МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И.УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №8
по дисциплине «Машинное обучение»
Тема: Классификация (линейный дискриминантный анализ, метод опорных векторов)

Студент гр. 6304 _____ Корытов П.В.
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы

Ознакомиться с методами классификации модуля Sklearn.

1. Линейный дискриминантный анализ

1. Произведена загрузка данных. Часть набора данных представлена на листинге 1.

Листинг 1. Набор данных

```
      0    1    2    3               4
0    5.1  3.5  1.4  0.2     Iris-setosa
1    4.9  3.0  1.4  0.2     Iris-setosa
2    4.7  3.2  1.3  0.2     Iris-setosa
3    4.6  3.1  1.5  0.2     Iris-setosa
4    5.0  3.6  1.4  0.2     Iris-setosa
..   ...  ...  ...  ...             ...
145  6.7  3.0  5.2  2.3  Iris-virginica
146  6.3  2.5  5.0  1.9  Iris-virginica
147  6.5  3.0  5.2  2.0  Iris-virginica
148  6.2  3.4  5.4  2.3  Iris-virginica
149  5.9  3.0  5.1  1.8  Iris-virginica

[150 rows x 5 columns]
```

2. Произведена классификация наблюдений с использованием LDA. Результаты представлены на листинге 2.

Листинг 2. Результаты классификации LDA

```
Количество ошибок: 4
score: 0.9466666666666667
```

3. Параметры LDA:
   - `solver`:
     - `svd` — разложение по сингулярным числам (singular value decompositi
     - `lsqr` — метод наименьшие квадратов;
     - `eigen` — разложение на собственные числа;
   - `shrinkage` — регуляризация, может применятся, если образцов невелико по сравнению количеством признаков;
   - `priors` — априорные верятности;
   - `n_components` — количество компонентов в `transform`;
   - `store_covariance` — вычислить матрицу ковариантности для `svd`;

- `tol` — порог значимости для X.

Атрибуты:

- `coef_` — вектора весов;
- `intercept_` — точки перехвата;
- `covariance_` — внутри-классовые матрицы ковариантности;
- `explained_variance_ratio_` — объясненная дисперсия для кажого компонента;
- `means_` — средние значения в классах;
- `priors_` — априорные вероятности классов;
- `scaling_` — масштабирование признаков в пространстве, охватываемом центроидами классов;
- `xbar_` — общее среднее;
- `classes_` — метки классов.

4. `transform` может применятся для уменьшения размерности данных. Результаты применения `transform` представлены на рис. 1.

