

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

Лучкин М. А.

Преподаватель

Иванов Д. В.

Санкт-Петербург

2024

Цель работы

Изучить основные принципы анализа данных и освоить ключевые инструменты для их обработки и анализа. Овладеть навыками работы с данными, включая сбор, очистку и интерпретацию результатов.

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

``load_data()`` функция загружает данные о винах из библиотеки `sklearn.datasets` и делит их на обучающий и тестовый наборы. Сначала данные о винах загружаются, затем ``train_test_split`` функция используется для разделения данных на обучающие и тестовые с заданным размером обучающего набора по умолчанию (0.8). Функция возвращает обучающие и тестовые данные.

``train_model()`` функция обучает модель классификации методом k-ближайших соседей (k-Nearest Neighbors, KNN) на обучающих данных. Сначала создается экземпляр классификатора KNN с определенными параметрами ``n_neighbors`` и ``weights``. Затем модель обучается на обучающих данных ``X_train`` и их метках ``y_train`` с использованием метода ``fit()``, после чего возвращает обученную модель.

``predict()`` функция предсказывает классы для тестовых данных с использованием обученной модели классификатора. С помощью метода ``predict()`` делается прогноз классов для тестовых данных ``X_test``.

``estimate()`` функция оценивает точность модели классификации, сравнивая предсказанные классы с истинными классами тестового набора данных. Точность модели вычисляется с использованием метода ``accuracy_score()``, сравнивая истинные метки классов ``y_test`` с предсказанными метками ``res`` и округляя результат до трех знаков после запятой.

``scale()`` функция масштабирует данные с помощью выбранного метода нормализации, в зависимости от переданного режима. В зависимости от значения ``mode`` выбирается соответствующий метод масштабирования с использованием конструкции `if-elif-else`, затем выбранный метод применяется к данным с использованием метода ``scaler.fit_transform(data)``.

Разработанный программный код см. в приложении А.

Тестирование

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

№	train_size	accuracy
1.	0.1	0.778
2.	0.3	0.839
3.	0.5	0.889
4.	0.7	0.944
5.	0.9	0.972

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

№	n_neighbors	accuracy
1.	3	0.944
2.	5	0.972
3.	9	0.972
4.	15	0.972
5.	25	0.944

Таблица 3 - Исследование работы классификатора с предобработанными данными

№	sacler	accuracy
1.	StandardScaler	0.972
2.	MinMaxScaler	0.972
3.	MaxAbsScaler	0.972

Выводы

Из данных в таблице 1 видно, что увеличение размера обучающей выборки приводит к повышению точности классификатора, достигая пика при размере выборки 0.7 и более.

Анализ результатов в таблице 2 показывает, что увеличение значения параметра `n_neighbors` улучшает точность классификации до определенного значения (например, при `n_neighbors = 5, 9` и `15`), после чего дальнейшее увеличение приводит к снижению точности.

Результаты из таблицы 3 показывают, что различные методы масштабирования данных не оказывают значительного влияния на точность классификации. Возможно, это связано с тем, что выбранные признаки (индексы 1 и 2) уже нормализованы и не требуют дополнительной обработки.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

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

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn import datasets

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)

    return clf.fit(X_train, y_train)

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

def estimate(pred, y_test):
    accuracy = accuracy_score(y_test, pred)
    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

    return scaler.fit_transform(data)
```