Teste 1

April 6, 2021

Qual a chance de 10 ações dentro do indice Ibovespa escolhidas aleatoriamente baterem o índice? 10 acoes com 10% cada.

Caso aconteça de uma ação com um alto peso no ibov subir um valor expressivo(e.g 200%) no mês, a chance de 10 ações aleatórias baterem o ibov é menor do que se nenhuma ação subir muito.

Teste para diferentes valores de n (numero de ações na carteira). Queremos seguir o indice bovespa porém queremos ter o menor erro com o menor número de ações possíveis.

Qual é o N e qual é a taxa de erro médio?

[1]:
```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import numpy as np
import random
```

[2]:
```
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='2020-01-01')[['Adj Close']]
```

```
[**********100%**********]  82 of 82 completed
```

1 Retirando NA's

[3]:
```
missing_fractions = dataset.isnull().mean().sort_values(ascending=False)

missing_fractions.head(10)

drop_list = sorted(list(missing_fractions[missing_fractions > 0.25].index))

dataset.drop(labels=drop_list, axis=1, inplace=True)
dataset.shape
```

[3]:
```
(311, 81)
```

[4]:
```
retorno = dataset.pct_change()
retorno
```

[4]:
```
            Adj Close                                                    \
             ABEV3.SA  AZUL4.SA  B3SA3.SA  BBAS3.SA  BBDC3.SA  BBDC4.SA
Date
2020-01-02       NaN       NaN       NaN       NaN       NaN       NaN
2020-01-03 -0.014063 -0.034694 -0.028818 -0.001673 -0.015604  0.000506
2020-01-06  0.004754 -0.031008 -0.009587 -0.013219 -0.005736 -0.017862
2020-01-07  0.002103  0.033091  0.032265 -0.007547 -0.017018 -0.017372
2020-01-08 -0.004722  0.001760 -0.002679 -0.009125 -0.015258 -0.015470
…               …         …         …         …         …         …
2021-03-30  0.018954  0.069353 -0.001866  0.030872  0.023474  0.016722
2021-03-31 -0.019243 -0.029736  0.020935 -0.008789 -0.019600 -0.021564
2021-04-01 -0.018967  0.003699 -0.018491 -0.016420 -0.034453 -0.036608
2021-04-02  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2021-04-05  0.018000  0.047644  0.013244 -0.001336  0.012992  0.010016

                                                                  …      \
            BBSE3.SA  BEEF3.SA BPAC11.SA  BRAP4.SA  … TAEE11.SA  TIMS3.SA
Date                                                  …
2020-01-02       NaN       NaN       NaN       NaN  …       NaN       NaN
2020-01-03 -0.011023  0.026253  0.002902  0.014199  … -0.001297  0.001902
2020-01-06  0.016454 -0.000775 -0.001447  0.004667  …  0.006169 -0.001266
2020-01-07  0.000522  0.045772  0.003990  0.004387  … -0.011939  0.010773
2020-01-08 -0.000522  0.011128 -0.008862 -0.000257  … -0.010124  0.010658
…               …         …         …         …  …         …         …
2021-03-30  0.029497 -0.010506  0.062493 -0.001465  …  0.021371  0.013375
2021-03-31 -0.021388 -0.015444 -0.003798  0.001907  … -0.006890 -0.016304
2021-04-01 -0.013608  0.000000 -0.025760 -0.032843  … -0.026465 -0.015785
2021-04-02  0.000000  0.000000  0.000000  0.000000  …  0.000000  0.000000
2021-04-05  0.006271  0.003922  0.041248  0.057692  …  0.020322 -0.011227

                                                                       \
```

```
            TOTS3.SA  UGPA3.SA  USIM5.SA  VALE3.SA  VIVT3.SA  VVAR3.SA
Date
2020-01-02       NaN       NaN       NaN       NaN       NaN       NaN
2020-01-03  0.021258 -0.011719 -0.010309 -0.007362  0.011608 -0.021313
2020-01-06 -0.022798 -0.021739 -0.018750 -0.005934 -0.009631  0.000000
2020-01-07  0.004202  0.022222  0.009554  0.007275  0.022139  0.014808
2020-01-08 -0.013420 -0.007905 -0.012618  0.000185  0.010121 -0.004292
…               …         …         …         …         …         …
2021-03-30  0.040545  0.014755  0.011998 -0.009288 -0.000439  0.044877
2021-03-31 -0.002414 -0.005160  0.013634  0.009272 -0.027912 -0.021880
2021-04-01 -0.006913 -0.019802 -0.028655 -0.005920  0.001591  0.014913
2021-04-02  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2021-04-05  0.040724  0.014911  0.027092  0.061608  0.007250  0.022041


            WEGE3.SA  YDUQ3.SA
Date
2020-01-02       NaN       NaN
2020-01-03 -0.012770 -0.020963
2020-01-06  0.002587  0.019227
2020-01-07  0.007741 -0.018650
2020-01-08 -0.037838 -0.005898
…               …         …
2021-03-30  0.040915  0.049794
2021-03-31 -0.006529 -0.047432
2021-04-01  0.006706 -0.002621
2021-04-02  0.000000  0.000000
2021-04-05  0.024647  0.086712

[311 rows x 81 columns]
```

