**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 |  | Хангулян С. К. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Целью работы является изучение основ анализа данных и ознакомление и работа с библиотекой sklearn 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, X\_test, y\_train, y\_test.

Пояснение: X\_train, X\_test - двумерный массив, y\_train, y\_test. — одномерный массив.

1. 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.

1. 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.

1. 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].

1. 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()** инициирует процесс загрузки информации о винах через библиотеку sklearn и использует функцию train\_test\_split() для разделения на наборы для обучения и тестирования. Параметр train\_size размер данных для обучения, которая по умолчанию составляет 0.8.

Функция **train\_model()** на основе обучающего набора X\_train и соответствующих ему меток y\_train проводит тренировку модели классификации методом k-ближайших соседей (KNeighborsClassifier). Параметр n\_neighbors устанавливает число соседей, а weight - тип весовой функции для модели. Значения по умолчанию: n\_neighbors установлен в 15, а weight - в 'uniform'.

Функция **predict()** осуществляет прогнозирование классов на основе тестового набора данных X\_test.

Функция **estimate()** рассчитывает и возвращает точность классификации, используя функцию accuracy\_score() из пакета sklearn.metrics, результат округляется до трех знаков после запятой.

Функция **scale()** принимает на вход массив данных X и параметр mode для выбора режима масштабирования, возвращая результат в виде масштабированного массива. Допустимые значения для mode включают 'standard', 'minmax', 'maxabs'. В случае недопустимого значения mode возвращает None. При mode равном 'standard' используется стандартное масштабирование (StandardScaler), 'minmax' применяет мин-макс масштабирование (MinMaxScaler), а 'maxabs' - масштабирование по максимальному абсолютному значению (MaxAbsScaler), используя соответствующие классы из sklearn.preprocessing.

При исследование работы классификатора, обученного на данных разного размера, выясняется, что эффективность классификационной модели коррелирует с размером используемой выборки. Когда выборка слишком мала, это может привести к недостаточному обучению. В то же время, чрезмерно большая выборка может содержать избыточные данные, что затрудняет процесс классификации и снижает общую точность модели.

|  |  |
| --- | --- |
| load\_data | Точность работы |
| *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 |

При исследование работы классификатора, обученного с различными значениями *n\_neighbors*, становится понятно, что количество соседей почти не влияет на точность работы классификатора.

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

При исследование работы классификатора с предобработанными данными становится очевидным, что методы масштабирования оказывают значительное влияние на точность классификационных моделей. Когда применяется стандартное масштабирование (StandardScaler) или масштабирование мин-макс (MinMaxScaler), точность достигает **0.417**. Использование максимального абсолютного масштабирования (MaxAbsScaler) приводит к снижению точности до **0.278**. Это демонстрирует, что выбор метода масштабирования важен для достижения оптимальной точности классификации.

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

**Тестирование**

Результаты тестирования представлены в таблице 1.

Таблица 1 – Результаты тестирования

|  |  |
| --- | --- |
| Тест | Комментарии |
| X\_train, X\_test, y\_train, y\_test = load\_data()  clf = train\_model(X\_train, y\_train)  res = predict(clf, X\_test)  est = estimate(res, y\_test)  print(X\_train)  print(y\_train)  print(X\_test)  print(y\_test)  print(clf)  print(res)  print(est)  # scaling  scaled\_standart = scale(X\_train)  scaled\_standart1 = scale(X\_train, mode='standard')  print(scaled\_standart == scaled\_standart1)  print(scale(X\_test, mode='my\_mode'))  print(scale(X\_test, mode='minmax'))  print(scale(X\_test, mode='maxabs')) | Корректно |

**Выводы**

Были изучены основы анализа данных и библиотека sklearn, позволяющая непосредственно выполнять анализ данных на python.

# ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Khangulyan\_Sargis\_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\_train, X\_test, y\_train, y\_test = train\_test\_split(wine.data[:, [0,1]], wine.target, 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)

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

def predict(classifier, X\_test):

    return classifier.predict(X\_test)

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 == 'maxabs':

        scaler = MaxAbsScaler()

    elif mode == 'minmax':

        scaler = MinMaxScaler()

    else:

        return None

    return scaler.fit\_transform(data)