

UNIVERSIDADE DE SÃO PAULO
Instituto de Instituto de Ciências Matemáticas e de Computação

Introdução à Computação no Mercado Financeiro

Prof. Denis Fernando Wolf

Trabalho 2:
Modelos de Fatores para a Seleção de Portfólios de Ações

Aluna: Bruna Gongora Bariccatti

Nº USP: 12674933

Data máxima de entrega: 11/06/2023

Objetivos

Com o presente trabalho tenho como objetivo criar 2 modelos de fatores para a seleção de portfólios de ações. O primeiro foquei em uma maior rentabilidade, visando a maior porcentagem de rentabilidade acumulada que consegui encontrar. Já a segunda tentei encontrar uma menor volatilidade, mantendo para isso a porcentagem da volatilidade o mais baixo possível.

Métodos Utilizados

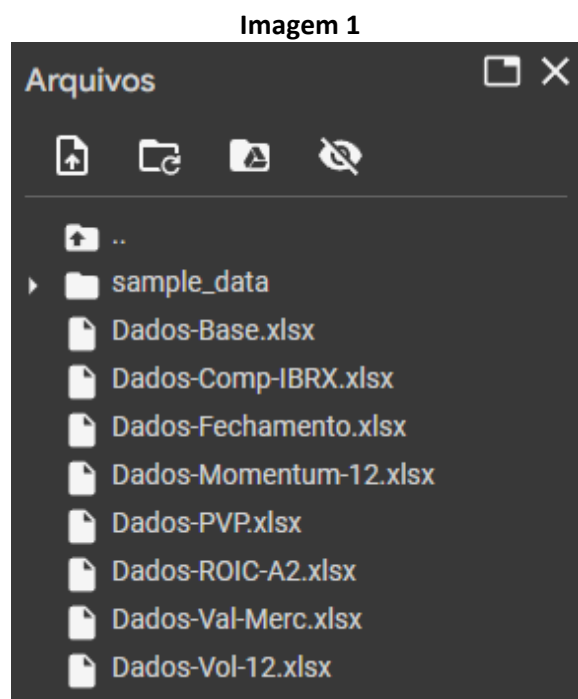
Para o primeiro portfólio utilizei uma composição com 3 fatores: o fator qualidade, o fator valor e o fator volatilidade. Nesse, usei a quantidade de parâmetros 45, 40, 60, respectivamente, para cada fator utilizado.

Já para o segundo utilizei somente dois fatores: fator qualidade e fator volatilidade. As quantidades dos parâmetros utilizados foram 40 e 25, respectivamente.

Em ambos os portfólios realizei 3 tipos de otimizações e, analisando os resultados da otimização, optei por utilizar ou não o portfólio otimizado para análises gráficas temporais.

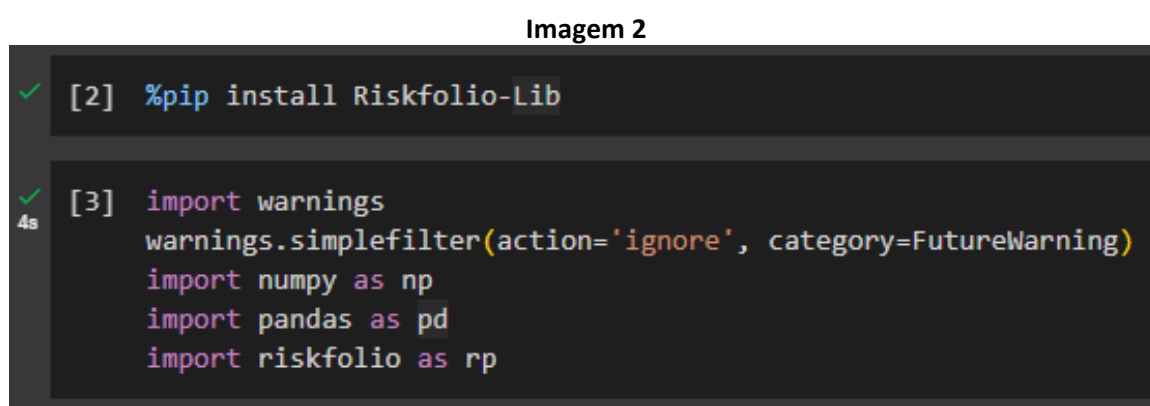
Procedimento e Resultados

Primeiramente importei os arquivos disponibilizados pelo professor, oriundos do Economatica, contendo os dados necessários para a realização do trabalho.



Fonte: Compilação pela autora

Posto isso, instalei o Riskfolio e realizei o carregamento das bibliotecas utilizadas nesse trabalho.



Fonte: Compilação pela autora

Realizo a leitura dos artigos e armazeno as informações contidas nas variáveis.

