acertos

June 30, 2024

[ ]:
```
import numpy as np
import pandas as pd
```

[ ]:
```
data1 = pd.read_csv('/home/darkcover/Documentos/Out/dados/data_final1.csv')
data2 = pd.read_csv('/home/darkcover/Documentos/Out/dados/data_final2.csv')
```

[ ]:
```
data1 = data1.drop(columns=['Unnamed: 0'])
# Excluir a linha com índice 0
data1 = data1.drop(0).reset_index(drop=True)

data1.head()
```

[ ]:

	Rodada	level	apostar	acerto	contagem	odd	odd_entrada	odd_saida	\
0	1.0	1.0	0.0	1.0	0.0	3.85	11.0	9.0	
1	2.0	1.0	0.0	1.0	0.0	6.96	9.0	10.0	
2	3.0	1.0	0.0	1.0	0.0	5.41	10.0	10.0	
3	4.0	1.0	0.0	0.0	0.0	1.05	10.0	2.0	
4	5.0	1.0	0.0	1.0	0.0	1.70	2.0	6.0	

	media80	desvpad80geral	…	media320	desvpad320geral	percentil320geral	\
0	0.0	0.0	…	0.0	0.0	0.0	
1	0.0	0.0	…	0.0	0.0	0.0	
2	0.0	0.0	…	0.0	0.0	0.0	
3	0.0	0.0	…	0.0	0.0	0.0	
4	0.0	0.0	…	0.0	0.0	0.0	

	cv320	roc320	media640	desvpad640geral	percentil640geral	cv640	roc640
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[5 rows x 28 columns]

[ ]:
```
data2 = data2.drop(columns=['Unnamed: 0'])
```

```
# Excluir a linha com índice 0
data2 = data2.drop(0).reset_index(drop=True)

data2.head()
```

[ ]:

	Rodada	level	apostar	acerto	contagem	odd	odd_entrada	odd_saida	\
0	1.0	1.0	0.0	0.0	0.0	1.41	1.0	4.0	
1	2.0	1.0	0.0	1.0	0.0	3.10	4.0	8.0	
2	3.0	1.0	0.0	0.0	0.0	1.22	8.0	3.0	
3	4.0	1.0	0.0	1.0	0.0	8.19	3.0	10.0	
4	5.0	1.0	0.0	1.0	0.0	62.14	10.0	11.0	

	media80	desvpad80geral	…	media320	desvpad320geral	percentil320geral	\
0	0.0	0.0	…	0.0	0.0	0.0	
1	0.0	0.0	…	0.0	0.0	0.0	
2	0.0	0.0	…	0.0	0.0	0.0	
3	0.0	0.0	…	0.0	0.0	0.0	
4	0.0	0.0	…	0.0	0.0	0.0	

	cv320	roc320	media640	desvpad640geral	percentil640geral	cv640	roc640
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[5 rows x 28 columns]

```
data1.describe()
```

[ ]:

	Rodada	level	apostar	acerto	\
count	199999.000000	199999.000000	199999.000000	199999.000000	
mean	100000.000000	2.398112	0.035475	0.681593	
std	57734.882581	1.741183	0.184978	0.465859	
min	1.000000	1.000000	0.000000	0.000000	
25%	50000.500000	1.000000	0.000000	0.000000	
50%	100000.000000	2.000000	0.000000	1.000000	
75%	149999.500000	3.000000	0.000000	1.000000	
max	199999.000000	7.000000	1.000000	1.000000	

