Лабораторная 4

Моисеев М33001, Муров М33011

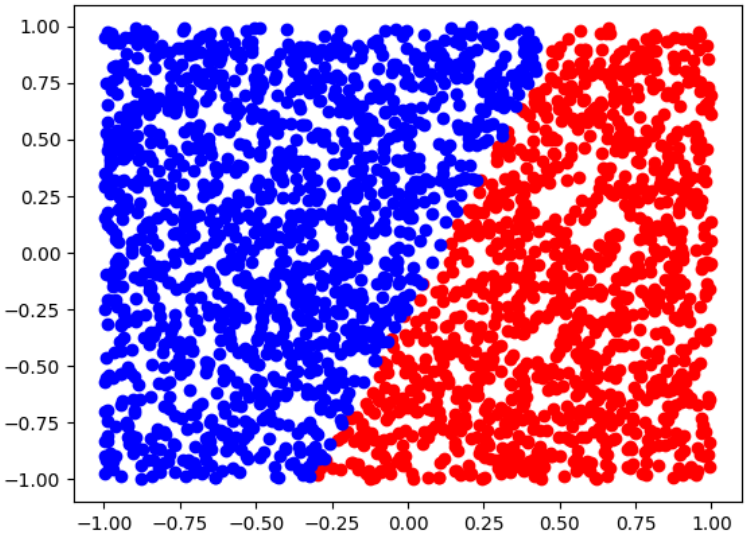
Результаты

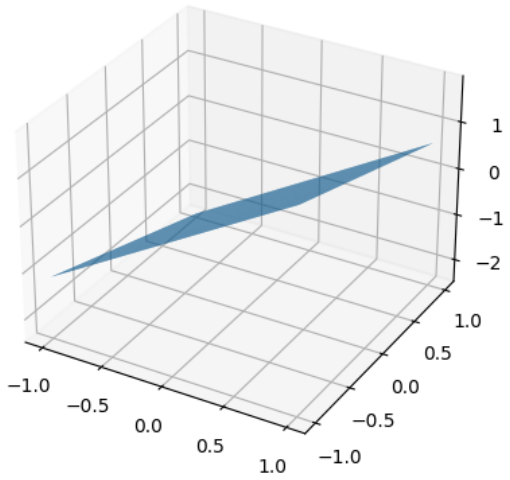
a, c

```
kernel='linear', nu=0.25
  accuracy=0.4956
  precision=0.1587719298245614
  recall=0.3747412008281574
  f1=0.2230437461491066

kernel='poly', nu=0.25
  accuracy=0.4992
  precision=0.27527761542957335
  recall=0.9751552795031055
  f1=0.4293527803099362

kernel='rbf', nu=0.25
  accuracy=0.984
  precision=1.0
  recall=0.917184265010352
  f1=0.9568034557235421

kernel='sigmoid', nu=0.25
  accuracy=0.4196
  precision=0.24896265560165975
  recall=0.9937888198757764
  f1=0.3981750311074243

kernel='rbf', nu=0.05
  accuracy=0.9972
  precision=0.9877049180327869
  recall=0.9979296066252588
  f1=0.9927909371781668

kernel='rbf', nu=0.2
  accuracy=0.9884
  precision=1.0
  recall=0.9399585921325052
  f1=0.9690501600853789

kernel='rbf', nu=0.3
  accuracy=0.9764
  precision=1.0
  recall=0.8778467908902692
  f1=0.9349503858875413
```

