CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE BOTUCATU

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PEDRO RODRIGUES SARNIGHAUSEN

UTILIZAÇÃO DAS FERRAMENTAS DO PYTHON PARA A COLETA E ANÁLISE DE DADOS DO MERCADO DE AÇÕES

Botucatu - SP

Dezembro – 2024

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE BOTUCATU

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PEDRO RODRIGUES SARNIGHAUSEN

UTILIZAÇÃO DAS FERRAMENTAS DO PYTHON PARA A COLETA E ANÁLISE DE DADOS DO MERCADO DE AÇÕES

Orientador: Prof. Dr. Thiago Santos Mota

Artigo entregue como Trabalho de Conclusão de Curso apresentado à FATEC - Faculdade de Tecnologia de Botucatu, para obtenção do título de Tecnólogo no Curso Superior de Análise e Desenvolvimento de Sistemas

Botucatu - SP

Dezembro – 2024

UTILIZAÇÃO DAS FERRAMENTAS DO PYTHON PARA A COLETA E ANÁLISE DE DADOS DO MERCADO DE AÇÕES

USE OF PYTHON TOOLS FOR DATA COLLECTION AND ANALYSIS IN THE STOCK MARKET

Pedro Rodrigues Sarnighausen [1] Prof. Dr. Thiago Santos Mota[2]

RESUMO

Este estudo tem como objetivo demonstrar a aplicação da linguagem Python na análise de ações do mercado financeiro, enfatizando como a programação pode facilitar a coleta, filtragem e visualização dessas ações. Utilizando bibliotecas como Pandas, Matplotlib e Fundamentus, foi possível desenvolver um processo estruturado simples e intuitivo. O programa foi implementado na IDE Jupyter, proporcionando flexibilidade na personalização de critérios financeiros, como a escolha de indicadores de análise fundamentalista para escolha de ações, incluindo, como por exemplo: preço/lucro, margem EBIT e *Dividend Yield*. Após a coleta e filtragem, os dados são apresentados em gráficos que permitem uma visualização clara das informações. Os resultados obtidos demonstram a eficácia do uso de ferramentas desenvolvidas em Python para a análise de dados financeiros, proporcionando análises acessíveis e práticas que auxiliam na tomada de decisões financeiras bem fundamentadas. Este trabalho evidencia como o uso de abordagens focadas na análise de dados pode simplificar processos e gerar *insights* valiosos, atendendo às necessidades de investidores e contribuindo para uma melhoria de sua gestão financeira.

Palavras-chave: Investimentos, Mercado Financeiro, Processo Estruturado, Programação.

ABSTRACT

This study aims to demonstrate the application of Python programming language in stock market analysis, emphasizing how programming can facilitate the collection, filtering, and visualization of financial data. Using libraries such as Pandas, Matplotlib, and Fundamentus, a structured, simple, and intuitive process was developed. The program was implemented in the

[1] Graduando em Análise e Desenvolvimento de Sistemas na Faculdade de Tecnologia de Botucatu, Rua Atílio Losi - Jardim Paraíso, Botucatu – São Paulo, 18610-260, pesarni@gmail.com
[2] Doutor e Professor na Faculdade de Tecnologia de Botucatu, thiago.mota01@fatec.sp.gov.br.

Jupyter IDE, offering flexibility in customizing financial criteria, such as selecting indices and metrics, including price-to-earnings ratio (P/E), EBIT margin, and Dividend Yield. After data collection and filtering, the results are presented in graphs that provide a clear visualization of the information. The results demonstrate the effectiveness of Python-developed tools for financial data analysis, offering accessible and practical insights that support well-founded financial decision-making. This study highlights how data-driven approaches can simplify complex processes and generate valuable insights, meeting the needs of investors and contributing to the improvement of financial management.

Key words: Financial Market, Investments, Structured Process, Programming.

1 INTRODUÇÃO

O mercado de ações tem um papel extremamente necessário para a economia global, servindo como um mecanismo para empresas conseguirem capital e para investidores diversificarem suas fontes de renda. Sua importância também é notável em sua capacidade de redistribuir recursos, alocando o capital para setores ou empresas com potencial de crescimento e inovação. Além disso, o mercado de ações reflete, em tempo real, as expectativas econômicas, o que o torna um importante indicador da saúde financeira de uma nação (MISHKIN, 2019).

