МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Информатика»
Тема: «Введение в анализ данных»

Студент гр. 3342 _____ Романов Е.А

Преподаватель _____ Иванов Д. В.

Санкт-Петербург
2024

Цель работы

Изучить основные принципы анализа данных и машинного обучения. Написать программу, анализирующую набор данных и обучает модель для классификации новых.

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

Функция load_data загружает набор данных из библиотеки sklearn.datasets и выбирает первые два столбца в качестве признаков. Затем она разделяет данные на обучающую и тестовую выборки с помощью функции train_test_split из библиотеки sklearn.model_selection. Результатом работы функции являются тренировочные и тестовые данные.

Функция train_model обучает модель KNN с помощью функции KNeighborsClassifier из библиотеки sklearn.neighbors. Она принимает в качестве аргументов обучающую выборку и метки классов, а также параметры n_neighbors и weights, которые задают количество соседей и весовую функцию соответственно. Обученная модель возвращается функцией

Функция predict применяет обученную модель к тестовой выборке данных и возвращает предсказанные метки классов. С помощью метода `predict()` обученной модели выполняется прогнозирование классов для переданных данных `X_test`.

Функция estimate считает точность модели на основе предсказанных меток классов и фактических меток классов из тестовой выборки. Результат округляется до трех знаков после запятой и возвращается функцией.

Функция scale масштабирует данные с помощью одного из трех доступных способов нормализации: StandardScaler, MinMaxScaler или MaxAbsScaler из библиотеки sklearn.preprocessing, в зависимости от переданного режима. Она принимает в качестве аргументов данные и режим масштабирования, затем с помощью метода `scaler.fit_transform(data)` выбранный метод масштабирования применяется к переданным данным, которые возвращаются функцией.

Разработанный программный код см. в приложении А.

Тестирование

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

№	Размер выборки	Точность модели
1.	0.1	0.391
2.	0.3	0.752
3.	0.5	0.798
4.	0.7	0.796
5.	0.9	0.778

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

№	n_neighbors	Точность модели
1.	3	0.711
2.	5	0.733
3.	9	0.756
4.	15	0.756
5.	25	0.711

Таблица 3 - Исследование работы классификатора с предобработанными данными

№	Scaler	Точность модели
1.	StandardScaler	0.778
2.	MinMaxScaler	0.778
3.	MaxAbsScaler	0.778

Из полученных результатов в таблице 1 видно, что при увеличении размера обучающей выборки (train_size) точность классификатора возрастает, достигая максимального значения при размере выборки 0.7, при меньших значениях размера, модель обучается недостаточно хорошо, что приводит к уменьшению точности, а при больших размерах точно снижается из-за переобучения.

Анализ результатов в таблице 2 показывает, что увеличение значения параметра n_neighbors улучшает точность классификации, но эта тенденция сохраняется только до определённых значений, в данном случае наилучшею точность модель показывает при параметре n_ neighbors равном 9 и 15, после чего дальнейшее увеличение приводит к уменьшению точности.

Результаты из таблицы 3 показывают, что применение различных методов масштабирования данных не влияет на точность классификации. Такой результат может быть вызван тем, что данные уже были масштабированы.

Выводы

В ходе лабораторной работы были изучены основы анализа данных и машинного обучения. Разработана программа, выполняющая обучение модели на основе исходного набора 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_data = datasets.load_wine()

    features = wine_data.data[:, :2]
    target = wine_data.target

    X_train, X_test, y_train, y_test = train_test_split(features, 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'):
    model = KNeighborsClassifier(n_neighbors=n_neighbors, weights=weights)
    model.fit(X_train, y_train)
    return model

def predict(clf, X_test):
    predictions = clf.predict(X_test)
    return predictions

def estimate(predictions, y_test):
    accuracy = (predictions == 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
```