Loading the data:

```
import pandas as pd

df = pd.read_csv("results.csv", index_col='id_alg')
df.head()
```

```
        param_1 param_2  param_3  id_dataset  param_4  mean_ind
std_ind  \
id_alg

1            -1       0        5           1       60      0.92
0.074833
2            -1       1        5           1       60      0.93
0.064031
3            -1       0       10           1       60      0.93
0.064031
4            -1       1       10           1       60      0.94
0.066332
5             5       0        0           1       60      0.91
0.094340

        ind_0  ind_1  ind_2  ind_3  ind_4  ind_5  ind_6  ind_7  ind_8
ind_9
id_alg

1         0.9    0.9    0.9    1.0    1.0    0.8    1.0    0.9    1.0
0.8
2         0.9    0.9    1.0    1.0    1.0    0.8    0.9    0.9    1.0
0.9
3         0.9    0.9    0.9    1.0    1.0    0.8    1.0    0.9    1.0
0.9
4         0.9    0.9    1.0    1.0    1.0    0.8    1.0    0.9    1.0
0.9
5         1.0    0.9    0.9    1.0    1.0    0.8    0.9    0.9    1.0
0.7
```

```
df.sample(3)
```

```
        param_1 param_2  param_3  id_dataset  param_4  mean_ind
std_ind  \
id_alg

OBLQ_1       -1       -        5           7        4  0.730833
0.079499
4            -1       1       10           6        3  1.000000
0.000000
7             5       1        0           7        2  0.800833
0.082281
```

```
           ind_0     ind_1  ind_2     ind_3     ind_4  ind_5     ind_6
\
id_alg

OBLQ_1  0.700000  0.791667  0.750  0.641667  0.641667   0.85  0.741667

4       1.000000  1.000000  1.000  1.000000  1.000000   1.00  1.000000

7       0.808333  0.683333  0.925  0.791667  0.775000   0.85  0.800000


           ind_7     ind_8     ind_9
id_alg
OBLQ_1  0.625000  0.708333  0.858333
4       1.000000  1.000000  1.000000
7       0.891667  0.841667  0.641667
```

Let's analise the data:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 560 entries, 1 to OBLQ_2
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   param_1     560 non-null    int64
 1   param_2     560 non-null    object
 2   param_3     560 non-null    int64
 3   id_dataset  560 non-null    int64
 4   param_4     560 non-null    int64
 5   mean_ind    560 non-null    float64
 6   std_ind     560 non-null    float64
 7   ind_0       560 non-null    float64
 8   ind_1       560 non-null    float64
 9   ind_2       560 non-null    float64
 10  ind_3       560 non-null    float64
 11  ind_4       560 non-null    float64
 12  ind_5       560 non-null    float64
 13  ind_6       560 non-null    float64
 14  ind_7       560 non-null    float64
 15  ind_8       560 non-null    float64
 16  ind_9       560 non-null    float64
dtypes: float64(12), int64(4), object(1)
memory usage: 78.8+ KB

df.isnull().sum()

param_1       0
param_2       0
```

```
param_3       0
id_dataset    0
param_4       0
mean_ind      0
std_ind       0
ind_0         0
ind_1         0
ind_2         0
ind_3         0
ind_4         0
ind_5         0
ind_6         0
ind_7         0
ind_8         0
ind_9         0
dtype: int64
```

```
df.describe()
```

```
          param_1     param_3  id_dataset     param_4    mean_ind
std_ind  \
count  560.000000  560.000000  560.000000  560.000000  560.000000
560.000000
mean     3.857143    4.642857    4.500000   14.000000    0.824046
0.059808
std      4.615555    3.520594    2.293336   23.042312    0.140952
0.040652
min     -1.000000    0.000000    1.000000    1.000000    0.478274
0.000000
25%     -1.000000    0.000000    2.750000    2.000000    0.706756
0.026300
50%      5.000000    5.000000    4.500000    3.000000    0.838750
0.060000
75%     10.000000    5.000000    6.250000    4.000000    0.954507
0.086318
max     10.000000   10.000000    8.000000   60.000000    1.000000
0.163918

             ind_0       ind_1       ind_2       ind_3       ind_4
ind_5  \
count  560.000000  560.000000  560.000000  560.000000  560.000000
560.000000
mean     0.825218    0.820586    0.809152    0.817594    0.826545
0.834645
std      0.155786    0.163345    0.167078    0.168294    0.172892
0.135144
min      0.321429    0.267857    0.267857    0.285714    0.255102
0.428571
25%      0.706296    0.706296    0.709091    0.701667    0.717778
0.736296
```

