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

Студентка гр. 3342 _____ Антипина В.А.

Преподаватель _____ Иванов И.И.

Санкт-Петербург

2024

Цель работы

Изучение методов анализа данных на языке Python, модуля scikit, реализация программы, анализирующей данные встроенной базы данных wine.

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 принимает на вход коэффициент выборки для тестирования (долю от общего объёма выборки). Загружается нужный датасет. Выбираются первые два столбца данных и загружаются в переменную X, в переменную y загружается столбец „target“. Вызывается функция train_test_split (которая была импортирована из sklearn.model_selection), результат записывается в переменные X_train, X_test, y_train, y_test. Эти же переменные возвращаются.

Функция train_model обучает модель классификации методом ближайших соседей. Функция создает экземпляр классификатора KNeighborsClassifier с помощью одноимённой функции от аргументов n_neighbors, weights и с помощью функции fit, применённой к тренировочной выборке, загружает в него данные. Возвращается обученная модель.

Функция predict прогнозирует классы для тестовых данных с помощью метода predict.

Функция estimate возвращает долю «успешности» построения модели, округленную до 3 знака после запятой. Это значение вычисляется с помощью функции accuracy_score и фактически является отношением предсказанных значений, совпавших с правильными данными, ко всем правильным данным.

Функция scale применяет один из трёх методов масштабирования к данным и возвращает обработанные данные.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования первой функции

№ п/п	Входные данные	Выходные данные	Комментарии
1.	train_size=0.1	0.379	Малый объём обучающих данных привёл к неточному результату
2.	train_size=0.3	0.547	
3.	train_size=0.5	0.843	
4.	train_size=0.7	0.892	
5.	train_size=0.9	0.971	Большой объём обучающих данных привёл к точному результату

Таблица 2 – Результаты тестирования второй функции

№ п/п	Входные данные	Выходные данные	Комментарии
1.	n_neighbors=3	0.915	
2.	n_neighbors=5	0.963	
3.	n_neighbors=9	0.971	При достижении максимума точность уменьшается
4.	n_neighbors=15	0.971	
5.	n_neighbors=25	0.915	

Таблица 3 – Результаты тестирования третьей функции

№ п/п	Входные данные	Выходные данные	Комментарии
1.	standart	0.971	Одинаковый результат
2.	minmax	0.971	

3.	maxabs	0.971	

Выводы

Результаты тестирования показывают, что с увеличением объёма обучающих данных увеличивается точность классификатора.

Также при изменении параметра n_neighbors точность растёт до достижения максимального значения, а затем убывает.

При использовании разных скейлеров точность не меняется, что доказывает, что исходные данные не требуют значительной предобработки.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
import numpy as np
import pandas as pd

def load_data(train_size=0.8):
    data = datasets.load_wine()
    X = data.data[:,[0,1]]
    y = data.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'):
    knn = KNeighborsClassifier(n_neighbors = n_neighbors, weights
= weights)
    knn.fit(X_train,y_train)
    return knn

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

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

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
    scaled_data = scaler.fit_transform(data)
    return scaled_data
```