes para a predição e prescrição dos preços de *asset*. Como resultado, eu apresentarei algumas ideias, conceitos, teorias e regras que encontrei no contexto científico. Embora já existam textos, bibliografias e pesquisas sobre os temas aqui apresentados, eu qualifico esta pesquisa como exploratória por pelo fator de eu saber muito pouco sobre o assunto, e por se tratar de um tema muito difícil de se obter respostas com boa acurácia. Esta pesquisa não tem a pretensão de apresentar resultados puro e teóricos, mais sim resultado aplicados ao tema em questão, podendo ser classificada como uma *P&D* para obter conhecimentos básicos para construir um modelo próximo ao título desde documento.

O foco inicial desta pesquisa era realizar comparações entre modelos de Aprendizado de Máquina, através de comparações quantitativa dos erros médios de cada modelo. A medida que eu evolui na pesquisa, e principalmente quando eu tive um melhor entendimento sobre as vantagens sobre os modelos de *Otimização de Portfólio*, eu percebi que faria mais sentido combinar os modelos de *Aprendizado de Máquina* e *Otimização de Portfólio*.

Eu não segui um Mapeamento Sistemático formal, contudo eu fiz *4* ciclos de pesquisa. No meu primeiro ciclo de pesquisa, dentre os vários que eu fiz, eu realizei buscas filtrando pela palavra chave "*asset prediction*" e identifiquei alguns temas interessante a partir de leitura do "*Abstract*" (resumo). Conforme lista abaixo contendo o tema e a frequência (quantidade de ocorrência) abaixo.

LocallyLinearModelTree (LoLiMot) = 21

LSTM (LongShortTermMemory) = 12

NeuralNetwork = 23

Wavelet = 5

GeneticAlgorithim = 10

DeepLearning = 6

SupportVectorMachine = 4

LS-SVM = 2

No segundo ciclo de pesquisa eu li as conclusões, e no terceiro ciclo eu li os trabalhos relacionados com o "*paper*" (pesquisa) do ciclo anterior. No terceiro ciclo de pesquisa eu filtrei por *paper* mais recentes, a partir de 2018, com maiores citações. O quarto ciclo foi para ler de forma completa, os principais *paper* que eu já havia filtrado até o momento.

12 Apêndice – Metodologia de Desenvolvimento

O desenvolvimento foi feito na linguagem *Python*, utilizando as boas práticas de orientação a objeto e baixo acoplamento entre os artefatos computacionais. Ao final de casa passo eu gero ao menos um arquivo *csv* de resultado da etapa em questão. Em alguns passos eu gero alguns arquivos *bin* (binários) que serão utilizados em passos posteriores, com os conteúdos abaixo.

- Algoritmo de *normalização*
- Lista contendo o nome das colunas
- Matriz de mask
- Matriz de *explainability*
- Importância das features
- *Plot* das *mask* (*png*)

O ideal é que a *série temporal* possua registro de aproximadamente 2 décadas, com o menor *timeframe* possível, que é o "*ticket-by-ticket*" (transação por transação). Contudo este cenário ideal não é factível por questões de custos para a aquisição deste tipo de *dataset*. A escolha do melhor *feeder* de *dataset* seguiu os critérios abaixo.

- Origem a partir de repositórios confiável.
- Quantidade substancial de objetos (instância), preferencialmente das últimas 2 (duas) décadas, com *timeframe* de *ticket-by-ticket*.
- Alto número de "*webhits*" (índice de popularidade).
- Possua os atributos (campos) mínimos para esta *P&D*.
- Ausência de *GAP*s (saltos nos dados).
- Preferência por dados já normalizados.
- Alto número de citações científicas.
- Quantidade de *papers* relevantes.

A análise de resultados ocorreu sobre as principais *métricas de erros* para modelos de *regressão*, conforme lista abaixo.

- *MAE* - Mean Absolute Error (Erro Absoluto Médio)
- *MAPE* - Mean Absolute Percentage Erro (Erro de Porcentagem Média Absoluta)
- *MSE* - Mean Squared Error (Error Médio Quadrado)
- *RMSE* - Root Mean Square Error (Erro de Raiz Quadrada Média)
- *RMSLE* - Root Mean Squared Logaritmic Error (Erro Logarítmico Médio Quadrático)
- *R2* – R-Squared (Error Quadrado)

As evidências do desenvolvimento e teste podem ser verificadas nos arquivos de log, conforme esclarecimentos das máscaras abaixo.

Nome do Arquivo = `./log/ut_X_00_run_all_YYYZZ.log`

Legenda:

X = É a macro etapa de processamento, dentre as opções abaixo.

b = Pré-Processamento da Predição

c = Processamento da Predição

d = Pré-Processamento da Prescrição

e = Processamento da Prescrição

YYY = É o tipo de processamento.

DBG = Modo de *Debug*, com a configuração de cenários reduzida e o nível de *log* máximo.

PRD = Modo de *Produção*, com a configuração de cenários completa e o nível de *log* reduzido.

ZZ = É o tamanho da *backtime window*.

05 = Cinco dias passados

10 = Cinco dias passados

15 = Cinco dias passados

20 = Cinco dias passados

13 Apêndice – Pesquisa – Aprendizado de Máquina

Outros modelos de Aprendizado de Máquina, como o *LSTM* e o *GRU* também foram considerados durante a fase de pesquisa, mas não foram priorizados por se tratarem de caixa-preta (caixa-preta). O *GRU* prover resultados semelhantes a *LSTM*. Maiores detalhes sobre o escopo desta *P&D* podem ser vistos no item "*Apêndice – Escopo*".

