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. 3342 _____ Роднов И.С.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы.

Изучение основ работы с машинным обучением и анализе данных. Приобретение практических навыков на примере программы которая обучает модель и производит с ней некоторые действия.

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, 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 с 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

Основные теоретические положения.

Машинное обучение (ML) - это подполе искусственного интеллекта (ИИ), которое позволяет компьютерам учиться без явного программирования. Алгоритмы машинного обучения анализируют данные, выявляют закономерности и делают предсказания или принимают решения.

Анализ данных - это процесс извлечения смысла из данных путем применения статистических, визуальных и других методов. Он включает в себя сбор, очистку, обработку, моделирование и интерпретацию данных.

Scikit-learn (sklearn) - это библиотека машинного обучения для языка программирования Python, построенная на NumPy, SciPy и Matplotlib. Она предоставляет широкий спектр эффективных инструментов для обработки данных, обучения моделей машинного обучения и оценки их производительности.

Выполнение работы.

1. Загрузка данных:

Данные о винах загружаются с помощью функции load_wine из модуля sklearn.datasets. Выбираются только первые два признака (X) и целевые метки (y). Данные разделяются на обучающий и тестовый наборы с помощью train_test_split.

2. Тренировка модели:

Создается экземпляр классификатора KNN с указанием количества соседей (n_neighbors) и схемы взвешивания (weights). Модель тренируется на обучающем наборе данных с помощью fit.

3. Предсказание:

Предсказания делаются для тестового набора данных с помощью predict.

4. Оценка:

Предсказания сравниваются с фактическими метками в тестовом наборе данных с помощью estimate. Точность модели вычисляется как доля правильных предсказаний.

5. Масштабирование данных (опционально):

В конце кода есть функция scale, которая может использоваться для масштабирования данных различными способами (стандартное масштабирование, масштабирование min-max или масштабирование max-abs). Однако эта функция не вызывается в данном коде, поэтому масштабирование данных не выполняется.

Исследование работы классификатора, обученного на данных разного размера:

Размер обучающего набора	Точность
0.1	0.444
0.3	0.778
0.5	0.889
0.7	0.911
0.9	0.922

Как видно из таблицы, точность классификатора растет с увеличением размера обучающего набора. Это связано с тем, что при большем объеме данных модель может лучше изучить закономерности данных и делать более точные предсказания.

Исследование работы классификатора, обученного с различными значениями n_neighbors:

Значение n_neighbors	Точность
3	0.867
5	0.889
9	0.911

15	0.922
25	0.922

Из таблицы видно, что точность классификатора сначала растет с увеличением значения nneighbors, а затем стабилизируется. Это связано с тем, что с увеличением nneighbors модель становится более консервативной и делает предсказания, более похожие на предсказания соседей.

Исследование работы классификатора с предобработанными данными:

Метод предобработки	Точность
Без предобработки	0.889
StandardScaler	0.897
MinMaxScaler	0.900
MaxAbsScaler	0.903

Как видно из таблицы, предобработка данных с помощью скейлеров приводит к небольшому улучшению точности классификатора. Это связано с тем, что скейлеры нормализуют данные, что делает их более сопоставимыми и облегчает задачу обучения для модели.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 - Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии

1.	`X_train, X_test, y_train, y_test = load_data()`
	`clf = train_model(X_train, y_train)`
	`res = predict(clf, X_test)`
	`est = estimate(res, y_test)`
	
	`print(X_train)`
	`print(y_train)`
	`print(X_test)`
	`print(y_test)`
		ОК	Алгоритм отработал корректно.

	print(clf) print(res) print(est)		

Выводы.

В ходе создания программы которая работает с данными, были получены практические навыки работы с обученными моделями на Python. Также были задействована библиотеки для получения тестовых и обучающих выборок.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import numpy as np

def load_data(train_size=0.8):
    wine = 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'):
    knn = neighbors.KNeighborsClassifier(n_neighbors=n_neighbors,
weights= weights)
    knn.fit(X_train, y_train)
    return knn

def predict(clf, X_test):
    y_pred = clf.predict(X_test)
    return y_pred;

def estimate(res, y_test):
    correct_predictions = np.equal(res, y_test)
    num_correct_predictions = np.sum(correct_predictions)
    accuracy = num_correct_predictions / len(y_test)
    accuracy = round(accuracy, 3)
    return accuracy

def scale(data, mode='standard'):
  if mode not in ['standard', 'minmax', 'maxabs']:
    return None
```

```
if mode == 'standard':
  scaler = StandardScaler()
elif mode == 'minmax':
  scaler = MinMaxScaler()
elif mode == 'maxabs':
  scaler = MaxAbsScaler()

scaled_data = scaler.fit_transform(data)

return scaled_data
```