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. 3341 _____ Самокрутов А.Р.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Цель работы заключается в изучении основ анализа данных с использованием библиотеки *sklearn* языка программирования *Python*, создание классификатора, его обучение и применение для классификации данных. Также необходимо реализовать программу, которая анализирует существующий ассортимент.

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

Описание функций:

1. *load_data(train_size=0.8):* загружает набор данных о вине из библиотеки sklearn в переменную wine и разбивает их на обучающую и тестовую части с использованием функции *sklearn.model_selection.train_test_split()* с установленным параметром рандомизации.

2. *train_model(train_x, train_y, n_neighbors=15, wights='uniform'):* принимает на вход обучающую выборку (train_x и train_y) и аргументы n_neighbors и weights. Длаее она создает объект класса KNeighborsClassifier и загружает в него данные n_neighbors и weights, после чего обучает его на данных train_x и train_y, полученных в функции выше. Функция возвращает экземпляр классификатора.

3. *predict(classifier, test_x):* принимает на вход обученный классификатор и тренировочный набор данных (test_x), после чего производит предсказание классов для тестовых данных test_x с помощью метода predict(). В качестве результата возвращает предсказанные данные.

4. *estimate(res, test_y):* принимает результаты классификации и истинные метки тестовых данных (test_y), считает процент предсказанных результатов, совпавших с «правильными» в test_y с помощью функции accuracy_score(). В качестве результата возвращает полученное отношение, округленное до 0,001.

5. *scale(data, mode='standard'):* функция принимает аргумент, содержащий данные, и аргумент mode - тип скейлера (допустимые значения: 'standard', 'minmax', 'maxabs', для других значений возвращается None, значение по умолчанию - 'standard'), обрабатывает данные соответствующим скейлером методом fit_transform(). В качестве результата возвращаются полученные после обработки данные.

Исследование работы классификатора, обученного на данных разного размера:

Таблица 1 – Точность работы классификаторов в зависимости от аргумента train_size

train_size	Точность работы
0.1	0.379
0.3	0.8
0.5	0.843
0.7	0.815
0.9	0.722

Наибольшая точность достигается при значении train_size = 0.5. Это явление можно обосновать так: при меньших значениях модель не получает достаточно данных для выявления правдоподобных зависимостей и закономерностей, а при значениях, больших 0.5, модель избыточно подстраивается под входные данные и теряет способность к их обощению на новую информацию.

Исследование работы классификатора, обученного с различными значениями n_neighbors:

Таблица 2 – точность работы классификаторов в зависимости от аргумента n_neighbors

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

Наибольшая точность достигается при значениях n_neighbors = 9 и 15. Слишком маленькое значение n_neighbors приводит к недообучению модели, а при слишком большом значении модель теряет способность к выявлению закономерностей, т.к. обучена на этих данных избыточно.

Исследование работы классификатора с предобработанными данными:

Таблица 3 – точность работы классификаторов в зависимости от скейлеров

Скейлер	Точность работы
Standard	0.889
MinMax	0.806
MaxAbs	0.75

Заметим, что наибольшая точность достигается при использовании StandardScaler. Можно предположить, что StandardScaler масштабирует данные таким образом, что их среднее значение равно 0, а стандартное отклонение — 1, что может улучшить работу некоторых моделей машинного обучения.

Разработанный код см. в приложении А.

Выводы

Были изучены основы анализа данных на языке программирования *Python* с применением библиотеки *sklearn*.

В результате работы была написана программа на языке программирования *Python*, которая проводит анализ существующего ассортимента, проверив возможности инструмента классификации данных для выделения различных классов.

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.metrics import accuracy_score
    from  sklearn.preprocessing  import  StandardScaler,  MinMaxScaler,
MaxAbsScaler


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

        x = wine.data[:, 0:2]
        y = wine.target

        train_x, test_x, train_y, test_y = train_test_split(x, y,

train_size=train_size,
                                                            test_size=1
- train_size,

random_state=42)

        return train_x, test_x, train_y, test_y


    def       train_model(train_x,       train_y,       n_neighbors=15,
weights='uniform'):
        classifier    =    KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights)
        classifier.fit(train_x, train_y)

        return classifier


    def predict(classifier, test_x):
        prediction_y = classifier.predict(test_x)

        return prediction_y


    def estimate(res, test_y):
        accuracy = accuracy_score(test_y, 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
```