PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Rafael Soares Melero

Sugestionamento de Investimento e Melhor Momento

São Paulo
2021

Rafael Soares Melero

Sugestionamento de Investimento e Melhor Momento

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

São Paulo

2021

SUMÁRIO

1. Introdução

1.1. Contextualização

Atualmente no Brasil o mercado financeiro tem atraído cada vez mais a atenção dos investidores comuns, pessoas que não tem o devido conhecimento de seus riscos e funcionamento, sendo assim, não conseguem distinguir alguns dos pilares de investimentos disponíveis e seu comportamento, como: melhor momento de investimento, seus principais influenciadores etc.

Em sincronia, nos últimos anos, multiplicaram-se as propagandas de corretoras, bancos e influenciadores digitais que difundem a aplicação do capital em um pilar de investimento específico desconsiderando as diversas possibilidades que o mercado financeiro oferece.

Contudo, ao mesmo tempo em que os interessados em investimentos parecem ter muitas referências no assunto com o surgimento dessas novas fontes de informação, a qualidade das mesmas disponíveis não é boa, tão pouco confiável.

O ato de investir é muito pessoal, é preciso levar em conta informações únicas, portanto a análise dessas também devem ser únicas e cuidadosas.

1.2. O problema proposto

Para reduzir as perdas dos novos investidores e apresentar-lhes outros pilares de investimento (*Why?*).

Investidores inexperientes que desejam aplicar em diversos pilares de investimento e identificar o melhor momento para tal (*Who?*).

Baseado em indicadores econômicos, elaborar um sugestionamento de pilar de investimentos e o momento a ser realizado para obter resultado em 30 pregões (*What?*).

O escopo geográfico considerado para o estudo contempla a bolsa de valores brasileira, a B3 (*Where?*).

Foram utilizados dados históricos dos períodos de 2010 a 2020, totalizando 2409 pregões da bolsa e o prazo definido entre início e fim da operação financeira é de 30 pregões (*When?*).

1.3. Objetivos

Identificar a escolha, em um prazo de 30 pregões, entre a existência de um melhor pilar de investimento e qual seria ou se a melhor opção é não realizar aplicações e aguardar o momento oportuno.

2. Coleta de Dados

O processo de coleta de dados iniciou-se com a definição dos pilares de investimentos a serem considerados na análise.

Foram incluídos também os índices das principais bolsas e mercados a serem analisados e por estarmos principalmente focados no mercado brasileiro, os principais influenciadores deste como algumas *commodities* e taxas.

Os dados serão divididos em 4 tipos:

- Taxas: que representam indicadores intimamente relacionados à economia local;
- *Commodities*: relacionadas à cotação das empresas que a negociam ou ativos que são utilizados como proteção;
- Índices: principal indicador do andamento da economia da região e das empresas que ali são negociadas;
- Ações: indicador específico de uma empresa negociada.

Estes dados foram coletados manualmente em sites especializados no mercado financeiro.

2.1. Taxas

2.1.1. Taxa Selic

Criada em 1999, a taxa Selic tem este nome por derivar do Sistema Especial de Liquidação e de Custódia do Banco Central. Ela representa a taxa de juros básicos da economia brasileira, ou seja, ela influencia todas as taxas de juros praticadas no país – sejam as que um banco cobra ao conceder um empréstimo ou as que um investidor recebe ao realizar uma aplicação financeira.

Sua representação se dá por meio de um percentual que é aplicado ao montante investido. As informações referentes a estes percentuais foram coletadas no Banco Central do Brasil (BC).

Selic.csv		
Nome da coluna	**Descrição**	**Tipo**
Data	Data em que foi apurada a taxa	Data

Valor	Percentual da taxa vigente no período	Numérica (%)

2.1.2. Taxa Câmbio: Dólar

A taxa de câmbio é o preço da moeda estrangeira, no caso o dólar (US$), com base na moeda nacional, real (R$). Esta cotação foi coletada no Banco Central do Brasil (BC).

Dolar.csv		
Nome da coluna	**Descrição**	**Tipo**
Data	Data em que foi apurada a cotação	Data
COLUNA1	Desprezada	Desprezada
COLUNA2	Desprezada	Desprezada
TIPO	Desprezada	String
COMPRA	Valor de compra do dólar	Numérico
VENDA	Valor de venda do dólar	Numérico
COLUNA3	Desprezada	Desprezada
COLUNA4	Desprezada	Desprezada
COLUNA5	Desprezada	Desprezada

2.2. *Commodities*

2.2.1. Minério de Ferro Refinado

Principal ativo comercializado pela empresa Vale que tem hoje a maior participação no índice Bovespa com aproximadamente 14%. Este é um dos principais fatores de influência do índice Bovespa. Os dados foram coletados no site especializado *Investing*.

Minerio_Ferro.csv		
Nome da coluna	**Descrição**	**Tipo**
DATA	Data em que foi apurada a cotação	Data
ÚLTIMO	Preço aplicado à última negociação do dia	Numérico
ABERTURA	Preço definido no leilão de abertura do pregão	Numérico

MÁXIMA	Preço máximo negociado no pregão	Numérico
MÍNIMA	Preço mínimo negociado no pregão	Numérico
VOL.	Volume de negociações realizadas no dia	Numérico
VAR%	Variação diária dos preços	Numérico (%)

2.2.2. Petróleo *Brent* Futuros

Seu nome vem de uma plataforma da Shell chamada *Brent* e hoje representa todo o petróleo extraído do Mar do Norte comercializado na bolsa de Londres. Sendo um dos principais indicadores do valor do petróleo no mundo e a principal *commodity* da Petrobrás. Os dados foram coletados no site especializado *Investing*.

Petroleo_Brent.csv		
Nome da coluna	**Descrição**	**Tipo**
DATA	Data em que foi apurada a cotação	Data
ÚLTIMO	Preço aplicado à última negociação do dia	Numérico
ABERTURA	Preço definido no leilão de abertura do pregão	Numérico
MÁXIMA	Preço máximo negociado no pregão	Numérico
MÍNIMA	Preço mínimo negociado no pregão	Numérico
VOL.	Volume de negociações realizadas no dia	Numérico
VAR%	Variação diária dos preços	Numérico (%)

2.2.3. Ouro

Sua negociação dá-se principalmente quando há insegurança e incertezas no mercado em geral, sendo um ativo internacionalmente aceito que confere mais segurança e proteção por tratar-se de um bem material. A variação de sua cotação está intimamente ligada a estas variações de baixa no mercado e por este motivo o elencamos como um importante pilar de investimento. Os dados foram coletados do site especializado *Investing*.

Ouro.csv		
Nome da	**Descrição**	**Tipo**

coluna		
DATA	Data em que foi apurada a cotação	Data
ÚLTIMO	Preço aplicado à última negociação do dia	Numérico
ABERTURA	Preço definido no leilão de abertura do pregão	Numérico
MÁXIMA	Preço máximo negociado no pregão	Numérico
MÍNIMA	Preço mínimo negociado no pregão	Numérico
VOL.	Volume de negociações realizadas no dia	Numérico
VAR%	Variação diária dos preços	Numérico (%)

2.3. Índices

2.3.1. Índice Bovespa

Criado em 1968, consolidou-se como referência para os investidores. Ele representa uma carteira teórica de ações e *units* de companhias listadas na B3 representando cerca de 80% do número de negócios e volume financeiro do mercado brasileiro de capitais. Seu histórico foi extraído do *site* especializado *Yahoo! Finance*.

Indice_Bovespa.csv		
Nome da coluna	**Descrição**	**Tipo**
DATE	Data da coleta do índice	Data
OPEN	Cotação da abertura do pregão	Numérico
HIGH	Cotação máxima atingida pelo índice na data	Numérico
LOW	Cotação mínima atingida pelo índice na data	Numérico
CLOSE	Cotação do último negócio realizado na data	Numérico
ADJ CLOSE	Ajuste realizado no fechamento do mercado	Numérico
VOLUME	Volume de negócios realizado no pregão.	Numérico

2.3.2. Índice S&P 500

Assim como o Ibovespa a bolsa americana também tem seu índice de referência, o S&P 500, que foi criado em 1957. Este índice é uma carteira teórica das 500 ações mais representativas negociadas na NYSE (Bolsa de Nova Iorque) e na NASDAQ. Seu histórico foi coletado no *site* especializado *Yahoo! Finance*.

Indice_S&P500.csv		
Nome da coluna	**Descrição**	**Tipo**
DATA	Data da coleta do índice	Data
ÚLTIMO	Cotação do último negócio realizado na data	Numérico
ABERTURA	Cotação da abertura do pregão	Numérico
MÁXIMA	Cotação máxima atingida pelo índice na data	Numérico
MÍNIMA	Cotação mínima atingida pelo índice na data	Numérico
VOL.	Volume de negócios realizado no pregão	Numérico
VAR%	Variação diária da cotação	Numérico

2.4. Ações

Foi levantado a composição do Índice Bovespa e separadas as top 5 empresas que representam 37% da composição total do índice conforme o quadro abaixo:

Código	Ação	Tipo	Qtde. Teórica	Part. (%)
VALE3	VALE	ON NM	3.380.233.503	14,25%
ITUB4	ITAUUNIBANCO	PN ED N1	4.780.002.924	6,12%
PETR4	PETROBRAS	PN N2	4.566.511.125	5,14%
BBDC4	BRADESCO	PN EJ N1	4.691.427.537	4,37%
PETR3	PETROBRAS	ON N2	3.426.385.188	3,97%
B3SA3	B3	ON NM	6.079.530.858	3,73%
			TOP 6 AÇÕES	37,58%
			OUTRAS	62,42%
ABEV3	AMBEV S/A	ON	4.358.814.864	3,12%
WEGE3	WEG	ON NM	1.484.859.030	2,55%
ITSA4	ITAUSA	PN ED N1	4.515.559.175	2,20%
MGLU3	MAGAZ LUIZA	ON NM	2.820.185.158	2,15%
GNDI3	INTERMEDICA	ON NM	592.114.608	2,14%

Composição Índice Bovespa

Para a empresa Petrobrás consideraremos apenas umas de suas ações negociadas devido a cotação de ambas serem influenciadas pelos mesmos fatores. Os arquivos foram coletados no *site Investing*.

b3sa3.csv, bbdc4.csv, itub4.csv, petr4.csv, vale3.csv

Nome da coluna	Descrição	Tipo
DATE	Data da coleta do índice	Data
OPEN	Cotação da abertura do pregão	Numérico
HIGH	Cotação máxima atingida pelo índice na data	Numérico
LOW	Cotação mínima atingida pelo índice na data	Numérico
CLOSE	Cotação do último negócio realizado na data	Numérico
ADJ CLOSE	Ajuste realizado no fechamento do mercado	Numérico
VOLUME	Volume de negócios realizado no pregão	Numérico

3. Processamento / Tratamento de Dados

Após a coleta dos dados na *internet*, foi realizada uma verificação simples dos principais indicadores em uma plataforma de negociação disponibilizada por uma corretora para garantir que reflitam a realidade. Todo o tratamento dos dados deu-se por meio de *script Python* e suas bibliotecas. Foram utilizadas as bibliotecas *pandas*, *numpy* e *sklearn* para o tratamento, manipulação, processamento e avaliação dos resultados, as bibliotecas *seaborn, matplotlib, plotly* e *yellowbrick* para a expressão gráfica dos dados e resultados obtidos.

