|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ) КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 6

по дисциплине «Интеллектуальные технологии информационной безопасности»

**Тема: «Алгоритмы кластерного анализа данных»**

Вариант 11

Выполнил: Мишков А.О., студент группы ИУ8-62

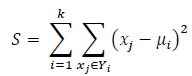
Проверил: Строганов И.С., преподаватель каф. ИУ8

г. Москва, 2022 г.

# Цель работы

Исследовать применение основных алгоритмов кластерного анализа, включая их модификации, на примере различных типов данных.

# Условие

Алгоритм: k-средних

где k – количество кластеров (задано заранее); 𝑌𝑖 – полученные кластеры, - центры масс

Исходные кластеризуемые данные: координатные точки на форме приложения

Расстояние: Манхэттен, Чебышева

Координатные точки: (20, -10);(3, 9);(36, -15);(26, 21);(12, -9);(-11, 33);(12, 8);(-30, 15);(-25, 8)

Координаты точек кластеров:

Манхэттен: (23, 7);(-4, 17);(5, 2)

Чебышев: (17, 8);(5, 1);(-12, 19)

# Ход работы

Суть алгоритма заключается в следующем: на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге.

Затем кластеризуемые точки разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Алгоритм завершается, когда на i-й итерации не изменяется центр масс кластеров.

Ниже представлен результат работы программы для Манхэттенского расстояния (рисунок 1 – 4):