```
50%      0.841402    0.850725    0.808712    0.837500    0.863636
0.839161
75%      0.988211    0.985310    0.973307    0.986842    0.986842
0.972471
max      1.000000    1.000000    1.000000    1.000000    1.000000
1.000000
```

```
            ind_6       ind_7       ind_8       ind_9
count  560.000000  560.000000  560.000000  560.000000
mean     0.825988    0.837126    0.823115    0.820490
std      0.159632    0.154013    0.142177    0.160281
min      0.354167    0.229167    0.437500    0.291667
25%      0.709091    0.708194    0.708333    0.699808
50%      0.852273    0.866259    0.823427    0.850000
75%      0.991803    0.995066    0.978645    0.986842
max      1.000000    1.000000    1.000000    1.000000
```

```
df.groupby("id_alg")['mean_ind'].count()
```

```
id_alg
1          40
10         40
11         40
12         40
2          40
3          40
4          40
5          40
6          40
7          40
8          40
9          40
OBLQ_1     40
OBLQ_2     40
Name: mean_ind, dtype: int64
```

As we see above the data is evenly distributed, every algorithm has 40 inputs

Let's agradate every algorithm and see how they preforme on avg

```
model_performance = df.groupby("id_alg")["mean_ind"].agg(["mean",
"std", "max", "min"])
model_performance = model_performance.sort_values(by="mean",
ascending=False)

model_performance
```

```
             mean       std  max       min
id_alg
10       0.842060  0.127637  1.0  0.586310
12       0.840358  0.134626  1.0  0.531845
8        0.836382  0.139865  1.0  0.565476
6        0.834642  0.139715  1.0  0.564286
5        0.830560  0.147111  1.0  0.501190
9        0.828987  0.135866  1.0  0.566369
11       0.828888  0.139180  1.0  0.555655
7        0.825373  0.148319  1.0  0.522619
2        0.818678  0.138843  1.0  0.533036
1        0.815138  0.136369  1.0  0.580952
4        0.811240  0.151226  1.0  0.482738
OBLQ_1   0.809318  0.147217  1.0  0.553550
OBLQ_2   0.809318  0.147217  1.0  0.553550
3        0.805701  0.153850  1.0  0.478274
```

```
model_performance = model_performance.sort_values(by="std",
ascending=True)
print(model_performance)
```

```
             mean       std  max       min
id_alg
10       0.842060  0.127637  1.0  0.586310
12       0.840358  0.134626  1.0  0.531845
9        0.828987  0.135866  1.0  0.566369
1        0.815138  0.136369  1.0  0.580952
2        0.818678  0.138843  1.0  0.533036
11       0.828888  0.139180  1.0  0.555655
6        0.834642  0.139715  1.0  0.564286
8        0.836382  0.139865  1.0  0.565476
5        0.830560  0.147111  1.0  0.501190
OBLQ_1   0.809318  0.147217  1.0  0.553550
OBLQ_2   0.809318  0.147217  1.0  0.553550
7        0.825373  0.148319  1.0  0.522619
4        0.811240  0.151226  1.0  0.482738
3        0.805701  0.153850  1.0  0.478274
```

It seeams taht algo 10 has the highest mean score while also having the lowest standard deviation, this indicates that it might be the best algo.

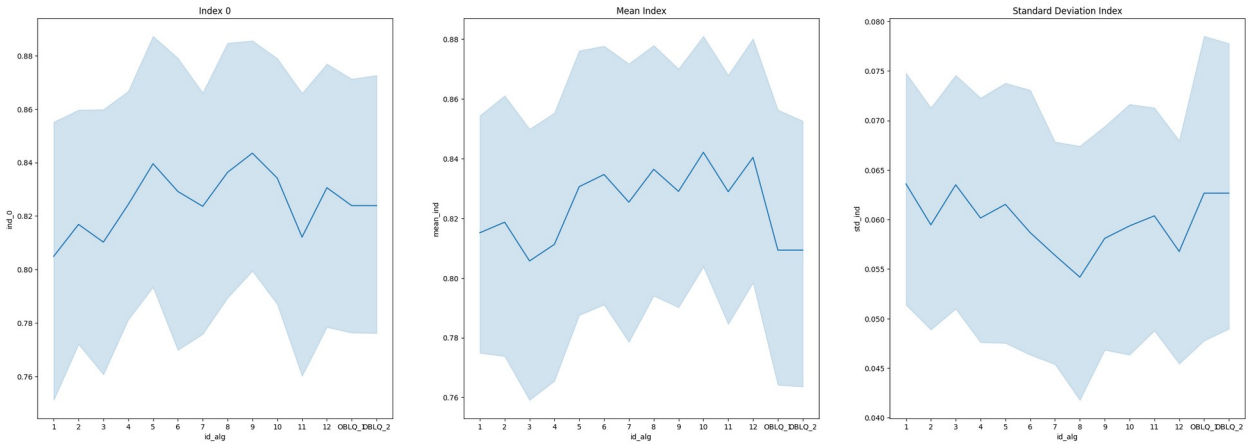
```
df.groupby("id_alg")["mean_ind"].describe()
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

