

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»

Студент гр. 3342

Гончаров С. А.

Преподаватель

Иванов Д. В.

Санкт-Петербург

2024

Цель работы

Изучить основные принципы анализа данных и освоить основные инструменты для их обработки и анализа.

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` 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(train_size=0.8)` загружает набор данных о вине из библиотеки `sklearn.datasets`. Извлекает первые два столбца из данных (`[:, :2]`) и разделяет данные на обучающую и тестовую выборки с заданным размером обучающей выборки (`train_size`).

Функция `train_model(X_train, y_train, n_neighbors=15, weights='uniform')` обучает модель классификации методом k-ближайших соседей (`KNeighborsClassifier`). Принимает параметры (`n_neighbors`, `weights`) для настройки классификатора.

Функция `predict(clf, X_test)` использует обученный классификатор для предсказания меток тестовых данных.

Функция `estimate(res, y_test)` оценивает качество предсказаний, вычисляя долю правильных ответов. Возвращает точность, округленную до трех знаков после запятой.

Функция `scale(data, mode='standard')` выполняет предобработку данных с использованием одного из трех типов скейлеров: “`StandardScaler`”, “`MinMaxScaler`” или “`MaxAbsScaler`”.

Разработанный программный код см. в приложении А.

Тестирование

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

№	train_size	accuracy
1.	0.1	0.444
2.	0.3	0.556
3.	0.5	0.667
4.	0.7	0.778
5.	0.9	0.889

Таблица 2 - Исследование работы классификатора, обученного с различными значениями n_neighbors

№	n_neighbors	accuracy
1.	3	0.667
2.	5	0.722
3.	9	0.778
4.	15	0.778
5.	25	0.722

Таблица 3 - Исследование работы классификатора с предобработанными данными

№	scaler	accuracy
1.	Standard	0.778
2.	MinMax	0.778
3.	MaxAbs	0.778

Выводы

Таблица 1 показывает, что с увеличением размера обучающей выборки точность классификатора растет, достигая максимума при размере выборки 0.7 и более.

Таблица 2 показывает, что при увеличении значения `n_neighbors`, точность сначала растет, достигая максимума, а затем начинает снижаться. Слишком малое значение `n_neighbors` приводит к переобучению, в то время как слишком большое значение – к недообучению.

Таблица 3 показывает, что различные методы масштабирования данных дали одинаковую точность. Это может быть связано с тем, что данные имеют сходные масштабы, и нормализация существенно не влияет на результат.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
MaxAbsScaler

def load_data(train_size=0.8):
    wine = datasets.load_wine()
    X = wine.data[:, :2]
    y = wine.target
    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'):
    clf = KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights)
    clf.fit(X_train, y_train)

    return clf

def predict(clf, X_test):
    predictions = clf.predict(X_test)

    return predictions

def estimate(res, y_test):
    accuracy = (res == y_test).mean()

    return round(accuracy, 3)

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

    if mode == 'standard':
```



```
        scaler = StandardScaler()
elif mode == 'minmax':
        scaler = MinMaxScaler()
elif mode == 'maxabs':
        scaler = MaxAbsScaler()
else:
        return None

scaled_data = scaler.fit_transform(data)

return scaled_data
```