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 _____ Золотухин М. А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2020

Линейный дискриминантный анализ¶

Проведем классификацию наблюдений используя LDA

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.5,
random_state=0)

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

clf = LinearDiscriminantAnalysis()
y_pred = clf.fit(X_train, y_train).predict(X_test)
print((y_test != y_pred).sum()) #количество наблюдений, который были неправильно
определены
3
```

Параметры классификатора:

- solver — метод решения
- shrinkage — параметр усадки
- priors — априорные вероятности
- n_components — число компонент для уменьшения размерности
- store_covariance — явно вычислить взвешенную ковариационную матрицу внутри класса при solver=svd
- tol — абсолютный порог для того, чтобы сингулярное число X считалось значимым
- covariance_estimator — используется для оценки covariance_estimator ковариационных матриц вместо того, чтобы полагаться на эмпирическую оценку ковариации

Атрибуты:

- coef_ — весовые вектора.
- intercept_ — intercept term.
- covariance_ — взвешивая внутри класса ковариационная матрица.
- means_ — по-классовые средние.
- priors_ — априорные вероятности классов (в сумме равны 1).

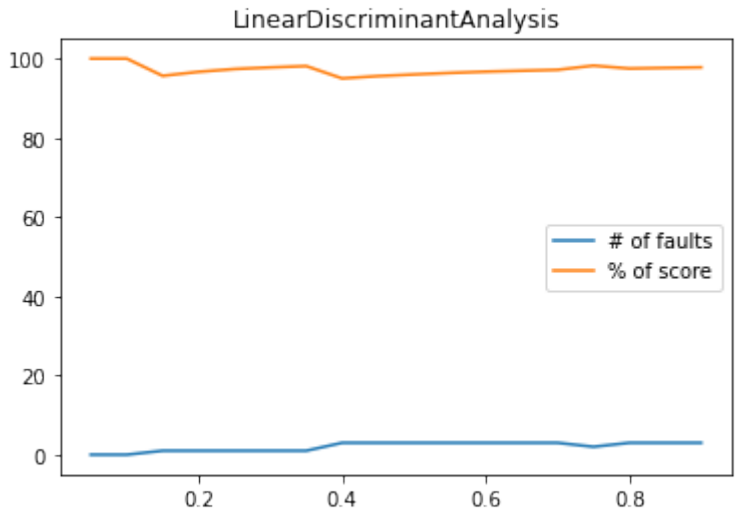
- scalings_ — масштабирование фич в пространстве составленном из центроидов классов. (Только для svd и eingen решателей)
- xbar_ — общее среднее (если решатель — svd).
- classes_ — лэйблы классов.

Используя функцию score() выведем точность классификации

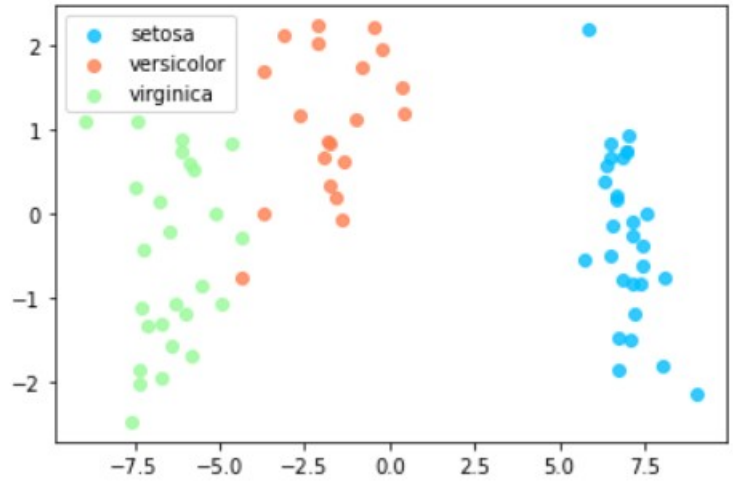
```
clf.score(X_test, y_test)
```

```
0.96
```

Постройте график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки. Размер тестовой выборки изменяйте от 0.05 до 0.95 с шагом 0.05. Параметр random_state сделайте равным номеру своей зачетной книжки. Обоснуйте полученные результаты.

