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. 3383 _____ Логинова А.Ю.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Целью данной работы является изучение процесса загрузки и разделения данных на обучающую и тестовую выборки, исследование процесса обучения модели KNN, оценка точности предсказаний модели на тестовой выборке и масштабирование данных. Для решения поставленных задач в данной лабораторной работе необходимо изучить модуль sklearn.

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.

В отчёте приведите (чек-лист преподавателя):

- описание реализации 5и требуемых функций
- исследование работы классификатора, обученного на данных разного размера

- приведите точность работы классификаторов, обученных на данных от функции load_data со значением аргумента train_size из списка: 0.1, 0.3, 0.5, 0.7, 0.9
- оформите результаты пункта выше в виде таблицы
- объясните полученные результаты

- исследование работы классификатора, обученного с различными значениями n_neighbors
  - приведите точность работы классификаторов, обученных со значением аргумента n_neighbors из списка: 3, 5, 9, 15, 25
  - в качестве обучающих/тестовых данных для всех классификаторов возьмите результат load_data с аргументами по умолчанию (учтите, что для достоверности результатов обучение и тестирование классификаторов должно проводиться на одних и тех же наборах)
  - оформите результаты в виде таблицы
  - объясните полученные результаты
- исследование работы классификатора с предобработанными данными
  - приведите точность работы классификаторов, обученных на данных предобработанных с помощью скейлеров из списка: StandardScaler, MinMaxScaler, MaxAbsScaler
  - в качестве обучающих/тестовых данных для всех классификаторов возьмите результат load_data с аргументами по умолчанию - учтите, что для достоверности сравнения результатов классификации обучение должно проводиться на одних и тех же данных, поэтому предобработку следует производить после разделения на обучающую/тестовую выборку.
  - оформите результаты в виде таблицы
  - объясните полученные результаты

Выполнение работы

Для написания функций был использован модуль scikit-learn.

1. Загрузка и разделение данных:

Данные загружаются из библиотеки sklearn, используя функцию datasets.load_wine().

Данные разделяются на обучающую и тестовую выборки с помощью функции train_test_split(), при этом используется только два первых признака для упрощения задачи классификации.

Параметр train_size определяет долю данных, используемых для обучения, и по умолчанию составляет 80%.

2. Масштабирование данных:

Масштабирование данных выполняется с использованием трех различных методов: StandardScaler, MinMaxScaler и MaxAbsScaler.

Для каждого метода масштабирования была написана функция scale(data, mode), которая принимает данные и тип масштабирования в качестве аргументов.

3. Обучение модели:

Обучение модели производится с использованием алгоритма KNN.

Функция train_model(x_train, y_train, n_neighbors, weights) принимает обучающие данные, количество соседей и тип весов в качестве аргументов, и возвращает обученную модель.

4. Предсказание и оценка:

Предсказание классов для тестовой выборки выполняется с помощью функции predict(classifier, x_test), которая принимает обученную модель и тестовые данные.

Оценка точности модели производится с помощью функции estimate(result, y_test), которая сравнивает предсказанные классы с истинными и вычисляет долю правильных предсказаний.

Выводы

В ходе выполнения данной лабораторной работы были изучены ключевые аспекты машинного обучения, такие как классификация данных с использованием алгоритма k-ближайших соседей (k-Nearest Neighbors, KNN). Программа, реализованная с помощью библиотеки sklearn, выполняет классификацию данных из датасета вин, учитывая такие параметры, как метод масштабирования данных, количество соседей (n_neighbors) и тип весов.

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.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler


def load_data(train_size=0.8):
    wine = datasets.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'):
                return KNeighborsClassifier(n_neighbors=n_neighbors, weights=weights).fit(x_train, y_train)


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


def estimate(result, y_test):
    accuracy = sum(result == y_test) / len(y_test)
    return round(accuracy, 3)


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