МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №8

по дисциплине «Машинное обучение»

Тема: Классификация (линейный дискриминантный анализ, метод опорных векторов)

Студент гр. 6307 _____ Новиков Б.М.

Преподаватель _____ Жангиров Т.Р.

2020

ЗАГРУЗКА ДАННЫХ

1-2. Загрузить данные в датафрейм
```
data = pd.read_csv('data/iris.data', header=None)
data.head()
```

	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

3. Выделить данные и их метки
```
X = data.iloc[:,:4].to_numpy()
labels = data.iloc[:,4].to_numpy()
```

4. Преобразовать тексты меток к числам
```
le = preprocessing.LabelEncoder()
Y = le.fit_transform(labels)
```

5. Разбить выборку на обучающую и тестовую
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.5)
```

ЛИНЕЙНЫЙ ДИСКРИМИНАНТНЫЙ АНАЛИЗ

1. Проведем классификацию методом LDA
```
clf = LinearDiscriminantAnalysis()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print((y_test != y_pred).sum())
```

Неправильно предсказаны: 4

Описать атрибуты и параметры классификатора

Параметр	Описание
solver	Решатель для использования
shrinkage	Параметр усадки
priors	Априорных вероятности классов
n_components	Количество компонентов (<= min (n_classes - 1, n_features)) для уменьшения размерности.
store_covariance	Если True, явно вычислить взвешенную ковариационную матрицу внутри класса, когда решатель - 'svd'. Матрица всегда вычисляется и сохраняется для других решателей.
tol	Абсолютный порог для того, чтобы единичное значение X считалось значимым, используется для оценки ранга X. Измерения, единичные значения которых не значимы, отбрасываются. Используется только если решатель - «svd».
covariance_estimator	Если не None, covariance_estimator используется для оценки ковариационных

	матриц вместо того, чтобы полагаться на эмпирическую оценку ковариации (с потенциальным сжатием). У объекта должен быть подходящий метод и атрибут covariance_, как у оценок в sklearn.covariance. если Нет, то оценка зависит от параметра усадки.

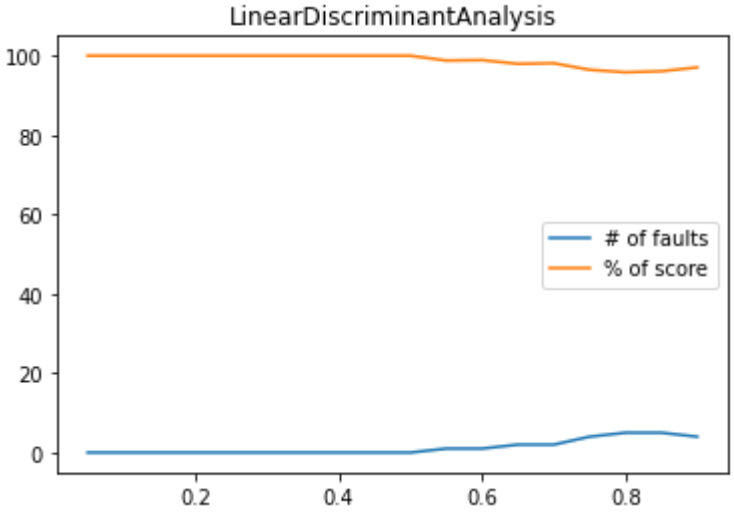
Атрибут	Описание
coef_	Вектор (ы) веса.
intercept_	Срок перехвата.
covariance_	Взвешенная матрица ковариаций внутри класса. Это соответствует sum_k prior_k * C_k, где C_k - ковариационная матрица выборок в классе k. C_k оцениваются с использованием (потенциально сокращенной) смещенной оценки ковариации. Если решатель - «svd», он существует только тогда, когда store_covariance имеет значение True.
explained_variance_ratio_	Процент отклонения, объясняемый каждым из выбранных компонентов. Если n_components не задано, то все компоненты сохраняются, а сумма объясненных отклонений равна 1,0. Доступно только при использовании собственного или svd-решателя.
means_	Классовые средние.
priors_	Вероятности класса (сумма 1).
scalings_	Масштабирование объектов в пространстве, охватываемом центроидами классов. Доступно только для решателей «svd» и «eigen».
xbar_	Общее среднее. Присутствует, только если решатель - «svd».
classes_	Уникальные метки классов.

2. Вывести точность классификации

