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. 3343 Лобова Е. И.

Преподаватель Иванов Д. В.

Санкт-Петербург

2024

Цель работы

Целью работы является введение в анализ данных, знакомство с базовыми понятиями темы и библиотекой scikit-learning для машинного обучения на Python.

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

В ходе выполнения лабораторной работы были реализованы следующие функции:

1. *def load_data(train_size = 0.8)* – загружает датасет с данными о вине (load_wines()) и разделяет данные на обучающую и тестовую выборки с помощью функции *train_test_split()* из библиотеки *sklearn.model_selection* в соотношении *train_size,* по умолчанию обучающая выборка будет занимать 80% данных, а тестовая выборка - 20%. Возвращает подмножество признаков и меток классов для обучающей и тестовой выборок. Подмножество признаков содержит только первые два столбца из исходного набора данных, благодаря срезу [: , :2].
2. *def train_model(X_train, y_train, n_neighbors = 15, weights = 'uniform')* - Создает экземпляр классификатора KNN с указанными параметрами:
    - *n_neighbors*: Число ближайших соседей, используемых для прогнозирования. По умолчанию 15.
    - *weights*: Тип весов, используемых для вычисления расстояния до соседей. По умолчанию *'uniform'*, что означает, что всем соседям присваивается одинаковый вес.

    Также обучает модель KNN на предоставленных обучающих данных *X_train* и *y_train* с помощью метода *fit()*.
3. *def predict(clf, X_test)* - вызывает метод *predict()* модели *clf*, который использует внутренние данные, вычисленные во время обучения, для прогнозирования меток классов для каждой точки данных в *X_test* и возвращает массив предсказанных меток классов.
4. *def estimate(res, y_test)* - Вызывает функцию *accuracy_score()* из библиотеки *sklearn.metrics* для вычисления точности предсказаний модели. Точность определяется как доля правильно предсказанных меток классов и округляется до трех знаков после запятой.

5. *def scale(X, mode = 'standard'*) - в зависимости от указанного метода масштабирования создает экземпляр соответствующего класса масштабирования из библиотеки *sklearn.preprocessing.*
    - *mode="standard": preprocessing.StandardScaler()* - центрирует и масштабирует данные, устанавливая среднее значение равным 0 и стандартное отклонение равным 1.
    - *mode="minmax": preprocessing.MinMaxScaler()* - масштабирует данные в диапазон от 0 до 1.
    - *mode="maxabs": preprocessing.MaxAbsScaler()* - масштабирует данные, деля каждое значение на максимальное абсолютное значение во всем наборе данных.

    Выполняет масштабирование данных X с использованием выбранного метода масштабирования, вызывая метод *fit_transform()* и возвращает масштабированные данные.

Исследование работы классификатора, обученного на данных разного размера:

train_size	Точность
0.1	0.379
0.3	0.8
0.5	0.843
0.7	0.815
0.8	0.861
0.9	0.722

Наибольшая точность работы классификаторов достигается при значении по умолчанию (train_size = 0.8). При значениях размера тренировочной выборки *train_size* в диапазоне от 0,3 до 0,8 модели демонстрируют высокую точность, в среднем около 0,8. Это указывает на то, что модели имеют достаточно данных для обучения и обобщения. Однако при *train_size* равном 0,1 точность резко падает, поскольку модели не хватает данных для правильного обучения. С другой

стороны, при *train_size* выше 0,8 точность также снижается, что связано с переобучением.

Исследование работы классификатора, обученного с различными значениями *n_neighbors:*

n_neighbors	Точность
3	0.861
5	0.833
9	0.861
15	0.861
25	0.833

Число ближайших соседей *n_neighbors* не оказывает значительного влияния на точность модели в данном примере. Точность незначительно снижается при значениях *n_neighbors* равных 5 и 25. Однако при больших значениях *n_neighbors* точность падает, это происходит потому, что при увеличении *n_neighbors* точки данных могут попадать в неправильные классы. Когда число рассматриваемых соседей велико, некоторые классы данных в окрестности исследуемой точки могут стать больше, и их влияние на исследуемую точку возрастает. Это приводит к неверной классификации.

Исследование работы классификатора с предобработанными данными:

Тип скейлера	Точность
StandardScaler	0.889
MinMaxScaler	0.806
MaxAbsScaler	0.75

Scaler StandardScaler дает наиболее точные предсказания, поскольку он нормализует данные и устраняет влияние экстремальных значений. MinMaxScaler дает умеренно точные предсказания, но он может быть чувствителен к выбросам и ненормальным значениям в наборе данных.

MaxAbsScaler дает наименее точные предсказания, поскольку он не устраняет влияние масштаба признаков.

Разработанный программный код см. в приложении А.

Выводы

Были изучены различных структуры данных и сложности их основных методов. Также была написана программа, в соответствии с заданным вариантом, в которой с помощью классов реализован однонаправленный связанный список с различными методами.

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.metrics import accuracy_score
from sklearn import preprocessing

def load_data(train_size = 0.8):
    wines = datasets.load_wine()
    X = wines.data
    y = wines.target
    X_train, X_test, y_train, y_test = train_test_split(X, y,
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 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(accuracy_score(res, y_test), 3)

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