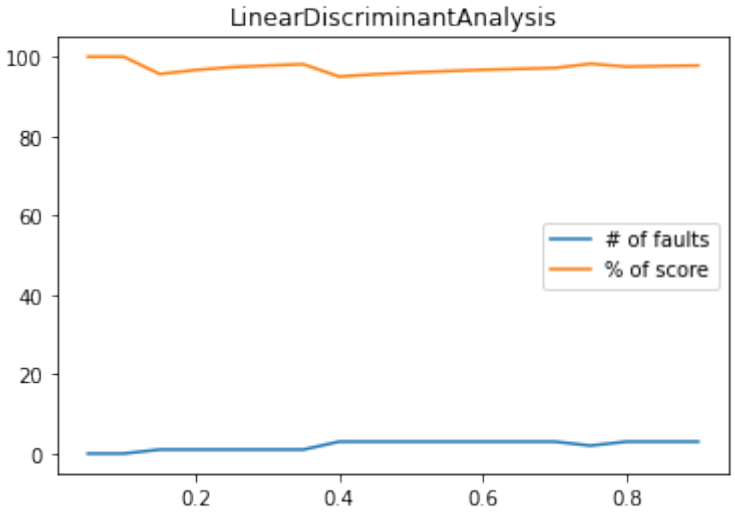


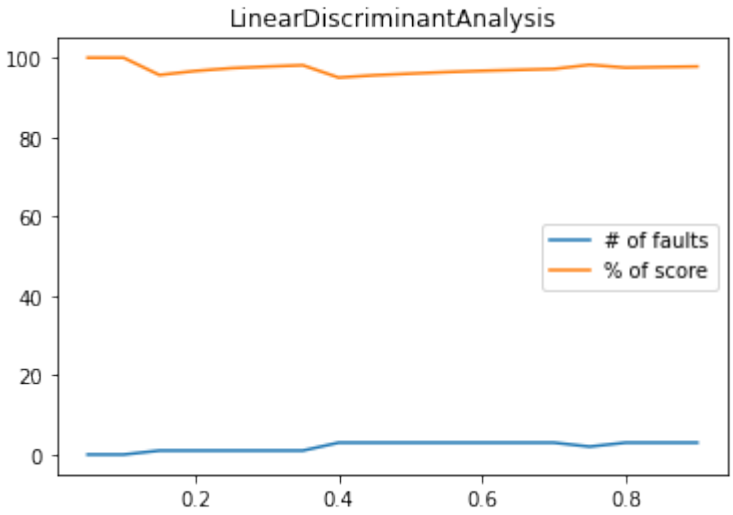
Функция transform позволяет спроецировать данные с увеличенной дисперсией.

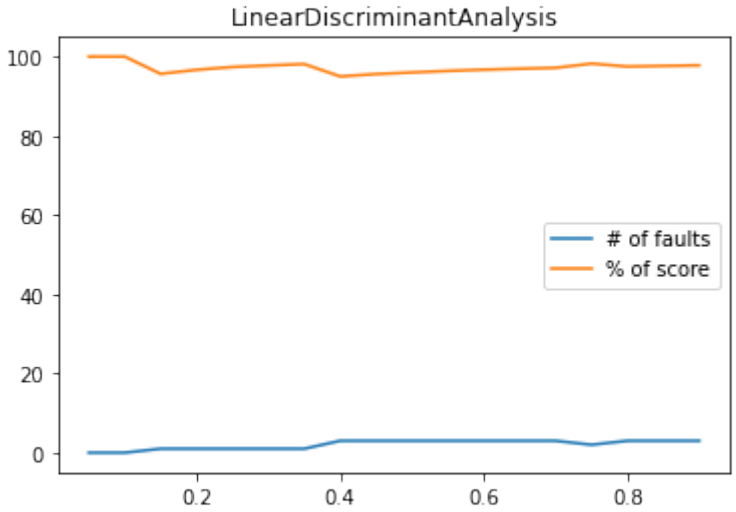


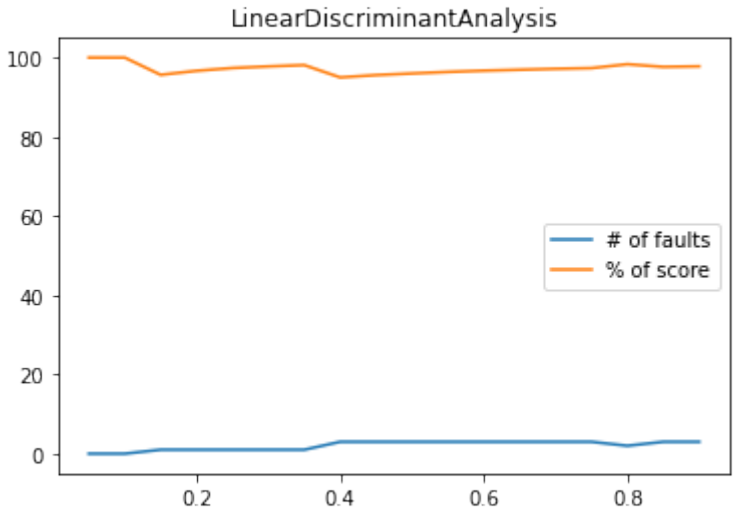
Исследуйте работу классификатор при различных параметрах solver, shrinkage.

