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

Студент гр. 3342 _____ Иванов Д. М.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург
2024

Цель работы

Изучить основы анализа данных и машинного обучения и их реализацию на языке Python. С их помощью написать программу, которая проводит анализ существующего ассортимента вина и обучает модель для классификации новых данных.

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 выполнения поставленной задачи необходимо было подключить необходимую библиотеку для машинного обучения (sklearn) и реализовать несколько функций, каждая из которых выполняет определенный алгоритм.

1) def load_data(train_size=0.8): Происходит загрузка датасета о вине из библиотеки для обучения модели. Разбивка на входные и выходные параметры для обучения (X, y). Через параметр train_size данные будут разбиты на выборки для обучения и тестирования модели (*_train, *_test).

2) def train_model(X_train, y_train, n_neighbors=15, weights='uniform'): Происходит создания и обучение модели на основе X_train, y_train и через алгоритм KneighborsClassifier (метод ближайших соседей). Этот модель будет работать следующим образом: для новой точки она будет находить количество соседних точек n_neighbors в данном пространстве и по ним относить ее к определенному классу. Возвращается обученная модель.

3) def predict(clf, X_test): Предсказание нашей моделью значений X_test по алгоритму, описанному выше.

4) def estimate(res, y_test): Оценка качества полученных результатов классификации через метод accuracy_score. Происходит сравнение настоящих значений и предсказуемых?.

5) def scale(data, mode='standard'): Происходит предварительная обработки данных, которая преобразует признаки в заданный масштаб для обеспечения более стабильного обучения модели по одному из следующих алгоритмов.

StandardScaler центрирует данные путем удаления среднего значения и масштабирует их путем деления на стандартное отклонение, что приводит к нулевому среднему значению и стандартному отклонению равному единице.

MinMaxScaler масштабирует данные путем приведения значений признаков к заданному диапазону, обычно от 0 до 1, путем вычитания минимального значения и деления на разницу между максимальным и минимальным значениями.

MaxAbsScaler масштабирует данные путем деления на максимальное абсолютное значение в каждом признаке, результатом являются значения в диапазоне [-1, 1].

Проведем исследования для нашей модели.

Таблица 1 – Исследование работы классификатора, обученного на данных разного размера

Размер обучающей выборки	Точность модели
0.1	0.379
0.3	0.8
0.5	0.843
0.7	0.815
0.9	0.722

Таблица 2 – Исследование работы классификатора, обученного с различными значениями n_neighbors

Значение n_neighbors	Точность модели
3	0.861
5	0.833
9	0.861
15	0.861
25	0.833

Таблица 3 – Исследование работы классификатора с предобработанными данными

Тип скейлера	Точность модели
StandardScaler	0.417
MinMaxScaler	0.417
MaxAbsScaler	0.278

Выводы

В данной работе была разработана программа, которая обучает модель для предсказания классов вин. Также были проведены с ней некоторые исследования.

1-ое исследование: На его основании можно придти к выводу, что наивысшую точность модель показывает при обучающей выборки 0.5. При меньших значениях из-за недообучения и при слишком больших из-за переобучения (когда модель слишком сильно подстраивается под обучающие данные) точность падает.

2-ое исследование: При различных значениях количества соседей точность особо не меняется. Можно сделать вывод, что этот параметр в приницпе не играет большой роли для обучения модели.

3-е исследование: StandardScaler и MinMaxScaler модель показала одинаковую точность, а при MaxAbsScaler точность хуже. Можно сделать вывод, что значения признаков имели очень больший диапазон, который при использовании MaxAbsScaler не был эффективно масштабирован, что привело к ухудшению производительности модели.

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 import preprocessing

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,
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'):
    classifier = KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights)

    classifier.fit(X_train, y_train)

    return classifier

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

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

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