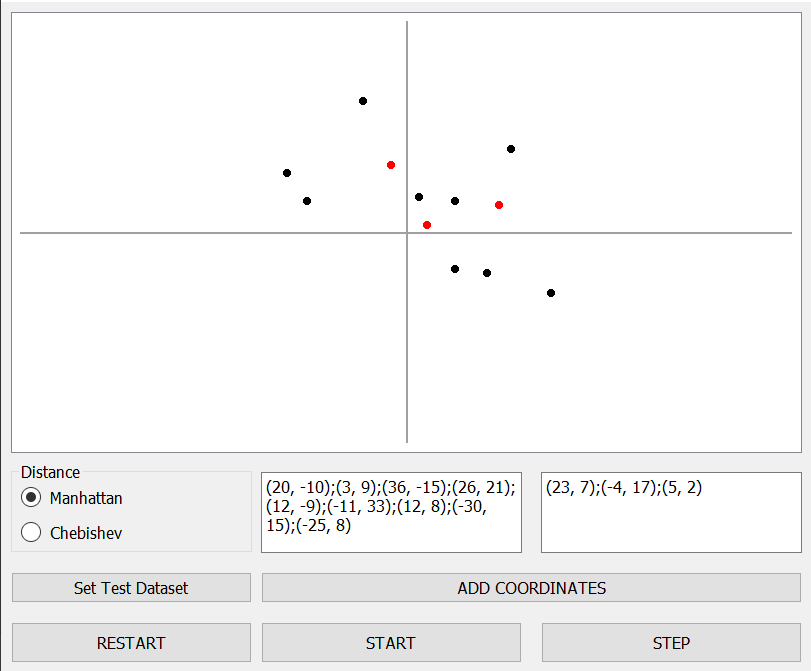


Рисунок 1 – Начальные данные

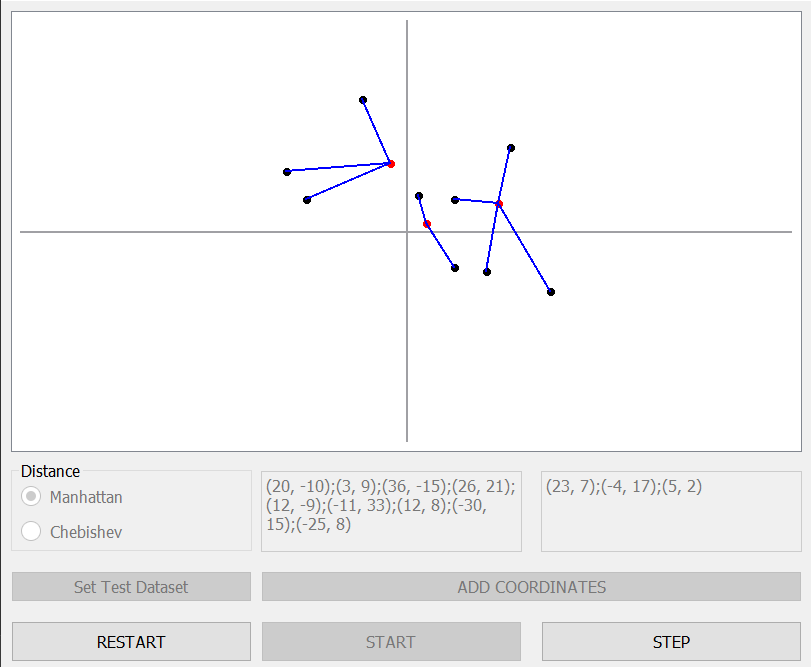


Рисунок 2 – Кластеризация на 1 шаге

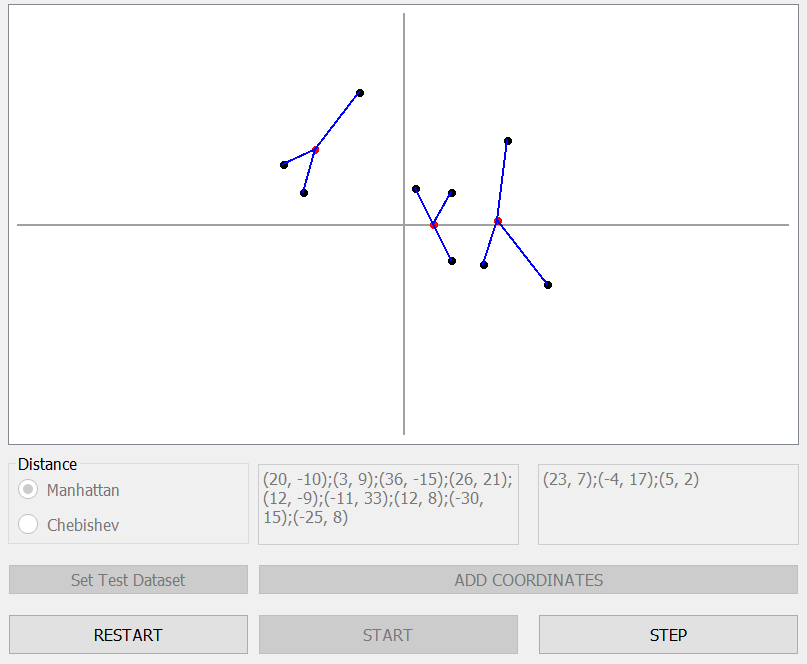


Рисунок 3 – Кластеризация на 2 шаге

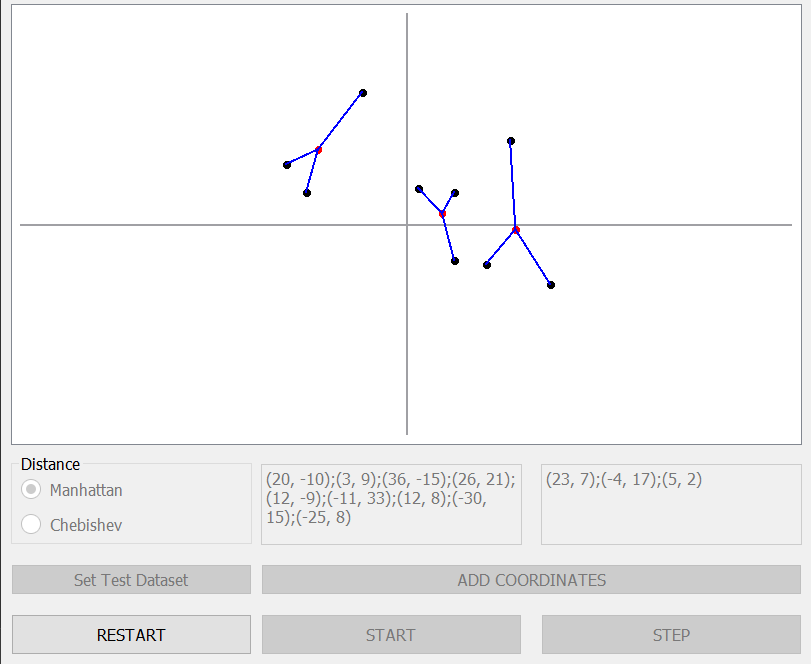


Рисунок 4– Кластеризация финальная

Ниже представлен результат работы программы для расстояния Чебышева (рисунок 5 – 8):

