**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. 3344 |  | Ханнанов А.Ф. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Ознакомиться с библиотеками для анализа данных.

## Задание.

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, y\_train, X\_test, 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

1. Реализация функций:
   1. 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
   2. train\_model – создаётся классификатор и обучается на основе тестовых наборов, после этого он возвращается
   3. predict – используется обученный классификатор для прогнозирования меток тестовых данных и возвращает эти метки
   4. estimate – вычисляет точность прогнозов через сравнение предсказанных и заданными метками, возвращает точность
   5. 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.

1. 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.

1. 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

В ходе лабораторной работы прошло ознакомление с основами анализа данных. Был получен опыт анализа данных с помощью библиотек языка python.

# Приложение А Исходный код программы

Название файла: Khannanov\_Artem\_lb3.py

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import MaxAbsScaler, MinMaxScaler, StandardScaler

import numpy as np

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

wine = datasets.load\_wine()

x = wine.data[:, [0, 1]]

y = wine.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'):

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

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

return classifier

def predict(clf, X\_test):

prd = clf.predict(X\_test)

return prd

def estimate(res, y\_test):

correct = np.sum(res == y\_test)

total = len(y\_test)

return round(correct / total, 3)

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

if mode == 'standard':

scaler = StandardScaler()

elif mode == 'minmax':

scaler = MinMaxScaler()

elif mode == 'maxabs':

scaler = MaxAbsScaler()

else:

return None

scaled = scaler.fit\_transform(args)

return scaled