Foi definido um *template* padrão ao qual as fontes de dados deveriam adaptar-se que deveriam seguir a tabela abaixo:

Nome da coluna	Descrição	Tipo
DATA_PREGAO	Data de apuração da cotação	Datetime / index
[METRICA]	A métrica a ser avaliada na fonte de dados. Ex. SELIC, OURO, PETROLEO etc.	Float

Para as fontes de dados que apresentavam mais de um valor como a cotação de ações que tem valores de Abertura, Fechamento, Máxima etc., foram considerados o valor de fechamento do pregão, ou seja, o valor que foi aplicado a sua última negociação do dia.

O total de registros apresentados por todas as fontes de dados neste momento são 23.245 e por tratar-se de pregões de mesma data, foi realizada a verificação de quantidade de registros por ano, para assegurar que todos os *inputs* tenham uma quantidade similar. Os pontos observados foram que a cotação do minério de ferro apresentou apenas 48 pregões em 2010, pois iniciou seu histórico em 25/10/2010 e que para a Selic havia cotação para todos os dias do ano. Também foi observado que a quantidade de pregões realizados no exterior é superior aos realizados no Brasil, devido aos feriados.

Como entre as fontes de dados temos a sobreposição de datas, será feita a união de todas elas em um único conjunto de dados para a realização das análises, o que resultou em 2440 registros. Com a base consolidada foi verificada a existência de valores nulos apenas para o Índice Bovespa para os dias 14/02/2018, 06/03/2019 e 26/02/2020, ocorrências em que não houve apuração por tratar-se de datas comemorativas no Brasil. Considerando a baixa representatividade destes 3 dias, eles foram removidos, resultando em 2437 registros.

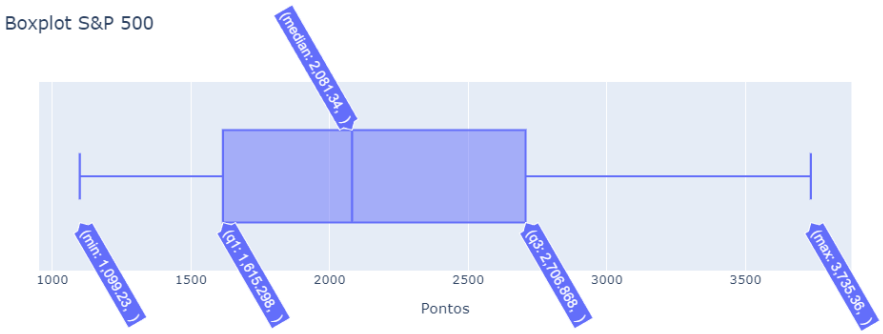
Para auxiliar na identificação de tendências nos dados foram incluídas métricas como a média móvel dos últimos 30 pregões. Foram excluídos os dados referentes ao valor futuro da cotação das métricas como o valor de fechamento do pregão e

adicionados os valores de fechamento do dia anterior para que com isso consigamos realizar as previsões.

4. Análise e Exploração dos Dados

Para a compreensão melhor das métricas utilizadas foram elaboradas representações gráficas para cada uma delas.

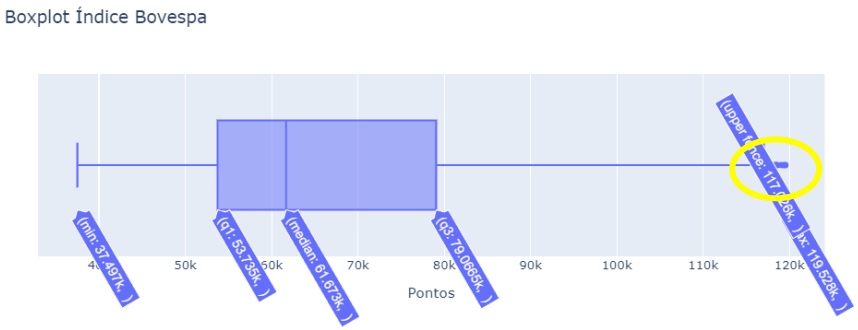
4.1. Índice S&P500



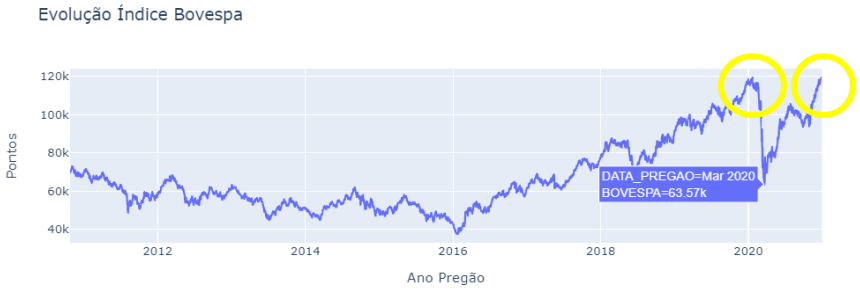
Para o SP500 não foi identificado nenhum *outlier* tendo sua valorização acontecido gradualmente com o passar do tempo conforme abaixo:

Observamos apenas uma queda abrupta em março de 2020 devido à pademia do Coronavirus, porém não gerando nenhuma ocorrência de *outlier*.

4.2. Índice Bovespa

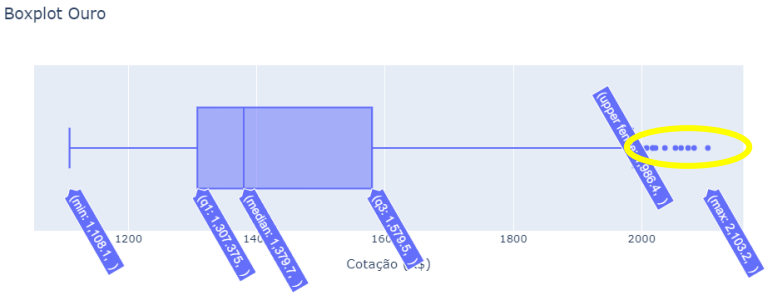


Foram identificados diversos *outliers.* A análise mostra que referem-se ao período de dezembro de 2019, janeiro e dezembro de 2020.

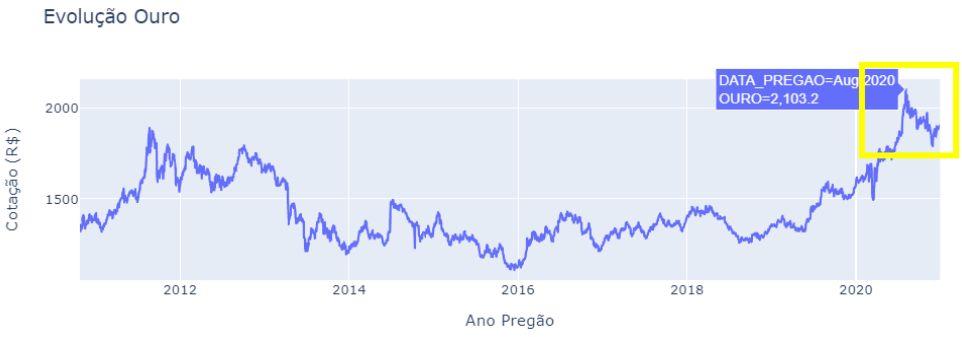


A ocorrência destes *outliers* deve-se ao crescimento acentuado do índice após 2016 e ao pico atingido em 2020 logo retraído pela ocorrência do Coronavírus, por este motivo deixaremos estes dados em nossa análise.

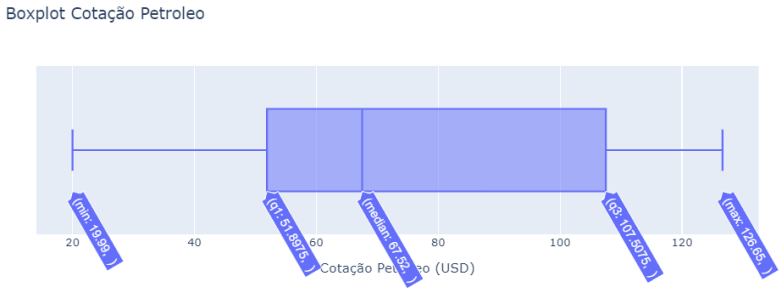
4.3. Ouro



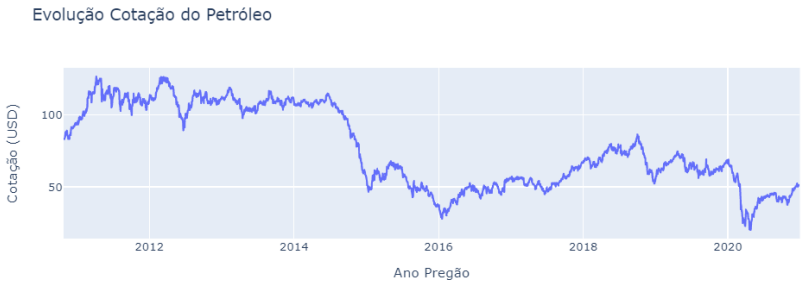
Os *outliers* que foram encontrados na cotação do ouro referem-se a 19 pregões realizados no mês de julho, agosto e setembro de 2020 quando a pandemia estava se consolidando e os investidores começando a buscar alternativas mais seguras para proteger-se das possíveis perdas nas bolsas mundiais, por este motivo não vamos alterá-los.



