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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Информатика»
Тема: Введение в анализ данных. Вариант 1.

Студент гр. 3341 Ступак А.А.

Преподаватель Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Изучить и научиться применять библиотеку языка Python scikit-learn для проведения анализа данных. Проверить возможности инструмента классификации данных с целью выделения из них различных классов, а также провести обработку 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.

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.

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

Описание функций:

- load_data(train_size=0.8): загружает набор данных о вине и разбивает его на обучающую и тестовую выборки в соответствии со значением train_size.
- train_model(X_train, y_train, n_neighbors=15, weights='uniform'): создает экземпляр классификатора k-ближайших соседей, после чего обучает модель на переданных обучающих данных.
- predict(clf, X_test): принимает на вход обученную модель и тестовые данные, после чего классифицирует их.
- estimate(res, y_test): оценивает и возвращает точность модели на тестовых данных.
- scale(data, mode='standard'): масштабирует данные скейлером, тип которого определяется аргументом mode.

Исследование работы классификатора, обученного на данных разного размера:

train_size	Точность классификатора
0.1	0.379
0.3	0.8
0.5	0.843
0.7	0.815
0.9	0.722

По результатам таблицы можно сделать вывод, что наименьшая точность будет наблюдаться при очень малом размере тренировочной выборки, так как данных для обучения недостаточно. С другой стороны при больших размерах тренировочной выборки можно вновь заметить постепенное снижение точности, так как происходит переобучение модели.

Исследование работы классификатора, обученного с различными значениями n_neighbors:

n_neighbors Точность классификатора

3 0.861

5 0.833

9 0.861

15 0.861

25 0.833

Из таблицы видно, что точность классификатора практически не зависит от количества соседей. Однако, если провести тестирование дальше и подставить значение n_neighbors, максимально близкое к значению n_samples_fit, точность упадет до 0.389, из чего следует вывод о том, что при тестировании с количеством соседей, близким к общему числу обучающих образцов качество классификации ухудшится.

Исследование работы классификатора с предобработанными данными:

Тип скейлера	Точность классификатора
StandardScaler	0.833
MinMaxScaler	0.806
MaxAbsScaler	0.75

Из таблицы следует, что наибольшая точность достигается при использовании стандартного скейлера, который менее чувствителен к выбросам в данных. MaxAbsScaler, напротив же, приводит к наименьшей точности предсказания.

Выводы

В ходе выполнения лабораторной работы были изучены и освоены необходимые навыки для применения библиотеки языка Python scikit-learn к решению практических задач по обучению модели и классификации данных. Были проверены возможности разных скейлеров и проведен анализ результатов, полученных в зависимости от разных параметров.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from sklearn import datasets, neighbors, model_selection, metrics, preprocessing

def load_data(train_size=0.8):
    wine = datasets.load_wine()
                    X_train,     X_test,     y_train,     y_test     = model_selection.train_test_split(wine.data,                 wine.target, train_size=train_size, random_state=42)
    return X_train[:, :2], X_test[:, :2], y_train, y_test

def train_model(X_train, y_train, n_neighbors=15, weights='uniform'):
          n  =  neighbors.KNeighborsClassifier(n_neighbors=n_neighbors, weights=weights)
    n.fit(X_train, y_train)
    return n

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

def estimate(res, y_test):
    accuracy = metrics.accuracy_score(y_test, res)
    return round(accuracy, 3)

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