**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. 3344 |  | Кузнецов Р.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

## Цель работы

Знакомство с библиотеками для работы с анализом данных.

## 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, y\_train, X\_test, 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

load\_data – функция загружает данные в переменную wine, выбирает первые два столбца для X и целевые значения для y. Данные разделяются на обучающую и тестовую выборки и возвращаются. train\_model – создаётся и обучается классификатор на основе обучающей выборки, после чего он возвращается. predict – использует обученный классификатор для прогнозирования меток тестовых данных и возвращает предсказанные метки. estimate – вычисляет точность прогнозов путём сравнения предсказанных меток с истинными и возвращает значение точности. scale – масштабирует данные и возвращает их.

Обучение на данных разного размера с результатом работы:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| Точность работы | 0.379 | 0.797 | 0.8 | 0.822 | 0.843 | 0.819 | 0.815 | 0.861 | 0.722 |

Заметно, что и размер обучающей выборки и точность увеличивается, но при 0.9 размерности точность снижается. Это случается потому, что при переобучении модели, она начинает учитывать больше шума вместо закономерностей.

Обучение с различными значениями n\_neighbors с результатом работы:

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

С увеличением количества соседей точность сначала растёт, но при большом их количестве происходит переобучение модели и точность снижается.

Обучение с предварительно обработанными данными с результатом работы:

|  |  |  |  |
| --- | --- | --- | --- |
| Скейлер | StandartSc | MinMaxSc | MaxAbsSc |
| Точность | 0.417 | 0.417 | 0.278 |

Как можно заметить, StandartSc и MinMaxSc имеют одинаковую точность классификатора, но есть разница с MaxAbsSc.  Выбор способа масштабирования влияет на точность классификации.

## Выводы

В ходе лабораторной работы прошло ознакомление с основами анализа данных. Был получен опыт анализа данных с помощью библиотек языка python. При выполнении был получен опыт работы и знания о базовых концепциях основ анализа данных.

# 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 = wine.data[:, :2]

y = wine.target

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

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

def predict(clf, X\_test):

predictions = clf.predict(X\_test)

return predictions

def estimate(res, y\_test):

accuracy = (res == y\_test).mean()

return round(accuracy, 3)

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

if mode == 'standard':

scaler = StandardScaler()

elif mode == 'minmax':

scaler = MinMaxScaler()

elif mode == 'maxabs':

scaler = MaxAbsScaler()

else:

return None

scaled\_data = scaler.fit\_transform(data)

return scaled\_data