	contagem	odd	odd_entrada	odd_saida	\
count	199999.000000	199999.000000	199999.000000	199999.000000	
mean	3.672943	16.353780	6.244266	6.244251	
std	5.420655	1964.967513	3.039212	3.039196	
min	-9.000000	0.000000	1.000000	1.000000	
25%	0.000000	1.310000	4.000000	4.000000	
50%	3.000000	1.960000	6.000000	6.000000	

```
75%         8.000000       3.900000       9.000000       9.000000
max        14.000000  837137.310000      11.000000      11.000000

             media80  desvpad80geral  …       media320  desvpad320geral  \
count  199999.000000   199999.000000  …  199999.000000    199999.000000
mean        6.241609        3.014911  …       6.233853         3.028491
std         0.361126        0.168839  …       0.300835         0.145860
min         0.000000        0.000000  …       0.000000         0.000000
25%         6.012500        2.910514  …       6.131250         2.977308
50%         6.237500        3.019494  …       6.240625         3.032698
75%         6.475000        3.125275  …       6.356250         3.090914
max         7.512500        3.591635  …       7.037500         3.294217

       percentil320geral          cv320         roc320       media640  \
count      199999.000000  199999.000000  199998.000000  199999.000000
mean            6.087675       0.485404       0.063773       6.223676
std             0.440125       0.027058       4.625663       0.372428
min             0.000000       0.000000     -33.987069       0.000000
25%             6.115625       0.473949      -3.123400       6.162500
50%             6.128125       0.486567      -0.049727       6.240625
75%             6.131250       0.498524       3.185379       6.320312
max             6.187500       0.572573      19.805720       6.814063

       desvpad640geral  percentil640geral          cv640         roc640
count    199999.000000      199999.000000  199999.000000  199998.000000
mean          3.026359           6.079568       0.484890       0.025905
std           0.181325           0.658617       0.030532       2.798579
min           0.000000           0.000000       0.000000     -33.485991
25%           2.996989           6.145313       0.477551      -1.848443
50%           3.035981           6.157813       0.486520       0.000000
75%           3.077006           6.164062       0.495788       1.909270
max           3.216642           6.168750       0.544474      10.132502

[8 rows x 28 columns]
```

[ ]:
```
data2.describe()
```

[ ]:
```
              Rodada          level        apostar         acerto  \
count  199999.000000  199999.000000  199999.000000  199999.000000
mean   100000.000000       2.620288       0.037190       0.678538
std     57734.882581       1.866546       0.189228       0.467039
min         1.000000       1.000000       0.000000       0.000000
25%     50000.500000       1.000000       0.000000       0.000000
50%    100000.000000       2.000000       0.000000       1.000000
75%    149999.500000       3.000000       0.000000       1.000000
max    199999.000000       8.000000       1.000000       1.000000
```

```
            contagem            odd    odd_entrada      odd_saida  \
count  199999.000000  199999.000000  199999.000000  199999.000000
mean        3.742634      11.155112       6.235671       6.235721
std         5.073139     365.286564       3.057063       3.057059
min        -9.000000       0.000000       1.000000       1.000000
25%         0.000000       1.310000       4.000000       4.000000
50%         4.000000       1.960000       6.000000       6.000000
75%         8.000000       3.930000       9.000000       9.000000
max        14.000000  101085.730000      11.000000      11.000000

             media80  desvpad80geral  …       media320  desvpad320geral  \
count  199999.000000   199999.000000  …  199999.000000    199999.000000
mean        6.232993        3.033121  …       6.225570         3.046421
std         0.359524        0.168926  …       0.301503         0.145159
min         0.000000        0.000000  …       0.000000         0.000000
25%         6.012500        2.928283  …       6.115625         2.999152
50%         6.237500        3.038272  …       6.234375         3.052759
75%         6.462500        3.142849  …       6.353125         3.105533
max         7.637500        3.632987  …       6.928125         3.332129

       percentil320geral          cv320         roc320       media640  \
count      199999.000000  199999.000000  199998.000000  199999.000000
mean            6.076171       0.488912       0.077787       6.215621
std             0.439416       0.026683       4.798179       0.371493
min             0.000000       0.000000     -21.117424       0.000000
25%             6.106250       0.477459      -3.187251       6.154687
50%             6.112500       0.489815       0.000000       6.232812
75%             6.115625       0.501994       3.303303       6.315625
max             6.121875       0.559437      54.218750       6.729687

       desvpad640geral  percentil640geral          cv640         roc640
count    199999.000000      199999.000000  199999.000000  199998.000000
mean          3.044394           6.078313       0.488407       0.041401
std           0.181340           0.658469       0.030557       2.778217
min           0.000000           0.000000       0.000000     -21.882102
25%           3.017339           6.148438       0.481442      -1.780660
50%           3.055233           6.154687       0.490039       0.000000
75%           3.092608           6.157813       0.498577       1.893939
max           3.228049           6.162500       0.533838      53.125000

[8 rows x 28 columns]
```