[5]:
```
ibov = yf.download('BOVA11.SA', start = '2020-01-01')["Adj Close"]
ibov = ibov / ibov.iloc[0]
```

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

[6]:
```
retorno_acumulado = (1 + retorno).cumprod()
retorno_acumulado.iloc[0] = 1
```

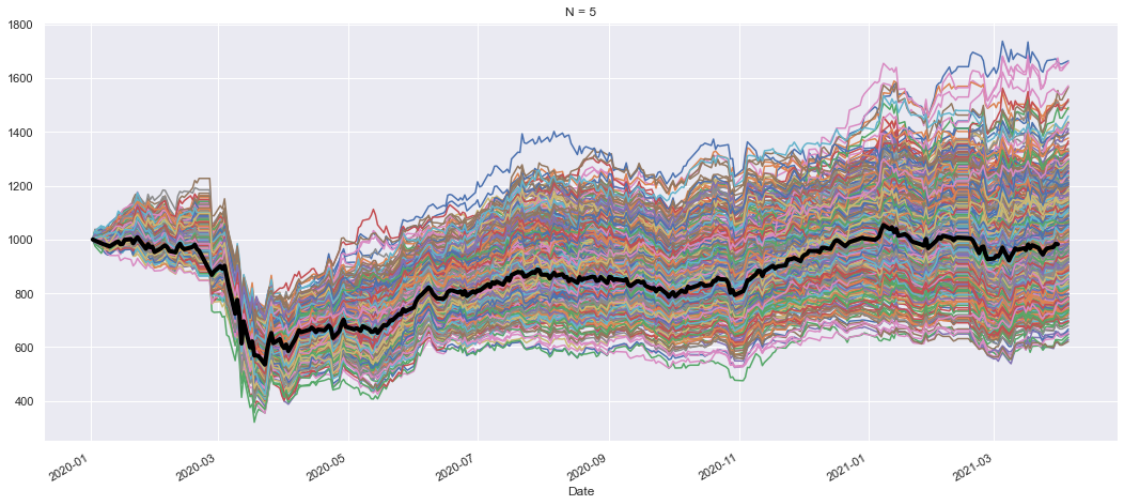
Primeiramente vamos rodar 500 carteiras com N = 5, 10, 25 e 50 para analisar como a dispersão das carteiras aleatorias se comportam em relação ao bova11

[7]:
```
for i in range(500):
  plt.title('N = 5')
  carteira = random.sample(list(dataset.columns) , k=5)
  carteira = 200 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
```

```
  carteira['saldo'].plot(figsize=(18,8))


(ibov*1000).plot(linewidth=4, color='black')
```

