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. 3344 _____ Кузнецов Р.А.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург

2024

Цель работы

Знакомство с библиотеками для работы с анализом данных.

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, 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.

Выполнение работы

load_data – функция загружает данные в переменную wine, выбирает первые два столбца для X и целевые значения для y. Данные разделяются на обучающую и тестовую выборки и возвращаются. train_model – создаётся и обучается классификатор на основе обучающей выборки, после чего он возвращается. predict – использует обученный классификатор для прогнозирования меток тестовых данных и возвращает предсказанные метки. estimate – вычисляет точность прогнозов путём сравнения предсказанных меток с истинными и возвращает значение точности. scale – масштабирует данные и возвращает их.

Обучение на данных разного размера с результатом работы:

Размер	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Точность работы	0.379	0.797	0.8	0.822	0.843	0.819	0.815	0.861	0.722

Заметно, что и размер обучающей выборки и точность увеличивается, но при 0.9 размерности точность снижается. Это случается потому, что при переобучении модели, она начинает учитывать больше шума вместо закономерностей.

Обучение с различными значениями n_neighbors с результатом работы:

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

С увеличением количества соседей точность сначала растёт, но при большом их количестве происходит переобучение модели и точность снижается.

Обучение с предварительно обработанными данными с результатом работы:

Скейлер	StandartSc	MinMaxSc	MaxAbsSc
Точность	0.417	0.417	0.278

Как можно заметить, StandartSc и MinMaxSc имеют одинаковую точность классификатора, но есть разница с MaxAbsSc. Выбор способа масштабирования влияет на точность классификации.

Выводы

В ходе лабораторной работы прошло ознакомление с основами анализа данных. Был получен опыт анализа данных с помощью библиотек языка python. При выполнении был получен опыт работы и знания о базовых концепциях основ анализа данных.

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_size=0.8):
    wine = datasets.load_wine()
    X = wine.data[:, :2]
    y = wine.target

    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=train_size, random_state=42)

    return X_train, X_test, y_train, y_test

def train_model(X_train, y_train, n_neighbors=15, weights='uniform'):
    clf = KNeighborsClassifier(n_neighbors=n_neighbors, weights=weights)

    clf.fit(X_train, y_train)

    return clf

def predict(clf, X_test):
    predictions = clf.predict(X_test)

    return predictions

def estimate(res, y_test):
    accuracy = (res == y_test).mean()

return round(accuracy, 3)
```

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

      scaled_data = scaler.fit_transform(data)

      return scaled_data
```