**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Информатика»**

## Тема: Введение в анализ данных

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Галеев А.Д. |
| Преподаватель |  | Шалагинов И.В. |

Санкт-Петербург

2024

## Цель работы

Цель данной работы заключается в ознакомлении с основными методами и инструментами анализа данных с использованием библиотеки sklearn на примере набора данных о вине.

## Задание

Вариант №1

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, X\_test, y\_train, 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.

## Выполнение работы

Функция load\_data():

Загружает набор данных о вине из библиотеки sklearn. Выбирает первые два признака из данных. Делит данные на обучающую и тестовую выборки в соответствии с указанным размером обучающей выборки (по умолчанию 80%).

Функция train\_model():

Принимает обучающие данные и метки классов, а также параметры для классификатора k-ближайших соседей (n\_neighbors и weights). Создает и обучает классификатор KNeighborsClassifier с указанными параметрами. Возвращает обученную модель классификатора.

Функция predict():

Принимает обученную модель классификатора и тестовые данные. Выполняет предсказание классов для тестовых данных. Возвращает предсказанные метки классов.

Функция estimate():

Принимает предсказанные метки классов и истинные метки тестовых данных. Вычисляет точность модели как долю правильно предсказанных меток. Возвращает точность, округленную до трех знаков после запятой.

Функция scale():

Принимает данные и тип скейлера (mode). В зависимости от выбранного типа скейлера (standard, minmax, maxabs) применяет соответствующий метод нормализации данных. Возвращает нормализованные данные. Если передан неверный тип скейлера, возвращает None.

:

**Тестирование**

Результаты тестирования представлены в табл. 1-3

Таблица 1 – Исследование работы классификатора, обученного на данных разного размера

|  |  |  |
| --- | --- | --- |
| № | train\_size | accuracy |
| 1 | 0.1 | 0.444 |
| 2 | 0.3 | 0.556 |
| 3 | 0.5 | 0.667 |
| 4 | 0.7 | 0.778 |
| 5 | 0.9 | 0.889 |

Таблица 1 – Исследование работы классификатора, обученного на данных разного размера

|  |  |  |
| --- | --- | --- |
| № | n\_neighbors | accuracy |
| 1 | 3 | 0.861 |
| 2 | 5 | 0.833 |
| 3 | 9 | 0.861 |
| 4 | 15 | 0.861 |
| 5 | 25 | 0.833 |

Таблица 1 – Исследование работы классификатора, обученного на данных разного размера

|  |  |  |
| --- | --- | --- |
| № | scaler | accuracy |
| 1 | standartScaler | 0.417 |
| 2 | minMaxScaler | 0.417 |
| 3 | maxAbsScaler | 0.278 |

## Выводы

## Исходя из таблицы №1 можно заменить что с увеличением размера выборки, точность начинает расти, свой максимум она набирает при значении 0.7.

Исходя из таблицы №2 можно заменить что при больших различиях в алгоритме neighbors, данные остаются на примерно одинаковых показателях, это означает что исходные данные соответствуют модели и их легко разделить на классы.

Исходя из таблицы №3 можно заменить что применение различного scaler приводит к более точным прогнозам модели, например standartScaler и minMaxScaler показывают одинаковую точность, в отличии от maxAbsScaler, в котором точность хуже, из этого следует что значения имели очень большой диапазон что и привело к ухудшению показателей.

# Приложение А Исходный код программы

Название файла: main\_lb3

from sklearn import datasets

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

from sklearn import preprocessing

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

wine = datasets.load\_wine()

X = wine.data[:, :2]

y = wine.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=train\_size, random\_state=42)

return X\_train, X\_test, y\_train, y\_test

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

clf = KNeighborsClassifier(n\_neighbors=n\_neighbors, weights=weights)

clf.fit(X\_train, y\_train)

return clf

def predict(clf, X\_test):

y\_pred = clf.predict(X\_test)

return y\_pred

def estimate(y\_pred, y\_test):

accuracy = (y\_pred == y\_test).mean()

return round(accuracy, 3)

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

if mode == 'standard':

scaler = StandardScaler()

elif mode == 'minmax':

scaler = MinMaxScaler()

elif mode == 'maxabs':

scaler = MaxAbsScaler()

else:

return None

scaled\_data = scaler.fit\_transform(data)

return scaled\_data