Федеральное агентство связи

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

Московский Технический Университет

Связи и Информатики

Кафедра Математической Кибернетики и Информационных Технологий

Отчёт по учебной практике

«Машинное обучение»

Выполнил студент

Группы БФИ 1701

Агамагомедов М.П.

Научный руководитель

Зав. кафедры МКиИТ.

Городничев М.Г.

Москва 2019

Оглавление

[Цель 3](#_Toc19465807)

[Задачи 3](#_Toc19465808)

[Анализ предметной области 3](#_Toc19465809)

[Проектирование 5](#_Toc19465810)

[Разработка 5](#_Toc19465811)

[Вывод 13](#_Toc19465812)

[Список источников 14](#_Toc19465813)

# **Цель**

Ознакомиться с анализом данных, как с частью машинного обучения.

# **Задачи**

Изучить задачу кластеризации, а также рассмотреть и использовать следующие методы кластеризации: k-means, Agglomerative, DBSCAN.

# **Анализ предметной области**

Машинное обучение – методики анализа данных, позволяющие системе обучаться, в ходе обработки данных и решения определённых задач. Задача машинного обучения: дано конечное множество (объектов, ситуаций), по каждому из которых есть некоторые данные (описание). Совокупность всех имеющихся описаний представляет собой обучающую выборку. Требуется по этим частным данным выявить общие зависимости, закономерности, взаимосвязи, присущие выборке.

Существует два типа машинного обучения: обучение без учителя и обучение с учителем. Обучение с учителем – каждая ситуация представляет собой пару «объект, ответ». Нужно найти зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ. Обучение без учителя – нужно найти зависимость между объектами, в этом случае ответы не задаются.

Далее в данной работе разберём только обучение без учителя.

В рамках обучения без учителя рассмотрим задачу кластеризации. Задача кластеризации заключается в том, чтобы сгруппировать объекты в кластеры. Функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.

Рассмотрим несколько методов кластеризации:

* K-means (К-средних)

Это итеративный алгоритм кластеризации, основанный на минимизации суммарных квадратичных отклонений точек кластеров от средних координат (центра) этих кластеров. Первоначально выбирается желаемое количество кластеров. P Затем случайным образом из входных данных выбираются выбранное кол-во элементов в соответствии с кол-вом кластеров, в каждый из которых теперь включено по одной точке. Далее ищем ближайшего соседа к текущему центра. Добавляем точку к соответствующему кластеру и пересчитываем положение центра с учетом координат новых точек. Алгоритм заканчивает работу, когда центр перестаёт меняться.

* DBSCAN

Алгоритм группирует тесно расположенные точки, тем самым разделяя их на кластеры. Точки, которые расположены далеко от кластеров отмечаются как выбросы. Для этого алгоритма не нужно заранее знать кол-во кластеров, но необходимо указывать расстояние между точками и минимальное кол-во точек входящих в зону для достижения связи.

* Agglomerative

Алгоритм без фиксированного кол-ва кластеров. С самого начала все точки обозначаем за кластеры, затем наиболее похожие (близкие) кластеры объединяются в новый более крупный кластер. Это происходит до тех пор, пока не образуется один большой корневой кластер.

# **Проектирование**

Для рассмотрения алгоритмов кластеризации использовался язык программирования Python в среде разработки PyCharm версии 2019.1.3 (Community Edition). Реализация алгоритмов осуществлялась при помощи двух подключаемых библиотек: scikit-learn и matplotlib.

Библиотека scikit-learn содержит ряд современных алгоритмов машинного обучения, а также полную документацию по каждому алгоритму. Это очень популярный инструмент и самая известная питоновская библиотека для машинного обучения. Она широко используется в промышленности и науке, а в интернете имеется богатый выбор обучающих материалов и примеров программного кода.

Matplotlib – это основная библиотека для построения научных графиков в Python. Она включает функции для создания высококачественных визуализаций типа линейных диаграмм, гистограмм, диаграмм разброса и т.д.

# **Разработка**

Алгоритм k-means

Листинг реализации представлен ниже (Листинг 1). Результаты работы представления на рисунках 1-3.