```
for solver in ['svd', 'lsqr', 'eigen']:
    test_results = estimate_clf(LinearDiscriminantAnalysis(solver=solver))
    show_result(LinearDiscriminantAnalysis, test_results)
```

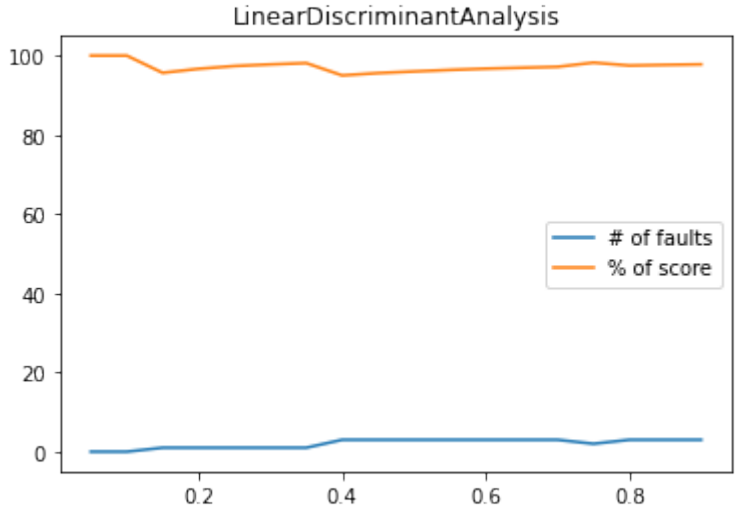



LinearDiscriminantAnalysis
100
80
60
40
20
0
0.2
0.4
0.6
0.8
# of faults
% of score

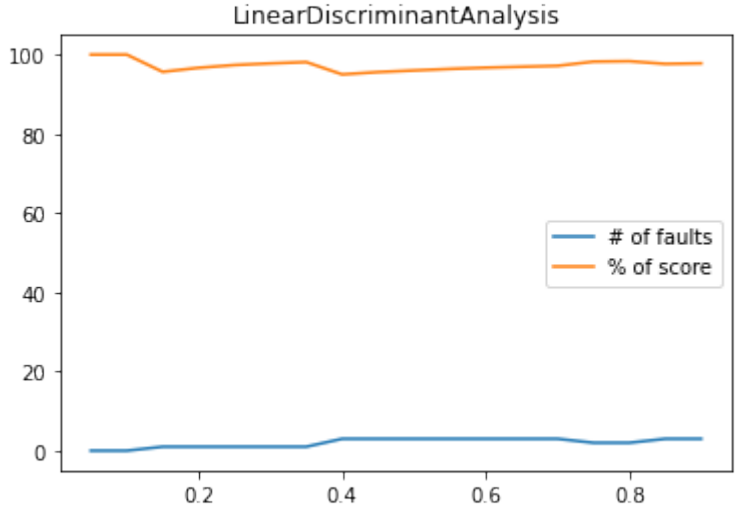

LinearDiscriminantAnalysis
100
80
60
40
20
0
0.2
0.4
0.6
0.8
# of faults
% of score


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

shrinkage



Зададим априорную вероятность классу с номером 1 равную 0.7, остальным классам зададим равные априорные вероятности. Как это сказалось на результате?



