

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Информационные технологии»
Тема: Введение в анализ данных

Студент гр. 3343

Отмахов Д. В.

Преподаватель

Иванов Д. В.

Санкт-Петербург

2024

Цель работы

Введение в анализ данных, изучение базовых аспектов работы библиотеки scikit-learn. Написание программы, обучающей модель и выполняющей с ней определенные функции.

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.

В отчёте приведите (чек-лист преподавателя):

- описание реализации 5и требуемых функций

- исследование работы классификатора, обученного на данных разного размера
 - приведите точность работы классификаторов, обученных на данных от функции *load_data* со значением аргумента *train_size* из списка: 0.1, 0.3, 0.5, 0.7, 0.9
 - оформите результаты пункта выше в виде таблицы
 - объясните полученные результаты
- исследование работы классификатора, обученного с различными значениями *n_neighbors*
 - приведите точность работы классификаторов, обученных со значением аргумента *n_neighbors* из списка: 3, 5, 9, 15, 25
 - в качестве обучающих/тестовых данных для всех классификаторов возьмите результат *load_data* с аргументами по умолчанию (учтите, что для достоверности результатов обучение и тестирование классификаторов должно проводиться на одних и тех же наборах)
 - оформите результаты в виде таблицы
 - объясните полученные результаты
- исследование работы классификатора с предобработанными данными
 - приведите точность работы классификаторов, обученных на данных предобработанных с помощью скейлеров из списка: *StandardScaler*, *MinMaxScaler*, *MaxAbsScaler*
 - в качестве обучающих/тестовых данных для всех классификаторов возьмите результат *load_data* с аргументами по умолчанию - учтите, что для достоверности сравнения результатов классификации обучение должно проводиться на одних и тех же данных, поэтому предобработку следует производить после разделения на обучающую/тестовую выборку.
 - оформите результаты в виде таблицы

- объясните полученные результаты

Выполнение работы

Программа состоит из пяти функций:

- *load_data(train_size = 0.8)* – загружает набор данных о вине из *load_wine()* в переменную *wine* и составляет на основе данных обучающую и тестовую выборку в соответствии с *train_size*.
- *train_model(X_train, y_train, n_neighbors = 15, weights = 'uniform')* – обучает модель: принимает на вход обучающую выборку и аргументы *n_neighbors* и *weights*, создает экземпляр классификатора *KNeighborsClassifier* с параметрами *n_neighbors* и *weights* и данными *X_train* и *y_train*.
- *predict(clf, X_test)* – принимает обученную модель классификатора и тренировочный набор данных, предсказывает данные тренировочного набора *X_test* используя обученную модель *clf*.
- *estimate(res, y_test)* – принимает результаты классификации *res* и истинные метки тестовых данных *y_test*, определяет качество обученной модели (отношение предсказанных результатов, совпавших с «правильными» в *y_test* к общему количеству результатов). Для сравнения используется функция *accuracy_score*, возвращающая результат в пределах [0, 1].
- *scale(data, mode='standard')* – принимает данные *data* и тип скейлера *mode*, обрабатывает тренировочные данные на основе выбранного скейлера.

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

Размер <i>train_size</i>	Точность
0.1	0.379
0.3	0.8
0.5	0.843
0.7	0.815

0.9	0.722
-----	-------

На основе результатов табл. 1 можно сделать следующие выводы: при $train_size = 0.8$ (дефолтном) достигается самая высокая точность работы; при очень маленьких значениях $train_size$ – самая маленькая точность, так как модели не хватает данных для обучения; при очень больших значениях $train_size$ – точность также уменьшается, так как происходит переобучение модели.

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

На основе результатов табл. 2 можно сделать следующий вывод: значение $n_neighbors$ не сильно влияет на точность работы классификатора. Также можно предположить, что при сильно большом значении $n_neighbors$, точность работы уменьшится, так как границы классификации станут менее четкими.

Таблица 3. Результаты исследования работы классификатора для разных скейлеров.

Скейлер	Точность
<i>StandardScaler</i>	0.889
<i>MinMaxScaler</i>	0.806
<i>MaxAbsScaler</i>	0.75

На основе результатов табл. 3 можно сделать следующие выводы: максимальная точность достигается при использовании *StandardScaler*; минимальная – при использовании *MaxAbsScale*. Это объясняется тем, что в отличие от *StandardScaler*, *MinMaxScale* и *MaxAbsScaler* чувствительны к выбросам.

Выводы

В ходе выполнения лабораторной работы были изучены базовые аспекты работы библиотеки `scikit-learn`, написана программа, обучающая модель и выполняющая с ней определенные функции.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.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,
wine.target, train_size = train_size, random_state = 42)
    return X_train[:, :2], X_test[:, :2], 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(res, y_test), 3)

def scale(data, mode = 'standard'):
    if mode not in ['standard', 'minmax', 'maxabs']:
        return None

    if mode == 'standard':
        scale = StandardScaler().fit_transform(data)
    elif mode == 'minmax':
        scale = MinMaxScaler().fit_transform(data)
    else:
        scale = MaxAbsScaler().fit_transform(data)

    return scale
```