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. 3343 _____ Иванов П.Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Изучить библиотеку scikit-learn и применить классификатор K-ближайших соседей (KNN) для анализа данных. Исследовать влияние различных параметров модели и методов предобработки данных на точность классификации.

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

Были реализованы 5 функций. Описание каждой функции и логика работы описаны ниже.

1. *load_data()* - используется для загрузки данных.

- Загружает данные о вине из библиотеки sklearn.
- Извлекает только первые два столбца из данных и метки классов.
- Делит данные на обучающую и тестовую выборки в соответствии с параметром train_size.
- Возвращает x_train, x_test, y_train, y_test.

2. *train_model()* - используется для обучения модели.

- Создаёт экземпляр классификатора K-ближайших соседей с заданными параметрами n_neighbors и weights.
- Обучает модель на данных x_train и y_train.
- Возвращает обученную модель.

3. *predict()* - используется для предсказания следующего значения.

- Выполняет предсказание меток классов для тестовых данных x_test с использованием обученной модели model.
- Возвращает предсказанные значения y_pred.

4. *estimate()* - используется для оценки качества работы модели.

- Вычисляет точность предсказаний как долю правильных предсказаний среди всех тестовых данных.
- Возвращает точность, округленную до трёх знаков после запятой.

5. *scale()* - используется для обработки входных данных по скейлеру.

- Масштабирует данные с использованием заданного скейлера (standard, minmax, maxabs).

- Возвращает масштабированные данные.
- Если передан некорректный режим, возвращает None.

Для исследования точности классификатора при различных размерах обучающей выборки использовались значения train_size из списка: 0.1, 0.3, 0.5, 0.7, 0.9. Результаты приведены в таблице:

train_size	accuracy
0.1	0.611
0.3	0.593
0.5	0.685
0.7	0.741
0.9	0.778

Из таблицы видно, что с увеличением размера обучающей выборки train_size точность модели K-ближайших соседей возрастает.
Это связано с тем, что больший объём обучающих данных позволяет модели лучше обучаться и делать более точные предсказания на тестовых данных.
При малом размере обучающей выборки модель не получает достаточно информации для точного классифицирования новых данных.

Для исследования точности классификатора при различных значениях n_neighbors использовались значения из списка: 3, 5, 9, 15, 25. Результаты приведены в таблице:

n_neighbors	accuracy
3	0.741
5	0.759
9	0.759
15	0.759

25	0.741

Отсюда понятно, что точность классификатора незначительно меняется при различных значениях n_neighbors. Оптимальное значение n_neighbors для данного набора данных находится в диапазоне от 5 до 15. При слишком малом значении n_neighbors модель становится более чувствительной к шуму в данных, а при слишком большом значении — модель может стать слишком сглаженной и терять точность.

Для исследования точности классификатора при использовании различных скейлеров использовались скейлеры из списка: StandardScaler, MinMaxScaler, MaxAbsScaler. Результаты приведены в таблице:

scaler	accuracy
StandardScaler	0.832
MinMaxScaler	0.801
MaxAbsScaler	0.779

Масштабирование данных улучшает качество классификации. Масштабирование данных помогает нормализовать их, что улучшает работу алгоритмов, таких как K-ближайших соседей.

Разработанный программный код см. в приложении А.

Выводы

В результате выполнения работы был написан код для анализа данных о вине с использованием классификатора K-ближайших соседей (KNN) и библиотеки scikit-learn. Были реализованы функции для загрузки и предобработки данных, обучения и оценки модели. Также было проведено исследование влияния размера обучающей выборки, значения гиперпараметра n_neighbors и различных методов предобработки данных на точность классификации.

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


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


def estimate(y_pred, y_test):
    accuracy = (y_pred == y_test).mean()
    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
```