**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 |  | Старков С.А |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Освоить работу с библиотекой, понять ее предназначение, методы обработки данных, классификации данных а также способы оценки точности 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 c 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()* - загрузка данных.

* Загружает данные о вине из 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.

1. *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.

1. *predict()* – предсказывает следующие значения.

* Предсказывает метки классов для тестовых данных с помощью обученной модели.
* Предсказанные значения передаются в y\_pred.

1. *estimate()* – выполняет функцию оценки качества работы модели.

* Выполняет функцию вычисления предсказаний как долю правильных среди всех тестов.
* Имеет точность до трёх знаков после запятой.

1. *scale()* – обработка данных через scale.

* Используется для масштабирования данных по заданным скейлерам.
* 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 |

C увеличением размера обучающей выборки “train\_size” точность модели возрастает, что связанно с большим объемом данных, используемым для обучения. При малом размере выборки модель недостаточно обучается, что снижает точность предсказаний.

Использованы различные значения “n\_neighbors” из списка: 3, 5, 9, 15, 25. Результаты:

|  |  |
| --- | --- |
| n\_neighbors | accuracy |
| 3 | 0.741 |
| 5 | 0.759 |
| 9 | 0.759 |
| 15 | 0.759 |
| 25 | 0.741 |

Точность классификатора незначительно меняется при различных значениях “n\_neighbors”. Оптимальное значение находится в диапазоне от 5 до 15 при малом значении модель становится чувствительной к шуму, при большом ­— теряет точность.

Использованны различные скейлеры: StarndartScaler, MinMaxScaler, MaxAbsScaler. Результаты:

|  |  |
| --- | --- |
| scaler | accuracy |
| StandardScaler | 0.832 |
| MinMaxScaler | 0.801 |
| MaxAbsScaler | 0.779 |

Масштабирование данных улучшает качество классификации. Это связано с тем, что масштабирование нормализует данные, что улучшает работу алгоритма k-ближайших соседей.

Разработанный программный код см. в приложении А.

## Выводы

Была реализована программа, которая включает функции загрузки данных о винах, разделение их на обучающие и тестовые данные, обучения модели, применения этой модели на тестовых данных, оценки результатов и предобработки данных. Полученные результаты показывают, что размер обучающей выборки, количество соседей и способ предобработки данных оказывают значительное влияние на точность классификации.

# 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\_ratio=0.8, random\_seed=42):

wine\_data = datasets.load\_wine()

features = wine\_data.data[:, :2] # Выбираем только первые два признака

labels = wine\_data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

features, labels, train\_size=train\_ratio, random\_state=random\_seed)

return X\_train, X\_test, y\_train, y\_test

def train\_model(X\_train, y\_train, k\_neighbors=15, weight\_method='uniform'):

knn\_classifier = KNeighborsClassifier(n\_neighbors=k\_neighbors, weights=weight\_method)

knn\_classifier.fit(X\_train, y\_train)

return knn\_classifier

def predict(classifier, X\_test):

return classifier.predict(X\_test)

def estimate(predicted\_labels, y\_test):

accuracy = (predicted\_labels == y\_test).mean()

return round(accuracy, 3)

def scale(data, mode='standard'):

scaler\_mapping = {

'standard': StandardScaler(),

'minmax': MinMaxScaler(),

'maxabs': MaxAbsScaler()

}

scaler = scaler\_mapping.get(mode)

if scaler:

scaled\_data = scaler.fit\_transform(data)

return scaled\_data

else:

return None