

```
1 import numpy as np
2 import pandas as pd
3 from datetime import datetime
4 import statsmodels.api as sm
5 from dateutil.relativedelta import relativedelta
6 import matplotlib.pyplot as plt
7 import itertools
8 from tqdm import tqdm
```

A partir do material visto nas aulas, criar 3 modelos de fatores para a seleção de portfólios de ações, sendo que cada modelo deve ter pelo menos 2 fatores combinados.

- Modelo 1: Maior retorno
- Modelo 2: Maior relação retorno/volatilidade
- Modelo 3: Maior relação Alpha/Beta

A avaliação levará em conta:

- a apresentação do relatório e dos resultados
- originalidade na combinação de fatores, além do que foi implementado em aula
- resultados obtidos

```
1 # Data de início e fim do Back test
2 start='4/2000'
3 end='9/2024'
4
5 dados = pd.read_csv('Aula-DB-Acoes.csv', index_col=0)
6 indices_acc = pd.read_csv('Aula-DB-Indices.csv')
7 indices_acc.set_index('Data', inplace=True)
8 indices = indices_acc.pct_change(fill_method=None)
```

1 dados

	Data	Empresa	Fechamento	Retorno	IBX	ROIC	ROE	PVP	Mom12	Mom6	Mom3	Volat	
0	Apr-2000	ABEV3	0.279614	0.052049	0.310	1.485570	-6.519093	4.322196	NaN	NaN	8.903227	22.232147	14
1	Apr-2000	ABYA3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	Apr-2000	ACES3	9.095609	0.045871	0.028	-3.038846	-18.978114	0.616900	NaN	NaN	28.235296	61.446073	14
3	Apr-2000	ACES4	10.389367	0.126050	0.278	-3.038846	-18.978114	0.729999	NaN	NaN	5.309734	63.707985	14
4	Apr-2000	ACGU3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	
103401	Sep-2024	WEGE3	53.620000	-0.009788	3.364	23.593823	30.114952	9.157914	53.777720	48.582301	44.968323	30.782609	192
103402	Sep-2024	WEGE4	NaN	NaN	NaN	23.593823	30.114952	NaN	NaN	NaN	NaN	NaN	192
103403	Sep-2024	WHMT3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
103404	Sep-2024	WIZC3	6.100000	-0.011345	NaN	19.710845	23.623256	1.663025	8.069882	2.849115	3.697479	38.278294	5
103405	Sep-2024	YDUQ3	10.910000	0.107614	0.120	6.347385	4.706970	0.968435	-50.779359	-50.755184	-18.662263	38.635438	31