O *Tabnet*, criado na *Google Cloud AI*, por SO Arık, T Pfister (2019, 2020), propõem a *explainability* e a *interpretability*.

Eu estudei *Algoritmo Genético* e fiz algumas simulações práticas, e não identifiquei como ele poderia ser útil sozinho. Entretanto ele pode ser útil em *Otimização de Portfólio*, ou combinado rede neural.

Eu também estudei e fiz simulações práticas utilizando *Lógica Fuzzy*, e entendi que se trata de um poderoso aliado para a tomada de decisão. A partir de *inputs*, passando por *regras* previamente definidas, e gerando *outputs*. Os *inputs* poderiam ser o resultado de uma *rede neural* em conjunto com alguma outra informação de *tendência*. As *regras* poderiam ser um conjunto de diretrizes especializadas para o mercado financeiro. Os outputs poderiam ser *sinais de compra* ou de *venda*, ponderados. Contudo, para utilizar a *Lógica Fuzzy* é necessário ter um especialista na área financeira para definir as *regras* e principalmente para realizar os ajustes inicias.

14 Apêndice – Pesquisa – Otimização de Portfólio

Inicialmente eu não pretendia dar foco a *variância* e ao *risco*, muito menos ao rendimento, e que o erro seria um bom indicador comparativo. Contudo, quando a *Otimização de Portfólio* ficou mais clara, eu entendi que a *Otimização* utiliza a *variância* e o *risco* como *input*, e que faria mais sentido comparar o *retorno* e não somente o *erro*.

15 Apêndice – Pesquisa – Outros Modelos

Durante as minhas pesquisas, eu me deparei com vários modelos os quais descreverei o meu entendimento a seguir de forma sucinta.

A "*Álgebra Linear*" em si é uma ferramenta dentro dos outros domínios, ela não aparece sozinha. O mesmo entendimento pode ser dado para o "*Estocástico*", embora que em menor grau, fazendo com que a "*Álgebra Linear*" e o "*Estocástico*" seja um ferramental útil a vários domínios da ciência. O *Aprendizado de Máquina* e a *Otimização de Portfólio* utilizam a "*Álgebra Linear*" e o "*Estocástico*" como ferramentas. Os dados (*série temporal*) do mercado financeiro não podem ser modelados, sem realizar transformações, por um processo linear como se o preço atual fosse uma combinação linear dos preços do passado. Com isso, não é possível obter bons resultado utilizando um modelo puramente algébrico linear porque a *série temporal* é não-linear e são representados por equações dinâmicas não-lineares. O uso de "*Álgebra Não-Linear*" poderia ser uma alternativa, mas não é certo que se encontre bons resultados.

A "*Cadeias de Markov*" embora me pareceu simples de ser implementada e de fácil explicação, ela me pareceu forte (robusta).

O "*Wavelet*" me pareceu um ótimo mecanismo para filtrar os ruídos da *volatilidade* dos preços, fato que poderia ser útil como ferramenta de "*denoise*" (retiradas de ruídos) pois tratasse de uma função capaz de decompor e descrever ou representar outra função originalmente descrita no domínio do tempo, assim como "*Transformadas de Furrier*", dentro de modelos *preditivos* e *prescritivos* mas não exatamente para realizar *predição* nem *prescrição*.

O "*Runge-Kutta*" me pareceu interessante utilizando alguma "*EDP*" (Equação Diferencial Parcial). Neste caso eu optaria por utiliza o "*Black-Scholes*" ou a "*NLSE*" (Non-Linear Schroding Equation) como *EDP* para precificar os *assets*.

Além das alternativas já descritas acima, eu também tive conhecimento sobre outras abordagens, que me fizeram entender o quão extenso pode ser o tema de predição e Prescrição no mercado financeiro: "*LoLiMot*" (Locally Linear Model Tree), "*BELRFS*" (Brain Emotional learning-Based Recurrent Fuzzy System, "*SVM*" (Support Vector Machine), *LSTM*, Programação Linear, Gradiente Descendente, "*Hill Climbing*" (técnica de *Otimização*), "*Random Walk*" (é um processo estocastico/randômico), Robusto Adaptativo, "Precision-Recall" (são modelos para avaliar métricas), entre muitos outros.

Durante as pesquisas foi possível identificar a evolução dos modelos de predição da variância. O modelo "*ARMA*" (Auto-regressive Moving Average) não performa bem com séries temporais "*não-estacionárias*", tornando necessário a manipulação dos dados antes deles serem consumidos pelo modelo *ARMA*. O modelo *ARIMA* (Auto-regressive Integrated Moving Average) performa bem com as séries temporais não estacionárias, realizando, internamente, a captura da variação de preço. Contudo o *ARIMA* não performa bem com dados "heterocedásticos" (variância condicionada ao tempo) e com agrupamento de variância. Considerando-se que os dados do mercado financeiro são heterocedásticos e com agrupamento de variância, o "*GARCH*" (Generalized Auto-regressive Conditional Heteroskedasticity) se mostrou mais realístico, principalmente para prever a *covariância* dos returnos em séries temporais do mercado financeiro.

16 Apêndice - Ambiente

Esta P&D será desenvolvida utilizando o ambiente computacional abaixo, podendo uma opção se tornar mais viável que outra.