Листинг 1 – k-means

*#k-means clustering***import** matplotlib.pyplot **as** plt  
  
  
**from** sklearn.cluster **import** KMeans  
**from** sklearn.datasets **import** make\_blobs  
  
*#Создаём двумерный набор данных из 150 точек*X, y = make\_blobs(n\_samples=150,  
 n\_features=2,  
 centers=3,  
 cluster\_std=0.5,  
 random\_state=0)  
*#Создание графика*fig = plt.figure(1)  
plt.scatter(X[:,0],  
 X[:,1],  
 c=**'white'**,edgecolors=**'black'**,  
 marker=**'o'**,  
 s=50)  
plt.grid()  
  
*#Кластеризация методом к-средних (первоначальные центроиды выбираются методом k-means++)*kmeans = KMeans(n\_clusters=3,init=**'k-means++'**,n\_init=10,max\_iter=300,tol=1e-04,random\_state=0)  
y\_kmeans = kmeans.fit\_predict(X)  
  
fig2 = plt.figure(2)  
plt.scatter(X[y\_kmeans==0,0],  
 X[y\_kmeans==0,1],  
 s=50,  
 c=**'green'**,  
 edgecolors=**'black'**,  
 marker=**'s'**)  
plt.scatter(X[y\_kmeans==1,0],  
 X[y\_kmeans==1,1],  
 s=50,  
 c=**'orange'**,  
 edgecolors=**'black'**,  
 marker=**'o'**)  
plt.scatter(X[y\_kmeans==2,0],  
 X[y\_kmeans==2,1],  
 s=50,  
 c=**'blue'**,  
 edgecolors=**'black'**,  
 marker=**'v'**)  
plt.scatter(kmeans.cluster\_centers\_[:,0],  
 kmeans.cluster\_centers\_[:,1],  
 s=250,  
 marker=**'\*'**,  
 c=**'red'**,  
 edgecolors=**'black'**)  
  
plt.legend([**"кластер 1"**, **"кластер 2"**, **"кластер 3"**,**"центроиды"**])  
plt.grid()  
*#plt.show()*print(**'Искажение : % .2f'** % kmeans.inertia\_)  
  
*#Используем метод локтя для нахождения оптимального кол-ва кластеров (на основе искажения)*fig3 = plt.figure()  
distortion = []  
**for** i **in** range(1,11):  
 kmeans = KMeans(n\_clusters=i,init=**'k-means++'**,n\_init=10,max\_iter=300,random\_state=0)  
 kmeans.fit(X)  
 distortion.append(kmeans.inertia\_)  
plt.plot(range(1,11),distortion,marker=**'o'**)  
plt.xlabel(**'Число кластеров'**)  
plt.ylabel(**'Искажение'**)  
plt.show()

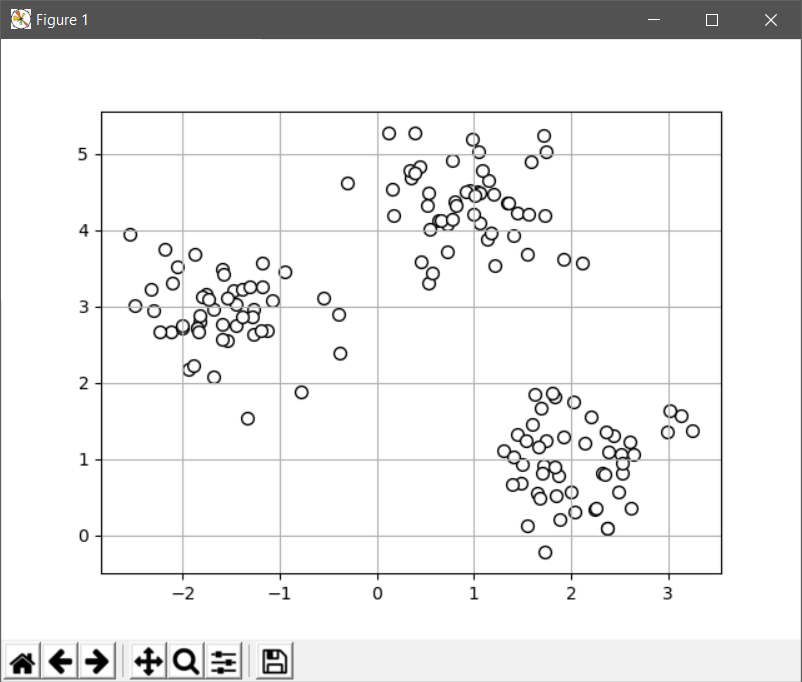


Рисунок 1 – k-means (отрисованные данные)

