Carteira usando Markowitz

February 24, 2021

Neste exercício utilizaremos os dados históricos das ações do índice Bovespa durante o mês de dezembro de 2020 para criar uma carteira hipotética de ações como sugestão de investimento para o mês de janeiro de 2021. Para isso usaremos a fronteira eficiente de Markowitz além de comparar a suposta rentabilidade da carteira comparada ao ibov

Desafio Automatizar o backtest para que seja feita a análise de n+1 meses e entregue o resultado como no exercício de médias móveis.

[1]:
```
import numpy as np
import pandas as pd
from pandas import set_option
import yfinance as yf
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

0.0.1 Extraindo o histórico das ações do IBOV

[23]:
```
ibovurl= "http://bvmf.bmfbovespa.com.br/indices/ResumoCarteiraTeorica.aspx?
 ↪Indice=ibov&idioma=pt-br"
ibov = pd.read_html(ibovurl)

# Selecionando apenas a coluna com os Tickers

tickers = ibov[0][0:]['Código'].tolist()
tickers.remove('Quantidade Teórica Total  Redutor')

# Adicionando .SA em todas as ações

for i in range(0, len(tickers)):
    tickers[i] = tickers[i] + '.SA'



# Buscando no yahoo finance o preço de fechamento ajustado.

dataset = yf.download(tickers=tickers, start='2020-12-01', end =␣
 ↪'2020-12-30')[['Adj Close']]
```

```
[**********100%**********]  81 of 81 completed
```

Nesta etapa, verificamos os NAs nas linhas e os eliminamos. Vamos nos livrar das colunas com mais de 25% de valores ausentes

[24]:
```
miss = dataset.isnull().mean().sort_values(ascending=False)
miss.head(10)
drop_list = sorted(list(miss[miss > 0.25].index))
dataset.drop(labels=drop_list, axis=1, inplace=True)
dataset.shape