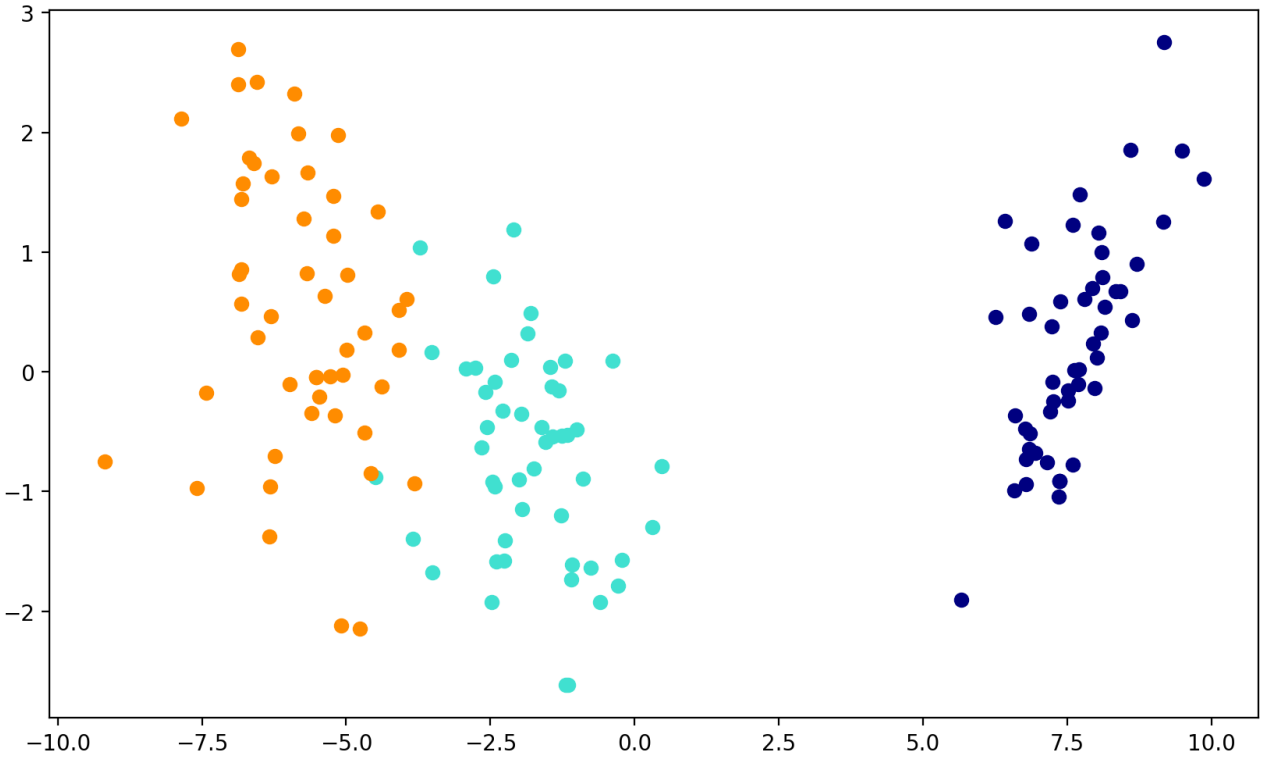

Рисунок 1 – Результаты transform

5. С помощью `GridSearchCV` проведено исследование работы классификатора при различных параметрах `solver`, `shrinkage`. Результаты в таблице 1.

Таблица 1. Работа линейного дискриминатного анализа

	param_solver	param_shrinkage	mean_test_score	std_test_score
0	lsqr	auto	0.98	0.0266667
1	eigen	auto	0.98	0.0266667
2	lsqr	0.0	0.98	0.0266667
3	eigen	0.0	0.98	0.0266667
4	lsqr	0.1	0.98	0.0266667
5	eigen	0.1	0.98	0.0266667
6	lsqr	0.2	0.973333	0.0249444
7	eigen	0.2	0.973333	0.0249444
8	lsqr	0.30000000000000004	0.966667	0.0298142
9	eigen	0.30000000000000004	0.966667	0.0298142
10	lsqr	0.4	0.98	0.0163299
11	eigen	0.4	0.98	0.0163299
12	lsqr	0.5	0.98	0.0163299
13	eigen	0.5	0.98	0.0163299
14	lsqr	0.6000000000000001	0.96	0.0249444
15	eigen	0.6000000000000001	0.96	0.0249444
16	lsqr	0.7000000000000001	0.953333	0.0266667
17	eigen	0.7000000000000001	0.953333	0.0266667
18	lsqr	0.8	0.953333	0.0266667
19	eigen	0.8	0.953333	0.0266667
20	lsqr	0.9	0.94	0.038873
21	eigen	0.9	0.94	0.038873
22	lsqr	1.0	0.92	0.0339935
23	eigen	1.0	0.92	0.0339935

Как видно, выбранный `solver` не оказал влияния на результаты. Ручная установка `shrinkage` не дала лучших результатов, чем установка по лемме Ледота-Вольфа.

6. Построен график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки. Результат на рис. 2.

