In [9]:

```python
import pandas as pd
import wandb
api = wandb.Api()

# Project is specified by <entity/project-name>
runs = api.runs("robpeop/Poisoning-MNIST-binary-PGD-scored")

summary_list, config_list, name_list = [], [], []
for run in runs:
    # .summary contains the output keys/values for metrics like accuracy.
    #  We call ._json_dict to omit large files
    summary_list.append(run.summary._json_dict)

    # .config contains the hyperparameters.
    #  We remove special values that start with _.
    config_list.append(
        {k: v for k,v in run.config.items()
          if not k.startswith('_')})

    # .name is the human-readable name of the run.
    name_list.append(run.name)

runs_df = pd.DataFrame({
    "summary": summary_list,
    "config": config_list,
    "name": name_list
    })
```

In [10]:

```python
runs_df = runs_df.drop(['name'], axis=1)
len(runs_df)
```

Out[10]:

```
168
```

In [11]:

```
(runs_df['summary'][1]['adv_acc'])

runs_df['config'][1]['score']


intervals = [[0, 7], [7, 12], [12, 14], [14, 15], [15, 16], [16, 17], [17, 18],
[18, 19],
                  [19, 20], [20, 25]]

inter_mapping = list(enumerate(intervals))
inter = dict()
for i, interval in enumerate(intervals):
    str_interval = str(list(range(*interval)))
    inter[str_interval] = i

df = pd.DataFrame({'bin': [], 'adv_acc': []})
random = []
# print(inter)
# get all bins
for config, summary in zip(runs_df['config'], runs_df['summary']):
    if 'adv_acc' in summary:
        adv_acc, score = summary['adv_acc'], config['score']
        if score in inter:
            df = df.append({'bin': inter[score], 'adv_acc': adv_acc}, ignore_ind
ex=True)
        else:
#             continue
            print(score)
            random.append(adv_acc)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39]
```

In [12]:

```
df
```

Out[12]:

	bin	adv_acc
0	9.0	0.075177
1	8.0	0.142790
2	7.0	0.289835
3	6.0	0.386288
4	5.0	0.542790
...	...	...
155	4.0	0.077069
156	3.0	0.035934
157	2.0	0.030733
158	1.0	0.095508
159	0.0	0.531915

160 rows × 2 columns

In [13]:

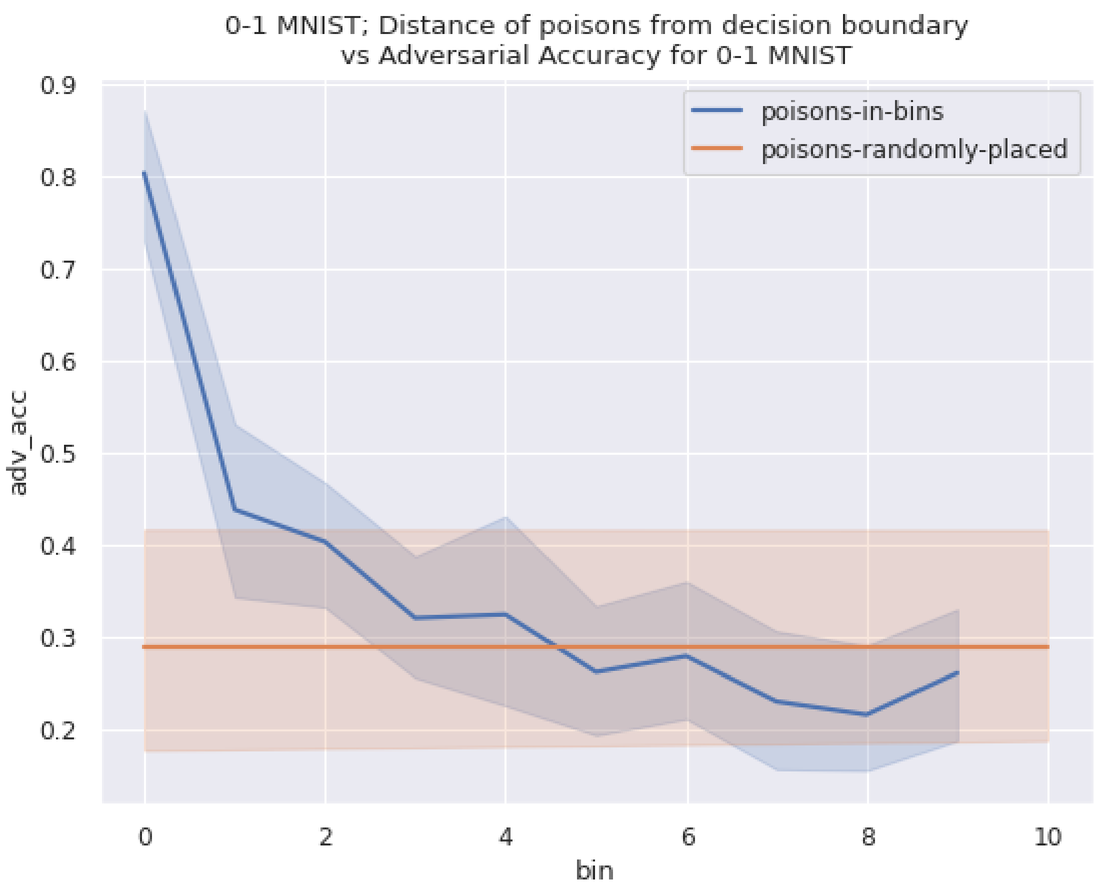
```
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

figure(figsize=(8, 6), dpi=80)
sns.set_theme()
ax = sns.lineplot(x='bin', y='adv_acc', data=df[:], label='poisons-in-bins', mar
kers=True, linewidth=2)

# def mean_confidence_interval(data, confidence=0.95):
#     a = 1.0 * np.array(data)
#     n = len(a)
#     m, se = np.mean(a), scipy.stats.sem(a)
#     h = se * scipy.stats.t.ppf((1 + confidence) / 2., n-1)
#     return m, m-h, m+h


# random_mean, low, high = mean_confidence_interval(random)
random_df = pd.DataFrame({'bin': [], 'adv_acc': []})
for interval in [0, len(intervals)]:
    for value in random:
#         print(interval, value)
        random_df = random_df.append({'bin': interval, 'adv_acc': value}, ignore
_index=True)
sns.lineplot(ax=ax, x='bin', y='adv_acc', data=random_df, label='poisons-randoml
y-placed', linewidth=2)

plt.title('0-1 MNIST; Distance of poisons from decision boundary\nvs Adversarial
Accuracy for 0-1 MNIST')
plt.show()
```



MNIST all classes

In [86]:

```
# Project is specified by <entity/project-name>
api = wandb.Api()
runs = api.runs("robpeop/Poisoning-MNIST-all-classes")

summary_list, config_list, name_list = [], [], []
for run in runs:
    # .summary contains the output keys/values for metrics like accuracy.
    #  We call ._json_dict to omit large files
    summary_list.append(run.summary._json_dict)

    # .config contains the hyperparameters.
    #  We remove special values that start with _.
    config_list.append(
        {k: v for k,v in run.config.items()
          if not k.startswith('_')})

    # .name is the human-readable name of the run.
    name_list.append(run.name)
print(len(runs))
runs_df = pd.DataFrame({
    "summary": summary_list,
    "config": config_list,
    "name": name_list
    })

runs_df = runs_df.drop(['name'], axis=1)
len(runs_df)



# (runs_df['summary'][1]['adv_acc'])

# runs_df['config'][1]['strategy']


intervals = [[0, 5], [5, 10], [10, 15], [15, 17], [17, 19], [19, 24], [24, 29]]

inter_mapping = list(enumerate(intervals))
inter = dict()
for i, interval in enumerate(intervals):
    str_interval = str(list(range(*interval)))
    inter[str_interval] = i

df = pd.DataFrame({'bin': [], 'adv_acc': []})
random = []
# print(inter)
# get all bins
for config, summary in zip(runs_df['config'], runs_df['summary']):
    if 'adv_acc' in summary:
        adv_acc, score = summary['adv_acc'], config['strategy']
#         print(adv_acc, score)
#         if score in inter:
        df = df.append({'bin': score, 'adv_acc': adv_acc}, ignore_index=True)
#         else:
# #             continue
#             print(score)
#             random.append(adv_acc)
# df
```

```
89
```