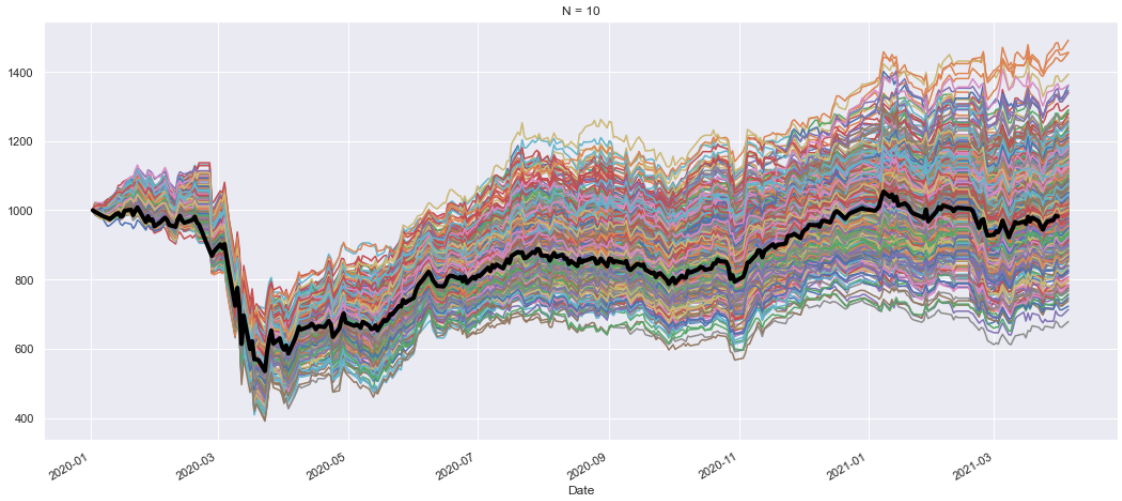
[7]: <AxesSubplot:title={'center':'N = 5'}, xlabel='Date'>



[8]:
```
for i in range(500):
  plt.title('N = 10')
  carteira = random.sample(list(dataset.columns) , k=10)
  carteira = 100 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))


(ibov*1000).plot(linewidth=4, color='black')
```

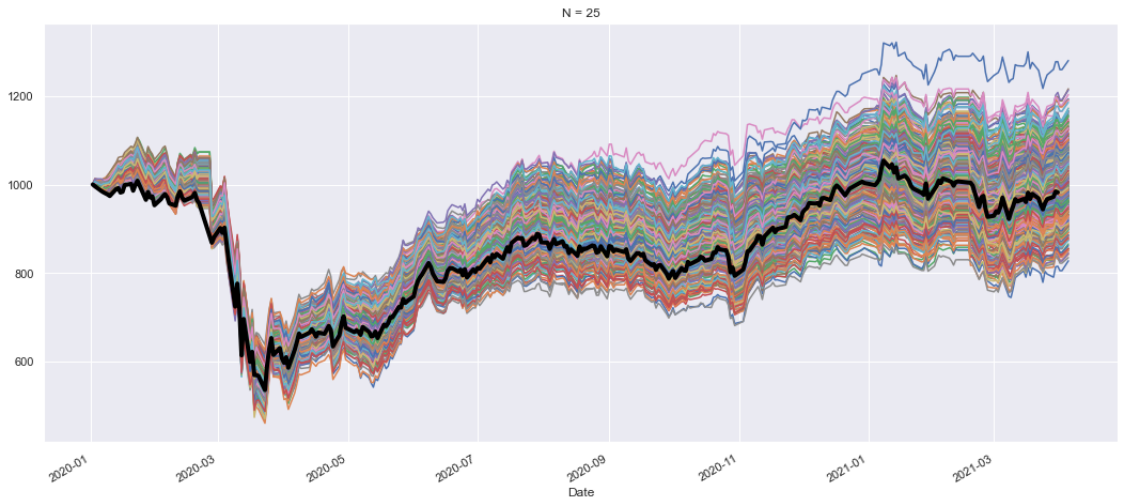
[8]: <AxesSubplot:title={'center':'N = 10'}, xlabel='Date'>



[9]:
```
for i in range(500):
  plt.title('N = 25')
  carteira = random.sample(list(dataset.columns) , k=25)
  carteira = 40 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))


(ibov*1000).plot(linewidth=4, color='black')
```