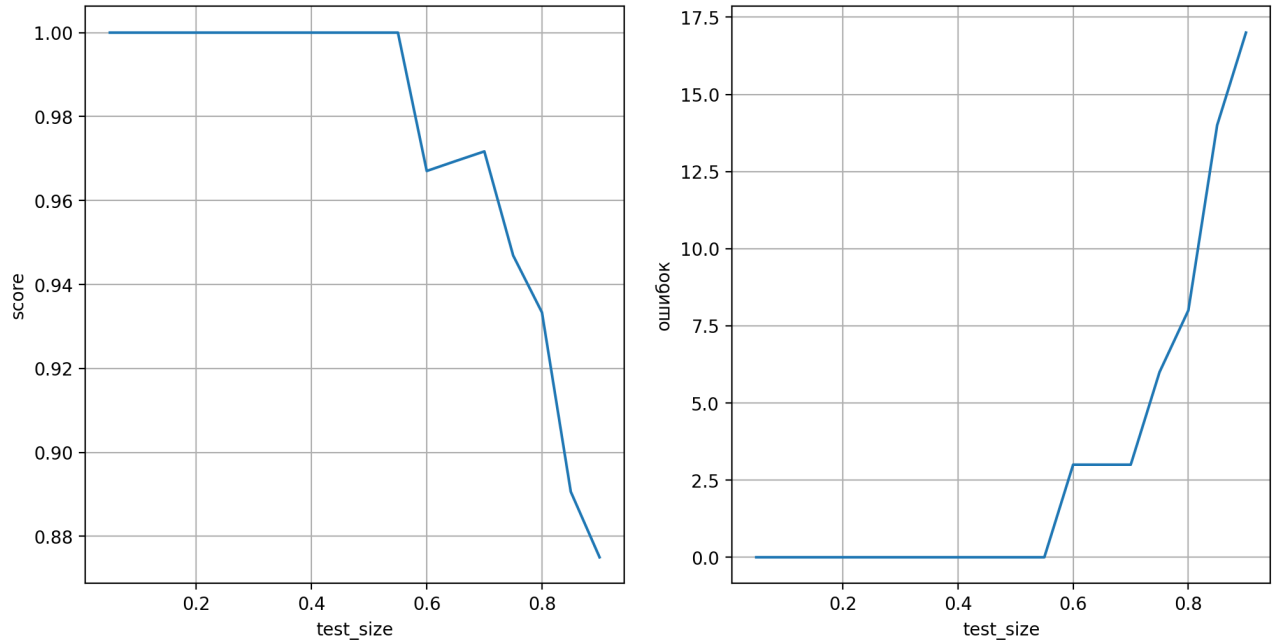

Рисунок 2 – График зависимости точности и количества ошибок от `test_size`

Как и следовало ожидать, алгоритм работает хуже при уменьшении размера набора для обучения.

7. Задание априорных вероятностей не изменило результаты работы алгоритма (рис. 3). Большее влияние на результаты оказало изменение `random_seed`

