**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. 3341 |  | Мильхерт А.С. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Изучить базовые принципы и инструменты анализа данных на языке *Python* с помощью библиотеки *sklearn*.

## 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].

## Выполнение работы

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

1.Описание реализации 5-ти функций:

Функция load\_data(train\_size=0.8) загружает данные о вине из набора данных в библиотеке sklearn.datasets. Данные разбиваются на обучающую и тестовую выборки с использованием функции train\_test\_split модуля sklearn.model\_selection. Размер обучающей выборки по умолчанию составляет 80%. Разбиение происходит с установленным параметром рандомизации для воспроизводимости результатов.

Функция train\_model(X\_train, y\_train, n\_neighbors=15, weights='uniform') создает экземпляр классификатора k-ближайших соседей (KNeighborsClassifier) с параметрами по умолчанию: число соседей равно 15, веса соседей равны 'uniform'. Затем этот классификатор обучается на обучающих данных, переданных в качестве аргументов X\_train и y\_train.

Функция predict(clf, X\_test) принимает обученную модель классификатора и тестовый набор данных и делает предсказание классов для тестовых данных с помощью метода predict классификатора. Результаты предсказаний возвращаются.

Функция estimate(res, y\_test) вычисляет точность предсказаний модели, сравнивая предсказанные значения с фактическими метками тестовых данных с помощью функции accuracy\_score из модуля sklearn.metrics. Результат выражается в виде отношения верных предсказаний к общему числу предсказаний и округляется до трех знаков после запятой.

Функция scale(X, mode='standard') принимает данные и режим масштабирования (по умолчанию - стандартное масштабирование). Внутри функции происходит масштабирование данных в соответствии с указанным режимом с использованием соответствующего преобразователя из библиотеки sklearn.preprocessing. Полученные данные возвращаются из функции.

2. 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 изменяется в зависимости от размера обучающей выборки. Это может быть связано с тем, что при слишком маленьком размере обучающей выборки модель может недообучиться, не получив достаточно информации для выявления закономерностей в данных. С другой стороны, при слишком большом размере обучающей выборки модель может переобучиться, избыточно подстраиваясь под тренировочные данные и теряя способность к обобщению на новые данные.

3. 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 меняется в зависимости 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 недообучению модели, когда 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.

3. 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, который масштабирует каждый 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

import sklearn as sl

from sklearn import datasets as ds

from sklearn import model\_selection as ms

from sklearn import neighbors as nb

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import MaxAbsScaler

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

wine = ds.load\_wine()

X, y = wine['data'][:, 0:2], wine['target']

X\_train, X\_test, y\_train, y\_test = ms.train\_test\_split(X, y, train\_size=train\_size,

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

return X\_train, X\_test, y\_train, y\_test

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

neigh = nb.KNeighborsClassifier(n\_neighbors=n\_neighbors, weights=weights)

return neigh.fit(X\_train, y\_train)

def predict(clf, X\_test):

return clf.predict(X\_test)

def estimate(res, y\_test):

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

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)