[9]: <AxesSubplot:title={'center':'N = 25'}, xlabel='Date'>



[10]:
```
for i in range(500):
  plt.title('N = 50')
  carteira = random.sample(list(dataset.columns) , k=50)
  carteira = 20 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))


(ibov*1000).plot(linewidth=4, color='black')
```

[10]:
```
<AxesSubplot:title={'center':'N = 50'}, xlabel='Date'>
```

Até esse ponto conseguimos ver claramente que, quanto mais ações adicionamos em nossa carteira, maior será a chance dela estar próxima do Ibov. Testaremos agora 6 carteiras aumentando nosso N de 5 até 10 para encontrar qual é o valor de N que nos retornará o melhor resultado possível com o menor erro médio. Para um melhor resultado iremos rodar 1000 carteiras em cada N

2 Testando carteiras de N=5 até N=10

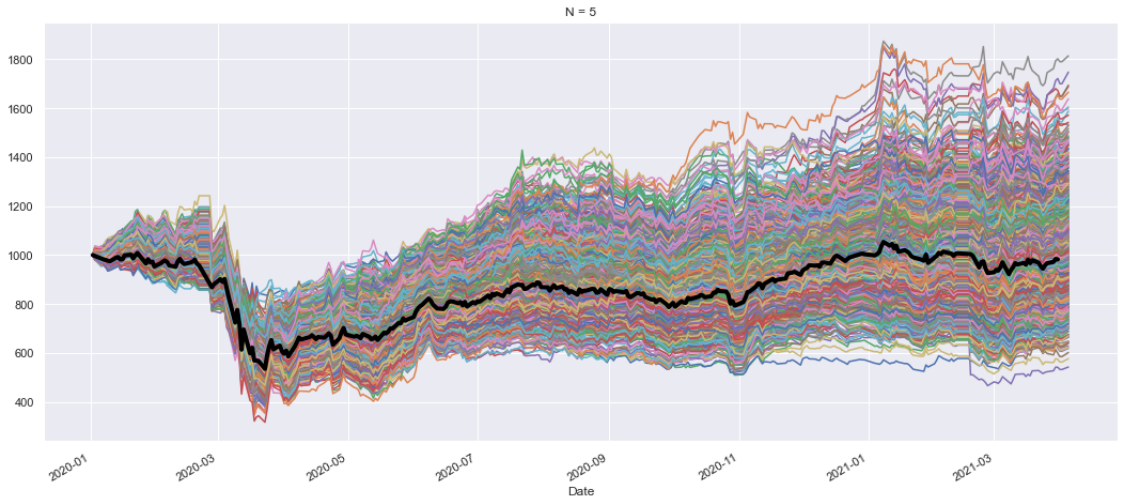
2.1 Para N = 5

[11]:
```
N_5 = []
saldos_finais= []
for i in range(1000):
  plt.title('N = 5')
  carteira = random.sample(list(dataset.columns) , k=5)
  carteira = 200 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))
  erro = ((carteira['saldo'] - ibov)**(2)).mean()
```

```
  N_5.append(erro)
  saldos_finais.append(carteira['saldo'].iloc[-1])


(ibov*1000).plot(linewidth=4, color='black')
```

[11]: <AxesSubplot:title={'center':'N = 5'}, xlabel='Date'>



[12]:
```
bateu_ibov = 0
saldo_final_ibov = ibov.iloc[-1]*1000
for saldo in saldos_finais:
    if (saldo > saldo_final_ibov):
        bateu_ibov += 1
bateu_ibov_porcento = round((bateu_ibov/1000)*100,2)
print("{} de 1000 carteiras ({}%) bateram o IBOV no período".format(bateu_ibov,␣
 ↪bateu_ibov_porcento))
```

```
540 de 1000 carteiras (54.0%) bateram o IBOV no período
```