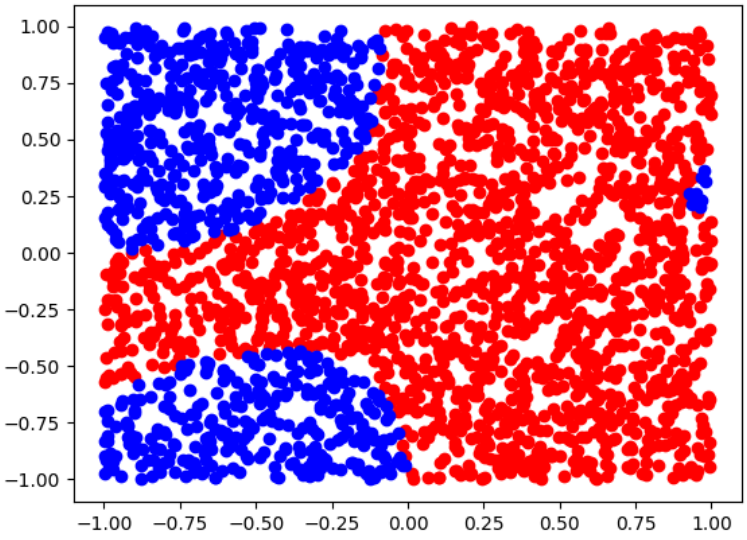
графики

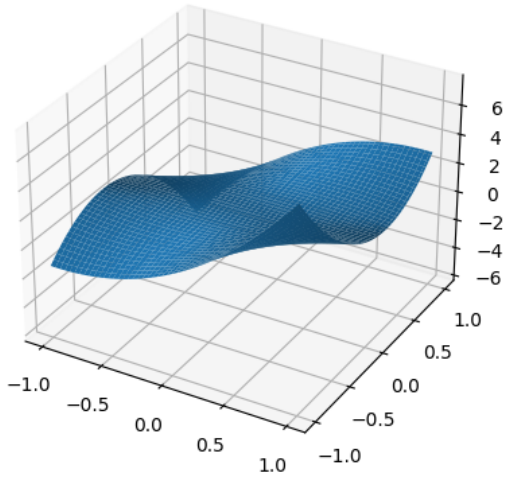
linear



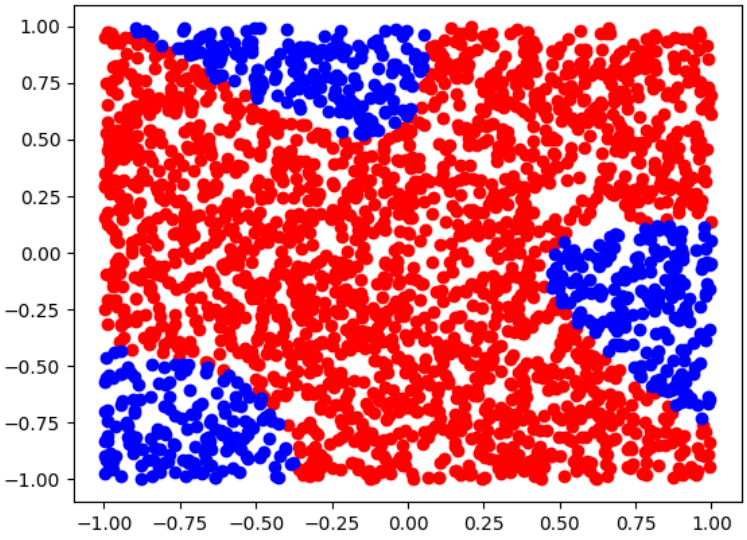


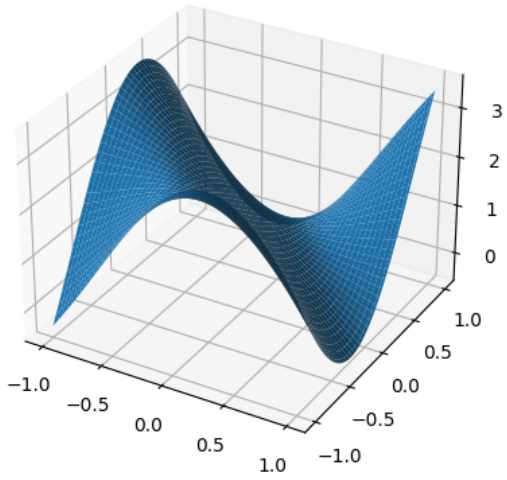
poly


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

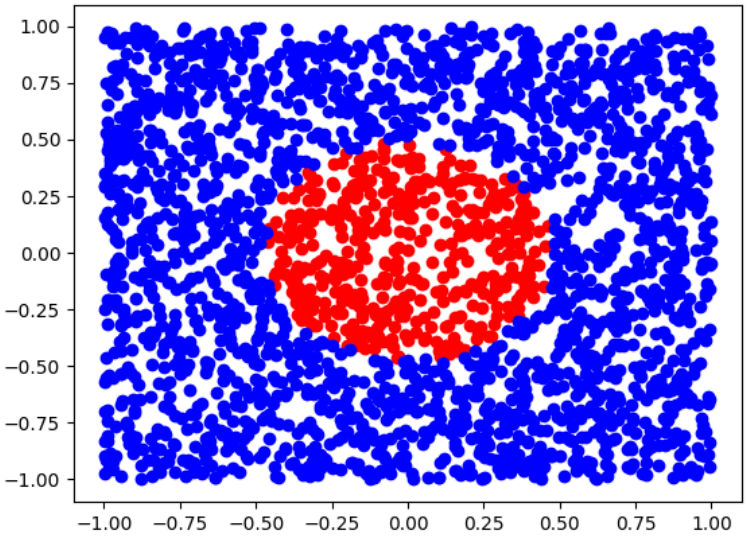

6
4
2
0
−2
−4
−6
−1.0 −0.5 0.0 0.5 1.0
1.0 0.5 0.0 −0.5 −1.0

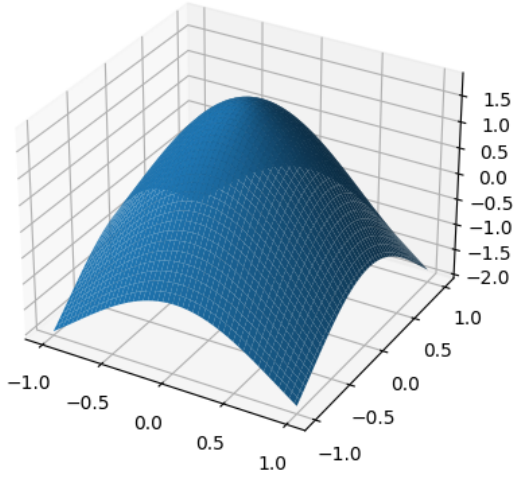
sigmoid


