

**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

Лапшов К. Н.

Преподаватель

Иванов Д. В.

Санкт-Петербург

2024

## **Цель работы**

Изучить основные принципы анализа данных и освоить ключевые инструменты для их обработки и анализа. Овладеть навыками работы с данными, включая сбор, очистку, визуализацию и интерпретацию результатов.

## 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` с 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 винах из библиотеки `datasets` и разделяет его на тренировочный и тестовый наборы данных. Сперва загружает набор данных о винах, который входит в библиотеку `sklearn.datasets`, после этого, с помощью `train_test_split`, разделяет данные на тренировочные и тестовые наборы в соответствии с заданным размером `train_size`, (который по умолчанию равен 0.8) и возвращает их.

Функция `train_model()` обучает модель классификации методом k-ближайших соседей (k-Nearest Neighbors, KNN) на предоставленных тренировочных данных. В начале создает экземпляр классификатора KNN с переданными параметрами `n_neighbors` и `weights`. После этого, обучает классификатор на тренировочных данных `X_train` и соответствующих метках `y_train` с помощью функции `fit()`, и возвращает обученную модель.

Функция `predict()` выполняет прогнозирование классов для тестовых данных с использованием обученной модели классификатора. С помощью функции `predict()`, у переданной обученной модели `clf`, выполняется прогнозирование классов для переданных данных `X_test`.

Функция `estimate()` оценивает точность модели классификации, сравнивая предсказанные метки классов с истинными метками тестового набора данных. С помощью функции `accuracy_score()` вычисляется точность модели, путем сравнения истинных меток класса `y_test` с предсказанными метками `res`. Ответ округляется до трех знаков после запятой.

Функция `scale()` масштабирует данные, используя один из нескольких способов нормализации, в зависимости от переданного режима. С помощью конструкции `if-elif-else` выбирает соответствующий метод масштабирования в зависимости от значения `mode`, а после с помощью `scaler.fit_transform(data)`, принимает выбранный метод масштабирования к переданным данным.

Разработанный программный код см. в приложении А.

## Тестирование

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

№	train_size	accuracy
1.	0.1	0.778
2.	0.3	0.839
3.	0.5	0.889
4.	0.7	0.944
5.	0.9	0.972

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

№	n_neighbors	accuracy
1.	3	0.944
2.	5	0.972
3.	9	0.972
4.	15	0.972
5.	25	0.944

Таблица 3 - Исследование работы классификатора с предобработанными данными

№	scaler	accuracy
1.	StandardScaler	0.972
2.	MinMaxScaler	0.972
3.	MaxAbsScaler	0.972

## **Выводы**

Полученные результаты из таблицы 1 показывают, что при увеличении размера обучающей выборки (`train_size`) точность работы классификатора увеличивается. При достаточно большом размере обучающей выборки (0.7 и выше) точность достигает максимального значения.

Полученные результаты из таблицы 2 показывают, что при увеличении значения параметра `n_neighbors` точность работы классификатора увеличивается до достижения максимального значения (при `n_neighbors = 5, 9` и `15`). При дальнейшем увеличении значения `n_neighbors` точность снова уменьшается.

Полученные результаты из таблицы 3 показывают, что применение различных скейлеров для предобработки данных не влияет на точность работы классификатора. Это может быть связано с тем, что выбранные столбцы данных (индексы 1 и 2) уже нормализованы и не требуют дополнительной предобработки.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from sklearn import datasets
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
MaxAbsScaler

def load_data(train_size=0.8):
    wine = datasets.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'):
    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(y_test, res), 3)

def scale(data, mode='standard'):
    if mode == 'standard':
        scaler = StandardScaler()
    elif mode == 'minmax':
        scaler = MinMaxScaler()
    elif mode == 'maxabs':
        scaler = MaxAbsScaler()
    else:
        return None

    return scaler.fit_transform(data)
```



