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 _____ Кузнецова С.Е.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Изучение основ анализа данных с применением языка Python и использованием библиотеки sklearn, создание классификатора, его обучение и применение для классификации данных. Написание программы на языке программирования Python, которая проводит анализ существующего ассортимента, проверив возможности инструмента классификации данных для выделения различных классов.

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

Для анализа данных о классах вин была использована библиотека sklearn и встроенный в него набор данных о вине.

Реализованы функции:

1. *load_data():* принимает на вход аргумент train_size (размер обучающей выборки, по умолчанию равен 0.8), загружает набор данных о вине из библиотеки sklearn в переменную wine. Функция разбивает данные для обучения и тестирования в соответствии со значением train_size с использованием функции train_test_split модуля sklearn.model_selection. Функция возвращает значения X_train, X_test, y_train, y_test.

2. *train_model():* принимает обучающую выборку (два аргумента - X_train и y_train) и аргументы n_neighbors и weights (значения по умолчанию 15 и 'uniform' соответственно), создает экземпляр классификатора KNeighborsClassifier и загружает в него данные X_train, y_train с параметрами n_neighbors и weights. Функция возвращает экземпляр классификатора.

3. *predict():* принимает обученную модель классификатора и тренировочный набор данных (X_test), выполняет классификацию данных из X_test. В качестве результата возвращает предсказанные данные.

4. *estimate():* функция принимает результаты классификации и истинные метки тестовых данных (y_test), считает отношение предсказанных результатов, совпавших с «правильными» в y_test к общему количеству результатов. В качестве результата возвращает полученное отношение, округленное до 0,001.

5. *scale():* 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. В качестве результата возвращаются полученные после обработки данные.

Исследование работы классификатора, обученного на данных разного размера.

Точность работы классификаторов, обученных на данных от функции load_data с разным значением аргумента train_size представлена в таблице 1:

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

Заметим, что наибольшая точность достигается при значении train_size = 0.5. Это обуславливается тем, что при меньших значениях модели не хватает данных для обучения, а при больших может происходить переобучение модели, которая будет избыточно подстраиваться под тренировочные данные.

Исследование работы классификатора, обученного с различными значениями n_neighbors

Точность работы классификаторов, обученных с разным значением аргумента n_neighbors представлена в таблице 2:

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

Заметим, что наибольшая точность достигается при значениях n_neighbors = 5 и 9. Слишком маленькое значение n_neighbors может привести к недостаточному обучению модели, тогда как слишком большое значение

аргумента может привести к переобучению модели и потере способности к выявлению закономерностей в данных.

Исследование работы классификатора с предобработанными данными.

Точность работы классификаторов, обученных на данных предобработанных с помощью скейлеров StandardScaler, MinMaxScaler, MaxAbsScaler представлена в таблице 3:

Таблица 3 – точность работы классификаторов в зависимости от скейлеров

Scaler	Точность работы
StandardScaler	0.889
MinMaxScaler	0.806
MaxAbsScaler	0.75

Наибольшая точность достигается при использовании скейлера StandardScaler. Это может быть связано с тем, 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 *Python* с применением библиотеки *sklearn*. 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.preprocessing import StandardScaler, MinMaxScaler,
MaxAbsScaler
    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

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

        return neighbors

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

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

        return accur

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