103406 rows × 13 columns

- ✓ Buscando encontrar as melhores combinações de 2 ou mais fatores para a seleção dos modelos.

```
1 # Função auxiliar para calcular diferença de anos entre duas datas
2 def calc_dif_dates(start, end):
3     data_dif = relativedelta(datetime.strptime(end, '%m/%Y'), datetime.strptime(start, '%m/%Y'))
4     return data_dif.years + data_dif.months/12 # Função para avaliar o desempenho de uma carteira baseada em fatores
5
6 def evaluate_portfolio(dados, referencia, start, end, fatores, ascending_flags, filtro_fim=10):
7     cost_trans = 0.0006 # custo de transação
8     list_date, list_ret = [], []
9
10    # Loop por cada mês no período
11    for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
12        # Filtra os dados do mês específico com dados válidos
13        cart = dados.loc[
14            (dados['Data'] == dt) &
15            (dados['IBX'] > 0) &
16            (~dados['Retorno'].isnull())
17        ].copy()
18
19        # Aplica ranking para cada fator (ordenação crescente ou decrescente)
20        for i, fator in enumerate(fatores):
21            cart[f'rank_{i}'] = cart[fator].rank(ascending=ascending_flags[i])
22
23        # Soma dos rankings e ranking final
24        cart['rank_sum'] = cart[[f'rank_{i}' for i in range(len(fatores))]].sum(axis=1)
25        cart['rank_final'] = cart['rank_sum'].rank(ascending=True)
26
27        # Retorno médio da carteira selecionada, descontando custo de transação
28        ret = cart.loc[cart['rank_final'] < filtro_fim, 'Retorno'].mean() - cost_trans
29        list_date.append(dt)
30        list_ret.append(ret)
31
32    # Criação do DataFrame com os retornos mensais
33    Port = pd.DataFrame({'Data': list_date, 'Ret': list_ret})
34    Port.set_index('Data', inplace=True)
35
36    # Cálculo do retorno acumulado e drawdown
37    Port['Ret_acc'] = (1 + Port['Ret']).cumprod() - 1
38    Port['acc_max'] = Port['Ret_acc'].cummax()
39    Port['drawdown'] = ((1 + Port['Ret_acc']) / (1 + Port['acc_max'])) - 1
40
41    # Métricas da carteira
42    ret_acc = Port['Ret_acc'].iloc[-1] * 100 # Retorno acumulado (%)
43    vol_aa = Port['Ret'].std() * np.sqrt(12) * 100 # Volatilidade anualizada
44    drawdown = Port['drawdown'].min() * 100 # Máximo drawdown
45
46    # Benchmark (referência) - retorno, volatilidade, acumulado
47    Port['Ref'] = referencia
48    Port['Ref_acc'] = (1 + Port['Ref']).cumprod() - 1
49    ret_ref_acc = Port['Ref_acc'].iloc[-1] * 100
50    vol_ref_aa = Port['Ref'].std() * np.sqrt(12) * 100
51
52    # Regressão OLS para Alpha e Beta em relação ao benchmark
53    model = sm.OLS(Port['Ret'].values, sm.add_constant(Port['Ref'].values), missing='drop').fit()
54    alpha = model.params[0] * 12 * 100 # Alpha anualizado (%)
55    beta = model.params[1] # Beta da carteira
56
57    # Retorno anualizado
58    ret_anual = (pow(ret_acc / 100 + 1, 1 / calc_dif_dates(start, end)) - 1) * 100
59
60    # Índice tipo Sharpe (sem taxa livre de risco)
61    sharpe_like = ret_anual / vol_aa if vol_aa != 0 else np.nan
62
63    # Relação alpha/beta
64    alpha_beta_ratio = alpha / beta if beta != 0 else np.nan
65
66    # Retorna um dicionário com as métricas do portfólio
67    return {
68        'fatores': fatores,
69        'ret_acc': ret_acc,
70        'ret_anual': ret_anual,
71        'vol_aa': vol_aa,
72        'drawdown': drawdown,
73        'sharpe_like': sharpe_like,
74        'alpha': alpha,
```

```

75     'beta': beta,
76     'alpha_beta_ratio': alpha_beta_ratio
77 }
78
79 # Função para rodar todas as combinações possíveis de fatores (de 2 até N fatores)
80 def run_all_combinations(dados, referencia, start, end, fator_list, filtro_fim=10):
81     results = []
82     print("🔄 Processando combinações...")
83
84     # Combinações de 2 até N fatores
85     for r in range(2, len(fator_list) + 1):
86         for combo in tqdm(itertools.combinations(fator_list, r), desc=f"{r} fatores"):
87             # Define a ordem de ranking para cada fator
88             # Crescente para fatores como Volatilidade ou P/VPA, decrescente para outros
89             ascending_flags = [True if f in ['Volat', 'PVP'] else False for f in combo]
90
91             # Avalia a performance da carteira baseada na combinação
92             result = evaluate_portfolio(dados, referencia, start, end, list(combo), ascending_flags, filtro_fim)
93             results.append(result)
94
95     # Converte os resultados para DataFrame
96     df_results = pd.DataFrame(results)
97
98     # Imprime os melhores modelos segundo 3 critérios diferentes
99     print("\n🏆 MELHORES PORTFÓLIOS:")
100    print("Modelo 1 - Maior Retorno Acumulado:")
101    print(df_results.loc[df_results['ret_acc'].idxmax()][['fatores', 'ret_acc']])
102
103    print("\nModelo 2 - Maior Retorno / Volatilidade (Sharpe-like):")
104    print(df_results.loc[df_results['sharpe_like'].idxmax()][['fatores', 'sharpe_like']])
105
106    print("\nModelo 3 - Maior Alpha / Beta:")
107    print(df_results.loc[df_results['alpha_beta_ratio'].idxmax()][['fatores', 'alpha_beta_ratio']])
108
109    return df_results
110
111 # =====
112 # 🚀 EXECUÇÃO DO CÓDIGO
113 # =====
114
115 # Lista de fatores disponíveis
116 fatores = ['ROIC', 'ROE', 'Mom12', 'Mom6', 'Mom3', 'Volat', 'PVP', 'Pat_Liq']
117
118 # Chamada da função para rodar todas as combinações de fatores
119 df_resultados = run_all_combinations(
120     dados=dados,                # DataFrame com os dados dos ativos
121     referencia= indices['IBOV'], # Série com retornos mensais do benchmark (IBX)
122     # Data de início e fim do Back test
123     start='4/2000',
124     end='9/2024',                # Fim do período
125     fator_list=fatores,          # Lista dos fatores a testar
126     filtro_fim=10                # Número de ativos a selecionar por mês (top-N)
127 )

```



```

fatores: 4it [00:10, 4.49s/it]<ipython-input-14-740151574e73>:11: FutureWarning: 'M' is deprecated and will be removed in a
for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
7 fatores: 5it [00:23, 4.80s/it]<ipython-input-14-740151574e73>:11: FutureWarning: 'M' is deprecated and will be removed in a
for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
7 fatores: 6it [00:27, 4.62s/it]<ipython-input-14-740151574e73>:11: FutureWarning: 'M' is deprecated and will be removed in a
for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
7 fatores: 7it [00:32, 4.51s/it]<ipython-input-14-740151574e73>:11: FutureWarning: 'M' is deprecated and will be removed in a
for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
7 fatores: 8it [00:37, 4.68s/it]
8 fatores: 0it [00:00, ?it/s]<ipython-input-14-740151574e73>:11: FutureWarning: 'M' is deprecated and will be removed in a futu
for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
8 fatores: 1it [00:04, 4.42s/it]
👉 MELHORES PORTFÓLIOS:
Modelo 1 - Maior Retorno Acumulado:
fatores      [Mom12, Mom6, Mom3, PVP]
ret_acc      42877.626347
Name: 140, dtype: object

Modelo 2 - Maior Retorno / Volatilidade (Sharpe-like):
fatores      [Mom6, Volat, PVP]
sharpe_like   1.141756
Name: 77, dtype: object

Modelo 3 - Maior Alpha / Beta:
fatores      [Mom12, Mom6, Volat, PVP]
alpha_beta_ratio  27.409856
Name: 142, dtype: object