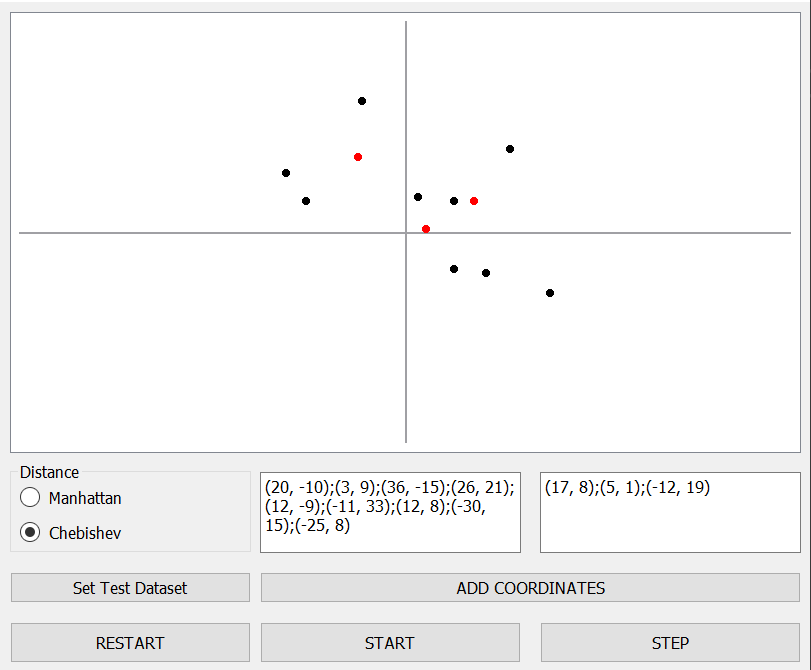


Рисунок 5 – Начальные данные

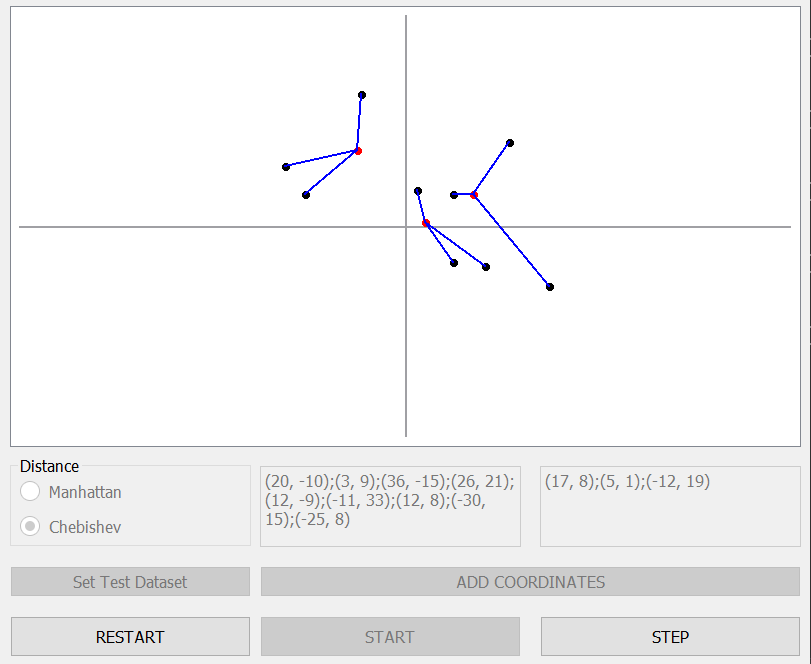


Рисунок 6 – Кластеризация на 1 шаге

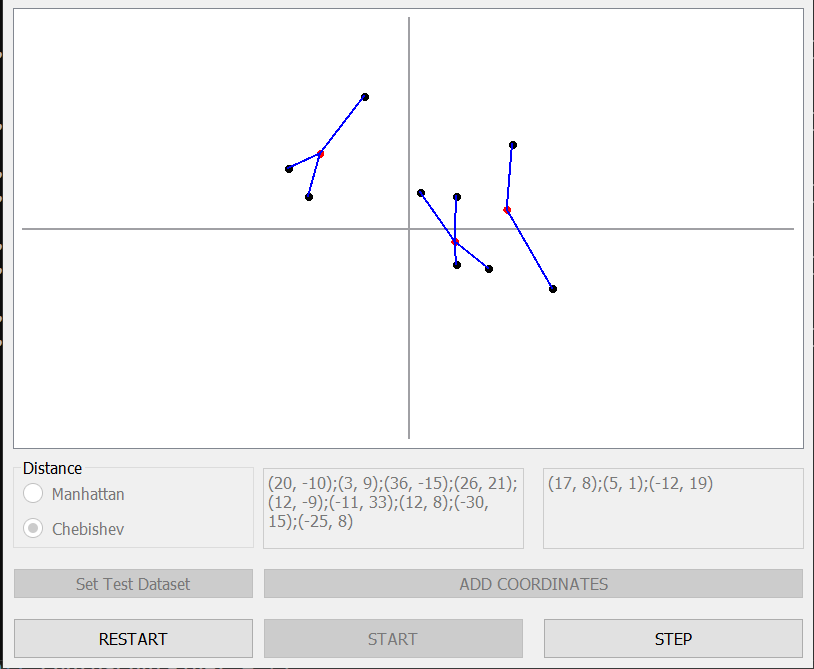


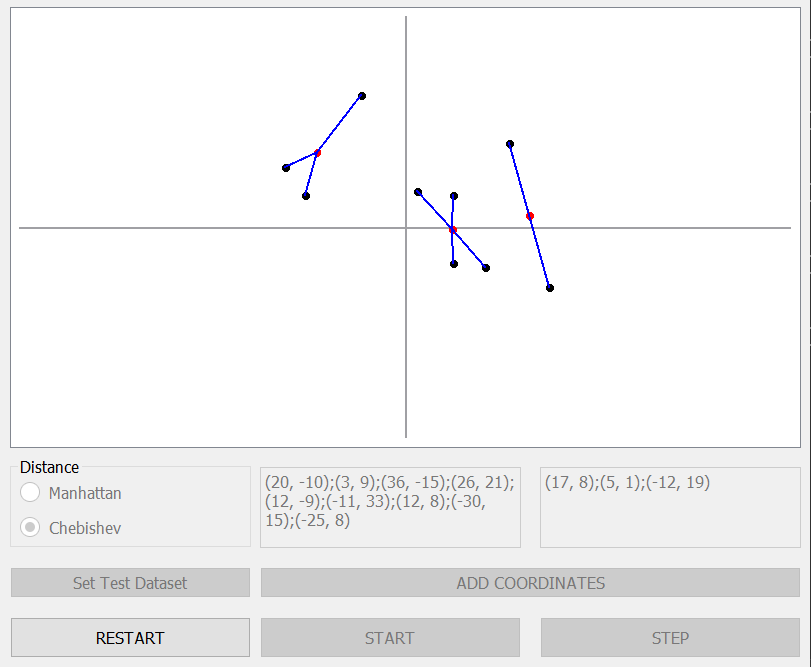
Рисунок 7 – Кластеризация на 2 шаге

Рисунок 8 – Кластеризация финальная

# Выводы

В ходе выполнения данной лабораторной работы было произведена кластеризация с помощью метода k-средних, используя Манхэттенское расстояние и расстояние Чебышева.

# Приложение А. Исходный код программы

import math

import sys

import itertools

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import QPushButton, QGraphicsView, QRadioButton

coord = "(20, -10);(3, 9);(36, -15);(26, 21);(12, -9);(-11, 33);(12, 8);(-30, 15);(-25, 8)"