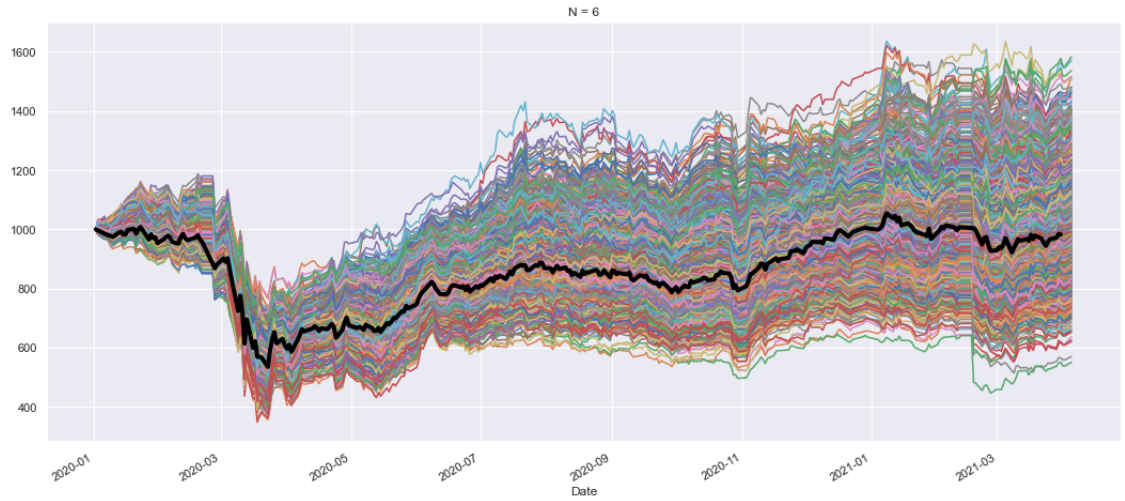
2.2 Para N = 6

[13]:
```
N_6 = []
saldos_finais= []
for i in range(1000):
  plt.title('N = 6')
  carteira = random.sample(list(dataset.columns) , k=6)
  carteira = 166.66 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))
  erro = ((carteira['saldo'] - ibov)**(2)).mean()
```

```
  N_6.append(erro)
  saldos_finais.append(carteira['saldo'].iloc[-1])


(ibov*1000).plot(linewidth=4, color='black')
```

[13]: <AxesSubplot:title={'center':'N = 6'}, xlabel='Date'>



[14]:
```
bateu_ibov = 0
saldo_final_ibov = ibov.iloc[-1]*1000
for saldo in saldos_finais:
    if (saldo > saldo_final_ibov):
        bateu_ibov += 1
bateu_ibov_porcento = round((bateu_ibov/1000)*100,2)
print("{} de 1000 carteiras ({}%) bateram o IBOV no período".format(bateu_ibov,␣
 ↪bateu_ibov_porcento))
```

```
533 de 1000 carteiras (53.3%) bateram o IBOV no período
```