1        40.0  0.815138  0.136369  0.580952  0.705882  0.817298
0.947007  1.0
10       40.0  0.842060  0.127637  0.586310  0.755058  0.868350
```

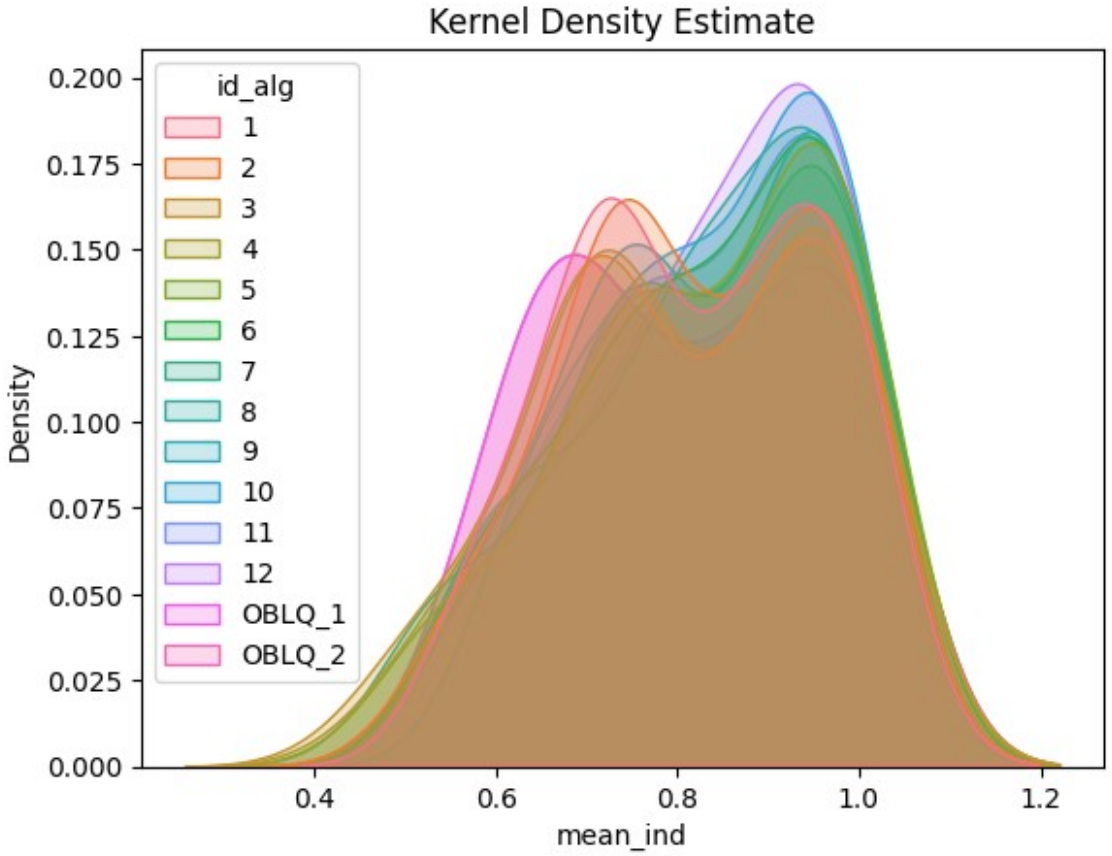
```
0.942826  1.0
11       40.0  0.828888  0.139180  0.555655  0.734432  0.851617
0.941845  1.0
12       40.0  0.840358  0.134626  0.531845  0.759883  0.871667
0.935679  1.0
2        40.0  0.818678  0.138843  0.533036  0.729569  0.814123
0.949837  1.0
3        40.0  0.805701  0.153850  0.478274  0.693838  0.806307
0.957743  1.0
4        40.0  0.811240  0.151226  0.482738  0.717869  0.811466
0.950696  1.0
5        40.0  0.830560  0.147111  0.501190  0.733436  0.862240
0.959427  1.0
6        40.0  0.834642  0.139715  0.564286  0.744681  0.862266
0.952333  1.0
7        40.0  0.825373  0.148319  0.522619  0.732721  0.843566
0.952337  1.0
8        40.0  0.836382  0.139865  0.565476  0.749860  0.862151
0.945040  1.0
9        40.0  0.828987  0.135866  0.566369  0.726525  0.841334
0.949179  1.0
OBLQ_1   40.0  0.809318  0.147217  0.553550  0.678261  0.810430
0.946308  1.0
OBLQ_2   40.0  0.809318  0.147217  0.553550  0.678261  0.810430
0.946308  1.0
```

Let's try to also visualise the data now