На результате это не отразилось.

Метод опорных векторов

Классификацию при SVM на тех же данных

```
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))
4
0.9533333333333334
```

Используя функцию score() выведите точность классификации

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

```
0.9466666666666667
```

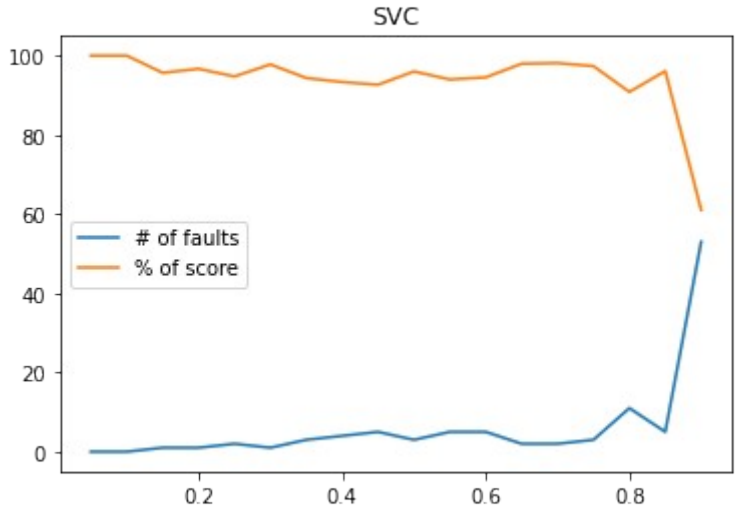
Выведите следующую информацию

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

- support_vectors_ — опорные вектора
- support_ — индексы векторов