```
clf.score(X_test, y_test)
```

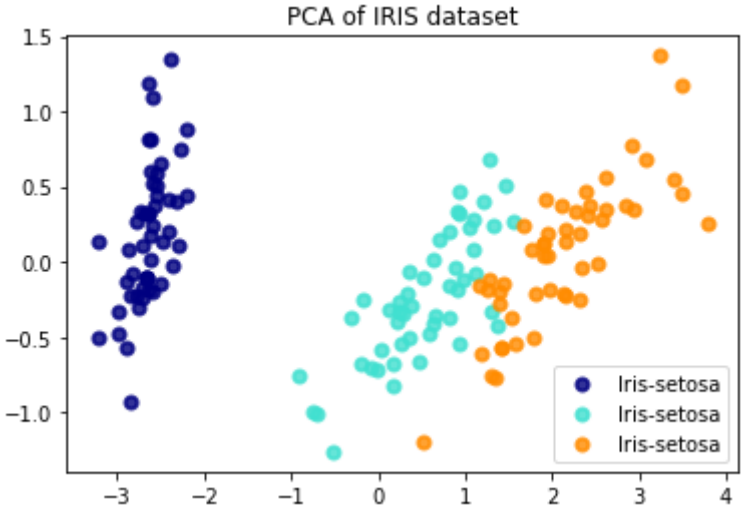
Точность: 0.9466666666666667

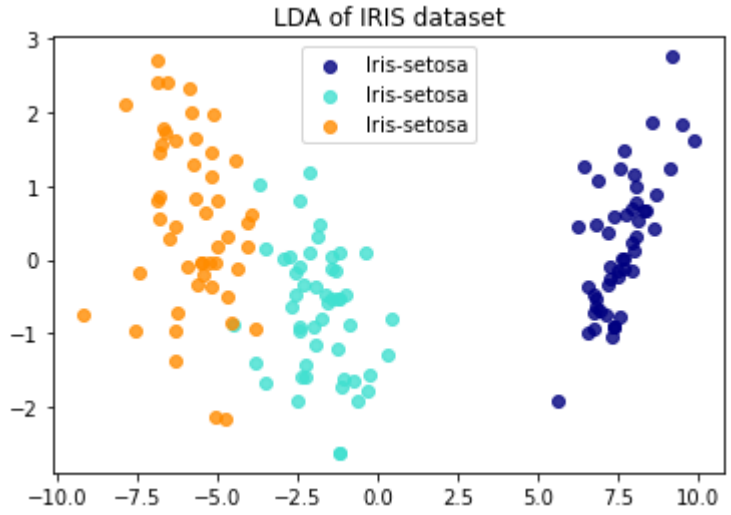
3. Построить график зависимости неправильно класс. наблюдений и точности классификации от размера выборки.



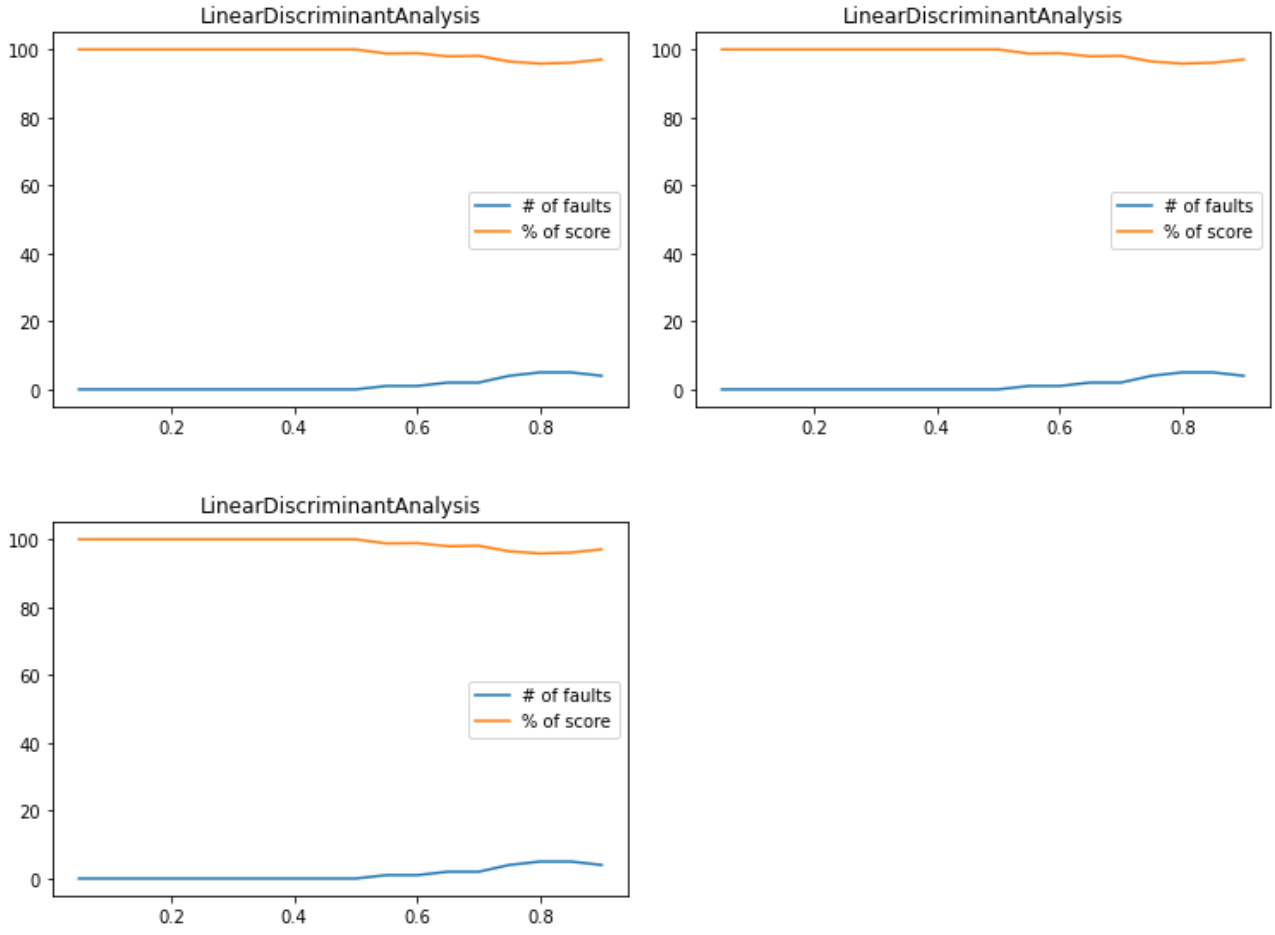
4. Для чего нужна функция transform? Визуализируйте ее результаты.
