**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. 3341 |  | Кузнецова С.Е. |
| 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. Написание программы на языке программирования Python, которая проводит анализ существующего ассортимента, проверив возможности инструмента классификации данных для выделения различных классов.

## 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

Для анализа данных о классах вин была использована библиотека sklearn и встроенный в него набор данных о вине.

Реализованы функции:

1. *load\_data():* принимает на вход аргумент train\_size (размер обучающей выборки, по умолчанию равен 0.8), загружает набор данных о вине из библиотеки sklearn в переменную wine. Функция разбивает данные для обучения и тестирования в соответствии со значением train\_size с использованием функции train\_test\_split модуля sklearn.model\_selection. Функция возвращает значения X\_train, X\_test, y\_train, y\_test.

2*. train\_model():* принимает обучающую выборку (два аргумента - X\_train и y\_train) и аргументы n\_neighbors и weights (значения по умолчанию 15 и 'uniform' соответственно), создает экземпляр классификатора KNeighborsClassifier и загружает в него данные X\_train, y\_train c параметрами n\_neighbors и weights. Функция возвращает экземпляр классификатора.

3. *predict():* принимает обученную модель классификатора и тренировочный набор данных (X\_test), выполняет классификацию данных из X\_test. В качестве результата возвращает предсказанные данные.

4. *estimate():* функция принимает результаты классификации и истинные метки тестовых данных (y\_test), считает отношение предсказанных результатов, совпавших с «правильными» в y\_test к общему количеству результатов. В качестве результата возвращает полученное отношение, округленное до 0,001.

5. *scale():* функцияпринимает аргумент, содержащий данные, и аргумент mode - тип скейлера (допустимые значения: 'standard', 'minmax', 'maxabs', для других значений возвращается None, значение по умолчанию - 'standard'), обрабатывает данные соответствующим скейлером. В качестве результата возвращаются полученные после обработки данные.

Исследование работы классификатора, обученного на данных разного размера.

Точность работы классификаторов, обученных на данных от функции load\_data с разным значением аргумента train\_size представлена в таблице 1:

Таблица 1 – Точность работы классификаторов в зависимости от аргумента train\_size

|  |  |
| --- | --- |
| train\_size | Точность работы |
| 0.1 | 0.379 |
| 0.3 | 0.8 |
| 0.5 | 0.843 |
| 0.7 | 0.815 |
| 0.9 | 0.722 |

Заметим, что наибольшая точность достигается при значении train\_size = 0.5. Это обуславливается тем, что при меньших значениях модели не хватает данных для обучения, а при больших может происходить переобучение модели, которая будет избыточно подстраиваться под тренировочные данные.

Исследование работы классификатора, обученного с различными значениями n\_neighbors

Точность работы классификаторов, обученных с разным значением аргумента n\_neighbors представлена в таблице 2:

Таблица 2 – точность работы классификаторов в зависимости от аргумента n\_neighbors

|  |  |
| --- | --- |
| n\_neighbors | Точность работы |
| 3 | 0.861 |
| 5 | 0.833 |
| 9 | 0.861 |
| 15 | 0.861 |
| 25 | 0.833 |

Заметим, что наибольшая точность достигается при значениях n\_neighbors = 5 и 9. Слишком маленькое значение n\_neighbors может привести к недостаточному обучению модели, тогда как слишком большое значение аргумента может привести к переобучению модели и потере способности к выявлению закономерностей в данных.

Исследование работы классификатора с предобработанными данными.

Точность работы классификаторов, обученных на данных предобработанных с помощью скейлеров StandardScaler, MinMaxScaler, MaxAbsScaler представлена в таблице 3:

Таблица 3 – точность работы классификаторов в зависимости от скейлеров

|  |  |
| --- | --- |
| Scaler | Точность работы |
| StandardScaler | 0.889 |
| MinMaxScaler | 0.806 |
| MaxAbsScaler | 0.75 |

Наибольшая точность достигается при использовании скейлера StandardScaler. Это может быть связано с тем, что StandardScaler масштабирует данные таким образом, что их среднее значение равно 0, а стандартное отклонение — 1, что может улучшить работу некоторых моделей машинного обучения.

Разработанный код см. в приложении А.

## Выводы

Были изучены основы анализа данных на языке *Python* с применением библиотеки *sklearn*. Написана программа на языке программирования Python, которая проводит анализ существующего ассортимента, проверив возможности инструмента классификации данных для выделения различных классов.

# Приложение А Исходный код программы

Название файла: main.py

from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

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

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

wine = 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'):

neighbors = KNeighborsClassifier(n\_neighbors = n\_neighbors, weights = weights)

neighbors.fit(X\_train, y\_train)

return neighbors

def predict(clf, X\_test):

return clf.predict(X\_test)

def estimate(res, y\_test):

accur = accuracy\_score(y\_test, res)

accur = round(accur, 3)

return accur

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

if (mode == 'standard'):

scaler = StandardScaler()

return scaler.fit\_transform(data)

elif (mode == 'minmax'):

scaler = MinMaxScaler()

return scaler.fit\_transform(data)

elif (mode == 'maxabs'):

scaler = MaxAbsScaler()

return scaler.fit\_transform(data)

else:

return None