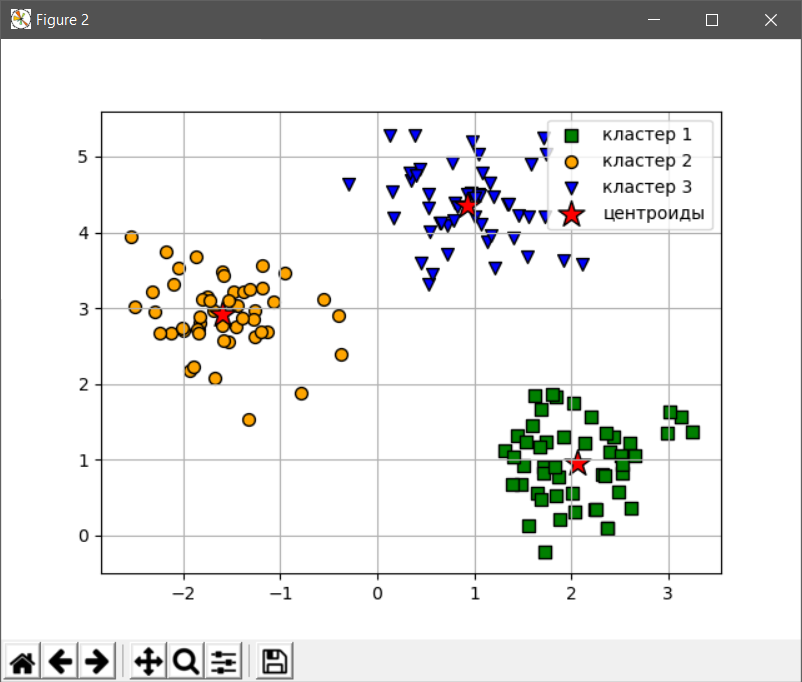


Рисунок 2 – k-means (набор данных после кластеризации)

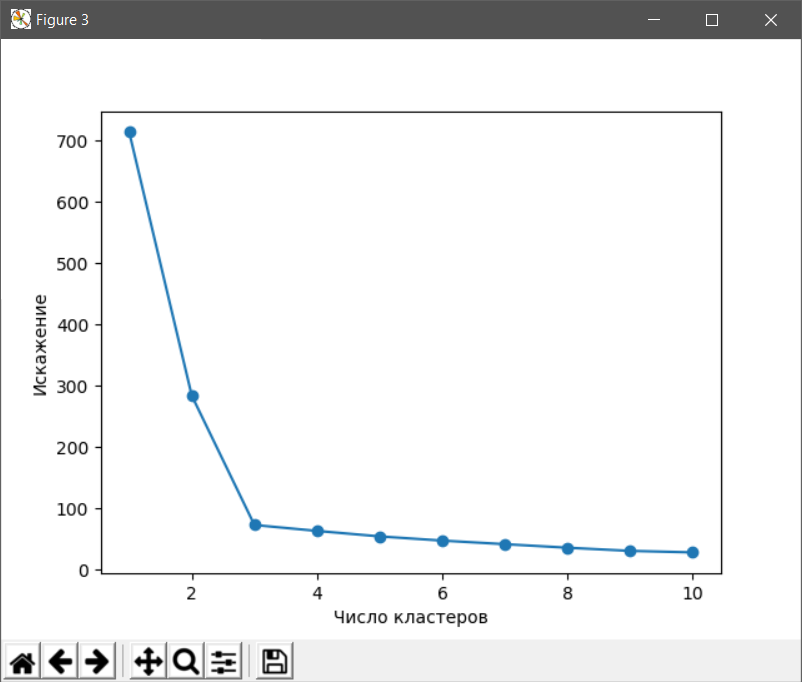


Рисунок 3 – k-means (график для определения оптимального кол-ва кластеров)

Алгоритм Agglomerative

Листинг реализации представлен ниже (Листинг 2). Результаты работы представления на рисунках 4-5.

