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

Тема: **Анализ данных student-mat**

Выполнили, гр 20209 Княжев М.А.

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

Новосибирск  
2023

**Содержание**

Оглавление

[Задание: 2](#_Toc149620692)

[Выполнение лабораторной работы: 3](#_Toc149620693)

[Пункт1 3](#_Toc149620694)

[Пункт 2, 3 3](#_Toc149620695)

[Распределение по шкалам 4](#_Toc149620696)

[Метрики шкал 4](#_Toc149620697)

[Подсчет матрицы попарных расстояний 5](#_Toc149620698)

[Пункт 4 7](#_Toc149620699)

[Проверка гипотезы унимодальной компактности 7](#_Toc149620700)

[Проверка гипотезы полимодальной компактности 8](#_Toc149620701)

[Проверка гипотезы локальной компактности 10](#_Toc149620702)

[Пункт 5, 6 11](#_Toc149620703)

[Ссылка на архив с файлом: 12](#_Toc149620704)

# 

# 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 было выполнено с помощью файла .ipynb

## Пункт1

Данные были взяты с сайта: <https://archive.ics.uci.edu/dataset/320/student+performance>, где G3 – целевая переменная.

## Пункт 2, 3

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

import pandas as pd

import numpy as np

import math

csv\_file\_path = "student-mat.csv"

df = pd.read\_csv(csv\_file\_path, sep=";")

В основном заметно что многие столбцы имели тип int64, это будет использовано для шкал порядка, а также выделили два столбца, которые относятся к абсолютным шкалам: age, absences. С помощью этого столбцы были разделены на 3 шкалы: порядковая(numeric\_df), наименований(categorical\_df), абсолютная(absolute\_df).

numeric\_df = []

categorical\_df = []

absolute\_df = ['age', 'absences']

for column in df.columns:

    if df[column].dtype in ['int64', 'float64'] and column not in absolute\_df:

        numeric\_df.append(column)

    elif column not in absolute\_df:

        categorical\_df.append(column)

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)

### Распределение по шкалам

Количество строк датасета: 395

Количество столбцов датасета: 33

Данные с шкалой порядка (количество столбцов: 14)

['Medu', 'Fedu', 'traveltime', 'studytime', 'failures', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'G1', 'G2', 'G3']

Данные с шкалой наименований (количество столбцов: 17)

['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic']

Данные с абсолютной шкалой (количество столбцов: 2)

['age', 'absences']

### Метрики шкал

для наименований:

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)

            else:

                d += math.pow(calc\_numeric(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 G3 на 3 кластера, от 0 до 8, от 9 до 17, от 18 до 20. После чего считаем средние значения в кластерах и считаем средние расстояния в G3. Вышло следующее:

Высшая средняя оценка: 0.43736511444860177

Средняя среди средних оценок: 0.47053059624355

Низшая средняя оценка: 0.4777465469446674

Средняя среди всех средних оценок: 0.4742699160289717

Как можно заметить “Низшая средняя оценка” выше “ Средняя среди всех средних оценок”, из этого следует что **гипотеза унимодальной компактности не выполняется.**

Ниже приведен код вычисляющий выше приведенные данные:

G3 = df['G3'].values.reshape(-1, 1)

top = 17

middle = 8

top\_mark = []

middle\_mark = []

bottom\_mark = []

for i in range(len(G3)):

    if G3[i] >= top:

        top\_mark.append(i)

    elif G3[i] >= middle:

        middle\_mark.append(i)

    else:

        bottom\_mark.append(i)

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"Высшая средняя оценка: {calc\_avg\_mark(top\_mark)}")

print(f"Средняя средни средних оценок: {calc\_avg\_mark(middle\_mark)}")

print(f"Низшая средняя оценка: {calc\_avg\_mark(bottom\_mark)}")

print(f"Средняя среди всех средних оценок: {calc\_avg\_mark(range(len(G3)))}")

### 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 = ['school', 'sex', 'address', 'famsize', 'Pstatus',

'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic',

'Mjob\_teacher', 'Mjob\_health', 'Mjob\_services', 'Mjob\_at\_home', 'Mjob\_other',

'Fjob\_teacher', 'Fjob\_health', 'Fjob\_services', 'Fjob\_at\_home', 'Fjob\_other',

'reason\_home', 'reason\_reputation', 'reason\_course', 'reason\_other',

'guardian\_mother', 'guardian\_father', 'guardian\_other']

df\_polymodal = pd.get\_dummies(df, columns=['Mjob', 'Fjob', 'reason', 'guardian'])

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(2, 6):

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

    kmeans.fit(df\_polymodal)

    inertia.append(kmeans.inertia\_)

