Прикладная математика
Лабораторная работа №8
SVM

M33001
ВОЕВОДСКИЙ ДМИТРИЙ
ЕВТУШЕНКО ИВАН
БЛАЖКОВ АЛЕКСАНДР

2022 год, университет ИТМО

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm, trange

np.random.seed(42)
```

Машина опорных векторов отлично показывает себя в случаях бинарной классификации, когда выборка не является линейно разделимой.

Для примера сгенерируем данные с признаками (x, y) и границей разделения $x^2 + y^2 = \frac{1}{4}$

```
def generate(size=5000, a=-1, b=1, treshold=1/4):
    coords = np.random.rand(size, 2)
    coords = coords * abs(b - a) + a
    marks = np.array(list(map(lambda x: int(x[0] ** 2 + x[1] ** 2 > treshold), coords)))
    if np.unique(marks).size == 1:
        return generate(size, a, b, treshold)
    df = pd.DataFrame(coords, columns=['x', 'y'])
    df['mark'] = marks
    return df
```

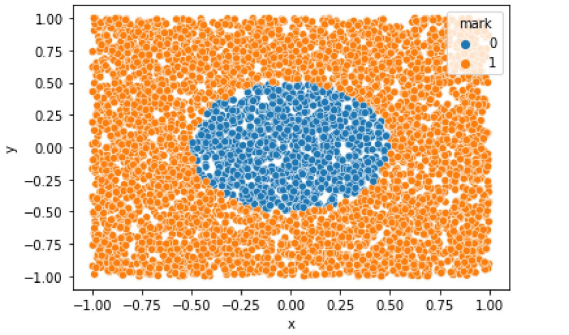
```
df = generate()
```

```
df.head()
```

	x	y	mark
0	-0.250920	0.901429	1
1	0.463988	0.197317	1
2	-0.687963	-0.688011	1
3	-0.883833	0.732352	1
4	0.202230	0.416145	0

```
sns.scatterplot(data=df, x='x', y='y', hue='mark')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2b0a636af0>
```



Подготовим вспомогательные методы для получения метрик качества будущих моделей.

```
class Metrics:
    @staticmethod
    def get_metrcis_names():
        return ['accuracy', 'precision', 'recall', 'f1']

    @staticmethod
    def get_metrics(y_true, y_pred):
        return {
            'accuracy': Metrics.accuracy(y_true, y_pred),
            'precision': Metrics.precision(y_true, y_pred),
            'recall': Metrics.recall(y_true, y_pred),
```

```
            'f1': Metrics.f1(y_true, y_pred)
        }

    @staticmethod
    def accuracy(y_true, y_pred):
        return np.mean(y_true == y_pred)

    @staticmethod
    def precision(y_true, y_pred):
        tp = np.sum(y_true * y_pred)
        fp = np.sum((1 - y_true) * y_pred)
        return tp / (tp + fp)

    @staticmethod
    def recall(y_true, y_pred):
        tp = np.sum(y_true * y_pred)
        fn = np.sum(y_true * (1 - y_pred))
        return tp / (tp + fn)

    @staticmethod
    def f1(y_true, y_pred):
        p = Metrics.precision(y_true, y_pred)
        r = Metrics.recall(y_true, y_pred)
        return 2 * p * r / (p + r)

    @staticmethod
    def report(y_true, y_pred):
        print(f'accuracy:  {Metrics.accuracy(y_true, y_pred):.4f}')
        print(f'precision: {Metrics.precision(y_true, y_pred):.4f}')
        print(f'recall:    {Metrics.recall(y_true, y_pred):.4f}')
        print(f'f1:        {Metrics.f1(y_true, y_pred):.4f}')
```

Импортируем необходимые инструменты из библиотеки `sklearn`.