Imagem 3

```
data_inicial = 13
data_final = data_inicial + 192
colunas = 291
step_port = 1
step_eval = 1

#IBX: índice Brasil, índice da Bolsa de Valores de São Paulo que avalia o retorno
#de uma carteira teoricamente composta pelas cem ações mais negociadas na BM&FBovespa
comp_indice=pd.read_excel('Dados-Comp-IBRX.xlsx', engine='openpyxl')
comp_indice.set_index(keys = 'Data', inplace = True)

#Últimos preços negociados nas sessões
fechamento=pd.read_excel('Dados-Fechamento.xlsx', engine='openpyxl')
fechamento.set_index(keys = 'Data', inplace = True)

#Índices de referência para comparações (Ibov, IBX, SELIC...)
referencias=pd.read_excel('Dados-Base.xlsx', engine='openpyxl')
referencias.set_index(keys = 'Data', inplace = True)

#---
#Definição dos fatores e criação de um dataframe de ranqueamento dos fatores de acordo com o roic em cada mês
#---

# Fator Qualidade (ROIC das empresas)
fator_ROIC=pd.read_excel('Dados-ROIC-A2.xlsx', engine='openpyxl')
fator_ROIC.set_index(keys = 'Data', inplace = True)
ranked_ROIC=fator_ROIC.rank(axis=1, numeric_only=True, ascending=False, method='first')

# Fator Momentum (Momentum de 12 meses)
fator_Mom=pd.read_excel('Dados-Momentum-12.xlsx', engine='openpyxl')
fator_Mom.set_index(keys = 'Data', inplace = True)
ranked_Mom=fator_Mom.rank(axis=1, numeric_only=True, ascending=False, method='first')

#Fator Tamanho (Valor de mercado das empresas)
fator_Val_Merc=pd.read_excel('Dados-Val-Merc.xlsx', engine='openpyxl')
fator_Val_Merc.set_index(keys = 'Data', inplace = True)
ranked_Val_Merc=fator_Val_Merc.rank(axis=1, numeric_only=True, ascending=True, method='first')

#Fator Valor (Preço / Valor Patrimonial)
fator_PVP=pd.read_excel('Dados-PVP.xlsx', engine='openpyxl')
fator_PVP.set_index(keys = 'Data', inplace = True)
ranked_PVP=fator_PVP.rank(axis=1, numeric_only=True, ascending=True, method='first')

#Fator Volatilidade (Volatilidade em 12 meses)
fator_Vol=pd.read_excel('Dados-Vol-12.xlsx', engine='openpyxl')
fator_Vol.set_index(keys = 'Data', inplace = True)
ranked_Vol=fator_Vol.rank(axis=1, numeric_only=True, ascending=True, method='first')

print("Periodo de avaliacao - de:", comp_indice.index[data_inicial], "(", data_inicial, ")",
      "ate:", comp_indice.index[data_final-1], "(", data_final-1, ")")
print("Rebalanceamento a cada", step_eval,"/", step_port, "meses")

Periodo de avaliacao - de: Jan-2005 ( 13 ) ate: Dez-2020 ( 204 )
Rebalanceamento a cada 1 / 1 meses
```

Fonte: Compilação pela autora

Feito as alocações, defino funções de apoio, criando definições para geração de portfólios de 2 e 3 fatores, chamadas Portifolio2Parametros e Portifolio3Parametros. Esses podem ser aplicados escolhendo o fator desejado e a quantidade de parâmetros.

Imagem 4

```
[13] #Definição para portfólio com 2 fatores (caso para menos volatilidade)
def Portifolio2Parametros(ranked_1, param_1, ranked_2, param_2): #Seleção dos parâmetros
    port_ranked_final = ranked_1.copy()
    port_ranked_final.loc[:, :] = 0

    for lin in range(data_inicial, data_final, step_port):
        for col in range(0, columnas):
            if ((ranked_1.iat[lin-1, col] >= 1) and (ranked_1.iat[lin-1, col] <= param_1) and
                (ranked_2.iat[lin-1, col] >= 1) and (ranked_2.iat[lin-1, col] <= param_2)):
                port_ranked_final.iat[lin-1, col] = 1

    return port_ranked_final #Retorna o portfólio de 2 fatores

#Definição para portfólio com 3 fatores (caso para maior rentabilidade)
def Portifolio3Parametros(ranked_1, param_1, ranked_2, param_2, ranked_3, param_3): #Seleção dos parâmetros
    port_ranked_final = ranked_1.copy()
    port_ranked_final.loc[:, :] = 0

    for lin in range(data_inicial, data_final, step_port):
        for col in range(0, columnas):
            if ((ranked_1.iat[lin-1, col] >= 1) and (ranked_1.iat[lin-1, col] <= param_1) and
                (ranked_2.iat[lin-1, col] >= 1) and (ranked_2.iat[lin-1, col] <= param_2) and
                (ranked_3.iat[lin-1, col] >= 1) and (ranked_3.iat[lin-1, col] <= param_3)):
                port_ranked_final.iat[lin-1, col] = 1

    return port_ranked_final #Retorna o portfólio de 3 fatores
```

Fonte: Compilação pela autora

Além do mais, crio definições para realizar cálculos de: retorno acumulado, volatilidade acumulada, drawdown máximo, retorno anualizado e volatilidade anualizada. Aplico essas definições de cálculo tanto para os portfólios criados por mim, nesse caso utilizando-se da def EvalPort, quanto para os portfólios de referência, nesse caso utilizando-se da def EvalRef.