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

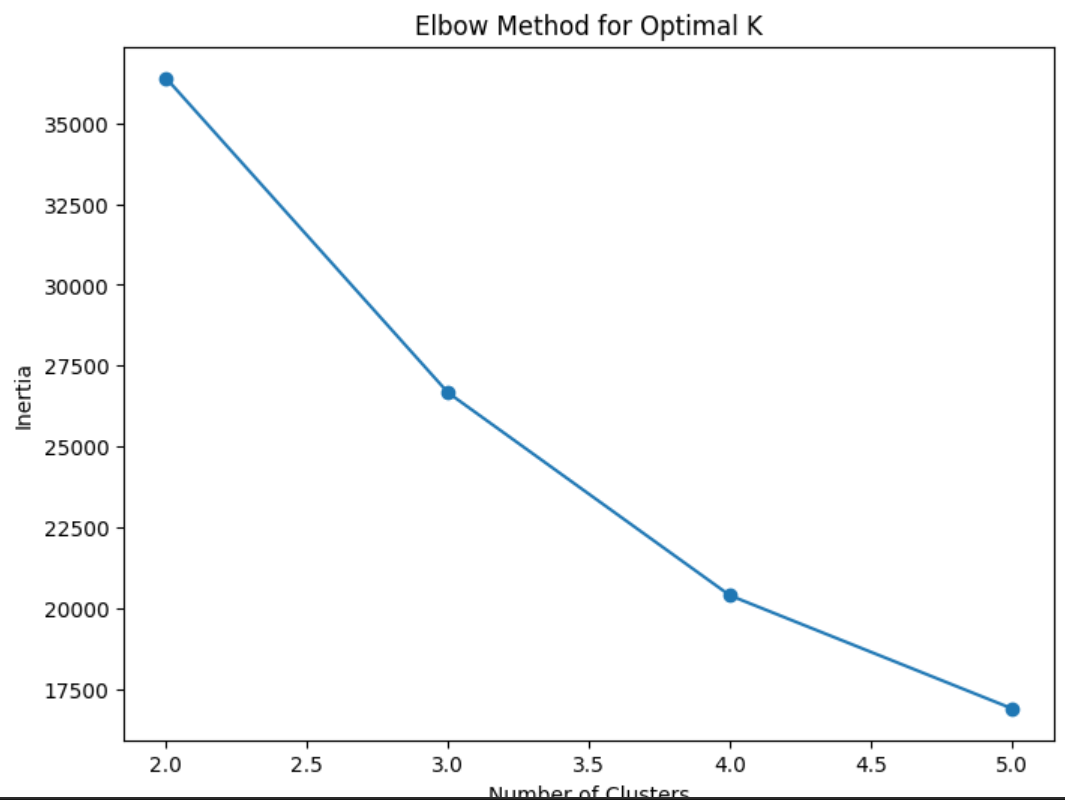
plt.plot(range(2, 6), inertia, marker='o')

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 4 кластера.

Тогда для проверки полимодальной компактности создадим новый столбец в датафрейме(gradeclass), который будет маркером категорий. А именно от 0 до 4, от 5 до 10, от 11 до 16, от 17 до 20. После чего сравниваем метки категорий с метками кластера и помечаем их либо true либо false. Ниже приведен код подсчитывающий этот алгоритм:

optimal\_n\_clusters = 4

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

kmeans.fit(df\_polymodal)

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

array = []

for i in df['G3']:

    if i < 5:

        array.append(0)

    elif i < 11:

        array.append(1)

    elif i < 17:

        array.append(2)

    else:

        array.append(3)

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 0.75

True 0.25

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 G3 на 3 кластера, от 0 до 8, от 9 до 17, от 18 до 20. 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 G3 лежит в одном кластере и подчитываем количество таких совпадений. Ниже приведен код описывающий этот алгоритм:

result = 0

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['G3'].iloc[i]

    G3\_nearest\_obj\_value = df['G3'].iloc[nearest\_obj\_idx]

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

        result += 1

    elif((G3\_i\_value < top and G3\_nearest\_obj\_value < top)

    and

    (G3\_i\_value >= middle and G3\_nearest\_obj\_value >= middle)

    ):

        result += 1

    elif((G3\_i\_value < 21 and G3\_nearest\_obj\_value < 21)

    and

    (G3\_i\_value >= top and G3\_nearest\_obj\_value >= top)

    ):

        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: 72.41%.

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:

father\_guardian\_students = df[df['guardian'] == 'father']

average\_grade\_father\_guardian = father\_guardian\_students['G3'].mean()

students\_in\_range = father\_guardian\_students[(father\_guardian\_students['G3'] >= 6) & (father\_guardian\_students['G3'] <= 15)]

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

print(f'Средняя оценка для студентов с отцом в качестве опекуна: {average\_grade\_father\_guardian:.2f}')

print(f'Процент студентов с отцом в качестве опекуна, находящихся в диапазоне от 6 до 15: {percentage\_in\_range:.2f}%')

Вывод данных:

Средняя оценка для студентов с отцом в качестве опекуна: 10.69

Процент студентов с отцом в качестве опекуна, находящихся в диапазоне от 6 до 15: 78.89%

Как можно заметить довольно высокий 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['guardian'] == 'father']

    average\_grade\_father\_guardian = father\_guardian\_students['G3'].mean()

    students\_in\_range = father\_guardian\_students[(father\_guardian\_students['G3'] >= 6) & (father\_guardian\_students['G3'] <= 15)]

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

    results.append((average\_grade\_father\_guardian, percentage\_in\_range))

std\_deviation = np.std([result[0] for result in results])

print(f"Стандартное отклонение: {std\_deviation:.2f}")

Вывод данных:

Стандартное отклонение: 0.48

Из этого можно сделать вывод что при случайном выборе строк датафрейма, стандартное отклонение оценок и процента студентов у которых оценка от 6 до 15 при условии, что guardian = father меньше 1, а значит, что гипотеза слабо потенциальна опровержима.

# Ссылка на архив с файлом:

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