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»

Студентка гр. 3342 _____ Епонишникова А.И.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2024

Цель работы

Познакомиться с базовыми принципами анализа данных. Изучить 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

load_data(train_size=0.8):

В переменную wine загружаем набор данных из datasets библиотеки sklearn. Далее разделяем выборку на обучающую (по умолчанию 0.8) и тестовую с помощью train_test_split с параметром random_state = 42.

train_model(X_train, y_train, n_neighbors = 15, weights = ‘uniform’):

функция train_model предназначена для обучения модели на основе переданных данных X_train и меток y_train. Внутри функции создается объект классификатора KNeighborsClassifier с заданными параметрами n_neighbors и weights (по умолчанию 15 и ‘uniform’ соответственно). Объект классификатора обучается на переданных данных X_train и метках y_train с использованием метода fit. Функция возвращает обученную модель.

predict(clf, X_test):

Функция использует метод predict модели clf, чтобы сделать прогнозы для набора тестовых данных X_test. Функция возвращает массив прогнозов, где каждый элемент представляет собой предсказанное значение для соответствующего элемента в X_test.

estimate(res, y_test):

Функция использует функцию accuracy_score для сравнения предсказанных значений res с фактическими значениями y_test. Функция возвращает оценку точности модели, которая представляет собой долю правильных предсказаний модели на тестовом наборе данных.

scale(data, mode = 'standard'):

В зависимости от выбранного режима масштабирования (mode), функция создает соответствующий объект масштабирования: StandardScaler,

MinMaxScaler или MaxAbsScaler. Затем функция использует этот объект для выполнения масштабирования данных с помощью метода fit_transform. Функция возвращает масштабированные данные в соответствии с выбранным режимом.

Разработанный программный код см. в приложении А.

Тестирование

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

№	train_size	accuracy
1.	0.1	0.379
2.	0.3	0.8
3.	0.5	0.843
4.	0.7	0.815
5.	0.9	0.722

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

№	n_neighbors	accuracy
1.	3	0.861
2.	5	0.833
3.	9	0.861
4.	15	0.861
5.	25	0.833

Таблица 3 - Исследование работы классификатора с предобработанными данными

№	sacler	accuracy
1.	StandardScaler	0.889
2.	MinMaxScaler	0.806
3.	MaxAbsScaler	0.75

Выводы

Исходя из таблицы 1 видно, что оптимальная точность достигается при train_size от 0.5 - 0.7. Если же train_size = 0.1, то модели не хватает данных на обучение, поэтому точность низкая. Если train_size = 0.9, то размер тестовой выборке слишком маленький, что приводит к неустойчивости оценки качества, то есть небольшие изменения в тестовой выборке могут привести к значительным изменениям в оценке качества модели.

Из таблицы 2 видно, что модель демонстрирует стабильность при больших изменениях в алгоритме "K ближайших соседей". Это свидетельствует о том, что исходные данные хорошо соответствуют модели, и их легко разделить на классы даже визуально.

Из таблицы 3 можно сделать вывод, что применение StandardScaler приводит к улучшению точности прогнозов модели, что делает его использование в данном контексте обоснованным. В то же время MinMaxScaler и MaxAbsScaler либо слабо влияют на точность, либо незначительно снижают её. Экспериментальным путем выяснено, что в условиях данной задачи следует использовать StandardScaler.

Были изучены основные принципы анализа данных в Python и разработана программа, использующая их. Реализованы методы для разделения данных для обучения и тестирования, создания экземпляра классификатора соседей, предсказания данных и оценки качества результатов классификации.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.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


def load_data(train_size = 0.8):
    wine = datasets.load_wine()

    y = wine.target
    x = wine.data[:,:2]

        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'):
        return KNeighborsClassifier(n_neighbors = n_neighbors,
weights=weights).fit(X_train, y_train)

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

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

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

    return transformer.fit_transform(data)
```