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

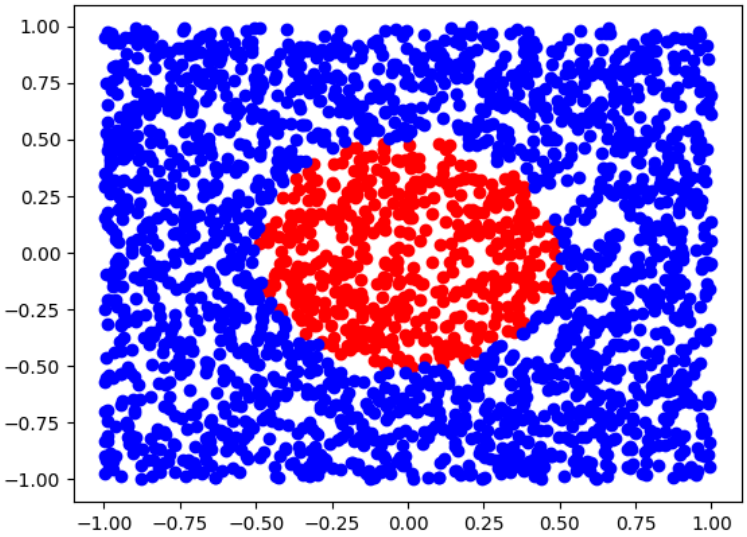

3
2
1
0
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

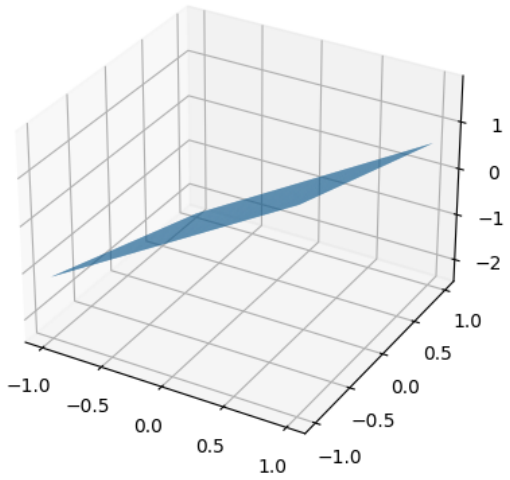
rbf 0.25


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

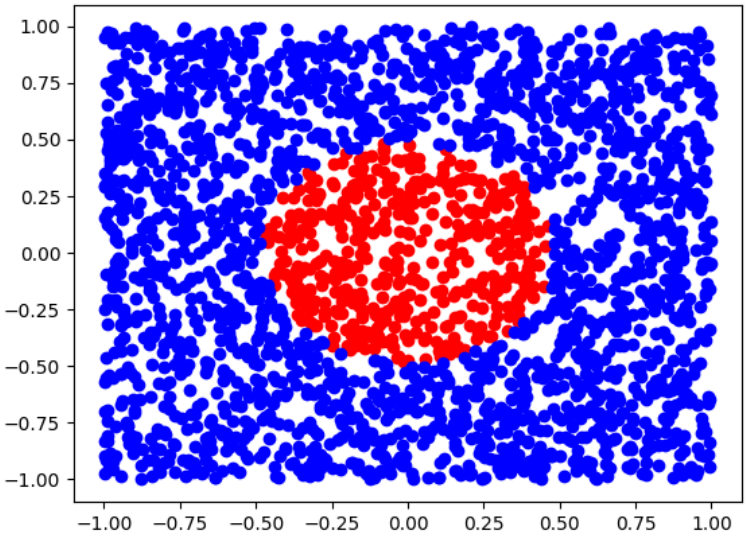

1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.0 −0.5 0.0 0.5 1.0
1.0 0.5 0.0 −0.5 −1.0