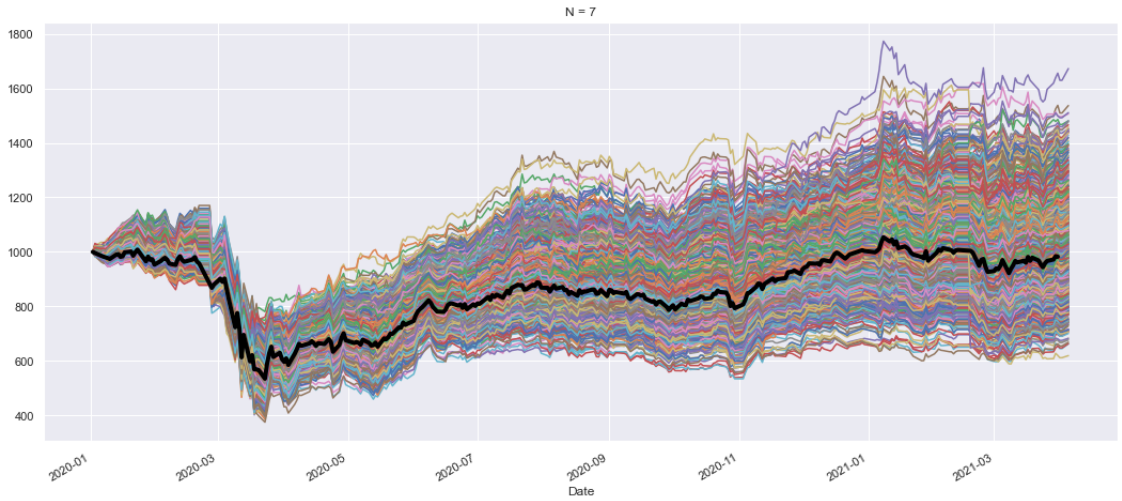
2.3 Para N = 7

[15]:
```
N_7 = []
saldos_finais= []
for i in range(1000):
  plt.title('N = 7')
  carteira = random.sample(list(dataset.columns) , k=7)
  carteira = 142.86 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))
  erro = ((carteira['saldo'] - ibov)**(2)).mean()
```

```
  N_7.append(erro)
  saldos_finais.append(carteira['saldo'].iloc[-1])


(ibov*1000).plot(linewidth=4, color='black')
```

[15]: <AxesSubplot:title={'center':'N = 7'}, xlabel='Date'>



[16]:
```
bateu_ibov = 0
saldo_final_ibov = ibov.iloc[-1]*1000
for saldo in saldos_finais:
    if (saldo > saldo_final_ibov):
        bateu_ibov += 1
bateu_ibov_porcento = round((bateu_ibov/1000)*100,2)
print("{} de 1000 carteiras ({}%) bateram o IBOV no período".format(bateu_ibov,␣
 ↪bateu_ibov_porcento))
```

```
547 de 1000 carteiras (54.7%) bateram o IBOV no período
```

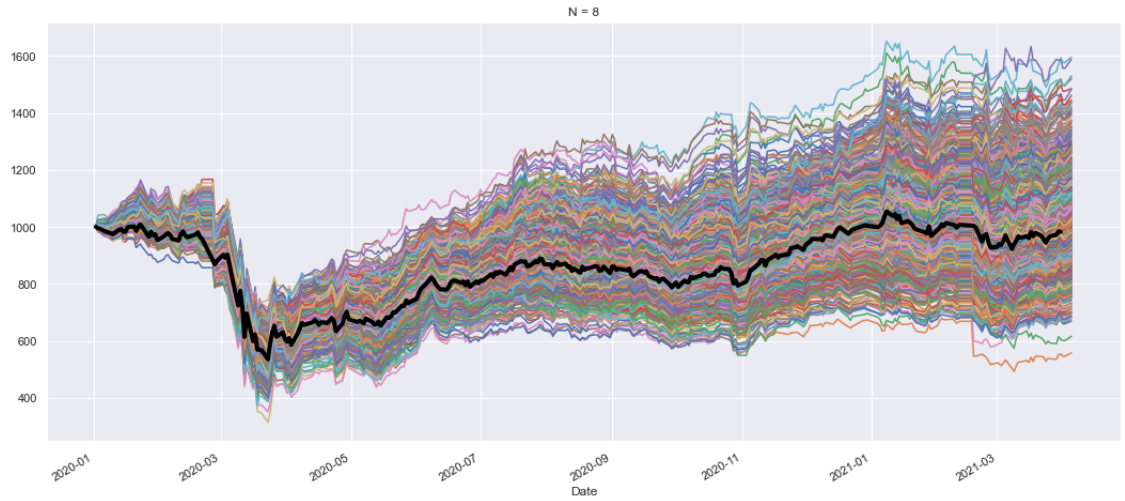
2.4 Para N = 8

[17]:
```
N_8 = []
saldos_finais= []
for i in range(1000):
  plt.title('N = 8')
  carteira = random.sample(list(dataset.columns) , k=8)
  carteira = 125 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))
  erro = ((carteira['saldo'] - ibov)**(2)).mean()
```

```
  N_8.append(erro)
  saldos_finais.append(carteira['saldo'].iloc[-1])

(ibov*1000).plot(linewidth=4, color='black')
```

[17]: <AxesSubplot:title={'center':'N = 8'}, xlabel='Date'>



[18]:
```
bateu_ibov = 0
saldo_final_ibov = ibov.iloc[-1]*1000
for saldo in saldos_finais:
    if (saldo > saldo_final_ibov):
        bateu_ibov += 1
bateu_ibov_porcento = round((bateu_ibov/1000)*100,2)
print("{} de 1000 carteiras ({}%) bateram o IBOV no período".format(bateu_ibov,␣
 ↪bateu_ibov_porcento))
```

```
557 de 1000 carteiras (55.7%) bateram o IBOV no período
```

