

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. 8304

Сергеев А.Д.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

Ознакомиться с методами классификации модуля *Sklearn*.

Загрузка данных

1. Данные загружены в датафрейм.

	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 × 5 columns

Рисунок 1 – Исходные данные

2. Выделены данные и их метки, тексты меток преобразованы к числам.
3. Выборка разбита на обучающую и тестовую *train_test_split*.

Линейный дискриминантный анализ

1. Проведена классификация наблюдений при помощи LDA.

```
clf = LinearDiscriminantAnalysis()
y_pred = clf.fit(X_train, y_train).predict(X_test)
print((y_test != y_pred).sum())
3
```

Рисунок 2 — Классификация при помощи LDA

Выявлено 3 неправильно классифицированных наблюдения.

Таблица 1 – Атрибуты *LDA*

Атрибут	Описание
---------	----------

coef_	Векторы веса
intercept_	Срок перехвата
explained_variance_ratio_	Взвешенная внутриклассовая матрица ковариаций
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

Параметр	Описание
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

2. Точность классификации получена с помощью функции `score()` и составляет 93.3%.

```
print(clf.score(X, Y))  
0.9733333333333334
```

Рисунок 3 — Точность классификации

3. Построен график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки (0.05 до 0.95 с шагом 0.05), `random_state = 830434`.

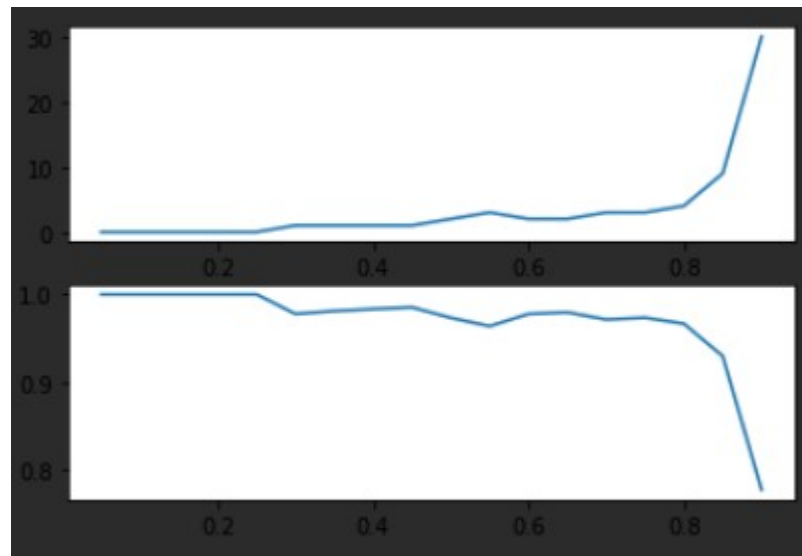


Рисунок 4 – График зависимости точности классификации и неправильно классифицированных наблюдений от размера тестовой выборки

4. Функция `transform` применяется для уменьшения размерности данных.

