

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

Бубякина Ю.В.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Введение в анализ данных. Ознакомление с базовыми концепциями и инструментами анализа данных на языке Python. Написание кода для лабораторной работы номер 3.

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

Функция `load_data` загружает набор данных о вине из библиотеки `sklearn`, разбивает его на обучающую и тестовую выборки с помощью функции `train_test_split` из модуля `sklearn.model_selection` и возвращает четыре массива: `X_train`, `X_test`, `y_train` и `y_test`. Параметр `train_size` определяет долю данных, отведенных для обучающей выборки, и по умолчанию равен 0.8. Функция `train_model` обучает модель классификатора К-ближайших соседей (`KNeighborsClassifier`) на обучающей выборке `X_train` с метками `y_train`. Параметры `n_neighbors` и `weights` задают количество соседей и весовую функцию для классификатора. По умолчанию `n_neighbors` равно 15, а `weights` установлено в `'uniform'`.

Функция `predict` принимает обученную модель классификатора и тестовую выборку `X_test`, предсказывает метки для этой выборки и возвращает массив предсказанных меток.

Функция `estimate` принимает массив предсказанных меток `res` и массив истинных меток `y_test`, вычисляет точность классификации с помощью функции `accuracy_score` из модуля `sklearn.metrics` и возвращает значение точности, округленное до трех знаков после запятой.

Функция `scale` принимает массив данных `X` и режим масштабирования `mode`, возвращая масштабированный массив данных. Допустимые значения для параметра `mode`: `'standard'`, `'minmax'` и `'maxabs'`. Если `mode` имеет недопустимое значение, функция возвращает `None`. При `mode='standard'` выполняется стандартное масштабирование с использованием `StandardScaler`, при `mode='minmax'` — мини-максимальное масштабирование с использованием `MinMaxScaler`, а при `mode='maxabs'` — масштабирование по максимальному абсолютному значению с использованием `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

Из полученных результатов видно, что точность классификации зависит от размера выборки. Слишком маленькая выборка (0.1) приводит к низкой точности классификации (0.379) из-за недостаточного количества данных для обучения модели. С увеличением размера выборки точность классификации увеличивается, достигая максимума при размере выборки 0.5 (0.843). Однако дальнейшее увеличение размера выборки не приводит к значительному улучшению точности классификации. При размере выборки 0.9 точность даже снижается до 0.722. Таким образом, можно заключить, что слишком большая выборка может быть неэффективна для классификации, так как может приводить к переобучению модели и увеличению времени обучения.

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

Из полученных результатов видно, что точность работы классификаторов с разными значениями `n_neighbors` отличается незначительно. Наивысшая точность достигается при значениях `n_neighbors`, равных 3, 9 и 15, и составляет 0.861. При значениях `n_neighbors`, равных 5 и 25, точность немного ниже и составляет 0.833. Таким образом, можно заключить, что для данного набора данных оптимальными значениями `n_neighbors` являются 3, 9 или 15, однако разница в точности незначительна.

Исследование работы классификатора с предобработанными данными:

Метод предобработки	Точность работы классификатора
StandardScaler	0.417
MinMaxScaler	0.417
MaxAbsScaler	0.278

Результаты показывают, что точность классификации варьируется в зависимости от выбранного метода масштабирования данных. При применении стандартного масштабирования (StandardScaler) и минимакс-масштабирования (MinMaxScaler) точность составляет 0.417, в то время как при использовании максимального абсолютного масштабирования (MaxAbsScaler) точность снижается до 0.278. Таким образом, выбор метода масштабирования данных может существенно влиять на точность классификации. В данном случае стандартное масштабирование и минимакс-масштабирование показали наилучшие результаты.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	<pre>X_train, X_test, y_train, y_test = load_data(0.7) 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)</pre>	0.37 0.389 0.463	-

Выводы

Были получены базовые знания о базовых концепциях и инструментах анализа данных на языке Python и написан код для лабораторной работы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.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.preprocessing import StandardScaler, MinMaxScaler,
MaxAbsScaler

def load_data(train_size=0.8):
    wine_dataset = datasets.load_wine()
    data = wine_dataset.data
    target = wine_dataset.target
    X_train, X_test, y_train, y_test = train_test_split(
        data[:, [0, 1]], 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).fit(X_train, y_train)
    return classifier

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

def estimate(res, y_test):
    accuracy = accuracy_score(y_true=y_test, y_pred=res)
    return round(accuracy, 3)

def scale(X, mode="standard"):
    if mode not in ["standard", "minmax", "maxabs"]:
```

```
        return None
    scaler = StandardScaler()
    if mode == "minmax":
        scaler = MinMaxScaler()
    elif mode == "maxabs":
        scaler = MaxAbsScaler()
    scaled = scaler.fit_transform(X)
    return scaled
```