- ***Software***

  SO :

  Windows 10 64bits (10.0.18352 Compilação 18362)

  WSL2 (Linux)

  IDE : *GoogleColab*[1], *Jupyter*[2], e *VSCode*[0]

  Linguagem : *Python 3.8*

  Gestão de Pacotes : Virtual Environment[3] (*Python*)

  Pacotes :

  Comuns = *scipy*, *numpy*, *pandas*

  Estocástico = *seaborn*, *statsmodels*, *patsy*, *numpy*

  Aprendizado de Máquina = *pytorch-tabnet, sklearn*, *torch* [4,5]

  Otimização de Portfólio = *pypfopt* (PyPortfolioOpt), *pyfolio*, *Riskfolio-Lib*

  Visualização = *pandas*, *matplotlib*, *sympy*, *seaborn*

([0]) Eu iniciei o desenvolvimento no ambiente *WSL2* (Windows Subsystem for Linux , versão 2), em uma versão beta do *windows*, chamada "*Windows Insiders*" (versão beta do *Windows*). Ocorreu uma atualização do *Windows Insiders* na minha máquina que tentou migrar do *Windows 10* para o *Windows 11*. A migração não pode ser finalizada por limitação do *HW* do meu laptop. Após o restabelecimento do *Windows 10*, o *WSL2* da minha máquina parou de funcionar. Como eu não tinha um outro ambiente *Linux* disponível, e poderia demorar a obter um outro ambiente *Linux*, eu preferi continuar o desenvolvimento no ambiente *Windows* e não mais no *Linux*.

([1]) O *GoogleColab* é muito rápido na execução. Embora esta P&D não demande muito volume de dados nem de processamento. Quando eu avancei no processamento e no uso de disk, o *GoogleColab* passou a mostrar uma tela de uso de recursos (*HW*) com muita frequência, fato que impactou um pouco a operacionalização do ambiente.

([2]) O *Jupyter* apresentou muitos problemas de incompatibilidades entre os pacotes, principalmente entre o *pytorch-tabnet* e o *torch*. Este problema ocorreu mesmo isolando e gestionando o *environment* com o *virtual environment* e o *pip*.

([3]) Eu utilizei o *virtual environment* para isolar o ambiente de desenvolvimento contendo os pacotes utilizados. A ativação do *environment* é realizada em nível de sessão e não em nível de diretório, com isso ao se fechar a sessão (*shell*) o *environment* perde o estado de *activated*.

([4]) O pacote *pytorch-tabnet* depende do pacote *torch*, e o pacote *torch* dependia de uma versão mais antiga do *Python*. Este detalhe não estava explícito nos manuais de nenhum dos dois pacotes. Eu precisei migrar da versão "*3.10*" para a versão "*3.8*" do *Python* para que tudo funcionasse no meu ambiente local com o *torch*. Estes problemas não foram encontrados utilizando ambiente *GoogleColab* pois ele já resolve os problemas de conflitos e dependência para o desenvolvedor. No caso do *GoogleColab*, ele utili-

zou a versão "*3.7.12*" do *Python* que é a última "*release*" (versão como foco em correções de *bugs* ou implementação de pequenas funcionalidades) mais segura, até o momento, da versão "*3.7*" do *Python*. Contudo para *Windows*, última *release* mais segura é a *"3.7.9" do Python*, até o momento.

([5]) O pacote *torch*, em ambiente *Windows*, necessita da biblioteca *torch_python.dll*. Esta biblioteca é obtida através da instalação do pacote "*VC_redist.x64.exe*", "*Microsoft Visual C++ Redistributable*", fornecido pela *Microsoft*. Embora eu tenha preferência pelo ambiente *Linux*, não foi possível continuar o desenvolvimento no "*WSL2*" (Windows Subsystem for Linux) porque o *WSL2* parou de funcionar após um *upgrade* do *Windows*, fato que me induziu a continuar o desenvolvimento no ambiente *Windows* por questões de tempo para reconfigurar, do zero, o meu ambiente de desenvolvimento.

Eu utilizei o *virtual environment* para isolar o ambiente de desenvolvimento contendo os pacotes utilizados. Durante o desenvolvimento, eu percebi que a ativação do *environment* é realizada em nível de sessão e não em nível de diretório, com isso ao se fechar a sessão (*shell*) o *environment* perde o estado de *activated*.

Tabela com a versão dos *pacotes* utilizadas...

- ***Hardware***

  CPU = Intel(R) Core(™) i5-3340M CPU @ 2.70GHz 2.70 GHz

  GPU = Eu não dispunha de GPU para realizar os processamentos.

  RAM = 16.0 GB

  HD = Samsung SSD 840 EVO 1TB

17 Apêndice - Reprodução

A possível reproduzir os experimentos desta *P&D* seguindo o roteiro abaixo:

- Baixar o código no repositório do "*github*" (versionador) no link abaixo.

  https://github.com/frajola-puc/saod/tree/main

  A "*branch* (ramificação) = *main*" já está configura em modo "*produção*", e sem os *outputs* que são gerados durante a execução, pois eles são grandes.

  A "*branch = frajola*" já está configurada em modo *debug*, e com todos os *outputs* já gerados, pois eles são pequenos. Esta *branch* também pode servir para evidência de *desenvolvimento* e *teste*.

