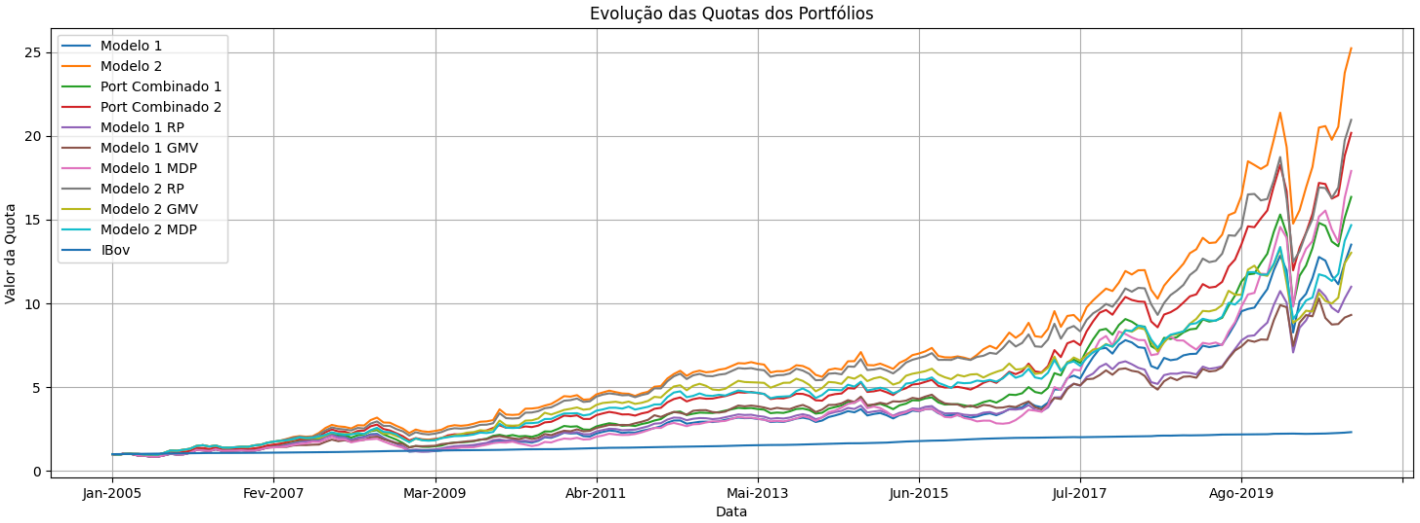
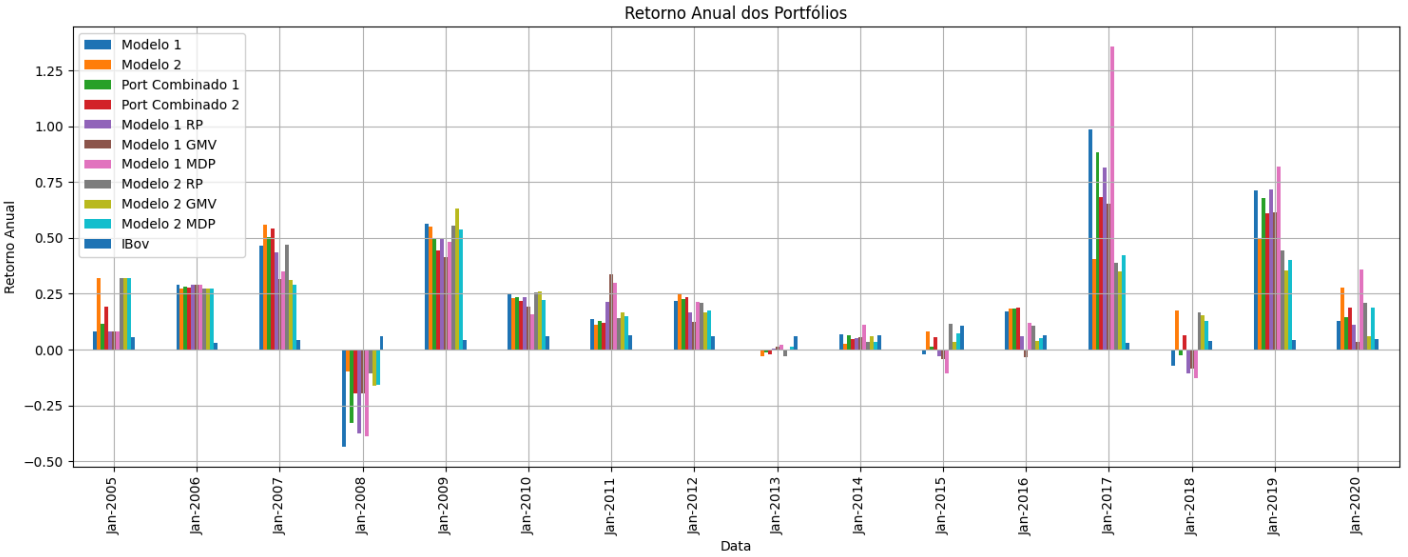
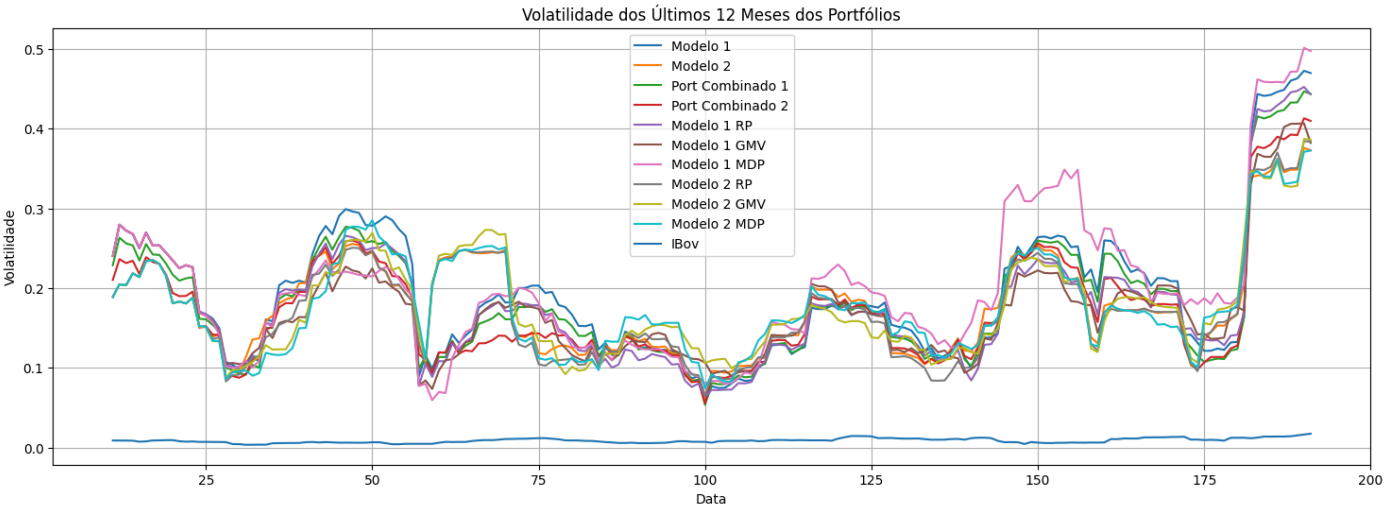
| Adriano da Silva de Carvalho | 13692400 |
|---|---|

