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. Вариант 1

Студент гр. 3343 _____ Калиберов Н.И

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2024

Цель работы

Научиться работать с библиотекой scikit-learn, понять, для чего она используется, научится анализу полученных вашим кодом результатов и формирование соответствующих разделов.

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, y_train, X_test, 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

Чтобы получить данные о винах, необходимо импортировать датасеты и вызвать функцию data_wine. Чтобы взять данные для осей X и Y используются data и target соответственно. После чего делим данные на тренировочные и тестовые с помощью функции train_test_split в соответствии с заданным train_size (в процентном соотношении, по умолчанию 0.8, т.е 80%). Чтобы взять первые 2 столбца что в тестовых, что в обучающих данных необходимо сделать двойной срез вида [: , :2].

Чтобы исследовать работу классификатора необходимо подготовить вот такой код:

```
if __name__ == '__main__':
    X_train, X_test, y_train, y_test = load_data()
    clf = train_model(X_train, y_train)
    res = predict(classifier, X_test)
    est = estimate(res, y_test)
    print(est)
```

После получения тестовых данных необходимо создать модель классификации K-ближайших соседей, и с помощью функции predict выполнить классификацию тестовых данных и получить какой-то ответ. А с помощью функции estimate сделать оценку предсказанных данных (сравнить данные на соответствие с y_test).

Значение train_size	Точность работы классификатора
0.1	0,379
0.3	0,8
0.5	0,843
0.7	0,815
0.8 (default)	0,861
0.9	0,722

Можно заметить, что при дефолтном значении train_size = 0.8 достигается самая высокая точность. В принципе, можно заметить, что значение держится в среднем в районе 0.8 при train_size >= 0.3 и train_size <= 0.8. При значении 0.1 точность работы очень низкая, модели не хватает данных, чтобы обучиться и делать правильную оценку. При повышении значения train_size выше 0.8

точность работы также начинает падать, т.к. модель начинает уже не предсказывать данные, а запоминать их (происходит переобучение). Она начинает терять способность обобщать на новые, ранее не виданные данные.

Рассмотрим, как значение n_neighbours влияет на точность алгоритма ближайших соседей. Классификация вычисляется простым большинством голосов ближайших соседей каждой точки: точке запроса назначается класс данных, который имеет наибольшее количество представителей среди ближайших соседей точки. Количество этих соседей, с которыми нужно сравнить точку надо задать самостоятельно. Изменение данного значения может приводить к тому, что точка может изменить свою группу принадлежности.

Значение n_neighbors при train_size=0.8	Точность работы классификатора
3	0,861
5	0,833
9	0,861
15 (default)	0,861
25	0,833

В данном примере видно, что значение n_neighbors не сильно влияет на точность работы. Видно, что точность незначительно уменьшается при значениях 5 и 25. При сильно больших значениях, точность будет уменьшаться уже к 0.7. Происходит это потому, что точки начинают принадлежать не к той группе, из-за того, что кол-во исследуемых соседей становиться больше, некоторые группы данных в окрестности исследуемой точки становятся больше и их влияние на исследуемую точку возрастает.

Для проверки точности работы скейлеров необходимо выполнить предобработку данных:

```
if __name__ == '__main__':
    X_train, X_test, y_train, y_test = load_data()
    X_train = scale(X_train)
    X_test = scale(X_test)
    clf = train_model(X_train, y_train)
```

```
res = predict(clf, X_test)
est = estimate(res, y_test)
print(est)
```

Стандартизация данных сводиться к тому, чтобы преобразовать данные к единому формату и представлению, который будет удобен для определённого вида обработки. Делать стандартизацию можно разными методами. Если говорить про стандартную стандартизацию, то она изменяет масштаб данных, так что среднее значение становится равным 0, а стандартное отклонение становится равным 1. Из каждого значения вычитается среднее из набора, а затем разделено на стандартное отклонение всего набора данных.

Скейлер MinMaxScaler преобразует значения по такой формуле:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Данные значения масштабируются так, чтобы значения находилась в диапазоне от 0 до 1.

Скейлер MaxAbsScaler масштабирует значения в диапазон от -1 до 1 с помощью деления каждого значения на максимальное значение по модулю.

Типы скейлеров	Точность работы классификатора
StandardScaler	0,889
MinMaxScaler	0,806
MaxAbsScaler	0,75

В данном случае видно, что скейлер по умолчанию отработал лучше всего и показал наивысшую точность, MaxAbsScaler показал худшие результаты среди представленных скейлеров.

Такое могло произойти, потому что MinMaxScaler и MaxAbsScaler чувствительны к выбросам, в то время как StandardScaler игнорирует выбросы.

Выводы

Была написана программа, которая состоит из функции загрузки данных о винах, которая разделяет их на обучающие и тестовые данные, функции обучении модели, которая загружает в классификатор ближайших соседей обучающие данные, функция, которая применяет эту модель на тестовых данных, функция оценки результатов и их предобработка.

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.preprocessing import StandardScaler, MinMaxScaler,
MaxAbsScaler

def load_data(train_size=0.8):
    wine = datasets.load_wine()
    X = wine.data[:, :2]
    y = wine.target
    X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42, train_size=train_size)
    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(classifier, X_test):
    return classifier.predict(X_test)

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

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

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Тест №1

Ввод (программа):

```
if __name__ == '__main__':
    X_train, X_test, y_train, y_test = load_data()
    print(X_train, X_test, sep="\n")
    X_train = scale(X_train)
    X_test = scale(X_test)
    print(X_train, X_test, sep="\n")
```

Вывод:

```
[[14.34  1.68]
 [12.53  5.51]]
[[13.64  3.1 ]
 [14.21  4.04]]
[[ 1.66529275 -0.60840587]
 [-0.54952506  2.7515415 ]]
[[0.71755938 0.91772503]
 [1.4552507  1.86541545]]
```

Тест №2

Ввод (программа):

```
if __name__ == '__main__':
    X_train, X_test, y_train, y_test = load_data()
    X_train = scale(X_train)
    X_test = scale(X_test)
    classifier = train_model(X_train, y_train)
    res = predict(classifier, X_test)
    est = estimate(res, y_test)
    print(est)
```

Вывод:

```
0.889
```