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 _____ Львов А. В.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург
2024

Цель работы

Изучение основ анализа данных и машинного обучения, освоение работы с основными инструментами обработки данных и тренировки модели с последующим их применением.

Задание

Вариант 1

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, реализованные в программе:

1) load_data принимает один необязательный аргумент train_size, отвечающий за размер обучающей выборки с значением по умолчанию 0.8. Она загружает датасет о вине из библиотеки sklearn и возвращает обучающие и тестовые выборки.
2) train_model принимает тестовые выборки и опциональные данные о количестве соседей и о весе каждой точки с значениями по умолчанию 15 и «uniform» соответственно. Возвращает экземпляр классификатора.
3) predict принимает экземпляр класса KNeighborsClassifier и тренировочный набор данных. Эта функция выполняет классификацию данных из переданной выборки и возвращает предсказанные данные.
4) estimate выполняет классификацию данных из переданной выборки и возвращает долю правильных предсказаний.
5) scale принимает аргумент, содержащий данные и тип скейлера, которым обрабатывает их и возвращает результат.

Исходный код программы см. в приложении А.

Тестирование

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

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

Полученные результаты связаны с тем, что при недостаточном размере обучающей выборки классификатору будет не хватать данных для более точной классификации. А при преобладании обучающей выборки над тестовой модель может переобучаться на обучающих данных, что приводит к снижению точности на тестовых данных.

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

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

В общем случае, недостаточное или слишком большое количество соседей приводит к переобучению и недообучению модели соответственно, однако в данном случае можно сделать вывод, что количество соседей существенно не влияет на точность.

Таблица 3 – Исследование работы классификатора с предобработанными данными

Скейлер	Точность
StandardScaler	0.417
MinMaxScaler	0.417
MaxAbsScaler	0.278

Скейлеры позволяют нормализовать данные для улучшения качества моделей и их обобщающей способности. В общем, их выбор зависит от конкретной задачи. На данном примере можно сделать вывод о том, что для предобработки данных о вине лучше всего подходят StandardScaler и MinMaxScaler.

Выводы

Было проведено ознакомление с основными инструментами обработки и анализа данных. Было проведено тестирование с различными наборами данных для изучения основных концепций и закономерностей машинного обучения.

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_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'):
    classifier     =     KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights)
    classifier.fit(X_train, y_train)
    return classifier


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


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


def scale(data, mode='standard'):
    mods  =  {'standard':  StandardScaler(),  'minmax':  MinMaxScaler(),
'maxabs': MaxAbsScaler()}
    if mode in mods:
        scaler = mods[mode]
    else:
        return
    scaled_data = scaler.fit_transform(data)
    return scaled_data
```