

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №3**  
**по дисциплине «Информационный технологии»**  
**Тема: Введение в анализ данных**

Студент гр. 3344		Тукалкин. В.А.
Преподаватель		Иванов Д.В.

Санкт-Петербург  
2024

## **Цель работы**

Изучить базовые принципы и инструменты анализа данных на языке Python с помощью библиотеки sklearn.

## **Задание.**

### **Вариант 2.**

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.

#### **2) 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

### 1) Реализация функций:

- `load_data` – функция загружает данные в переменную `wine`, после этого выбираются первые два столбца и записываются в `x`, а поля `target` в `y`. В конце данные разделяются и возвращаются
- `train_model` – создаётся классификатор и обучается на основе тестовых наборов, после этого он возвращается
- `predict` – используется обученный классификатор для прогнозирования меток тестовых данных и возвращает эти метки
- `estimate` – вычисляет точность прогнозов через сравнение предсказанных и заданными метками, возвращает точность
- `scale` – масштабирует данные и возвращает их

### 2) Обучение на данных разного размера (табл. 1):

Таблица 1 – Результаты работы классификатора

Размер набора	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Точность	0.379	0.797	0.8	0.822	0.843	0.819	0.815	0.861	0.722

Видно, что при увеличении аргумента точность увеличивается, но при значении 0.9 он уменьшается. Это связано с переобучением модели, она получает больше о “шуме”, а не о закономерностях.

### 3) Обучение с различными значениями `n_neighbors` (табл. 2):

Таблица 2 – результаты работы классификатора

Количество соседей	3	5	9	15	25
Точность	0.861	0.833	0.861	0.861	0.833

При увеличении количества соседей в среднем идёт рост точности, но при большом их количестве так же происходит переобучение модели, поэтому точность уменьшается.

4) Обучение с пред обработанными данными (табл. 3):

Таблица 3 – результаты работы классификатора

Скейлер	standart	minmax	maxabs
Точность	0.889	0.806	0.806

Особой разницы между скейлерами нет. В основном она зависит от особенности настроек конкретных данных.

## **Выводы**

Были изучены основы анализа данных на языке Python с применением библиотеки `sklearn`. Разработаны функции для разделения данных для обучения и тестирования, обучения модели, вычисления предсказаний на основе данных и оценки качества полученных результатов классификации.



## 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.preprocessing import StandardScaler, MinMaxScaler,
MaxAbsScaler
from sklearn.metrics import accuracy_score

def load_data(train_size=0.8):
    wine=load_wine()
    X,y=wine.data, wine.target
    X_train, X_test, y_train, y_test = train_test_split(X[:, :2],
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'):
    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_true=y_test, y_pred=res), 3)

def scale(X, mode='standard'):
    if mode not in ['standard', 'minmax', 'maxabs']: return None
    scaler = StandardScaler()
    if mode == 'minmax': scaler = MinMaxScaler()
    elif mode == 'maxabs': scaler = MaxAbsScaler()
    scaler = scaler.fit(X)
    x_scaled = scaler.transform(X)
    return x_scaled
```