НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) Программная инженерия и компьютерные науки

Предмет: Интеллектуальный анализ данных

Задание № 1(дополнительное)

Тема: **Анализ данных flare1.xlsx**

Выполнили, гр 20209 Княжев М.А.

гр 20207 Намсараева Д.Ж.  
Преподаватель Борисова И.А.

Новосибирск  
2023

**Содержание**

Оглавление

[Задание: 2](#_Toc149630773)

[Выполнение лабораторной работы: 3](#_Toc149630774)

[Пункт1 3](#_Toc149630775)

[Пункт 2, 3 3](#_Toc149630776)

[Распределение по шкалам 4](#_Toc149630777)

[Метрики шкал 5](#_Toc149630778)

[Подсчет матрицы попарных расстояний 5](#_Toc149630779)

[Пункт 4 7](#_Toc149630780)

[Проверка гипотезы унимодальной компактности 7](#_Toc149630781)

[Проверка гипотезы полимодальной компактности 8](#_Toc149630782)

[Проверка гипотезы локальной компактности 11](#_Toc149630783)

[Пункт 5, 6 12](#_Toc149630784)

[Ссылка на архив с файлом: 13](#_Toc149630785)

# 

# Задание:

1. Изучить описание и скачать данные.

2. Разработать и реализовать процедуру вычисления расстояния между объектами в зависимости от типов шкал, используя метрики из лекций. Процедура должна легко адаптироваться под любые данные.

3. Определить типы шкал в предложенной таблице и подсчитать матрицу попарных расстояний.

4. Разработать и реализовать механизм проверки гипотезы компактности в трех ее вариантах: унимодальная компактность, полимодальная компактность и локальная компактность.

5. Ориентируясь на природу своей задачи, предложить свою гипотезу о связях между описывающими и целевым признаком в ней.

6. Вычислить подтвержденность и потенциальную опровержимость каждой гипотезы на своей таблице.

7. Написать и представить отчет о результатах проделанной командной работы.

# 

# Выполнение лабораторной работы:

Задание №1(дополнительное) было выполнено с помощью файла .ipynb.

Описание столбцов:

1. Code for class (modified Zurich class) (A,B,C,D,E,F,H)

2. Code for largest spot size (X,R,S,A,H,K)

3. Code for spot distribution (X,O,I,C)

4. Activity (1 = reduced, 2 = unchanged)

5. Evolution (1 = decay, 2 = no growth, 3 = growth)

6. Previous 24 hour flare activity code (1 = nothing as big as an M1, 2 = one M1, 3 = more activity than one M1)

7. Historically-complex (1 = Yes, 2 = No)

8. Did region become historically complex on this pass across the sun's disk (1 = yes, 2 = no)

9. Area (1 = small, 2 = large)

10. Area of the largest spot (1 = <=5, 2 = >5)

11. C-class flares production by this region in the following 24 hours (common flares); Number

12. M-class flares production by this region in the following 24 hours (moderate flares); Number

13. X-class flares production by this region in the following 24 hours (severe flares); Number

## Пункт1

Данные были взяты с сайта: https://archive.ics.uci.edu/dataset/89/solar+flare, где 11-13 – целевая переменная.

## Пункт 2, 3

Первым делом подключили библиотеки и считали данные с studet-mat.csv у которого разделители были ;

import pandas as pd

file\_path = 'flare1\_100.xlsx'

df = pd.read\_excel(file\_path)

Столбцы были разделены на 4 шкалы: порядковая(numeric\_df), наименований(categorical\_df), абсолютная(absolute\_df), бинарная(binare\_df).

numeric\_df = [4, 5, 6, 9, 10]

categorical\_df = [1, 2, 3]

absolute\_df = [11, 12, 13]

binare\_df = [7, 8]

num\_rows\_df = df.shape[0]

print(f"Количество строк датасета: {num\_rows\_df}")

num\_columns\_df = df.shape[1]

print(f"Количество столбцов датасета: {num\_columns\_df}")

num\_columns\_numeric\_df = len(numeric\_df)

print(f"Данные с шкалой порядка (количество столбцов: {num\_columns\_numeric\_df})")

print(numeric\_df)

num\_columns\_categorical\_df = len(categorical\_df)

print(f"\nДанные с шкалой наименований (количество столбцов: {num\_columns\_categorical\_df})")

print(categorical\_df)

num\_columns\_absolute\_df = len(absolute\_df)

print(f"\nДанные с абсолютной шкалой (количество столбцов: {num\_columns\_absolute\_df})")

print(absolute\_df)

num\_columns\_binare\_df = len(binare\_df)

print(f"\nДанные с бинарной шкалой (количество столбцов: {num\_columns\_binare\_df})")

print(binare\_df)