Nos últimos anos, esse mercado global cresceu significantemente, atingindo um valor de aproximadamente $109 trilhões em 2023. Dentre as principais economias, o mercado norte-americano se destaca como líder, representando 42,5% do total global, seguido pela União Europeia e pela China, juntos consolidando a influência dos mercados de ações na economia mundial (VISUAL CAPITALIST, 2023).

Neste contexto a análise de dados tornou-se uma área essencial para investidores e analistas, devido ao crescente volume de dados gerados no mercado financeiro.

A capacidade de organizar, analisar e interpretar esses dados permite a identificação de padrões que passariam despercebidos por abordagens tradicionais, contribuindo para a compreensão de tendências do mercado e permitindo a tomada de decisões mais fundamentadas. O uso da análise de dados também tem sido importante para a otimização de

operações financeiras, através da avaliação de riscos e retornos, consolidando-se como área crucial para atividades envolvendo o mercado de ações (GUIJARRO, 2020).

Nesse contexto, este trabalho tem como objetivo principal demonstrar a aplicação da linguagem Python na análise de ações do mercado financeiro. Foram desenvolvidos processos que permitem a coleta, filtragem e visualização de dados financeiros, tendo a facilidade de uso e simplicidade como prioridades. Para isso, foram utilizadas bibliotecas reconhecidas por sua eficiência na manipulação e visualização de dados. Com isso, procura-se demonstrar como programas desenvolvidos em Python podem auxiliar na tomada de decisões financeiras, oferecendo análises claras e precisas.

2 MATERIAL E MÉTODOS

2.1 Python

A linguagem de programação escolhida para o desenvolvimento foi o Python, devido à sua versatilidade e o fato de ser amplamente utilizada na análise de dados. Sua popularidade na área deve-se, em grande parte, ao fato de possuir muitas bibliotecas desenvolvidas especificamente para facilitar o tratamento, visualização e interpretação de grandes volumes de dados (DATACAMP, 2024), tornando-o uma ferramenta essencial para os analistas.

2.2 Jupyter

O desenvolvimento foi realizado na IDE Jupyter, um ambiente muito utilizado e especialmente voltado para aplicações em ciência de dados. O Jupyter permite a execução e documentação de cada etapa do código em blocos, facilitando tanto testes incrementais no código-fonte quanto a visualização dos dados e gráficos gerados. É uma IDE essencial para a análise organizada e para um controle preciso das operações de tratamento, filtragem e visualização de dados (DATACAMP, 2024).

2.3 Bibliotecas Utilizadas na Coleta e Organização dos Dados

Para fazer a coleta dos dados financeiros, foram utilizadas as seguintes bibliotecas:

- Fundamentus: Oferece acesso a dados financeiros e métricas utilizadas para analisar fundamentus das empresas (lucro, receita, custos, etc) e seus múltiplos. A Fundamentus fornece uma base de dados ampla para a análise detalhada de empresas específicas, utilizando indicadores fundamentais (ANÁLISE MACRO, 2024);
- A precificação de uma ação por múltiplos serve para comparar empresas do mesmo setor e buscar a partir dos próximos resultados da empresa estimar o seu valor justo no mercado de ações. Esta biblioteca apresenta vários múltiplos e a interpretação desses indicadores pode ser encontrada, por exemplo, no site Investopedia (2024). Abaixo, segue os múltiplos utilizados nesse trabalho:
    1. P/L (preço atual por ação sobre o lucro por ação) - Representa o número de anos necessários para que os juros ganhos se igualem ao valor investido, considerando o mesmo lucro por ação atual. P/L baixo indica que ação está com preço atrativo no mercado de ações.
    2. Margem EBIT (porcentagem do lucro antes impostos e juros, é uma aproximação do lucro operacional da empresa) - ajuda o investidor a entender a rentabilidade da ação ao longo do tempo,
    3. Dividend Yield (Dividendo pago por ação dividido pelo preço da ação) - É o rendimento gerado para o investidor da ação pelo pagamento de dividendos.
- Requests e Requests Cache: A biblioteca Requests é utilizada para realizar requisições HTTP aos sites de coleta de dados, enquanto a Requests Cache armazena essas respostas temporariamente, o que acelera o tempo de carregamento e reduz a necessidade de consultas repetidas à mesma fonte (SCHUCH, 2020).

Esses dados foram organizados e preparados para análise, com filtros específicos aplicados para selecionar empresas com critérios de desempenho compatíveis com o interesse do usuário.