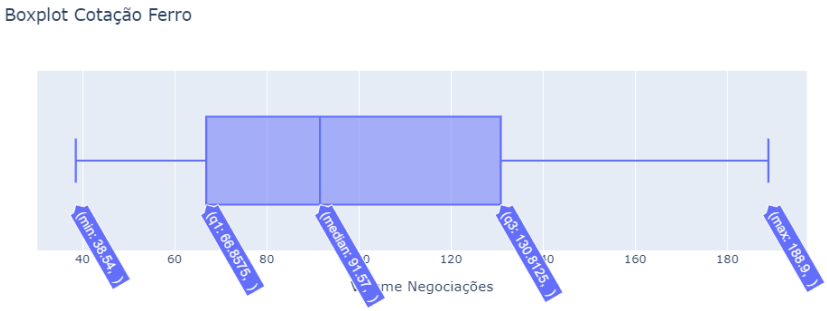
4.4. Petróleo



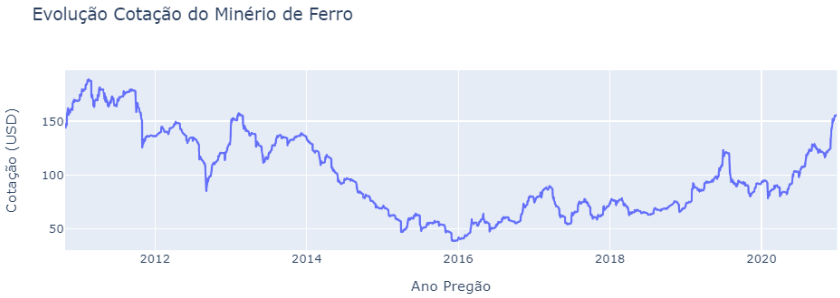
A cotação do petróleo não apresentou nenhum *outlier* e sua evolução é consistente conforme abaixo:



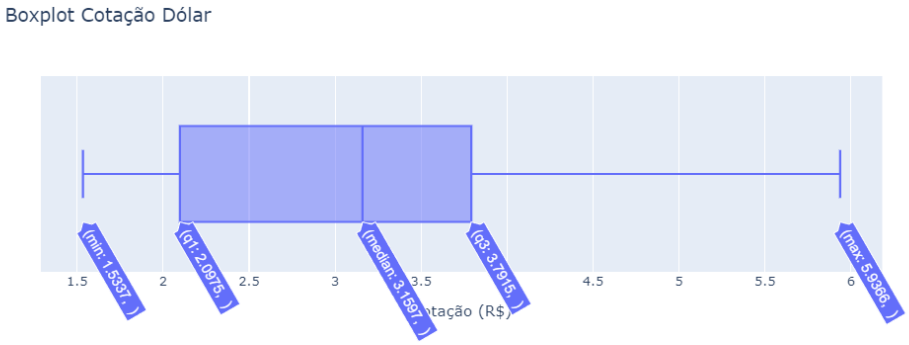
4.5. Ferro



O mesmo ocorre com a cotação do minério de ferro.

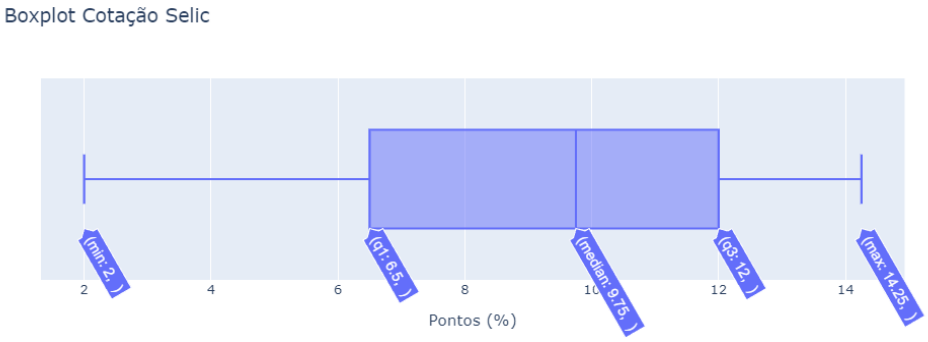


4.6. Dólar

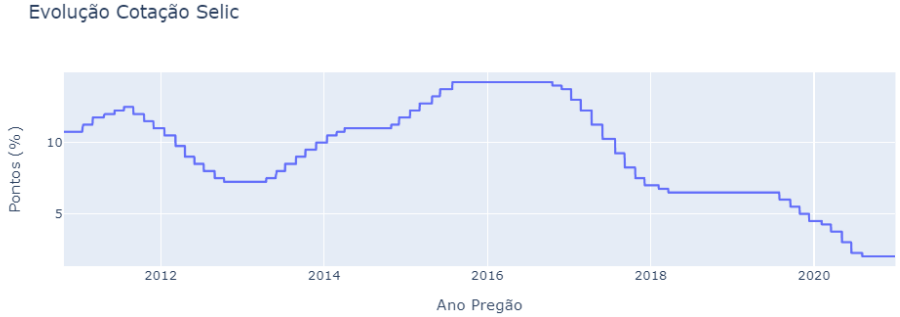


A cotação do dólar também não apresentou alteração.

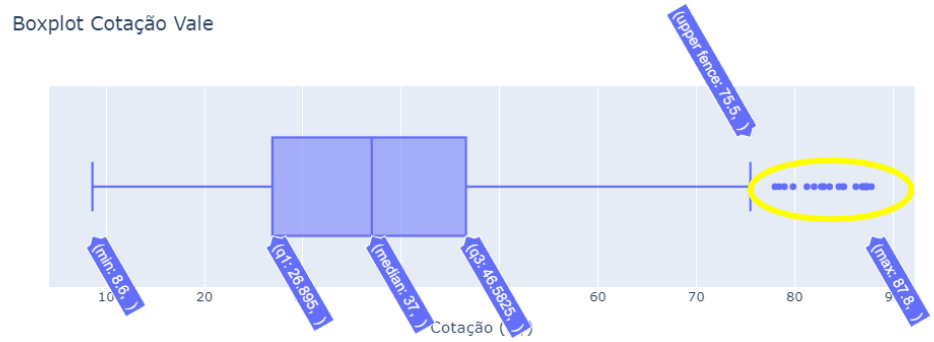
4.7. Taxa Selic



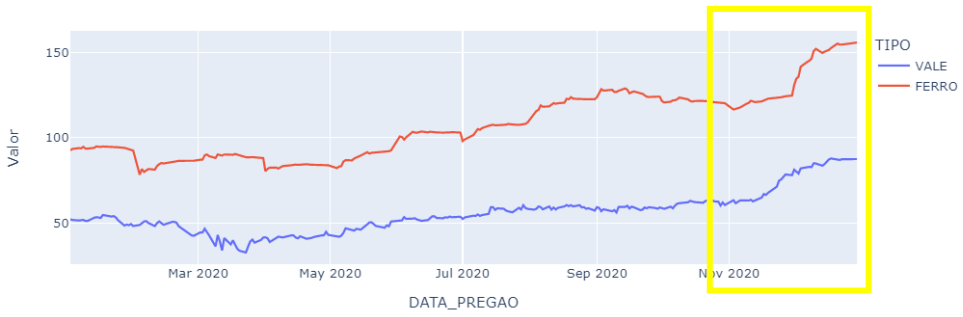
Assim como a cotação do dólar, a taxa Selic também não apresentou alteração.



4.7. Vale



Foram levantados 22 pregões que evidenciam estes *outliers*, todos ocorridos em novembro e dezembro de 2020 quando haviam fortes indícios de recuperação na China (maior cliente da Vale) que causaram uma maior demanda e por consequência a elevação do preço do minério de ferro no mundo. O gráfico abaixo mostra o período específico e a relação entre a cotação da Vale e a cotação do minério de ferro:

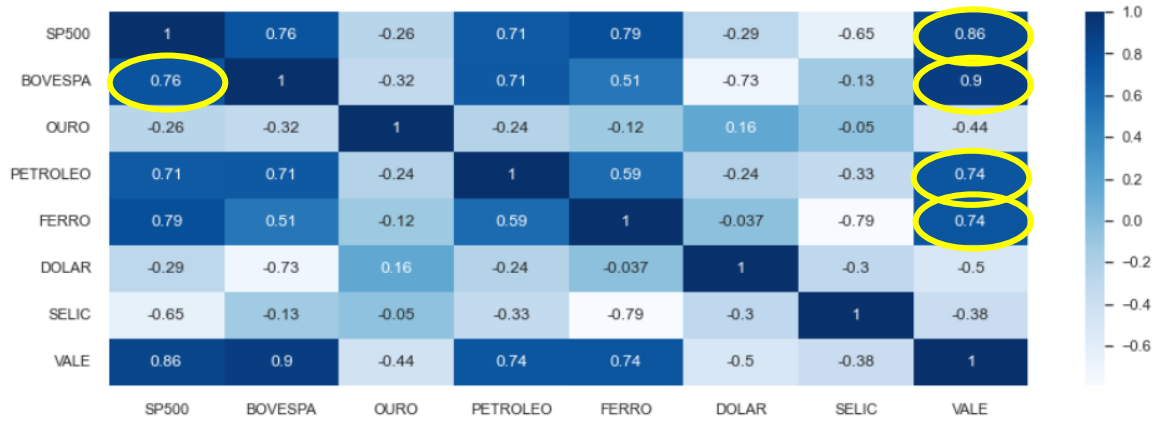


Por estes dados mostrarem consistência aos acontecimentos da época e por terem relevância no impacto em mais de um indicador, estes serão mantidos.

