**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №8**

**по дисциплине «Машинное обучение»**

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

|  |  |  |
| --- | --- | --- |
| Студент гр. 8304 |  | Кирьянов Д.И. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

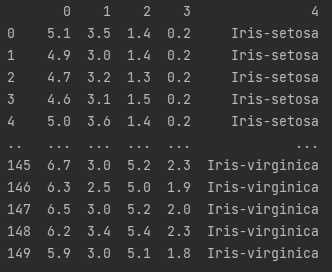
2021

**Цель работы.**

Ознакомиться с методами классификации модуля Sklearn

**Ход работы.**

1. **Загрузка данных**
   1. Произведена загрузка данных. Создан Python скрипт. Данные загружены в датафрейм.



* 1. Выделены данные и их метки, преобразованы тексты меток к числам, разбита выборка на обучающую и тестовую.

**2. Линейный дискриминантный анализ**

2.1. Проведена классификация данных при помощи LDA. Количество найденных неправильно классифицированных наблюдений = 3.

Атрибуты:

coef\_ - векторы веса.

intercept\_ - срок перехвата.

covariance\_ - взвешенная внутриклассовая матрица ковариаций.

explained\_variance\_ratio\_ - процент отклонения для каждого выбранного компонента.

means\_ - классовые средние.

priors\_ - приоры класса.

scalings\_ - масштабирование объектов в пространстве, охватываемом центрами классов.

xbar\_ - общее среднее.

classes\_ - уникальные лейблы класса.

n\_features\_in\_ - количество видимых деталей во время посадки.

feature\_names\_in\_ - названия видимых особенностей во время посадки.

*Attributes  
----------  
coef\_ : ndarray of shape (n\_features,) or (n\_classes, n\_features)  
 Weight vector(s).  
  
intercept\_ : ndarray of shape (n\_classes,)  
 Intercept term.  
  
covariance\_ : array-like of shape (n\_features, n\_features)  
 Weighted within-class covariance matrix. It corresponds to  
 `sum\_k prior\_k \* C\_k` where `C\_k` is the covariance matrix of the  
 samples in class `k`. The `C\_k` are estimated using the (potentially  
 shrunk) biased estimator of covariance. If solver is 'svd', only  
 exists when `store\_covariance` is True.  
  
explained\_variance\_ratio\_ : ndarray of shape (n\_components,)  
 Percentage of variance explained by each of the selected components.  
 If ``n\_components`` is not set then all components are stored and the  
 sum of explained variances is equal to 1.0. Only available when eigen  
 or svd solver is used.  
  
means\_ : array-like of shape (n\_classes, n\_features)  
 Class-wise means.  
  
priors\_ : array-like of shape (n\_classes,)  
 Class priors (sum to 1).  
  
scalings\_ : array-like of shape (rank, n\_classes - 1)  
 Scaling of the features in the space spanned by the class centroids.  
 Only available for 'svd' and 'eigen' solvers.  
  
xbar\_ : array-like of shape (n\_features,)  
 Overall mean. Only present if solver is 'svd'.  
  
classes\_ : array-like of shape (n\_classes,)  
 Unique class labels.  
  
n\_features\_in\_ : int  
 Number of features seen during :term:`fit`.  
  
 .. versionadded:: 0.24  
  
feature\_names\_in\_ : ndarray of shape (`n\_features\_in\_`,)  
 Names of features seen during :term:`fit`. Defined only when `X`  
 has feature names that are all strings.*

Параметры:

solver – используемый метод решения.

shrinkage – параметр усадки.

priors – класс априорных вероятностей.

n\_components – количество компонентов для уменьшения размерности.

store\_covariance – флаг для вычисления взвешенной ковариационной матрицы внутри класса.

tol – абсолютный порог, чтобы единичное значение X считалось значимым, используется для оценки ранга X.