```

```

1 # Implementando os modelos
2
3 def create_port_3_fatores_par(fator1, ascending1, fator2, ascending2, fator3, ascending3, filtro_fim, dados, referencia):
4
5     cost_trans = 0.0006
6
7     list_date = []
8     list_ret = []
9
10    for dt in pd.date_range(start=start, end=end, freq='ME').strftime('%b-%Y'):
11        cart = dados.loc[(dados['Data']==dt) & (dados['IBX'] > 0) & (dados['Retorno'].isnull() == False)].copy()
12
13        cart['rank1'] = cart[fator1].rank(ascending=ascending1)
14        cart['rank2'] = cart[fator2].rank(ascending=ascending2)
15        cart['rank3'] = cart[fator3].rank(ascending=ascending3)
16        cart['rank_sum'] = cart['rank1']+cart['rank2']
17
18        cart['rank4'] = cart['rank_sum'].rank(ascending=True)
19
20        ret = cart.loc[(cart['rank4'] < filtro_fim)]['Retorno'].mean() - cost_trans
21
22        list_date.append(dt)
23        list_ret.append(ret)
24
25    dic = {'Data': list_date,
26          'Ret': list_ret}
27
28    # Cria um dataframe com data e retornos mensais
29    Port1 = pd.DataFrame(dic)
30    Port1.set_index('Data', inplace=True)
31    Port1['Ret_acc'] = (1+Port1['Ret']).cumprod()-1 #Calcula os retornos acumulados
32
33    #Calcula o drawdown
34    Port1['acc_max'] = Port1['Ret_acc'].cummax()
35    Port1['drawdown'] = ((1+Port1['Ret_acc'])/(1+Port1['acc_max']))-1
36    drawdown = Port1['drawdown'].min()*100
37
38    #Calcula retorno acumulado e volatilidade anualizada
39    ret_acc = Port1['Ret_acc'].iloc[-1]*100
40    vol_aa = Port1['Ret'].std()*(12**(1/2))*100
41
42    #Seleciona referência e calcula retorno acumulado
43    Port1['Ref'] = referencia
44    Port1['Ref_acc'] = (1+Port1['Ref']).cumprod()-1
45
46    ret_ref_acc = Port1['Ref_acc'].iloc[-1]*100
47    vol_ref_aa = Port1['Ref'].std()*(12**(1/2))*100
48
49    print('Portfolio [' , fator1, fator2, filtro_fim, ' ] \nRet Acc:', round(ret_acc, 2), '% Ret anual.:', round((pow(ret_acc/100+1
50    print('\nReferência \nRet Acc:', round(ret_ref_acc, 2), '% Ret anual.:', round((pow(ret_ref_acc/100+1, 1/calc_dif_dates(start
51
52    #Calcula o Alpha e o Beta do portfólio
53    model = sm.OLS(Port1['Ret'].to_list(), sm.add_constant(Port1['Ref'].to_list()), missing='drop').fit()
54    print("\nPort Alpha:", round(model.params[0]*(12)*100, 2), "% Beta:", round(model.params[1], 2), " / P-values:", round(model.pvalu