Imagem 5

```
[13] def EvalPort(port, fechamento): #Definição para realização de cálculos
    port_acc_vet = []
    port_chg_vet = []
    port_ddown_vet = []

    port_acc = 1.0
    port_acc_vet.append(1.0)
    cost_trans = 0.0006
    #cost_trans = 0.0005 + (0.004*step_eval/12)

    for lin in range(data_inicial, data_final, step_eval):
        cont = 0.0
        rent = 0.0
        for col in range(0, colunas):
            if (port.iat[lin-1, col] > 0 and fechamento.iat[lin-1, col]>0 and fechamento.iat[lin-1+step_eval, col]>0):
                rent = rent + (fechamento.iat[lin-1+step_eval,col]/fechamento.iat[lin-1,col]-1)*(port.iat[lin-1, col])
                cont = cont + port.iat[lin-1, col]
            if (cont == 0):
                return [1,1], [1,1], [0,0], 0, 0.000001
        port_acc = port_acc * (1.0 + rent/cont - cost_trans)
        port_chg_vet.append(rent/cont - cost_trans)
        port_acc_vet.append(port_acc)
        port_ddown_vet.append(port_acc/(np.max(port_acc_vet))-1)

    ret_aa = pow(port_acc, 12/(data_final-data_inicial))-1
    vol_aa = np.std(port_chg_vet)*((12/step_eval)**(1/2))
    return port_acc_vet, port_chg_vet, port_ddown_vet, ret_aa, vol_aa #Retorno dos valores calculados

def EvalRef(ref, ind): #Definição para realização de cálculos das referências
    ref_acc_vet = []
    ref_chg_vet = []
    ref_ddown_vet = []

    ref_acc = 1.0
    ref_acc_vet.append(1.0)

    for lin in range(data_inicial, data_final, step_eval):
        rent = ref.iat[lin-1+step_eval,ind]/ref.iat[lin-1,ind]
        ref_acc = ref_acc * rent
        ref_chg_vet.append(rent-1)
        ref_acc_vet.append(ref_acc)
        ref_ddown_vet.append(ref_acc/(np.max(ref_acc_vet))-1)

    ret_aa = pow(ref_acc, 12/(data_final-data_inicial))-1
    vol_aa = np.std(ref_chg_vet)*((12/step_eval)**(1/2))
    return ref_acc_vet, ref_chg_vet, ref_ddown_vet, ret_aa, vol_aa #Retorno dos valores calculados
```

Fonte: Compilação pela autora

Para melhores resultados defino cálculos de otimização do portfólio, utilizando 3 otimizações e, portanto, criando novos três portfólios: Portfólio de Paridade de Risco, Portfólio de Mínima Variância e Portfólio de Máxima Descorrelação.

Imagem 6

```
def calc_riskfolio_opt (ranked, otim_opt): #Recebe o portfólio e seleciona a otimização desejada
    hist_size = 24
    port = ranked.copy()

    if (otim_opt == 'RP'): #Portfólio de Paridade de Risco
        print("\nCálculo do Portfólio de Paridade de Risco")
    elif (otim_opt == 'GMV'): #Portfólio de Mínima Variância
        print("\nCálculo do Portfólio de Mínima Variância")
    elif (otim_opt == 'MDP'): #Portfólio de Máxima Descorrelação
        print("\nCálculo do Portfólio de Máxima Descorrelação")
    else:
        print("\nOpcao Invalida.")

    for lin in range(data_inicial+hist_size, data_final, 1):
        print("\r",lin, "/", data_final-1, end=' ')
        port_comp = pd.DataFrame()
        for col in range(0, colunas):
            if (port.iat[lin-1, col] > 0):
                port_comp[port.columns[col]] = fechamento[port.columns[col]].iloc[lin-1-hist_size:lin-1]

        port_comp_chg = port_comp.pct_change().dropna()

        #Processo de otimização do Portfólio de Paridade de Risco
        if (otim_opt == 'RP'):
            rp_port = rp.Portfolio(returns=port_comp_chg)
            rp_port.assets_stats(d=0.94)
            w = rp_port.rp_optimization(rm='MV', b = None)

        #Processo de otimização do Portfólio de Mínima Variância
        elif (otim_opt == 'GMV'):
            gmv_port = rp.Portfolio(returns=port_comp_chg)
            gmv_port.assets_stats(d=0.94)
            w = gmv_port.optimization(model='Classic', rm='MV', obj='MinRisk')

        #Processo de otimização do Portfólio de Máxima Descorrelação
        elif (otim_opt == 'MDP'):
            mdp_port = rp.Portfolio(returns=port_comp_chg)
            mdp_port.assets_stats(d=0.94)
            mdp_port.cov = port_comp_chg.corr()
            w = mdp_port.optimization(model='Classic', rm='MV', obj='MinRisk')

        port_len = len(port_comp_chg.columns)
        for at in range(port_len):
            port.at[port.index[lin-1], port_comp.columns[at]] = w['weights'].iat[at]

    port_final = port.copy()
    return port_final
```

Fonte: Compilação pela autora

Calculo a rentabilidade, volatilidade e drawdown da SELIC, Ibovespa e IBX para serem utilizados para comparação e realização de cálculos.

