**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 |  | Кудин А.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Цель работы заключается в изучении и освоении методов анализа данных с использованием классификаторов, а также в получении практического опыта создания, использования и взаимодействия различных классификационных алгоритмов в программном коде. Работа направлена на анализ ассортимента вин с применением инструмента классификации данных, используя библиотеку sklearn и встроенный набор данных о вине.

## 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.

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 получения исходных данных и последующего их анализа была использована библиотека sklearn.

Описание реализации 5-ти функций:

load\_data(train\_size=0.8): Функция предназначена для загрузки данных и их разделения на тренировочные и тестовые наборы.

Аргументы:

train\_size (float, по умолчанию 0.8): Размер обучающей выборки.

Реализация:

Загружает данные о вине из библиотеки sklearn.

Берет первые два столбца из данных для использования в анализе.

Разделяет данные на обучающую и тестовую выборки с помощью функции train\_test\_split из sklearn.model\_selection, используя train\_size и фиксированный рандомизатор (random\_state=42).

Возвращает:

X\_train, X\_test: Двумерные массивы, содержащие обучающие и тестовые данные.

y\_train, y\_test: Одномерные массивы, содержащие метки классов для обучающей и тестовой выборок.

train\_model(X\_train, y\_train, n\_neighbors=15, weights='uniform'): Функция для обучения модели методом k-ближайших соседей (K-Nearest Neighbors).

Аргументы:

X\_train: Двумерный массив с обучающими данными.

y\_train: Одномерный массив с метками классов для обучающей выборки.

n\_neighbors (int, по умолчанию 15): Количество ближайших соседей.

weights (str, по умолчанию 'uniform'): Весовая функция, используемая в прогнозировании ('uniform' или 'distance').

Реализация:

Создает экземпляр классификатора KNeighborsClassifier с указанными параметрами.

Обучает классификатор на данных X\_train и y\_train.

Возвращает:

Обученный экземпляр классификатора.

predict(clf, X\_test): Функция для применения обученного классификатора к тестовым данным и получения предсказанных классов.

Аргументы:

clf: Обученный классификатор.

X\_test: Двумерный массив с тестовыми данными.

Реализация:

Выполняет классификацию данных из X\_test с использованием обученного классификатора clf.

Возвращает:

Одномерный массив предсказанных классов.

estimate(res, y\_test): Функция для оценки качества работы модели путем сравнения предсказанных классов с истинными метками тестовых данных.

Аргументы:

res: Одномерный массив предсказанных классов.

y\_test: Одномерный массив истинных меток тестовых данных.

Реализация:

Считает точность классификации как отношение числа правильных предсказаний к общему числу предсказаний, используя функцию accuracy\_score из sklearn.metrics.

Округляет полученную точность до трех знаков после запятой.

Возвращает:

Точность классификации (float).

scale(X, mode='standard'): Функция для нормализации данных с использованием различных типов скейлеров.

Аргументы:

X: Двумерный массив данных для нормализации.

mode (str, по умолчанию 'standard'): Тип скейлера ('standard', 'minmax', 'maxabs').

Реализация:

В зависимости от значения аргумента mode, выбирает соответствующий скейлер (StandardScaler, MinMaxScaler или MaxAbsScaler).

Применяет выбранный скейлер к данным X.

Если mode имеет недопустимое значение, возвращает None.

Возвращает:

Данные, нормализованные выбранным скейлером (двумерный массив).

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

Очевидно, что чем больше устанавливать train\_size, тем лучше будет обучаться модель, однако после какого-то порога видно, что модель начинает переобучаться

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

n\_neighbors является гиперпараметром для модели KNN поэтому точность прогнозов зависит от оптимального выбора этого параметра

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

Под данный датасет оптимальным оказался MaxAbsScaler

Разработанный код см. в приложении А.

## Выводы

Были изучены основы анализа данных с использованием языка Python и библиотеки sklearn. Разработаны функции для выполнения следующих задач: разделение данных на обучающие и тестовые наборы, обучение модели, предсказание результатов на основе данных и оценка качества 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.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

# Функция для загрузки данных и их разделения на тренировочные и тестовые наборы

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

# Загрузка данных

wine = datasets.load\_wine()

X = wine.data[:, :2] # Берем только первые два столбца

y = wine.target

# Разделение данных на обучающую и тестовую выборки

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

# Функция для обучения модели k-ближайших соседей

def train\_model(X\_train, y\_train, n\_neighbors=15, weights='uniform'):

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

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

return clf

# Функция для предсказания классов

def predict(clf, X\_test):

return clf.predict(X\_test)

# Функция для оценки качества модели

def estimate(res, y\_test):

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

return round(accuracy, 3)

# Функция для нормализации данных

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

if ((mode == 'standard')):

scaler = StandardScaler()

elif mode == 'minmax':

scaler = MinMaxScaler()

elif mode == 'maxabs':

scaler = MaxAbsScaler()

else:

return None

return scaler.fit\_transform(X)