```
import matplotlib.pyplot as plt
import seaborn as sns

fig, axes = plt.subplots(1, 3, figsize=(30, 10))

sns.lineplot(ax=axes[0], x=df.index, y=df["ind_0"])
axes[0].set_title("Index 0")

sns.lineplot(ax=axes[1], x=df.index, y=df["mean_ind"])
axes[1].set_title("Mean Index")

sns.lineplot(ax=axes[2], x=df.index, y=df["std_ind"])
axes[2].set_title("Standard Deviation Index")

fig.suptitle("Analysis of Model Performance", fontsize=30,
fontweight='bold')
plt.show()
```

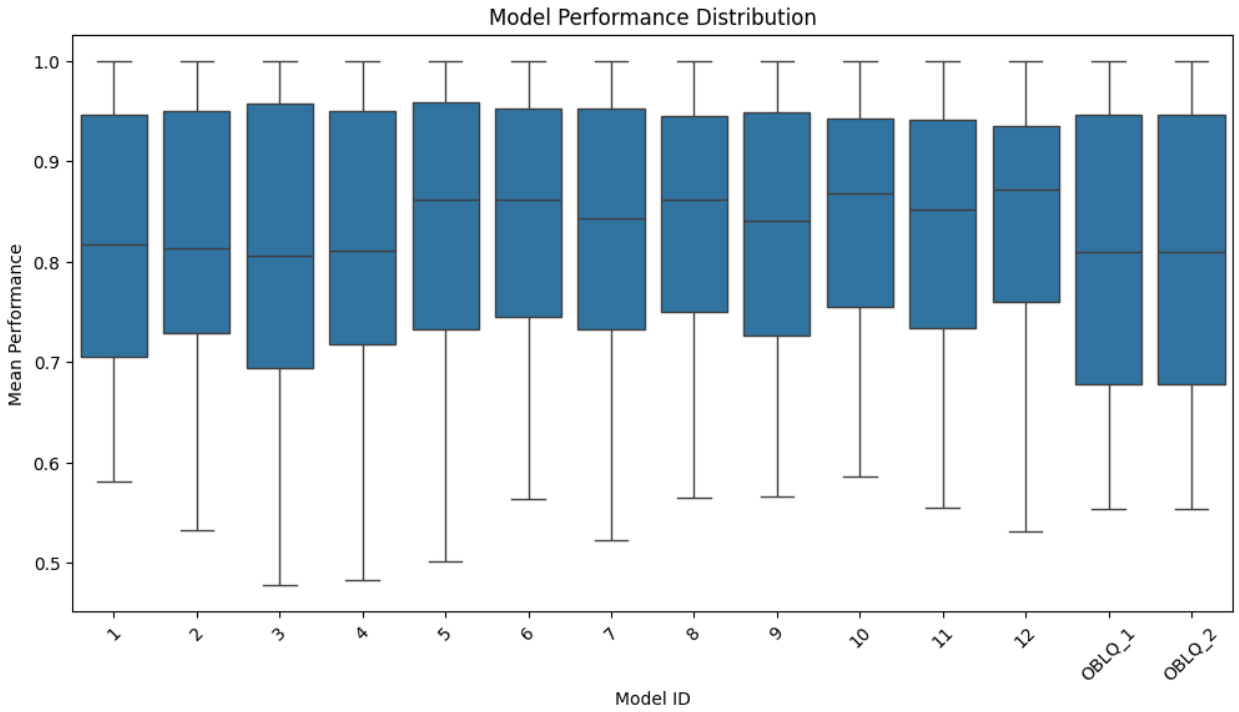