```

```


55
56 return Port1.copy()
57
58 def create_port_4_fatores_par(fator1, ascending1,
59                             fator2, ascending2,
60                             fator3, ascending3,
61                             fator4, ascending4,
62                             filtro_fim, dados, referencia):
63
64     cost_trans = 0.0006
65     list_date = []
66     list_ret = []
67
68     for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
69         cart = dados.loc[
70             (dados['Data'] == dt) &
71             (dados['IBX'] > 0) &
72             (dados['Retorno'].notnull())
73         ].copy()
74
75         # Rankeia os fatores
76         cart['rank1'] = cart[fator1].rank(ascending=ascending1)
77         cart['rank2'] = cart[fator2].rank(ascending=ascending2)
78         cart['rank3'] = cart[fator3].rank(ascending=ascending3)
79         cart['rank4'] = cart[fator4].rank(ascending=ascending4)
80
81         # Soma dos ranks
82         cart['rank_sum'] = cart['rank1'] + cart['rank2'] + cart['rank3'] + cart['rank4']
83         cart['rank_final'] = cart['rank_sum'].rank(ascending=True)
84
85         # Retorno médio dos melhores ativos
86         ret = cart.loc[cart['rank_final'] < filtro_fim, 'Retorno'].mean() - cost_trans
87
88         list_date.append(dt)
89         list_ret.append(ret)
90
91     # Monta DataFrame com os resultados mensais
92     Port1 = pd.DataFrame({'Data': list_date, 'Ret': list_ret})
93     Port1.set_index('Data', inplace=True)
94
95     Port1['Ret_acc'] = (1 + Port1['Ret']).cumprod() - 1
96
97     # Drawdown
98     Port1['acc_max'] = Port1['Ret_acc'].cummax()
99     Port1['drawdown'] = ((1 + Port1['Ret_acc']) / (1 + Port1['acc_max'])) - 1
100     drawdown = Port1['drawdown'].min() * 100
101
102     # Métricas do portfólio
103     ret_acc = Port1['Ret_acc'].iloc[-1] * 100
104     vol_aa = Port1['Ret'].std() * np.sqrt(12) * 100
105
106     # Referência
107     Port1['Ref'] = referencia
108     Port1['Ref_acc'] = (1 + Port1['Ref']).cumprod() - 1
109     ret_ref_acc = Port1['Ref_acc'].iloc[-1] * 100
110     vol_ref_aa = Port1['Ref'].std() * np.sqrt(12) * 100
111
112     # Impressão dos resultados
113     print('Portfolio [' , fator1, fator2, fator3, fator4, '| filtro:', filtro_fim, ']')
114     print('Ret Acumulado:', round(ret_acc, 2), '%',
115           'Ret. Anualizado:', round((pow(ret_acc / 100 + 1, 1 / calc_dif_dates(start, end)) - 1) * 100, 2), '%',
116           'Vol. Anualizada:', round(vol_aa, 2), '%',
117           'Drawdown:', round(drawdown, 2), '%')
118
119     print('\nReferência:')
120     print('Ret Acumulado:', round(ret_ref_acc, 2), '%',
121           'Ret. Anualizado:', round((pow(ret_ref_acc / 100 + 1, 1 / calc_dif_dates(start, end)) - 1) * 100, 2), '%',
122           'Vol. Anualizada:', round(vol_ref_aa, 2), '%')
123
124     # Alpha e Beta
125     model = sm.OLS(Port1['Ret'].values, sm.add_constant(Port1['Ref'].values), missing='drop').fit()
126     alpha = model.params[0] * 12 * 100
127     beta = model.params[1]
128     pvals = model.pvalues
129
130     print("\nPort Alpha:", round(alpha, 2), "% Beta:", round(beta, 2),
131           " / P-values:", round(pvals[0], 3), round(pvals[1], 3), "\n")
132
133     return Port1.copy()

```

```

1 #fator = 'ROIC', 'ROE', 'Mom12', 'Mom6', 'Mom3', 'Volat', 'PVP', 'Pat_Liq'
2 #filtro_ini / filtro_fim = seleção de ações de acordo com o ranking
3 #ascending = 'True' para quanto menor melhor, 'False' para quanto maior melhor
4 #referencia = escolha do índice de referência dentro do dataframe índices. Ex: índices['IBOV'], índices['SP500BR'], índices['IPCA']
5
6 Port1 = create_port_4_fatores_par( 'Mom12', False, 'Mom6', False, 'Mom3', False, 'PVP', True, 10, dados=dados, referencia=índices
7
8 Port2 = create_port_3_fatores_par( 'Mom6', False, 'Volat', True, 'PVP', True, 10, dados=dados, referencia=índices['IBOV'])
9
10 Port3 = create_port_4_fatores_par( 'Mom12', False, 'Mom6', False, 'Volat', True, 'PVP', True, 10, dados=dados, referencia=índices

```

 <ipython-input-12-f9f0d92018f9>:66: FutureWarning: 'M' is deprecated and will be removed in a future version, please use 'ME' instead
for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
Portfolio [Mom12 Mom6 Mom3 PVP | filtro: 10]
Ret Acumulado: 17822.8 % Ret. Anualizado: 23.68 % Vol. Anualizada: 26.05 % Drawdown: -43.54 %

Referência:
Ret Acumulado: 663.21 % Ret. Anualizado: 8.68 % Vol. Anualizada: 24.22 %

Port Alpha: 15.94 % Beta: 0.79 / P-values: 0.0 0.0

Portfolio [Mom6 Volat 10]
Ret Acc: 9624.71 % Ret anual.: 20.62 % Vol anual.: 18.93 % Drawdown: -27.82 %

Referência
Ret Acc: 663.21 % Ret anual.: 8.68 % Vol anual.: 24.22 %

Port Alpha: 14.37 % Beta: 0.56 / P-values: 0.0 0.0

<ipython-input-12-f9f0d92018f9>:66: FutureWarning: 'M' is deprecated and will be removed in a future version, please use 'ME' instead
for dt in pd.date_range(start=start, end=end, freq='M').strftime('%b-%Y'):
Portfolio [Mom12 Mom6 Volat PVP | filtro: 10]
Ret Acumulado: 27973.77 % Ret. Anualizado: 25.97 % Vol. Anualizada: 21.7 % Drawdown: -31.32 %

Referência:
Ret Acumulado: 663.21 % Ret. Anualizado: 8.68 % Vol. Anualizada: 24.22 %

Port Alpha: 18.49 % Beta: 0.65 / P-values: 0.0 0.0

Por alguma razão, os resultados dos três portfólios divergem na seleção e na implementação, após implementados, observamos que o portfólio 3 é o que possui maior retorno total, superando o portfólio 1, em torno de 28000%, além da maior razão alpha/beta (em torno de 28.45), o que era esperado durante o processo de seleção. Também possui maior retorno/volatilidade, superando o portfólio 2 (calculado em 1.197), considerado, inicialmente, como o que maximizava esta relação.