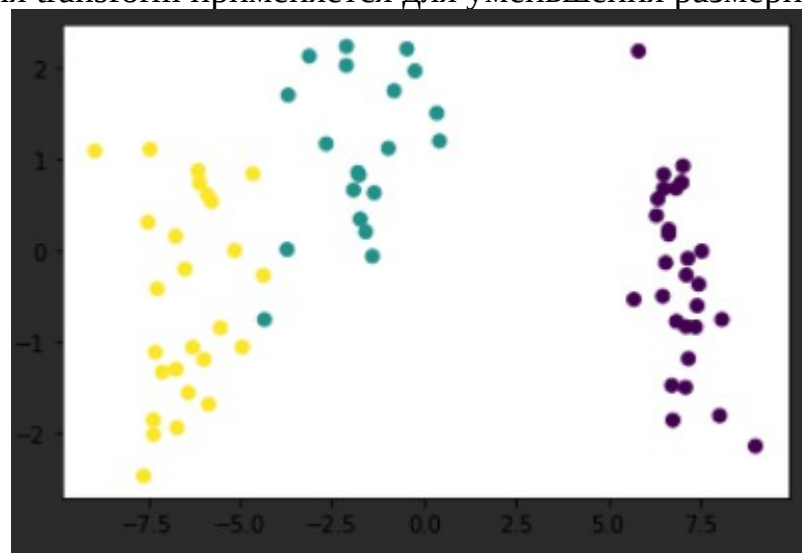


Рисунок 5 – Результат применения функции `transform`

5. Исследована работа классификатора при различных параметрах solver и shrinkage

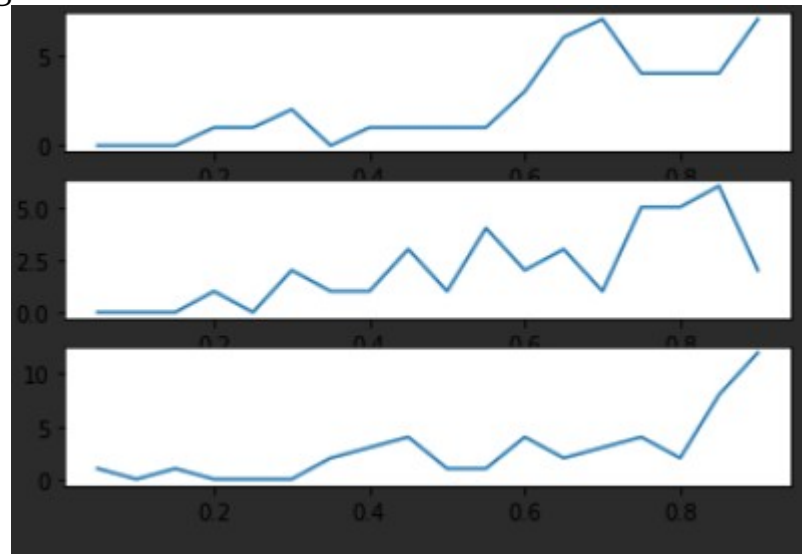


Рисунок 6 — Точность классификации при различных значениях параметра solver

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.

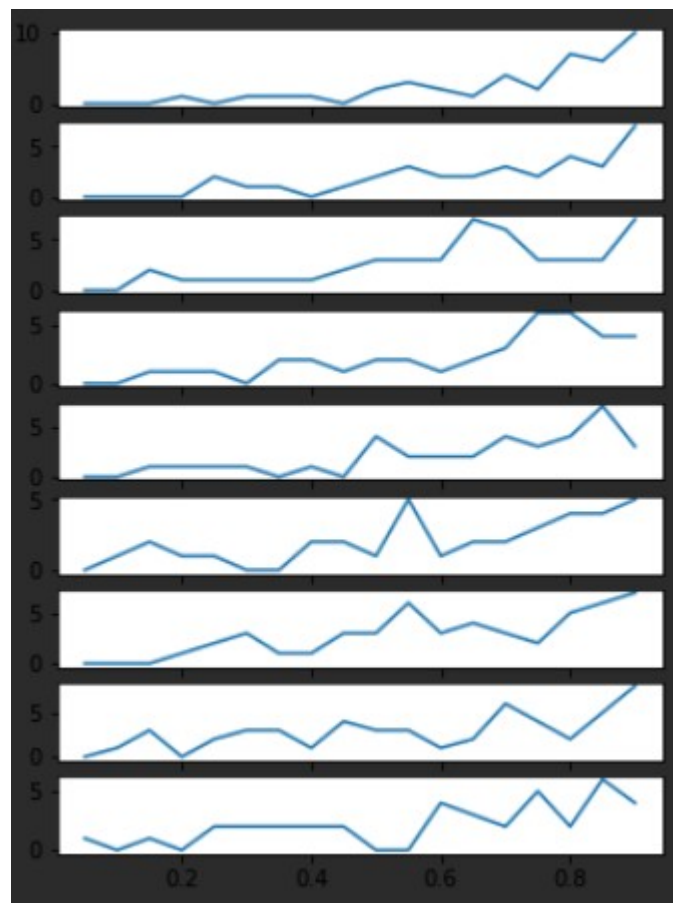


Рисунок 7 — Точность классификации при различных значениях параметра shrinkage

None – без усадки. (по умолчанию)

auto – автоматическая усадка с использованием леммы Ледуа-Вольфа.