```
sns.kdeplot(data=df, x='mean_ind', hue='id_alg',
fill=True).set_title("Kernel Density Estimate")
plt.show()
```




```
plt.figure(figsize=(12, 6))
sns.boxplot(x="id_alg", y="mean_ind", data=df)
plt.xlabel("Model ID")
```

```
plt.ylabel("Mean Performance")
plt.title("Model Performance Distribution")
plt.xticks(rotation=45)
plt.show()
```



Let's see witch algo is the best for every individual metric:

```
df.groupby("id_alg")["ind_1"].describe().sort_values(by="mean",
ascending=False).head(3)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

8        40.0  0.844586  0.155832  0.464286  0.715278  0.867045
0.991803  1.0
10       40.0  0.841861  0.150500  0.392857  0.727748  0.864773
0.980440  1.0
5        40.0  0.835643  0.157641  0.383929  0.713636  0.865761
0.993506  1.0
```

```
df.groupby("id_alg")["ind_2"].describe().sort_values(by="mean",
ascending=False).head(3)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg
```

```
12       40.0  0.829001  0.142481  0.517857  0.735694  0.833704
0.963007  1.0
10       40.0  0.823770  0.144349  0.517857  0.712500  0.834964
0.967941  1.0
6        40.0  0.821465  0.166794  0.464286  0.706667  0.874094
0.980290  1.0
```

```
df.groupby("id_alg")["ind_3"].describe().sort_values(by="mean",
ascending=False).head(3)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

10       40.0  0.836584  0.161053  0.428571  0.731742  0.879167
0.988918  1.0
12       40.0  0.831936  0.165982  0.428571  0.709091  0.879167
0.986869  1.0
9        40.0  0.831571  0.155273  0.428571  0.716071  0.887500
0.965514  1.0
```

```
df.groupby("id_alg")["ind_4"].describe().sort_values(by="mean",
ascending=False).head(3)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

10       40.0  0.853260  0.142408  0.500000  0.753611  0.884964
0.988487  1.0
12       40.0  0.852121  0.161810  0.428571  0.753392  0.900000
0.990132  1.0
6        40.0  0.849859  0.157657  0.428571  0.764583  0.897464
0.990132  1.0
```

```
df.groupby("id_alg")["ind_5"].describe().sort_values(by="mean",
ascending=False).head(3)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

5        40.0  0.859240  0.118467  0.571429  0.790093  0.856818
0.968831  1.0
12       40.0  0.846611  0.127362  0.571429  0.747963  0.879808
0.964170  1.0
10       40.0  0.845794  0.124023  0.568182  0.764495  0.806818
0.975814  1.0
```

```
df.groupby("id_alg")["ind_6"].describe().sort_values(by="mean",
ascending=False).head(3)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

6        40.0  0.849084  0.162052  0.395833  0.726690  0.895833
1.000000  1.0
10       40.0  0.841602  0.151658  0.541667  0.710227  0.887500
0.986060  1.0
12       40.0  0.836797  0.153441  0.500000  0.706061  0.883974
0.988082  1.0
```

```
df.groupby("id_alg")["ind_7"].describe().sort_values(by="mean",
ascending=False).head(4)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

12       40.0  0.864294  0.135085  0.541667  0.765972  0.888811
0.993421  1.0
8        40.0  0.855460  0.141148  0.583333  0.753611  0.900000
1.000000  1.0
9        40.0  0.851781  0.135188  0.541667  0.753333  0.851777
0.988487  1.0
10       40.0  0.847975  0.158103  0.479167  0.706250  0.901564
0.993421  1.0
```

```
df.groupby("id_alg")["ind_8"].describe().sort_values(by="mean",
ascending=False).head(3)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