Manhat = "(23, 7);(-4, 17);(5, 2)"

chebi = "(17, 8);(5, 1);(-12, 19)"

class qt(object):

def \_\_init\_\_(self, obj):

super().\_\_init\_\_()

self.Uiset(obj)

self.retranslateUi(obj)

self.graphicsView = QGraphicsView(obj)

self.graphicsView.setGeometry(QtCore.QRect(10, 10, 791, 441))

self.graphicsView.setObjectName("graphicsView")

self.scene = QtWidgets.QGraphicsScene()

self.graphicsView.setScene(self.scene)

pen = QtGui.QPen(QtCore.Qt.GlobalColor.gray)

for i in range(-1 \* self.graphicsView.height() // 2 + 10, self.graphicsView.height() // 2 - 10):

r1 = QtCore.QRectF(QtCore.QPointF(0, i), QtCore.QSizeF(1, 1))

self.scene.addRect(r1, pen)

for i in range(-1 \* self.graphicsView.width() // 2 + 10, self.graphicsView.width() // 2 - 10):

r2 = QtCore.QRectF(QtCore.QPointF(i, 0), QtCore.QSizeF(1, 1))

self.scene.addRect(r2, pen)

self.coordsContainer = []

self.centersContainer = []

self.clastersContainer = []

self.distance = None

def Uiset(self, Form):

Form.setObjectName("Form")

Form.resize(815, 678)

self.startPushButton = QPushButton(Form)

self.startPushButton.clicked.connect(self.click)

self.startPushButton.setGeometry(QtCore.QRect(260, 620, 261, 41))

self.startPushButton.setObjectName("startPushButton")

self.coordsTextBox = QtWidgets.QPlainTextEdit(Form)

self.coordsTextBox.setGeometry(QtCore.QRect(260, 470, 261, 81))

self.coordsTextBox.setObjectName("coordsTextBox")

self.CentersTextBox = QtWidgets.QPlainTextEdit(Form)

self.CentersTextBox.setGeometry(QtCore.QRect(540, 470, 261, 81))

self.CentersTextBox.setObjectName("CentersTextBox")

self.addCordsPushButton = QPushButton(Form)

self.addCordsPushButton.clicked.connect(self.pushButt)

self.addCordsPushButton.setGeometry(QtCore.QRect(260, 570, 541, 31))

self.addCordsPushButton.setObjectName("addCordsPushButton")

self.groupBox = QtWidgets.QGroupBox(Form)

self.groupBox.setGeometry(QtCore.QRect(10, 460, 241, 91))

self.groupBox.setObjectName("groupBox")

self.manhattanRadioButton = QRadioButton(self.groupBox)

self.manhattanRadioButton.toggled.connect(self.manhatt)

self.manhattanRadioButton.setGeometry(QtCore.QRect(10, 20, 221, 31))

self.manhattanRadioButton.setObjectName("manhattanRadioButton")

self.chebishevRadioButton = QRadioButton(self.groupBox)

self.chebishevRadioButton.toggled.connect(self.chebi)

self.chebishevRadioButton.setGeometry(QtCore.QRect(10, 50, 221, 41))

self.chebishevRadioButton.setObjectName("chebishevRadioButton")

self.stepPushButton = QPushButton(Form)

self.stepPushButton.clicked.connect(self.stepPushButton\_button\_clicked)

self.stepPushButton.setGeometry(QtCore.QRect(540, 620, 261, 41))

self.stepPushButton.setObjectName("stepPushButton")

self.restartPushButton = QPushButton(Form)

self.restartPushButton.clicked.connect(self.restar)

self.restartPushButton.setGeometry(QtCore.QRect(10, 620, 241, 41))

self.restartPushButton.setObjectName("restartPushButton")

self.testPushButton = QPushButton(Form)

self.testPushButton.clicked.connect(self.testPushButton\_button\_clicked)

self.testPushButton.setGeometry(QtCore.QRect(10, 570, 241, 31))

self.testPushButton.setObjectName("testPushButton")

self.retranslateUi(Form)

QtCore.QMetaObject.connectSlotsByName(Form)

def retranslateUi(self, Form):

\_translate = QtCore.QCoreApplication.translate

Form.setWindowTitle(\_translate("Form", "Form"))

self.startPushButton.setText(\_translate("Form", "START"))

self.addCordsPushButton.setText(\_translate("Form", "ADD COORDINATES"))

self.groupBox.setTitle(\_translate("Form", "Distance"))

self.manhattanRadioButton.setText(\_translate("Form", "Manhattan"))

self.chebishevRadioButton.setText(\_translate("Form", "Chebishev"))

self.stepPushButton.setText(\_translate("Form", "STEP"))

self.restartPushButton.setText(\_translate("Form", "RESTART"))

self.testPushButton.setText(\_translate("Form", "Set Test Dataset"))

def LineDr(self):

pen = QtGui.QPen(QtCore.Qt.GlobalColor.blue)

brush = QtGui.QBrush(QtCore.Qt.GlobalColor.blue)

pen.setWidth(2)

pen.setColor(QtCore.Qt.GlobalColor.blue)

for i in range(len(self.clastersContainer)):

for j in self.clastersContainer[i]:

self.scene.addLine(QtCore.QLineF(4 \* j[0], -4 \* j[1], 4 \* self.centersContainer[i][0],