# Trabalho 2 SSC0964 - Portfólio de Multifatores e Otimização

## Gráfico dos resultados obtidos:



| Adriano da Silva de Carvalho | 13692400 |
|---|---|

Código:

```
!pip install fpdf matplotlib pandas riskfolio-lib statsmodels

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
import numpy as np
import pandas as pd
import riskfolio as rp
import statsmodels
import matplotlib.pyplot as plt
from fpdf import FPDF


# ESSE TRECHO DE CÓDIGO ESTÁ DEFININDO OS VALORES DE REFERÊNCIA GLOBAIS E ARMAZENANDO OS
DADOS QUE ESTÃO NOS ARQUIVOS .xlsx


##########################################################################################
#########################################################################




data_inicial = 13 #  13 = 01/2005
data_final   = data_inicial + 192 # 12/2020
colunas = 291 #len(comp_indice.columns)
step_port = 1
step_eval = 1


#Composição do índice IBX
comp_indice=pd.read_excel('Dados-Comp-IBRX.xlsx', engine='openpyxl')
comp_indice.set_index(keys = 'Data', inplace = True)


#Preços de fechamento dos ativos
fechamento=pd.read_excel('Dados-Fechamento.xlsx', engine='openpyxl')
fechamento.set_index(keys = 'Data', inplace = True)
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

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


# 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")
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