# Eliminando as linhas que ainda possuirem NA's
dataset=dataset.dropna()
dataset.head()
```

[24]:

	Adj Close						\
	ABEV3.SA	AZUL4.SA	B3SA3.SA	BBAS3.SA	BBDC3.SA	BBDC4.SA	
Date							
2020-12-01	14.234010	38.939999	56.269363	34.160362	22.385792	25.201660	
2020-12-02	14.427935	40.520000	59.064003	34.307648	22.589430	25.475851	
2020-12-03	14.253403	42.279999	58.360390	34.602219	22.432220	25.141680	
2020-12-04	14.292187	42.000000	58.241470	35.328823	22.550127	25.357908	
2020-12-07	14.515200	41.889999	57.934258	35.495747	22.628735	25.524996	

					…	\
	BBSE3.SA	BEEF3.SA	BPAC11.SA	BRAP4.SA	…	TAEE11.SA
Date					…	
2020-12-01	27.908201	9.788735	81.270020	60.279999	…	32.935101
2020-12-02	28.065542	9.689155	78.241264	59.080002	…	32.586739
2020-12-03	28.262217	9.768820	85.738678	58.570000	…	33.124210
2020-12-04	28.252382	9.788735	81.508354	61.200001	…	32.606644
2020-12-07	28.488392	9.967979	81.994942	62.310001	…	33.472572

							\
	TIMS3.SA	TOTS3.SA	UGPA3.SA	USIM5.SA	VALE3.SA	VIVT3.SA	
Date							
2020-12-01	13.550141	25.668327	21.280001	13.90	81.250000	44.613266	
2020-12-02	13.727139	25.598577	20.809999	13.82	79.839996	43.367191	
2020-12-03	14.012301	25.229893	21.250000	13.11	78.959999	43.170959	
2020-12-04	14.140133	26.017082	22.000000	14.02	81.980003	42.974728	
2020-12-07	13.904137	25.748043	21.920000	14.46	82.949997	42.817741	

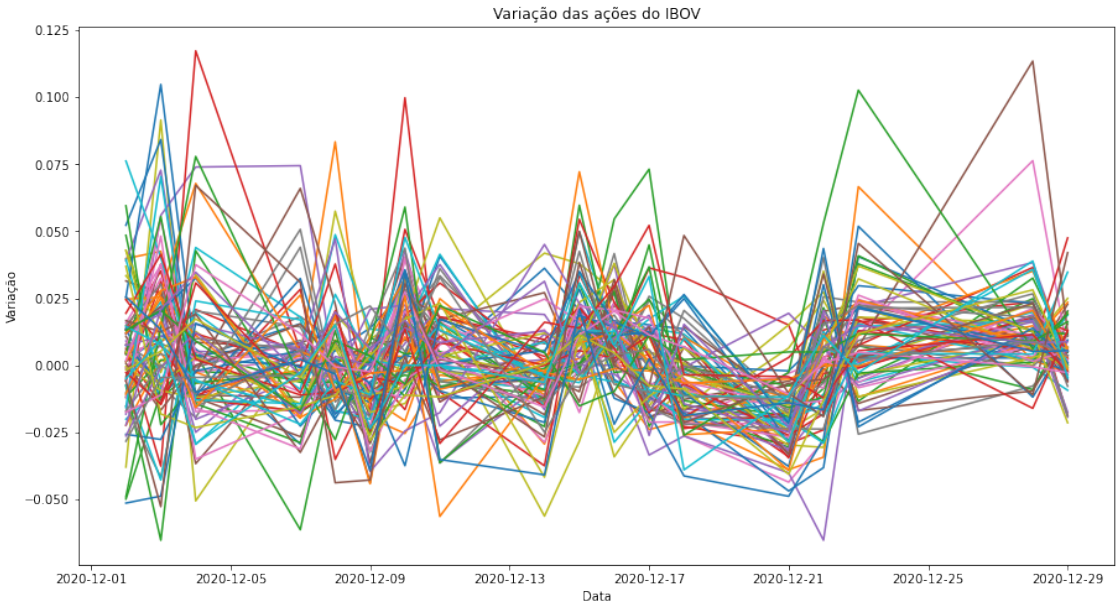
	VVAR3.SA	WEGE3.SA	YDUQ3.SA
Date			
2020-12-01	17.150000	73.562744	35.630001
2020-12-02	17.120001	73.192940	36.549999
2020-12-03	17.209999	73.892578	37.040001

```
2020-12-04  17.510000  71.763664  36.669998
2020-12-07  17.430000  71.573753  36.750000

[5 rows x 81 columns]
```

0.0.2 Transformando os dados (Normalizando)

[25]:
```
data_log = np.log(dataset / dataset.shift(1))
```

[26]:
```
plt.figure(figsize=(15,8))
plt.plot(data_log)
plt.xlabel("Data")
plt.ylabel('Variação')
plt.title('Variação das ações do IBOV')
plt.show()
```



0.1 Carteira ideal para Janeiro/2021

Análise por Índice de Sharpe Iremos nesta etapa da análise simular 50000 carteiras e computar, por carteira, o índice de Sharpe

O indice mensura a relação entre retorno e risco de um investimento. Por meio de seu resultado, você descobre se há vantagem entre determinado fundo de investimento na comparação com os ativos livres de risco.

Em outras palavras, o Índice de Sharpe destaca qual alternativa tende a trazer a melhor remuneração com o menor risco possível.

[27]:
```
n_portfolio = 50000
all_weights = np.zeros((n_portfolio, len(data_log.columns)))
ret_arr = np.zeros(n_portfolio)
vol_arr = np.zeros(n_portfolio)
sharpe_arr = np.zeros(n_portfolio)

for x in range(n_portfolio):
    # Weights
    weights = np.array(np.random.random(81))
    weights = weights/np.sum(weights)

    # Save weights
    all_weights[x,:] = weights

    # Expected return
    ret_arr[x] = np.sum( (data_log.mean() * weights * 252))

    # Expected volatility
    vol_arr[x] = np.sqrt(np.dot(weights.T, np.dot(data_log.cov()*252, weights)))

    # Sharpe Ratio
    sharpe_arr[x] = ret_arr[x]/vol_arr[x]
```

[28]:
```
print(f'índice de Sharp máximo: {sharpe_arr.max():.4f}')
```

