**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: Введение в анализ данных

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3342 |  | Антипина В.А. |
| Преподаватель |  | Иванов И.И. |

Санкт-Петербург

2024

## Цель работы

Изучение методов анализа данных на языке Python, модуля scikit, реализация программы, анализирующей данные встроенной базы данных wine.

## 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 принимает на вход коэффициент выборки для тестирования (долю от общего объёма выборки). Загружается нужный датасет. Выбираются первые два столбца данных и загружаются в переменную X, в переменную y загружается столбец „target“. Вызывается функция train\_test\_split (которая была импортирована из sklearn.model\_selection), результат записывается в переменные X\_train, X\_test, y\_train, y\_test. Эти же переменные возвращаются.

Функция train\_model обучает модель классификации методом ближайших соседей. Функция создает экземпляр классификатора KNeighborsClassifier с помощью одноимённой функции от аргументов n\_neighbors, weights и с помощью функции fit, применённой к тренировочной выборке, загружает в него данные. Возвращается обученная модель.

Функция predict прогнозирует классы для тестовых данных с помощью метода predict.

Функция estimate возвращает долю «успешности» построения модели, округленную до 3 знака после запятой. Это значение вычисляется с помощью функции accuracy\_score и фактически является отношением предсказанных значений, совпавших с правильными данными, ко всем правильным данным.

Функция scale применяет один из трёх методов масштабирования к данным и возвращает обработанные данные.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования первой функции

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | train\_size=0.1 | 0.379 | Малый объём обучающих данных привёл к неточному результату |
|  | train\_size=0.3 | 0.547 |  |
|  | train\_size=0.5 | 0.843 |  |
|  | train\_size=0.7 | 0.892 |  |
|  | train\_size=0.9 | 0.971 | Большой объём обучающих данных привёл к точному результату |

Таблица 2 – Результаты тестирования второй функции

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | n\_neighbors=3 | 0.915 |  |
|  | n\_neighbors=5 | 0.963 |  |
|  | n\_neighbors=9 | 0.971 | При достижении максимума точность уменьшается |
|  | n\_neighbors=15 | 0.971 |  |
|  | n\_neighbors=25 | 0.915 |  |

Таблица 3 – Результаты тестирования третьей функции

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | standart | 0.971 | Одинаковый результат |
|  | minmax | 0.971 |  |
|  | maxabs | 0.971 |  |

## Выводы

Результаты тестирования показывают, что с увеличением объёма обучающих данных увеличивается точность классификатора.

Также при изменении параметра n\_neighbors точность растёт до достижения максимального значения, а затем убывает.

При использовании разных скейлеров точность не меняется, что доказывает, что исходные данные не требуют значительной предобработки.

# Приложение А Исходный код программы

Название файла: main.py

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

import numpy as np

import pandas as pd

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

data = datasets.load\_wine()

X = data.data[:,[0,1]]

y = data.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'):

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

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

return knn

def predict(clf, X\_test):

knn\_predict = clf.predict(X\_test)

return knn\_predict

def estimate(res, y\_test):

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

return accuracy

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

if(mode=='standard'):

scaler = preprocessing.StandardScaler()

elif(mode=='minmax'):

scaler = preprocessing.MinMaxScaler()

elif(mode=='maxabs'):

scaler = preprocessing.MaxAbsScaler()

else:

return None

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

return scaled\_data