```
# ESSE TRECHO DE CÓDIGO ESTÁ CRIANDO FUNÇÕES GLOBAIS QUE VÃO SER UTILIZADAS


##########################################################################################
#####################################################################



#Seleção das ações que compõe um portfólio.
#Parâmetros: (fator, ranking_inicio, ranking_fim)
#Retorno: portfólio
def SelPort1(port_ranked_1, param_1a, param_1b):


  port_ranked_final = port_ranked_1.copy()
  port_ranked_final.loc[:, :] = 0


  for lin in range(data_inicial, data_final, step_port):
      for col in range(0, colunas):
          if ((port_ranked_1.iat[lin-1, col]  >= param_1a) and (port_ranked_1.iat[lin-1,
col]  <= param_1b)):
              port_ranked_final.iat[lin-1, col] = 1


  return port_ranked_final
#-------------------------


#Seleção das ações que compõe um portfólio com 2 fatores.
#Parâmetros: (fator1, limite1, fator2, limite2)
#Retorno: portfólio


def SelPort2Par(ranked_1, param_1, ranked_2, param_2):
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, colunas):
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

```
    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
#-------------------------


#Avaliação de um portfólio.
#Parâmetros: (portfólio, histórico de preços dos ativos)
#Retorno: vetor com retorno acumulado, vetor com retornos periódicos, vetor com drawdown,
retorno anualizado, volatilidade anualizada
def EvalPort(port, fechamento):
  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
#--------------------------


#Avaliação de um índice de referência.
#Parâmetros: (dataframe de referências, indice da referência desejada [0 - Ibovespa, 1 - IBX])
##Retorno: vetor com retorno acumulado, vetor com retornos periódicos, vetor com drawdown, retorno anualizado, volatilidade anualizada
def EvalRef(ref, ind):
  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

#Função para calcular otimização de portfólio com Riskfolio-Lib
#Parâmetros: portifólio, fechamento e tipo de otimização
#Retorno: portifólio otimizado
def calc_riskfolio_opt(ranked, fechamento, otim_opt):
  hist_size = 24
  port = ranked.copy()


  for lin in range(data_inicial+hist_size, data_final, 1):
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

```
      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.fillna(method='backfill', axis=0, inplace=True)
      port_comp_chg = port_comp.pct_change().dropna()


      if (otim_opt == 'RP'):
          rp_port = rp.Portfolio(returns=port_comp_chg)
          rp_port.assets_stats(d=0.94, method_cov='hist')
          w = rp_port.rp_optimization(rm='MV', b=None)
      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')
      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