### Распределение по шкалам

Количество строк датасета: 100

Количество столбцов датасета: 14

Данные с шкалой порядка (количество столбцов: 5)

[4, 5, 6, 9, 10]

Данные с шкалой наименований (количество столбцов: 3)

[1, 2, 3]

Данные с абсолютной шкалой (количество столбцов: 3)

[11, 12, 13]

Данные с бинарной шкалой (количество столбцов: 2)

[7, 8]

### Метрики шкал

для наименований и как частного случая бинарной:

d(i,l) = 0, если (Xi = X и Xl = X) или (Xi = X И Xl = X)

d(i,l) = 1, если (Xi = X и Xl ≠ X) или (Xi ≠ X и Xl = X)

для порядка:

d(i,l) = 0, если (Xi > X и Xl > X) или (Xi < X и Xl < X) или (Xi = X и Xl = X)

d(i,l) = 1, если (Xi > X и Xl < X) или (Xi < X И Xl > X)

d(i,l) = 0.5, если (Xi >< X и Xl = X) или (Xi = X И Xl >< X)

для абсолютной:

d(i,l) = 0, если |(Xi - Xl)| / (X.max - X.min)

Пояснение, X пробегает все значения в столбце, а Xi это значение 1 объекта, Xl значений 2 объекта

Так как расстояния должны быть [0; 1], то нормируем их деля на количество всех объектов

### Подсчет матрицы попарных расстояний

matrix\_df = df.values.T

То есть каждый столбец переводим в строчный вид чтобы было проще использовать для вычислений.

Вводим 3 функции для подсчета расстояний:

matrix\_pairwise\_distances\_df = np.zeros((num\_rows\_df, num\_rows\_df), dtype=np.float64)

def calc\_categorical(i, l , column):

    d = np.float64(0)

    xi = column[i]

    xl = column[l]

    for x in column:

        if(((xi == x) and (xl == x)) or ((xi != x) and (xl != x))):

            d += 0

        else:

            d += 1

    d = d / np.float64(num\_rows\_df)

    return d

def calc\_numeric(i, l , column):

    d = np.float64(0)

    xi = column[i]

    xl = column[l]

    for x in column:

        if(((xi > x) and (xl > x)) or ((xi < x) and (xl < x)) or ((xi == x) and (xl == x))):

            d += 0

        elif(((xi > x) and (xl < x)) or ((xi < x) and (xl > x))):

            d += 1

        else:

            d += 0.5

    d = d / np.float64(num\_rows\_df)

    return d

def calc\_absolute(i, l , column):

    d = np.float64(0)

    xi = column[i]

    xl = column[l]

    d = abs(xi - xl) / (np.float64(max(column)) - np.float64(min(column)))

    return d

После чего в цикле, который фиксирует строки и пробегает только верхний треугольник матрицы попарных расстояний(matrix\_pairwise\_distances\_df) и дублирует значение симметрично относительно главной диагонали:

for i in range(0, num\_rows\_df - 1):

    for l in range(i + 1, num\_rows\_df):

        t = 0

        d = np.float64(0)

        for column in df.columns:

            if(column in categorical\_df):

                d += math.pow(calc\_categorical(i, l, matrix\_df[t]), 2)

            elif(column in absolute\_df):

                d += math.pow(calc\_absolute(i, l, matrix\_df[t]), 2)

            elif(column in numeric\_df):

                d += math.pow(calc\_numeric(i, l, matrix\_df[t]), 2)

            elif(column in binare\_df):

                d += math.pow(calc\_categorical(i, l, matrix\_df[t]), 2)

            t += 1

        matrix\_pairwise\_distances\_df[i][l] = math.sqrt(d) / math.sqrt(num\_columns\_df)

        matrix\_pairwise\_distances\_df[l][i] = matrix\_pairwise\_distances\_df[i][l]

pairwise\_distances\_df = pd.DataFrame(matrix\_pairwise\_distances\_df)

csv\_file\_path = "matrix\_pairwise\_distances\_df.csv"

pairwise\_distances\_df.to\_csv(csv\_file\_path, sep=';', index=False)

## Пункт 4

### Проверка гипотезы унимодальной компактности

Для проверки гипотезы унимодальной компактности разобьём 11-13 на 2 кластера следующий образом:

max\_sum = df[[11, 12, 13]].sum(axis=1).max()

min\_sum = df[[11, 12, 13]].sum(axis=1).min()

avg = (max\_sum - min\_sum) / 2

После чего считаем в кластерах среднее расстояние и считаем средние расстояния в 11-13. Вышло следующее:

