**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: «Алгоритмы и структуры данных в Python»

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Корниенко А. Е. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

# Цель работы

Изучить основы анализа данных и машинного обучения, освоить основные инструменты для обработки и анализа данных.

# Задание

Вариант 1.

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. Загрузка данных:

Функция load\_data() загружает данные набора Wine из sklearn.datasets. Разделяет данные на обучающую и тестовую выборки с заданным split\_ratio, после чего возвращает обучающие и тестовые наборы признаков и меток.

1. Тренировка модели:

Функция train\_model() создает экземпляр KNeighborsClassifier с заданными n\_neighbors и weights. Обучает модель, после чего возвращает обученную модель.

1. Предсказание:

Функция predict() предсказывает значение x\_test. Возвращает вектор предсказанных меток.

1. Оценка:

Происходит предварительная обработки данных, которая преобразует признаки в заданный масштаб для обеспечения более стабильного обучения модели по одному из следующих алгоритмов..

1. Масштабирование данных:

Происходит предварительная обработки данных, которая преобразует признаки в заданный масштаб для обеспечения более стабильного обучения модели по одному из алгоритмов(StandardScaler, MinMaxScaler, MaxAbsScaler).

Исследование работы классификатора, обученного на данных разного размера:

|  |  |
| --- | --- |
| Размер обучающего набора | Точность |
| 0.1 | 0.545 |
| 0.3 | 0.725 |
| 0.5 | 0.861 |
| 0.7 | 0.915 |
| 0.9 | 0.924 |

Исследование работы классификатора, обученного с различными значениями n\_neighbors:

|  |  |
| --- | --- |
| Значение n\_neighbors | Точность |
| 3 | 0.861 |
| 5 | 0.889 |
| 9 | 0.921 |
| 15 | 0.935 |
| 25 | 0.919 |

Исследование работы классификатора с предобработанными данными:

|  |  |
| --- | --- |
| Метод предобработки | Точность |
| Без предобработки | 0.878 |
| StandardScaler | 0.945 |
| MinMaxScaler | 0.935 |
| MaxAbsScaler | 0.925 |

Разработанный программный код см. в приложении А.

# Выводы

В данной работе была разработана программа, которая обучает модель для предсказания классов вин. Также были проведены с ней некоторые исследования.

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 import preprocessing

from sklearn.metrics import accuracy\_score

def load\_data(split\_ratio=0.8):

data\_wine = load\_wine()

x = data\_wine.data[:, :2]

y = data\_wine.target

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

x, y, train\_size=split\_ratio, random\_state=42

)

return x\_train, x\_test, y\_train, y\_test

def train\_model(x\_train, y\_train, neighbors=15, weights='uniform'):

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

knn\_model.fit(x\_train, y\_train)

return knn\_model

def predict(clf, x\_test):

PredictData = clf.predict(x\_test)

return PredictData

def estimate(res, y\_test):

return round(accuracy\_score(y\_test, res), 3)

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

if mode == 'standard':

Scaler = preprocessing.StandardScaler()

elif mode == 'minmax':

Scaler = preprocessing.MinMaxScaler()

elif mode == 'maxabs':

Scaler = preprocessing.MaxAbsScaler()

else:

return None

return Scaler.fit\_transform(data)