Entretanto, para a análise comparativa, serão considerados os desempenhos dos três portfólios.

✓ Resultados

```


1 #Cria dataframe com resultados acumulados
2 Port1_acc = pd.DataFrame(Port1['Ret_acc'])
3 Port1_acc.rename(columns={"Ret_acc": "Port1"}, inplace=True)
4
5 Port1_ret_men = pd.DataFrame(Port1['Ret'])
6 Port1_ret_men.rename(columns={"Ret": "Port1"}, inplace=True)
7
8 Port1_ret_men['Referencia']=Port1['Ref']
9 Port1_acc['Referencia'] = Port1['Ref_acc']
10
11 Port2_acc = pd.DataFrame(Port2['Ret_acc'])
12 Port2_acc.rename(columns={"Ret_acc": "Port2"}, inplace=True)
13
14 Port2_ret_men = pd.DataFrame(Port2['Ret'])
15 Port2_ret_men.rename(columns={"Ret": "Port2"}, inplace=True)
16
17 Port2_ret_men['Referencia']=Port2['Ref']
18 Port2_acc['Referencia'] = Port2['Ref_acc']
19
20 Port3_acc = pd.DataFrame(Port3['Ret_acc'])
21 Port3_acc.rename(columns={"Ret_acc": "Port3"}, inplace=True)
22
23 Port3_ret_men = pd.DataFrame(Port3['Ret'])
24 Port3_ret_men.rename(columns={"Ret": "Port3"}, inplace=True)
25
26 Port3_ret_men['Referencia']=Port3['Ref']
27 Port3_acc['Referencia'] = Port3['Ref_acc']

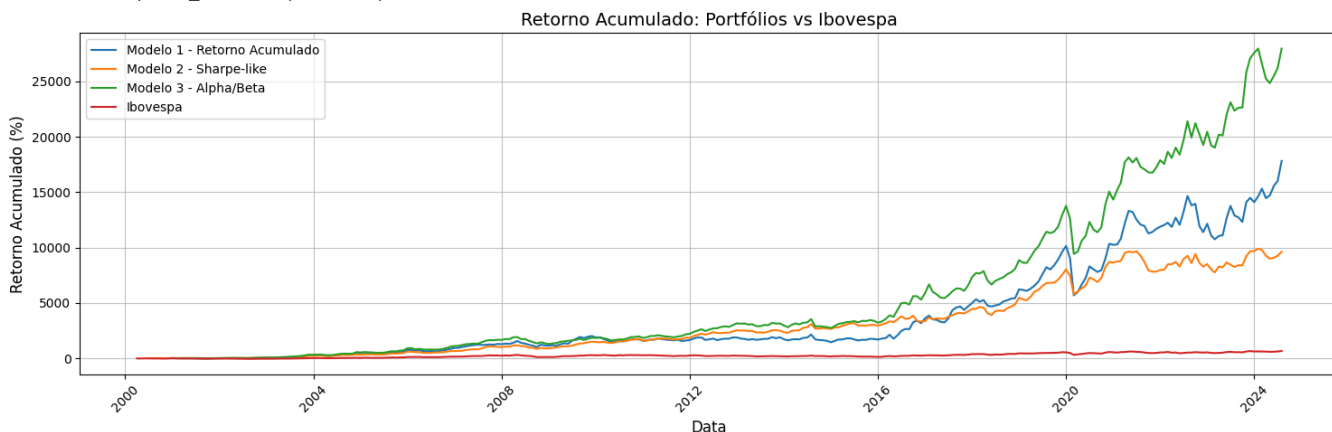
```

```

1 # Garantir que os índices estão em datetime
2 for df in [Port1_acc, Port2_acc, Port3_acc]:
3     df.index = pd.to_datetime(df.index)
4
5 # Separar as séries individuais
6 Ref = Port1_acc['Referencia'].copy()
7 Port1 = Port1_acc['Port1'].copy()
8 Port2 = Port2_acc['Port2'].copy()
9 Port3 = Port3_acc['Port3'].copy()
10
11 # Verificar e remover valores infinitos ou ausentes
12 for serie in [Port1, Port2, Port3, Ref]:
13     serie.replace([np.inf, -np.inf], np.nan, inplace=True)
14
15 Port1.dropna(inplace=True)
16 Port2.dropna(inplace=True)
17 Port3.dropna(inplace=True)
18 Ref.dropna(inplace=True)
19
20 # Criar o gráfico com o estilo OO do matplotlib
21 fig, ax = plt.subplots(figsize=(15, 5))
22
23 # Plotar as séries
24 ax.plot(Port1.index, Port1 * 100, label='Modelo 1 - Retorno Acumulado')
25 ax.plot(Port2.index, Port2 * 100, label='Modelo 2 - Sharpe-like')
26 ax.plot(Port3.index, Port3 * 100, label='Modelo 3 - Alpha/Beta')
27 ax.plot(Ref.index, Ref * 100, label='Ibovespa')
28
29 # Ajustes visuais
30 ax.set_title('Retorno Acumulado: Portfólios vs Ibovespa', fontsize=14)
31 ax.set_xlabel('Data', fontsize=12)
32 ax.set_ylabel('Retorno Acumulado (%)', fontsize=12)
33 ax.legend(fontsize=10)
34 ax.grid(True, alpha=0.7)
35 ax.tick_params(axis='x', rotation=45)
36
37 plt.tight_layout()
38 plt.show()