Средняя оценка 1: 0.0267960707174414

Средняя оценка 2: 0.13322657112741237

Средняя по 11-13 столбцам: 0.2

Как можно заметить средние оценки 1 и 2 меньше чем “ Средняя по 11-13 столбцам”, из этого следует что **гипотеза унимодальной компактности выполняется.**

Ниже приведен код вычисляющий выше приведенные данные:

max\_sum = df[[11, 12, 13]].sum(axis=1).max()

min\_sum = df[[11, 12, 13]].sum(axis=1).min()

avg = (max\_sum - min\_sum) / 2

sums\_1 = []

sums\_2 = []

for index, row in df.iterrows():

    sum\_row = row[11] + row[12] + row[13]

    if sum\_row < avg:

        sums\_1.append(sum\_row)

    else:

        sums\_2.append(sum\_row)

def calc\_avg\_mark(samples):

    ptr = 0

    count = 0

    for i in samples:

        for j in samples[i:]:

            ptr += matrix\_pairwise\_distances\_df[i][j]

            count += 1

    return ptr / count

print(f"Средняя оценка 1: {calc\_avg\_mark(sums\_1)}")

print(f"Средняя оценка 2: {calc\_avg\_mark(sums\_2)}")

print(f"Средняя среди всех средних оценок: {calc\_avg\_mark(range(len(df[11])))}")

### Проверка гипотезы полимодальной компактности

Воспользуемся преобразования пандаса чтобы привести все значения столбцов к численному:

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

binary\_arr = ['7', '8', '1\_B', '1\_C', '1\_D', '1\_E', '1\_F', '1\_H',

 '2\_X', '2\_R', '2\_S', '2\_A', '2\_H', '2\_K', '3\_X', '3\_O', '3\_I', '3\_C']

df.columns = df.columns.map(str)

df\_polymodal = pd.get\_dummies(df, columns=['1', '2', '3'])

df\_polymodal.columns = df\_polymodal.columns.map(str)

for binary in binary\_arr:

    df\_polymodal[binary] = df\_polymodal[binary].astype('category')

    df\_polymodal[binary] = df\_polymodal[binary].cat.codes

После чего воспользуемся методом K-Means для анализа данных, а именно на сколько кластеров оптимально разделить данные:

inertia = []

for n\_clusters in range(1, 8):

    kmeans = KMeans(n\_clusters=n\_clusters, random\_state=0)

    kmeans.fit(df\_polymodal)

    inertia.append(kmeans.inertia\_)

plt.figure(figsize=(8, 6))

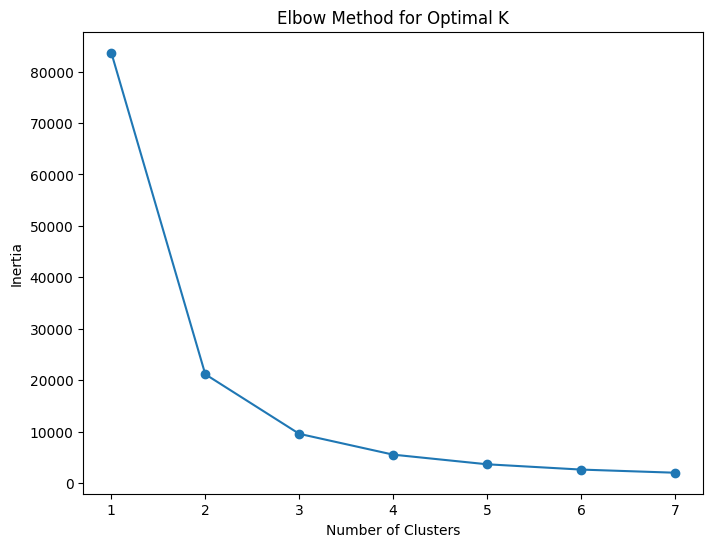
plt.plot(range(1, 8), inertia, marker='o')

plt.xlabel('Number of Clusters')

plt.ylabel('Inertia')

plt.title('Elbow Method for Optimal K')

plt.show()



Из графика видно, что самое оптимальное разделение на 3 кластера.