```
índice de Sharp máximo: 7.7618
```

0.1.1 Adicionando Markowitz

[29]:
```
def get_ret_vol_sr(weights):
    weights = np.array(weights)
    ret = np.sum(data_log.mean() * weights)
    vol = np.sqrt(np.dot(weights.T, np.dot(data_log.cov(), weights)))
    sr = ret / vol
    return np.array([ret, vol, sr])

def neg_sharpe(weights):
# the number 2 is the sharpe ratio index from the get_ret_vol_sr
    return get_ret_vol_sr(weights)[2] * - 1

def check_sum(weights):
    #return 0 if sum of the weights is 1
    return np.sum(weights) - 1
```

[30]:
```
from scipy.optimize import minimize
```

[31]:
```python
cons = ({'type':'eq', 'fun':check_sum})
bounds = ((0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
          (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
          (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
          (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
          (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
          (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
          (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
          (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1))
init_guess = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
              0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
              0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
              0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
              0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
              0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
              0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,
              0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
```

[32]:
```python
otim_resultados = minimize(neg_sharpe, init_guess, method = 'SLSQP', bounds = bounds, constraints = cons)
```

[33]:
```python
print(f"índice de Sharp máximo: {get_ret_vol_sr(otim_resultados['x'])[2]:.4f}")
```

```
índice de Sharp máximo: 1.4854
```

[42]:
```python
pesos_mark = otim_resultados['x']
print('-=' * 15)
print('Portfólio selecionado'.center(30))
print('-=' * 15)
for i in range(len(tickers)):
    print(f'{tickers[i]}: {pesos_mark[i] * 100:4f}%')
```

```
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    Portfólio selecionado
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
ABEV3.SA: 22.635830%
AZUL4.SA: 0.000000%
B3SA3.SA: 0.000000%
BBAS3.SA: 0.000000%
```

BBDC3.SA: 0.000000%
BBDC4.SA: 0.000000%
BBSE3.SA: 0.000000%
BEEF3.SA: 0.000000%
BPAC11.SA: 6.995320%
BRAP4.SA: 0.000000%
BRDT3.SA: 0.000000%
BRFS3.SA: 0.000000%
BRKM5.SA: 0.000000%
BRML3.SA: 0.000000%
BTOW3.SA: 0.000000%
CCRO3.SA: 0.000000%
CIEL3.SA: 0.000000%
CMIG4.SA: 0.672644%
COGN3.SA: 0.000000%
CPFE3.SA: 0.000000%
CPLE6.SA: 10.277154%
CRFB3.SA: 0.000000%
CSAN3.SA: 0.000000%
CSNA3.SA: 5.580815%
CVCB3.SA: 0.000000%
CYRE3.SA: 0.000000%
ECOR3.SA: 0.000000%
EGIE3.SA: 0.000000%
ELET3.SA: 0.000000%
ELET6.SA: 0.000000%
EMBR3.SA: 0.000000%
ENBR3.SA: 0.000000%
ENEV3.SA: 0.000000%
ENGI11.SA: 0.000000%
EQTL3.SA: 0.000000%
EZTC3.SA: 0.000000%
FLRY3.SA: 0.000000%
GGBR4.SA: 0.000000%
GNDI3.SA: 6.264099%
GOAU4.SA: 0.000000%
GOLL4.SA: 0.000000%
HAPV3.SA: 0.000000%
HGTX3.SA: 0.000000%
HYPE3.SA: 0.000000%
IGTA3.SA: 0.000000%
IRBR3.SA: 0.000000%
ITSA4.SA: 0.000000%
ITUB4.SA: 0.000000%
JBSS3.SA: 0.000000%
JHSF3.SA: 0.000000%
KLBN11.SA: 0.000000%
LAME4.SA: 0.000000%

