**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**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3344 |  | Мурдасов М.К. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Введение в анализ данных. Изучение основных инструментов анализа данных на Python.

## 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. *load\_data()*:

Загружает данные о вине из библиотеки sklearn и с помощью *train\_test\_split()*разделяет данные на тренировочные и тестируемые массивы. Аргумент *train\_size* задает размер обучающей выборки и по умолчанию равен 0.8.

1. *train\_model()*:

Обучает и возвращает модель классификатора k-ближайштх соседей (*KNeighborsClassifier*) на выборке *X\_train* с метками *y\_train*. Аргументы *n\_neighbors*задает кол-во соседей, а*weight -* весовую функцию для классификатора. По умолчанию *n\_neighbors* равен15 и *weights равен 'uniform'*.

1. *predict()*:

Предсказывает значения меток, основываясь на тестовой выборке *X\_test*.

1. *estimate():*

Вычисляет точность классификации с помощью *accuracy\_score*() из sklearn.*metrics* и возвращает ее значение, округленное до 0.001.

1. *scale():*

Принимает массив данных X, режим масштабирования *mode* и возвращает масштабированный массив данных. Допустимые значения для *mode*: *'standard'*, *'minmax'*, *'maxabs'*. Если значение *mode* не является допустимым, функция возвращает *None*. Если *mode = 'standard'*, функция использует стандартное масштабирование (*StandardScaler*), если *mode = 'minmax'* - мини-максимальное масштабирование (*MinMaxScaler*), если *mode = 'maxabs'* - масштабирование по максимальному абсолютному значению (*MaxAbsScaler*). Масштабирование выполняется с помощью соответствующих классов из модуля *sklearn.preprocessing*.

Исследование работы классификатора, обученного на данных разного размера:

|  |  |
| --- | --- |
| load\_data с размерами данных | Точность работы классификатора |
| *load\_data(0.1)* | 0.379 |
| *load\_data(0.3)* | 0.8 |
| *load\_data(0.5)* | 0.843 |
| *load\_data(0.7)* | 0.815 |
| *load\_data(0.9)* | 0.722 |

Можно заметить, что точность классификации зависит от размера выборки. Слишком маленькая и слишком большая выборки снижают точность, тк имеют маленький объем данных или, напротив, слишком большой разброс этих данных.

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

|  |  |
| --- | --- |
| значения *n\_neighbors* | Точность работы классификатора |
| 3 | 0.861 |
| 5 | 0.833 |
| 9 | 0.861 |
| 15 | 0.861 |
| 25 | 0.833 |

Можно заметить, что количество соседей незначительно влияет на точность работы классификатора.

Исследование работы классификатора с предобработанными данными:

|  |  |
| --- | --- |
| Метод предобработки | Точность работы классификатора |
| *StandardScaler* | 0.417 |
| *MinMaxScaler* | 0.417 |
| *MaxAbsScaler* | 0.278 |

Из полученных результатов можно заметить, что точность классификации зависит от способа масштабирования данных. При использовании стандартного масштабирования (StandardScaler) и минимакс-масштабирования (MinMaxScaler) точность классификации составляет 0.417, а при использовании максимального абсолютного масштабирования (MaxAbsScaler) точность классификации ниже и составляет 0.278. Таким образом, выбор способа масштабирования данных имеет влияние на точность классификации. В данном случае, стандартное масштабирование и минимакс-масштабирование показали лучшие результаты.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | X\_train, X\_test, y\_train, y\_test = load\_data(0.3)  scaled\_x = scale(X\_train)  scaled\_x\_mm = scale(X\_train, mode='minmax')  scaled\_x\_abs = scale(X\_train, mode='maxabs')  c1 = train\_model(scaled\_x, y\_train, 9)  c3 = train\_model(scaled\_x\_mm, y\_train, 9)  c5 = train\_model(scaled\_x\_abs, y\_train, 9)  r1 = predict(c1, X\_test)  r3 = predict(c3, X\_test)  r5 = predict(c5, X\_test)  e1 = estimate(r1, y\_test)  e3 = estimate(r3, y\_test)  e5 = estimate(r5, y\_test)  print(e1, e3, e5) | 0.368 0.392 0.384 | Корректно |

## Выводы

Были изучены основные инструменты анализа данных в языке Python. Получены навыки работы с библиотекой sklearn.

# Приложение А Исходный код программы

Название файла: Murdasov\_Mikhail\_lb3.py

from sklearn.datasets import load\_wine

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

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

    wine = load\_wine()

    x = wine.data

    y = wine.target

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(x[:, [0,1]], 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(classifier, X\_test):

    pred = classifier.predict(X\_test)

    return pred

def estimate(res, y\_test):

    return round(accuracy\_score(y\_true = y\_test, y\_pred = 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)