- n_support_ — количество опорных векторов для каждого класса

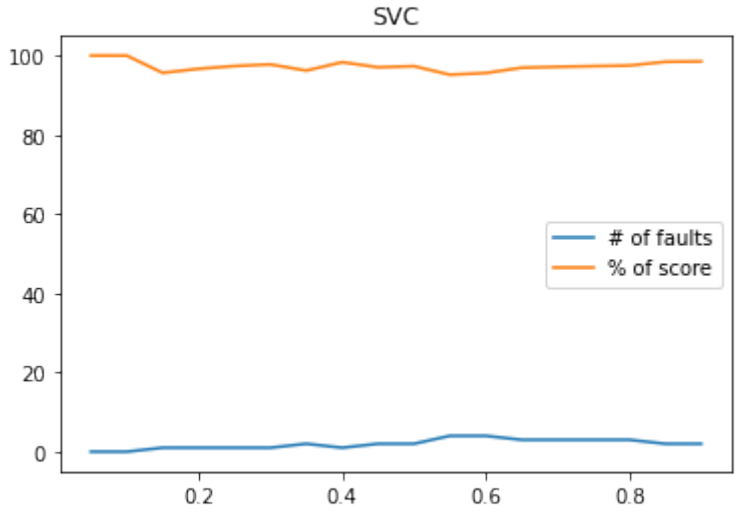
Постройте график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки. Размер тестовой выборки изменяйте от 0.05 до 0.95 с шагом 0.05. Параметр random_state сделайте равным номеру своей зачетной книжки. Обоснуйте полученные результаты.

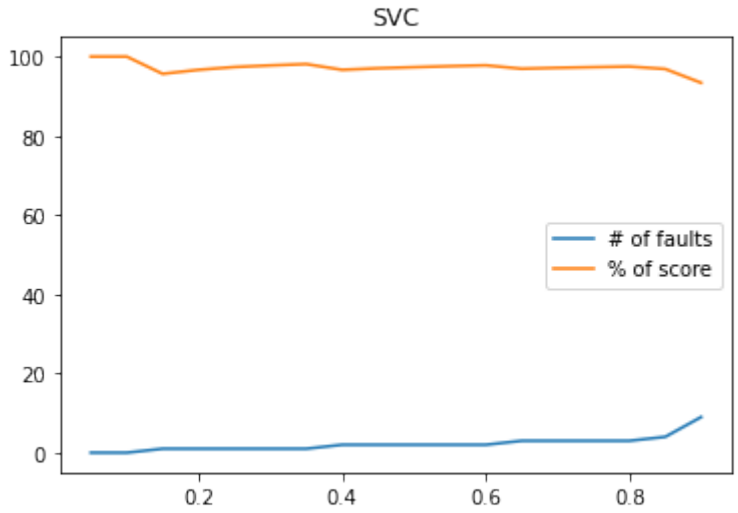
```
test_results = estimate_clf(svm.SVC())
show_result(svm.SVC, test_results)
```

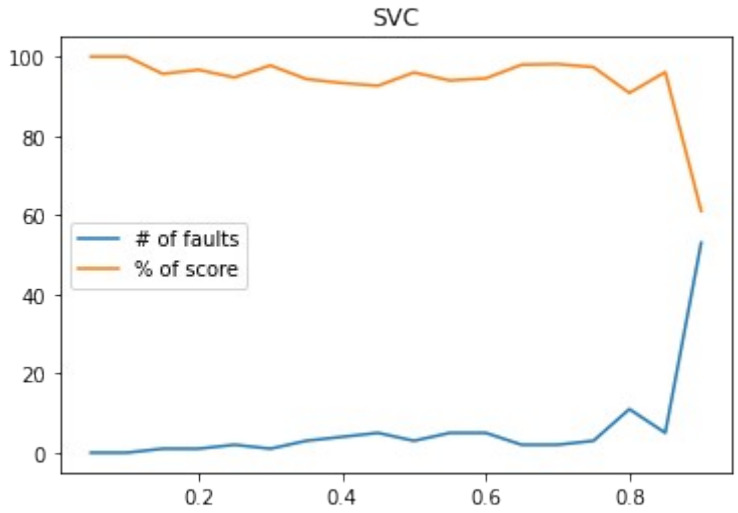


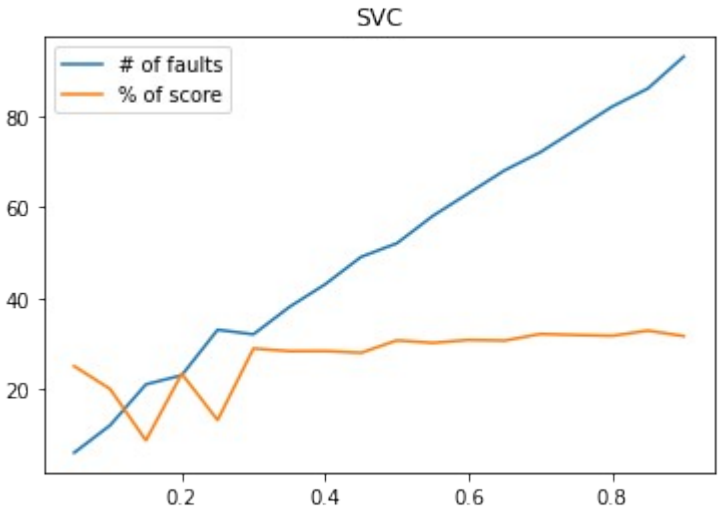
Исследуйте работу метода опорных векторов при различных значениях kernel, degree, max_iter