Imagem 7

```
[29] # Cálculos de rentabilidade / volatilidade / drawdown da SELIC
ref_acc_vet, ref_chg_vet, ref_ddown_vet, ret_aa_ref, vol_aa_ref = EvalRef(referencias, 2)
print("Ref SELIC:\nRentabilidade Acumulada:",round(ref_acc_vet[-1]*100-100, 2) ,"% ; Rentabilidade Anualizada:",
      round(ret_aa_ref*100,2), "% ; \nVolatilidade Anualizada:", round(vol_aa_ref*100,2), "% ; Rentabilidade por Volatilidade:",
      round(ret_aa_ref/vol_aa_ref, 2), "DDown:", round(np.min(ref_ddown_vet)*100,2), "%.")

# Cálculos de rentabilidade / volatilidade / drawdown do Ibovespa
ref_acc_vet, ref_chg_vet, ref_ddown_vet, ret_aa_ref, vol_aa_ref = EvalRef(referencias, 0)
print("Ref Ibov:\nRentabilidade Acumulada:",round(ref_acc_vet[-1]*100-100, 2) ,"% ; Rentabilidade Anualizada:",
      round(ret_aa_ref*100,2), "% ; \nVolatilidade Anualizada:", round(vol_aa_ref*100,2), "% ; Rentabilidade por Volatilidade:",
      round(ret_aa_ref/vol_aa_ref, 2), "DDown:", round(np.min(ref_ddown_vet)*100,2), "%.")

# Cálculos de rentabilidade / volatilidade / drawdown do IBX
ref_acc_vet, ref_chg_vet, ref_ddown_vet, ret_aa_ref, vol_aa_ref = EvalRef(referencias, 1)
print("Ref IBX:\nRentabilidade Acumulada:",round(ref_acc_vet[-1]*100-100, 2) ,"% ; Rentabilidade Anualizada:",
      round(ret_aa_ref*100,2), "% ; \nVolatilidade Anualizada:", round(vol_aa_ref*100,2), "% ; Rentabilidade por Volatilidade:",
      round(ret_aa_ref/vol_aa_ref, 2), "Drawdown:", round(np.min(ref_ddown_vet)*100,2), "%.")

Ref SELIC:
Rentabilidade Acumulada: 397.54 % ; Rentabilidade Anualizada: 10.55 % ;
Volatilidade Anualizada: 1.05 % ; Rentabilidade por Volatilidade: 10.04 DDown: 0.0 %.
Ref Ibov:
Rentabilidade Acumulada: 354.33 % ; Rentabilidade Anualizada: 9.92 % ;
Volatilidade Anualizada: 23.45 % ; Rentabilidade por Volatilidade: 0.42 DDown: -49.59 %.
Ref IBX:
Rentabilidade Acumulada: 546.22 % ; Rentabilidade Anualizada: 12.37 % ;
Volatilidade Anualizada: 22.69 % ; Rentabilidade por Volatilidade: 0.55 Drawdown: -49.74 %.
```

Fonte: Compilação pela autora

Realizo a criação do primeiro portfólio, onde tento encontrar o maior valor de rentabilidade. Testei inúmeras combinações de fatores e, no meu caso, achei a maior rentabilidade acumula sendo 4240.95%, utilizando para isso 3 fatores.

Imagem 8

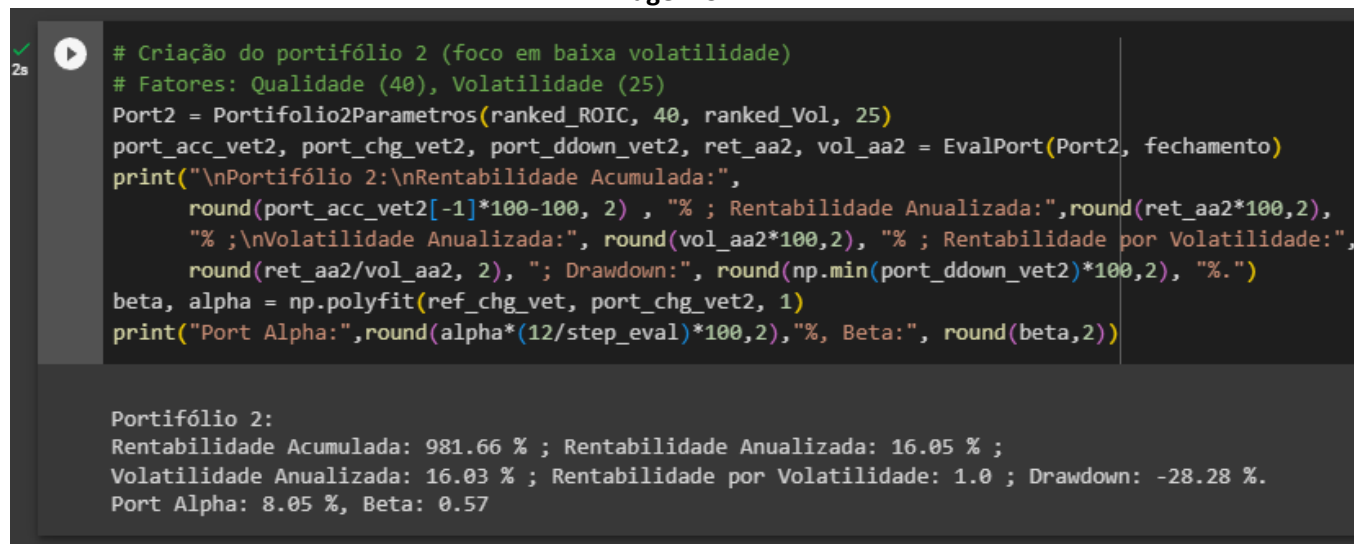
```
# Criação do portfólio 1 (foco em rentabilidade)
# Fatores: Qualidade (45), Valor (40), Volatilidade (60)
Port1 = Portifolio3Parametros(ranked_ROIC, 45, ranked_PVP, 40, ranked_Vol, 60)
port_acc_vet1, port_chg_vet1, port_ddown_vet1, ret_aa1, vol_aa1 = EvalPort(Port1, fechamento)
print("\nPortfólio 1:\nRentabilidade Acumulada:",
      round(port_acc_vet1[-1]*100-100, 2) , "% ; Rentabilidade Anualizada:",round(ret_aa1*100,2),
      "% ;\nVolatilidade Anualizada:", round(vol_aa1*100,2), "% ; Rentabilidade por Volatilidade:",
      round(ret_aa1/vol_aa1, 2), "% ; Drawdown:", round(np.min(port_ddown_vet1)*100,2), "%.")
beta, alpha = np.polyfit(ref_chg_vet, port_chg_vet1, 1)
print("Port Alpha:",round(alpha*(12/step_eval)*100,2),"% , Beta:", round(beta,2))

Portfólio 1:
Rentabilidade Acumulada: 4240.95 % ; Rentabilidade Anualizada: 26.58 % ;
Volatilidade Anualizada: 22.54 % ; Rentabilidade por Volatilidade: 1.18 ; Drawdown: -32.46 %.
Port Alpha: 15.76 % , Beta: 0.74
```