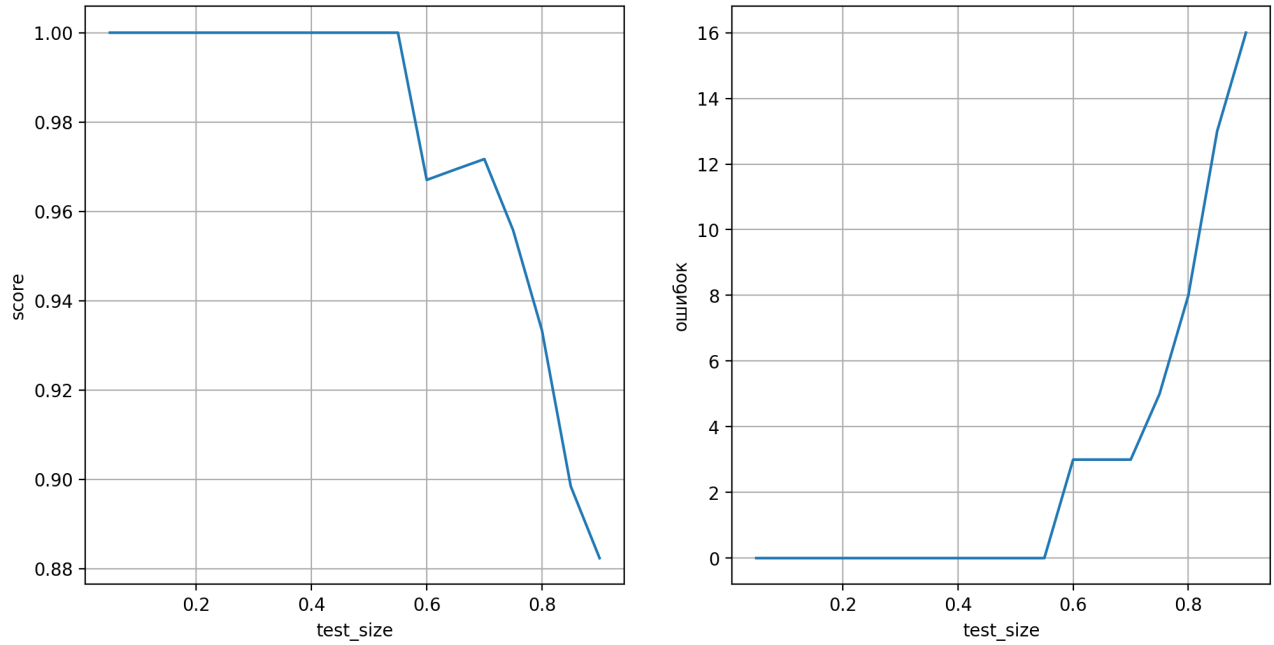

Рисунок 3 – График зависимости точности и количества ошибок в случае заданных априорных вероятностей

2. Метод опорных векторов

1. Проведено классификация `SVM` для тех же данных. Результаты на рис. 3.

Листинг 3. Резульаты классификации SVM

```
Количество ошибок: 4
score: 0.9466666666666667
```

2. Выведены некоторые атрибуты классификатора. Результаты на листинге 4.
   - `support_` — индексы опорных векторов;
   - `support_vectors_` — опорные векторы;
   - `n_support_` — количество опорных векторов для каждого класса.

3. Построен график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки. Результат на рис. 4.

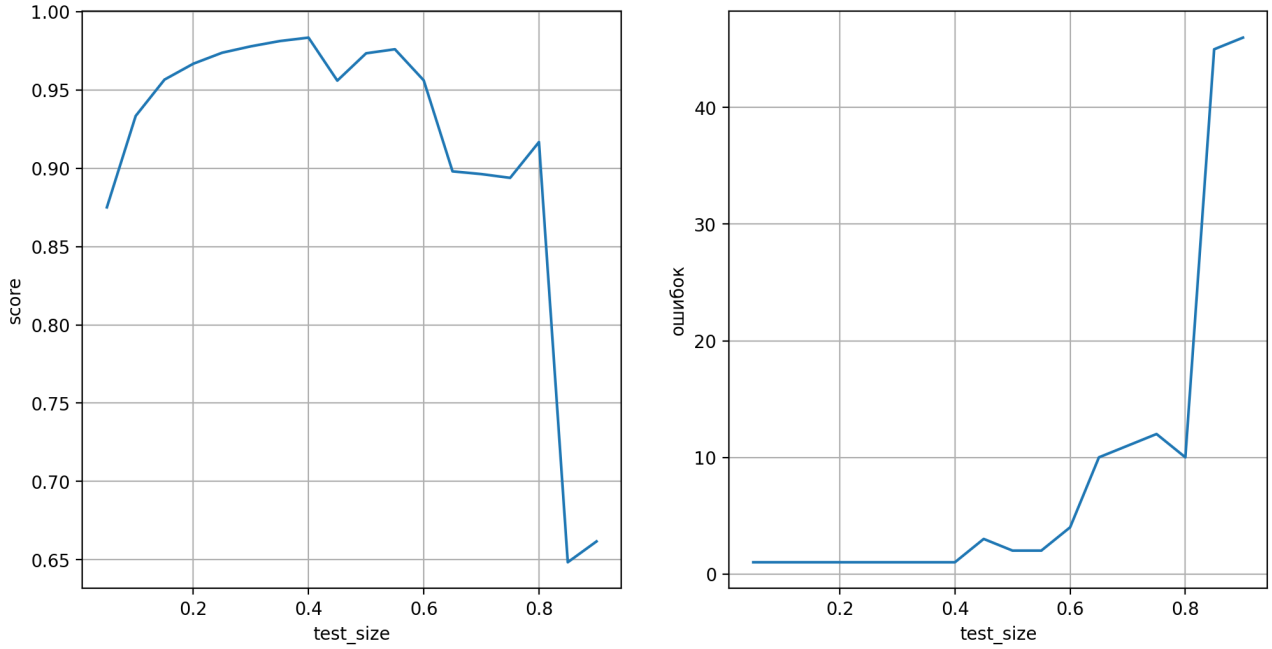

Рисунок 4 – SVC

Как и в предыдущем случае, результаты работы становятся хуже при мальньком наборе тестовой выборки.