4.8. Correlação

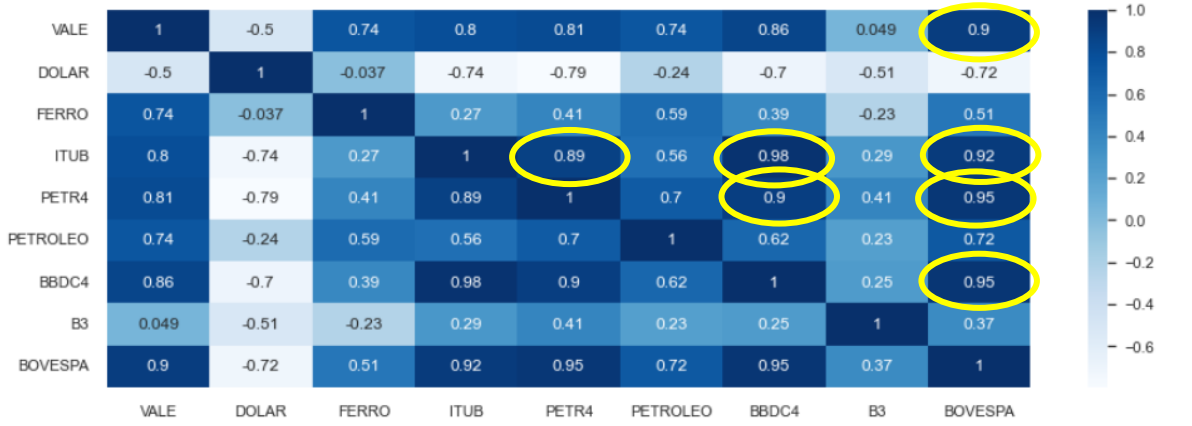
Será realizada a verificação da correlação dentre todos os fatores que foram levantados com os dados do último ano de análise.

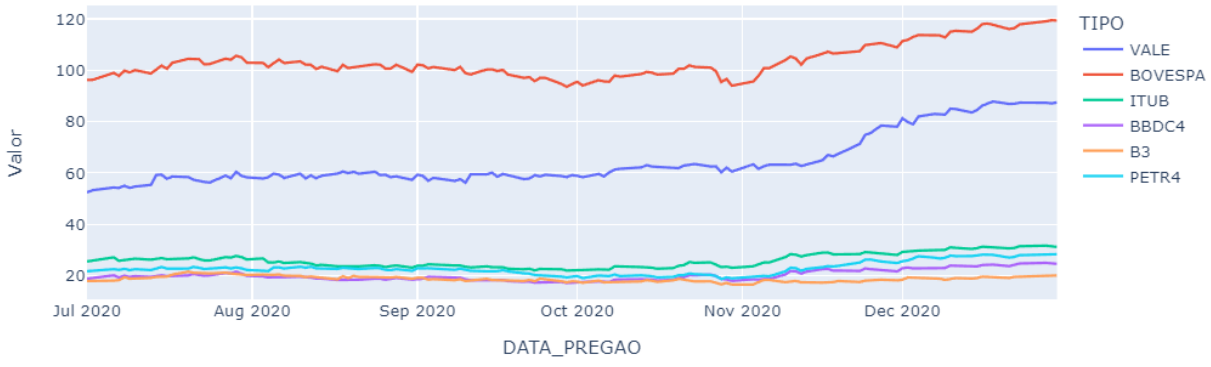
No gráfico abaixo, vemos a forte influência que a cotação do petróleo e do minério de ferro tem sobre as ações da Vale e a forte correlação que ela tem com o índice Bovespa. Fora isso o índice S&P também tem forte influência sobre sua cotação. Há uma correlação interessante também do índice Bovespa com o S&P.



Vamos também avaliar a correlação entre as cotações das top 5 empresas do índice Bovespa.

No gráfico temos a demonstração de como a relação entre as top 5 empresas da B3 estão relacionadas com o Índice Bovespa. Fora isso podemos também encontrar correlações fortes entre empresas de um mesmo segmento como Bradesco (*BBDC4*) e Itaú (*ITUB*) e até mesmo de outro segmento como Petrobrás (*PETR4*) e Bradesco (*BBDC4*) ou Itaú (*ITUB*).





5. Criação de Modelos de *Machine Learning*

As ferramentas utilizadas para a análise dos dados foram *python* e as suas biliotecas como *pandas, numpy* e *sklearn* para o tratamento, manipulação, processamento e avaliação dos resultados, as bibliotecas *seaborn, matplotlib, plotly* e *yellowbrick* para a expressão gráfica dos dados e resultados obtidos.

Em relação às *features*, foi adicionado o valor de fechamento do dia anterior e a média móvel dos 30 últimos pregões desconsiderando o valor de fechamento atual, pois para atuarmos no pregão atual ainda não teríamos seu valor de fechamento.

Com esta definição do valor de fechamento foi realizado o cálculo da coluna *target* seguindo o critério definido de trazer o investimento que desse o maior retorno dentre os 5 pilares e que este chegasse a pelo menos 5% de ganho em relação ao valor de fechamento do dia anterior caso contrário será considerado não fazer nada neste pregão. Os valores abaixo foram setados para a coluna *target*:

Valor	Descrição	% Distribuição Target
0	Não fazer nada	25,32%
1	Investir em Índice Bovespa	23,85%
2	Investir em Dólar	17,87%
3	Investir em Ouro	14,42%
4	Investir em Selic	9,04%
5	Investir em S&P 500	6,40%
6	Investir na ação (*VALE*)	3,09%

Com a coluna *target* definida foi realizada a divisão de dados de treinamento e dados de teste considerando a proporção de 70%/30%, onde para treinamento foram utilizados 1663 registros e para testes 714 registros.

Os algoritmos escolhidos para a predição foram: Árvore, *Random Forest* e kNN por serem algoritmos de classificação que levam em consideração os parametros inputados para predizer a qual classe pertencem. Como foram elencados diversos fatores de influencia e seu comportamento em um período de 30 pregões o algoritmo conseguiria basear-se neles para chegar a uma conclusão.

6. Interpretação dos Resultados

Primeiro precisamos definir o que seria um bom resultado para a predição: o bom resultado se dará pelo percentual de acurácia superior a 50%, ou seja, do total de investimentos realizados mais da metade será realizado no momento correto. Tendo isto em vista, a identificação do melhor pilar a se investir no momento será somente parte do caminho a ser percorrido pelo investidor tendo ele que apoiar-se em outros estudos como risco/retorno, percentual de retorno etc.

6.1. Árvore

Aplicando o algoritmo de Árvore obtivemos uma acurácia de quase 79% conforme foi constatado pela matriz da confusão. Verificando somente os casos em que o investidor efetivamente colocou o capital em risco apurou-se que para as 582 vezes em que aplicou em algo em 431 estava correto, nos dando uma acurácia de 74%.

Segue detalhamento das métricas apuradas:

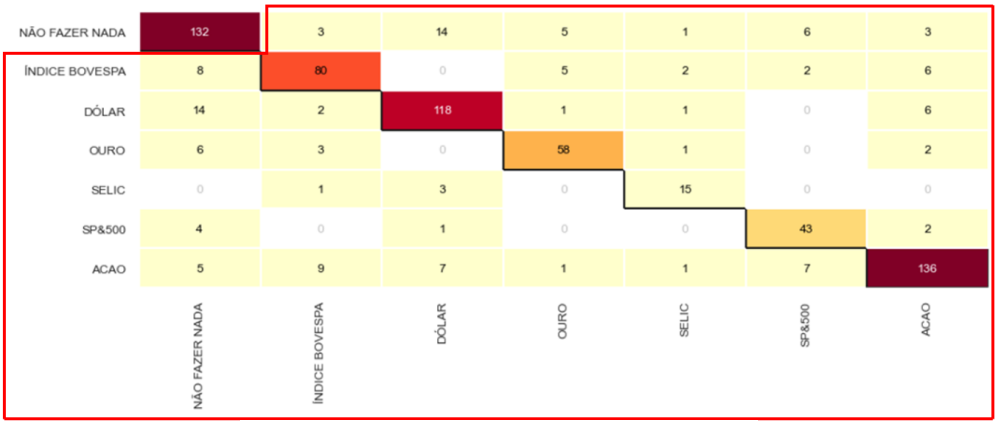
	precision	recall	f1-score	support
0	0.77	0.80	0.79	164
1	0.72	0.70	0.71	103
2	0.81	0.83	0.82	142
3	0.83	0.77	0.80	70
4	0.67	0.74	0.70	19
5	0.73	0.80	0.76	50
6	0.85	0.80	0.82	166
accuracy			0.79	714
macro avg	0.77	0.78	0.77	714
weighted avg	0.79	0.79	0.79	714

6.2. kNN

Utilizando o algoritmo kNN na identificação do momento oportuno e pilar de investimento obtivemos uma acurácia geral de 81% e das 582 vezes em que o investidor colocou seu capital em risco, 450 deram o retorno esperado, ou seja, 77% de acurácia.

