**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. 3341 |  | Рябов М. Л. |
| Преподаватель |  | Иванов Д. В. |

Санкт-Петербург

2024

## Цель работы

Изучение основ анализа данных с применением языка Python и использованием библиотеки sklearn. Создание классификатора, его обучение и применение для классификации данных.

## 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

Описание функций:

1. *load\_data(train\_size=0.8)* функция загружает данные о вине из набора данных библиотеки sklearn. Затем данные разбиваются на обучающую и тестовую выборки. Размер обучающей выборки составляет 80%, также разбиение происходит с установленным параметром рандомизации.

2. *train\_model(X\_train, y\_train, n\_neighbors=15, weights='uniform')* функция создает экземпляр классификатора k ближайших соседей (значение k равно 15), который обучается на данных *X\_train, y\_train*, полученных в результате работы предыдущей функции.

3. *predict(classifier, X\_test)* функцияпринимает обученный в предыдущей функции классификатор (*classifier*), после чего делает предсказание классов для тестовых данных (*X\_test*) и возвращает его.

4. *estimate(predicitions, y\_test)* функция оценивает точность предсказаний классификатора, полученных на предыдущем шаге, сравнивая их с фактическими метками тестовых данных. Возвращает отношение верных предсказаний к их общему числу, округленное до тысячных.

5. *scale(X, mode='standard')* функцияпринимает данные и режим масштабирования (по умолчанию — Standard), после чего происходит масштабирование данных в соответствии с указанным режимом. Полученные данные возвращаются из функции.

Исследование работы классификатора, обученного на данных разного размера.

|  |  |
| --- | --- |
| train\_size | Точность |
| 0.1 | 0.528 |
| 0.3 | 0.722 |
| 0.5 | 0.611 |
| 0.7 | 0.667 |
| 0.9 | 0.611 |

Точность классификатора зависит от размера обучающей выборки. При слишком маленьком размере обучающей выборки модель может не получить достаточное количество информации для выявления закономерностей в данных. С другой стороны, при слишком большом размере обучающей выборки модель может начать избыточно подстраиваться под тренировочные данные и терять способность к обобщению на новых данных.

Исследование работы классификатора, обученного с различными значениями n\_neighbors

|  |  |
| --- | --- |
| n\_neighbors | Точность |
| 3 | 0.722 |
| 5 | 0.778 |
| 9 | 0.778 |
| 15 | 0.722 |
| 25 | 0.611 |

Точность классификатора зависит от количества соседей, используемых для классификации. Общий тренд показывает, что для данного набора данных оптимальными значениями n\_neighbors являются 5 и 9. Слишком маленькое значение n\_neighbors может привести к недостаточному обучению модели, в результате она будет чрезмерно чувствительна к шуму или выбросам, тогда как слишком большое значение n\_neighbors может привести к упрощению модели и потере способности к выявлению сложных закономерностей в данных.

Исследование работы классификатора с предобработанными данными

|  |  |
| --- | --- |
| Scaler | Точность |
| StandardScaler | 0.778 |
| MinMaxScaler | 0.833 |
| MaxAbsScaler | 0.889 |

Предобработка данных с использованием различных скейлеров позволяет улучшить качество работы классификатора. В данном случае наилучшим скейлером оказался MaxAbsScaler, масштабирующий признак по максимальному по модулю значению, сохраняя при этом знак. Это позволяет эффективно учитывать различия в масштабах признаков и повышает качество классификации.

Разработанный код см. в приложении А.

## Выводы

Были изучены основы анализа данных на языке *Python* с применением библиотеки *sklearn*. Разработаны функции для разделения данных для обучения и тестирования, обучения модели, вычисления предсказаний на основе данных и оценки качества полученных результатов 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.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

def load\_data(train\_size = 0.8):

wine = load\_wine()

trainX, testX, trainY, testY = train\_test\_split(wine.data[:, 0:2], wine.target, train\_size=train\_size,

test\_size=0.2, random\_state=42)

return trainX, testX, trainY, testY

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):

perdictions = classifier.predict(X\_test)

return perdictions

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

def estimate(predicitions, y\_test):

accuracy = accuracy\_score(y\_test, predicitions)

accuracy = round(accuracy, 3)

return accuracy