```
LCAM3.SA: 0.000000%
LREN3.SA: 0.000000%
MGLU3.SA: 11.802883%
MRFG3.SA: 0.000000%
MRVE3.SA: 0.000000%
MULT3.SA: 0.000000%
NTCO3.SA: 0.000000%
PCAR3.SA: 0.000000%
PETR3.SA: 0.000000%
PETR4.SA: 0.000000%
PRIO3.SA: 0.000000%
QUAL3.SA: 0.000000%
RADL3.SA: 0.000000%
RAIL3.SA: 0.000000%
RENT3.SA: 0.000000%
SANB11.SA: 4.632018%
SBSP3.SA: 0.000000%
SULA11.SA: 0.000000%
SUZB3.SA: 0.000000%
TAEE11.SA: 0.000000%
TIMS3.SA: 29.632386%
TOTS3.SA: 0.000000%
UGPA3.SA: 1.506852%
USIM5.SA: 0.000000%
VALE3.SA: 0.000000%
VIVT3.SA: 0.000000%
VVAR3.SA: 0.000000%
WEGE3.SA: 0.000000%
YDUQ3.SA: 0.000000%
```

[43]:
```
resultados_mark = get_ret_vol_sr(otim_resultados['x'])
print(f'Método de Markowitz | Retorno de {resultados_mark[0]:.4f} e␣
 ↪{resultados_mark[1]:.4f} de volatilidade')
```

```
Método de Markowitz | Retorno de 0.0064 e 0.0043 de volatilidade
```

Iniciando a análise da carteira recomendada para o mês de jan/2021

[49]:
```
acao = ['ABEV3.SA','BPAC11.SA','CMIG4.SA','CPLE6.SA','CSNA3.SA',
        'GNDI3.SA', 'MGLU3.SA', 'SANB11.SA', 'TIMS3.SA','UGPA3.SA']

pesos = [0.226, 0.07, 0.006, 0.103, 0.055,
         0.066, 0.1180, 0.045, 0.296, 0.015]

carteira = yf.download(tickers=acao, start='2021-01-01', end =␣
 ↪'2021-01-31')[['Adj Close']]
ibov = yf.download('^BVSP', start = '2021-01-01', end = '2021-01-31')[['Adj␣
 ↪Close']]
```

```
[**********100%**********]  10 of 10 completed
[**********100%**********]  1 of 1 completed
```

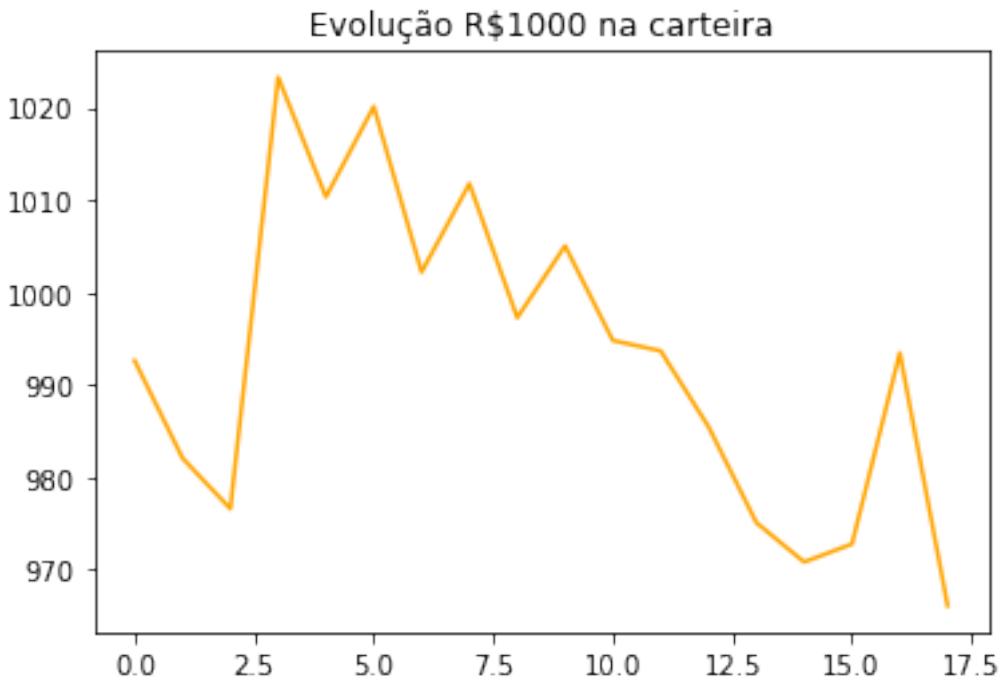
[50]:
```
### Transformando os dados em retornos

ret_acao = carteira.pct_change()
ret_ibov = ibov.pct_change()