  Existe uma parte do código, relacionado a *Otimização de Portfólio*, que por restrições de tempo não pode ser migrada do *GoogleColab* para o meu ambiente *local/github* acima. Por questão de organização de código e modularização, este código importa outros 2 (dois) *Python Notebooks* através do artifício disponibilizado pelo pacote *ipynb*. Um notebook possui configurações e classes básicas/comuns a *Aprendizado de Máquina* e a *Optimização de Portfólio*, o outro notebook possui configurações e classes básicas/comuns a *Otimização de Portfólio*. Este código pode ser acessado no link abaixo.

  https://colab.research.google.com/drive/1jp0lEZCDwrIq1bE5fJrZ-HONd2xd-UCKI?usp=sharing

- Uma das *pacote*, a *torch*, depende do *Python 3.8*, com isso é necessário instalar esta versão antes de seguir para as próximas etapas abaixo.
- O pacote *torch*, em ambiente *Wndows*, necessita da biblioteca *torch_python.dl*". Esta biblioteca é obtida através da instalação do pacote "*VC_redist.x64.exe*", "*Microsoft Visual C++ Redistributable*", fornecido pela "*Microsoft*".
- As boas práticas sugerem que utilizemos um gerenciador de ambiente, principalmente para resolver conflitos e dependência de *pacotes*. Existem várias alternativas para gerenciar/instalar as pacotes necessárias para a execução. Eu utilizei a mais clássica e nativa que é a *virtual env*, mas conhecida como *venv*. Para a instalação dos pacotes basta executar os comandos abaixo, no diretório raiz do projeto. Abaixo, na primeira linha existe um *ponto* antes da última palavra *venv*, e na segunda e terceira linha nas únicas palavras *venv* em cada linha. Não é necessário atualizar o pacote *pip*, em caso isso seja sugerido durante a execução do programa *pip*.

```
python -m venv .venv
cd .venv/Scripts
activate
.venv/Scripts/pip install -r ./../../requirements_python8.txt
```

O ambiente criado possui referências estáticas, fato que impossibilita o reúso do ambiente, através do "*copy/past*" (copiar/colar) do diretório dos *pacotes* de um local para outro.

- Alguns *pacotes* de *otimização* como o "*scs*" (Splitting Conic Solver), entre outras, necessitam serem recompiladas em *C/C++*. Por este motivo, em caso de ambiente *Windows*, é necessário instalar o "*MS Visual C++ 14.0*", não precisa instalar o *Visual Studio* inteiro, basta o "*Visual C++ Build Tools*" que se encontra no link abaixo. Talvez pode ser instalado outras distribuições do *C/C++* como o *MinGW*, "*Minimalist GNU For Windows*". Após a instalação, é preciso colocar o *path* do executável do *compilador* e do *linkeditor* na variável *PATH* do *usuário* logado ou do *systema*, uma única vez para *compilar* e *linkeditar* os *pacotes* que foram codificadas em *C/C++*.

  https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2019

- Não é necessário realizar configurações adicionais. Em caso se deseje realizar execuções rápidas com pouco volume de dados, basta manter a configuração da variável "*B_DEBUG=True*", no arquivo de configuração (*./a_com/com_0_config.py*), ou "*B_DEBUG=False*" para operacionalizar execuções mais fidedignas e próximo da realidade com séries históricas de mais de *20* (vinte) anos, com *10* (dez) *assets*, e uma ampla combinação de cenários de execução sobre as séries dos *10* (dez) *assets*, conforme descrito no item "*5.2. Desenvolvimento*". A mudança de configuração na variável "*B_DEBUG*" propagará mudanças em outras variáveis automaticamente para que outras variáveis também reflitam o mesmo modo desejado de execução. A execução com "*B_DEBUG=True*" leva aproximadamente *05* (cinco) minutos, e a com "*B_DEBUG=False*" leva aproximadamente *35* (trinta e cinco) minutos.

  Na tabela abaixo (tabela "Reprodução") vemos quais parâmetros terão os seus domínios reduzidos e quais serão os seus domínios, estando a variável "*B_DEBUG*" sendo setada como "*True*".

Tabela 17

Parâmetros do Modelo Tabnet			
Nome	**Descrição**	**Domínio**	**Qt**
V_RAW_ASSET_PRD	Lista de *assets* financeiros.	["PETR3.SA', 'VALE3.SA']	2
S_DT_BEGIN	Data Inicial da Série Histórica	01/Out/2021	n/a
V_I_MAX_EPOCH	Número máximo de "*epoch*".	[2]	1
V_I_BATCH_SIZE	Tamanho do "*batch*" de treino	[128]	1
V_I_VIRTUAL_BATCH_SIZE	Tamanho do "*batch*" para o "*Ghost Batch Normalization*".	[32]	1
Total de Cenários			320

Fonte: Elaborada pelo autor.

- A *seed* já está configurada como "*I_SEED=2021*", no arquivo de configuração (*./a_com/com_0_config.py*).

- Os programas executores principais, contendo o método "*main( )*", possuem a máscara "*./_main_*.py*". Estes instanciam "*class*" principais que estão dentro dos arquivos com a máscara "*./_cls_*.py*".
- O pré-processamento (tratamento dos dados) da predição pode ser realizado executando o programa " *./main_b_b_predição.py*", através do comando abaixo no diretório raiz do projeto. Abaixo, existe um *ponto* antes da palavra *venv*.

  ```
  .venv/Scripts/python main_b_b_predição.py
  ```

  Em caso seja necessário executar o pré-processamento da predição novamente, os diretórios abaixo serão deletados automaticamente. Em caso seja necessário manter estes diretórios, é necessário fazer um *back-up* (cópia) deles. Esta etapa tem a duração de aproximadamente *7 (sete)* segundos em modo *debug* (*B_DEBUG=True*), e de aproximadamente *09* (nove) minutos em modo *production* (*B_DEBUG=True*).

  ```
  ./dataset/b_01_ds_fetch
  ./dataset/b_02_feature_add
  ./dataset/b_03_feature_del
  ./dataset/b_04_stationarity
  ./dataset/b_05_stationarity_del
  ./dataset/b_06_slice_tvt
  ./dataset/b_07_date_del
  ./dataset/b_08_scale
  ./dataset/b_09_slice_xy
  ```

