

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

Малиновский А.А.

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

- о 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

- о оформите результаты пункта выше в виде таблицы

- о объясните полученные результаты

- исследование работы классификатора, обученного с различными значениями `n_neighbors`

- о приведите точность работы классификаторов, обученных со значением аргумента `n_neighbors` из списка: 3, 5, 9, 15, 25

- о 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)

- о оформите результаты в виде таблицы

- о объясните полученные результаты

- исследование работы классификатора с предобработанными данными

- о приведите точность работы классификаторов, обученных на данных предобработанных с помощью скейлеров из списка: `StandardScaler`, `MinMaxScaler`, `MaxAbsScaler`

- о 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.

- о оформите результаты в виде таблицы

- о объясните полученные результаты

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 <code>n_neighbors</code>	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
MinMaxScaler	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.

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

```
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'):
    return neighbors.KNeighborsClassifier(n_neighbors =
n_neighbors, weights = weights).fit(X_train, y_train)

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

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

def scale(data,mode='standard'):
    scalers = {
        'standard': preprocessing.StandardScaler(),
        'maxabs': preprocessing.MaxAbsScaler(),
        'minmax': preprocessing.MinMaxScaler()
    }
    if mode not in scalers.keys():
        return None
    return scalers[mode].fit_transform(data)
```