Листинг 2 – Agglomerative

*#Agglomerative кластеризация***import** matplotlib.pyplot **as** plt  
  
  
**from** sklearn.cluster **import** AgglomerativeClustering  
**from** sklearn.datasets **import** make\_blobs  
  
*#Генерация двумерного набора данных*X, y = make\_blobs(random\_state=2)  
  
*#Кластеризация Agglomerative методом (критерий остановки 3 кластера)*agl = AgglomerativeClustering(n\_clusters=3)  
y\_ag = agl.fit\_predict(X)  
  
*#Отрисовка*plt.scatter(X[y\_ag==0,0],  
 X[y\_ag==0,1],  
 c=**'lightblue'**,  
 marker=**'o'**,  
 s=40,  
 edgecolors=**'black'**,  
 label=**'кластер 1'**)  
plt.scatter(X[y\_ag==1,0],  
 X[y\_ag==1,1],  
 c=**'red'**,  
 marker=**'s'**,  
 s=40,  
 edgecolors=**'black'**,  
 label=**'кластер 2'**)  
plt.scatter(X[y\_ag==2,0],  
 X[y\_ag==2,1],  
 c=**'green'**,  
 marker=**'v'**,  
 s=40,  
 edgecolors=**'black'**,  
 label=**'кластер 3'**)  
plt.grid()  
plt.legend([**"кластер 1"**, **"кластер 2"**, **"кластер 3"**])  
*#plt.show()***from** scipy.cluster.hierarchy **import** dendrogram, ward  
fig = plt.figure()  
*# применяем кластеризацию ward к массиву данных X  
# функция SciPy ward возвращает массив с расстояниями вычисленными в ходе выполнения агломеративной кластеризации*linkage\_array = ward(X)  
*# строим дендрограмму для массива связей, содержащего расстояния между кластерами*dendrogram(linkage\_array)  
  
plt.show()

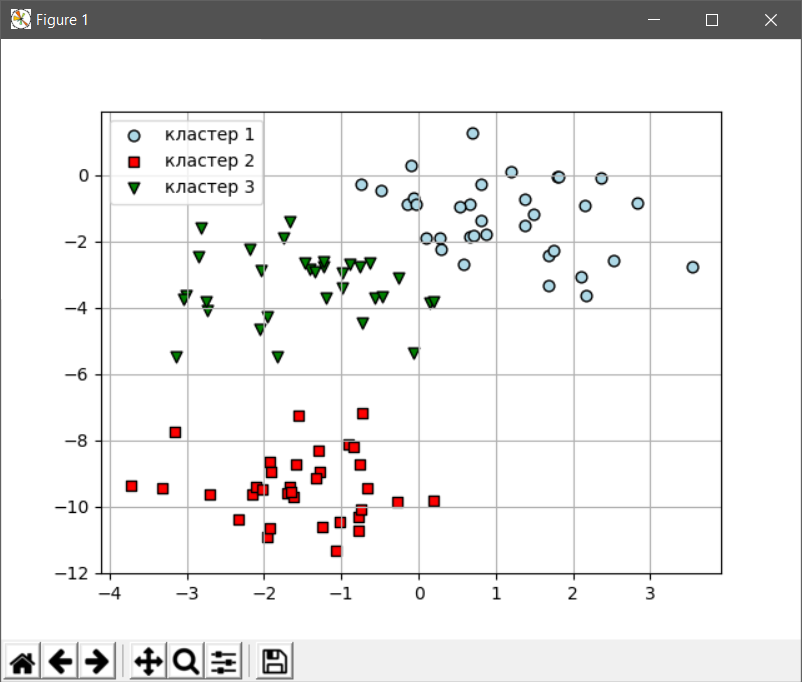


Рисунок 4 – Agglomerative (набор данных после кластеризации)

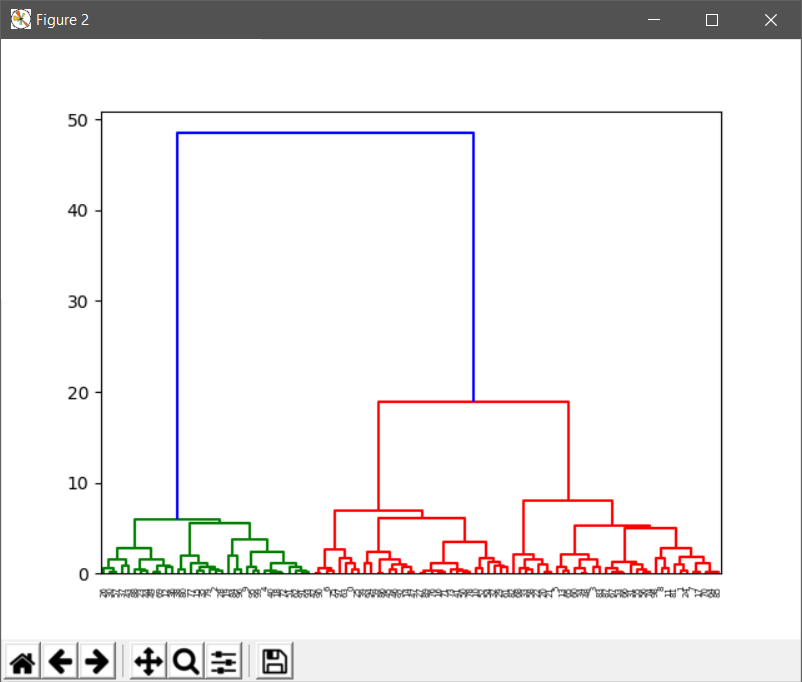


Рисунок 5 – Agglomerative (дендограмма визуализирующая работу алгоритма)