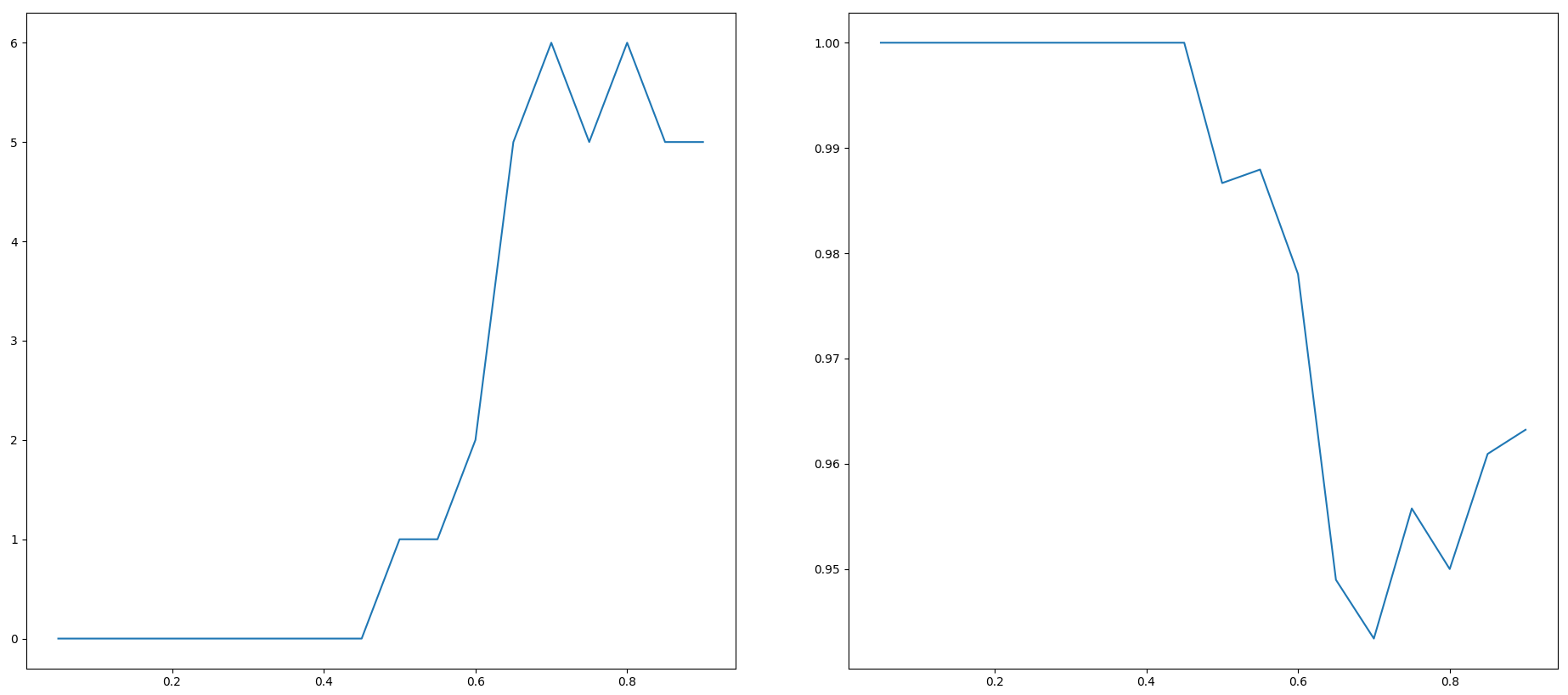
covariance\_estimator – используется для оценки ковариационных матриц вместо эмпирической оценки ковариации.

*solver : {'svd', 'lsqr', 'eigen'}, default='svd'  
 Solver to use, possible values:  
 - 'svd': Singular value decomposition (default).  
 Does not compute the covariance matrix, therefore this solver is  
 recommended for data with a large number of features.  
 - 'lsqr': Least squares solution.  
 Can be combined with shrinkage or custom covariance estimator.  
 - 'eigen': Eigenvalue decomposition.  
 Can be combined with shrinkage or custom covariance estimator.  
  
shrinkage : 'auto' or float, default=None  
 Shrinkage parameter, possible values:  
 - None: no shrinkage (default).  
 - 'auto': automatic shrinkage using the Ledoit-Wolf lemma.  
 - float between 0 and 1: fixed shrinkage parameter.  
  
 This should be left to None if `covariance\_estimator` is used.  
 Note that shrinkage works only with 'lsqr' and 'eigen' solvers.  
  
priors : array-like of shape (n\_classes,), default=None  
 The class prior probabilities. By default, the class proportions are  
 inferred from the training data.  
  
n\_components : int, default=None  
 Number of components (<= min(n\_classes - 1, n\_features)) for  
 dimensionality reduction. If None, will be set to  
 min(n\_classes - 1, n\_features). This parameter only affects the  
 `transform` method.  
  
store\_covariance : bool, default=False  
 If True, explicitly compute the weighted within-class covariance  
 matrix when solver is 'svd'. The matrix is always computed  
 and stored for the other solvers.  
  
