SDV Seminar 1

In [2]:

```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
```

Loading the data

In [3]:

```
results = pd.read_csv('/kaggle/input/sdv-seminar1/results.csv')
```

In [4]:

```
results.sample(15)
```

Out[4]:

	id_alg	param_1	param_2	param_3	id_dataset	param_4	mean_ind	std_ind	ind_0	ind_1	ind_2	ind_3	ind_4	ind_5	ind_6	ind_7	ind_8	ind_9
408	1	-1	0	5	7	4	0.762500	0.083437	0.733333	0.816667	0.591667	0.858333	0.800000	0.808333	0.841667	0.700000	0.658333	0.816667
62	3	-1	0	10	2	60	0.700758	0.045847	0.664815	0.714815	0.753333	0.707778	0.800370	0.706296	0.640741	0.664815	0.657692	0.696923
488	OBLQ_1	-1	-	5	1	4	0.900000	0.100000	1.000000	1.000000	0.800000	0.900000	1.000000	1.000000	0.700000	0.900000	0.800000	0.900000
365	6	10	0	0	7	60	0.828333	0.079250	0.891667	0.816667	0.816667	0.616667	0.808333	0.916667	0.875000	0.883333	0.816667	0.841667
551	OBLQ_2	-1	-	10	8	60	0.594643	0.156472	0.875000	0.669643	0.473214	0.489796	0.795918	0.642857	0.364583	0.687500	0.500000	0.447917
127	8	10	1	0	3	60	0.795101	0.062818	0.813636	0.859091	0.718182	0.672727	0.854545	0.759091	0.809091	0.777778	0.888889	0.797980
150	7	5	1	0	3	2	0.772071	0.076168	0.663636	0.763636	0.709091	0.663636	0.850000	0.859091	0.709091	0.805556	0.843434	0.853535
215	12	10	1	5	4	2	0.983129	0.012119	0.957633	0.987013	0.991803	0.991803	0.986842	0.972066	0.973684	0.993421	0.977028	1.000000
440	9	5	0	5	8	1	0.632440	0.120602	0.598214	0.607143	0.714286	0.642857	0.500000	0.428571	0.875000	0.750000	0.562500	0.645833
421	2	-1	1	5	8	60	0.566369	0.116612	0.616071	0.589286	0.455357	0.571429	0.642857	0.642857	0.541667	0.770833	0.520833	0.312500
345	10	10	0	5	6	3	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
118	11	5	1	5	2	4	0.692305	0.046145	0.699259	0.707778	0.713333	0.756296	0.642222	0.753333	0.699259	0.679259	0.679231	0.593077
15	4	-1	1	10	1	1	0.920000	0.060000	0.900000	0.900000	0.900000	0.900000	1.000000	0.800000	1.000000	1.000000	0.900000	0.900000
123	4	-1	1	10	3	60	0.724116	0.114692	0.859091	0.809091	0.572727	0.618182	0.904545	0.677273	0.718182	0.597222	0.843434	0.641414
555	OBLQ_2	-1	-	10	8	2	0.634503	0.107378	0.535714	0.642857	0.571429	0.765306	0.500000	0.673469	0.500000	0.687500	0.843750	0.625000

In [5]:

```
len(results)
```

Out[5]:

```
560
```

In [6]:

```
results.describe()
```

Out[6]:

	param_1	param_3	id_dataset	param_4	mean_ind	std_ind	ind_0	ind_1	ind_2	ind_3	ind_4	ind_5	ind_6	ind_7	ind_8	ind_9
count	560.000000	560.000000	560.000000	560.000000	560.000000	560.000000	560.000000	560.000000	560.000000	560.000000	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	0.825218	0.820586	0.809152	0.817594	0.826545	0.834645	0.825988	0.837126	0.823115	0.820490
std	4.615555	3.520594	2.293336	23.042312	0.140952	0.040652	0.155786	0.163345	0.167078	0.168294	0.172892	0.135144	0.159632	0.154013	0.142177	0.160281
min	-1.000000	0.000000	1.000000	1.000000	0.478274	0.000000	0.321429	0.267857	0.267857	0.285714	0.255102	0.428571	0.354167	0.229167	0.437500	0.291667
25%	-1.000000	0.000000	2.750000	2.000000	0.706756	0.026300	0.706296	0.706296	0.709091	0.701667	0.717778	0.736296	0.709091	0.708194	0.708333	0.699808
50%	5.000000	5.000000	4.500000	3.000000	0.838750	0.060000	0.841402	0.850725	0.808712	0.837500	0.863636	0.839161	0.852273	0.866259	0.823427	0.850000
75%	10.000000	5.000000	6.250000	4.000000	0.954507	0.086318	0.988211	0.985310	0.973307	0.986842	0.986842	0.972471	0.991803	0.995066	0.978645	0.986842
max	10.000000	10.000000	8.000000	60.000000	1.000000	0.163918	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