Алгоритм DBSCAN

Листинг реализации представлен ниже (Листинг 3). Результаты работы представления на рисунке 6.

Листинг 3 – DBSCAN

*#DBSCAN кластеризация***import** matplotlib.pyplot **as** plt  
  
  
**from** sklearn.cluster **import** DBSCAN  
**from** sklearn.datasets **import** make\_moons  
*#Генерация двумерного набора данных (в виде двух лун)*X, y = make\_moons(n\_samples=200,  
 noise=0.05,  
 random\_state=0)  
*#Кластеризация DBSCAN*db = DBSCAN(eps=0.2,  
 min\_samples=5)  
y\_db = db.fit\_predict(X)  
*#Отрисовка*plt.scatter(X[y\_db==0,0],  
 X[y\_db==0,1],  
 c=**'lightblue'**,  
 marker=**'o'**,  
 s=40,  
 edgecolors=**'black'**,  
 label=**'кластер 1'**)  
plt.scatter(X[y\_db==1,0],  
 X[y\_db==1,1],  
 c=**'red'**,  
 marker=**'s'**,  
 s=40,  
 edgecolors=**'black'**,  
 label=**'кластер 2'**)  
plt.legend()  
plt.show()

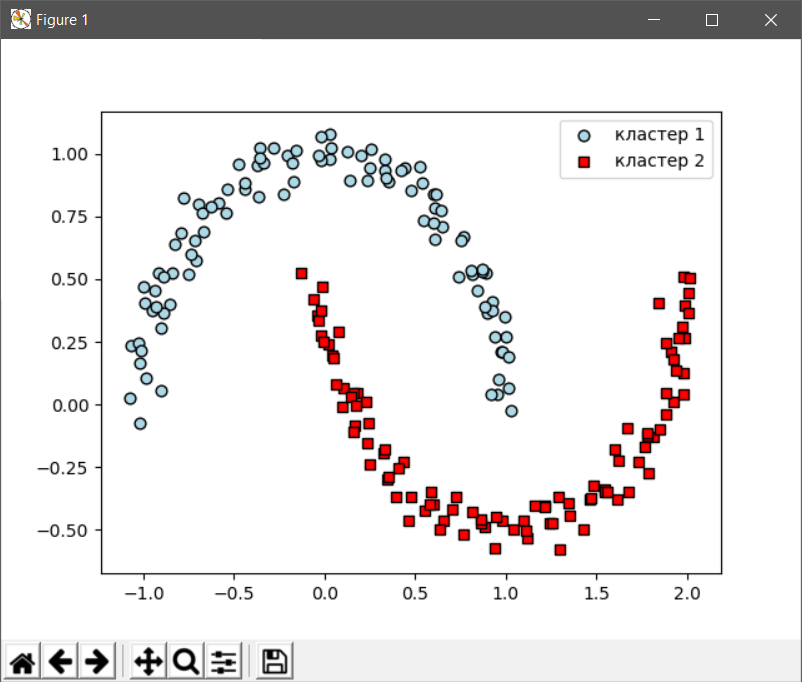


Рисунок 6 – DBSCAN (набор данных после кластеризации)

# **Вывод**

В ходе учебной практики были кратко изучены некоторые задачи машинного обучения, в основном это касается обучения без учителя, а именно задача кластеризации. Были рассмотрены три разных алгоритма кластеризации, каждый из которых способен помочь обнаружить структуру и/или информацию о данных. Каждый алгоритм обладает своими плюсами и минусами, что, безусловно, необходимо учитывать при выборе алгоритмов для работы с большими данными.

# **Список источников**

1. Рашка С. «Python и машинное обучуние»/пер. с англ. А.В. Логунова. – М.: ДМК Пресс, 2017. – 418 с.: ил.
2. Мюлер А., Гвидо С. «Введение в машинное обучение с помощью Python» – 2017.