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

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

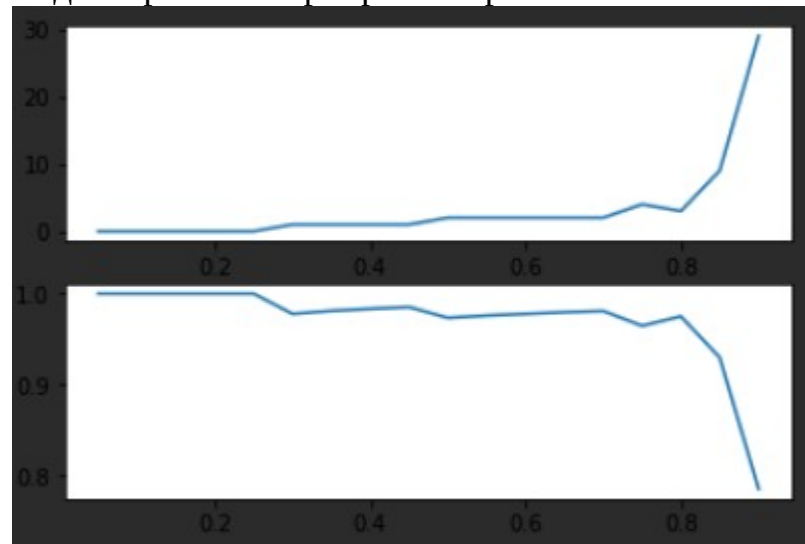


Рисунок 8 – Точности классификации и количество неправильно классифицированных наблюдений при заданных априорных вероятностях

Метод опорных векторов

1. Проведена классификация при помощи SVM на тех же данных.

```
clf = tree.DecisionTreeClassifier()
y_pred = clf.fit(X_train, y_train).predict(X_test)
print((y_test != y_pred).sum())
7
```

Рисунок 9 — Классификация с помощью SVM

Выявлено 7 неправильно классифицированных наблюдений.

2. Точность классификации получена с помощью функции score() и составляет 66.7%.

```
print(clf.score(X, Y))
0.6666666666666666
```

Рисунок 10 — Точность классификации

3. Выведена следующая информация.