[ ]:
```
j5, j6, j7, j8 = 0, 0, 0, 0
k5, k6, k7, k8 = 0, 0, 0, 0
l5, l6, l7, l8 = 0, 0, 0, 0
for i in range(len(data1)):
    if data1['media80'][i] <= data1['percentil80geral'][i]:
```

```
        j5 += 1
        if data1['acerto'][i] == 1:
            k5 += 1
        else:
            l5 += 1
    if data1['media160'][i] <= data1['percentil160geral'][i]:
        j6 += 1
        if data1['acerto'][i] == 1:
            k6 += 1
        else:
            l6 += 1
    if data1['media320'][i] <= data1['percentil320geral'][i]:
        j7 += 1
        if data1['acerto'][i] == 1:
            k7 += 1
        else:
            l7 += 1
    if data1['media640'][i] <= data1['percentil640geral'][i]:
        j8 += 1
        if data1['acerto'][i] == 1:
            k8 += 1
        else:
            l8 += 1

print(f'\nQuantidade de elementos em percentil de 25% média80: {j5}␣
 ↪\nQuantidade de acertos: {k5} ({k5 / j5}) \nQuantidade de erros:{l5} ({l5 /␣
 ↪j5}) \nQuantidade de elementos em percentil de 25% média160: {j6}␣
 ↪\nQuantidade de acertos: {k6} ({k6 / j6}) \nQuantidade de erros:{l6} ({l6 /␣
 ↪j6}) \nQuantidade de elementos em percentil de 25% média320: {j7}␣
 ↪\nQuantidade de acertos: {k7} ({k7 / j7}) \nQuantidade de erros:{l7} ({l7 /␣
 ↪j7}) \nQuantidade de elementos em percentil de 25% média640: {j8}␣
 ↪\nQuantidade de acertos: {k8} ({k8 / j8}) \nQuantidade de erros:{l8} ({l8 /␣
 ↪j8})')
```

```
Quantidade de elementos em percentil de 25% média80: 50421
Quantidade de acertos: 34512 (0.6844767061343487)
Quantidade de erros:15909 (0.3155232938656512)
Quantidade de elementos em percentil de 25% média160: 49034
Quantidade de acertos: 33536 (0.6839335970958926)
Quantidade de erros:15498 (0.31606640290410737)
Quantidade de elementos em percentil de 25% média320: 47022
Quantidade de acertos: 32118 (0.6830419803496236)
Quantidade de erros:14904 (0.3169580196503764)
Quantidade de elementos em percentil de 25% média640: 44635
Quantidade de acertos: 30546 (0.6843508457488517)
Quantidade de erros:14089 (0.3156491542511482)
```

[ ]:
```
j5, j6, j7, j8 = 0, 0, 0, 0
k5, k6, k7, k8 = 0, 0, 0, 0
l5, l6, l7, l8 = 0, 0, 0, 0
for i in range(len(data2)):
    if data2['media80'][i] <= data2['percentil80geral'][i]:
        j5 += 1
        if data2['acerto'][i] == 1:
            k5 += 1
        else:
            l5 += 1
    if data2['media160'][i] <= data2['percentil160geral'][i]:
        j6 += 1
        if data2['acerto'][i] == 1:
            k6 += 1
        else:
            l6 += 1
    if data2['media320'][i] <= data2['percentil320geral'][i]:
        j7 += 1
        if data2['acerto'][i] == 1:
            k7 += 1
        else:
            l7 += 1
    if data2['media640'][i] <= data2['percentil640geral'][i]:
        j8 += 1
        if data2['acerto'][i] == 1:
            k8 += 1
        else:
            l8 += 1

print(f'Quantidade de elementos em percentil de 25% média80: {j5} \nQuantidade␣
 ↪de acertos: {k5} ({k5 / j5}) \nQuantidade de erros:{l5} ({l5 / j5})␣
 ↪\nQuantidade de elementos em percentil de 25% média160: {j6} \nQuantidade de␣
 ↪acertos: {k6} ({k6 / j6}) \nQuantidade de erros:{l6} ({l6 / j6})␣
 ↪\nQuantidade de elementos em percentil de 25% média320: {j7} \nQuantidade de␣
 ↪acertos: {k7} ({k7 / j7}) \nQuantidade de erros:{l7} ({l7 / j7})␣
 ↪\nQuantidade de elementos em percentil de 25% média640: {j8} \nQuantidade de␣
 ↪acertos: {k8} ({k8 / j8}) \nQuantidade de erros:{l8} ({l8 / j8})')
```

```
Quantidade de elementos em percentil de 25% média80: 50225
Quantidade de acertos: 34158 (0.6800995520159283)
Quantidade de erros:16067 (0.3199004479840717)
Quantidade de elementos em percentil de 25% média160: 49707
Quantidade de acertos: 33809 (0.6801657714205243)
Quantidade de erros:15898 (0.31983422857947574)
Quantidade de elementos em percentil de 25% média320: 47657
Quantidade de acertos: 32345 (0.6787040728539354)
Quantidade de erros:15312 (0.3212959271460646)
```

```
Quantidade de elementos em percentil de 25% média640: 47847
Quantidade de acertos: 32583 (0.6809831337387924)
Quantidade de erros:15264 (0.3190168662612076)
```

