**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3383 |  | Логинова А.Ю. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Целью данной работы является изучение процесса загрузки и разделения данных на обучающую и тестовую выборки, исследование процесса обучения модели KNN, оценка точности предсказаний модели на тестовой выборке и масштабирование данных. Для решения поставленных задач в данной лабораторной работе необходимо изучить модуль sklearn.

## 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 написания функций был использован модуль scikit-learn.

1. Загрузка и разделение данных:

Данные загружаются из библиотеки sklearn, используя функцию datasets.load\_wine().

Данные разделяются на обучающую и тестовую выборки с помощью функции train\_test\_split(), при этом используется только два первых признака для упрощения задачи классификации.

Параметр train\_size определяет долю данных, используемых для обучения, и по умолчанию составляет 80%.

1. Масштабирование данных:

Масштабирование данных выполняется с использованием трех различных методов: StandardScaler, MinMaxScaler и MaxAbsScaler.

Для каждого метода масштабирования была написана функция scale(data, mode), которая принимает данные и тип масштабирования в качестве аргументов.

1. Обучение модели:

Обучение модели производится с использованием алгоритма KNN.

Функция train\_model(x\_train, y\_train, n\_neighbors, weights) принимает обучающие данные, количество соседей и тип весов в качестве аргументов, и возвращает обученную модель.

1. Предсказание и оценка:

Предсказание классов для тестовой выборки выполняется с помощью функции predict(classifier, x\_test), которая принимает обученную модель и тестовые данные.

Оценка точности модели производится с помощью функции estimate(result, y\_test), которая сравнивает предсказанные классы с истинными и вычисляет долю правильных предсказаний.

## Выводы

В ходе выполнения данной лабораторной работы были изучены ключевые аспекты машинного обучения, такие как классификация данных с использованием алгоритма k-ближайших соседей (k-Nearest Neighbors, KNN). Программа, реализованная с помощью библиотеки sklearn, выполняет классификацию данных из датасета вин, учитывая такие параметры, как метод масштабирования данных, количество соседей (n\_neighbors) и тип весов.

# 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\_size=0.8):

wine = datasets.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'):

return KNeighborsClassifier(n\_neighbors=n\_neighbors, weights=weights).fit(x\_train, y\_train)

def predict(classifier, x\_test):

return classifier.predict(x\_test)

def estimate(result, y\_test):

accuracy = sum(result == y\_test) / len(y\_test)

return round(accuracy, 3)

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

if mode == 'standard':

return StandardScaler().fit\_transform(data)

elif mode == 'minmax':

return MinMaxScaler().fit\_transform(data)

elif mode == 'maxabs':

return MaxAbsScaler().fit\_transform(data)

else:

return None