Segue resultado apontado para o algoritmo:

0.8151260504201681



	precision	recall	f1-score	support
0	0.79	0.84	0.82	164
1	0.85	0.77	0.81	103
2	0.87	0.87	0.87	142
3	0.84	0.81	0.83	70
4	0.74	0.74	0.74	19
5	0.77	0.82	0.80	50
6	0.87	0.87	0.87	166
accuracy			0.84	714
macro avg	0.82	0.82	0.82	714
weighted avg	0.84	0.84	0.84	714

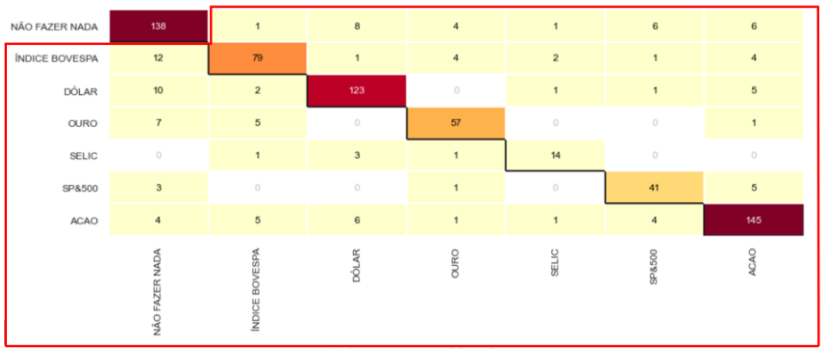
6.3. *Random Forest*

Para o algoritmo *Random Forest* temos que adequar a quantidade de estimadores que serão aplicados aos dados. Começamos com apenas 1, porém a precisão ficou em 73%. Foi aumentado este parâmetro de 10 em 10 estimadores até chegarmos ao percentual de acerto de mais de 83% que se deu com 40 estimadores.

Seguindo o mesmo racional, foi o algoritmo que nos deu a maior assertividade dentre os três testados, das 576 vezes em que o investidor colocou o seu capital em risco, em 459 ele acertou o momento para investir, ou seja, 79% de acurácia.

Seguem dados para verificação:

0.8361344537815126



	precision	recall	f1-score	support
0	0.79	0.84	0.82	164
1	0.85	0.77	0.81	103
2	0.87	0.87	0.87	142
3	0.84	0.81	0.83	70
4	0.74	0.74	0.74	19
5	0.77	0.82	0.80	50
6	0.87	0.87	0.87	166
accuracy			0.84	714
macro avg	0.82	0.82	0.82	714
weighted avg	0.84	0.84	0.84	714

7. Apresentação dos Resultados

Title: Sugestionamento de Pilar / Momento de Investimento

Problem Statement
What problem are you trying to solve?
What larger issues do the problem address?

Divulgações pretensiosas de investimentos alavancando efeitos manada.

Hora inoportuna para a realização de investimento.

Perda de capital ocasionada pela falta de conhecimento dos investidores de mais de um pilar de investimentos.

Outcomes/Predictions
What prediction(s) are you trying to make?
Identify applicable predictor (X) and/or target (y) variables.

Identificar o pilar de investimentos que dará o melhor retorno em um prazo de 30 pregões.

Reduzir o risco de perda de capital.

Data Acquisition
Where are you sourcing your data from?
Is there enough data? Can you work with it?

Sites oficiais: Banco Central do Brasil (Taxa de Câmbio: Dólar e taxa Selic)

Sites especializados verificados em plataforma de investimentos oficiais: *Yahoo! Finance*; Índice Bovespa e Índice S&P 500; *Investing*: cotações de minério de ferro, petróleo *brent* futuro, ouro, Vale, Bradesco, Itaú, B3 e Petrobrás.

Modeling
What models are appropriate to use given your outcomes?

Árvore

kNN

Random Forest

Model Evaluation
How can you evaluate your model's performance?

A *performance* será verificada por meio da acurácia, pois essa nos dirá se a operação realizada foi a correta. Para nos apresentar um resultado consistente ela deverá ser superior a 50%.

Data Preparation
What do you need to do to your data in order to run your model and achieve your outcomes?

Limpeza dos dados.
Seleção de métricas.
Ajuste de períodos (dados do passado apenas).
Adição de médias móveis.
Merge dentre as diversas fontes.
Padronização dos dados.

8. Links

Link para o vídeo: https://www.youtube.com/watch?v=aJmMW_ZHHeM

Link para o repositório: https://github.com/rafaelsoaresmelero/MBA_PUC

Taxa Selic: https://www.bcb.gov.br/controleinflacao/historicotaxasjuros

Taxa Selic: https://www.infomoney.com.br/guias/taxa-selic/

Taxa Cambio: Dólar https://www.bcb.gov.br/estabilidadefinanceira/historicocotacoes

Minério de ferro refinado https://br.investing.com/commodities/iron-ore-62-cfr-futures

Petróleo Brent Futuros https://br.investing.com/commodities/brent-oil-historical-data

Petróleo Brent Futuros https://www.ipea.gov.br/desafios/index.php?option=com_content&view=article&id=2083:catid=28&Itemid=23

Ouro https://br.investing.com/commodities/gold-historical-data

Índice Bovespa https://finance.yahoo.com/quote/%5EBVSP?p=^BVSP&.tsrc=fin-srch

Índice Bovespa https://www.b3.com.br/pt_br/market-data-e-indices/indices/indices-amplos/ibovespa.htm

Índice S&P 500 https://finance.yahoo.com/quote/%5EGSPC?p=^GSPC&.tsrc=fin-srch

Recuperação da China Dezembro 2020 https://www.moneytimes.com.br/acoes-da-china-fecham-em-alta-com-otimismo-sobre-promessa-de-sustentar-a-recuperacao/