2.4 Bibliotecas utilizadas no Processamento e Visualização dos Dados

No processamento e análise dos dados coletados, foram utilizadas bibliotecas específicas que possibilitam organizar, filtrar e interpretar esses dados:

- Pandas: Principal biblioteca para manipulação e organização de dados. Ele fornece estruturas de dados otimizadas, como DataFrames, que permitem trabalhar com dados de maneira estruturada e eficiente. Essa biblioteca é amplamente utilizada para tarefas de tratamento, limpeza e análise de grandes volumes de dados, sendo essencial em análises de ciência de dados e finanças (DATACAMP, 2024). Nesse trabalho, o Pandas foi utilizado para transformar os dados em DataFrames, facilitando a aplicação de filtros e a organização dos resultados;
- Numpy : NumPy é uma biblioteca fundamental para o processamento numérico em python. Ela permite a manipulação de arrays e matrizes de múltiplas dimensões, além de oferecer funções matemáticas altamente otimizadas para o processamento eficiente de grandes volumes de dados (DATACAMP, 2024);
- Matplotlib: Biblioteca essencial para criação de gráficos e visualizações em Python, muito utilizada na análise de dados devido à sua flexibilidade e variedade de modelos de gráficos (DATACAMP, 2024). Neste trabalho, Matplotlib teve a utilidade de gerar gráficos personalizados, permitindo uma visão clara e intuitiva dos dados analisados.

Figura 1 – Importação das bibliotecas

```
# Importando as bibliotecas
import fundamentus
import requests_cache
import requests
import time, logging
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Fonte: O Autor (2024)

2.5 Código

O código desenvolvido para este trabalho foi estruturado de forma modular, priorizando a clareza na manipulação e visualização dos dados financeiros. Esse método facilitado pelo uso da IDE Jupyter, que permite a organização do código em células independentes, cada uma contendo uma etapa específica do programa. Essa estrutura modular torna o código mais organizado e legível, além de permitir teste incrementais durante o desenvolvimento.

Além disso, o código foi projetado para garantir uma interação intuitiva com o usuário. O programa solicita parâmetros, como o índice de interesse (IBOV, IBRX e IDIV) e os critérios mínimos para as métricas financeiras, (P/L, margem EBIT e Dividend Yield) (SUNO RESEARCH, 2017), permitindo o ajuste de filtros. Isso facilita muito o processo de análise.

3 RESULTADOS E DISCUSSÃO

Nesta sessão, são apresentados os resultados obtidos com o código desenvolvido e a organização das células individuais do Jupyter, representando etapas específicas do projeto:

3.1 Coleta de Dados

A coleta de dados foi feita através da função *busca_carteira_teorica()*, conforme ilustrado na Figura 2. Essa função acessa o site da B3 (Bolsa de Valores do Brasil) e extrai a carteira teórica dos índices financeiros escolhidos pelo usuário, como IBOV, IBRX e IDIV.

Figura 2 – Função busca_carteira_teorica.

```
# Função para buscar carteira teorica
def busca_carteira_teorica(indice):
    # Faz o url para o indicie escolhido
    url = "https://bvmf.bmfbovespa.com.br/indices/ResumoCarteiraTeorica.aspx?Indice={}&idioma=pt-br"
    .format(indice.upper())
    # Le a tabela HTML para um DataFrame do Pandas
    return pd.read_html(url, decimal=',', thousands='.', index_col='Código')[0][:-1]
```

Fonte: O Autor (2024)

A operação funciona em 3 etapas:

1. Construção do URL: É gerado um URL apropriado para o índice selecionado, utilizando o parâmetro fornecido pelo usuário;
2. Leitura dos Dados: Os dados da página HTML são extraídos como tabela utilizando o método *pd.read_html()* da biblioteca Pandas, convertendo-os para um DataFrame estruturado;
3. Ajuste dos Dados: Apenas as informações relevantes são retornadas, sem a última linha de resumo.

Figura 3 – Célula de detalhamento de dados financeiros com a biblioteca Fundamentus

```
#Pega todos os dados
df = fundamentus.get_resultado()

def get_dados_indice(indice):
    dados_indice = fundamentus.get_papel(list(indice.index.values))
    return dados_indice.drop_duplicates()

# Variaveis feitas com a função busca_carteira_teorica
indices_fundamentus = {
    'IBOV': ibov,
    'IBRX': ibrx,
    'IDIV': idiv
}