 .. versionadded:: 0.17  
  
tol : float, default=1.0e-4  
 Absolute threshold for a singular value of X to be considered  
 significant, used to estimate the rank of X. Dimensions whose  
 singular values are non-significant are discarded. Only used if  
 solver is 'svd'.  
  
 .. versionadded:: 0.17  
  
covariance\_estimator : covariance estimator, default=None  
 If not None, `covariance\_estimator` is used to estimate  
 the covariance matrices instead of relying on the empirical  
 covariance estimator (with potential shrinkage).  
 The object should have a fit method and a ``covariance\_`` attribute  
 like the estimators in :mod:`sklearn.covariance`.  
 if None the shrinkage parameter drives the estimate.*

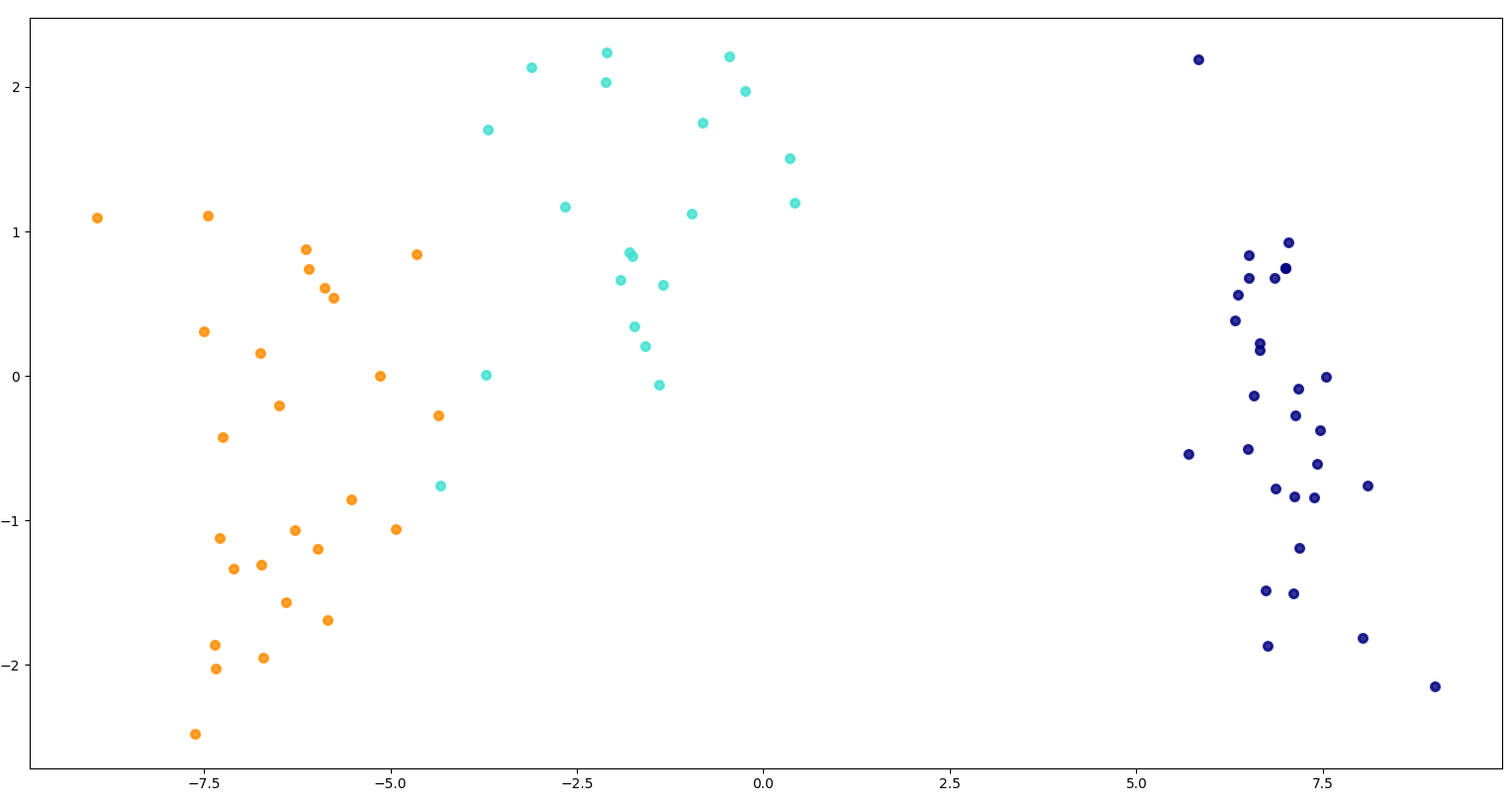
2.2. Используя функцию score() выведена точность классификации