- Проецирует данные для максимизации разделения классов

Сравнение PCA и LDA





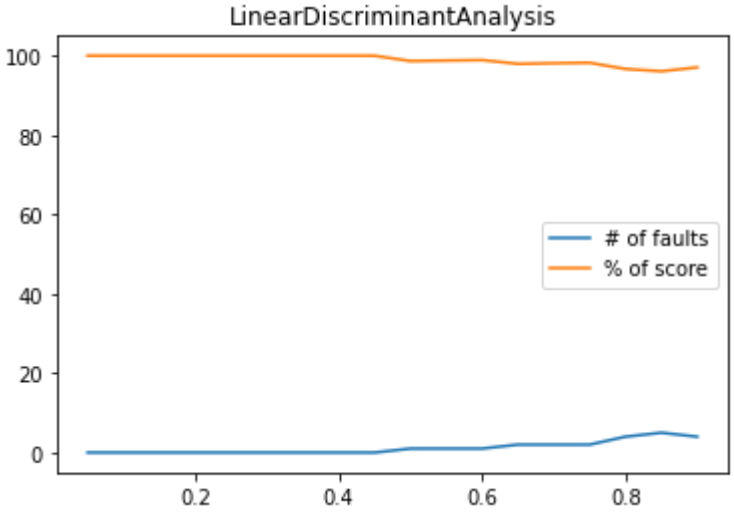
5. Исследуйте работу классификатора при параметрах solver, shrinkage.
`solver in ['svd', 'lsqr', 'eigen']`



6. Задать априорную вероятность классу с номером 1 — 0.7, остальным одинаковые. Какой результат?