- **id_alg and param_2 are missing from the summary**
- **we must have string/missing data**

Obtain the algorithms and the dataset

In [7]:

```
algo_ids = list(sorted(set(results['id_alg'].values)))
print(algo_ids)
```

```
['1', '10', '11', '12', '2', '3', '4', '5', '6', '7', '8', '9', 'OBLQ_1', 'OBLQ_2']
```

In [8]:

```
dataset_ids = list(sorted(set(results['id_dataset'].values)))
print(dataset_ids)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

Get all experiments performed by one algorithm and one dataset:

In [9]:

```
results[(results.id_alg=='1') & (results.id_dataset==2)]
```

Out[9]:

	id_alg	param_1	param_2	param_3	id_dataset	param_4	mean_ind	std_ind	ind_0	ind_1	ind_2	ind_3	ind_4	ind_5	ind_6	ind_7	ind_8	ind_9
60	1	-1	0	5	2	60	0.701365	0.039514	0.677778	0.714815	0.756296	0.662222	0.761852	0.655185	0.719259	0.667778	0.738462	0.660000
72	1	-1	0	5	2	1	0.704066	0.077575	0.632222	0.724815	0.747407	0.805926	0.670741	0.783333	0.575185	0.743333	0.766923	0.590769

	id_alg	param_1	param_2	param_3	id_dataset	param_4	mean_ind	std_ind	ind_0	ind_1	ind_2	ind_3	ind_4	ind_5	ind_6	ind_7	ind_8	ind_9
84	1	-1	0	5	2	2	0.705433	0.031674	0.699259	0.691852	0.713333	0.652222	0.721852	0.733333	0.774815	0.700741	0.677692	0.689231
96	1	-1	0	5	2	3	0.706031	0.038374	0.682222	0.783333	0.687778	0.744815	0.727778	0.736296	0.694815	0.674815	0.677692	0.650769
108	1	-1	0	5	2	4	0.697037	0.041825	0.633704	0.693333	0.751852	0.699259	0.679259	0.764815	0.669259	0.748889	0.669231	0.660769

For each of columns above, we can build a matrix M[id_alg, id_dataset]=avg(col).

In [10]:

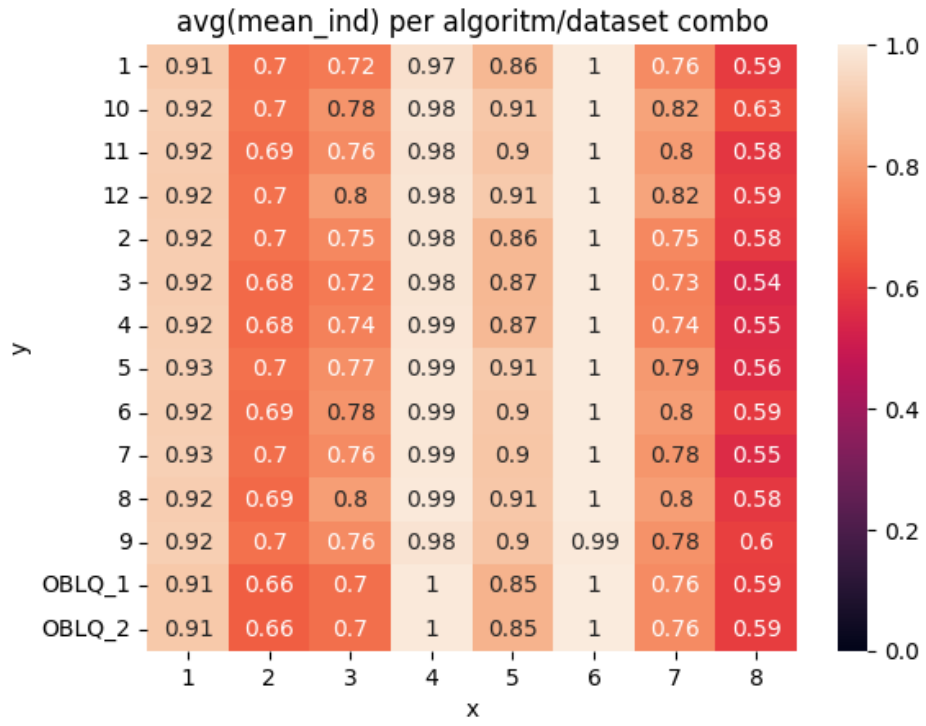
```
def create_mat(col):
    m = np.zeros((len(algo_ids), len(dataset_ids)))
    for i in range(len(algo_ids)):
        for j in range(len(dataset_ids)):
            rows = results[(results.id_alg==algo_ids[i]) & (results.id_dataset==dataset_ids[j])][col]
            m[i,j] = np.mean(rows)
    return m
```

In [11]:

```
import seaborn as sns
```

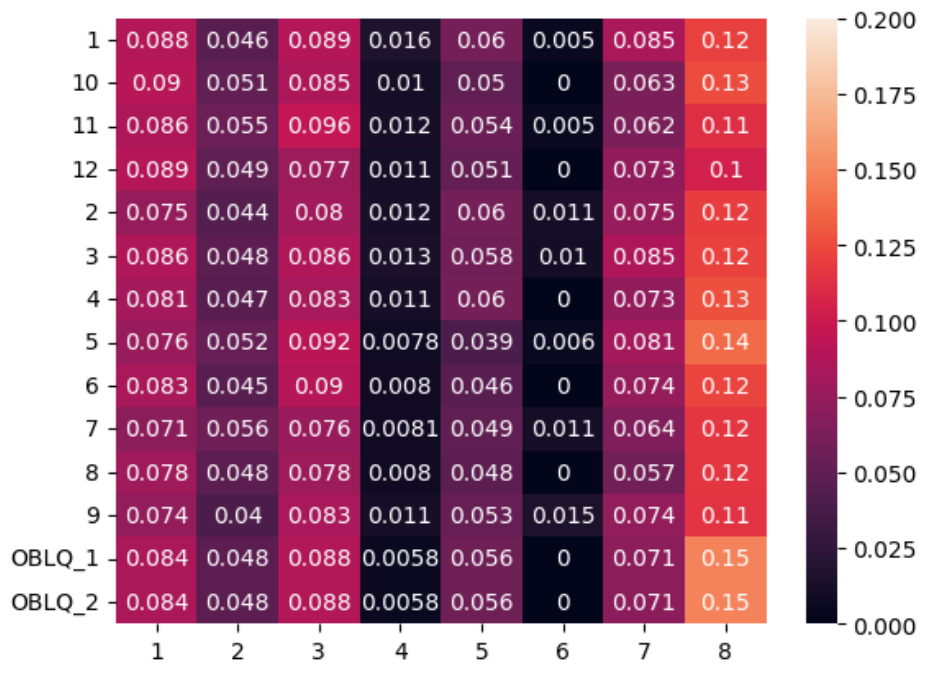
In [26]:

```
ax = sns.heatmap(create_mat('mean_ind'),vmin=0,vmax=1, annot=True,
            xticklabels=dataset_ids, yticklabels=algo_ids
           )
ax.set_title("avg(mean_ind) per algoritm/dataset combo");
ax.set_xlabel('x')
ax.set_ylabel('y')
plt.show()
```



In [27]:

```
sns.heatmap(create_mat('std_ind'),vmin=0,vmax=0.2, annot=True,
            xticklabels=dataset_ids, yticklabels=algo_ids
           )
ax.set_title("avg(std_ind) per algoritm/dataset combo");
ax.set_xlabel('x')
ax.set_ylabel('y')
plt.show()
```



There seems not to be a trivial best algorithm for all datasets. All algorithms can perform well on some datasets (ds 6: 0.99..1) and poorly on others (ds 8: ~0.5). The variances also seem to be dependent on the dataset.

Overall performance

For each algorithm we collapse all metrics for all kfolds and all datasets and perform a per-algorithm statistical analysis.

In [42]:

