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. 3343 _____ Старков С.А

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Освоить работу с библиотекой, понять ее предназначение, методы обработки данных, классификации данных а также способы оценки точности классификации.

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, 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

Были реализованы 5 функций. Описание каждой функции и логика работы описаны ниже.

1. *load_data()* - загрузка данных.

- Загружает данные о вине из sklearn.
- Извлекает только первые два столбца из данных и метки классов.
- Делит данные на обучающую и тестовую выборки в соответствии с параметром train_size.
- Возвращает x_train, x_test, y_train, y_test.

2. *train_model()* - используется для обучения модели.

- Создаёт экземпляр классификатора K-ближайших соседей с заданными параметрами n_neighbors и weights.
- Обучает модель на данных x_train и y_train.
- Возвращает обученную модель.

3. *predict()* – предсказывает следующие значения.

- Предсказывает метки классов для тестовых данных с помощью обученной модели.
- Предсказанные значения передаются в y_pred.

4. *estimate()* – выполняет функцию оценки качества работы модели.

- Выполняет функцию вычисления предсказаний как долю правильных среди всех тестов.
- Имеет точность до трёх знаков после запятой.

5. *scale()* – обработка данных через scale.

- Используется для масштабирования данных по заданным скейлерам.
- Возвращает масштабированные данные.

- Если передан некорректный режим, возвращает None.

Для исследования точности классификатора при различных размерах обучающей выборки использовались значения train_size из списка: 0.1, 0.3, 0.5, 0.7, 0.9. Результаты приведены в таблице:

train_size	accuracy
0.1	0.611
0.3	0.593
0.5	0.685
0.7	0.741
0.9	0.778

С увеличением размера обучающей выборки “train_size” точность модели возрастает, что связанно с большим объемом данных, используемым для обучения. При малом размере выборки модель недостаточно обучается, что снижает точность предсказаний.

Использованы различные значения “n_neighbors” из списка: 3, 5, 9, 15, 25. Результаты:

n_neighbors	accuracy
3	0.741
5	0.759
9	0.759
15	0.759
25	0.741

Точность классификатора незначительно меняется при различных значениях “n_neighbors”. Оптимальное значение находится в диапазоне от 5 до 15 при малом значении модель становится чувствительной к шуму, при большом — теряет точность.

Использованны различные скейлеры: StarndartScaler, MinMaxScaler, MaxAbsScaler. Результаты:

scaler	accuracy
StandardScaler	0.832
MinMaxScaler	0.801
MaxAbsScaler	0.779

Масштабирование данных улучшает качество классификации. Это связано с тем, что масштабирование нормализует данные, что улучшает работу алгоритма k-ближайших соседей.

Разработанный программный код см. в приложении А.

Выводы

Была реализована программа, которая включает функции загрузки данных о винах, разделение их на обучающие и тестовые данные, обучения модели, применения этой модели на тестовых данных, оценки результатов и предобработки данных. Полученные результаты показывают, что размер обучающей выборки, количество соседей и способ предобработки данных оказывают значительное влияние на точность классификации.

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_ratio=0.8, random_seed=42):
    wine_data = datasets.load_wine()
    features = wine_data.data[:, :2]  # Выбираем только первые два признака
    labels = wine_data.target
    X_train, X_test, y_train, y_test = train_test_split(
        features, labels, train_size=train_ratio, random_state=random_seed)
    return X_train, X_test, y_train, y_test

def train_model(X_train, y_train, k_neighbors=15, weight_method='uniform'):
    knn_classifier = KNeighborsClassifier(n_neighbors=k_neighbors, weights=weight_method)
    knn_classifier.fit(X_train, y_train)
    return knn_classifier

def predict(classifier, X_test):
    return classifier.predict(X_test)

def estimate(predicted_labels, y_test):
    accuracy = (predicted_labels == y_test).mean()
    return round(accuracy, 3)

def scale(data, mode='standard'):
    scaler_mapping = {
        'standard': StandardScaler(),
```

```
    'minmax': MinMaxScaler(),
    'maxabs': MaxAbsScaler()
}
scaler = scaler_mapping.get(mode)
if scaler:
    scaled_data = scaler.fit_transform(data)
    return scaled_data
else:
    return None
```