APÊNDICE

Programação/Scripts

TCC MBA PUC.py

```
!pip install pandas
!pip install seaborn --upgrade
!pip install sklearn --upgrade
!pip install scikit-learn --upgrade
!pip install plotly --upgrade
!pip install -U kaleido

#Bibliotecas padrão
import numpy as np
import pandas as pd

#Bibliotecas gráficas
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

#Preparação dos dados
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

#Bibliotecas de Algoritmos
from sklearn.tree import DecisionTreeClassifier #Árvore de decisão
from sklearn.ensemble import RandomForestClassifier #Random Forest
from sklearn.neighbors import KNeighborsClassifier #kNN

#Avaliação de algoritmo
from sklearn import tree
from yellowbrick.classifier import ConfusionMatrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

#Configurando as biblioteca
##Seaborn
sns.set_theme(style="darkgrid")
sns.set(rc = {'figure.figsize':(15,5)})
##Pandas - Mostrar todas as colunas do Dataframe
```

```
pd.set_option("display.max_columns", None)

"""## Importação e tratativa inicial dos dados"""

def importaInvesting(df_resultado, nomemetrica):
    #Padronizando a nomenclatura das colunas
    df_resultado.rename(columns={'Data' : 'DATA_PREGAO', 'Último' : nomemetrica}, inplace = True)

    #Removendo colunas desnecessárias
    df_resultado.drop(['Abertura', 'Máxima', 'Mínima', 'Vol.', 'Var%'], inplace=True, axis=1)

    #Padronizando a coluna DATA_PREGAO
    df_resultado['DATA_PREGAO'] = df_resultado['DATA_PREGAO'].apply(lambda x: x.replace('.', '/'))
    df_resultado['DATA_PREGAO'] = pd.to_datetime(df_resultado['DATA_PREGAO'], format="%d/%m/%Y")

    #Padronizando a coluna de valor
    df_resultado[nomemetrica] = df_resultado[nomemetrica].apply(lambda x: x.replace('.', '').replace(',', '.'))
    df_resultado[nomemetrica] = pd.to_numeric(df_resultado[nomemetrica], downcast='float')

    #Criando índice
    df_resultado.set_index('DATA_PREGAO', inplace=True)
    return df_resultado

"""### Taxas

#### Taxa Selic
"""

selic = pd.read_csv('dados/Selic.csv',sep=';')

selic.dtypes

selic.head(1)

#Padronizando a nomenclatura das colunas
selic.rename(columns={'Data' : 'DATA_PREGAO', 'Valor' : 'SELIC'}, inplace = True)

#Padronizando a coluna DATA_PREGAO
selic['DATA_PREGAO'] = pd.to_datetime(selic['DATA_PREGAO'], format="%d/%m/%Y")

#Padronizando a coluna de valor
selic['SELIC'] = selic['SELIC'].apply(lambda x: x.replace(',', '.'))
selic['SELIC'] = pd.to_numeric(selic['SELIC'], downcast='float')
```

```
#Criando índice
selic.set_index('DATA_PREGAO', inplace=True)

selic.dtypes, type(selic.index)

selic.head(1)

"""#### Taxa Câmbio: Dólar"""

dolar = pd.read_csv('dados/Dolar.csv', sep=';',dtype='str')

dolar.dtypes

dolar.head(1)

#Padronizando a nomenclatura das colunas
dolar.rename(columns={'DATA' : 'DATA_PREGAO', 'COMPRA' : 'DOLAR'}, inplace = True)

#Removendo colunas desnecessárias
dolar.drop(['COLUNA1', 'COLUNA2', 'TIPO', 'VENDA', 'COLUNA3', 'COLUNA4', 'COLUNA5'], inplace=True, axis=1)

#Padronizando a coluna DATA_PREGAO
dolar['DATA_PREGAO'] = dolar['DATA_PREGAO'].apply(lambda x: x[0:2] + '/' + x[2:4] + '/' + x[4:])
dolar['DATA_PREGAO'] = pd.to_datetime(dolar['DATA_PREGAO'], format="%d/%m/%Y")

#Padronizando a coluna de valor
dolar['DOLAR'] = dolar['DOLAR'].apply(lambda x: x.replace(',', '.'))
dolar['DOLAR'] = pd.to_numeric(dolar['DOLAR'], downcast='float')

#Criando índice
dolar.set_index('DATA_PREGAO', inplace=True)

dolar.dtypes, type(dolar.index)

dolar.head(1)

"""### Commodities

#### Minério de Ferro Refinado
"""
```

```
ferro = pd.read_csv('dados/Minerio_Ferro.csv', sep=',')

ferro.dtypes

ferro.head(1)

ferro = importaInvesting(ferro, "FERRO")

ferro.dtypes, type(ferro.index)

ferro.head(1)

"""#### Petróleo Brent Futuros"""

petroleo = pd.read_csv('dados/Petroleo_Brent.csv', sep=',')

petroleo.dtypes

petroleo.head(1)

petroleo = importaInvesting(petroleo, "PETROLEO")

petroleo.dtypes

petroleo.head(1)

"""#### Ouro"""

ouro = pd.read_csv('dados/Ouro.csv', sep=',')

ouro.dtypes

ouro.head(1)

ouro = importaInvesting(ouro, "OURO")

ouro.dtypes

ouro.head(1)

"""### Índices

#### Índice Bovespa
```

```
"""

bovespa = pd.read_csv('dados/Indice_Bovespa.csv')

bovespa.dtypes

bovespa.head(1)

#Padronizando a nomenclatura das colunas
bovespa.rename(columns={'Date' : 'DATA_PREGAO', 'Close' : 'BOVESPA'}, inplace = True)

#Removendo colunas desnecessárias
bovespa.drop(['Open', 'High', 'Low', 'Adj Close', 'Volume'], inplace=True, axis=1)

#Padronizando a coluna DATA_PREGAO
bovespa['DATA_PREGAO'] = pd.to_datetime(bovespa['DATA_PREGAO'])

#Padronizando a coluna de VALOR
bovespa['BOVESPA'] = bovespa['BOVESPA'].apply(lambda x: float(x))

#Criando índice
bovespa.set_index('DATA_PREGAO', inplace=True)

bovespa.dtypes

bovespa.head(1)

"""#### Índice S&P 500"""

sp500 = pd.read_csv('dados/Indice_S&P500.csv')

sp500.dtypes

sp500.head(1)

#Padronizando a nomenclatura das colunas
sp500.rename(columns={'Data' : 'DATA_PREGAO', 'Último' : 'SP500'}, inplace = True)

#Removendo colunas desnecessárias
sp500.drop(['Abertura', 'Máxima', 'Mínima', 'Var%', 'Vol.'], inplace=True, axis=1)

#Padronizando a coluna DATA_PREGAO
sp500['DATA_PREGAO'] = sp500['DATA_PREGAO'].apply(lambda x: x.replace('.', '/'))
```

```
sp500['DATA_PREGAO'] = pd.to_datetime(sp500['DATA_PREGAO'], format="%d/%m/%Y")

#Padronizando a coluna de VALOR
sp500['SP500'] = sp500['SP500'].apply(lambda x: float(x.replace('.', '').replace(',', '.')))

#Criando índice
sp500.set_index('DATA_PREGAO', inplace=True)

sp500.dtypes

sp500.head(1)

"""### Ações"""

def importaAcoes(df_resultado, nomemetrica):
    #Padronizando a nomenclatura das colunas
    df_resultado.rename(columns={'Data' : 'DATA_PREGAO', 'Último' : nomemetrica}, inplace = True)

    #Removendo colunas desnecessárias
    df_resultado.drop(['Abertura', 'Máxima', 'Mínima', 'Vol.', 'Var%'], inplace=True, axis=1)

    #Padronizando a coluna DATA_PREGAO
    df_resultado['DATA_PREGAO'] = df_resultado['DATA_PREGAO'].apply(lambda x: x.replace('.', '/'))
    df_resultado['DATA_PREGAO'] = pd.to_datetime(df_resultado['DATA_PREGAO'], format="%d/%m/%Y")

    #Padronizando a coluna de valor
    df_resultado[nomemetrica] = df_resultado[nomemetrica].apply(lambda x: x.replace(',', '.'))
    df_resultado[nomemetrica] = pd.to_numeric(df_resultado[nomemetrica], downcast='float')

    #Criando índice
    df_resultado.set_index('DATA_PREGAO', inplace=True)
    return df_resultado

"""#### VALE3 - VALE"""

vale = pd.read_csv('dados/vale3.csv')

vale.dtypes

vale.head(1)

vale = importaAcoes(vale, "VALE")
```

```
vale.dtypes

vale.head(1)

"""#### ITUB4 - ITAU UNIBANCO"""

itub = pd.read_csv('dados/itub4.csv')

itub.dtypes

itub.head(1)

itub = importaAcoes(itub, "ITUB")

itub.dtypes

itub.head(1)

"""#### PETR4 - PETROBRÁS"""

petr4 = pd.read_csv('dados/petr4.csv')

petr4.dtypes

petr4.head(1)

petr4 = importaAcoes(petr4, "PETR4")

petr4.dtypes

petr4.head(1)

"""#### BBDC4 - BRADESCO"""

bbdc4 = pd.read_csv('dados/bbdc4.csv')

bbdc4.dtypes

bbdc4.head(1)

bbdc4 = importaAcoes(bbdc4, "BBDC4")

bbdc4.dtypes
```

```
bbdc4.head(1)

"""#### B3SA3 - B3 SA"""

b3 = pd.read_csv('dados/b3sa3.csv')

b3.dtypes

b3.head(1)

b3 = importaAcoes(b3, "B3")

b3.dtypes

b3.head(1)

"""## Unindo os Datasets"""

# Exportando os datasets de entrada
sp500.to_csv('saida/01_sp500.csv')
bovespa.to_csv('saida/01_bovespa.csv')
ouro.to_csv('saida/01_ouro.csv')
petroleo.to_csv('saida/01_petroleo.csv')
ferro.to_csv('saida/01_ferro.csv')
dolar.to_csv('saida/01_dolar.csv')
selic.to_csv('saida/01_selic.csv')

sp500.count(), bovespa.count(), ouro.count(), petroleo.count(), ferro.count(), dolar.count(), selic.count(), vale.count()

#Verificando a contagem dos pregões por indicador foi verificado que o Minério de Ferro não tem tantos pregões disponíveis para o ano de 2010
sp500.groupby(sp500.index.year).agg({'count'}).join(bovespa.groupby(bovespa.index.year).agg({'count'})).join(ouro.groupby(ouro.index.year).agg({'count'})).join(petroleo.groupby(petroleo.index.year).agg({'count'})).join(ferro.groupby(ferro.index.year).agg({'count'})).join(dolar.groupby(dolar.index.year).agg({'count'})).join(selic.groupby(selic.index.year).agg({'count'})).join(vale.groupby(vale.index.year).agg({'count'}))

#O Minério de Ferro só apresenta dados a partir de 25/10/2010
ferro.index.min()

# Unindo todos os arquivos importados: sp500, bovespa, ouro, petroleo, ferro, dolar, selic
df = sp500.join(bovespa, how="inner")
```

```
df = df.join(ouro, how="inner")
df = df.join(petroleo, how="inner")
df = df.join(ferro, how="inner")
df = df.join(dolar, how="inner")
df = df.join(selic, how="inner")
df = df.join(vale, how="inner")

#Ordenar dataframe pelo indice
