**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Информационные технологии»**

**Тема: Введение в анализ данных**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Отмахов Д. В. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Введение в анализ данных, изучение базовых аспектов работы библиотеки scikit-learn. Написание программы, обучающей модель и выполняющей с ней определенные функции.

## Задание

Вы работаете в магазине элитных вин и собираетесь провести анализ существующего ассортимента, проверив возможности инструмента классификации данных для выделения различных классов вин.

Для этого необходимо использовать библиотеку sklearn и встроенный в него набор данных о вине.

1) Загрузка данных:

Реализуйте функцию *load\_data*(), принимающей на вход аргумент *train\_size*(размер обучающей выборки, по умолчанию равен 0.8), которая загружает набор данных о вине из библиотеки sklearn в переменную wine. Разбейте данные для обучения и тестирования в соответствии со значением *train\_size*, следующим образом: из данного набора запишите *train\_size* данных из *data*, взяв при этом только 2 столбца в переменную *X\_train* и *train\_size* данных поля target в *y\_train*. В переменную *X\_test* положите оставшуюся часть данных из *data*, взяв при этом только 2 столбца, а в *y\_test* — оставшиеся данные поля target, в этом вам поможет функция *train\_test\_split* модуля *sklearn.model\_selection* (в качестве состояния рандомизатора функции *train\_test\_split* необходимо указать 42).

В качестве результата верните *X\_train*, *y\_train*, *X\_test*, y*\_test*.

Пояснение: *X\_train*, *X\_test* - двумерный массив, *y\_train*, *y\_test*. — одномерный массив.

2) Обучение модели. Классификация методом k-ближайших соседей:

Реализуйте функцию train\_model(), принимающую обучающую выборку (два аргумента - *X\_train* и *y\_train*) и аргументы *n\_neighbors* и *weights* (значения по умолчанию 15 и '*uniform*' соответственно), которая создает экземпляр классификатора *KNeighborsClassifier* и загружает в него данные *X\_train*, *y\_train* c параметрами *n\_neighbors* и *weights*.

В качестве результата верните экземпляр классификатора.

3) Применение модели. Классификация данных

Реализуйте функцию *predict*(), принимающую обученную модель классификатора и тренировочный набор данных (*X\_test*), которая выполняет классификацию данных из  *X\_test*.

В качестве результата верните предсказанные данные.

4) Оценка качества полученных результатов классификации.

Реализуйте функцию *estimate()*, принимающую результаты классификации и истинные метки тестовых данных (*y\_test*), которая  считает отношение предсказанных результатов, совпавших с «правильными» в *y\_test* к общему количеству результатов. (или другими словами, ответить на вопрос «На сколько качественно отработала модель в процентах»).

В качестве результата верните полученное отношение, округленное до 0,001. В отчёте приведите объяснение полученных результатов.

Пояснение: так как это вероятность, то ответ должен находиться в диапазоне [0, 1].

5) Забытая предобработка:

После окончания рабочего дня перед сном вы вспоминаете лекции по предобработке данных и понимаете, что вы её не сделали...

Реализуйте функцию *scale*(), принимающую аргумент, содержащий данные, и аргумент *mode* - тип скейлера (допустимые значения: '*standard*', '*minmax*', '*maxabs*', для других значений необходимо вернуть *None* в качестве результата выполнения функции, значение по умолчанию - '*standard*'), которая обрабатывает данные соответствующим скейлером.

В качестве результата верните полученные после обработки данные.

В отчёте приведите (чек-лист преподавателя):

* описание реализации 5и требуемых функций
* исследование работы классификатора, обученного на данных разного размера  
  + приведите точность работы классификаторов, обученных на данных от функции *load\_data* со значением аргумента *train\_size* из списка: 0.1, 0.3, 0.5, 0.7, 0.9
  + оформите результаты пункта выше в виде таблицы
  + объясните полученные результаты
* исследование работы классификатора, обученного с различными значениями *n\_neighbors*
  + приведите точность работы классификаторов, обученных со значением аргумента *n\_neighbors*из списка: 3, 5, 9, 15, 25
  + в качестве обучающих/тестовых данных для всех классификаторов возьмите результат *load\_data* с аргументами по умолчанию (учтите, что для достоверности результатов обучение и тестирование классификаторов должно проводиться на одних и тех же наборах)
  + оформите результаты в виде таблицы
  + объясните полученные результаты
* исследование работы классификатора с предобработанными данными
  + приведите точность работы классификаторов, обученных на данных предобработанных с помощью скейлеров из списка: *StandardScaler*, *MinMaxScaler*, *MaxAbsScaler*
  + в качестве обучающих/тестовых данных для всех классификаторов возьмите результат *load\_data* с аргументами по умолчанию - учтите, что для достоверности сравнения результатов классификации обучение должно проводиться на одних и тех же данных, поэтому предобработку следует производить после разделения на обучающую/тестовую выборку.
  + оформите результаты в виде таблицы
  + объясните полученные результаты

