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. 3341 Чинаева М.Р.

Преподаватель Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Изучение основ анализа данных с использованием библиотеки sklearn, создание классификатора, его обучение и применение для классификации данных. Написание программы, анализирующей и классифицирующей данные, в зависимости от заданных параметров.

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:

1. *load_data(train_size=0.8)*

Принимает на вход размер обучающей выборки, по умолчанию равный 0,8. Загружает набор данных о вине из библиотеки sklearn в переменную wine с помощью функции load_wine(). Функция разбивает данные для обучения и тестирования с использованием функции train_test_split().

Функция возвращает значения X_train и X_test – двумерные массивы, y_train, y_test – одномерные массивы.

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

Принимает на вход обучающую выборку и аргументы n_neighbors и weights, имеющие значения по умолчанию 15 и 'uniform'. Инициализирует экземпляр классификатора KNeighborsClassifier с параметрами n_neighbors и weights. После классификатор обучается на загруженных в него данных.

Функция возвращает экземпляр классификатора.

3. *predict(clf, X_test)*

Принимает на вход обученную модель классификатора и тренировочный набор данных. Выполняет классификацию данных из X_test.

В качестве результата возвращает предсказанные данные.

4. *estimate(predicitions, y_test)*

Принимает на вход результаты классификации и истинные метки тестовых данных. Считает отношение предсказанных результатов, совпавших с «правильными» в y_test к общему количеству результатов.

Возвращает полученное отношение, округленное до 0,001.

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

Функция принимает на вход данные и тип скейлера с допустимыми значениями: 'standard', 'minmax', 'maxabs'. Если же переданное значение не является ни одним из этих возвращается 'None'. Далее обрабатывает данные соответствующим скейлером.

Возвращает данные, полученные после обработки.

1. Исследование работы классификатора, обученного на данных разного размера.

Таблица 1 – Точность работы классификаторов, обученных на данных от функции load_data, для разных train_size

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

Точность классификатора увеличивается с увеличением объема выборки, однако после train_size = 0,5 начинает происходить переобучение модели, так как она избыточно подстраивается под данные.

2. Исследование работы классификатора, обученного с различными значениями n_neighbors

Таблица 2 – точность работы классификаторов для разных 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, равных 3, 9 и 15. Слишком большое значение n_neighbors скорее всего приведет к переобучению модели, так как часть данных становится менее полезной из-за своей избыточности. Однако при слишком маленьких у модели не происходит достаточного обучения.

3. Исследование работы классификатора с предобработанными данными.

Таблица 3 – точность работы классификаторов для различных скейлеров

Скейлер	Точность работы
StandardScaler	0.889
MinMaxScaler	0.806

MaxAbsScaler	0.75

Наибольшая точность достигается при использовании скейлера StandardScaler. Это связанно с тем, что он подходит для большего количества алгоритмов машинного обучения, он устраняет смещения в данных, масштабируя их.

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 анализирует и классифицирует данные в зависимости от заданных параметров.

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 = wine.data[:, :2]
        y = wine.target
        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'):
        model = KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights)
        return model.fit(X_train, y_train)

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


    def estimate(predicitions, y_test):
        accuracy = accuracy_score(y_test, predicitions)
        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
        return scaler.fit_transform(data)
```