#Função para combinar portfólios
#Parâmetros: portifólios e peso de distribuição
#Retorno: um portifólio combinado
def combine_portfolios(port1, port2, weight1, weight2):
  combined_port = (port1 * weight1 + port2 * weight2) / (weight1 + weight2)
  return combined_port
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

```
# Cálculo 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:\nRet. Acc.:",round(ref_acc_vet[-1]*100-100, 2) ,"% Ret.
Anual.:",round(ret_aa_ref*100,2), "% Vol.:", round(vol_aa_ref*100,2), "% Ret/Vol:",
round(ret_aa_ref/vol_aa_ref, 2), "DDown:", round(np.min(ref_ddown_vet)*100,2), "%")
print("\n")

# Cálculo 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:\nRet. Acc.:",round(ref_acc_vet[-1]*100-100, 2) ,"% Ret.
Anual.:",round(ret_aa_ref*100,2), "% Vol.:", round(vol_aa_ref*100,2), "% Ret/Vol:",
round(ret_aa_ref/vol_aa_ref, 2), "DDown:", round(np.min(ref_ddown_vet)*100,2), "%")
print("\n")

# Cálculo 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:\nRet. Acc.:",round(ref_acc_vet[-1]*100-100, 2) ,"% Ret.
Anual.:",round(ret_aa_ref*100,2), "% Vol.:", round(vol_aa_ref*100,2), "% Ret/Vol:",
round(ret_aa_ref/vol_aa_ref, 2), "DDown:", round(np.min(ref_ddown_vet)*100,2), "%")
print("\n")

# Cálculo de rentabilidade / volatilidade / drawdown do IPCA
ref_acc_vet, ref_chg_vet, ref_ddown_vet, ret_aa_ref, vol_aa_ref = EvalRef(referencias, 3)
print("Ref IPCA:\nRet. Acc.:",round(ref_acc_vet[-1]*100-100, 2) ,"% Ret.
Anual.:",round(ret_aa_ref*100,2), "% Vol.:", round(vol_aa_ref*100,2), "% Ret/Vol:",
round(ret_aa_ref/vol_aa_ref, 2), "DDown:", round(np.min(ref_ddown_vet)*100,2), "%")

#--------------------------

# Parâmetros de seleção dos fatores
param_1_roic = 30  # Top 30 por ROIC
param_1_mom = 30   # Top 30 por Momentum
param_2_mom = 30   # Top 30 por Momentum
param_2_vol = 30   # Top 30 por Baixa Volatilidade


# Seleção dos portfólios baseados nos fatores combinados
port_modelo1 = SelPort2Par(ranked_ROIC, param_1_roic, ranked_Mom, param_1_mom)
port_modelo2 = SelPort2Par(ranked_Mom, param_2_mom, ranked_Vol, param_2_vol)
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

```
# Avaliação dos portfólios
eval_modelo1 = EvalPort(port_modelo1, fechamento)
eval_modelo2 = EvalPort(port_modelo2, fechamento)


# Combinação dos portfólios para atingir o melhor resultado
combined_port1 = combine_portfolios(port_modelo1, port_modelo2, 0.7, 0.3)
eval_combined1 = EvalPort(combined_port1, fechamento)


combined_port2 = combine_portfolios(port_modelo1, port_modelo2, 0.3, 0.7)
eval_combined2 = EvalPort(combined_port2, fechamento)


# Otimização dos portfólios com Riskfolio-Lib
port_modelo1_rp = calc_riskfolio_opt(port_modelo1, fechamento, 'RP')
port_modelo1_gmv = calc_riskfolio_opt(port_modelo1, fechamento, 'GMV')
port_modelo1_mdp = calc_riskfolio_opt(port_modelo1, fechamento, 'MDP')


port_modelo2_rp = calc_riskfolio_opt(port_modelo2, fechamento, 'RP')
port_modelo2_gmv = calc_riskfolio_opt(port_modelo2, fechamento, 'GMV')
port_modelo2_mdp = calc_riskfolio_opt(port_modelo2, fechamento, 'MDP')


# Avaliação dos portfólios otimizados
eval_modelo1_rp = EvalPort(port_modelo1_rp, fechamento)
eval_modelo1_gmv = EvalPort(port_modelo1_gmv, fechamento)
eval_modelo1_mdp = EvalPort(port_modelo1_mdp, fechamento)


eval_modelo2_rp = EvalPort(port_modelo2_rp, fechamento)
eval_modelo2_gmv = EvalPort(port_modelo2_gmv, fechamento)
eval_modelo2_mdp = EvalPort(port_modelo2_mdp, fechamento)