- O processamento, relativo a predição, pode ser realizado executando o programa " *./_main_d_c_tabnet_all.py*", através do comando abaixo no diretório raiz do projeto. Abaixo, existe um *ponto* antes da palavra *venv*.

  ```
  .venv/Scripts/python main_c_prediction.py
  ```

  Os diretórios abaixo serão deletados automaticamente. Em caso seja necessário manter estes diretórios, é necessário fazer um *back-up* (cópia) deles.

  ```
  ./dataset/c_10_pred_*
  ./dataset/c_11_pred_best
  ```

- O pré-processamento (tratamento dos dados) da prescrição pode ser realizado executando o programa " *./main_d_b_prescription.py*", através do comando abaixo no diretório raiz do projeto. Abaixo, existe um *ponto* antes da palavra *venv*.

  ```
  .venv/Scripts/python main_d_b_prescription.py
  ```

  Em caso seja necessário executar o pré-processamento da prescrição novamente, os diretórios abaixo serão deletados automaticamente. Em caso seja necessário manter estes diretórios, é necessário fazer um *back-up* (cópia) deles.

  ```
  ./dataset/d_12_ds_concat
  ```

- O segundo processamento, relativo a prescrição (Prescrição), pode ser realizado executando o programa " *./_main_e_e_markowitz.py*", através do comando abaixo no diretório raiz do projeto. Abaixo, existe um *ponto* antes da palavra *venv*.

  ```
  .venv/Scripts/python main_e_prescription.py
  ```

  Os diretórios abaixo serão deletados automaticamente. Em caso seja necessário manter estes diretórios, é necessário fazer um *back-up* (cópia) deles.

  ```
  ./dataset/e_13_pres_*
  ./dataset/e_14_pres_best
  ```

- Eu codifiquei utilizando o "*vscode*" (IDE da Microsoft). Em caso se deseje visualizar o projeto no *vscode*, basta executar o comando abaixo no diretório raiz do projeto. Abaixo, existe um *espaço* seguido de um *ponto*, logo após a palavra *code*.

  ```
  code .
  ```

- O nome dos programas executores de testes unitários possuem a máscara "*./ut_*.py*". Em caso seja necessário executar alguns testes unitários, basta executar um ou mais programas de teste unitário, conforme comandos que se seguem. Abaixo, existe um *ponto* antes da palavra *venv*.

  ```
  cd .\u_unit_test
  .venv/Scripts/python ut_*.py
  ```

- Configurações adicionais podem ser feitas, a depender do objetivo, a partir dos arquivos de configuração abaixo.

  ```
  .\a_common\a_0_config.py
  .\c_prediction\c_0_config.py
  .\c_prediction\c_0_config_tabnet.py
  .\e_prescription\e_0_config_markowitz.py
  ```

- A reprodução em modo *debug* (*B_DEBUG=True*), assim como no modo *production* (*B_DEBUG=False*), não é aconselhável modificar o intervalo de datas (DT_BEGIN_XXX e DT_BEGIN_XXX) para um intervalo menor que 2 meses para não resultar em menos de 20 dias de "*business-day*" (dias comerciais), o que poderia resultar em menos de 2 linhas nos arquivos de *valid* e *test.* Se estes dois arquivos possuírem menos de 2 linhas, impossibilitará o uso de algumas *métricas de erros* como a de *R2*. Estas configurações podem ser feitas no arquivo " *./a_common/a_0_config.py*".

Para realizar *future engineering*, particulares a um modelo em específico, basta criar uma nova "*class*" (classe) que herdará o comportamento da *class* base, chamada *RunOneBase* do arquivo "*./a_com/com_0_run_one_base.py*". Esta nova "*class*" terá o mesmo nome da *class* pai chamada *FeatureAdd* do arquivo "*./a_art/c_0_b_2_feature_add.py*", e fará um "*overwrite*" (sobrescrever o método) no método "*run()*" de modo a efetuar as devidas *future engineering* particulares ao modelo em questão.

18 Apêndice - Customização

A possível adicionar funcionalidades ao pré-processamento (tratamento dos dados), e aos vários passos de processamento, tanto os de predição como os de *prescrição*, conforme detalhamento que se segue:

- No pré-processamento, basta criar uma nova *class* que herdará o comportamento da *class* base, chamada "*./a_com/com_0_run_one_base.py*". Esta nova *class* será semelhante as *classes* abaixo.

```
.\b_pre\b_1_ds_fetch.py
.\b_pre\b_2_feature_add.py
.\b_pre\b_3_feature_del.py
.\b_pre\b_4_stationarity.py
.\b_pre\b_5_stationarity_del.py
.\b_pre\b_6_slice_tvt.py
.\b_pre\b_7_date_del.py
.\b_pre\b_8_scaled.py
.\b_pre\b_9_slice_xy.py
```

- No processamento de predição, basta criar uma nova *class* que herdará o comportamento da *class* base, chamada "*./a_com/com_0_run_one_base.py*". Esta nova *class* será semelhante as *classes* abaixo.

```
.\c_prediction\c_1_pred_tabnet.py
```