Fonte: Compilação pela autora

Crio também um segundo portfólio, visando a menor volatilidade. Encontro para esse caso uma volatilidade de 16,03%. Tendo em base os valores de volatilidade do Ibov (23.45%) e IBX (22.69%) o valor da volatilidade do portfólio 2 é um ótimo valor.

Imagem 9



```
# Criação do portfólio 2 (foco em baixa volatilidade)
# Fatores: Qualidade (40), Volatilidade (25)
Port2 = Portifolio2Parametros(ranked_ROIC, 40, ranked_Vol, 25)
port_acc_vet2, port_chg_vet2, port_ddown_vet2, ret_aa2, vol_aa2 = EvalPort(Port2, fechamento)
print("\nPortfólio 2:\nRentabilidade Acumulada:",
      round(port_acc_vet2[-1]*100-100, 2), "% ; Rentabilidade Anualizada:", round(ret_aa2*100, 2),
      "% ;\nVolatilidade Anualizada:", round(vol_aa2*100, 2), "% ; Rentabilidade por Volatilidade:",
      round(ret_aa2/vol_aa2, 2), "% ; Drawdown:", round(np.min(port_ddown_vet2)*100, 2), "%.")
beta, alpha = np.polyfit(ref_chg_vet, port_chg_vet2, 1)
print("Port Alpha:", round(alpha*(12/step_eval)*100, 2), "% , Beta:", round(beta, 2))
```

Portfólio 2:
Rentabilidade Acumulada: 981.66 % ; Rentabilidade Anualizada: 16.05 % ;
Volatilidade Anualizada: 16.03 % ; Rentabilidade por Volatilidade: 1.0 ; Drawdown: -28.28 %.
Port Alpha: 8.05 % , Beta: 0.57

Fonte: Compilação pela autora

Otimizo o portfólio 1 pelos 3 métodos discutidos anteriormente. Pode-se ver visto em seguida que todos causaram uma diminuição da rentabilidade acumulada, entretanto, o valor de rentabilidade por volatilidade no portfólio de paridade de risco se mantém quase igual, sendo uma possibilidade de otimização viável, pois, mesmo que esse cause a diminuição da rentabilidade ela causa uma diminuição também da volatilidade. Todavia, como o objetivo é ter maior rentabilidade acumulada manterei o portfólio de antes, ou seja, o sem otimização.