```
priors = np.zeros(len(set(labels)))
priors[0] = 0.7
priors[1:] = 0.3 / (len(set(labels)) - 1)
priors
```

```
array([0.7 , 0.15, 0.15])
```


LinearDiscriminantAnalysis
100
80
60
40
20
0
0.2
0.4
0.6
0.8
# of faults
% of score

МЕТОД ОПОРНЫХ ВЕКТОРОВ

1. Классификация тех же данных методом SVM

```
clf = svm.SVC()
y_pred = clf.fit(X_train, y_train).predict(X_test)
print((y_test!= y_pred).sum())
print(clf.score(X, Y))
```

Неправильно угадано: 2

2. Точность классификации

```
clf.score(X_test, y_test)
```

Точность: 0.9733333333333334

3. Выведем следующую информацию

```
print(clf.support_vectors_)
print(clf.support_)
print(clf.n_support_)
```

```
Опорные векторы.
[[5.1 3.8 1.9 0.4]
 [5.1 3.3 1.7 0.5]
 [5.  3.5 1.6 0.6]
 [4.8 3.4 1.9 0.2]
 [5.4 3.4 1.5 0.4]
 [5.9 3.  4.2 1.5]
 [5.6 3.  4.1 1.3]
 [6.5 2.8 4.6 1.5]
 [6.1 2.9 4.7 1.4]
 [6.1 2.8 4.7 1.2]
 [6.3 3.3 4.7 1.6]
 [6.  2.9 4.5 1.5]
 [5.1 2.5 3.  1.1]
 [6.2 2.9 4.3 1.3]
 [5.2 2.7 3.9 1.4]
 [6.7 3.1 4.7 1.5]
 [5.5 2.4 3.7 1. ]
 [6.  3.4 4.5 1.6]
 [6.3 2.3 4.4 1.3]
 [6.6 2.9 4.6 1.3]
 [5.7 2.6 3.5 1. ]
 [6.3 2.5 4.9 1.5]
 [5.6 2.9 3.6 1.3]
 [6.7 3.  5.  1.7]
 [6.  2.7 5.1 1.6]
 [6.3 2.7 4.9 1.8]
 [5.8 2.7 5.1 1.9]
 [5.8 2.7 5.1 1.9]
 [5.9 3.  5.1 1.8]
 [6.5 3.2 5.1 2. ]
 [6.4 2.7 5.3 1.9]
 [7.7 3.8 6.7 2.2]
 [6.4 3.1 5.5 1.8]
 [6.2 2.8 4.8 1.8]
 [7.2 3.  5.8 1.6]
 [6.4 3.2 5.3 2.3]
 [6.9 3.1 5.4 2.1]
 [5.7 2.5 5.  2. ]
 [5.8 2.8 5.1 2.4]
 [6.9 3.1 5.1 2.3]
 [5.6 2.8 4.9 2. ]]
```

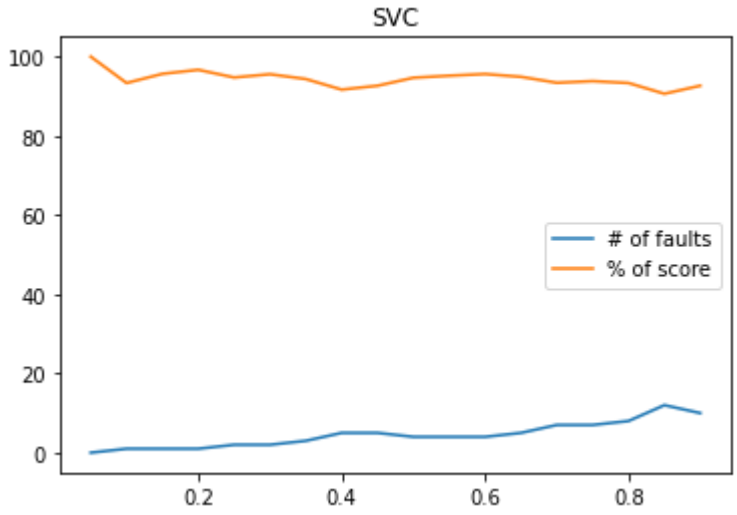
```
Индексы опорных векторов.
[14 22 48 51 68  4  6  8 15 17 25 26 27 28 34 40 41 47 50 58 60 61 63 67
 72  0  3  7  9 10 16 19 36 43 46 54 56 59 64 65 73]

