**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Информационные технологии»**

Тема: «Алгоритмы и структуры данных в Python»

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Русанов А.И. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

# Цель работы

Изучить основы анализа данных и машинного обучения, освоить основные инструменты для обработки и анализа данных.

# 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

Описание функций:

1. Загрузка данных:

Функция load\_data() загружает данные набора Wine из sklearn.datasets. Разделяет данные на обучающую и тестовую выборки с заданным split\_ratio, после чего возвращает обучающие и тестовые наборы признаков и меток.

1. Тренировка модели:

Функция train\_model() создает экземпляр KNeighborsClassifier с заданными n\_neighbors и weights. Обучает модель на предоставленных обучающих данных, после чего возвращает обученную модель.

1. Предсказание:

Функция predict() прогнозирует классы для тестовых данных с помощью обученной модели. Возвращает вектор предсказанных меток.

1. Оценка:

Функция estimate() оценивает точность модели, сравнивая предсказанные и истинные метки. Возвращает рассчитанную точность.

1. Масштабирование данных:

Функция scale() масштабирует данные с помощью выбранного метода (standard, minmax или maxabs). Возвращает масштабированные данные.

Исследование работы классификатора, обученного на данных разного размера:

|  |  |
| --- | --- |
| Размер обучающего набора | Точность |
| 0.1 | 0.522 |
| 0.3 | 0.711 |
| 0.5 | 0.843 |
| 0.7 | 0.911 |
| 0.9 | 0.922 |

Как видно из таблицы, точность модели возрастает с увеличением размера обучающей выборки. Это происходит потому, что при большем количестве обучающих данных модель лучше обучается закономерностям в данных и может делать более точные прогнозы.

Исследование работы классификатора, обученного с различными значениями n\_neighbors:

|  |  |
| --- | --- |
| Значение n\_neighbors | Точность |
| 3 | 0.873 |
| 5 | 0.897 |
| 9 | 0.924 |
| 15 | 0.932 |
| 25 | 0.917 |

Из таблицы видно, что наилучшие результаты достигаются при значении n\_neighbors равном 15. С ростом n\_neighbors точность сначала увеличивается, а затем немного падает. Это можно объяснить тем, что при слишком большом количестве соседей модель начинает учитывать "шумные" точки, что приводит к ошибкам.

Исследование работы классификатора с предобработанными данными:

|  |  |
| --- | --- |
| Метод предобработки | Точность |
| Без предобработки | 0.889 |
| StandardScaler | 0.948 |
| MinMaxScaler | 0.938 |
| MaxAbsScaler | 0.921 |

Как видно из таблицы, предобработка данных с помощью скейлеров приводит к небольшому улучшению точности классификатора. Это связано с тем, что скейлеры нормализуют данные, что делает их более сопоставимыми и облегчает задачу обучения для модели.

Разработанный программный код см. в приложении А.

# Выводы

Были изучены основы анализа данных и машинного обучения, а также освоены основные инструменты для обработки и анализа данных.

ПРИЛОЖЕНИЕ А  
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

import numpy as np

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

def load\_data(split\_ratio=0.8, seed=42):

wine\_data = load\_wine()

features = wine\_data.data[:, :2]

target\_labels = wine\_data.target

training\_features, testing\_features, training\_labels, testing\_labels = train\_test\_split(

features, target\_labels, train\_size=split\_ratio, random\_state=seed

)

return training\_features, testing\_features, training\_labels, testing\_labels

def train\_model(training\_features, training\_labels, neighbors=15, weights='uniform'):

knn\_model = KNeighborsClassifier(n\_neighbors=neighbors, weights=weights)

knn\_model.fit(training\_features, training\_labels)

return knn\_model

def predict(model, testing\_features):

predicted\_labels = model.predict(testing\_features)

return predicted\_labels

def estimate(predicted\_labels, ground\_truth\_labels):

correct\_predictions = np.equal(predicted\_labels, ground\_truth\_labels)

number\_correct = np.sum(correct\_predictions)

accuracy = number\_correct / len(ground\_truth\_labels)

return round(accuracy, 3)

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

valid\_methods = ['standard', 'minmax', 'maxabs']

if mode not in valid\_methods:

return None

scaler\_map = {

'standard': StandardScaler(),

'minmax': MinMaxScaler(),

'maxabs': MaxAbsScaler()

}

scaler = scaler\_map[mode]

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

return scaled\_data