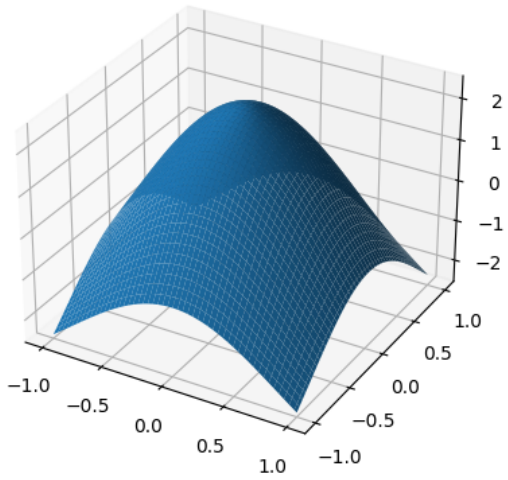
rbf 0.05


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

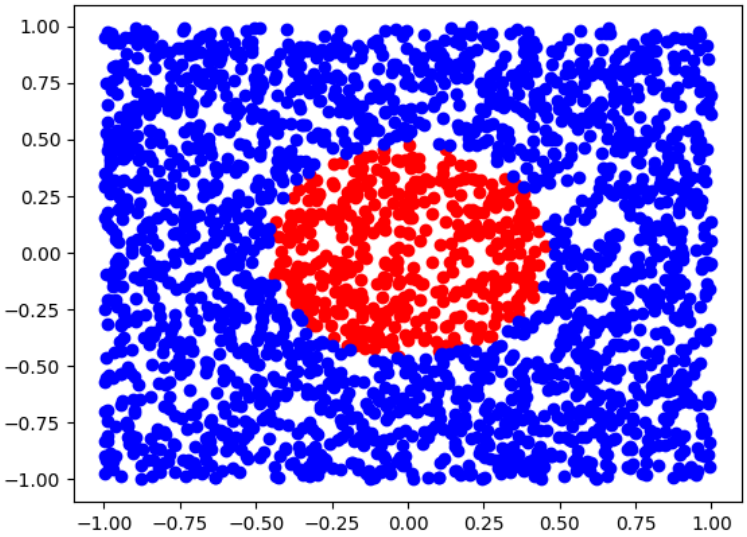

1
0
−1
−2
−1.0 −0.5 0.0 0.5 1.0
1.0 0.5 0.0 −0.5 −1.0

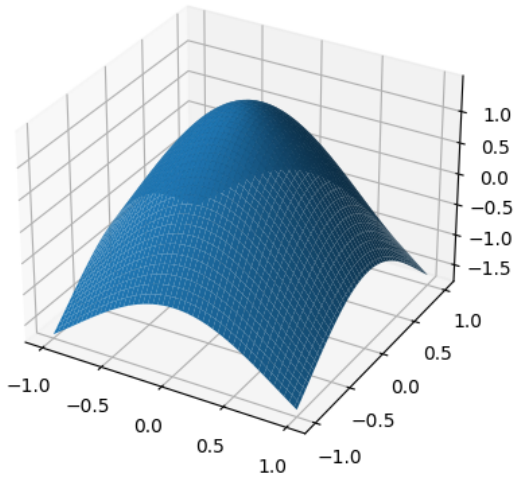
rbf 0.2