```
for kernel in ['linear', 'poly', 'rbf', 'sigmoid']:
    test_results = estimate_clf(svm.SVC(kernel=kernel))
    show_result(svm.SVC, test_results)
```



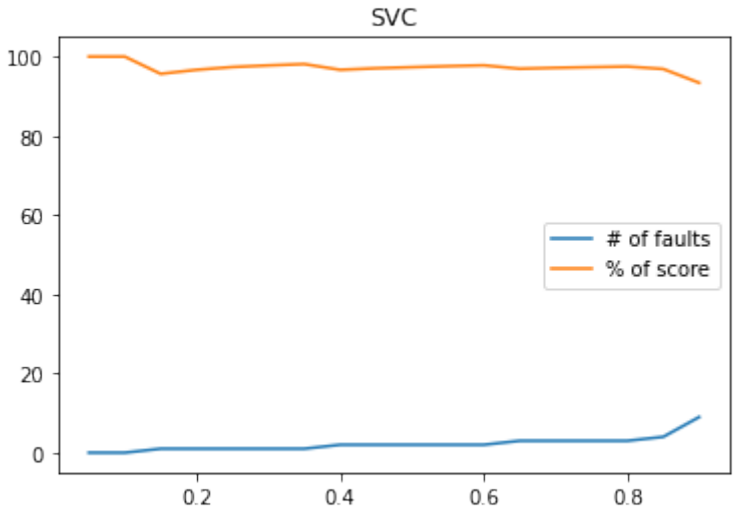


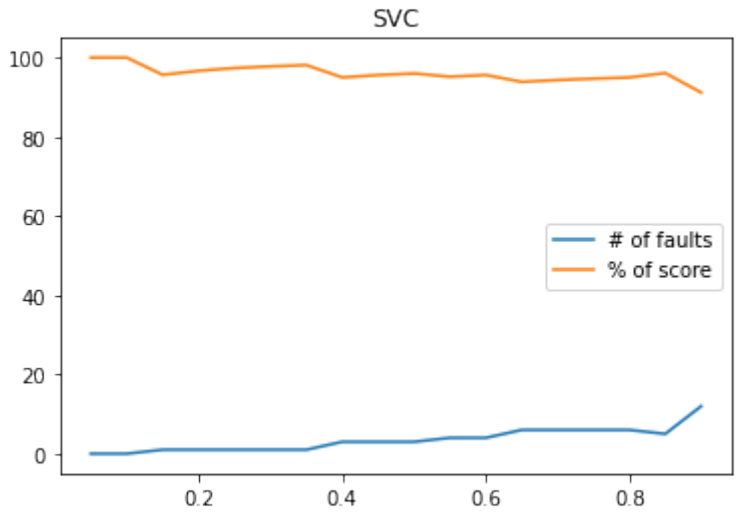


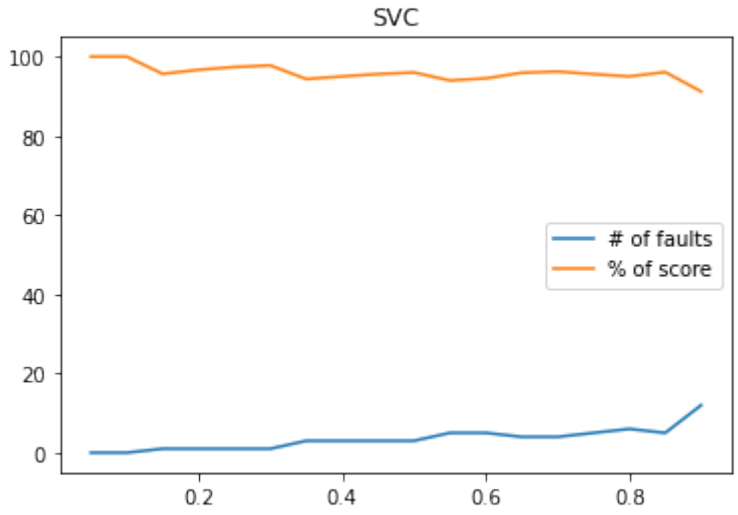


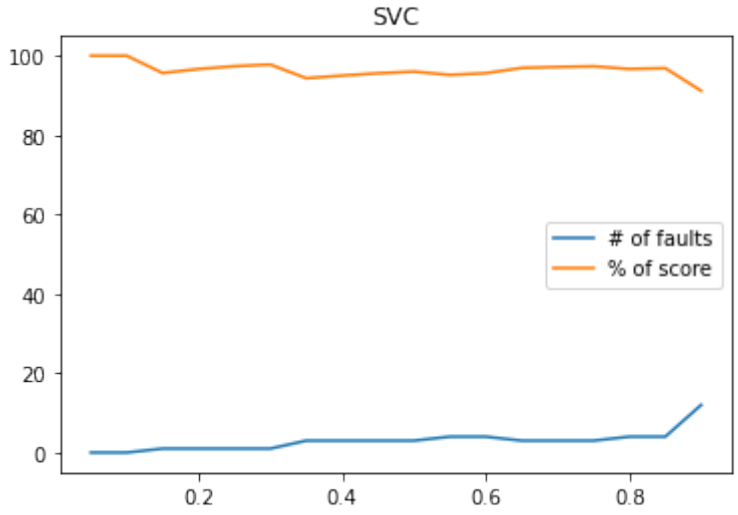
In [61]:

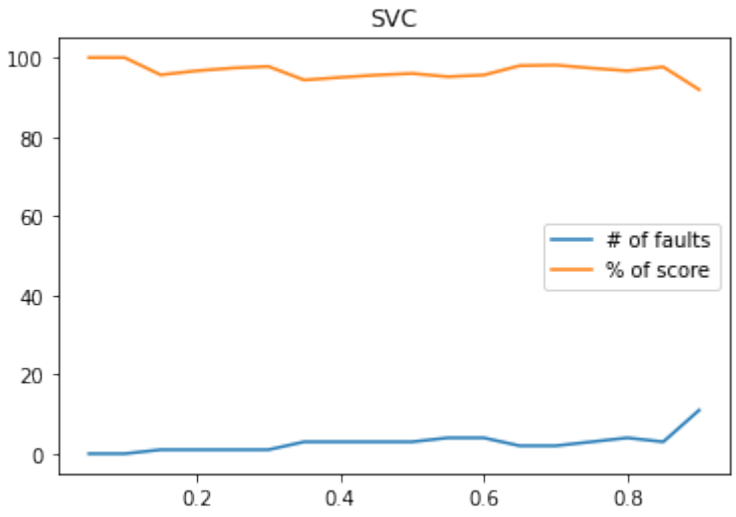
```
for degree in range(3, 10):
    test_results = estimate_clf(svm.SVC(kernel='poly', degree=degree))
    show_result(svm.SVC, test_results)
```

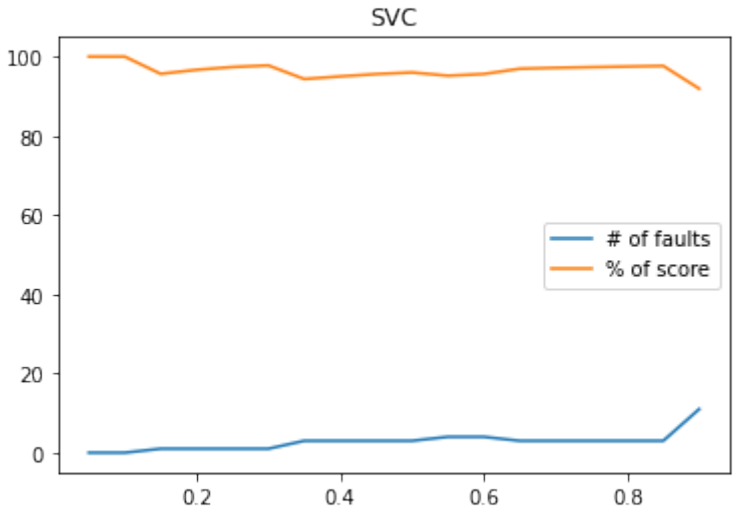


