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 _____ Мильхерт А.С.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Изучить базовые принципы и инструменты анализа данных на языке *Python* с помощью библиотеки *sklearn*.

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].

Выполнение работы

Для получения исходных данных и последующего их анализа была использована библиотека *sklearn*.

1.Описание реализации 5-ти функций:

Функция load_data(train_size=0.8) загружает данные о вине из набора данных в библиотеке sklearn.datasets. Данные разбиваются на обучающую и тестовую выборки с использованием функции train_test_split модуля sklearn.model_selection. Размер обучающей выборки по умолчанию составляет 80%. Разбиение происходит с установленным параметром рандомизации для воспроизводимости результатов.

Функция train_model(X_train, y_train, n_neighbors=15, weights='uniform') создает экземпляр классификатора k-ближайших соседей (KNeighborsClassifier) с параметрами по умолчанию: число соседей равно 15, веса соседей равны 'uniform'. Затем этот классификатор обучается на обучающих данных, переданных в качестве аргументов X_train и y_train.

Функция predict(clf, X_test) принимает обученную модель классификатора и тестовый набор данных и делает предсказание классов для тестовых данных с помощью метода predict классификатора. Результаты предсказаний возвращаются.

Функция estimate(res, y_test) вычисляет точность предсказаний модели, сравнивая предсказанные значения с фактическими метками тестовых данных с помощью функции accuracy_score из модуля sklearn.metrics. Результат выражается в виде отношения верных предсказаний к общему числу предсказаний и округляется до трех знаков после запятой.

Функция scale(X, mode='standard') принимает данные и режим масштабирования (по умолчанию - стандартное масштабирование). Внутри функции происходит масштабирование данных в соответствии с указанным режимом с использованием соответствующего преобразователя из библиотеки sklearn.preprocessing. Полученные данные возвращаются из функции.

2. исследование работы классификатора, обученного на данных разного размера.

train_size	Точность
0.1	0.528
0.3	0.722
0.5	0.611
0.7	0.667
0.9	0.611

Точность классификатора изменяется в зависимости от размера обучающей выборки. Это может быть связано с тем, что при слишком маленьком размере обучающей выборки модель может недообучиться, не получив достаточно информации для выявления закономерностей в данных. С другой стороны, при слишком большом размере обучающей выборки модель может переобучиться, избыточно подстраиваясь под тренировочные данные и теряя способность к обобщению на новые данные.

3. исследование работы классификатора, обученного с различными значениями n_neighbors

n_neighbors	Точность
3	0.722
5	0.778
9	0.778
15	0.722
25	0.611

Точность классификатора меняется в зависимости от количества соседей, используемых для классификации. Общий тренд показывает, что для данного набора данных оптимальными значениями n_neighbors являются 5 и 9. Слишком маленькое значение n_neighbors может привести к недообучению модели, когда она будет чрезмерно чувствительна к шуму или выбросам, тогда как слишком большое значение n_neighbors может привести к упрощению модели и потере способности к выявлению сложных закономерностей в данных.

3. исследование работы классификатора с предобработанными данными

Scaler	Точность
StandardScaler	0.778

MinMaxScaler	0.833
MaxAbsScaler	0.889

Предобработка данных с использованием различных скейлеров позволяет улучшить качество работы классификатора. В данном случае наилучшим скейлером оказался MaxAbsScaler, который масштабирует каждый признак по максимальному по модулю значению, сохраняя при этом знак. Это позволяет эффективно учитывать различия в масштабах признаков и повышает качество классификации.

Разработанный код см. в приложении А.

Выводы

Были изучены основы анализа данных на языке *Python* с применением библиотеки *sklearn*. Разработаны функции для разделения данных для обучения и тестирования, обучения модели, вычисления предсказаний на основе данных и оценки качества полученных результатов классификации.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import sklearn as sl
from sklearn import datasets as ds
from sklearn import model_selection as ms
from sklearn import neighbors as nb
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import MaxAbsScaler


def load_data(train_size=0.8):
    wine = ds.load_wine()
    X, y = wine['data'][:, 0:2], wine['target']
    X_train, X_test, y_train, y_test = ms.train_test_split(X, y,
train_size=train_size,

test_size=0.2, random_state=42)
    return X_train, X_test, y_train, y_test


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


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


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


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