Imagem 10

```

40s ▶ port_riskfolio = calc_riskfolio_opt(Port1, 'RP') #Otimizando o portfólio 1 pela Paridade de Risco
port_acc_vet3, port_chg_vet3, port_ddown_vet3, ret_aa3, vol_aa3 = EvalPort(port_riskfolio, fechamento)
print("\nPort RP:\nRentabilidade Acumulada:",round(port_acc_vet3[-1]*100-100, 2), "% Rentabilidade Anualizada:",
      round(ret_aa3*100,2), "% \nVolatilidade Anualizada:", round(vol_aa3*100,2), "% Rentabilidade por Volatilidade:",
      round(ret_aa3/vol_aa3, 2), "Drawdown:", round(np.min(port_ddown_vet3)*100,2), "%")

beta, alpha = np.polyfit(ref_chg_vet, port_chg_vet3, 1)
print("Port Alpha:",round(alpha*(12/step_eval)*100,2), "%, Beta:", round(beta,2))

port_riskfolio = calc_riskfolio_opt(Port1, 'GMV') #Otimizando o portfólio 1 pela Mínima Variância
port_acc_vet4, port_chg_vet4, port_ddown_vet4, ret_aa4, vol_aa4 = EvalPort(port_riskfolio, fechamento)
print("\nPort GMV:\nRentabilidade Acumulada:",round(port_acc_vet4[-1]*100-100, 2), "% Rentabilidade Anualizada:",
      round(ret_aa4*100,2), "% \nVolatilidade Anualizada:", round(vol_aa4*100,2), "% Rentabilidade por Volatilidade:",
      round(ret_aa4/vol_aa4, 2), "Drawdown:", round(np.min(port_ddown_vet4)*100,2), "%")

beta, alpha = np.polyfit(ref_chg_vet, port_chg_vet4, 1)
print("Port Alpha:",round(alpha*(12/step_eval)*100,2), "%, Beta:", round(beta,2))

port_riskfolio = calc_riskfolio_opt(Port1, 'MDP') #Otimizando o portfólio 1 pela Máxima Descorrelação
port_acc_vet5, port_chg_vet5, port_ddown_vet5, ret_aa5, vol_aa5 = EvalPort(port_riskfolio, fechamento)
print("\nPort MDP:\nRentabilidade Acumulada:",round(port_acc_vet5[-1]*100-100, 2), "% Rentabilidade Anualizada:",
      round(ret_aa5*100,2), "% \nVolatilidade Anualizada:", round(vol_aa5*100,2), "% Rentabilidade por Volatilidade:",
      round(ret_aa5/vol_aa5, 2), "Drawdown:", round(np.min(port_ddown_vet5)*100,2), "%")

beta, alpha = np.polyfit(ref_chg_vet, port_chg_vet5, 1)
print("Port Alpha:",round(alpha*(12/step_eval)*100,2), "%, Beta:", round(beta,2))

```

Cálculo do Portfólio de Paridade de Risco
204 / 204
Port RP:
Rentabilidade Acumulada: 3534.28 % Rentabilidade Anualizada: 25.18 %
Volatilidade Anualizada: 21.56 % Rentabilidade por Volatilidade: 1.17 Drawdown: -31.21 %
Port Alpha: 14.87 %, Beta: 0.7

Cálculo do Portfólio de Mínima Variância
204 / 204
Port GMV:
Rentabilidade Acumulada: 1880.23 % Rentabilidade Anualizada: 20.52 %
Volatilidade Anualizada: 21.12 % Rentabilidade por Volatilidade: 0.97 Drawdown: -34.01 %
Port Alpha: 11.78 %, Beta: 0.64

Cálculo do Portfólio de Máxima Descorrelação
204 / 204
Port MDP:
Rentabilidade Acumulada: 3463.12 % Rentabilidade Anualizada: 25.02 %
Volatilidade Anualizada: 23.0 % Rentabilidade por Volatilidade: 1.09 Drawdown: -33.18 %
Port Alpha: 14.66 %, Beta: 0.73

Fonte: Compilação pela autora

Realizando o processo de otimização para o portfólio 2 temos que nesse a otimização reduziu a volatilidade quando aplicado o método de paridade de risco. Essa redução foi de 0,24%. Portanto já que o objetivo é manter esse valor o mais baixo vou adotar o Portifólio de paridade de risco criado a partir do portfólio 2 para próximas análises.