Количество опорных векторов для каждого класса.
[ 5 20 16]
```

Объяснить

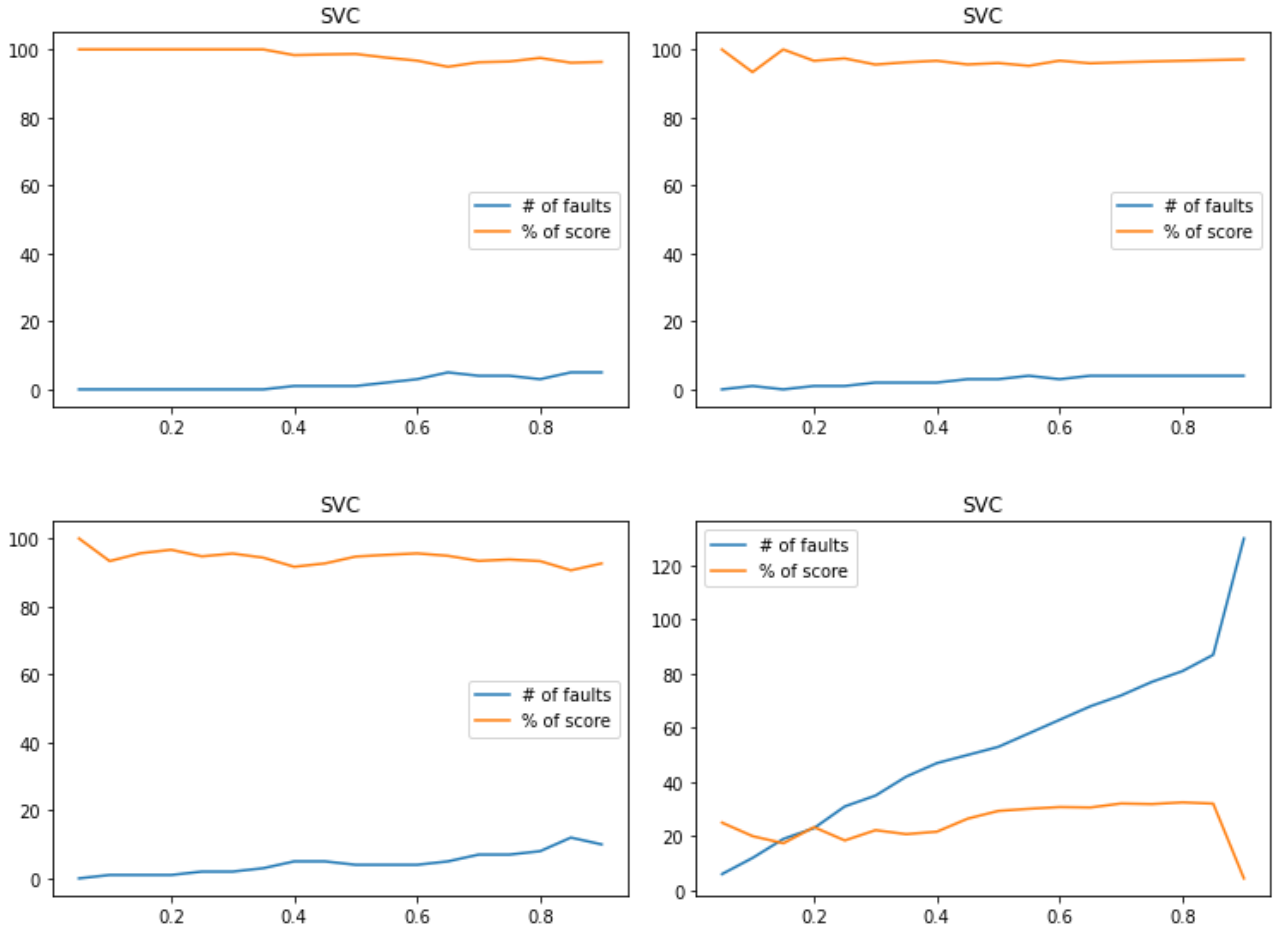
Машины опорных векторов считаются классификационным подходом, но его можно использовать как в задачах классификации, так и в задачах регрессии. Он может легко обрабатывать несколько непрерывных и категориальных переменных. SVM создает гиперплоскость в многомерном пространстве для разделения различных классов. SVM генерирует оптимальную гиперплоскость итеративным способом, который используется для минимизации ошибки. Основная