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

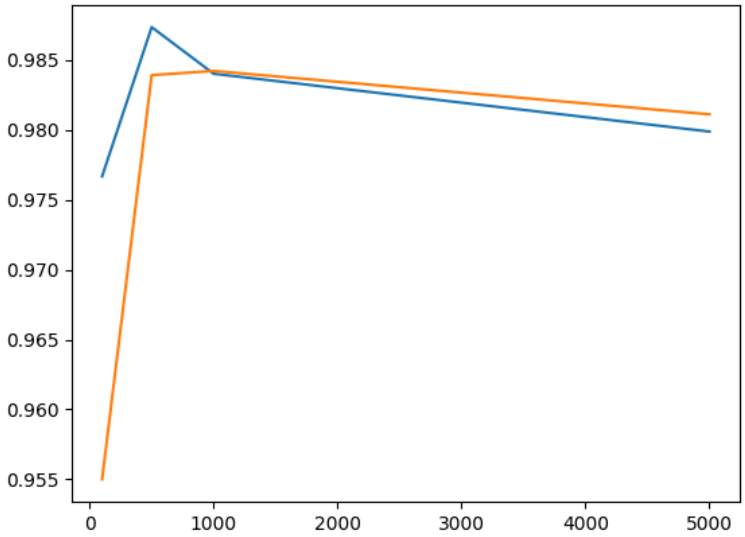

2
1
0
−1
−2
−1.0 −0.5 0.0 0.5 1.0
1.0 0.5 0.0 −0.5 −1.0

rbf 0.3

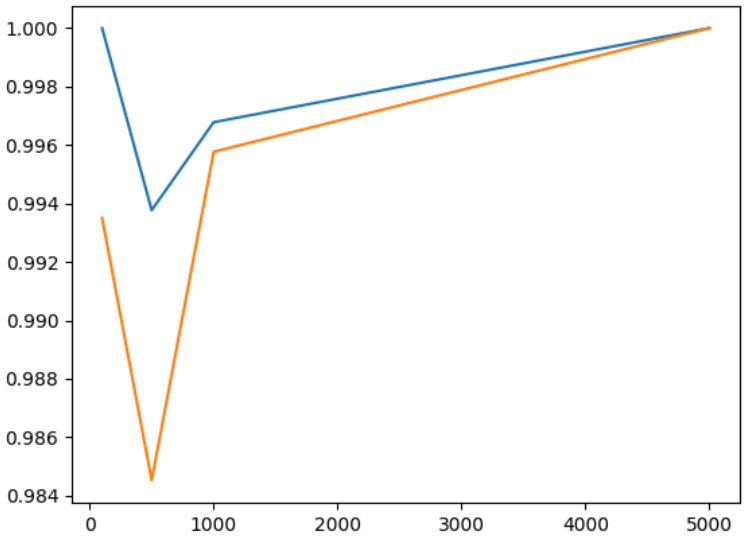

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0 −0.5 0.0 0.5 1.0
1.0 0.5 0.0 −0.5 −1.0