Score = 0.9866666666666667

2.3. Построен график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки.



2.4. Применена функция transform и визуализированы ее результаты.



Функция transform применяется для уменьшения размерности данных.

2.5. Исследована работа классификатора при различных параметрах solver, shrinkage.

solver:

svd - разложение по сингулярным значениям (по умолчанию). Не вычисляет ковариационную матрицу, поэтому этот метод решения рекомендуется для данных с большим количеством функций.

lsqr - решение методом наименьших квадратов. Можно комбинировать с оценкой усадки или настраиваемой ковариационной оценкой.

eigen - разложение по собственным значениям. Можно комбинировать с оценкой усадки или настраиваемой ковариационной оценкой.

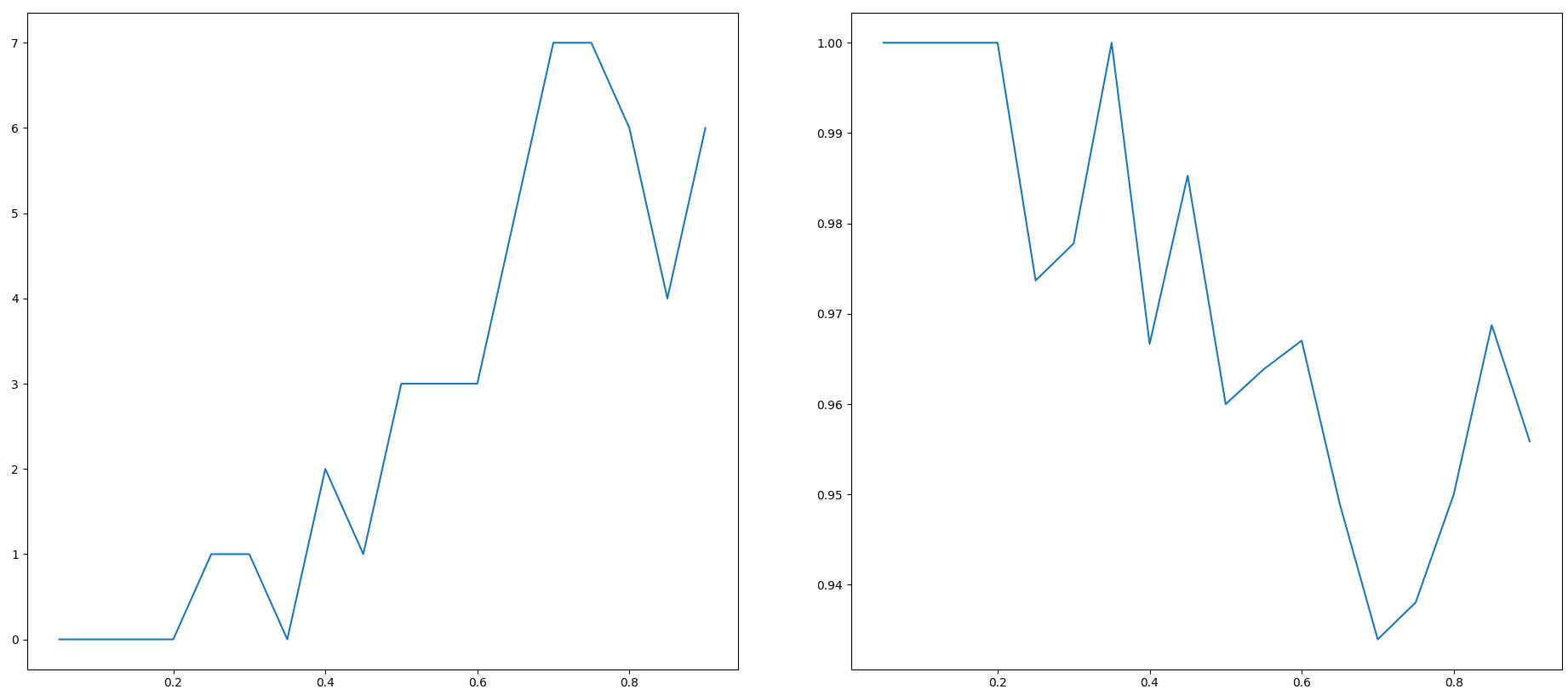
shrinkage:

None – без усадки. (по умолчанию)

auto - автоматическая усадка с использованием леммы Ледуа-Вольфа.

float – значение между 0 и 1. фиксированный параметр усадки.

2.6. Задана априорную вероятность класса с номером 1 равная 0.7, остальным классам заданы равные априорные вероятности.



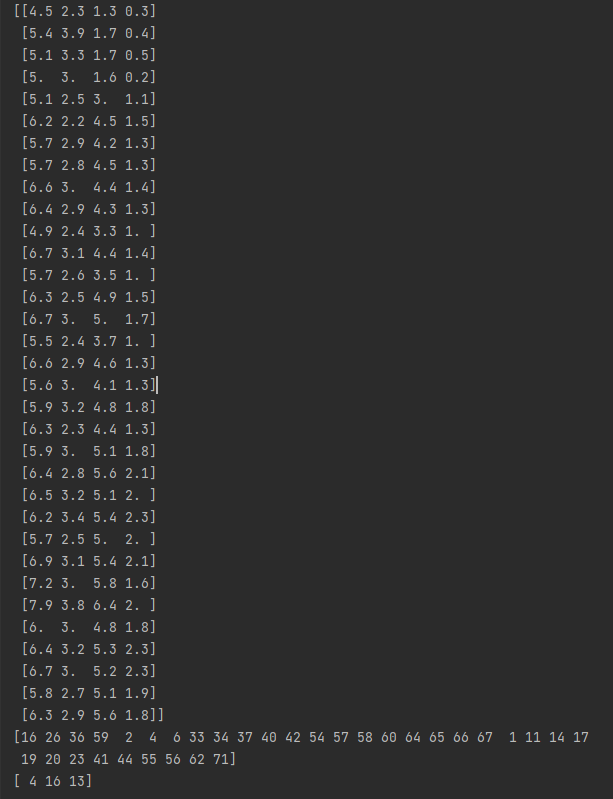
**3. Метод опорных векторов**

3.1. Проведена классификация при помощи SVM на тех же данных. Количество найденных неправильно классифицированных наблюдений = 4.

3.2. Используя функцию score() выведена точность классификации

Score = 0.9533333333333334

3.3. Выведена следующая информация.