# Processar cada índice e obter os dados detalhados
dados_indices = []
for nome, indice in indices_fundamentus.items():
    dados_indice = get_dados_indice(indice)
    dados_indice['Indice'] = nome  # Adiciona coluna indicando o índice
    dados_indices.append(dados_indice)

# Combina os dados detalhados de todos os índices
dados_combinados = pd.concat(dados_indices, axis=0)

# Faz o merge entre os dados financeiros gerais e os dados combinados dos índices
dados_setor = df.merge(dados_combinados, left_index=True, right_index=True)

# Seleciona e organiza as colunas relevantes
dados_setor = dados_setor[['Setor', 'Indice', 'cotacao', 'pl', 'pvp', 'psr', 'dy', 'pa', 'pcg', 'pebit', 'pacl',
                           'evebit', 'evebitda', 'mrgebit', 'mrgliq', 'roic', 'roe', 'liqc',
                           'liq2m', 'patrliq', 'divbpatr', 'c5y', 'Div_Bruta', 'Div_Liquida',
                           'Patrim_Liq', 'Receita_Liquida_12m', 'Marg_Bruta',
                           'EBIT_12m', 'Lucro_Liquido_12m', 'Receita_Liquida_3m', 'EBIT_3m', 'Lucro_Liquido_3m']]

# Resultado final
dados_setor
```

Fonte: O Autor (2024)

A integração com a biblioteca Fundamentus (Figura 3) complementa os dados obtidos pela função *busca_carteira_teorica()*, detalhando informações financeiras e setoriais das ações pertencentes aos índices IBOV, IBRX e IDIV. O processo segue quatro etapas principais:

1. Obtenção dos dados gerais: A função *fundamentus.get_resultado()* captura métricas financeiras de todas as ações da bolsa;
2. Coleta de dados detalhadas por índice: A função *get_dados_indice()* busca informações detalhadas das ações pertencentes a cada índice, eliminando duplicatas;
3. Integração dos dados: As informações gerais e detalhadas das ações são combinadas em um único *DataFrame* utilizando as funções *pd.concat()* e *merge(),* reunindo todas as métricas financeiras em uma estrutura única;
4. Organização dos dados: As colunas mais relevantes, como P/L, margem EBIT, DY, receitas e lucros, são selecionadas e organizadas em um DataFrame final, chamado *dados_setor*, que será utilizado nas análises.

3.2 Filtragem de dados

A filtragem foi implementada em duas etapas principais, conforme ilustrado nas Figuras 4 e 5. Essas etapas permitem que o usuário selecione o índice desejado e defina os critérios mínimos para as métricas financeiras.

Figura 4 – Célula que permite que o usuário escolha o índice.

```
# Solicita ao usuario o indice desejado
u_ind = input('Digite seu Indice desejado: 1-Ibov, 2-Ibrx, 3-Idiv: ')

ibov_acoes = ibov.index
ibrx_acoes = ibrx.index
idiv_acoes = idiv.index

indices = {
    '1': {'acoes': ibov_acoes, 'ind': ibov, 'nome': 'IBOV'},
    '2': {'acoes': ibrx_acoes, 'ind': ibrx, 'nome': 'IBRX'},
    '3': {'acoes': idiv_acoes, 'ind': idiv, 'nome': 'IDIV'}
}

def filtro_indice(u_ind):
    #Verifica se é uma das opções
    if u_ind in indices:
        acoes = indices[u_ind]['acoes']
        # Filtra o DataFrame para as ações do índice escolhido
        dados_ind = df[df.index.isin(acoes)]
        return dados_ind
    else:
        print("Indice inválido!")
        return none

dados_ind = filtro_indice(u_ind)
```

Fonte: O Autor (2024)

Na Figura 4, o programa solicita ao usuário que selecione um dos índices disponíveis (IBOV, IBRX ou IDIV), com base na entrada fornecida, as ações pertencentes ao índice selecionado são identificadas por meio do dicionário índices. A função *filtro_indice()* verifica se a escolha do usuário é válida e retorna um conjunto contendo apenas as ações do índice escolhido.

Figura 5 – Parte de célula que permite ao usuário definir os parâmetros de filtragem.