- No processamento de prescrição (Prescrição), basta criar uma nova *class* que herdará o comportamento da *class* base, chamada "*./a_com/com_0_run_one_base.py*". Esta nova *class* será semelhante a *classe* abaixo.

```
.\e_prescription\e_1_pres_markowitz.py
```

19 Apêndice – Binário do Modelo

O tempo de execução e o tamanho do binário, do modelo, são importantes, principalmente para definir a melhor arquitetura de utilização em "*cloud*" (Ambiente para hospedagem com especialização em infraestrutura de *HW* e *SW*). Em *cloud*, existe o conceito de funcionalidade "*lambda*" (em *cloud*, na *AWS*, é um serviço atômico que se assemelha a um evento) que podemos entender conceitualmente como um serviço "*ServerLess*" (Sem Contratação de Servidor). O uso de serviços *lambda* possui restrições no tamanho do "*payload*" (pacote do binário) e no tempo de execução. A ordem de grande do tamanho pode girar entre *50Mb* e *300Mb*, e a do tempo pode girar entre *5 minutos* e *15 minutos*.

Tabela Apêndice – 19

Informações sobre o Binário do Modelo (Média)					
#	**Modelo**	**DBG (debug)**		**PRD (produção)**	
		Tamanho (Mb)	**Tempo (hh:mm:ss)**	**Tamanho (Mb)**	**Tempo (hh:mm:ss)**
01	Tabnet	272	00:00:01	411	00:02:53

Fonte: Elaborada pelo autor.

20 Apêndice - Premissas

Esta *P&D* foi desenvolvida assumindo as premissas abaixo:

- O *home broker* possua funcionalidades de "*stop loss*" (gatilho de saída, diante de prejuízo) e "*take profit*" (gatilho de saída, realizando lucro) no *home broker*.
- Existência de *liquidez*, principalmente para a "*entrada e a saída do mercado*" com valores próximo do desejado. Na "entrada no mercado" com valores próximo do valor de "*open*" (abertura do dia corrente) ou "*close*" (fechamento do dia anterior). Na "*saída do mercado*" com valores próximo do "*predito*" e principalmente ser possível "*sair do mercado*" durante os últimos minutos do "*pregão*".

21 Apêndice - Escopo

As funcionalidades abaixo são "*in-scope*" (dentro do escopo) desta *P&D*.

Comum

- Não será analisado a latência do ambiente. Mesmo entendendo que a latência pode impactar a "*acatamento da ordem*" (realização efetiva da ordem de compra ou venda) este problema seria melhor analisado em uma *P&D* com foco em operações de "*alta frequência*" (segundos ou minutos).
- *Normalização* e Codificação dos Atributos do *dataset*.
- Estou assumindo que o *home broker* possui a funcionalidade de gatilho de entrada e saída, que já é bastante comum entre os *home brokers*. Esta funcionalidade é fundamental para a "*saída do mercado*" (encerar as operações em aberto/curso) quando valores pré-determinados já forem atingidos a partir de "*stop loss*" (gatilho de saída, diante de prejuízo) e "*take profit*" (gatilho de saída, realizando lucro).
- Análise sobre a volatilidade.
- Pesquisa exploratória, correlacional, pura, e comparativa.

As funcionalidades abaixo são "*out-of-scope*" (fora do escopo) desta *P&D*.

Comum

- Considerando que esta P&D não foca no "*backend*" (camada de retaguarda), "*frontend*" (camada de visualização da solução), nem na arquitetura de "*middleware*" (camada intermediária da solução), não é necessário especificar nem desenvolver as camadas de persistência, de barramento de comunicação, e de visualização. Esta *P&D* tem foco na pesquisa e desenvolvimento de um modelo computacional para predição e Prescrição, e não no desenvolvimento de uma aplicação (aplicativo) computacional.
- Geração manual ou automática de *dataset*.
- Identificação e tratamento de *outliers* (dados muito diferentes).
- O *clean-up* do *dataset*. Durante a seleção do *dataset*, eu darei preferência por um *dataset* sem *gaps* nos dados.
- Análise da proporcionalidade ideal entre *train*, *valid* e *test* do *dataset*.
- Avaliação da performance do tempo de execução do modelo apresentado nesta *P&D*.

- Análise e melhoria da complexidade do algoritmo do modelo apresentado nesta *P&D*.
- Considerações utilizando variáveis explicativas da realidade econômica como "*PIB*" (Produto Interno Bruto), *câmbio*, *sazonalidade*, entre outras.
- Foco ou especialização em um tipo de mercado em específico como *cripto-moedas* (moedas digitais), "*forex*" (investimento baseado em moedas), *mercado futuro*, *mercado de ações*, entre outros.
- Análise de erros utilizando métodos diferentes dos especificados como sendo "*in-of-scope*" (dentro do escopo).
- Comparações entre mais de 1 métodos distintos de predição e de prescrição.
- Foco cambial, inflacionário e de correção monetária. Esta *P&D* possui mais foco na variação do saldo do *portfólio* sem considerar qualquer exogeneidade.
- Considerações sobre qualquer tipo de despesas sobre as operações, tais como despesa de corretagem (*home broker*), aluguel de *assets*, bancárias ou com impostos.
- Não será analisado a latência do ambiente. Mesmo entendendo que a latência pode impactar a "*acatamento da ordem*" (realização efetiva da ordem de compra ou venda) este problema seria melhor analisado em uma *P&D* com foco em operações de "*alta frequência*" (segundos ou minutos).
- Mudança de estratégia após uma ordem (operação) ser dada (iniciada). Eu entendo que a mudança de estratégia é importante e muito difícil de ser definida.
- Busca por uma técnica para *normalização* da data, por entender que a data absoluta não é tão importante quando a relativa. É racional pensar que o mercado possa aumentar ou diminuir a *volatilidade* antes e depois de datas importantes como feriado, final de semana, final de mês, publicação de indicadores de juros, desemprego, câmbio, entre outros.
- Operacionalização real de entrada e saída no mercado financeiro.
- Uso de mecanismos de *Analise Técnica*.
- Definição de Heurística para o processamento massivo, em busca dos melhores "*hiper-parameter*" (parâmetros da rede neural).
- Horizonte de predição maior que *1* (um) dia, a frente.
- Execução de "*backtest*" (teste do passado) rolante, para acompanhar o saldo do *portfólio*.