Тогда для проверки полимодальной компактности создадим новый столбец в датафрейме(gradeclass), который будет маркером категорий. А именно от 0 до 1, от 2 до 3, от 4 до 4+. После чего сравниваем метки категорий с метками кластера и помечаем их либо true либо false. Ниже приведен код подсчитывающий этот алгоритм:

optimal\_n\_clusters = 3

kmeans = KMeans(n\_clusters=optimal\_n\_clusters, random\_state=0)

kmeans.fit(df\_polymodal)

y\_pred = kmeans.fit\_predict(df\_polymodal)

array = []

for index, row in df.iterrows():

    sum\_row = row[11] + row[12] + row[13]

    if i < 2:

        array.append(0)

    elif i < 4:

        array.append(1)

    else:

        array.append(2)

df\_polymodal['gradeclass'] = array

d = {'Target': df\_polymodal['gradeclass'], 'Prediction' : y\_pred}

result = pd.DataFrame(d, columns = ['Target', 'Prediction'])

comparison = np.where(result['Target'] == result['Prediction'], True, False)

result['Comparison'] = comparison

result['Comparison'].value\_counts(normalize = True).round(2)

Вывод данных:

Comparison

False 0.67

True 0.33

Name: proportion, dtype: float64

Отсюда следует что больше не совпадений, а значит **гипотеза полимодальной компактности не выполняется**.

## 

### Проверка гипотезы локальной компактности

Для проверки гипотезы локальной компактности разобьём 11-13 на 2 кластера, как в унимодальной компакнтонсти. Также воспользуемся матрицей попарных расстояний. В матрице проходимся по строчки и ищем по столбцам для них ближайший объект. После чего проверяем, что у обоих объектов значение 11-13 лежит в одном кластере и подчитываем количество таких совпадений. Ниже приведен код описывающий этот алгоритм:

result = 0

max\_sum = df[['11', '12', '13']].sum(axis=1).max()

min\_sum = df[['11', '12', '13']].sum(axis=1).min()

avg = (max\_sum - min\_sum) / 2

for i in range(len(matrix\_pairwise\_distances\_df)):

    nearest\_obj\_idx = np.argmin(np.delete(matrix\_pairwise\_distances\_df[i], i))

    if nearest\_obj\_idx >= i:

        nearest\_obj\_idx += 1

    G3\_i\_value = df['11'].iloc[i] + df['12'].iloc[i] + df['13'].iloc[i]

    G3\_nearest\_obj\_value = df['11'].iloc[nearest\_obj\_idx] + df['12'].iloc[nearest\_obj\_idx] + df['13'].iloc[nearest\_obj\_idx]

    if(G3\_i\_value < avg and G3\_nearest\_obj\_value < avg):

        result += 1

    elif(G3\_i\_value >= avg and G3\_nearest\_obj\_value >= avg):

        result += 1

local\_compactness\_percentage = (result / len(matrix\_pairwise\_distances\_df)) \* 100

print(f"Процент локальной компактности: {local\_compactness\_percentage:.2f}%")

Вывод данных: Процент локальной компактности: 98.00%.

Это значит что **гипотеза локальной компактности выполняется.**

## Пункт 5, 6

Гипотеза: проанализировав данные, можно заметить, что у кого в столбце guardian значение father, то у того оценка часто от 6 до 15.

Проверяем подверженность:

condition\_true = (df['1'] == 'D') & (df['2'] == 'K') & (df['11'] + df['12'] + df['13'] != 0)

condition\_false = (df['1'] == 'D') & (df['2'] == 'K') & (df['11'] + df['12'] + df['13'] == 0)

count\_true = condition\_true.sum()

count\_false = condition\_false.sum()

print(f'Процент правильных гипотез: {count\_true / (count\_true + count\_false) \* 100}%')

Вывод данных:

Процент правильных гипотез: 40.0%

Как можно заметить довольно средний процент совпадения.

Проверим потенциальную опровержимость с помощью метода бутсрапа:

 df\_bootstrap = df.copy()

n\_iterations = 1000

results = []

for \_ in range(n\_iterations):

    bootstrap\_sample = df\_bootstrap.sample(frac=1, replace=True)

    father\_guardian\_students = bootstrap\_sample[(bootstrap\_sample['1'] == 'D') & (bootstrap\_sample['2'] == 'K')]

    students\_in\_range = father\_guardian\_students[(father\_guardian\_students['11'] + father\_guardian\_students['12'] + father\_guardian\_students['13'] != 0)]

    percentage\_in\_range = (len(students\_in\_range) / len(father\_guardian\_students)) \* 100

    results.append(percentage\_in\_range)

std\_deviation = np.std(results)

print(f"Стандартное отклонение для percentage\_in\_range: {std\_deviation:.2f}")

Вывод данных:

Стандартное отклонение для percentage\_in\_range: 16.35

Из этого можно сделать вывод что при случайном выборе строк датафрейма, стандартное отклонение довольно значительно, а значит, что гипотеза потенциальна опровержима.

# Ссылка на архив с файлом:

https://github.com/Knyazhev-Misha/IAD-NSU