```
# Solicita os critérios mínimos de filtro ao usuário
u_pl = float(input("Digite o mínimo da razão de preço a lucro desejada: "))
u_ebit = float(input("Digite o mínimo da margem EBIT desejada (em %): "))
u_dy = float(input("Digite o mínimo da porcentagem de dividendos desejada (em %): "))

# Aplicar filtros no DataFrame
filtro = dados_ind[
    (dados_ind['pl'] >= u_pl) &
    (dados_ind['mrgebit'] >= u_ebit/100) &
    (dados_ind['dy'] >= u_dy/100)
]

if filtro.empty:
    print("Nenhuma ação encontrada com os critérios fornecidos.")
else:
    # Selecionar e exibir as colunas desejadas
    colunas_desejadas = ['cotacao', 'pl', 'mrgebit', 'dy']
    filtro_reduzido = filtro[colunas_desejadas]
    disp.display(filtro_reduzido)
```

Fonte: O Autor (2024)

Na Figura 5, o programa solicita que o usuário defina critérios mínimos para três métricas financeiras: razão de preço a lucro (P/L), margem EBIT e *Dividend Yield* (DY). Os valores fornecidos são utilizados para filtrar o conjunto de dados retornado pela etapa anterior, mantendo apenas as ações que atendem aos critérios definidos no DataFrame final. Por fim, os resultados filtrados são exibidos em um formato reduzido, mostrando apenas as colunas de maior interesse (cotação, P/L, margem EBIT e DY).

3.3 Visualização dos Dados

A visualização dos dados é uma etapa essencial para a compreensão e interpretação dos resultados de uma análise financeira. No programa desenvolvido neste trabalho, foram criados dois tipos de gráficos utilizando a biblioteca *Matplotlib*: um gráfico de barras, que destaca ações

com base nas métricas definidas pelo usuário, e um gráfico de rosca, que ilustra as 10 ações com maior peso percentual no índice selecionado, a seguir, são apresentados os códigos e os gráficos gerados.

3.3.1 Gráfico de Barras

Figura 6 – Célula de definição dos parâmetros e criação do Gráfico de Barras

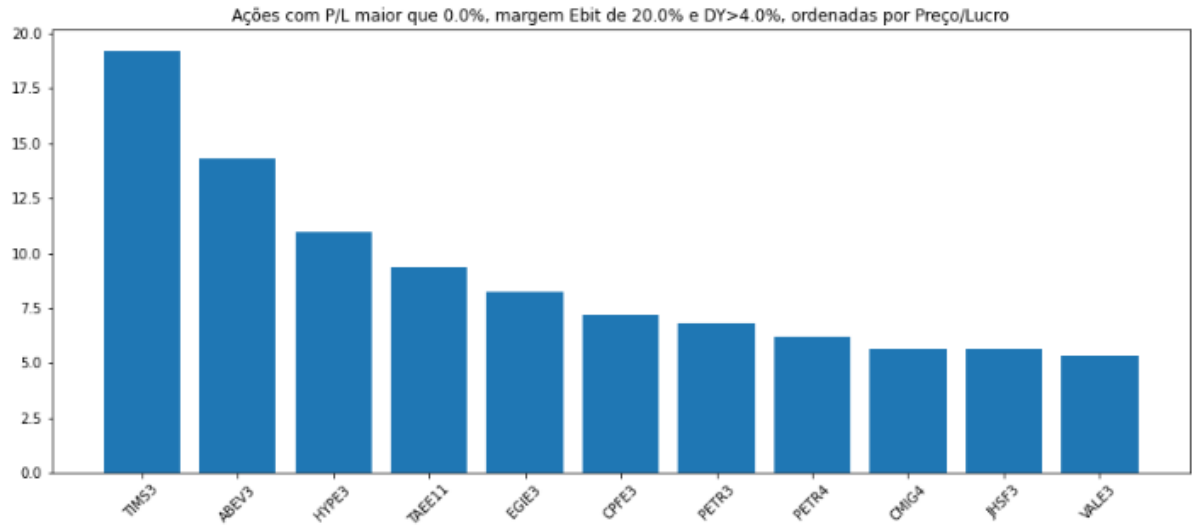
```
met = input("Escolha a métrica que você deseja usar para ordenar o gráfico: 1 - Preço/Lucro, 2 - Margem EBIT, 3 - Dividend Yield
if met not in ['1', '2', '3']:
    raise ValueError("Opção inválida! Escolha entre 1, 2 ou 3.")



metricas = {
    '1': {'col': 'pl', 'nome': 'Preço/Lucro'},
    '2': {'col': 'mrgebit', 'nome': 'Margem EBIT'},
    '3': {'col': 'dy', 'nome': 'Dividend Yield'}
}