[ ]:
```
j5, j6, j7, j8 = 0, 0, 0, 0
k5, k6, k7, k8 = 0, 0, 0, 0
l5, l6, l7, l8 = 0, 0, 0, 0
for i in range(len(data1)):
    if data1['media80'][i] <= data1['percentil80geral'][i] and␣
 ↪data1['roc80'][i] >= -4:
        j5 += 1
        if data1['acerto'][i] == 1:
            k5 += 1
        else:
            l5 += 1
    if data1['media160'][i] <= data1['percentil160geral'][i] and␣
 ↪data1['roc160'][i] >= -4:
        j6 += 1
        if data1['acerto'][i] == 1:
            k6 += 1
        else:
            l6 += 1
    if data1['media320'][i] <= data1['percentil320geral'][i] and␣
 ↪data1['roc320'][i] >= -4:
        j7 += 1
        if data1['acerto'][i] == 1:
            k7 += 1
        else:
            l7 += 1
    if data1['media640'][i] <= data1['percentil640geral'][i] and␣
 ↪data1['roc640'][i] >= -4:
        j8 += 1
        if data1['acerto'][i] == 1:
            k8 += 1
        else:
            l8 += 1

print(f'\nQuantidade de elementos em percentil de 25% média80: {j5}␣
 ↪\nQuantidade de acertos: {k5} ({k5 / j5}) \nQuantidade de erros:{l5} ({l5 /␣
 ↪j5}) \nQuantidade de elementos em percentil de 25% média160: {j6}␣
 ↪\nQuantidade de acertos: {k6} ({k6 / j6}) \nQuantidade de erros:{l6} ({l6 /␣
 ↪j6}) \nQuantidade de elementos em percentil de 25% média320: {j7}␣
 ↪\nQuantidade de acertos: {k7} ({k7 / j7}) \nQuantidade de erros:{l7} ({l7 /␣
 ↪j7}) \nQuantidade de elementos em percentil de 25% média640: {j8}␣
 ↪\nQuantidade de acertos: {k8} ({k8 / j8}) \nQuantidade de erros:{l8} ({l8 /␣
 ↪j8})')
```

```
Quantidade de elementos em percentil de 25% média80: 15664
Quantidade de acertos: 10743 (0.6858401430030644)
Quantidade de erros:4921 (0.31415985699693566)
Quantidade de elementos em percentil de 25% média160: 20482
Quantidade de acertos: 14008 (0.6839175861732253)
Quantidade de erros:6474 (0.3160824138267747)
Quantidade de elementos em percentil de 25% média320: 26126
Quantidade de acertos: 17791 (0.680969149506239)
Quantidade de erros:8335 (0.319030850493761)
Quantidade de elementos em percentil de 25% média640: 33868
Quantidade de acertos: 23250 (0.6864887209164994)
Quantidade de erros:10618 (0.3135112790835006)
```