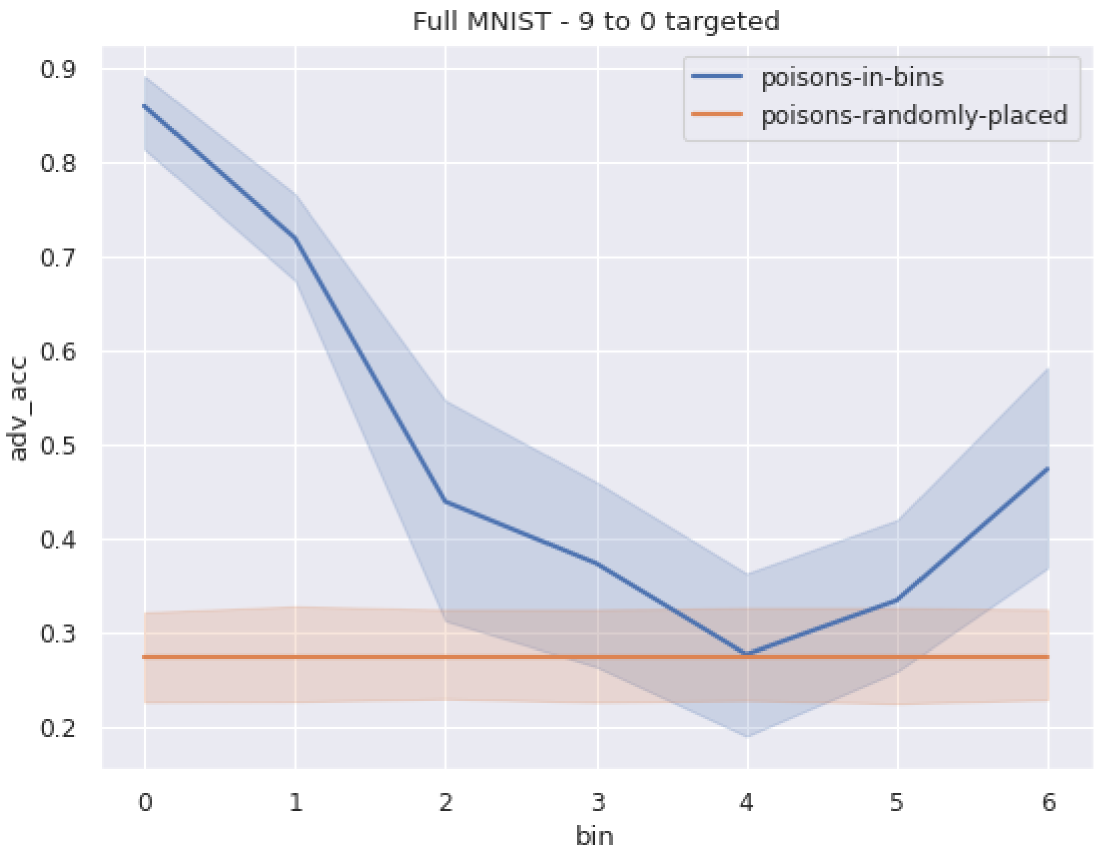
In [89]:

```
figure(figsize=(8, 6), dpi=80)
sns.set_theme()
ax = sns.lineplot(x='bin', y='adv_acc', data=df[df.bin != 'random'],
                  label='poisons-in-bins', markers=True, linewidth=2)

# def mean_confidence_interval(data, confidence=0.95):
#     a = 1.0 * np.array(data)
#     n = len(a)
#     m, se = np.mean(a), scipy.stats.sem(a)
#     h = se * scipy.stats.t.ppf((1 + confidence) / 2., n-1)
#     return m, m-h, m+h


# random_mean, low, high = mean_confidence_interval(random)
# random_df = pd.DataFrame({'bin': [], 'adv_acc': []})
# for interval in [0, len(intervals)]:
#     for value in random:
# #         print(interval, value)
random_df = pd.DataFrame({'bin': [], 'adv_acc': []})
# random_df =
for i in df[df.bin == 'random'].adv_acc:
    for j in range(7):
        random_df = random_df.append({'bin': j, 'adv_acc': i}, ignore_index=True
)
sns.lineplot(ax=ax, x='bin', y='adv_acc', data=random_df, label='poisons-randoml
y-placed', linewidth=2)

plt.title('Full MNIST - 9 to 0 targeted')
plt.show()
```



In [44]:

```
len(df[df.bin == 'random'].adv_acc)
```

Out[44]:

```
13
```