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: «Алгоритмы и структуры данных в Python»

Студент гр. 3342 _____ Корниенко А. Е.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Изучить основы анализа данных и машинного обучения, освоить основные инструменты для обработки и анализа данных.

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

Описание функций:

1. Загрузка данных:

   Функция load_data() загружает данные набора Wine из sklearn.datasets. Разделяет данные на обучающую и тестовую выборки с заданным split_ratio, после чего возвращает обучающие и тестовые наборы признаков и меток.

2. Тренировка модели:

   Функция train_model() создает экземпляр KNeighborsClassifier с заданными n_neighbors и weights. Обучает модель, после чего возвращает обученную модель.

3. Предсказание:

   Функция predict() предсказывает значение x_test. Возвращает вектор предсказанных меток.

4. Оценка:

   Происходит предварительная обработки данных, которая преобразует признаки в заданный масштаб для обеспечения более стабильного обучения модели по одному из следующих алгоритмов..

5. Масштабирование данных:

   Происходит предварительная обработки данных, которая преобразует признаки в заданный масштаб для обеспечения более стабильного обучения модели по одному из алгоритмов(StandardScaler, MinMaxScaler, MaxAbsScaler).

Исследование работы классификатора, обученного на данных разного размера:

Размер обучающего набора	Точность
0.1	0.545
0.3	0.725
0.5	0.861
0.7	0.915
0.9	0.924

Исследование работы классификатора, обученного с различными значениями n_neighbors:

Значение n_neighbors	Точность
3	0.861
5	0.889
9	0.921
15	0.935
25	0.919

Исследование работы классификатора с предобработанными данными:

Метод предобработки	Точность
Без предобработки	0.878
StandardScaler	0.945
MinMaxScaler	0.935

MaxAbsScaler	0.925

Разработанный программный код см. в приложении А.

Выводы

В данной работе была разработана программа, которая обучает модель для предсказания классов вин. Также были проведены с ней некоторые исследования.

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 import preprocessing
from sklearn.metrics import accuracy_score


def load_data(split_ratio=0.8):
    data_wine = load_wine()
    x = data_wine.data[:, :2]
    y = data_wine.target

    x_train, x_test, y_train, y_test = train_test_split(
        x, y, train_size=split_ratio, random_state=42
    )

    return x_train, x_test, y_train, y_test


def train_model(x_train, y_train, neighbors=15, weights='uniform'):
    knn_model = KNeighborsClassifier(n_neighbors=neighbors,
weights=weights)
    knn_model.fit(x_train, y_train)

    return knn_model


def predict(clf, x_test):
    PredictData = clf.predict(x_test)

    return PredictData


def estimate(res, y_test):
```

```
    return round(accuracy_score(y_test, res), 3)


def scale(data, mode='standard'):
    if mode == 'standard':
        Scaler = preprocessing.StandardScaler()
    elif mode == 'minmax':
        Scaler = preprocessing.MinMaxScaler()
    elif mode == 'maxabs':
        Scaler = preprocessing.MaxAbsScaler()
    else:
        return None

    return Scaler.fit_transform(data)
```