df.sort_index(axis=0, ascending=True, inplace=True)

#Organizando o index para auxiliar no cálculo de métricas complementares
df

#Verificação de valores nulos nos itens do dataset
df["SP500"].hasnans, df["BOVESPA"].hasnans, df["OURO"].hasnans, df["PETROLEO"].hasnans

df["FERRO"].hasnans, df["DOLAR"].hasnans, df["SELIC"].hasnans, df["VALE"].hasnans

#Dias em que os dados do Bovespa não foram informados
#Foi constatado que se trata da quarta-feira de cinzas em que os demais indicadores tem cotação por não serem negociados na B3
df.query("BOVESPA.isnull()")

"""Devido a não haver negociações na B3 em feriados, a listagem abaixo justifica os dias em que apareceram como nulo no dataset:
- 2020-02-26: Quarta-feira de cinzas
- 2019-03-06: Quarta-feira de cinzas
- 2018-02-14: Quarta-feira de cinzas
"""

df.shape

#Por se tratarem de poucos dias e em que não há negociação na B3 em são paulo vamos remover estas datas
df.dropna(subset = ["BOVESPA"], inplace=True)

#Verificando seainda há valores nulos no dataframe
df["SP500"].hasnans, df["BOVESPA"].hasnans, df["OURO"].hasnans, df["PETROLEO"].hasnans

df["FERRO"].hasnans, df["DOLAR"].hasnans, df["SELIC"].hasnans, df["VALE"].hasnans

#Verificando métricas com valores zerados
df.query("SP500 == 0").SP500.count(), df.query("BOVESPA == 0").BOVESPA.count(), df.query("OURO == 0").OURO.count()
```

```
#Verificando métricas com valores zerados
df.query("PETROLEO == 0").PETROLEO.count(), df.query("FERRO == 0").FERRO.count(), df.query("DOLAR == 0").DOLAR.count()

df.query("SELIC == 0").SELIC.count(), df.query("VALE == 0").VALE.count()

#Verificando chaves duplicadas no dataframe
df.index.duplicated().sum()

df.head(5)

df.shape

df.describe()

df.info()

"""## Exploração dos dados

### Verificando Outliers

#### SP500
"""

#Verificando os quartis do valor
fig = px.box(df, x="SP500", title="Boxplot S&P 500")
fig.update_layout(showlegend=True, xaxis_title="Pontos")
fig.show()

#Verificando range de valores
fig = px.histogram(df, x="SP500", title='Histograma S&P 500')
fig.update_layout(showlegend=True, yaxis_title="Qtde. Ocorrências", xaxis_title="Pontos")
fig.show()

"""#### BOVESPA"""

fig = px.box(df, x="BOVESPA", title="Boxplot Índice Bovespa")
fig.update_layout(showlegend=True, xaxis_title="Pontos")
fig.show()

#Verificando range de valores
fig = px.histogram(df, x="BOVESPA", title='Histograma Índice Bovespa')
```

```
fig.update_layout(showlegend=True, yaxis_title="Qtde. Ocorrências", xaxis_title="Pontos")
fig.show()

#Verificar período de pico do Índice Bovespa
fig = px.line(df, x=df.index, y="BOVESPA", title='Evolução Índice Bovespa')
fig.update_layout(showlegend=True, yaxis_title="Pontos", xaxis_title="Ano Pregão")
fig.show()

np.unique(df.query("BOVESPA > 117026").index.year), np.count_nonzero(np.unique(df.query("BOVESPA > 117026").index.date)), df.query("BOVESPA > 117026").index.date

"""#### OURO

"""

fig = px.box(df, x="OURO", title="Boxplot Ouro")
fig.update_layout(showlegend=True, xaxis_title="Cotação (R$)")
fig.show()

#Geralmente a busca pelo ouro decorre da busca de aplicações consideradas mais seguras como metais preciosos
np.unique(df.query("OURO > 1986.4").index.year), np.unique(df.query("OURO > 1986.4").index), np.count_nonzero(np.unique(df.query("OURO > 1986.4").index.date))

#Verificando range de valores
fig = px.histogram(df, x="OURO", title='Histograma Ouro')
fig.update_layout(showlegend=True, yaxis_title="Qtde. Ocorrências", xaxis_title="Cotação (R$)")
fig.show()

#Verificar período de pico do valor
fig = px.line(df, x=df.index, y="OURO", title='Evolução Ouro')
fig.update_layout(showlegend=True, yaxis_title="Cotação (R$)", xaxis_title="Ano Pregão")
fig.show()

"""#### PETROLEO"""

fig = px.box(df, x="PETROLEO", title="Boxplot Cotação Petroleo")
fig.update_layout(showlegend=True, xaxis_title="Cotação Petróleo (USD)")
fig.show()

#Verificando range de valores
fig = px.histogram(df, x="PETROLEO", title='Histograma da Cotação do Petróleo')
fig.update_layout(showlegend=True, yaxis_title="Qtde. Ocorrências", xaxis_title="Cotação Petróleo (USD)")
```

```
fig.show()

#Verificar período de pico do valor
fig = px.line(df, x=df.index, y="PETROLEO", title='Evolução Cotação do Petróleo')
fig.update_layout(showlegend=True, yaxis_title="Cotação (USD)", xaxis_title="Ano Pregão")
fig.show()

"""#### FERRO"""

fig = px.box(df, x="FERRO", title="Boxplot Cotação Ferro")
fig.update_layout(showlegend=True, xaxis_title="Volume Negociações")
fig.show()

#Verificando range de valores
fig = px.histogram(df, x="FERRO", title='Histograma da Cotação do Minério de Ferro')
fig.update_layout(showlegend=True, yaxis_title="Qtde. Ocorrências", xaxis_title="Cotação (USD)")
fig.show()

#Verificar período de pico do valor
fig = px.line(df, x=df.index, y="FERRO", title='Evolução Cotação do Minério de Ferro')
fig.update_layout(showlegend=True, yaxis_title="Cotação (USD)", xaxis_title="Ano Pregão")
fig.show()

"""#### DOLAR"""

fig = px.box(df, x="DOLAR", title="Boxplot Cotação Dólar")
fig.update_layout(showlegend=True, xaxis_title="Cotação (R$)")
fig.show()

#Verificando range de valores
fig = px.histogram(df, x="DOLAR", title='Histograma da Cotação Dólar')
fig.update_layout(showlegend=True, yaxis_title="Qtde. Ocorrências", xaxis_title="Cotação (R$)")
fig.show()

#Verificar período de pico do valor
fig = px.line(df, x=df.index, y="DOLAR", title='Evolução Cotação Dólar')
fig.update_layout(showlegend=True, yaxis_title="Cotação (R$)", xaxis_title="Ano Pregão")
fig.show()

"""#### SELIC"""

fig = px.box(df, x="SELIC", title="Boxplot Cotação Selic")
fig.update_layout(showlegend=True, xaxis_title="Pontos (%)")
```

```
fig.show()

#Verificando range de valores
fig = px.histogram(df, x="SELIC", title='Histograma da Cotação Selic')
fig.update_layout(showlegend=True, yaxis_title="Qtde. Ocorrências", xaxis_title="Pontos (%)")
fig.show()

#Verificar período de pico do valor
fig = px.line(df, x=df.index, y="SELIC", title='Evolução Cotação Selic')
fig.update_layout(showlegend=True, yaxis_title="Pontos (%)", xaxis_title="Ano Pregão")
fig.show()

"""#### VALE"""

fig = px.box(df, x="VALE", title="Boxplot Cotação Vale")
fig.update_layout(showlegend=True, xaxis_title="Cotação (R$)")
fig.show()

#Verificando range de valores
fig = px.histogram(df, x="VALE", title='Histograma da Cotação Vale')
fig.update_layout(showlegend=True, yaxis_title="Qtde. Ocorrências", xaxis_title="Cotação (R$)")
fig.show()

#Verificar período de pico do valor
fig = px.line(df, x=df.index, y="VALE", title='Evolução Cotação Vale')
fig.update_layout(showlegend=True, yaxis_title="Cotação (R$)", xaxis_title="Ano Pregão")
fig.show()

np.unique(df.query("VALE > 75.50").index.year), np.unique(df.query("VALE > 75.50").index),
np.count_nonzero(np.unique(df.query("VALE > 75.50").index.date))

#Mostrando a relação entre o minério de ferro e a ação da Vale
df_vale_ferro = df.loc[(df.index >= '2020-01-01') & (df.index < '2021-04-10')]
df_vale_ferro.reset_index(inplace=True)
df_vale_ferro = pd.melt(df_vale_ferro, id_vars=['DATA_PREGAO'], value_vars=['VALE', 'FERRO'])
df_vale_ferro.rename(columns={'variable' : 'TIPO', 'value' : 'Valor'}, inplace = True)
df_vale_ferro.set_index('DATA_PREGAO', inplace=True)
fig = px.line(df_vale_ferro, x=df_vale_ferro.index, y="Valor", color='TIPO')
fig.show()

"""### Correlação Ações com Fatores de influência"""

corr_vale = vale.join(dolar)
```

```
corr_vale = corr_vale.join(ferro)
corr_vale = corr_vale.join(itub)
corr_vale = corr_vale.join(petr4)
corr_vale = corr_vale.join(petroleo)
corr_vale = corr_vale.join(bbdc4)
corr_vale = corr_vale.join(b3)
corr_vale = corr_vale.join(bovespa)

corr_vale['PETROLEO'] = corr_vale.PETROLEO * corr_vale.DOLAR
corr_vale['FERRO'] = corr_vale.FERRO * corr_vale.DOLAR

corr_vale = corr_vale.loc[(corr_vale.index >= '2020-07-01') & (corr_vale.index < '2020-12-31')]
corr_vale = corr_vale.dropna()

corr_vale.corr()

sns.heatmap(corr_vale.corr(), annot=True, cmap="Blues" )

df_corr = df.loc[(df.index >= '2020-07-01') & (df.index < '2020-12-31')]
df_corr['PETROLEO'] = df_corr.PETROLEO * df_corr.DOLAR
df_corr['FERRO'] = df_corr.FERRO * df_corr.DOLAR
sns.heatmap(df_corr.corr(), annot=True, cmap="Blues")

corr_vale.reset_index(inplace=True)
corr_vale['BOVESPA'] = corr_vale.BOVESPA/1000
df_corr = pd.melt(corr_vale, id_vars=['DATA_PREGAO'], value_vars=['VALE', 'BOVESPA', 'ITUB', 'BBDC4', 'B3',
'PETR4'])
df_corr.rename(columns={'variable' : 'TIPO', 'value' : 'Valor'}, inplace = True)
df_corr.set_index('DATA_PREGAO', inplace=True)
fig = px.line(df_corr, x=df_corr.index, y="Valor", color='TIPO')
fig.show()

"""## Feature engineering

### Métricas para cálculo
"""

#Criando médias móveis para os dados
df['SP500_MM_30'] = df.SP500.rolling(30).mean().shift()
