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 _____ Яковлева А.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Целью данной работы является:

- изучение основ анализа данных в Python
- разработка программы на языке Python с применением библиотеки *sklearn,* реализующей загрузку данных, обучение и применение модели, оценку качества полученных результатов.

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 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.

Выполнение работы

Функции:

1) *load_data(train_size = 0.8)* принимает на вход аргумент *train_size* (размер обучающей выборки, по умолчанию равен 0.8), в переменную *wine* загружает набор данных о вине из библиотеки *sklearn,* затем с помощью функции *train_test_split* разбивает данные на обучающую и тестовую выборки в соответствии со значением *train_size.*

2) *train_model(X_train, y_train, n_neighbors = 15, weights = 'uniform')* принимает обучающую выборку (два аргумента - *X_train* и *y_train*) и аргументы *n_neighbors* и *weights* (значения по умолчанию 15 и 'uniform' соответственно), создаёт экземпляр классификатора *KNeighborsClassifier* (метод k-ближайших соседей) с параметрами *n_neighbors* (количество соседей, которые будут использоваться) и *weights* (весовая функция, используемая при прогнозировании), затем обучается на данных *X_train*, *y_train* и возвращает данный экземпляр классификатора.

3) *predict(clf, X_test)* принимает обученную модель классификатора и тестовый набор данных *X_test,* с помощью метода *predict* выполняет классификацию данных из *X_test* и возвращает предсказанные данные.

4) *estimate(y_pred, y_test)* принимает результаты классификации и истинные метки тестовых данных (*y_test*), считает отношение предсказанных результатов, совпавших с «правильными» в *y_test* к общему количеству результатов и возвращает полученное отношение, округленное до 0,001.

5) *scale(data, mode = 'standard')* принимает данные, и аргумент *mode* - тип скейлера (допустимые значения: 'standard', 'minmax', 'maxabs', значение по умолчанию - 'standard'), возвращает данные, обработанные соответствующим скейлером, или *None*, если *mode* не соответствует допустимым значениям.

Таблица 1 – Исследование работы классификатора, обученного на данных разного размера

train_size	Точность работы
0.1	0.379
0.3	0.8
0.5	0.843

0.7	0.815
0.9	0.722

Наилучшая точность работы достигается при значениях *train_size 0.5* или *0.7*; при недостаточном размере обучающей выборки модели не хватает данных для обучения, с другой стороны, при слишком большом размере обучающей выборки может происходить переобучение модели.

Таблица 2 – Исследование работы классификатора, обученного с различными значениями n_neighbors

n_neighbors	Точность работы
3	0.861
5	0.833
9	0.861
15	0.861
25	0.833

Наилучшая точность достигается при значениях *n_neighbors* 3, 9 и 15. При слишком большом значении *n_neighbors* модель упрощается и её точность снижается.

Таблица 3 – Исследование работы классификатора с предобработанными данными

Скейлер	Точность работы
StandardScaler	0.889
MinMaxScaler	0.806
MaxAbsScaler	0.75

В данном случае наилучшим скейлером является *StandardScaler.* Предобработка данных с использованием этого скейлера улучшила точность работы классификатора.

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(0.5) clf = train_model(X_train, y_train) print(estimate(predict(clf, X_test), y_test))	0.843	Обучение на данных разного размера, при train_size = 0.5 точность 0.843
2.	X_train, X_test, y_train, y_test = load_data() clf = train_model(X_train, y_train, 5) print(estimate(predict(clf, X_test), y_test))	0.833	Обучение с различными значениями *n_neighbors*, при *n_neighbors* = 5 точность 0.833
3.	X_train, X_test, y_train, y_test = load_data() clf = train_model(scale(X_train, 'minmax'), y_train) print(estimate(predict(clf, scale(X_test, 'minmax')), y_test))	0.806	Обучение на данных предобработанных с помощью скейлеров, при *mode* = '*minmax*' точность 0.806

Выводы

В ходе выполнения лабораторной работы были изучены основы анализа данных. Была написана программа на языке Python с применением библиотеки *sklearn,* реализующая загрузку данных, обучение и применение модели, оценку качества полученных результатов.

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.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler

def load_data(train_size = 0.8):
    wine = load_wine()
    X_train, X_test, y_train, y_test = train_test_split(wine.data[:,0:2], 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(clf, X_test):
    return clf.predict(X_test)

def estimate(y_pred, y_test):
    return round(accuracy_score(y_test, y_pred), 3)

def scale(data, mode = 'standard'):
    if (mode == 'standard'):
        scaler = StandardScaler()
    elif (mode == 'minmax'):
        scaler = MinMaxScaler()
    elif (mode == 'maxabs'):
        scaler = MaxAbsScaler()
    else:
        return None
    return scaler.fit_transform(data)
```