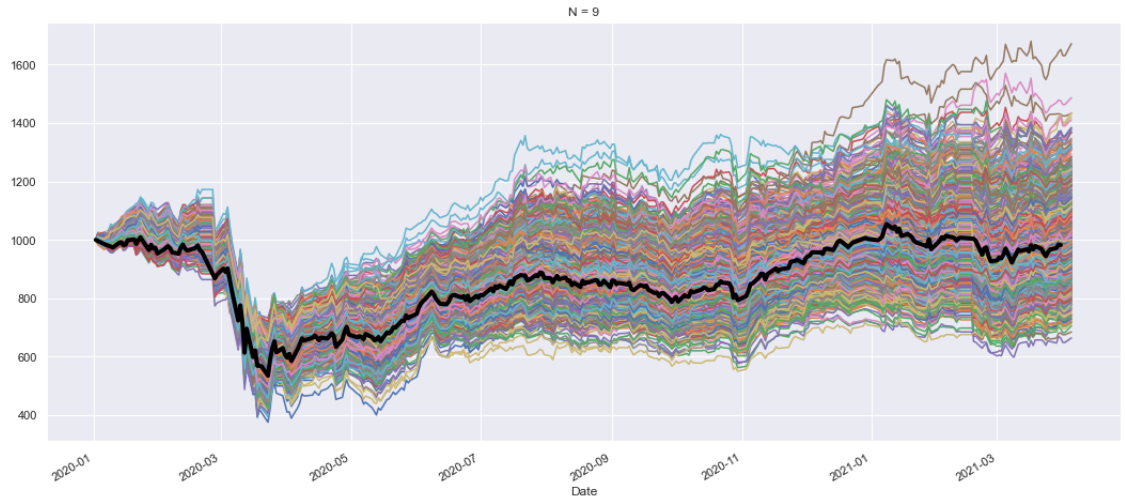
2.5 Para N = 9

[19]:
```
N_9 = []
saldos_finais= []
for i in range(1000):
  plt.title('N = 9')
  carteira = random.sample(list(dataset.columns) , k=9)
  carteira = 111.11 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))
  erro = ((carteira['saldo'] - ibov)**(2)).mean()
  N_9.append(erro)
```

```
  saldos_finais.append(carteira['saldo'].iloc[-1])


(ibov*1000).plot(linewidth=4, color='black')
```

[19]: <AxesSubplot:title={'center':'N = 9'}, xlabel='Date'>



[20]:
```
bateu_ibov = 0
saldo_final_ibov = ibov.iloc[-1]*1000
for saldo in saldos_finais:
    if (saldo > saldo_final_ibov):
        bateu_ibov += 1
bateu_ibov_porcento = round((bateu_ibov/1000)*100,2)
print("{} de 1000 carteiras ({}%) bateram o IBOV no período".format(bateu_ibov,␣
 ↪bateu_ibov_porcento))
```

```
569 de 1000 carteiras (56.9%) bateram o IBOV no período
```