10       40.0  0.850140  0.129482  0.541667  0.736154  0.865579
0.976219  1.0
12       40.0  0.845859  0.134632  0.458333  0.759615  0.883974
0.977028  1.0
11       40.0  0.843937  0.118750  0.541667  0.752525  0.846154
0.967564  1.0
```

```
df.groupby("id_alg")["ind_9"].describe().sort_values(by="mean",
ascending=False).head(5)
```

```
        count      mean       std       min       25%       50%
75%  max
id_alg

10       40.0  0.845433  0.152475  0.333333  0.724936  0.860111
0.988487  1.0
8        40.0  0.840062  0.145364  0.479167  0.747727  0.858586
```

```
0.981557  1.0
12       40.0  0.839014  0.141028  0.458333  0.714583  0.854215
0.986842  1.0
6        40.0  0.837082  0.146068  0.500000  0.699423  0.856250
0.992208  1.0
5        40.0  0.831280  0.164435  0.312500  0.685433  0.872611
1.000000  1.0
```

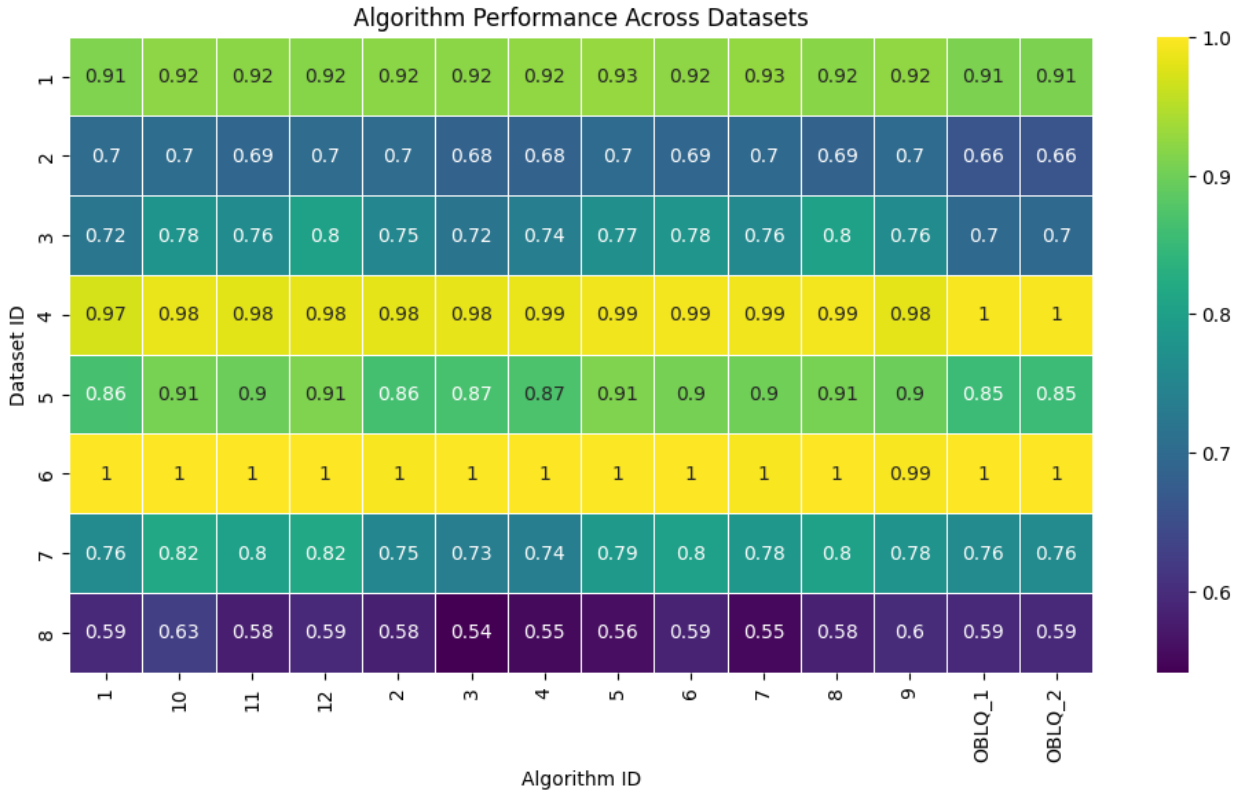
As we can see, for all individual metrics algo 10 is at least in top 3 for most of them!

Let's now go over each dataset and see how every algo does on them:

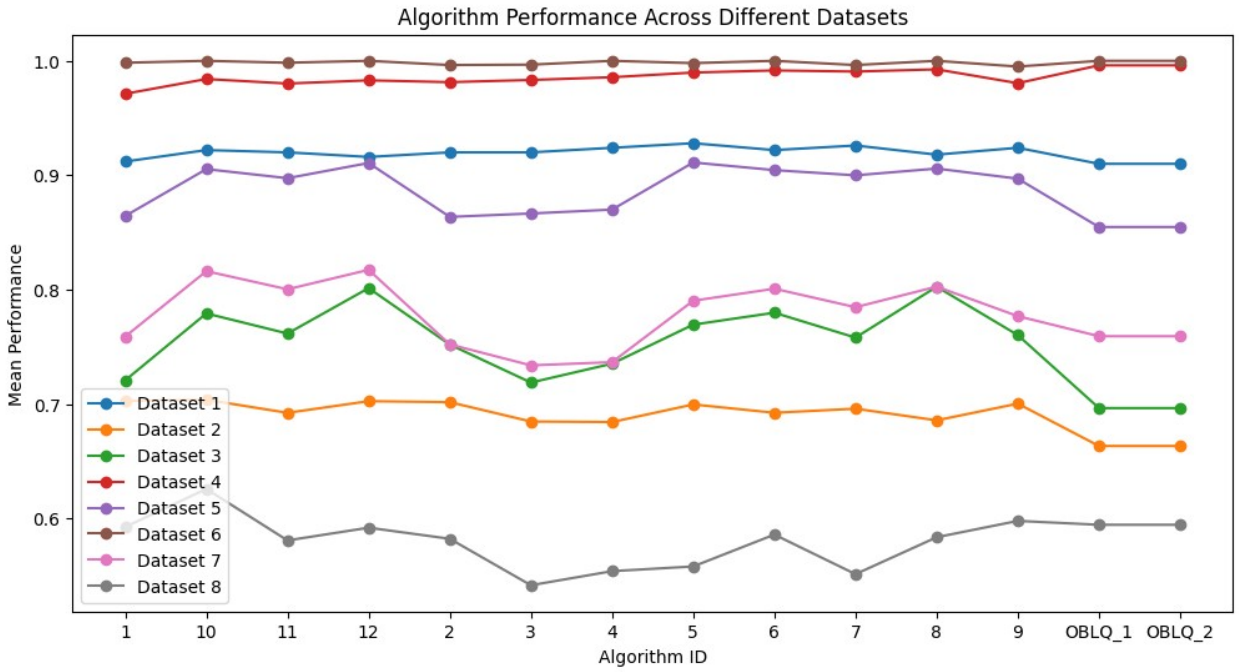
```
algo_performance = df.groupby(["id_dataset", "id_alg"])
["mean_ind"].agg(["mean", "std"])
best_algos =
algo_performance.loc[algo_performance.groupby("id_dataset")
["mean"].idxmax()]
print(best_algos)
```