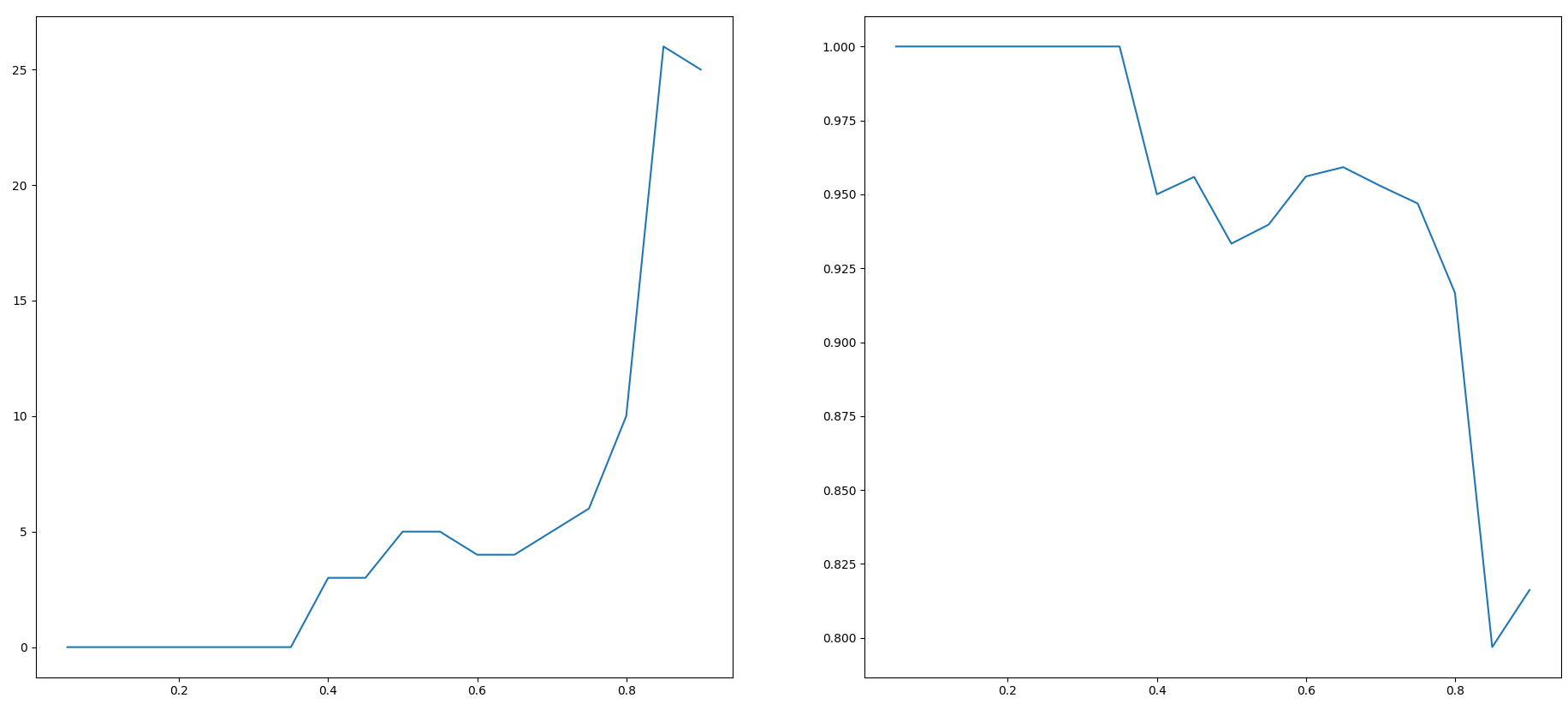
Это информация об опорных векторах.

support\_ хранит индексы опорных векторов.

support\_vectors\_ хранит сами опорные вектора.

n\_support\_ хранит количество опорных векторов для каждого класса.

3.4. Построен график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки.



3.5. Исследована работа метода опорных векторов при различных значениях kernel, degree, max\_iter.

kernel – тип ядра, который будет использоваться внутри алгоритма.

