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. 3343 _____ Гельман П.Е.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Изучить и освоить использование библиотеки scikit-learn на Python для выполнения анализа данных, включая классификацию данных для выделения различных групп и последующую обработку полученных результатов.

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

Функции, реализованные в программе:

1. load_data(train_size=0.8):

- Эта функция загружает набор данных о вине и разделяет его на обучающие и тестовые данные с помощью train_test_split.

- Параметр train_size определяет размер обучающего набора данных (по умолчанию 80%).

- Функция возвращает 4 массива: X_train (обучающие данные), X_test (тестовые данные), y_train (метки обучающих данных) и y_test (метки тестовых данных).

2. train_model(X_train, y_train, n_neighbors=15, weights='uniform'):

- Эта функция обучает модель классификатора K ближайших соседей на переданных обучающих данных X_train с соответствующими метками `y_train`.

- Параметры по умолчанию для n_neighbors - 15 и weights - 'uniform'.

- Функция возвращает обученную модель.

3. predict(model, X_test):

- Эта функция принимает обученную модель model и тестовые данные X_test, а затем возвращает предсказанные метки для тестовых данных с использованием этой модели.

4. estimate(res, y_test):

- Эта функция оценивает точность предсказанных меток res по сравнению с фактическими метками y_test с помощью метрики точности.

- Она использует внутреннюю функцию accuracy_score из sklearn.metrics для вычисления точности.

- Функция возвращает округленное значение точности до трех знаков после запятой.

5. scale(data, mode='standard'):

- Эта функция принимает данные data и тип метода шкалирования mode (по умолчанию - 'standard') для преобразования данных с помощью соответствующего метода шкалирования.

- В зависимости от значения mode функция создает объект соответствующего шкалировщика (StandardScaler, MinMaxScaler, MaxAbsScaler).

- После чего функция преобразует и возвращает отмасштабированные данные с помощью метода fit_transform шкалировщика.

Исследование работы классификатора, обученного на данных разного размера:

train_size	Точность классификатора
0.1	0.379
0.3	0.8
0.5	0.798
0.7	0.796
0.9	0.722

Малый размер выборки (0.1): При малом размере обучающей выборки модель может столкнуться с недостаточным количеством данных для обучения. Это может привести к недообучению, что отражается в низкой точности.

Средний размер выборки (0.3-0.7): При увеличении размера обучающей выборки точность классификатора существенно повышается. Это связано с тем, что модель получает больше информации и имеет возможность лучше обобщать зависимости в данных.

Большой размер выборки (0.9): Опять же, при очень большой обучающей выборке может возникнуть проблема переобучения модели. Модель может излишне подстраиваться под обучающий набор данных и терять способность обобщения на новых данных, что может привести к снижению точности на тестовой выборке.

Исследование работы классификатора, обученного с различными значениями n_neighbors:

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

n_neighbors = 3 и n_neighbors = 9 и n_neighbors = 15: В этом случае точность модели одинакова (0.861), что может означать, что какое-то конкретное значение n_neighbors (в данном случае 3, 9, 15) работает наилучшим образом для данного набора данных. Возможно, что эти значения обеспечивают должный баланс между смещением и дисперсией модели, что приводит к хорошей обобщающей способности.

При значениях n_neighbors равных 5 и 25 наблюдается немного более низкая точность (0.833). Это может быть связано с тем, что слишком маленькое или слишком большое количество соседей может привести к недообучению (слишком простая модель) или переобучению (слишком сложная модель) соответственно.

Исследование работы классификатора с предобработанными данными:

Тип скейлера	Точность классификатора
StandardScaler	0.889
MinMaxScaler	0.806
MaxAbsScaler	0.75

StandardScaler: Точность классификации при использовании StandardScaler составляет 0.889. Это может означать, что стандартизация

данных помогает модели лучше улавливать закономерности и делает их более интерпретируемыми, что приводит к повышению точности.

MinMaxScaler: Точность классификации при использовании MinMaxScaler равна 0.806. В данном случае можно предположить, что масштабирование в заданный диапазон ограничивает вариативность данных, что может привести к частичной потере информации и снижению точности модели.

MaxAbsScaler: Точность классификации при использовании MaxAbsScaler равна 0.75. В данном случае такая низкая точность может указывать на то, что масштабирование по максимальному абсолютному значению не эффективно или даже искажает данные, делая их менее информативными для модели.

Выводы

В ходе лабораторной работы была изучена библиотека scikit-learn, обучение и классификация моделей с помощью неё. Проведен анализ данных с использованием метода k-ближайших соседей, что позволило классифицировать информацию.

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.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[:, :2], 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'):
    model = KNeighborsClassifier(n_neighbors=n_neighbors, weights=weights)
    model.fit(X_train, y_train)

    return model

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

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

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