Imagem 11

```

36s 36s
port_riskfolio = calc_riskfolio_opt(Port2, 'RP') #Otimizando o portfólio 1 pela Paridade de Risco
port_acc_vet6, port_chg_vet6, port_ddown_vet6, ret_aa6, vol_aa6 = EvalPort(port_riskfolio, fechamento)
print("\nPort RP:\nRentabilidade Acumulada:",round(port_acc_vet6[-1]*100-100, 2) , "% Rentabilidade Anualizada:",
      round(ret_aa6*100,2), "% \nVolatilidade Anualizada:", round(vol_aa6*100,2), "% Rentabilidade por Volatilidade:",
      round(ret_aa6/vol_aa6, 2), "Drawdown:", round(np.min(port_ddown_vet6)*100,2), "%")

beta, alpha = np.polyfit(ref_chg_vet, port_chg_vet6, 1)
print("Port Alpha:",round(alpha*(12/step_eval)*100,2), "%, Beta:", round(beta,2))

port_riskfolio = calc_riskfolio_opt(Port2, 'GMV') #Otimizando o portfólio 1 pela Mínima Variância
port_acc_vet7, port_chg_vet7, port_ddown_vet7, ret_aa7, vol_aa7 = EvalPort(port_riskfolio, fechamento)
print("\nPort GMV:\nRentabilidade Acumulada:",round(port_acc_vet7[-1]*100-100, 2) , "% Rentabilidade Anualizada:",
      round(ret_aa7*100,2), "% \nVolatilidade Anualizada:", round(vol_aa7*100,2), "% Rentabilidade por Volatilidade:",
      round(ret_aa7/vol_aa7, 2), "Drawdown:", round(np.min(port_ddown_vet7)*100,2), "%")

beta, alpha = np.polyfit(ref_chg_vet, port_chg_vet7, 1)
print("Port Alpha:",round(alpha*(12/step_eval)*100,2), "%, Beta:", round(beta,2))

port_riskfolio = calc_riskfolio_opt(Port2, 'MDP') #Otimizando o portfólio 1 pela Máxima Descorrelação
port_acc_vet8, port_chg_vet8, port_ddown_vet8, ret_aa8, vol_aa8 = EvalPort(port_riskfolio, fechamento)
print("\nPort MDP:\nRentabilidade Acumulada:",round(port_acc_vet8[-1]*100-100, 2) , "% Rentabilidade Anualizada:",
      round(ret_aa8*100,2), "% \nVolatilidade Anualizada:", round(vol_aa8*100,2), "% Rentabilidade por Volatilidade:",
      round(ret_aa8/vol_aa8, 2), "Drawdown:", round(np.min(port_ddown_vet8)*100,2), "%")

beta, alpha = np.polyfit(ref_chg_vet, port_chg_vet8, 1)
print("Port Alpha:",round(alpha*(12/step_eval)*100,2), "%, Beta:", round(beta,2))

Cálculo do Portifólio de Paridade de Risco
204 / 204
Port RP:
Rentabilidade Acumulada: 1007.4 % Rentabilidade Anualizada: 16.22 %
Volatilidade Anualizada: 15.81 % Rentabilidade por Volatilidade: 1.03 Drawdown: -26.57 %
Port Alpha: 8.5 %, Beta: 0.55

Cálculo do Portifólio de Mínima Variância
204 / 204
Port GMV:
Rentabilidade Acumulada: 1033.72 % Rentabilidade Anualizada: 16.39 %
Volatilidade Anualizada: 16.16 % Rentabilidade por Volatilidade: 1.01 Drawdown: -22.17 %
Port Alpha: 9.5 %, Beta: 0.49

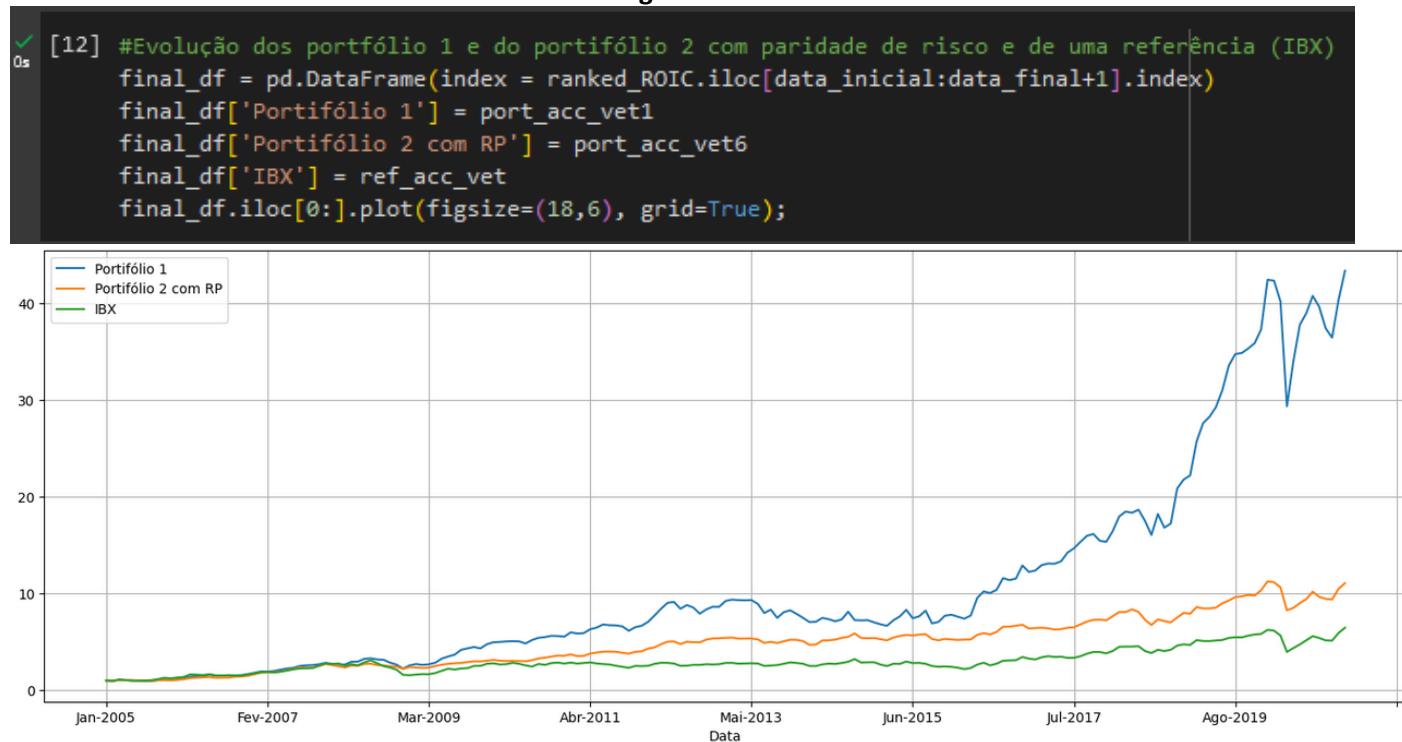
Cálculo do Portifólio de Máxima Descorrelação
204 / 204
Port MDP:
Rentabilidade Acumulada: 1091.06 % Rentabilidade Anualizada: 16.75 %
Volatilidade Anualizada: 16.62 % Rentabilidade por Volatilidade: 1.01 Drawdown: -31.54 %
Port Alpha: 9.41 %, Beta: 0.53