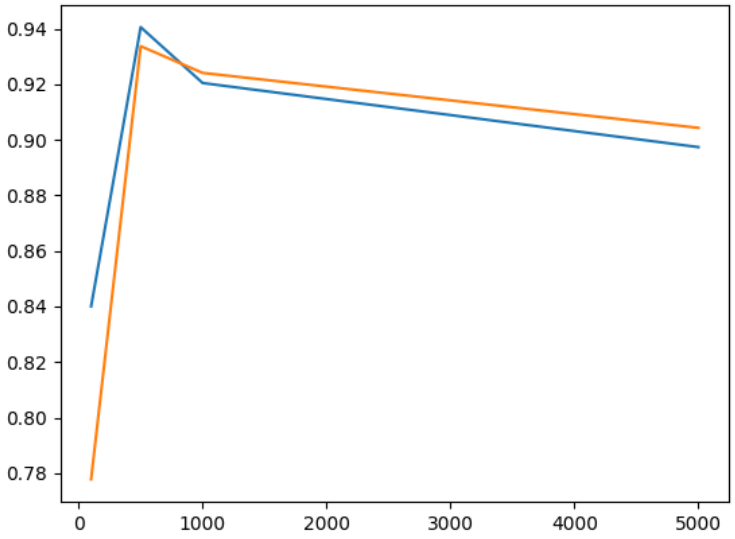
accuracy



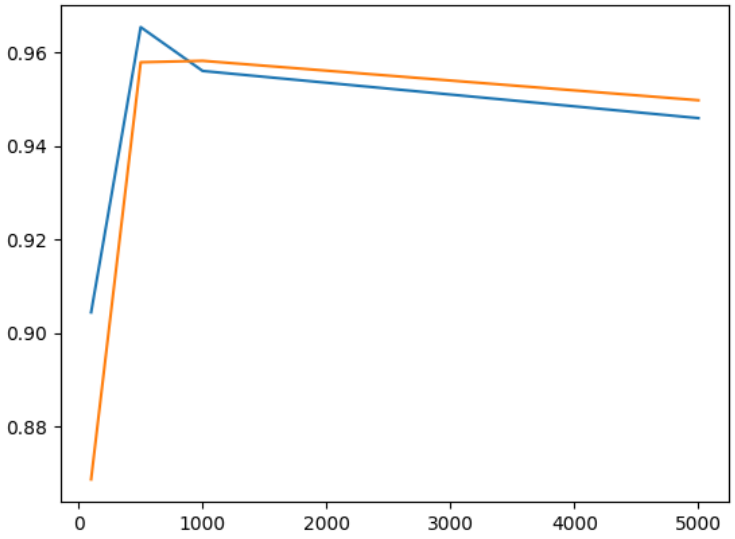
precision



recall



f1



Код

1

```
import numpy as np

rng = np.random.default_rng()


def generate(n):
    x = rng.uniform(-1, 1, (n, 2))
    y = x[:, 0] ** 2 + x[:, 1] ** 2 <= 0.25
```

```
    return x, y
```

2

```
import numpy as np
from sorcery import dict_of


def accuracy(y, y_pred):
    return np.count_nonzero(y == y_pred) / y.size


def precision(y, y_pred):
    return np.count_nonzero(y_pred[y]) / np.count_nonzero(y_pred)


def recall(y, y_pred):
    return np.count_nonzero(y_pred[y]) / np.count_nonzero(y)


def f1(y, y_pred):
    return 2 * np.count_nonzero(y_pred[y]) / (np.count_nonzero(y_pred) + np.count_nonzero(y))


metrics = dict_of(accuracy, precision, recall, f1)
```

3

```
import sys
from pathlib import Path

import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve, train_test_split
from sklearn.metrics import make_scorer
from sklearn.preprocessing import StandardScaler
from sklearn.svm import NuSVC

from .gen import generate
from .metrics import metrics

res_dir = str(Path(__file__).parent.resolve() / "res")
sys.stdout = open(f"{res_dir}/text", "w")

Xog, y = generate(10_000)
X = StandardScaler().fit_transform(Xog)
Xog_train, Xog_test, X_train, X_test, y_train, y_test = train_test_split(Xog, X, y)

xx, yy = np.meshgrid(np.linspace(-1, 1, 100), np.linspace(-1, 1, 100))


def do_ac(kernel="rbf", nu=0.25):
    svc = NuSVC(kernel=kernel, nu=nu)
    svc.fit(X_train, y_train)
```

```
    y_pred = svc.predict(X_test)

    print(f"{kernel=}, {nu=}")
    for name, metric in metrics.items():
        print(f"  {name}={metric(y_test, y_pred)}")
    print()

    plt.plot(Xog_test[y_pred, 0], Xog_test[y_pred, 1], "ro")
    plt.plot(Xog_test[~y_pred, 0], Xog_test[~y_pred, 1], "bo")
    plt.savefig(f"{res_dir}/ac_points_{kernel}_{nu}.png")
    plt.clf()

    fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
    z = svc.decision_function(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
    ax.plot_surface(xx, yy, z)
    plt.savefig(f"{res_dir}/ac_surface_{kernel}_{nu}.png")
    plt.clf()


for kernel in ["linear", "poly", "rbf", "sigmoid"]:
    do_ac(kernel=kernel)
for nu in [0.05, 0.2, 0.3]:
    do_ac(nu=nu)

for name, metric in metrics.items():
    train_sizes, train_scores, test_scores = learning_curve(
        NuSVC(kernel="rbf", nu=0.25),
        X,
        y,
        cv=3,
        scoring=make_scorer(metric),
        train_sizes=[10, 50, 100, 500, 1000, 5000],
    )
    plt.plot(train_sizes, np.mean(train_scores, axis=1))
    plt.plot(train_sizes, np.mean(test_scores, axis=1))
    plt.savefig(f"{res_dir}/b_{name}.png")
    plt.clf()

sys.stdout.close()
```