```
from sklearn.model_selection import train_test_split
from sklearn.svm import NuSVC
from sklearn.model_selection import GridSearchCV
```

Разделяем данные на данные для обучения и для тестирования.

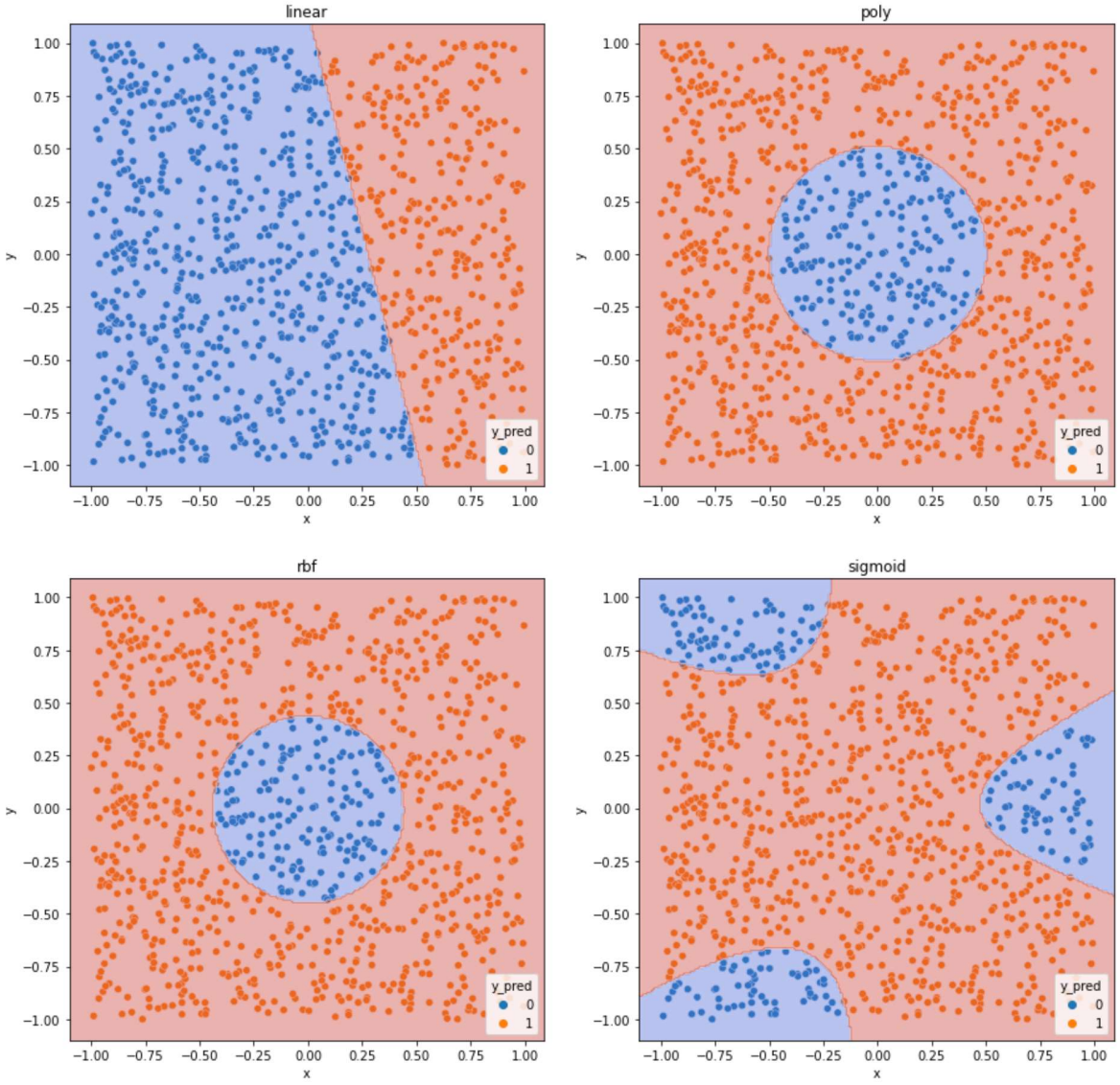
```
X_train, X_test, y_train, y_test = train_test_split(df[['x', 'y']], df['mark'], test_size=0.2)
```

Определим, какое ядро модели лучше всего решает задачу классификации в нашем случае.

- `linear`
- `poly`
- `rbf`
- `sigmoid`