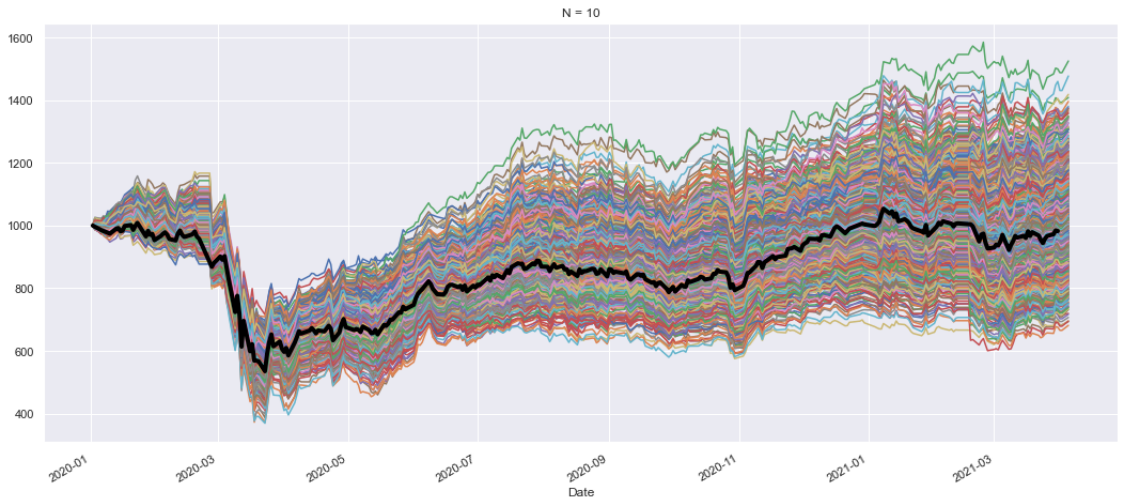
2.6 Para N = 10

[21]:
```
saldos_finais= []
N_10 = []
for i in range(1000):
  plt.title('N = 10')
  carteira = random.sample(list(dataset.columns) , k=10)
  carteira = 100 * retorno_acumulado.loc[: , carteira]
  carteira['saldo'] = carteira.sum(axis=1)
  carteira['saldo'].plot(figsize=(18,8))
  erro = ((carteira['saldo'] - ibov)**(2)).mean()
  N_10.append(erro)
```

```
  saldos_finais.append(carteira['saldo'].iloc[-1])

(ibov*1000).plot(linewidth=4, color='black')
```

[21]: <AxesSubplot:title={'center':'N = 10'}, xlabel='Date'>



[22]:
```
bateu_ibov = 0
saldo_final_ibov = ibov.iloc[-1]*1000
for saldo in saldos_finais:
    if (saldo > saldo_final_ibov):
        bateu_ibov += 1
bateu_ibov_porcento = round((bateu_ibov/1000)*100,2)
print("{} de 1000 carteiras ({}%) bateram o IBOV no período".format(bateu_ibov,␣
 ↪bateu_ibov_porcento))
```

```
600 de 1000 carteiras (60.0%) bateram o IBOV no período
```

3 Resumo

Para N = 5:

495 de 1000 carteiras (54%) bateram o IBOV no período

Para N = 6:

548 de 1000 carteiras (53.3%) bateram o IBOV no período

Para N = 7:

554 de 1000 carteiras (54.7%) bateram o IBOV no período

Para N = 8:

543 de 1000 carteiras (55.7%) bateram o IBOV no período

Para N = 9:

547 de 1000 carteiras (56.9%) bateram o IBOV no período

Para N = 10:

560 de 1000 carteiras (60%) bateram o IBOV no período

Diante dessa análise percebemos que em razão do incremento de porcentagem ganho em carteiras batendo o ibov por N nos sugere que o N que peformou melhor foi o N = 10

4 Análise do erro médio

[23]:
```
erro_medio_5 = sum(N_5)/len(N_5)
print(erro_medio_5)
```

```
829020.6616456396
```

[24]:
```
erro_medio_6 = sum(N_6)/len(N_6)
print(erro_medio_6)
```

```
822979.2645035173
```

[25]:
```
erro_medio_7 = sum(N_7)/len(N_7)
print(erro_medio_7)
```

```
821625.0402030945
```

[26]:
```
erro_medio_8 = sum(N_8)/len(N_8)
print(erro_medio_8)
```

```
817073.4535316783
```

[27]:
```
erro_medio_9 = sum(N_9)/len(N_9)
print(erro_medio_9)
```

```
817863.4237493257
```

[28]:
```
erro_medio_10 = sum(N_10)/len(N_10)
print(erro_medio_10)
```

```
818728.1788034993
```

Pela Análise do erro médio em relação aos retornos do ibov, o N que apresentou menor erro médio foi N = 8.

Com isso concluímos que, com N = 10 obtemos o maior percentual de carteiras que podem bater o ibov enquanto ao usarmos N = 8 obteremos um maior percentual de carteiras que acompanharam o ibov de perto.