```

 <ipython-input-15-6ab0c17ee4ea>:3: UserWarning: Could not infer format, so each element will be parsed individually, falling back
 df.index = pd.to_datetime(df.index)
 <ipython-input-15-6ab0c17ee4ea>:3: UserWarning: Could not infer format, so each element will be parsed individually, falling back
 df.index = pd.to_datetime(df.index)
 <ipython-input-15-6ab0c17ee4ea>:3: UserWarning: Could not infer format, so each element will be parsed individually, falling back
 df.index = pd.to_datetime(df.index)



```

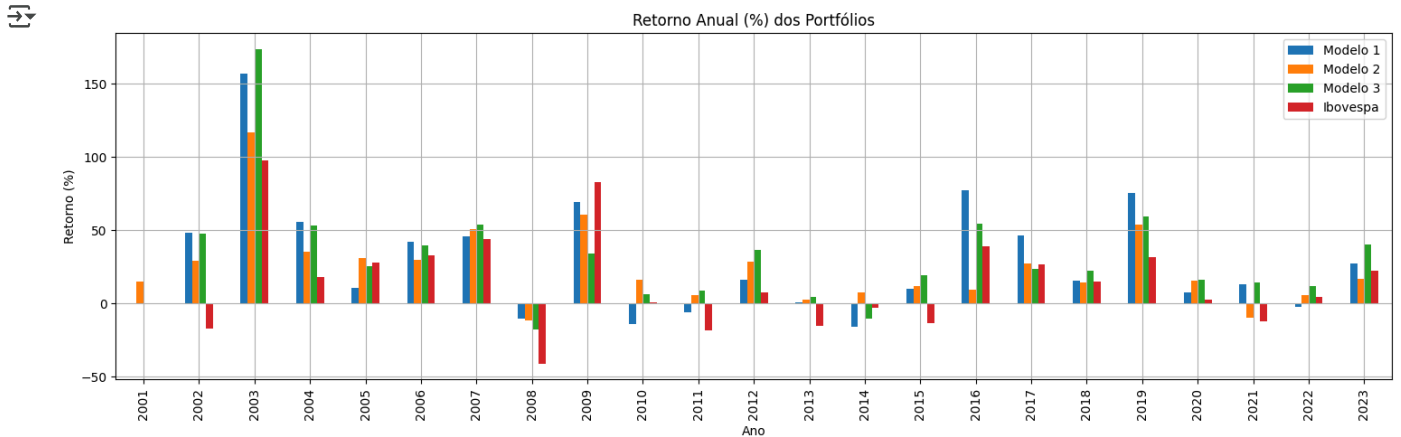
1 # Calcular retorno anual para cada portfólio
2 Port1_12_temp = ((1 + Port1_acc) / (1 + Port1_acc.shift(12)) - 1) * 100
3 Port2_12_temp = ((1 + Port2_acc) / (1 + Port2_acc.shift(12)) - 1) * 100
4 Port3_12_temp = ((1 + Port3_acc) / (1 + Port3_acc.shift(12)) - 1) * 100
5
6 # Selecionar datas de fim de ano
7 datas_anuais = pd.date_range(start=start, end=end, freq='YE').strftime('%b-%Y')
8
9 # Obter os valores de retorno anual nas datas específicas
10 Port1_12m = Port1_12_temp.loc[Port1_12_temp.index.strftime('%b-%Y').isin(datas_anuais)].dropna()
11 Port2_12m = Port2_12_temp.loc[Port2_12_temp.index.strftime('%b-%Y').isin(datas_anuais)].dropna()
12 Port3_12m = Port3_12_temp.loc[Port3_12_temp.index.strftime('%b-%Y').isin(datas_anuais)].dropna()
13
14 # Criar DataFrame combinado para plotar
15 retornos_anuais = pd.DataFrame({