Листинг 4. Атрибуты классификатора

```
support_vectors_: [[4.5 2.3 1.3 0.3]
 [5.4 3.9 1.7 0.4]
 [5.1 3.3 1.7 0.5]
 [5.  3.  1.6 0.2]
 [5.1 2.5 3.  1.1]
 [6.2 2.2 4.5 1.5]
 [5.7 2.9 4.2 1.3]
 [5.7 2.8 4.5 1.3]
 [6.6 3.  4.4 1.4]
 [6.4 2.9 4.3 1.3]
 [4.9 2.4 3.3 1. ]
 [6.7 3.1 4.4 1.4]
 [5.7 2.6 3.5 1. ]
 [6.3 2.5 4.9 1.5]
 [6.7 3.  5.  1.7]
 [5.5 2.4 3.7 1. ]
 [6.6 2.9 4.6 1.3]
 [5.6 3.  4.1 1.3]
 [5.9 3.2 4.8 1.8]
 [6.3 2.3 4.4 1.3]
 [5.9 3.  5.1 1.8]
 [6.4 2.8 5.6 2.1]
 [6.5 3.2 5.1 2. ]
 [6.2 3.4 5.4 2.3]
 [5.7 2.5 5.  2. ]
 [6.9 3.1 5.4 2.1]
 [7.2 3.  5.8 1.6]
 [7.9 3.8 6.4 2. ]
 [6.  3.  4.8 1.8]
 [6.4 3.2 5.3 2.3]
 [6.7 3.  5.2 2.3]
 [5.8 2.7 5.1 1.9]
 [6.3 2.9 5.6 1.8]]
support_: [16 26 36 59  2  4  6 33 34 37 40 42 54 57 58 60 64 65 66 67  1 11 14
 ↪  17
 19 20 23 41 44 55 56 62 71]
n_support_: [ 4 16 13]
```

4. С помощью `GridSearchCV` исследована работа метода при разных значениях параметров. Результаты представлена в таблице 2.

Таблица 2. Результаты работы

	param_kernel	param_max_iter	param_degree	mean_test_score	std_test_score
1	linear	10	nan	0.993	0.013
16	poly	-1	2.000	0.987	0.016
19	poly	1000	2.000	0.987	0.016
18	poly	100	2.000	0.987	0.016
13	poly	10	1.000	0.980	0.027
23	poly	1000	3.000	0.980	0.016
22	poly	100	3.000	0.980	0.016
20	poly	-1	3.000	0.980	0.016
0	linear	-1	nan	0.980	0.016
5	rbf	10	nan	0.980	0.016
2	linear	100	nan	0.980	0.016
3	linear	1000	nan	0.980	0.016
31	poly	1000	5.000	0.973	0.033
28	poly	-1	5.000	0.973	0.033
30	poly	100	5.000	0.967	0.042
27	poly	1000	4.000	0.967	0.042
26	poly	100	4.000	0.967	0.042
24	poly	-1	4.000	0.967	0.042
21	poly	10	3.000	0.967	0.042
4	rbf	-1	nan	0.967	0.021
6	rbf	100	nan	0.967	0.021
7	rbf	1000	nan	0.967	0.021
15	poly	1000	1.000	0.953	0.027
14	poly	100	1.000	0.953	0.027
12	poly	-1	1.000	0.953	0.027
17	poly	10	2.000	0.940	0.049
29	poly	10	5.000	0.840	0.151
25	poly	10	4.000	0.807	0.188
10	sigmoid	100	nan	0.067	0.060
8	sigmoid	-1	nan	0.067	0.060
11	sigmoid	1000	nan	0.067	0.060
9	sigmoid	10	nan	0.053	0.050

Как видно, результаты для ядра `sigmoid` значительно хуже.

5. Аналогичные исследования проведены для `NuSVC` и `LinearSVC`.
   - `NuSVC` — аналогичен `SVC`, но дает параметр для контроля числа опорных векторов.
   - `LinearSVC` — аналогичен `SVC` с `kernel='linear'`, но дает больше параметров для этого случае (регуляризация и т.п.).

   Для `NuSVC` построен график зависимости `score` из кросс-валидации и параметра `nu`. Результаты на рис. 5.