-4 \* self.centersContainer[i][1]), pen)

def Pars(self, coords, centers):

coords\_l = None

centers\_l = None

try:

coords\_string\_array = coords.split(';')

centers\_string\_array = centers.split(';')

coords\_l = []

centers\_l = []

for i in coords\_string\_array:

l = [float(k) for k in i.strip('()').split(',')]

coords\_l.append(l)

for i in centers\_string\_array:

l = [float(k) for k in i.strip('()').split(',')]

centers\_l.append(l)

except:

self.CentersTextBox.clear()

self.coordsTextBox.clear()

msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Critical)

msg.setText("Format Error")

msg.setInformativeText('Follow the format!')

msg.setWindowTitle("Error")

msg.setStyleSheet("QLabel{font-size: 20px;}")

msg.exec\_()

pass

return coords\_l, centers\_l

def Coordin(self):

pen = QtGui.QPen(QtCore.Qt.GlobalColor.black)

brush = QtGui.QBrush(QtCore.Qt.GlobalColor.black)

side = 4

for i in self.coordsContainer:

self.scene.addEllipse(i[0] \* side - 3, -1 \* i[1] \* side - 3, 7, 7, pen, brush)

pen = QtGui.QPen(QtCore.Qt.GlobalColor.red)

brush = QtGui.QBrush(QtCore.Qt.GlobalColor.red)

for i in self.centersContainer:

self.scene.addEllipse(i[0] \* side - 3, -1 \* i[1] \* side - 3, 7, 7, pen, brush)

def pushButt(self):

coordinates = self.coordsTextBox.toPlainText()

centers = self.CentersTextBox.toPlainText()

if coordinates == '' or centers == '':

msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Critical)

msg.setText("Data Empty")

msg.setInformativeText('Please, enter coords')

msg.setWindowTitle("Error")

msg.setStyleSheet("QLabel{font-size: 20px;}")

msg.exec\_()

return

coordinates\_l, centers\_l = self.Pars(coordinates, centers)

if coordinates\_l is not None and centers\_l is not None:

co = self.coordsContainer.copy()

ce = self.centersContainer.copy()

co += coordinates\_l.copy()

ce += centers\_l.copy()

co.sort()

ce.sort()

co\_new = list(num for num, \_ in itertools.groupby(co))

ce\_new = list(num for num, \_ in itertools.groupby(ce))

self.centersContainer = ce\_new.copy()

self.coordsContainer = co\_new.copy()

print(self.centersContainer)

print(self.coordsContainer)

self.Coordin()

def click(self):

if self.coordsContainer == [] or self.centersContainer == []:

msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Critical)

msg.setText("Data Empty")

msg.setInformativeText('Please, enter coords')

msg.setWindowTitle("Error")

msg.setStyleSheet("QLabel{font-size: 20px;}")

msg.exec\_()

return

self.chebishevRadioButton.setEnabled(False)

self.manhattanRadioButton.setEnabled(False)

self.addCordsPushButton.setEnabled(False)

self.coordsTextBox.setEnabled(False)

self.CentersTextBox.setEnabled(False)

self.startPushButton.setEnabled(False)

self.testPushButton.setEnabled(False)

if self.distance == 'M':

for \_ in range(len(self.centersContainer)):

self.clastersContainer.append([])

for i in self.coordsContainer:

range\_l = []

for c in self.centersContainer:

range\_l.append(abs(i[0] - c[0]) + abs(i[1] - c[1]))

minindex = range\_l.index(min(range\_l))

self.clastersContainer[minindex].append(i)

self.LineDr()

elif self.distance == 'H':

for \_ in range(len(self.centersContainer)):

self.clastersContainer.append([])

for i in self.coordsContainer:

range\_l = []

for c in self.centersContainer:

range\_l.append(max(abs(i[0] - c[0]), abs(i[1] - c[1])))

minindex = range\_l.index(min(range\_l))

self.clastersContainer[minindex].append(i)

self.LineDr()

def stepPushButton\_button\_clicked(self):

if self.centersContainer is None or self.coordsContainer is None:

msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Critical)

msg.setText("Empty Error")

msg.setInformativeText('Not enough dots!')