Aprendizado de Máquina

- Utilização de "*Auto-ML*" (*Aprendizado de Máquina* automático).
- Considerações e tratamento com as *seeds* para suportar a randomização, necessário para alguns algoritmos como os de *neural network*, entre outros.
- Por questões de tempo, os modelos *LSTM* e *GRU* não fazem parte desta *P&D*. Embora estes dois modelos sejam mais apropriados para séries temporais, eu priorizei o *Tabnet* por conta da *explainability* e da *interpretability*

oferecida por este modelo, diferente dos problemas comuns de caixa-preta (caixa-preta) da "*RNN*" (Rede Neural Recorrente).

Otimização de Portfólio

- Por restrição de tempo, os modelos com base na abordagem "*Robust*" (modelo de *Otimização de Portfólio* onde exista incerteza).

22 Apêndice – Background

Os meus conhecimentos previamente acumulados, até o momento inicial desta *P&D* que supostamente poderiam me ajudar eram:

- ***Cálculo (Básico) –*** Através dos cursos de "Cálculo 1", "Cálculo 2" e "Calculo 3", cursados na PUC-Rio.
- ***Álgebra Linear (Básico*) –** Através do curso de "Álgebra Linear I", cursado na PUC-Rio.
- ***Probabilidade e Estatística (Básico) –*** Através do curso de "Probabilidade e Estatística", assistido como ouvinte na PUC-Rio.
- ***Análise Técnica (Básico*) –** Através do curso de "Análise Técnica", na Apligraf.
- ***Forex (Básico) –*** Através de experiência prática, e mal sucedida, em operações reais.
- ***Contrato Futuro (Básico) –*** Através de experiência prática, e mal sucedida, em operações reais.
- ***Python (Básico) –*** Através de experiência prática, bem-sucedida, desenvolvendo pequenos programas.

23 Apêndice – Dificuldades e Soluções

Aprender novas tecnologias computacionais não foi impactante nesta P&D. Os "*road blocks*" (bloqueios) ocorreram na parte matemática, mais precisamente na parte de *Probabilidade, Estatística e Otimização*, e eu precisei de mais tempo para me ambientar nestas áreas.

As soluções para as dificuldades encontradas foram obtidas mediante pesquisa e estudo nos temas em questão. Algumas matérias básicas foram importantes para facilitar a leitura e pesquisa dos *papers*. Algumas outras matérias extras foram fundamentais para clarificar os pontos faltantes.

- **Matérias Básicas**

  MAT1200 – Álgebra 1

  MAT1154 - Cálculo 4

  ENG1029 - Probabilidade e Estatística

  INF1771 - Inteligência Artificial

  ENG1036 – Probabilidade Computacional

- **Matérias Extras**

  ENG1456 – Introdução a Computação Aplicada

  ELE2709 – Redes Neurais 2

  INF2979 – Aprendizado de Máquina

Outras matérias igualmente interessantes não puderam ser cursadas por falta de tempo.

24 Apêndice - Backtest

Por restrições de tempo não foi possível realizar o "*backtest*" (teste com dados do passado) rolante. Contudo, neste item eu esboço como um *backtest* poderia ser realizado, conforme detalhamento que se segue.

O *backtest* utiliza dados históricos dos últimos "*+20 anos*", de "*03/jan/2000*" até "*30/nov/2021*", de todos os *assets* que compõem o *portfólio* definido na "*Tabela 5.2.1*".

O *portfólio* é inicializado com *10000* dinheiros, independente da moeda, conforme definido no item "*5.2.1 Comum*". O *backtest* simula somente *1* (uma) única operação diária, ao início do dia, onde baseado no valor *predito* pelo modelo *preditivo*, é realizado uma operação acreditando que o valor *predito* irá se realizar até o final do dia.

Se o valor *predito* for negativo, a operação será acreditando que o mercado terá uma queda e neste caso a operação será de "*entrar vendido*" (comprar contratos de venda ou alugar *asset*) no mercado. Se o valor *predito* for positivo, a operação será acreditando que o mercado terá uma subida e neste caso a operação será de "*entrar comprado*" (operação tradicional de compra).

O *backtest* realiza a contabilização do resultado diário para a obtenção do saldo a cada dia. O "*saldo atual*" (saldo do dia) é calculado a partir de uma aritmética trivial, conforme fórmula abaixo:

Fórmula 24a

$$S_d = S_{d-1} + (R_1 + R_2 + ... + R_{n-1} + R_n)$$

Onde:
S_d = Saldo do Dia Atual (resultado total do *Portfólio*, ao final do dia/pregão).
S_{d-1} = Saldo do Dia Anterior.
$R_1 \dots R_n = R_x$ = Resultado da Operação sobre de cada "*asset X*" (ao final do dia, "*1 <= x <= n*").
n = Quantidade de *asset* no *Portfólio* (*n* = | *"Ambev", Eletrobrás", ..."Petrobrás", "Vale"* | = *10*).