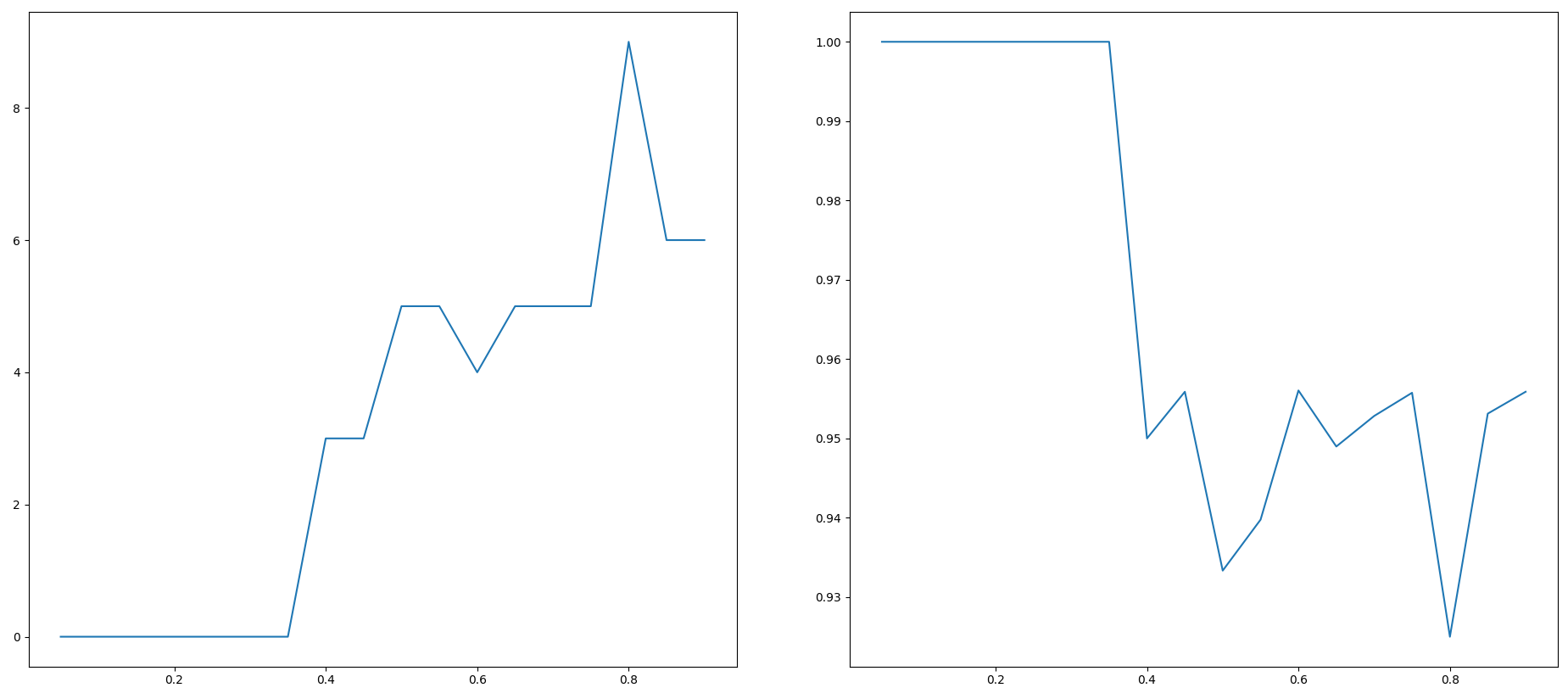
kernel: {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}

degree – степень полиномиальной функции ядра. Только при kernel = 'poly'.

max\_iter – ограничение на количество итераций. При -1 неограниченно.

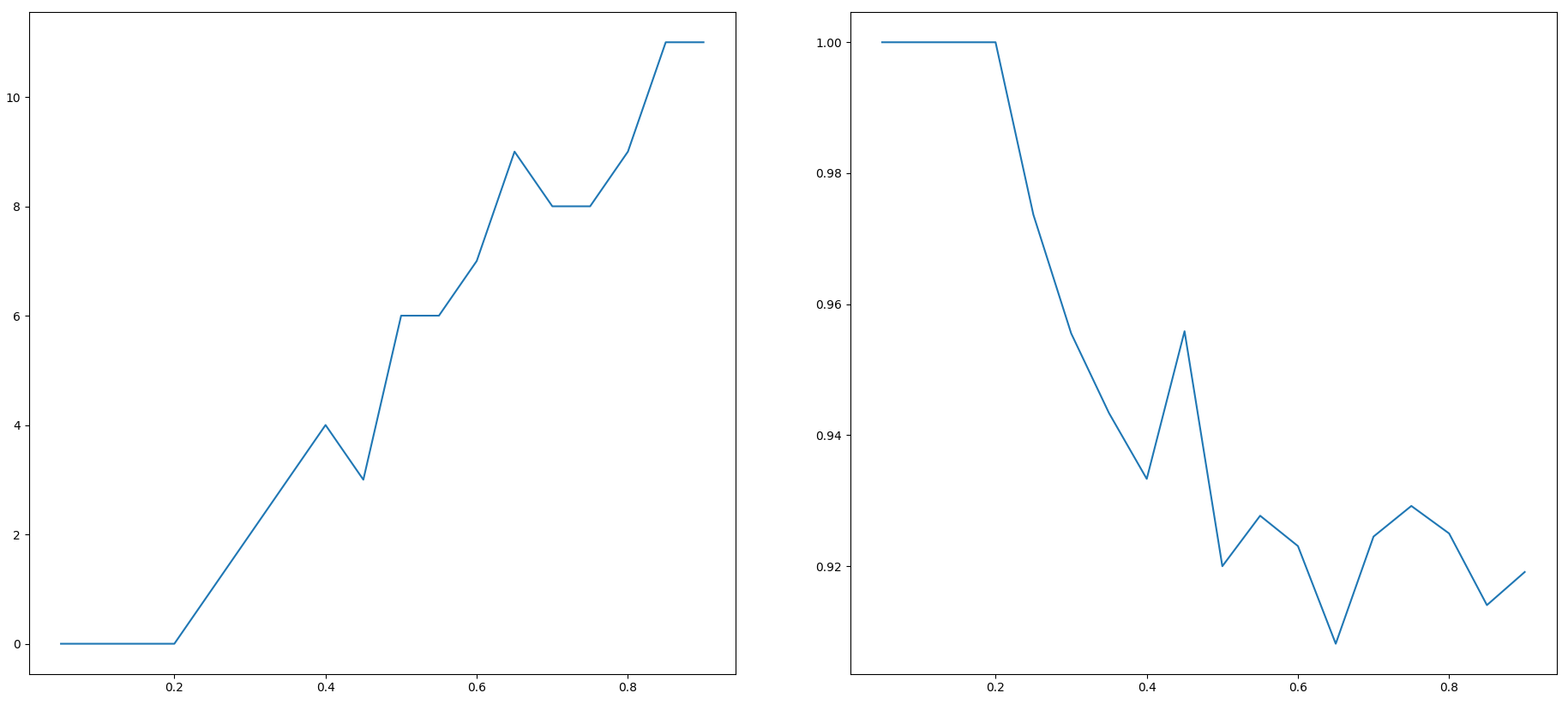
3.6. Проведено исследование для методов NuSVC и LinearSVC.

