**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»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3342 |  | Епонишникова А.И. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Познакомиться с базовыми принципами анализа данных. Изучить 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

load\_data(train\_size=0.8):

В переменную wine загружаем набор данных из datasets библиотеки sklearn. Далее разделяем выборку на обучающую (по умолчанию 0.8) и тестовую с помощью train\_test\_split с параметром random\_state = 42.

train\_model(X\_train, y\_train, n\_neighbors = 15, weights = ‘uniform’):

функция train\_model предназначена для обучения модели на основе переданных данных X\_train и меток y\_train. Внутри функции создается объект классификатора KNeighborsClassifier с заданными параметрами n\_neighbors и weights (по умолчанию 15 и ‘uniform’ соответственно). Объект классификатора обучается на переданных данных X\_train и метках y\_train с использованием метода fit. Функция возвращает обученную модель.

predict(clf, X\_test):

Функция использует метод predict модели clf, чтобы сделать прогнозы для набора тестовых данных X\_test. Функция возвращает массив прогнозов, где каждый элемент представляет собой предсказанное значение для соответствующего элемента в X\_test.

estimate(res, y\_test):

Функция использует функцию accuracy\_score для сравнения предсказанных значений res с фактическими значениями y\_test. Функция возвращает оценку точности модели, которая представляет собой долю правильных предсказаний модели на тестовом наборе данных.

scale(data, mode = 'standard'):

В зависимости от выбранного режима масштабирования (mode), функция создает соответствующий объект масштабирования: StandardScaler, MinMaxScaler или MaxAbsScaler. Затем функция использует этот объект для выполнения масштабирования данных с помощью метода fit\_transform. Функция возвращает масштабированные данные в соответствии с выбранным режимом.

Разработанный программный код см. в приложении А.

## Тестирование

Таблица 1 - Исследование работы классификатора, обученного на данных разного размера

|  |  |  |
| --- | --- | --- |
| № | train\_size | accuracy |
| 1. | 0.1 | 0.379 |
| 2. | 0.3 | 0.8 |
| 3. | 0.5 | 0.843 |
| 4. | 0.7 | 0.815 |
| 5. | 0.9 | 0.722 |

Таблица 2 - Исследование работы классификатора, обученного с различными значениями n\_neighbors

|  |  |  |
| --- | --- | --- |
| № | n\_neighbors | accuracy |
| 1. | 3 | 0.861 |
| 2. | 5 | 0.833 |
| 3. | 9 | 0.861 |
| 4. | 15 | 0.861 |
| 5. | 25 | 0.833 |

Таблица 3 - Исследование работы классификатора с предобработанными данными

|  |  |  |
| --- | --- | --- |
| № | sacler | accuracy |
| 1. | StandardScaler | 0.889 |
| 2. | MinMaxScaler | 0.806 |
| 3. | MaxAbsScaler | 0.75 |

## Выводы

Исходя из таблицы 1 видно, что оптимальная точность достигается при train\_size от 0.5 - 0.7. Если же train\_size = 0.1, то модели не хватает данных на обучение, поэтому точность низкая. Если train\_size = 0.9, то размер тестовой выборке слишком маленький, что приводит к неустойчивости оценки качества, то есть небольшие изменения в тестовой выборке могут привести к значительным изменениям в оценке качества модели.

Из таблицы 2 видно, что модель демонстрирует стабильность при больших изменениях в алгоритме "К ближайших соседей". Это свидетельствует о том, что исходные данные хорошо соответствуют модели, и их легко разделить на классы даже визуально.

Из таблицы 3 можно сделать вывод, что применение StandardScaler приводит к улучшению точности прогнозов модели, что делает его использование в данном контексте обоснованным. В то же время MinMaxScaler и MaxAbsScaler либо слабо влияют на точность, либо незначительно снижают её. Экспериментальным путем выяснено, что в условиях данной задачи следует использовать StandardScaler.

Были изучены основные принципы анализа данных в Python и разработана программа, использующая их. Реализованы методы для разделения данных для обучения и тестирования, создания экземпляра классификатора соседей, предсказания данных и оценки качества результатов классификации.

ПРИЛОЖЕНИЕ А  
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.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

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

wine = datasets.load\_wine()

y = wine.target

x = wine.data[:,:2]

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'):

return KNeighborsClassifier(n\_neighbors = n\_neighbors, weights=weights).fit(X\_train, y\_train)

def predict(clf, X\_test):

return clf.predict(X\_test)

def estimate(res, y\_test):

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

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

if mode == "standard":

transformer = preprocessing.StandardScaler()

elif mode == 'minmax':

transformer = preprocessing.MinMaxScaler()

elif mode == 'maxabs':

transformer = preprocessing.MaxAbsScaler()

else:

return None

return transformer.fit\_transform(data)