```
                        mean       std
id_dataset id_alg
1          5       0.928000  0.016432
2          10      0.703789  0.007890
3          8       0.802566  0.023875
4          OBLQ_1  0.996061  0.000747
5          5       0.911168  0.007335
6          10      1.000000  0.000000
7          12      0.817333  0.018795
8          10      0.625833  0.026760
```

```
algo_performance = algo_performance.reset_index()
heatmap_data = algo_performance.pivot(index="id_dataset",
columns="id_alg", values="mean")

plt.figure(figsize=(12, 6))
sns.heatmap(heatmap_data, annot=True, cmap="viridis", linewidths=0.5)
plt.title("Algorithm Performance Across Datasets")
plt.xlabel("Algorithm ID")
plt.ylabel("Dataset ID")
plt.show()
```




```
plt.figure(figsize=(12, 6))

for dataset_id in df["id_dataset"].unique():
    subset = algo_performance[algo_performance["id_dataset"] ==
dataset_id]
    plt.plot(subset["id_alg"], subset["mean"], marker="o",
label=f"Dataset {dataset_id}")

plt.xlabel("Algorithm ID")
plt.ylabel("Mean Performance")
plt.title("Algorithm Performance Across Different Datasets")
plt.legend()
plt.show()
```



We can see in the plot above that algo 10 is the highest pick or one of the higest in most datasets.