df['BOVESPA_MM_30'] = df.BOVESPA.rolling(30).mean().shift()
df['OURO_MM_30'] = df.OURO.rolling(30).mean().shift()
df['PETROLEO_MM_30'] = df.PETROLEO.rolling(30).mean().shift()
df['FERRO_MM_30'] = df.FERRO.rolling(30).mean().shift()
```

```
df['DOLAR_MM_30'] = df.DOLAR.rolling(30).mean().shift()
df['SELIC_MM_30'] = df.SELIC.rolling(30).mean().shift()
df['BOVESPA_MM_30'] = df.BOVESPA.rolling(30).mean().shift()
df['VALE_MM_30'] = df.VALE.rolling(30).mean().shift()

#Setar o fechamento do dia anterior
df['SP500_FECHAMENTO_ANTERIOR'] = df.SP500.shift(1)
df['BOVESPA_FECHAMENTO_ANTERIOR'] = df.BOVESPA.shift(1)
df['OURO_FECHAMENTO_ANTERIOR'] = df.OURO.shift(1)
df['PETROLEO_FECHAMENTO_ANTERIOR'] = df.PETROLEO.shift(1)
df['FERRO_FECHAMENTO_ANTERIOR'] = df.FERRO.shift(1)
df['DOLAR_FECHAMENTO_ANTERIOR'] = df.DOLAR.shift(1)
df['SELIC_FECHAMENTO_ANTERIOR'] = df.SELIC.shift(1)
df['BOVESPA_FECHAMENTO_ANTERIOR'] = df.BOVESPA.shift(1)
df['VALE_FECHAMENTO_ANTERIOR'] = df.VALE.shift(1)

#Para setar o resultado esperado criaremos estas variáveis para verificação
df['SP500_RESULT_30'] = df.SP500.shift(-30)
df['BOVESPA_RESULT_30'] = df.BOVESPA.shift(-30)
df['OURO_RESULT_30'] = df.OURO.shift(-30)
df['PETROLEO_RESULT_30'] = df.PETROLEO.shift(-30)
df['FERRO_RESULT_30'] = df.FERRO.shift(-30)
df['DOLAR_RESULT_30'] = df.DOLAR.shift(-30)
df['SELIC_RESULT_30'] = df.SELIC.shift(-30)
df['VALE_RESULT_30'] = df.VALE.shift(-30)

#Criando métricas condicionais para medir o retorno mínimo esperado e o maior retorno dentre os pilares de investimento
df['SP500_RESULT_DIFF'] = ((df.SP500_RESULT_30 / df.SP500_FECHAMENTO_ANTERIOR) - 1) * 100
df['BOVESPA_RESULT_DIFF'] = ((df.BOVESPA_RESULT_30 / df.BOVESPA_FECHAMENTO_ANTERIOR) - 1) * 100
df['OURO_RESULT_DIFF'] = ((df.OURO_RESULT_30 / df.OURO_FECHAMENTO_ANTERIOR) - 1) * 100
df['DOLAR_RESULT_DIFF'] = ((df.DOLAR_RESULT_30 / df.DOLAR_FECHAMENTO_ANTERIOR) - 1) * 100
df['SELIC_RESULT_DIFF'] = ((df.SELIC_RESULT_30 / df.SELIC_FECHAMENTO_ANTERIOR) - 1) * 100
df['VALE_RESULT_DIFF'] = ((df.VALE_RESULT_30 / df.VALE_FECHAMENTO_ANTERIOR) - 1) * 100

#Devido à inclusão das métricas acima, algumas colunas apresentarão os dados nulos para os primeiros 30 registros e para outras colunas nos últimos registros
df.isnull().values.any(), df.shape

#Faremos a exclusão delas
df = df.dropna()
```

```
df.shape

"""### Calculando o resultado esperado
0 - Não fazer nada
1 - Aplicar no índice Bovespa
2 - Aplicar em Dólar
3 - Aplicar em Ouro
4 - Aplicar em Selic
5 - ETF SP&500
6 - Ações (VALE)
"""

# Gerar a métrica de resultados de acordo com os itens que foram definidos
condicoes = [
(df['BOVESPA_RESULT_DIFF'] > df['SP500_RESULT_DIFF']) & (df['BOVESPA_RESULT_DIFF'] > df['OURO_RESULT_DIFF'])
  & (df['BOVESPA_RESULT_DIFF'] > df['DOLAR_RESULT_DIFF']) & (df['BOVESPA_RESULT_DIFF'] > df['SELIC_RESULT_DIFF'])
  & (df['BOVESPA_RESULT_DIFF'] > df['VALE_RESULT_DIFF']) & (df['BOVESPA_RESULT_DIFF'] > 5)
, (df['DOLAR_RESULT_DIFF'] > df['SP500_RESULT_DIFF']) & (df['DOLAR_RESULT_DIFF'] > df['BOVESPA_RESULT_DIFF'])
  & (df['DOLAR_RESULT_DIFF'] > df['OURO_RESULT_DIFF']) & (df['DOLAR_RESULT_DIFF'] > df['SELIC_RESULT_DIFF'])
  & (df['DOLAR_RESULT_DIFF'] > df['VALE_RESULT_DIFF']) & (df['DOLAR_RESULT_DIFF'] > 5)
, (df['OURO_RESULT_DIFF'] > df['SP500_RESULT_DIFF']) & (df['OURO_RESULT_DIFF'] > df['BOVESPA_RESULT_DIFF'])
  & (df['OURO_RESULT_DIFF'] > df['DOLAR_RESULT_DIFF']) & (df['OURO_RESULT_DIFF'] > df['SELIC_RESULT_DIFF'])
  & (df['OURO_RESULT_DIFF'] > df['VALE_RESULT_DIFF']) & (df['OURO_RESULT_DIFF'] > 5)
, (df['SELIC_RESULT_DIFF'] > df['SP500_RESULT_DIFF']) & (df['SELIC_RESULT_DIFF'] > df['BOVESPA_RESULT_DIFF'])
  & (df['SELIC_RESULT_DIFF'] > df['OURO_RESULT_DIFF']) & (df['SELIC_RESULT_DIFF'] > df['DOLAR_RESULT_DIFF'])
  & (df['SELIC_RESULT_DIFF'] > df['VALE_RESULT_DIFF']) & (df['SELIC_RESULT_DIFF'] > 5)
, (df['SP500_RESULT_DIFF'] > df['BOVESPA_RESULT_DIFF']) & (df['SP500_RESULT_DIFF'] > df['OURO_RESULT_DIFF'])
  & (df['SP500_RESULT_DIFF'] > df['DOLAR_RESULT_DIFF']) & (df['SP500_RESULT_DIFF'] > df['SELIC_RESULT_DIFF'])
  & (df['SP500_RESULT_DIFF'] > df['VALE_RESULT_DIFF']) & (df['SP500_RESULT_DIFF'] > 5)
, (df['VALE_RESULT_DIFF'] > df['BOVESPA_RESULT_DIFF']) & (df['VALE_RESULT_DIFF'] > df['OURO_RESULT_DIFF'])
  & (df['VALE_RESULT_DIFF'] > df['DOLAR_RESULT_DIFF']) & (df['VALE_RESULT_DIFF'] > df['SELIC_RESULT_DIFF'])
```

```
    & (df['VALE_RESULT_DIFF'] > df['SP500_RESULT_DIFF']) & (df['VALE_RESULT_DIFF'] > 5)
]

valores = [1, 2, 3, 4, 5, 6]
descritivos_target = ['NÃO FAZER NADA','ÍNDICE BOVESPA','DÓLAR','OURO','SELIC','SP&500', 'ACAO']

df['RESULTADO'] = 0
df['RESULTADO'] = np.select(condicoes, valores)

#Validação da distribuição do Target
df_count_acao = df['RESULTADO'].value_counts().to_frame()
df_count_acao.reset_index(inplace=True)
df_count_acao['ACAO'] = df_count_acao['index'].apply(lambda x: descritivos_target[x])
total = df_count_acao.RESULTADO.sum()
df_count_acao['SHARE'] = df_count_acao['RESULTADO'].apply(lambda x: (x/total)*100)
df_count_acao, total

#Primeiro arquivo de saída gerado para conferência
df.to_csv('saida/00_conferência.csv')

df.columns

#Apagando colunas utilizadas para calcular o resultado e que não serão utilizadas na predição
df_predicao = df.drop(['SP500', 'BOVESPA', 'OURO', 'PETROLEO', 'FERRO', 'DOLAR', 'VALE'
                       , 'SELIC', 'SP500_RESULT_30', 'BOVESPA_RESULT_30', 'OURO_RESULT_30'
                       , 'PETROLEO_RESULT_30', 'FERRO_RESULT_30', 'DOLAR_RESULT_30'
                       , 'SELIC_RESULT_30', 'VALE_RESULT_30', 'SP500_RESULT_DIFF'
                       , 'BOVESPA_RESULT_DIFF', 'OURO_RESULT_DIFF', 'DOLAR_RESULT_DIFF'
                       , 'SELIC_RESULT_DIFF', 'VALE_RESULT_DIFF'
                  ], axis='columns')

df_predicao.reset_index(inplace=True)

df_predicao.drop('DATA_PREGAO', axis='columns', inplace=True)

#Comparando as colunas dos dataframes mostrando as colunas a serem utilizadas para a predição
df.columns, df_predicao.columns

#Setando colunas usadas para a predição
descritivos_fatores = df_predicao.columns.to_list()[0:-1]
len(descritivos_fatores)

#Exportar dataframes para validação no excel
```

```
df.to_csv('saida/04_saida_total.csv')
df_predicao.to_csv('saida/04_saida_predicao.csv')

"""## Submissão aos algorítmos

### Separação de Dados e Target
"""

df_predicao.columns

df_predicao.describe()

#Separando os valores do resultado
x_df_predicao = df_predicao.iloc[:,0:-1].values
y_df_predicao = df_predicao.iloc[:,16:17].values

x_df_predicao[0]

y_df_predicao[0:10]

"""### Standarização dos dados"""

#Standarização dos valores
scaler_df = StandardScaler()

x_df_predicao [1:3]

x_df_predicao = scaler_df.fit_transform(x_df_predicao)

x_df_predicao [1:3]

"""### Gerando os dados de Treinamento e Testes"""

#Gerando os dados de testes e treinamento em 30%/70%
x_df_treinamento, x_df_teste, y_df_treinamento, y_df_teste = train_test_split(x_df_predicao, y_df_predicao, test_size = 0.30, random_state = 0)

x_df_treinamento.shape, y_df_treinamento.shape

x_df_teste.shape, y_df_teste.shape

"""### Aplicação dos algorítmos
```

```
#### Árvore
"""

arvore_df_predicao = DecisionTreeClassifier(criterion='entropy', random_state=0)
arvore_df_predicao.fit(x_df_treinamento, y_df_treinamento)

tree_df_previsores = arvore_df_predicao.predict(x_df_teste)
tree_df_previsores[:10]

accuracy_score(y_df_teste, tree_df_previsores)

cm = ConfusionMatrix(arvore_df_predicao, classes=descritivos_target)
cm.fit(x_df_treinamento, y_df_treinamento)
cm.score(x_df_teste, y_df_teste)

print(classification_report(y_df_teste, tree_df_previsores))

fig, axes = plt.subplots(nrows=1, ncols=1)
a = tree.plot_tree(arvore_df_predicao, feature_names=descritivos_fatores)

"""#### Árvore de decisão Randomica"""

random_forest_df = RandomForestClassifier(n_estimators=40, criterion='entropy', random_state=0)
random_forest_df.fit(x_df_treinamento, y_df_treinamento.ravel())

random_forest_df_previsores = random_forest_df.predict(x_df_teste)
random_forest_df_previsores[:10]

accuracy_score(y_df_teste, random_forest_df_previsores)

cm = ConfusionMatrix(random_forest_df, classes=descritivos_target)
cm.fit(x_df_treinamento, y_df_treinamento)
cm.score(x_df_teste, y_df_teste)

print(classification_report(y_df_teste, random_forest_df_previsores))

"""#### KNN"""

knn_df = KNeighborsClassifier(n_neighbors=1, metric='minkowski', p=1)

knn_df.fit(x_df_treinamento, y_df_treinamento.ravel())

knn_df_previsores = knn_df.predict(x_df_teste)
```

```
accuracy_score(y_df_teste, knn_df_previsores)

cm = ConfusionMatrix(knn_df, classes=descritivos_target)
cm.fit(x_df_treinamento, y_df_treinamento)
cm.score(x_df_teste, y_df_teste)

print(classification_report(y_df_teste, random_forest_df_previsores))
```