```

```

16 'Modelo 1': Port1_12m['Port1'],
17 'Modelo 2': Port2_12m['Port2'],
18 'Modelo 3': Port3_12m['Port3'],
19 'Ibovespa': Port1_12m['Referencia']
20 })
21
22 # Ajustar índice para mostrar os anos no gráfico
23 retornos_anuais.index = retornos_anuais.index.strftime('%Y')
24
25 # Plotar gráfico de barras
26 retornos_anuais.plot.bar(figsize=(15, 5), grid=True)
27 plt.title('Retorno Anual (%) dos Portfólios')
28 plt.ylabel('Retorno (%)')
29 plt.xlabel('Ano')
30 plt.tight_layout()
31 plt.show()

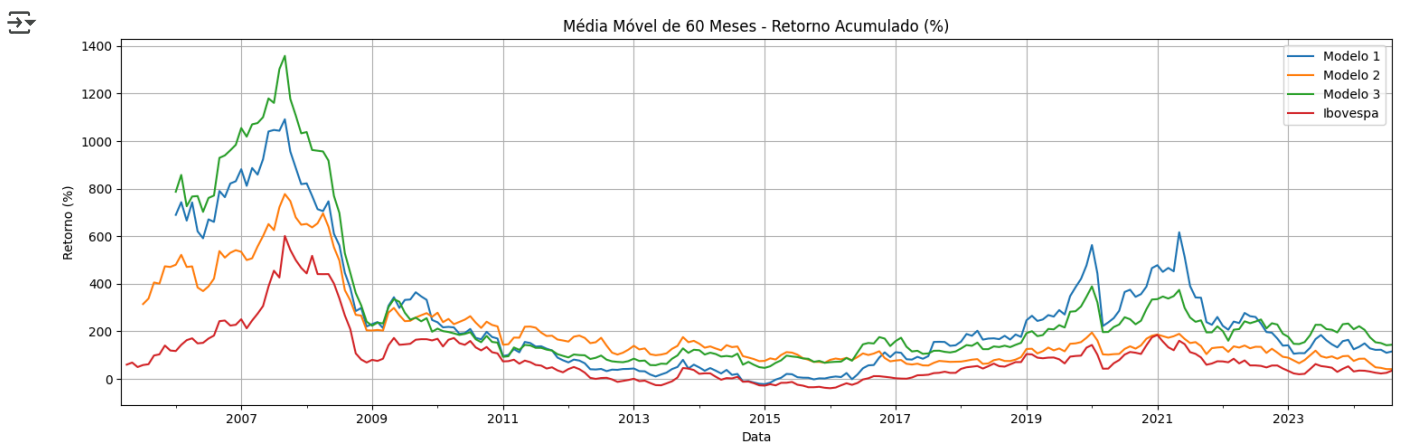
```



```

1 # Parâmetro da média móvel
2 med_movel = 60
3
4 # Calcular médias móveis para cada portfólio
5 Port1_med_movel = ((1 + Port1_acc) / (1 + Port1_acc.shift(med_movel)) - 1) * 100
6 Port2_med_movel = ((1 + Port2_acc) / (1 + Port2_acc.shift(med_movel)) - 1) * 100
7 Port3_med_movel = ((1 + Port3_acc) / (1 + Port3_acc.shift(med_movel)) - 1) * 100
8
9 # Combinar em DataFrame para plotar juntos
10 med_movel_df = pd.DataFrame({
11     'Modelo 1': Port1_med_movel['Port1'],
12     'Modelo 2': Port2_med_movel['Port2'],
13     'Modelo 3': Port3_med_movel['Port3'],
14     'Ibovespa': Port1_med_movel['Referencia']
15 })
16
17 # Plotar a partir do ponto onde todos têm dados válidos
18 med_movel_df.iloc[med_movel - 1:].plot(figsize=(15, 5), grid=True)
19 plt.title(f'Média Móvel de {med_movel} Meses - Retorno Acumulado (%)')
20 plt.ylabel('Retorno (%)')
21 plt.xlabel('Data')
22 plt.tight_layout()
23 plt.show()

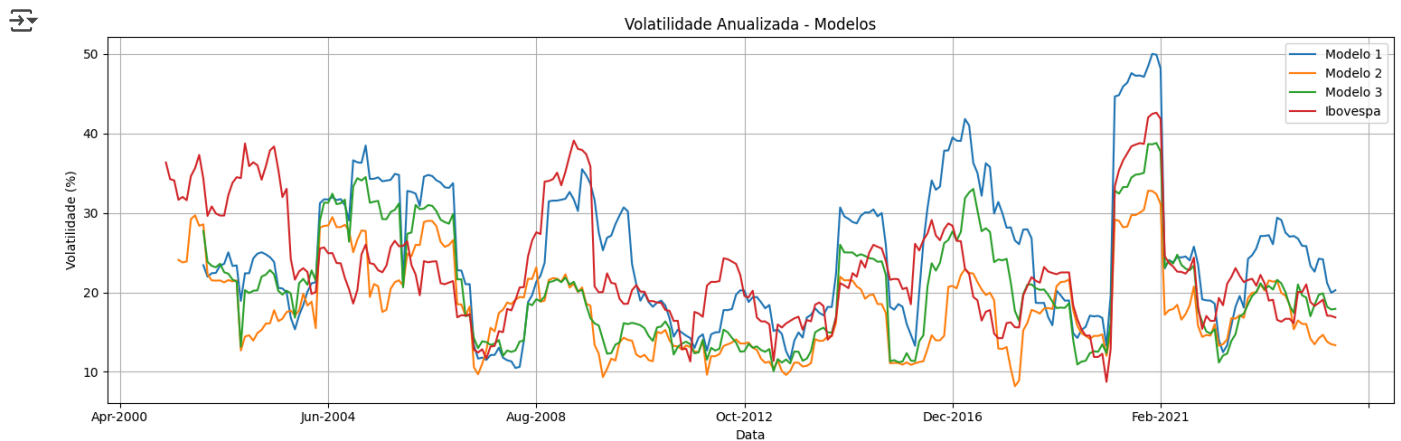
```




```

1 # Calcular volatilidade anualizada (rolling 12 meses)
2 Port1_vol_anual = (1 + Port1_ret_men).rolling(12).std() * (12 ** 0.5) * 100
3 Port2_vol_anual = (1 + Port2_ret_men).rolling(12).std() * (12 ** 0.5) * 100
4 Port3_vol_anual = (1 + Port3_ret_men).rolling(12).std() * (12 ** 0.5) * 100
5
6 # Combinar em um único DataFrame
7 vol_anual_df = pd.DataFrame({