```
fig, axs = plt.subplots(ncols=2, nrows=2, figsize=(15, 15))
for i, kernel in enumerate(['linear', 'poly', 'rbf', 'sigmoid']):
    model = NuSVC(kernel=kernel, nu=0.4, degree=2)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(kernel)

    res_df = pd.DataFrame(X_test, columns=['x', 'y'])
    res_df['y_pred'] = y_pred

    sns.scatterplot(data=res_df, x='x', y='y', hue='y_pred', ax=axs[i // 2, i % 2])
    axs[i // 2, i % 2].set_title(kernel)
    x_min, x_max = X_train['x'].min() - 0.1, X_train['x'].max() + 0.1
    y_min, y_max = X_train['y'].min() - 0.1, X_train['y'].max() + 0.1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    axs[i // 2, i % 2].contourf(xx, yy, Z, alpha=0.4, cmap='coolwarm')


    Metrics.report(y_test, y_pred)
    print()
```

```
linear
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but NuSVC was fitted with
  warnings.warn(
accuracy:  0.4590
precision: 0.9043
recall:    0.3649
f1:        0.5200

poly
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but NuSVC was fitted with
  warnings.warn(
accuracy:  0.9930
precision: 1.0000
recall:    0.9913
f1:        0.9956

rbf
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but NuSVC was fitted with
  warnings.warn(
accuracy:  0.9680
precision: 0.9617
recall:    1.0000
f1:        0.9805

sigmoid
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but NuSVC was fitted with
  warnings.warn(
accuracy:  0.5980
precision: 0.7522
recall:    0.7447
f1:        0.7484
```

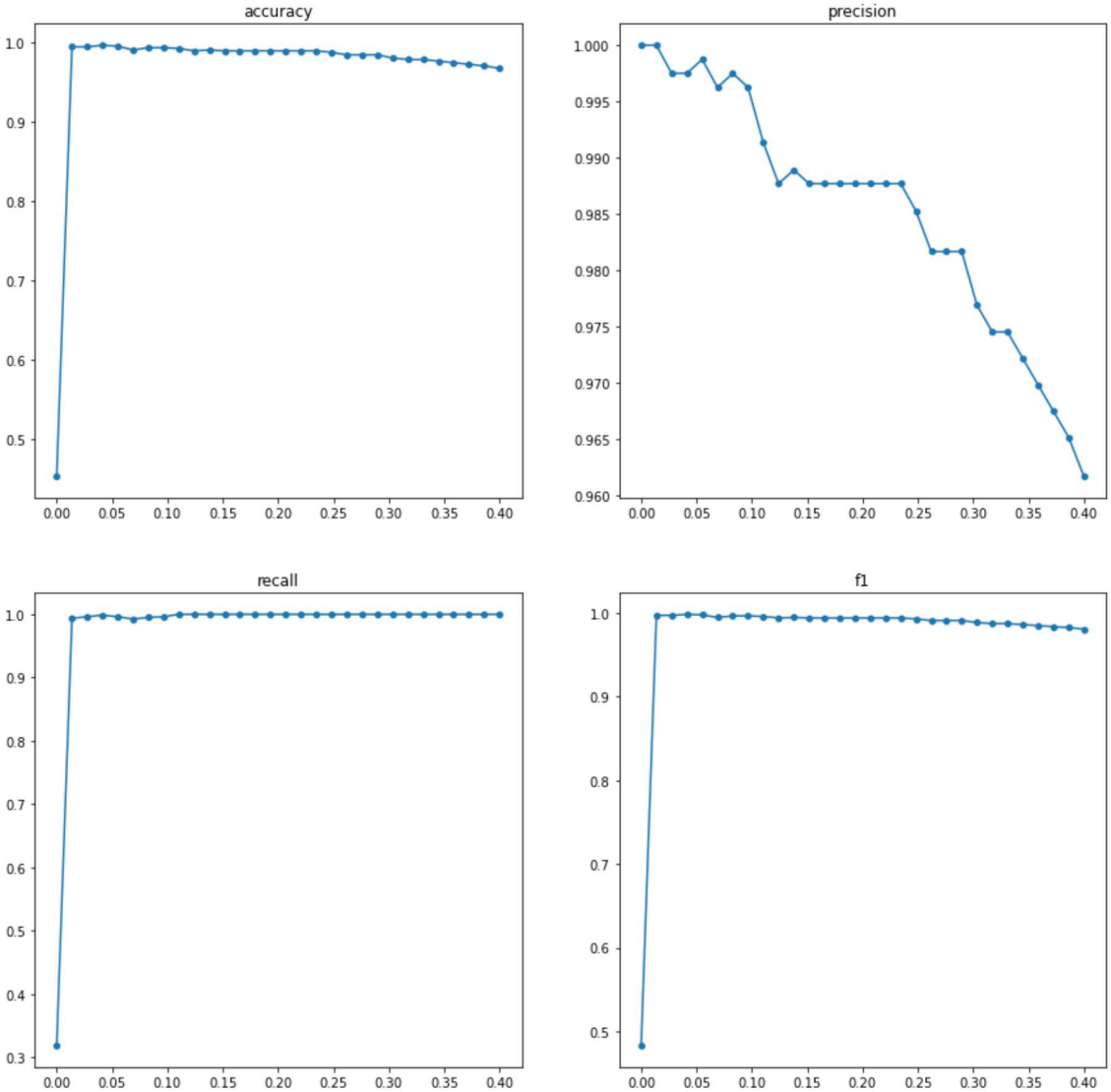


Как видим, правильная классификация может быть осуществлена с помощью ядер `poly` и `rbf`. Далее будем рассматривать ядро `rbf`, так как именно это нужно делать по плану лабораторной работы!

Double-click (or enter) to edit

Определим, как метрики качества зависят от нижней границы доли опорных векторов ν.