# Cria uma cópia do DataFrame ordenado em ordem decrescente baseado na métrica escohlia pelo usuário
filtro_graf = filtro.sort_values(str(metricas[met]['col']), ascending=False, inplace=False)
# inplace=False garante que o DataFrame original não seja modificado


#Criação de um gráfico de barras
plt.figure(figsize=(15,6))

if met == '1':  # Preço/Lucro
    plt.bar(filtro_graf.index, filtro_graf.pl)
elif met == '2':  # Margem EBIT
    plt.bar(filtro_graf.index, filtro_graf.mrgebit)
elif met == '3':  # Dividend Yield
    plt.bar(filtro_graf.index, filtro_graf.dy)
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.title('Ações com P/L maior que '+str(u_pl)+'%, margem Ebit de '+str(u_ebit)+'% e DY>'+str(u_dy)+'%, ordenadas por '+metricas
plt.show()
```

Fonte: O Autor (2024)

O código apresentado na Figura 6 permite que o usuário escolha a métrica que ordena as ações exibidas no gráfico, e então exibe o gráfico, essa função segue 3 etapas principais:

1. Escolha da métrica: O programa solicita que o usuário selecione uma métrica entre as opções disponíveis (P/L, margem EBIT ou DY);
2. Ordenação dos Dados: O conjunto de dados filtrado é ordenado em ordem decrescente, utilizando o métodos *sort_values()* da biblioteca Pandas;
3. Criação do gráfico de barras: O gráfico de barras é gerado com as ações no eixo X e os valores da métrica no eixo Y, exibido conforme ilustrado na Figura 5.

Figura 7 – Gráfico de barras das ações ordenadas por Preço/Lucro (P/L).



Fonte: O Autor (2024)

3.3.2 Gráfico de rosca

Figura 8 – Célula de criação do Gráfico de Rosca

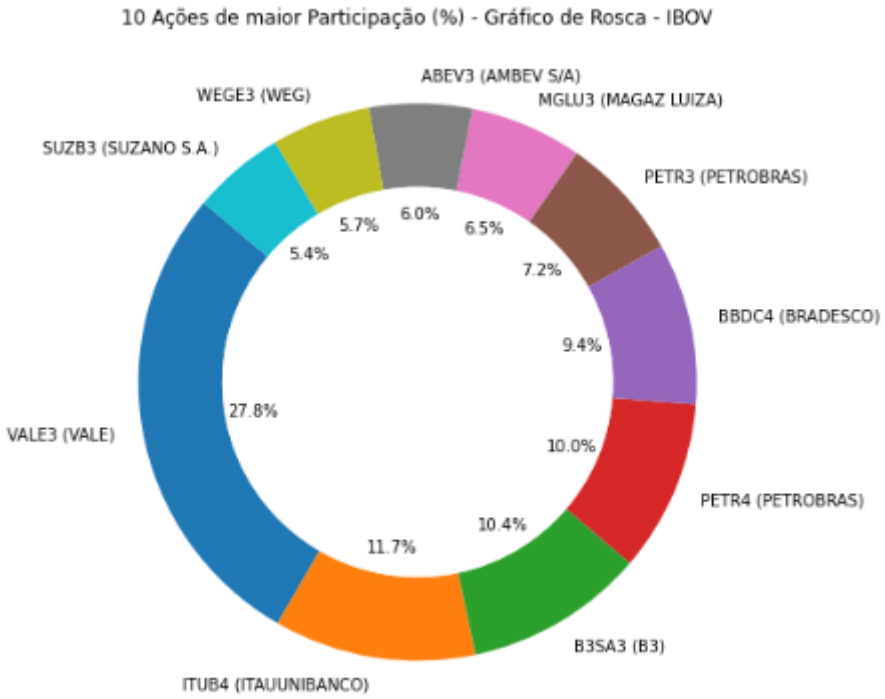
```
#Criação de um gráfico de rosca
mpeso_10_acoes = indices[u_ind]['ind'].nlargest(10, 'Part. (%)')
print(mpeso_10_acoes)
plt.figure(figsize=(8, 8))

#Definindo o formato (ação / nome)
labels = [f"{codigo} ({nome})" for codigo, nome in zip(mpeso_10_acoes.index, mpeso_10_acoes['Ação'])]

plt.pie(mpeso_10_acoes['Part. (%)'], labels=labels, autopct='%1.1f%%', startangle=140, wedgeprops=dict(width=0.3))