```
inds = [f'ind_{i}' for i in range(0,10)]
metrics = results[results.id_alg=='1'][inds]
```

In [45]:

```
metrics.sample(3)
```

Out[45]:

	ind_0	ind_1	ind_2	ind_3	ind_4	ind_5	ind_6	ind_7	ind_8	ind_9
48	1.000000	1.000000	0.800000	1.000000	1.000000	0.900000	0.700000	1.000000	0.800000	0.900000
96	0.682222	0.783333	0.687778	0.744815	0.727778	0.736296	0.694815	0.674815	0.677692	0.650769
360	0.858333	0.700000	0.783333	0.600000	0.658333	0.791667	0.883333	0.916667	0.833333	0.841667

In [47]:

```
values = metrics.values.reshape((-1,))
values.shape
```

Out[47]:

```
(400,)
```

In [67]:

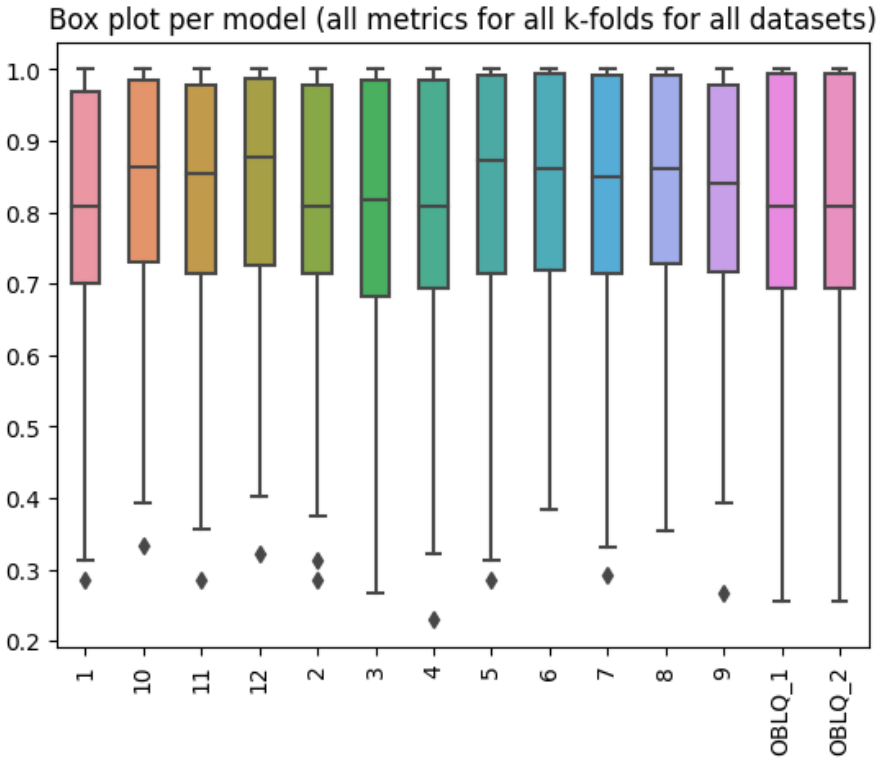
```
df = None

for i in range(len(algo_ids)):
    algo = algo_ids[i]
    metrics = results[results.id_alg==algo][inds]
    values = metrics.values.reshape((-1,))
    if df is None:
        df = pd.Series(values, name=algo).to_frame()
    else:
        df = df.join(pd.Series(values, name=algo))

ax = sns.boxplot(data=df, width = 0.5)
ax.tick_params(axis='x', labelrotation=90)
ax.set_title('Box plot per model (all metrics for all k-folds for all datasets)')
```

Out[67]:

```
Text(0.5, 1.0, 'Box plot per model (all metrics for all k-folds for all datasets)')
```



Conclusions

- **CBLQ_1 and CBLQ_2 have the highest IQR but low median, so the average performance is expected to be lower**
- **High median algorithms (11, 2) should give the best expected results**
- **Algorithm 6 seems to have the best confidence interval (highest lower bound) among all algorithms and no outliers**