```
nus = np.linspace(1e-10, 0.4, 30)
metrics = {}
for metric in Metrics.get_metrcis_names():
    metrics[metric] = []

for nu in nus:
    model = NuSVC(kernel='rbf', nu=nu)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    calc_metrics = Metrics.get_metrics(y_test, y_pred)
    for metric in calc_metrics.keys():
        metrics[metric].append(calc_metrics[metric])

fig, axs = plt.subplots(ncols=2, nrows=2, figsize=(15, 15))
for i, metric in enumerate(metrics.keys()):
    sns.lineplot(x=nus, y=metrics[metric], ax=axs[i // 2, i % 2])
    sns.scatterplot(x=nus, y=metrics[metric], ax=axs[i // 2, i % 2])
    axs[i // 2, i % 2].set_title(metric)
```



Как видно по графикам, с увеличением доли опорных векторов качество модели снижается. Скорее всего, это связано с т.н. переобучением. Переобучение в этом случае, это не "обучение заново", как могли подумать некоторые читатели, а феномен "хороших" показателей метрик на данных для обучения, и не таких "хороших" показалетей на тестовых данных.

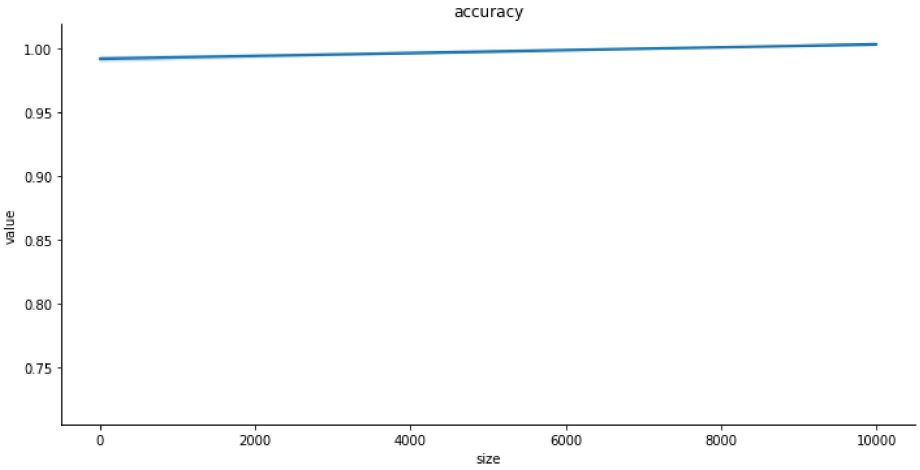
```
def smooth_line(y, window=10):
    return np.convolve(y, np.ones(window) / window, mode='valid')
```

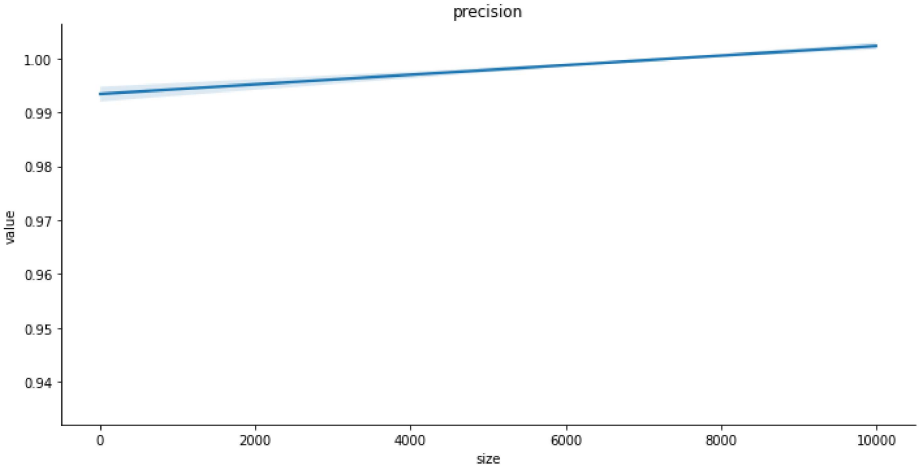
Посмотрим, как размер обучающей выборки влияет на метрики качества.

```
metrics = {}

import warnings
warnings.filterwarnings("ignore")

test = generate(50)
X_test, y_test = test[['x', 'y']], test['mark']
# nus = []

for metric in Metrics.get_metrcis_names():
    metrics[metric] = []

sizes = range(10, 10000, 10)

for size in tqdm(sizes):
    model = NuSVC(kernel='rbf')
    train = generate(size)

    # search = GridSearchCV(model, {'nu': np.linspace(1e-10, 0.4, 30)}, cv=5, scoring='f1', refit=True, n_jobs=-1, verbose=0).fit(train[['
    # nus.append(search.best_params_['nu'])

    model = NuSVC(kernel='rbf', nu=0.1)

    X_train, y_train = train[['x', 'y']], train['mark']
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    calc_metrics = Metrics.get_metrics(y_test, y_pred)

    for metric in calc_metrics.keys():
        metrics[metric].append(calc_metrics[metric])
```

```
100%                                                  999/999 [05:37<00:00, 1.08it/s]
```

```
for i, metric in enumerate(metrics.keys()):
    data = pd.DataFrame({'size': sizes, 'value': metrics[metric]})
    sns.lmplot(data=data, x='size', y='value', height=5, aspect=2, markers='')
    plt.title(metric)
```





Невероятно, но факт: чем больше размер тестовых данных, тем лучше показатели метрик. Однако в нашем случае ощутимого роста, по крайней мере, визуально не наблюдается. Это связано с тем, что в сгенерированных данных отсутствует шум.