NuSVC имеет параметр для управления количеством опорных векторов.



NuSVC

LinearSVC аналогичен SVC при kernel = linear, но лучше масштабируется.



LinearSVC

**Выводы**

В ходе выполнения лабораторной работы было произведено знакомство

с классификацией методами GaussianNB, MultinominalNB, ComplementNB, BernoulliNB и DecisionTreeClassifier модуля Sklearn.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

import pandas as pd  
import numpy as np  
from sklearn import preprocessing, svm  
from sklearn.model\_selection import train\_test\_split  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
import matplotlib.pyplot as plt  
  
  
data = pd.read\_csv('iris.data',header=None)  
  
print(data)  
  
X = data.iloc[:,:4].to\_numpy()  
labels = data.iloc[:,4].to\_numpy()  
le = preprocessing.LabelEncoder()  
Y = le.fit\_transform(labels)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.5)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.5, random\_state=0)  
clf = LinearDiscriminantAnalysis()  
y\_pred = clf.fit(X\_train, y\_train).predict(X\_test)  
print((y\_test != y\_pred).sum())  
print(clf.score(X\_train, y\_train))  
  
def grafics(clf, title=""):  
 test\_sizes = np.arange(0.05, 0.95, 0.05)  
 wrong\_results = []  
 scores = []  
  
 for test\_size in test\_sizes:  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=test\_size, random\_state=830406)  
 y\_pred = clf.fit(X\_train, y\_train).predict(X\_test)  
 wrong\_results.append((y\_test != y\_pred).sum())  
 scores.append(clf.score(X\_test, y\_test))  
  
  
 fig, axs = plt.subplots(1, 2, figsize=(8, 4))  
 axs[0].plot(test\_sizes, wrong\_results)  
 axs[1].plot(test\_sizes, scores)  
 fig.suptitle(title)  
 plt.tight\_layout()  
 plt.show()  
  
  
grafics(LinearDiscriminantAnalysis(), 'LinearDiscriminantAnalysis')  
  
target\_names = ['setosa', 'versicolor', 'virginica']  
y = y\_train  
X\_r2 = clf.transform(X\_train)  
  
plt.figure()  
colors = ['navy', 'turquoise', 'darkorange']  
lw = 2  
for color, i, target\_name in zip(colors, [0, 1, 2], target\_names):  
 plt.scatter(X\_r2[y == i, 0], X\_r2[y == i, 1], color=color, alpha=.8, lw=lw,  
 label=target\_name)  
plt.show()  
  
grafics(LinearDiscriminantAnalysis(priors=[0.15, 0.7, 0.15]), 'priors=[0.15, 0.7, 0.15]')  
  
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))  
  
print(clf.support\_vectors\_)  
print(clf.support\_)  
print(clf.n\_support\_)  
  
grafics(svm.SVC(), 'SVC')  
grafics(svm.NuSVC(), 'NuSVC')  
grafics(svm.LinearSVC(), 'LinearSVC')