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 _____ Пушко К. Д.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург
2024

Цель работы

Познакомиться с базовыми принципами анализа данных. Изучить основные инструменты для обработки и анализа данных.

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

Функция load_data(train_size = 0.8) принимает на вход аргумент train_size – размер обучающей выборки, по умолчанию равный 0.8. Функция загружает набор данных о вине из библиотеки sklearn в переменную wine и разбивает на выборки для тестирования и обучения. Возвращает X_train, X_test, y_train, y_test.

Функция train_model(X_train, y_train, n_neighbors=15, weights='uniform') Обучающую выборку, количество «соседей» и веса. Функция создает экземпляр классификатора KNeighborsClassifier и загружает в него данные X_train, y_train с параметрами n_neighbors и weights, и возвращает экземпляр классификатора.

Функция predict(clf, X_test) принимает обученную модель классификатора и тренировочный набор данных (X_test), которая выполняет классификацию данных из X_test. Возвращает предсказанные данные.

Функция estimate(res, y_test) принимает 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»). Возвращает полученное отношение.

Функция scale(data, mode='standard') принимает 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. Возвращает обработанные данные.

Разработанный программный код см. в приложении А.

Тестирование

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

№	train_size	accuracy
1.	0.1	0.897
2.	0.3	0.946
3.	0.5	0.963
4.	0.7	0.973
5.	0.9	0.981

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

№	n_neighbors	accuracy
1.	3	0.963
2.	5	0.973
3.	9	0.976
4.	15	0.976
5.	25	0.968

Таблица 3 - Исследование работы классификатора с предобработанными данными

№	sacler	accuracy
1.	StandardScaler	0.976
2.	MinMaxScaler	0.976
3.	MaxAbsScaler	0.976

Выводы

Полученные результаты из таблицы 1 показывают, что точность работы классификатора увеличивается с увеличением размера обучающей выборки. Это объясняется тем, что при большем количестве обучающих данных классификатор лучше обучается на различных вариантах данных и может более точно предсказывать метки для новых данных.

Полученные результаты из таблицы 2 показывают, что точность работы классификатора зависит от выбранного значения параметра n_neighbors. Наилучшие результаты получены при значении n_neighbors равном 9 и 15, что соответствует оптимальному количеству соседей для данного набора данных.

Полученные результаты из таблицы 3 показывают, что предобработка данных с помощью различных скейлеров не оказывает значительного влияния на точность работы классификатора. Все три скейлера дают одинаковый результат, что говорит о том, что данные уже достаточно нормализованы и не требуют дополнительной предобработки.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_wine
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[:, [0, 1]]
    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)
    model.fit(X_train, y_train)
    return model

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

def estimate(res, y_test):
    accuracy = sum(res == y_test) / len(y_test)
    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
```