

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. 3341

Мокров И.О.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Целью работы является изучение основ анализа данных и написание программы на языке Python, анализирующей и классифицирующей данные с помощью библиотеки *sklearn*.

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`, `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.

В отчёте приведите (чек-лист преподавателя):

- описание реализации 5и требуемых функций

- исследование работы классификатора, обученного на данных разного размера
 - о 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
 - о оформите результаты пункта выше в виде таблицы
 - о объясните полученные результаты
- исследование работы классификатора, обученного с различными значениями `n_neighbors`
 - о приведите точность работы классификаторов, обученных со значением аргумента `n_neighbors` из списка: 3, 5, 9, 15, 25
 - о 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)
 - о оформите результаты в виде таблицы
 - о объясните полученные результаты
- исследование работы классификатора с предобработанными данными
 - о приведите точность работы классификаторов, обученных на данных предобработанных с помощью скейлеров из списка: `StandardScaler`, `MinMaxScaler`, `MaxAbsScaler`
 - о 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.
 - о оформите результаты в виде таблицы
 - о объясните полученные результаты

Выполнение работы

1. Функция `load_data` загружает данные из встроенного датасета `load_wine`, используя только первые 2 колонки в качестве признаков `X` и метки классов `y`. Затем данные разбиваются на обучающую и тестовую выборки с разделением, заданным параметром `train_size`, и возвращает эти данные.

2. Функция `train_model` обучает модель классификации ближайших соседей (`KNeighborsClassifier`) с заданными параметрами `n_neighbors` и `weights`, используя обучающие данные.

3. Функция `predict` предсказывает метки классов для тестовых данных, используя обученную модель.

4. Функция `estimate` оценивает точность модели путем сравнения предсказанных меток `res` и реальных меток `y_test` с помощью `accuracy_score`. Результат округляется до трех знаков после запятой и возвращается.

5. Функция `scale` использует различные методы масштабирования данных, такие как `StandardScaler`, `MinMaxScaler` и `MaxAbsScaler` в зависимости от значения параметра `mode`. Возвращает преобразованные данные.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 4.

Таблица 4 – Результаты тестирования

| № п/п | Входные данные | Выходные данные | Комментарии |
|----------|--|-----------------|-----------------------|
| 1. | <pre>X_train, X_test, y_train, y_test = load_data() clf = train_model(X_train, y_train) res = predict(clf, X_test) est = estimate(res, y_test) print(est)</pre> | 0.861 | Стандартное обучение |
| 2. | <pre>X_train, X_test, y_train, y_test = load_data() X_train_scaled = scale(X_train, 'minmax') X_test_scaled = scale(X_test, 'minmax') clf = train_model(X_train_sca led, y_train) res = predict(clf, X_test_scaled) est = estimate(res, y_test) print(est)</pre> | 0.806 | Обучение со скейлером |

Выводы

В ходе выполнения работы мы изучили основы анализа данных на языке Python с применением библиотеки *sklearn*. Были разработаны функции для выгрузки данных, обучения модели, применения модели, оценки её эффективности и предобработки данных. Так же мы проанализировали точность работы моделей при различных условиях обучения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
MaxAbsScaler
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

def load_data(train_size=0.8):
    wine=load_wine()
    X=wine.data[:, 0:2]
    y=wine.target
    X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=train_size, test_size=1-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):
    y_pred = clf.predict(X_test)
    return y_pred

def estimate(res, y_test):
    accur = accuracy_score(y_test, res)
    accur = round(accur, 3)

    return accur

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
```