[ ]:
```
j5, j6, j7, j8 = 0, 0, 0, 0
k5, k6, k7, k8 = 0, 0, 0, 0
l5, l6, l7, l8 = 0, 0, 0, 0
for i in range(len(data2)):
    if data2['media80'][i] <= data2['percentil80geral'][i] and␣
 ↪data2['roc80'][i] >= -4:
        j5 += 1
        if data2['acerto'][i] == 1:
            k5 += 1
        else:
            l5 += 1
    if data2['media160'][i] <= data2['percentil160geral'][i] and␣
 ↪data2['roc160'][i] >= -4:
        j6 += 1
        if data2['acerto'][i] == 1:
            k6 += 1
        else:
            l6 += 1
    if data2['media320'][i] <= data2['percentil320geral'][i] and␣
 ↪data2['roc320'][i] >= -4:
        j7 += 1
        if data2['acerto'][i] == 1:
            k7 += 1
        else:
            l7 += 1
    if data2['media640'][i] <= data2['percentil640geral'][i] and␣
 ↪data2['roc640'][i] >= -4:
        j8 += 1
        if data2['acerto'][i] == 1:
            k8 += 1
        else:
            l8 += 1
```

```
print(f'Quantidade de elementos em percentil de 25% média80: {j5} \nQuantidade␣
 ↪de acertos: {k5} ({k5 / j5}) \nQuantidade de erros:{l5} ({l5 / j5})␣
 ↪\nQuantidade de elementos em percentil de 25% média160: {j6} \nQuantidade de␣
 ↪acertos: {k6} ({k6 / j6}) \nQuantidade de erros:{l6} ({l6 / j6})␣
 ↪\nQuantidade de elementos em percentil de 25% média320: {j7} \nQuantidade de␣
 ↪acertos: {k7} ({k7 / j7}) \nQuantidade de erros:{l7} ({l7 / j7})␣
 ↪\nQuantidade de elementos em percentil de 25% média640: {j8} \nQuantidade de␣
 ↪acertos: {k8} ({k8 / j8}) \nQuantidade de erros:{l8} ({l8 / j8})')
```

```
Quantidade de elementos em percentil de 25% média80: 15595
Quantidade de acertos: 10594 (0.6793202949663354)
Quantidade de erros:5001 (0.32067970503366466)
Quantidade de elementos em percentil de 25% média160: 20890
Quantidade de acertos: 14196 (0.6795595978937291)
Quantidade de erros:6694 (0.32044040210627095)
Quantidade de elementos em percentil de 25% média320: 25581
Quantidade de acertos: 17433 (0.6814823501817755)
Quantidade de erros:8148 (0.31851764981822445)
Quantidade de elementos em percentil de 25% média640: 36472
Quantidade de acertos: 24890 (0.6824413248519412)
Quantidade de erros:11582 (0.3175586751480588)
```

[ ]:
```
j5, j6, j7, j8 = 0, 0, 0, 0
k5, k6, k7, k8 = 0, 0, 0, 0
l5, l6, l7, l8 = 0, 0, 0, 0
for i in range(len(data1)):
    if data1['roc80'][i] >= 0:
        j5 += 1
        if data1['acerto'][i] == 1:
            k5 += 1
        else:
            l5 += 1
    if data1['roc160'][i] >= 0:
        j6 += 1
        if data1['acerto'][i] == 1:
            k6 += 1
        else:
            l6 += 1
    if data1['roc320'][i] >= 0:
        j7 += 1
        if data1['acerto'][i] == 1:
            k7 += 1
        else:
            l7 += 1
    if data1['roc640'][i] >= 0:
        j8 += 1
        if data1['acerto'][i] == 1:
```

```
            k8 += 1
        else:
            l8 += 1

print(f'\nQuantidade de elementos em percentil de 25% média80: {j5}␣
 ↪\nQuantidade de acertos: {k5} ({k5 / j5}) \nQuantidade de erros:{l5} ({l5 /␣
 ↪j5}) \nQuantidade de elementos em percentil de 25% média160: {j6}␣
 ↪\nQuantidade de acertos: {k6} ({k6 / j6}) \nQuantidade de erros:{l6} ({l6 /␣
 ↪j6}) \nQuantidade de elementos em percentil de 25% média320: {j7}␣
 ↪\nQuantidade de acertos: {k7} ({k7 / j7}) \nQuantidade de erros:{l7} ({l7 /␣
 ↪j7}) \nQuantidade de elementos em percentil de 25% média640: {j8}␣
 ↪\nQuantidade de acertos: {k8} ({k8 / j8}) \nQuantidade de erros:{l8} ({l8 /␣
 ↪j8})')
```

```
Quantidade de elementos em percentil de 25% média80: 101006
Quantidade de acertos: 68943 (0.6825634120745302)
Quantidade de erros:32063 (0.31743658792546975)
Quantidade de elementos em percentil de 25% média160: 100679
Quantidade de acertos: 68541 (0.6807874531928207)
Quantidade de erros:32138 (0.31921254680717925)
Quantidade de elementos em percentil de 25% média320: 99691
Quantidade de acertos: 67802 (0.6801215756688166)
Quantidade de erros:31889 (0.31987842433118335)
Quantidade de elementos em percentil de 25% média640: 101121
Quantidade de acertos: 68880 (0.681164149879847)
Quantidade de erros:32241 (0.3188358501201531)
```