Fonte: Elaborada pelo autor.

O resultado sobre cada operação de compra ou venda de cada *asset* também possui uma aritmética simples, conforme fórmula abaixo:

Fórmula 24b

$$R_x = V_x * P_x$$

Onde:
$R_x = R_1 \dots R_n$ = Resultado da Operação sobre de cada "*asset X*" (ao final do dia, "*1 <= x <= n*").
$V_x = V_1 \dots V_n$ = Valor a ser Investido em um "*asset X*" (durante o dia, "*1 <= x <= n*").
$P_x = P_1 \dots P_n$ = Percentual de Valorização/Desvalorização de um "*asset X*" (do dia corrente, "*1 <= x <= n*").
n = Quantidade de *asset* no *Portfólio* (*n* = | *"Ambev", Eletrobrás", ..."Petrobrás", "Vale"* | = *10*).

Fonte: Elaborada pelo autor.

Este *backtest* inicial não visa *otimizar* os *pesos* do *portfólio*. Neste *backtest* eu utilizo *pesos* simplistas onde todos os *assets* possuem *pesos* iguais. O "*backtest*" utilizado no modelo de prescrição focará na *Otimização* dos *pesos* do *portfólio*, e este é detalhado no item "*5.2.14 Prescrição*". Desde modo o *saldo atual* do *portfólio* é sempre dividido pelo número de *assets* no *portfólio*, que no caso desta *P&D* é igual a "*10*".

Fórmula 24c

$$V_x = int\ (V_t / n)$$

Onde:
$V_x = V_1 \ldots V_n$ = Resultado da Operação sobre de cada "*asset X*" (ao final do dia, "*1 <= x <= n*").
V_t = Valor Total do *Portfólio* (no início do dia/pregão, antes de qualquer operação de compra ou venda).
n = Quantidade de *asset* no *Portfólio* (*n* = | *"Ambev", Eletrobrás", ..."Petrobrás", "Vale"* | = *10*).

Fonte: Elaborada pelo autor.

Conforme definido nos itens "*Apêndice – Escopo*" e "*Apêndice - Premissas*", esta P&D considera que existência da funcionalidade de "*stop loss*" (gatilho de saída, diante de prejuízo) e "*take profit*" (gatilho de saída, realizando lucro) no "*home broker*". Isso é importante para ser possível sair do mercado quando a meta de lucro for atingida sem a necessidade de aguardar até o final do *pregão* (expediente para operações de compra e venda). Eu emulo este comportamento avaliando os valores de "*high*" (maior) e "*low*" (menor) do dia em questão, para identificar se a meta foi ou não atingida, conforme critérios previamente definidos.

A "*Estratégia I*", descrita na tabela abaixo, possui a "*target*" (objetivo) de atingir o valor que o modelo *preditor* efetivamente previu. Em caso este valor não seja atingido, temos 2 cenários possíveis. No primeiro cenário, linha "*#1*" da tabela abaixo, com tendência de "*alta*", "entramos comprado" no mercado, executando uma ordem com o valor igual ao valor de "*open*" (abertura), ou de "*close*" (fechamento) do dia anterior, e em caso a "*meta*" (valor predito) não seja atingida, será computado o maior valor entre o valor de *close* e o valor "*high*" (maior alta do dia). No segundo cenário, linha "*#2*" da tabela abaixo, com tendência de *baixa*, "*entramos vendido*" no mercado, executando uma ordem com o valor igual ao valor de *open*, ou de *close* do dia anterior. Em caso a *meta* não seja atingida, será computado o menor valor entre o valor de *close* e o valor *low*.

Tabela 24a

Backtest - Estratégia I (meta="*valor predito*")				
#	**Tendência**	**Operação**	**Critério**	
			Entrada	**Saída**
01	Alta	Comprado	open	min (meta , max (close, high))
02	Baixa	Vendido	open	max (meta , min (close, low))

Fonte: Elaborada pelo autor.

A "*Estratégia II*", descrita na tabela abaixo, é análoga a "*Estratégia I*", com a diferença que a *meta* é um percentual de lucratividade definido pelo "*trader*".

Tabela 5.2.17b

Backtest - Estratégia II (meta="*% de lucro*")				
#	**Tendência**	**Operação**	**Critério**	
			Entrada	**Saída**
01	Alta	Comprado	open	min (open * meta , close)
02	Baixa	Vendido	open	max (open * meta , close)

Fonte: Elaborada pelo autor.

Conforme premissas descritas no item "*Apêndice – Premissas*", esta *P&D* considera que não existiram problemas de *liquidez* e que sempre será possível executar a ordem desejada no momento desejado, desde que a *ordem* seja uma *ordem* racional e utilize o valor instantâneo (momentâneo) do mercado.

Conforme definido no item "*Apêndice – Escopo*", esta *P&D* não tem o foco em modificações da estratégia "*on the fly*" (durante a execução da ordem) durante um mesmo dia. Uma vez definida a estratégia para "*entrada no mercado*" (ordem de compra ou venda para entrar) em um determinado dia, esta será seguida até a sua "*saída do mercado*" (ordem de venda ou compra para saída) no mesmo dia.

Durante o *backtest,* é fundamental tentar entender os eventos ruins. Estes são mais importantes que os eventos bons.