# Função para criar e salvar gráficos
def save_plot(fig, filename):
  fig.savefig(filename, format='png', bbox_inches='tight')
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

```
# Gráfico da evolução das quotas dos portfólios
final_df = pd.DataFrame(index = ranked_ROIC.iloc[data_inicial:data_final+1].index)
final_df['Modelo 1'] = eval_modelo1[0]
final_df['Modelo 2'] = eval_modelo2[0]
final_df['Port Combinado 1'] = eval_combined1[0]
final_df['Port Combinado 2'] = eval_combined2[0]
final_df['Modelo 1 RP'] = eval_modelo1_rp[0]
final_df['Modelo 1 GMV'] = eval_modelo1_gmv[0]
final_df['Modelo 1 MDP'] = eval_modelo1_mdp[0]
final_df['Modelo 2 RP'] = eval_modelo2_rp[0]
final_df['Modelo 2 GMV'] = eval_modelo2_gmv[0]
final_df['Modelo 2 MDP'] = eval_modelo2_mdp[0]
final_df['IBov'] = ref_acc_vet


fig, ax = plt.subplots(figsize=(18, 6))
final_df.plot(ax=ax, grid=True)
ax.set_title('Evolução das Quotas dos Portfólios')
ax.set_xlabel('Data')
ax.set_ylabel('Valor da Quota')
save_plot(fig, 'evolucao_quotas.png')


# Cálculo do retorno anual de cada portfólio e do IBov
final_df12 = pd.DataFrame(columns=['Data', 'Modelo 1', 'Modelo 2', 'Port Combinado 1',
'Port Combinado 2', 'Modelo 1 RP', 'Modelo 1 GMV', 'Modelo 1 MDP', 'Modelo 2 RP', 'Modelo 2
GMV', 'Modelo 2 MDP', 'IBov'])
for ind in range(0, len(final_df.index)-12, 12):
  final_temp = final_df.iloc[ind+12]/final_df.iloc[ind]-1
  final_df12 = pd.concat([final_df12, pd.DataFrame([final_temp])], ignore_index=True)
  final_df12.iat[len(final_df12)-1, 0] = final_df.index[ind]


final_df12.set_index(keys='Data', inplace=True)


# Gráfico de retorno anual
fig, ax = plt.subplots(figsize=(18, 6))
final_df12.plot.bar(ax=ax, grid=True)
ax.set_title('Retorno Anual dos Portfólios')
ax.set_xlabel('Data')
ax.set_ylabel('Retorno Anual')
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

```
save_plot(fig, 'retorno_anual.png')


# Cálculo da volatilidade dos últimos 12 meses para cada portfólio e o IBov
final_vol_df = pd.DataFrame()
final_vol_df['Modelo 1'] =
pd.Series(eval_modelo1[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Modelo 2'] =
pd.Series(eval_modelo2[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Port Combinado 1'] =
pd.Series(eval_combined1[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Port Combinado 2'] =
pd.Series(eval_combined2[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Modelo 1 RP'] =
pd.Series(eval_modelo1_rp[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Modelo 1 GMV'] =
pd.Series(eval_modelo1_gmv[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Modelo 1 MDP'] =
pd.Series(eval_modelo1_mdp[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Modelo 2 RP'] =
pd.Series(eval_modelo2_rp[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Modelo 2 GMV'] =
pd.Series(eval_modelo2_gmv[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['Modelo 2 MDP'] =
pd.Series(eval_modelo2_mdp[1]).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))
final_vol_df['IBov'] =
pd.Series(ref_chg_vet).rolling(int(12/step_eval)).std()*(int(12/step_eval)**(1/2))


# Gráfico de volatilidade
fig, ax = plt.subplots(figsize=(18, 6))
final_vol_df.plot(ax=ax, grid=True)
ax.set_title('Volatilidade dos Últimos 12 Meses dos Portfólios')
ax.set_xlabel('Data')
ax.set_ylabel('Volatilidade')
save_plot(fig, 'volatilidade.png')


# Criação do PDF com os gráficos
pdf = FPDF()
pdf.add_page()
pdf.set_font("Arial", size=12)
```

| Adriano da Silva de Carvalho | 13692400 |
|---|---|

```
pdf.cell(200, 10, txt="Evolução das Quotas dos Portfólios", ln=True, align='C')
pdf.image("evolucao_quotas.png", x=10, y=20, w=190)


pdf.add_page()
pdf.cell(200, 10, txt="Retorno Anual dos Portfólios", ln=True, align='C')
pdf.image("retorno_anual.png", x=10, y=20, w=190)


pdf.add_page()
pdf.cell(200, 10, txt="Volatilidade dos Últimos 12 Meses dos Portfólios", ln=True,
align='C')
pdf.image("volatilidade.png", x=10, y=20, w=190)


pdf.output("portfolios_report.pdf")


print("PDF com os gráficos foi gerado com sucesso.")
```