```

Fonte: Compilação pela autora

No gráfico a seguir temos a comparação da evolução dos dois portfólios escolhidos e uma referência, que nesse caso utilizei o IBX.

Imagem 12



Fonte: Compilação pela autora

Já nesse outro gráfico temos as mudanças de volatilidade dos portfólios e do IBX.

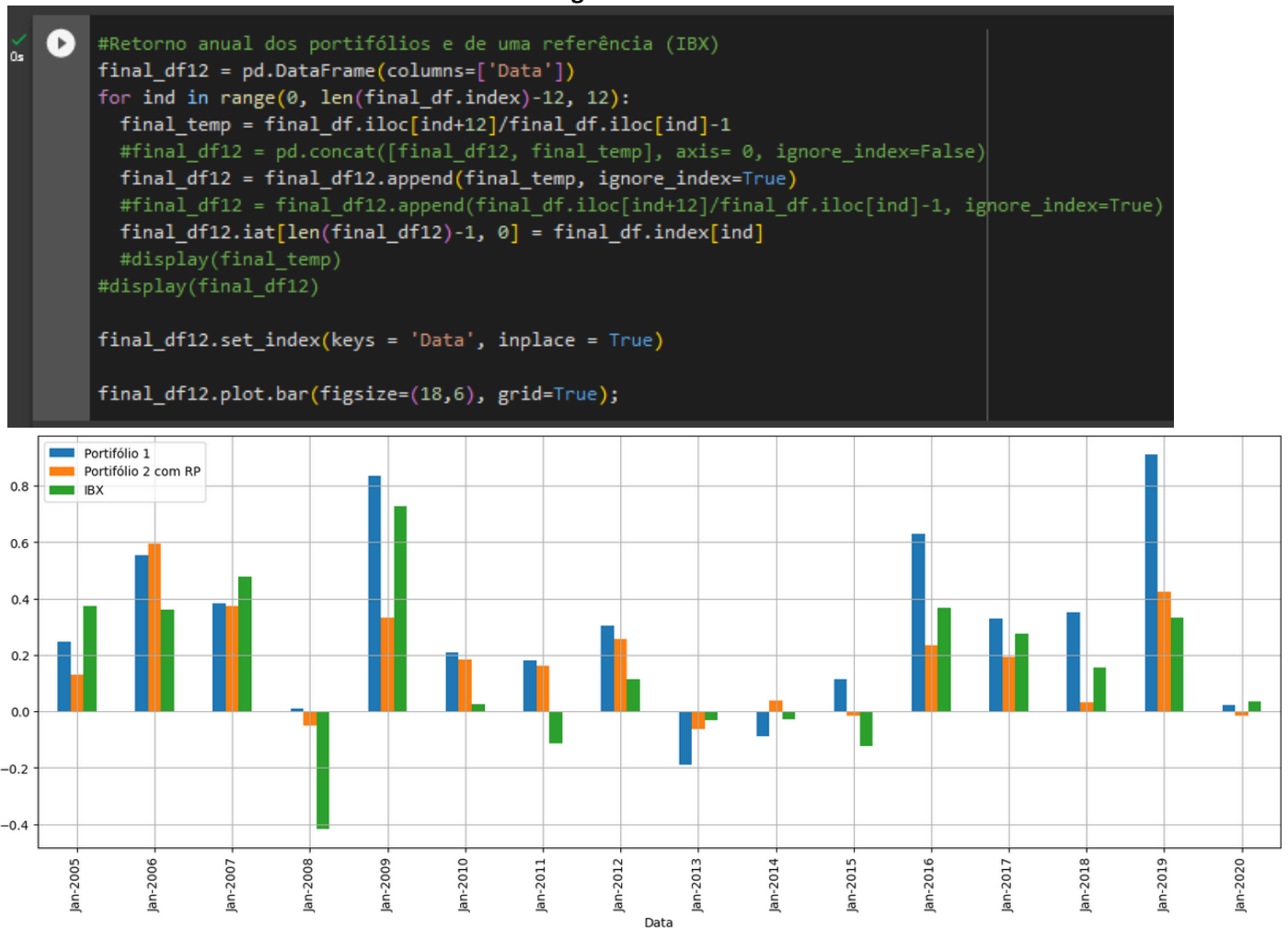
Imagem 13



Fonte: Compilação pela autora

Por fim, crio um histograma com as variações dos retornos anuais dos portfólios criados por mim, o Portfólio 1 e o Portfólio 2 com a otimização de paridade de risco, juntamente uma referência, o IBX.

Imagem 14



Fonte: Compilação pela autora