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

[[5.1  3.3  1.7  0.5]
 [5.1  3.8  1.9  0.4]
 [5.5  2.4  3.7  1. ]
 [6.7  3.1  4.4  1.4]
 [6.7  3.   5.2  2.3]
 [6.5  3.2  5.1  2. ]
 [6.9  3.1  5.1  2.3]]
[ 6  9  4 10  3  8 12]
[2 2 3]
```

Рисунок 11 – Информация о SVM

Это информация описывает опорные вектора.

support_ содержит индексы опорных векторов.

support_vectors_ содержит сами опорные вектора.

n_support_ содержит количество опорных векторов для каждого класса

4. Построен график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки. Размер тестовой выборки изменяется от 0.05 до 0.95 с шагом 0.05. Параметр random_state выбран равным номеру зачетной книжки (830434).

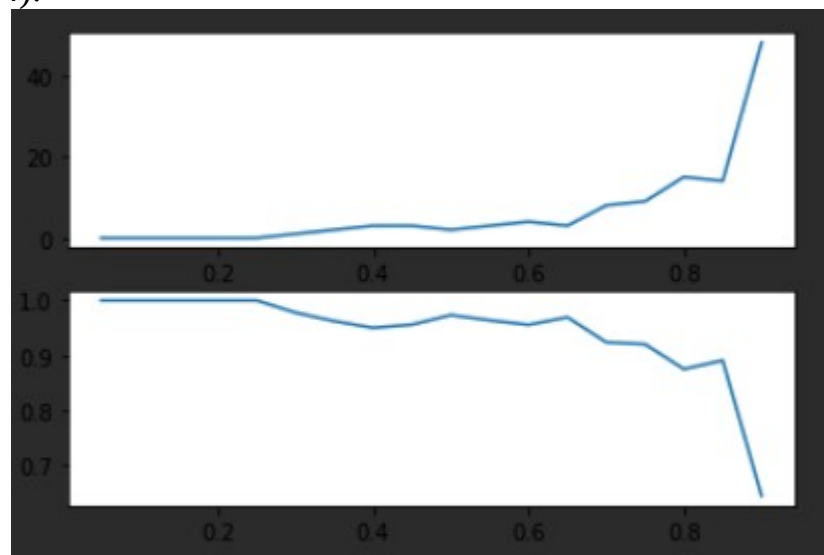
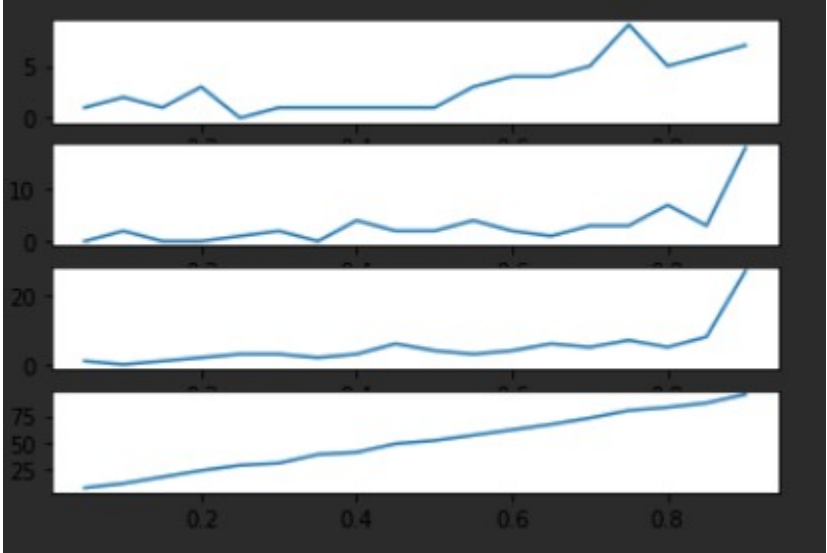
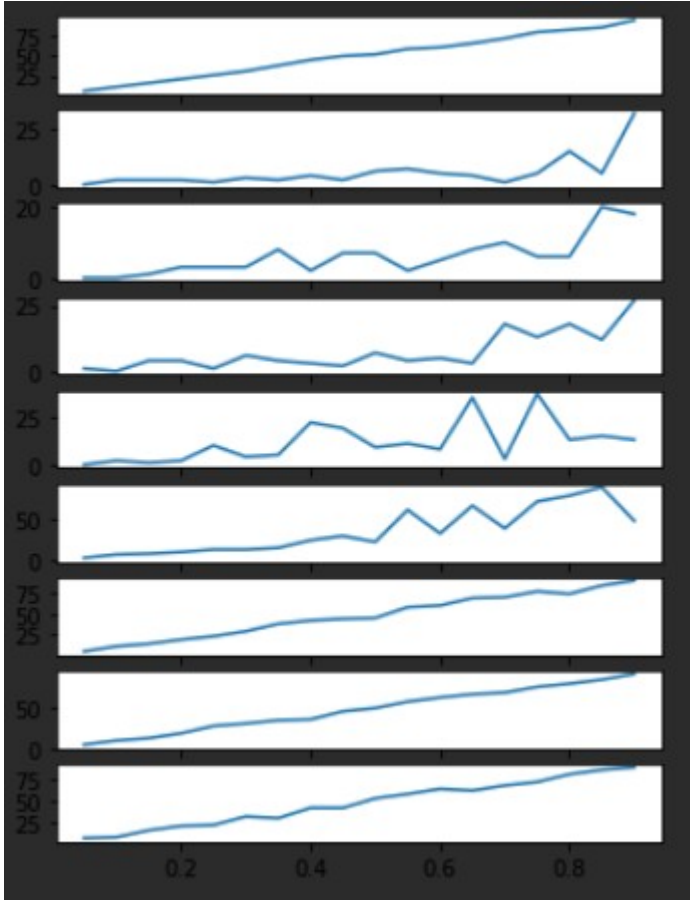
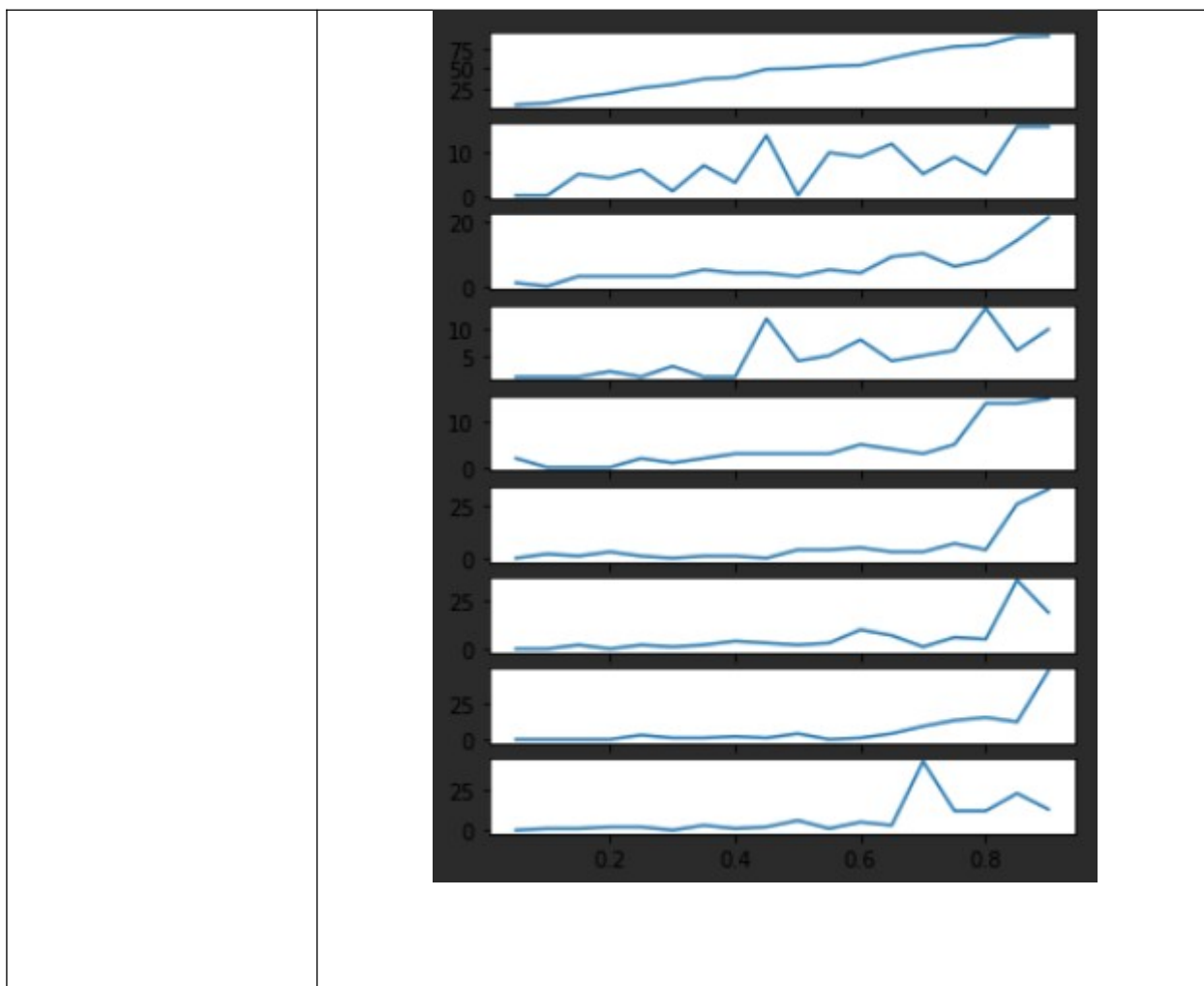


Рисунок 12 – График зависимости точности классификации и неправильно классифицированных наблюдений от размера тестовой выборки

5. Исследована работа метода опорных векторов при различных значениях kernel, degree, max_iter.