идея SVM - найти максимальную маргинальную гиперплоскость (ММН), которая наилучшим образом разделяет набор данных на классы.

4. Построить график зависимости неправильно класс. наблюдений и точности классификации от размера выборки.



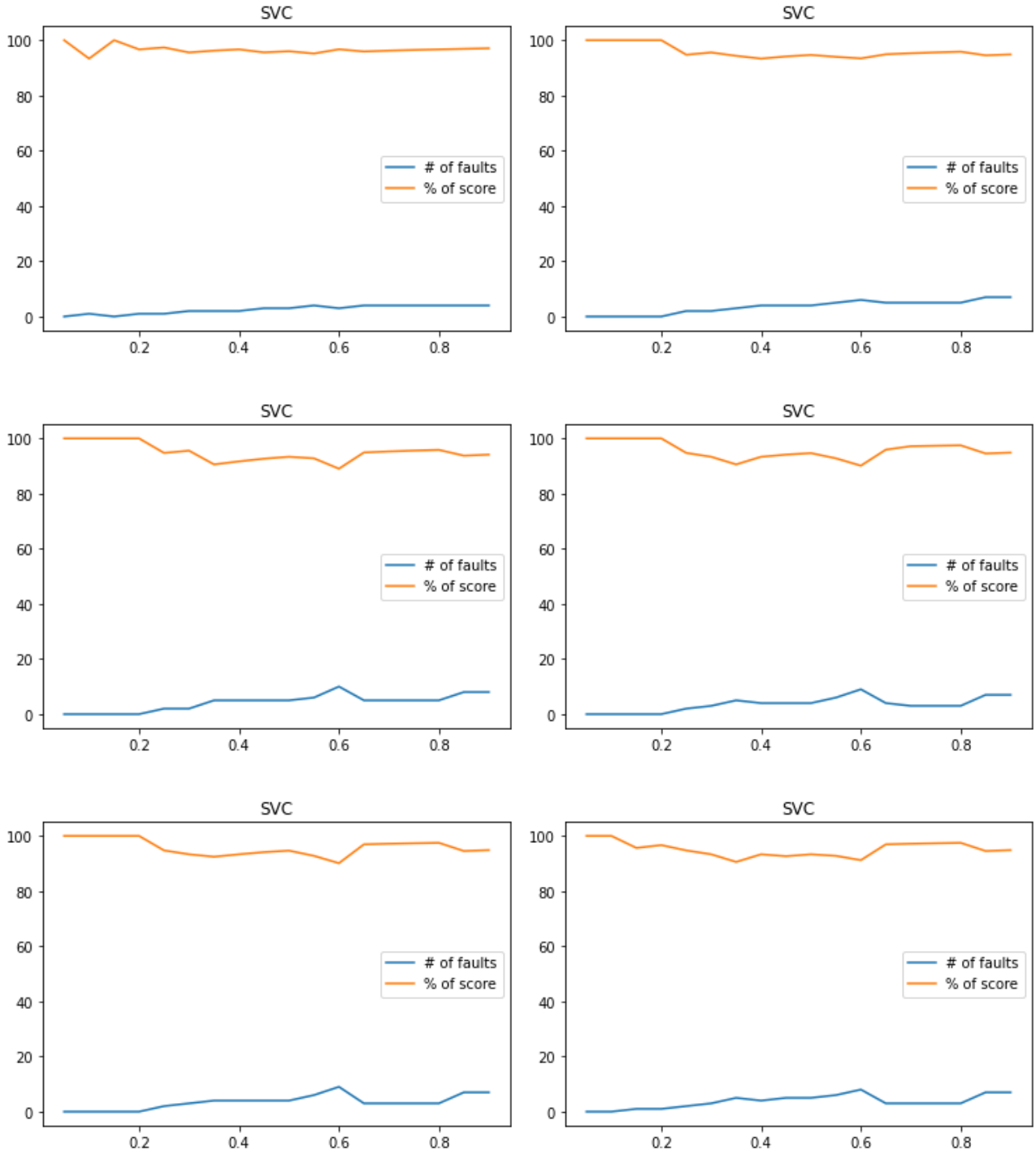
5. Исследовать работу при различных значениях kernel, degree, max_iter

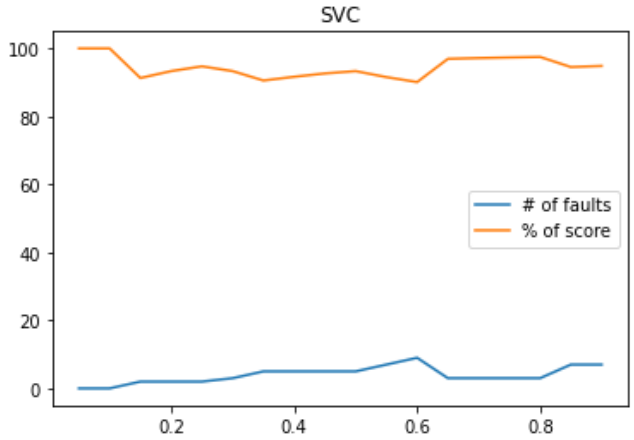
`kernel in ['linear', 'poly', 'rbf', 'sigmoid']`
`- Задает тип ядра, который будет использоваться в алгоритме. Это должно быть одно из "linear", "poly", "rbf", "sigmoid", "precomputed" или вызываемое. Если ничего не указано, будет использоваться «rbf». Если задан вызываемый объект, он используется для предварительного вычисления матрицы ядра из матриц данных; эта матрица должна быть массивом формы (n_samples, n_samples).`



`degree in range(3, 10)`

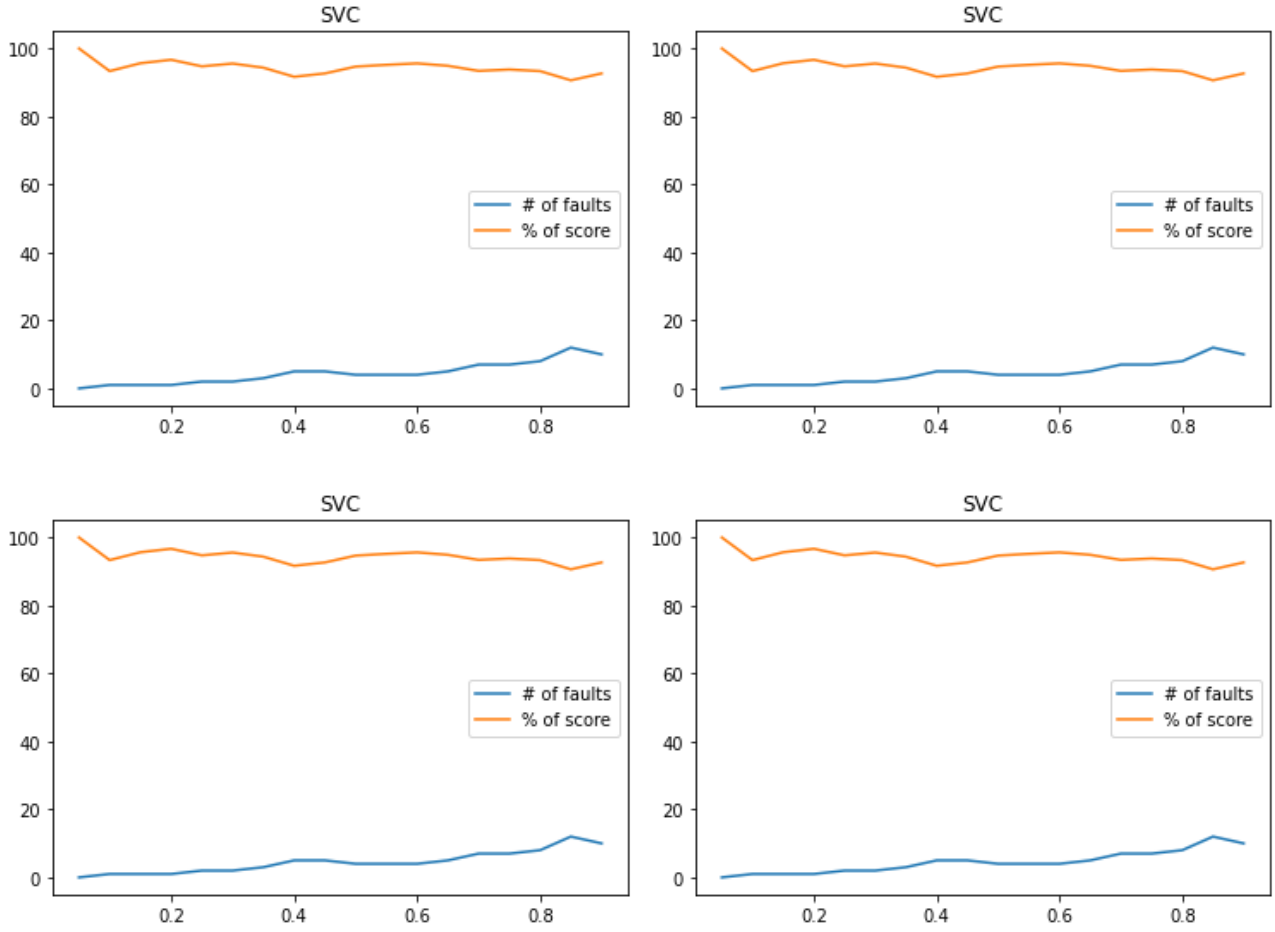
- Степень полиномиальной ядерной функции («поли»). Игнорируется всеми остальными ядрами.




SVC
100
80
60
40
20
0
0.2
0.4
0.6
0.8
# of faults
% of score

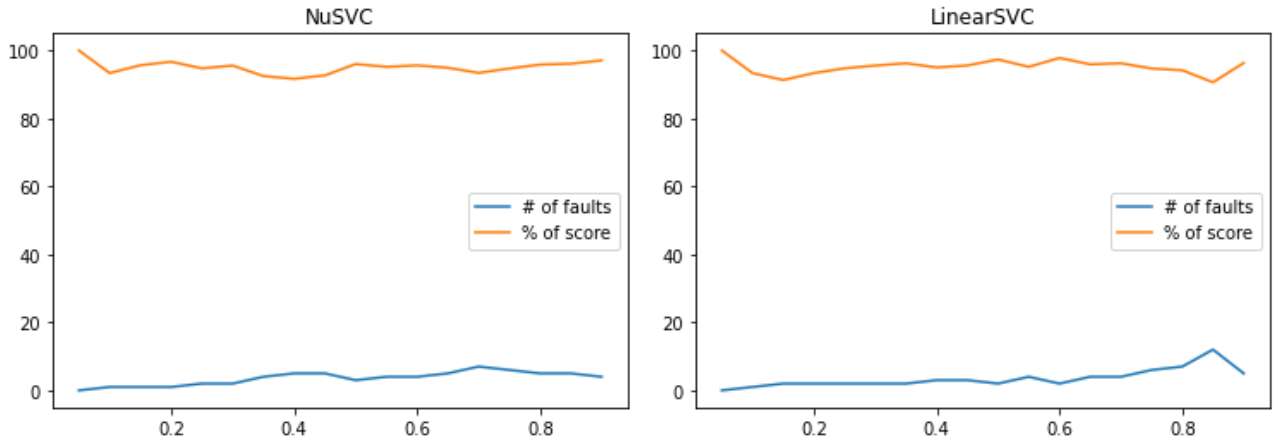
max_iter in [-1, 40, 200, 300]

- Жесткое ограничение на количество итераций в решателе или -1 без ограничения.



6. Исследовать NuSVC и LinearSVC

```
svc in [svm.NuSVC, svm.LinearSVC]
```



В чем отличие?

Метод	Отличие
NuSVC	Подобен SVC, но использует параметр для управления количеством опорных векторов.
LinearSVC	Подобен SVC с параметром kernel = ’linear’, но реализован в терминах liblinear, а не libsvm, поэтому он имеет большую гибкость в выборе функций штрафов и потерь и должен лучше масштабироваться для большого количества выборок.