# Adicionando um círculo no centro para criar o efeito de rosca
centro = plt.Circle((0, 0), 0.70, fc='white')
plt.gca().add_artist(centro)

# Título do gráfico
plt.title(str('10 Ações de maior Participação (%) - Gráfico de Rosca'+' - '+indices[u_ind]['nome']))

# Exibir o gráfico
plt.show()
```

Fonte: O Autor (2024)

O código apresentado na Figura 8 gera um gráfico de rosca que representa as 10 ações com maior participação percentual no índice selecionado pelo usuário. Essas ações são identificadas pelo método *nlargest(10, 'part. (%)')*. O gráfico em si é criado utilizando a função

plt.pie(), da biblioteca Matplotlib, e um círculo branco é adicionado no centro com *plt.circle()* para criar o efeito de rosca, conforme demonstrado na Figura 8.

Figura 8 – Gráfico de Rosca das 10 ações mais importantes do índice selecionado.



Fonte: O Autor (2024)

4 CONCLUSÃO

Este estudo mostrou que a programação em Python é uma ferramenta eficiente para apoiar a análise de dados financeiros. Por meio de um programa estruturado e personalizado, foi possível implementar funcionalidades que facilitam a coleta e o tratamento de informações financeiras, enquanto os gráficos desenvolvidos permitem identificar padrões e analisar métricas de desempenho de forma prática, atendendo aos critérios estabelecidos pelo usuário.

Além de tornar a análise mais clara e acessível, o programa oferece flexibilidade ao usuário, que poderá personalizar filtros e métricas. A busca automatizada na web manterá os dados sempre atualizados, e a coleta de dados abre oportunidades para aprimoramento, como novas técnicas e funcionalidades.

Dessa forma, o estudo demonstrou a importância da análise de dados no mercado financeiro e da programação em Python para auxiliar na tomada de decisões financeiras.

REFERÊNCIAS

ANÁLISE MACRO. **Criando indicadores fundamentalistas com Python.** Disponível em: https://analisemacro.com.br/mercado-financeiro/criando-indicadores-fundamentalistas-com-python/. 2022. Acesso em: 6 nov. 2024.

DATACAMP. **As 26 principais bibliotecas Python para ciência de dados em 2024**. Disponível em: https://www.datacamp.com/pt/blog/top-python-libraries-for-data-science. 2024. Acesso em: 6 nov. 2024.

DATACAMP**. Como usar o Jupyter Notebooks: O guia definitivo**. Disponível em: https://www.datacamp.com/pt/tutorial/tutorial-jupyter-notebook. 2024. Acesso em: 6 nov. 2024.

FISCHER, Marcel; KUEHN, Lars-Alexander; WEBER, Andreas**. Liquidity-Driven Dynamic Asset Allocation. The Review of Financial Studies,** 2021. Disponível em: https://academic.oup.com/rfs/article/34/7/3213/6210658?login=false. 2021. Acesso em: 24 out. 2024.

GUIJARRO, Francisco (Ed.). **Data Analysis for Financial Markets**. *Data*, v. 5, 2020. Disponível em: https://www.mdpi.com/journal/data/special_issues/Data_Analysis. 2020. Acesso em: 24 out. 2024.

INVESTOPEDIA. *Financial Term Dictionary.* Investopedia, 2024. Disponível em: https://www.investopedia.com/financial-term-dictionary-4769738. 2024. Acesso em: 22 nov. 2024.

MISHKIN, Frederic S**. The Economics of Money, Banking, and Financial Markets.** 12. ed. Nova York: Pearson, 2019.

SCHUCH**, F. Obtenção e manipulação de dados históricos do mercado financeiro**. Disponível em: https://www.fschuch.com/blog/2020/05/21/obtencao-e-manipulacao-de-dados-historicos-do-mercado-financeiro/. 2020. Acesso em: 8 nov. 2024.

SUNO RESEARCH**. Os indicadores mais importantes em uma análise fundamentalista**. Disponível em: https://www.suno.com.br/artigos/os-indicadores-mais-importantes-em-uma-analise/. 2017. Acesso em: 10 fev. 2025.

VISUAL CAPITALIST**. The $109 Trillion Global Stock Market in One Chart. Visual Capitalist,** 2023. Disponível em: https://www.visualcapitalist.com. Acesso em: 24 out. 2024.