**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](https://e.moevm.info/course/view.php?id=69#section-4)

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Львов А. В. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Изучение основ анализа данных и машинного обучения, освоение работы с основными инструментами обработки данных и тренировки модели с последующим их применением.

## 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

Функции, реализованные в программе:

1. load\_data принимает один необязательный аргумент train\_size, отвечающий за размер обучающей выборки с значением по умолчанию 0.8. Она загружает датасет о вине из библиотеки sklearn и возвращает обучающие и тестовые выборки.
2. train\_model принимает тестовые выборки и опциональные данные о количестве соседей и о весе каждой точки с значениями по умолчанию 15 и «uniform» соответственно. Возвращает экземпляр классификатора.
3. predict принимает экземпляр класса KNeighborsClassifier и тренировочный набор данных. Эта функция выполняет классификацию данных из переданной выборки и возвращает предсказанные данные.
4. estimate выполняет классификацию данных из переданной выборки и возвращает долю правильных предсказаний.
5. scale принимает аргумент, содержащий данные и тип скейлера, которым обрабатывает их и возвращает результат.

Исходный код программы см. в приложении А.

## Тестирование

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

|  |  |
| --- | --- |
| Размер обучающей выборки | Точность |
| 0.1 | 0.379 |
| 0.3 | 0.8 |
| 0.5 | 0.843 |
| 0.7 | 0.815 |
| 0.9 | 0.722 |

Полученные результаты связаны с тем, что при недостаточном размере обучающей выборки классификатору будет не хватать данных для более точной классификации. А при преобладании обучающей выборки над тестовой модель может переобучаться на обучающих данных, что приводит к снижению точности на тестовых данных.

Таблица 2 – Исследование работы классификатора, обученного с различными значениями n\_neighbors.

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

В общем случае, недостаточное или слишком большое количество соседей приводит к переобучению и недообучению модели соответственно, однако в данном случае можно сделать вывод, что количество соседей существенно не влияет на точность.

Таблица 3 – Исследование работы классификатора с предобработанными данными

|  |  |
| --- | --- |
| Скейлер | Точность |
| StandardScaler | 0.417 |
| MinMaxScaler | 0.417 |
| MaxAbsScaler | 0.278 |

Скейлеры позволяют нормализовать данные для улучшения качества моделей и их обобщающей способности. В общем, их выбор зависит от конкретной задачи. На данном примере можно сделать вывод о том, что для предобработки данных о вине лучше всего подходят StandardScaler и MinMaxScaler.

## Выводы

Было проведено ознакомление с основными инструментами обработки и анализа данных. Было проведено тестирование с различными наборами данных для изучения основных концепций и закономерностей машинного обучения.

# 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[:, :2], wine.target, 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'):

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

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

return classifier

def predict(classifier, X\_test):

return classifier.predict(X\_test)

def estimate(res, y\_test):

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

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

mods = {'standard': StandardScaler(), 'minmax': MinMaxScaler(), 'maxabs': MaxAbsScaler()}

if mode in mods:

scaler = mods[mode]

else:

return

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

return scaled\_data