msg.setWindowTitle("Error")

msg.setStyleSheet("QLabel{font-size: 20px;}")

msg.exec\_()

return

claster\_backup = self.clastersContainer.copy()

new\_centers = []

for i in self.clastersContainer:

new\_x, new\_y = 0, 0

for k in i:

new\_x += k[0]

new\_y += k[1]

new\_x /= len(i)

new\_y /= len(i)

new\_centers.append([new\_x, new\_y])

self.centersContainer = new\_centers.copy()

self.redr(False)

self.Coordin()

self.clastersContainer.clear()

for \_ in range(len(self.centersContainer)):

self.clastersContainer.append([])

for i in self.coordsContainer:

range\_l = []

for c in new\_centers:

range\_l.append(math.sqrt((i[0] - c[0]) \*\* 2 + (i[1] - c[1]) \*\* 2))

minindex = range\_l.index(min(range\_l))

self.clastersContainer[minindex].append(i)

self.LineDr()

new\_back\_clasters = self.clastersContainer.copy()

if claster\_backup == new\_back\_clasters:

self.stepPushButton.setEnabled(False)

def redr(self, full):

self.scene.clear()

pen = QtGui.QPen(QtCore.Qt.GlobalColor.gray)

for i in range(-1 \* self.graphicsView.height() // 2 + 10, self.graphicsView.height() // 2 - 10):

r1 = QtCore.QRectF(QtCore.QPointF(0, i), QtCore.QSizeF(1, 1))

self.scene.addRect(r1, pen)

for i in range(-1 \* self.graphicsView.width() // 2 + 10, self.graphicsView.width() // 2 - 10):

r2 = QtCore.QRectF(QtCore.QPointF(i, 0), QtCore.QSizeF(1, 1))

self.scene.addRect(r2, pen)

if not full:

pen2 = QtGui.QPen(QtCore.Qt.GlobalColor.black)

brush2 = QtGui.QBrush(QtCore.Qt.GlobalColor.black)

side = 4

for i in self.coordsContainer:

self.scene.addEllipse(i[0] \* side - 3, -1 \* i[1] \* side - 3, 7, 7, pen2, brush2)

def restar(self):

self.chebishevRadioButton.setEnabled(True)

self.manhattanRadioButton.setEnabled(True)

self.addCordsPushButton.setEnabled(True)

self.coordsTextBox.setEnabled(True)

self.CentersTextBox.setEnabled(True)

self.testPushButton.setEnabled(True)

self.startPushButton.setEnabled(True)

self.stepPushButton.setEnabled(True)

self.redr(True)

self.coordsContainer.clear()

self.centersContainer.clear()

self.clastersContainer.clear()

def testPushButton\_button\_clicked(self):

self.coordsTextBox.setPlainText(coord)

if self.distance == 'M':

self.CentersTextBox.setPlainText(Manhat)

elif self.distance == 'H':

self.CentersTextBox.setPlainText(chebi)

else:

self.coordsTextBox.clear()

msg = QtWidgets.QMessageBox()

msg.setIcon(QtWidgets.QMessageBox.Critical)

msg.setText("Distance not set")

msg.setInformativeText('Please, pick the distance')

msg.setWindowTitle("Error")

msg.setStyleSheet("QLabel{font-size: 20px;}")

msg.exec\_()

pass

def manhatt(self):

if self.manhattanRadioButton.isChecked():

self.distance = 'M'

def chebi(self):

if self.chebishevRadioButton.isChecked():

self.distance = 'H'

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

widget = QtWidgets.QWidget()

app2 = qt(widget)

widget.setWindowTitle("Neural\_network")

widget.setFixedWidth(810)

widget.setFixedHeight(670)

widget.show()

exit(app.exec\_())