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. Вариант 1

Студент гр. 3343 _____ Гребнев Е.Д.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург
2024

Цель работы

Цель данной лабораторной работы состоит в исследовании применения метода k-ближайших соседей для классификации данных о винах. Цели включают:

1. Ознакомление с основными концепциями метода k-ближайших соседей (kNN).
2. Понимание процесса загрузки данных, их разделения на обучающую и тестовую выборки.
3. Исследование влияния различных параметров на точность работы классификатора:
    - Размера обучающей выборки.
    - Количества соседей (значения k).
    - Предобработки данных с использованием различных скейлеров.
4. Оценка и сравнение результатов классификации при различных настройках параметров.
5. Понимание принципов выбора оптимальных параметров для повышения точности работы модели.
6. Оформление отчёта, включающего описание реализованных функций, результаты экспериментов и их анализ.

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 с 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

Описание реализации функций

1. **load_data(train_size=0.8, random_state=42):**
   - Эта функция загружает набор данных о вине из библиотеки sklearn.
   - Разбивает данные на обучающую и тестовую выборки в соответствии с заданным размером обучающей выборки.
   - Возвращает четыре массива: X_train, X_test, y_train, y_test.
2. **train_model(X_train, y_train, n_neighbors=15, weights='uniform'):**
   - Создает экземпляр классификатора KNeighborsClassifier с заданными параметрами.
   - Обучает классификатор на обучающей выборке (X_train, y_train).
   - Возвращает обученный классификатор.
3. **predict(clf, X_test):**
   - Принимает обученную модель классификатора и тестовую выборку (X_test).
   - Выполняет классификацию данных из X_test с помощью обученной модели.
   - Возвращает предсказанные метки классов.
4. **estimate(predictions, y_test):**
   - Принимает предсказанные метки классов и истинные метки тестовых данных (y_test).
   - Вычисляет точность работы модели как отношение предсказанных результатов, совпавших с "правильными", к общему количеству результатов.
   - Возвращает полученное отношение, округленное до трех знаков после запятой.
5. **scale(data, mode='standard'):**
   - Принимает аргумент, содержащий данные, и тип скейлера для предобработки данных.

– Обрабатывает данные соответствующим скейлером.
– Возвращает обработанные данные.

Исследование работы классификатора

1. Классификация данных с разным размером обучающей выборки

Таблица 1. Результаты исследования работы классификатора для данных разного размера.

Размер *train_size*	Точность
0.1	0.379
0.3	0.8
0.5	0.843
0.7	0.815
0.9	0.722

Вывод: Точность классификации достигает максимума при размере обучающей выборки 0.8 (по умолчанию). При слишком маленьком размере выборки модель недостаточно обучается, а при слишком большом может наблюдаться переобучение.

2. Классификация данных с различными значениями k-ближайших соседей

Таблица 2. Результаты исследования работы классификатора для различных значений *n_neighbors*.

Размер *n_neighbors*	Точность
3	0.861
5	0.833
9	0.861
15	0.861

25	0.833

Вывод: Точность классификации не сильно зависит от значения k. Однако, при слишком больших значениях k точность может начать снижаться из-за увеличения числа соседей, что может привести к потере гибкости модели.

3. Классификация данных с предобработкой

Таблица 3. Результаты исследования работы классификатора для разных скейлеров.

Скейлер	Точность
StandardScaler	0.889
MinMaxScaler	0.806
MaxAbsScaler	0.75

Вывод: Наилучшую точность показал StandardScaler, что может быть связано с тем, что он центрирует и масштабирует данные так, чтобы они имели стандартное отклонение 1 и среднее значение 0. MinMaxScaler и MaxAbsScaler также показали приемлемые результаты, но несколько хуже.

Выводы

Была разработана программа для классификации данных о вине с использованием метода k-ближайших соседей. Проведено исследование работы классификатора при различных параметрах, таких как размер обучающей выборки, количество соседей и предобработка данных. Полученные результаты позволяют выбирать оптимальные параметры для обучения модели с наилучшей точностью.

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.preprocessing import StandardScaler, MinMaxScaler,
MaxAbsScaler

def load_data(train_size=0.8, random_state=42):
    wine = datasets.load_wine()
    X_train, X_test, y_train, y_test = train_test_split(wine.data[:, :2],
wine.target, train_size=train_size, random_state=random_state)
    return X_train, X_test, y_train, y_test

def train_model(X_train, y_train, n_neighbors=15, weights='uniform'):
    clf = KNeighborsClassifier(n_neighbors=n_neighbors, weights=weights)
    clf.fit(X_train, y_train)
    return clf

def predict(clf, X_test):
    return clf.predict(X_test)

def estimate(predictions, y_test):
    return round((predictions == y_test).mean(), 3)

def scale(data, mode='standard'):
    scalers = {
        'standard': StandardScaler(),
        'minmax': MinMaxScaler(),
        'maxabs': MaxAbsScaler()
    }
    scaler = scalers.get(mode)
    return scaler.fit_transform(data) if scaler else None
```