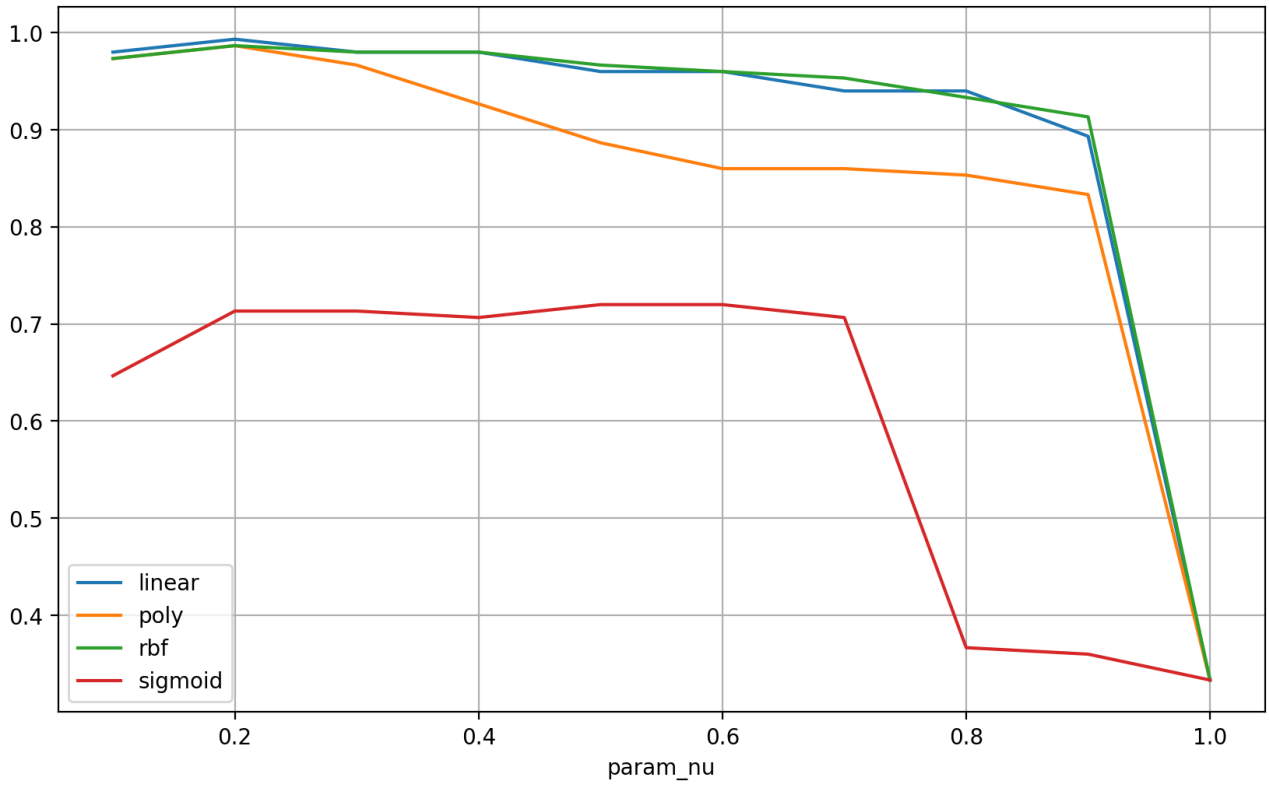

Рисунок 5 – NuSVC

Как видно, в случае `NuSVC` большой параметр `nu` снижает качество классификации, причем сильнее в случае полиномиального ядра.

Исследование для `LinearSVC` представлено в таблице 3.

Таблица 3. LinearSVC

	param_penalty	param_loss	param_C	mean_test_score	std_test_score
2	l2	hinge	1	0.94	0.0489898
4	l2	hinge	10	0.933333	0.0557773
0	l2	hinge	0.1	0.766667	0.0210819
1	l2	squared_hidge	0.1	nan	nan
3	l2	squared_hidge	1	nan	nan
5	l2	squared_hidge	10	nan	nan
6	l1	squared_hidge	0.1	nan	nan
7	l1	squared_hidge	1	nan	nan
8	l1	squared_hidge	10	nan	nan

3. Выводы

Произведено знакомство с классификацией методом линейного дискриминатного анализа и методом опорных векторов в `sklearn`.