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 _____ Хангулян С. К.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург
2024

Цель работы

Целью работы является изучение основ анализа данных и ознакомление и работа с библиотекой sklearn на python.

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.

Выполнение работы

Функция **load_data()** инициирует процесс загрузки информации о винах через библиотеку sklearn и использует функцию train_test_split() для разделения на наборы для обучения и тестирования. Параметр train_size размер данных для обучения, которая по умолчанию составляет 0.8.

Функция **train_model()** на основе обучающего набора X_train и соответствующих ему меток y_train проводит тренировку модели классификации методом k-ближайших соседей (KNeighborsClassifier). Параметр n_neighbors устанавливает число соседей, а weight - тип весовой функции для модели. Значения по умолчанию: n_neighbors установлен в 15, а weight - в 'uniform'.

Функция **predict()** осуществляет прогнозирование классов на основе тестового набора данных X_test.

Функция **estimate()** рассчитывает и возвращает точность классификации, используя функцию accuracy_score() из пакета sklearn.metrics, результат округляется до трех знаков после запятой.

Функция **scale()** принимает на вход массив данных X и параметр mode для выбора режима масштабирования, возвращая результат в виде масштабированного массива. Допустимые значения для mode включают 'standard', 'minmax', 'maxabs'. В случае недопустимого значения mode возвращает None. При mode равном 'standard' используется стандартное масштабирование (StandardScaler), 'minmax' применяет мин-макс масштабирование (MinMaxScaler), а 'maxabs' - масштабирование по максимальному абсолютному значению (MaxAbsScaler), используя соответствующие классы из sklearn.preprocessing.

При исследование работы классификатора, обученного на данных разного размера, выясняется, что эффективность классификационной модели коррелирует с размером используемой выборки. Когда выборка слишком мала, это может привести к недостаточному обучению. В то же время, чрезмерно большая выборка может содержать избыточные данные, что затрудняет процесс классификации и снижает общую точность модели.

load_data	Точность работы
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

При исследование работы классификатора, обученного с различными значениями *n_neighbors*, становится понятно, что количество соседей почти не влияет на точность работы классификатора.

n_neighbors	Точность работы
3	0.861
5	0.833
9	0.861
15	0.861
25	0.833

При исследование работы классификатора с предобработанными данными становится очевидным, что методы масштабирования оказывают значительное влияние на точность классификационных моделей. Когда применяется стандартное масштабирование (StandardScaler) или масштабирование мин-макс (MinMaxScaler), точность достигает **0.417**. Использование максимального абсолютного масштабирования (MaxAbsScaler)

приводит к снижению точности до **0.278**. Это демонстрирует, что выбор метода масштабирования важен для достижения оптимальной точности классификации.

Метод предобработки	Точность работы
StandardScaler	0.417
MinMaxScaler	0.417
MaxAbsScaler	0.278

Тестирование

Результаты тестирования представлены в таблице 1.

Таблица 1 – Результаты тестирования

Тест	Комментарии

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(X_train)
print(y_train)
print(X_test)
print(y_test)
 	
print(clf)
print(res)
print(est)
 	
# scaling 	
scaled_standart = scale(X_train)
scaled_standart1 = scale(X_train, mode='standard')
print(scaled_standart == scaled_standart1)
print(scale(X_test, mode='my_mode'))
print(scale(X_test, mode='minmax'))
print(scale(X_test, mode='maxabs'))
	Корректно

Выводы

Были изучены основы анализа данных и библиотека sklearn, позволяющая непосредственно выполнять анализ данных на python.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Khangulyan_Sargis_lb3.py

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

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

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

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

def scale(data, mode = "standard"):
    if mode == 'standard':
        scaler = StandardScaler()
    elif mode == 'maxabs':
        scaler = MaxAbsScaler()
    elif mode == 'minmax':
        scaler = MinMaxScaler()
    else:
        return None
    return scaler.fit_transform(data)
```