8     'Modelo 1': Port1_vol_anual['Port1'],
9     'Modelo 2': Port2_vol_anual['Port2'],
10    'Modelo 3': Port3_vol_anual['Port3'],
11    'Ibovespa': Port1_vol_anual['Referencia']
12 })
13
14 # Plotar tudo junto
15 vol_anual_df.plot(figsize=(15, 5), grid=True)
16 plt.title('Volatilidade Anualizada - Modelos')
17 plt.ylabel('Volatilidade (%)')
18 plt.xlabel('Data')
19 plt.tight_layout()
20 plt.show()

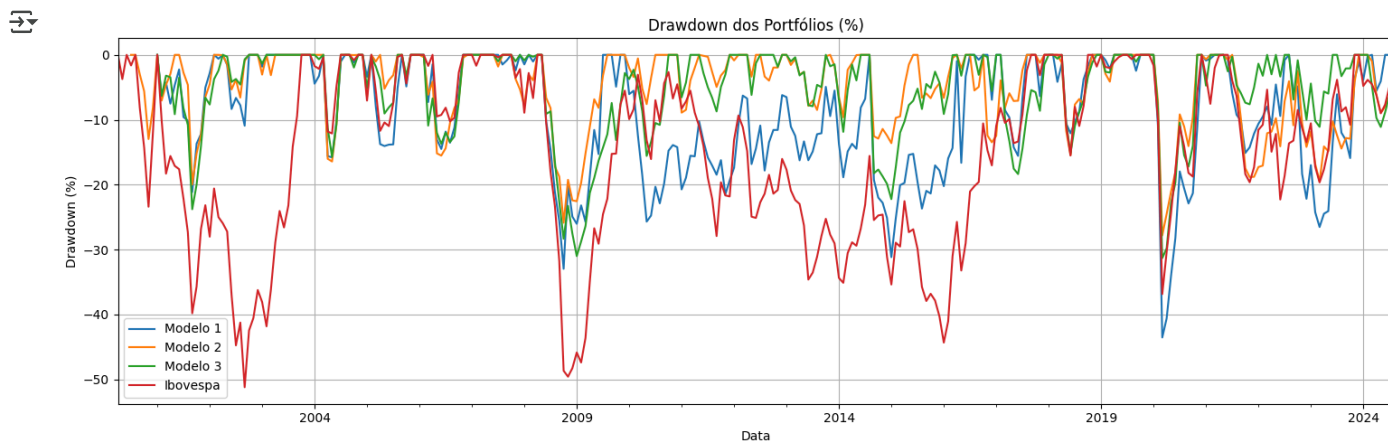
```



```

1 # Calcular o drawdown para cada portfólio
2 Port1_acc_max = Port1_acc.cummax()
3 Port1_drawdown = (((1 + Port1_acc) / (1 + Port1_acc_max)) - 1) * 100
4
5 Port2_acc_max = Port2_acc.cummax()
6 Port2_drawdown = (((1 + Port2_acc) / (1 + Port2_acc_max)) - 1) * 100
7
8 Port3_acc_max = Port3_acc.cummax()
9 Port3_drawdown = (((1 + Port3_acc) / (1 + Port3_acc_max)) - 1) * 100
10
11 # Criar DataFrame com os drawdowns
12 df_drawdown = pd.DataFrame({
13     'Modelo 1': Port1_drawdown['Port1'],
14     'Modelo 2': Port2_drawdown['Port2'],
15     'Modelo 3': Port3_drawdown['Port3'],
16     'Ibovespa': Port1_drawdown['Referencia']
17 })
18
19 # Plotar os três drawdowns no mesmo gráfico
20 df_drawdown.plot(figsize=(15, 5), grid=True)
21 plt.title('Drawdown dos Portfólios (%)')
22 plt.ylabel('Drawdown (%)')
23 plt.xlabel('Data')
24 plt.tight_layout()
25 plt.show()

```



Como pode ser observado, o modelo 3 se destaca perante os demais durante a maior parte do tempo, apresentando o maior retorno anual percentual durante boa parte da série, volatilidade e drawdown bastante moderados e boa performance nos demais comparativos. Enquanto o portfólio 2 se destaca por ser mais estável, devido a sua volatilidade e drawdown serem bastante baixos, em contraste ao seu menor retorno percentual acumulado, apresentando performance inferior aos demais a partir de 2016, mas ainda superando o Ibovespa.