# Retorno Acumulado

ret_acu = (1+ ret_acao).cumprod()

ret_ibov_acu = (1 + ret_ibov).cumprod()

### Retorno do Portfolio

portfolio = np.dot(ret_acao, pesos)
pd.DataFrame(portfolio, columns = ['R'])
```

[50]:

	R
0	NaN
1	-0.007388
2	-0.010615
3	-0.005585
4	0.047831
5	-0.012660
6	0.009681
7	-0.017565
8	0.009553
9	-0.014343
10	0.007798
11	-0.010196
12	-0.001128
13	-0.008261
14	-0.010537
15	-0.004372
16	0.002007
17	0.021273
18	-0.027567

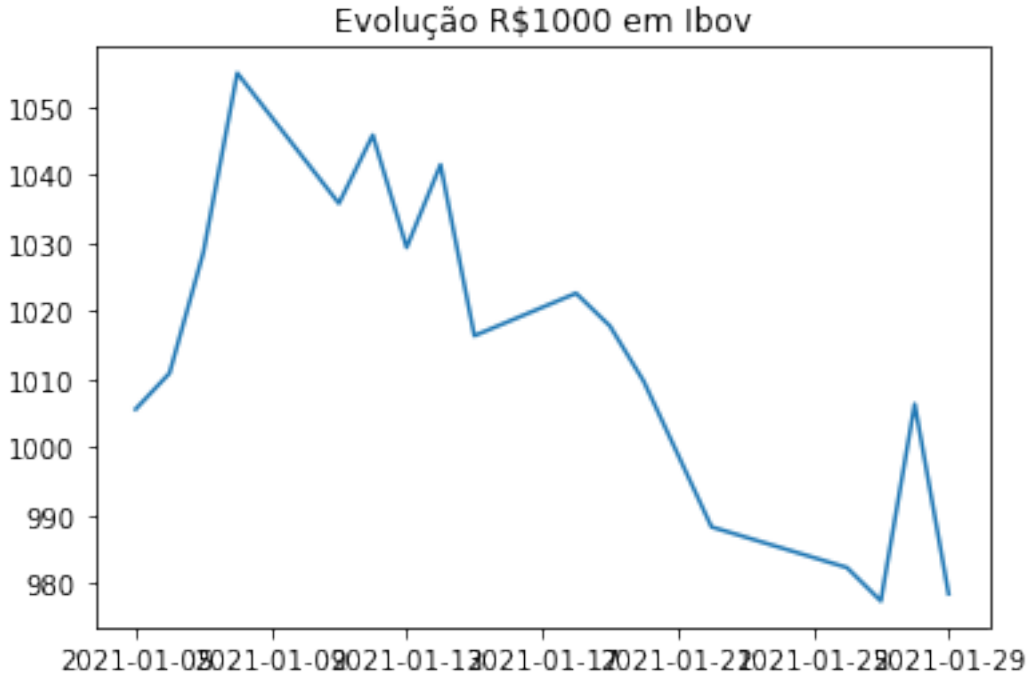
[51]:
```
plt.plot(1000*(1+portfolio[1:]).cumprod(), color = 'orange')
plt.title('Evolução R$1000 na carteira')
```

[51]:
```
Text(0.5, 1.0, 'Evolução R$1000 na carteira')
```



[52]:
```
plt.plot(1000*(1+ibov.pct_change()).cumprod())
plt.title('Evolução R$1000 em Ibov')
```

[52]: Text(0.5, 1.0, 'Evolução R$1000 em Ibov')



Referência https://github.com/leoIeracitano/First_projects/blob/main/Fronteira%20Eficiente%20de%20Mark