Параметр	Описание
kernel	<p>Тип ядра, который будет использован алгоритмом.</p> <p>Поддерживается 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'.</p> <p>Видно, что хуже всего себя показывает ядро sigmoid.</p> 

<p>degree</p>	<p>Степень полиномиальной функции ядра. Только при kernel = 'poly'.</p> <p>Видно, что лучше всего работает ядро в 2, 3, 4 и 5 степени.</p> 
<p>max_iter</p>	<p>Ограничение на количество итераций. При -1 неограниченно.</p> <p>Видно, что при одной итерации алгоритм показывает наихудшие результаты работы.</p>



6. Проведено исследование для методов NuSVC и LinearSVC.

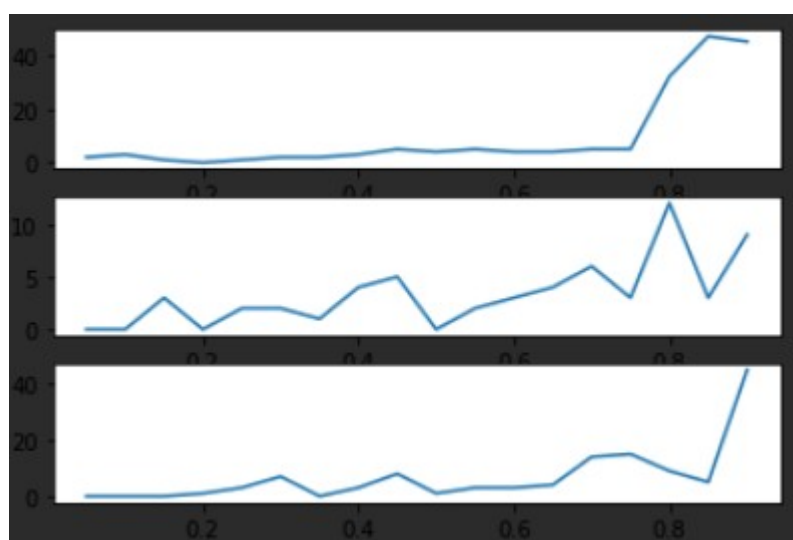


Рисунок 13 — График сравнения алгоритмов SVC, NuSVC и LinearSVC.

NuSVC имеет параметр для управления количеством опорных векторов.

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

Выводы

В ходе выполнения лабораторной работы было произведено знакомство с классификацией методами GaussianNB, MultinomialNB, ComplementNB, BernoulliNB и DecisionTreeClassifier модуля Sklearn.