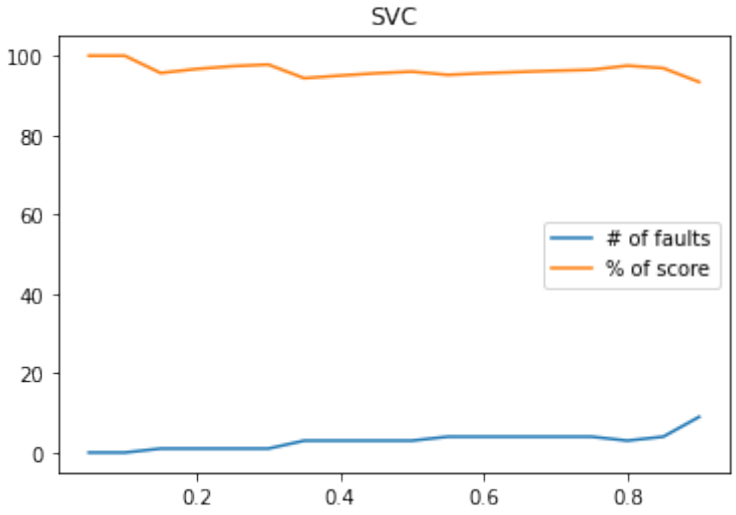

SVC
100
80
60
40
20
0
0.2
0.4
0.6
0.8
# of faults
% of score


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


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

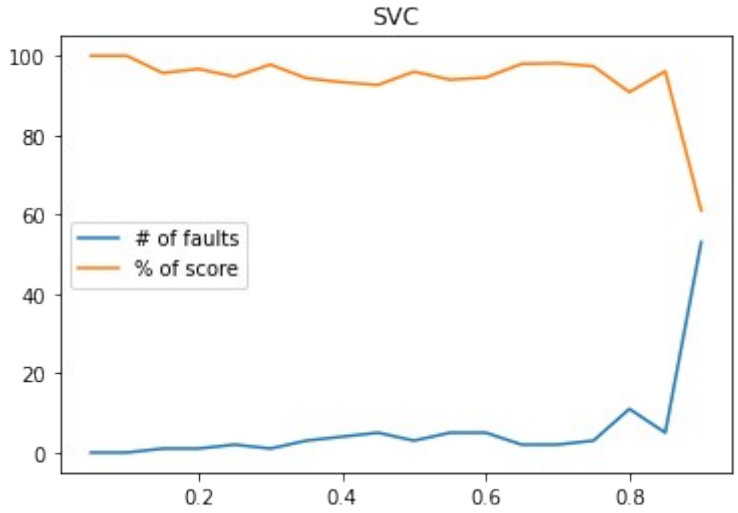

SVC
100
80
60
40
20
0
0.2
0.4
0.6
0.8
# of faults
% of score

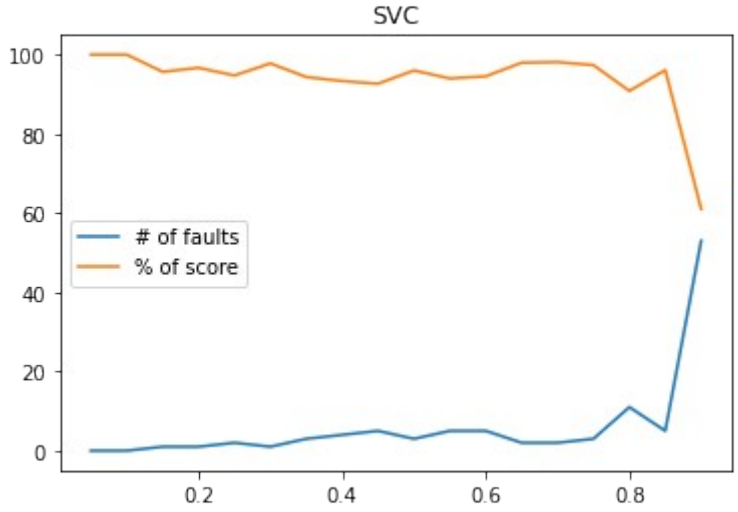

SVC
100
80
60
40
20
0
0.2
0.4
0.6
0.8
# of faults
% of score

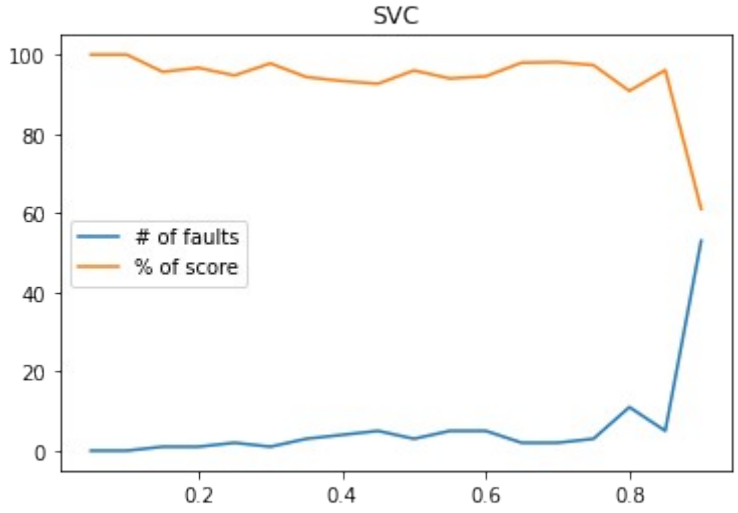

SVC
100
80
60
40
20
0
0.2
0.4
0.6
0.8
# of faults
% of score

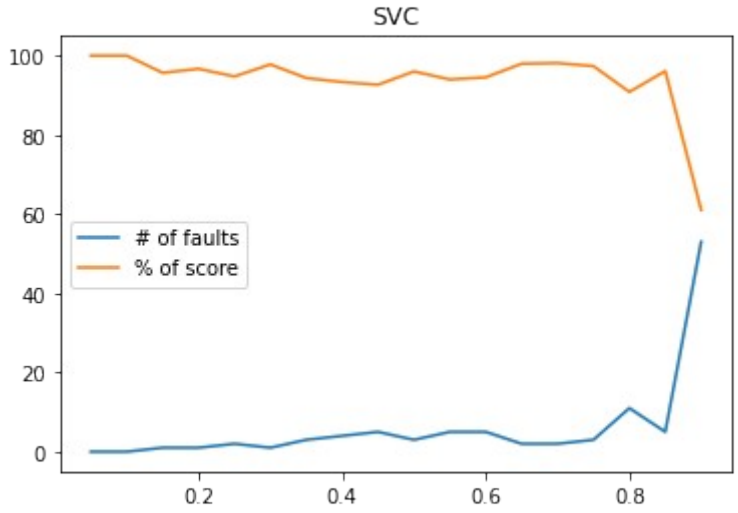
In [62]:

```
for max_iter in [-1, 40, 200, 300]:
    test_results = estimate_clf(svm.SVC(max_iter=max_iter))
    show_result(svm.SVC, test_results)
```







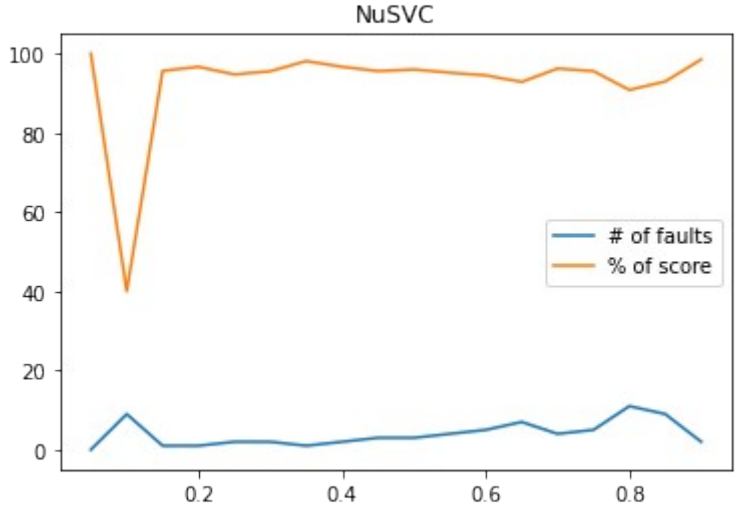


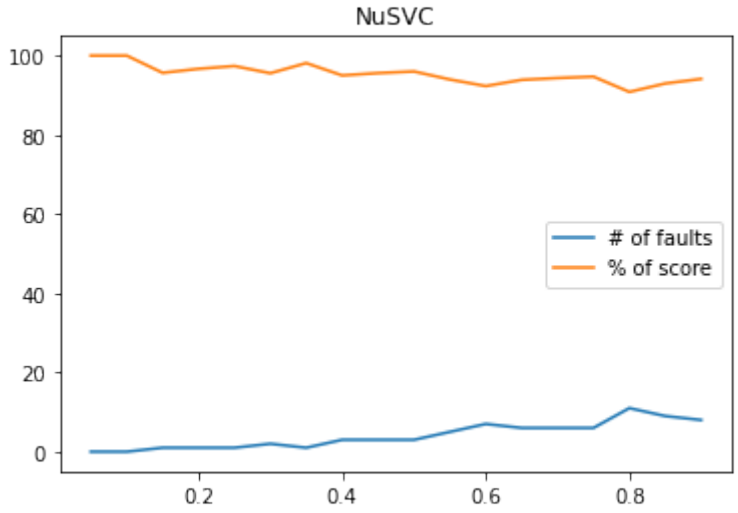
Проведите исследование для методов NuSVC и LinearSVC. В чем их отличие от SVC

In [72]:

```
test_results = estimate_clf(svm.NuSVC(nu=0.01))
show_result(svc, test_results)

test_results = estimate_clf(svm.LinearSVC())
show_result(svc, test_results)
```





LinearSVC — это SVC с линейным ядром, но под капотом имеющий другую реализацию. NuSVC — тот же SVC, но имеет параметр для количества поддерживающих векторов.