## Выполнение работы

Программа состоит из пяти функций:

* *load\_data(train\_size = 0.8)* – загружает набор данных о вине из *load\_wine()* в переменную *wine* и составляет на основе данных обучающую и тестовую выборку в соответствии с *train\_size*.
* *train\_model(X\_train, y\_train, n\_neighbors = 15, weights = ’uniform’)* – обучает модель: принимает на вход обучающую выборку и аргументы *n\_neighbors* и *weights*, создает экземпляр классификатора *KNeighborsClassifier* с параметрами *n\_neighbors* и *weights* и данными *X\_train* и *y\_train*.
* *predict(clf, X\_test)* – принимает обученную модель классификатора и тренировочный набор данных, предсказывает данные тренировочного набора *X\_test* используя обученную модель *clf*.
* *estimate(res, y\_test)* – принимает результаты классификации *res* и истинные метки тестовых данных *y\_test*, определяет качество обученной модели (отношение предсказанных результатов,совпавших с «правильными» в y\_test к общему количеству результатов). Для сравнения используется функция *accuracy\_score*, возвращающая результат в пределах [0, 1].
* *scale(data, mode=’standard’)* – принимает данные *data* и тип скейлера *mode*, обрабатывает тренировочные данные на основе выбранного скейлера.

Таблица 1. Результаты исследования работы классификатора для данных разного размера.

|  |  |
| --- | --- |
| Размер *train\_size* | Точность |
| 0.1 | 0.379 |
| 0.3 | 0.8 |
| 0.5 | 0.843 |
| 0.7 | 0.815 |
| 0.9 | 0.722 |

На основе результатов табл. 1 можно сделать следующие выводы: при *train\_size* = 0.8 (дефолтном) достигается самая высокая точность работы; при очень маленьких значениях *train\_size* – самая маленькая точность, так как модели не хватает данных для обучения; при очень больших значениях *train\_size* – точность также уменьшается, так как происходит переобучение модели.

Таблица 2. Результаты исследования работы классификатора для различных значений *n\_neighbors*.

|  |  |
| --- | --- |
| Размер *n\_neighbors* | Точность |
| 3 | 0.861 |
| 5 | 0.833 |
| 9 | 0.861 |
| 15 | 0.861 |
| 25 | 0.833 |

На основе результатов табл. 2 можно сделать следующий вывод: значение *n\_neighbors* не сильно влияет на точность работы классификатора. Также можно предположить, что при сильно большом значении *n\_neighbors*, точность работы уменьшиться, так как границы классификации станут менее четкими.

Таблица 3. Результаты исследования работы классификатора для разных скейлеров.

|  |  |
| --- | --- |
| Скейлер | Точность |
| *StandardScaler* | 0.889 |
| *MinMaxScaler* | 0.806 |
| *MaxAbsScaler* | 0.75 |

На основе результатов табл. 3 можно сделать следующие выводы: максимальная точность достигается при использовании *StandardScaler*; минимальная – при использовании *MaxAbsScale*. Это объясняется тем, что в отличие от *StandardScaler*, *MinMaxScale* и *MaxAbsScaler* чуствительны к выбросам.

## Выводы

В ходе выполнения лабораторной работы были изучены базовые аспекты работы библиотеки scikit-learn, написана программа, обучающая модель и выполняющая с ней определенные функции.

# Приложение А Исходный код программы

Название файла: main.py

from sklearn.datasets import load\_wine

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

def load\_data(train\_size = 0.8):

wine = load\_wine()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(wine.data, wine.target, train\_size = train\_size, random\_state = 42)

return X\_train[:, :2], X\_test[:, :2], y\_train, y\_test

def train\_model(X\_train, y\_train, n\_neighbors = 15, weights = 'uniform'):

return KNeighborsClassifier(n\_neighbors = n\_neighbors, weights = weights).fit(X\_train, y\_train)

def predict(clf, X\_test):

return clf.predict(X\_test)

def estimate(res, y\_test):

return round(accuracy\_score(res, y\_test), 3)

def scale(data, mode = 'standard'):

if mode not in ['standard', 'minmax', 'maxabs']:

return None

if mode == 'standard':

scale = StandardScaler().fit\_transform(data)

elif mode == 'minmax':

scale = MinMaxScaler().fit\_transform(data)

else:

scale = MaxAbsScaler().fit\_transform(data)

return scale