МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

кафедра «Вычислительная техника».

**Лабораторная работа №4 (вариант 5)**

по дисциплине: «Системы искусственного интеллекта»

Выполнил:

студент 4 курса, гр. ИВТВМбд-41

Захарычев Никита Алексеевич.

Проверил:

кандидат технических наук

Святов Кирилл Валерьевич.

г. Ульяновск, 2017

**Задание**

1. Произвести масштабирование признаков (scaling).
2. С использованием библиотеки scikit-learn написать программу с использованием алгоритмов кластеризации данных, позволяющую разделить исходную выборку на классы.
3. Провести эксперименты и определить наилучший алгоритм кластеризации, параметры алгоритма. Необходимо использовать не менее 3-х алгоритмов. Данные экспериментов необходимо представить в отчете (графики, ход проведения эксперимента, выводы).

**Вариант 5**

1. Набор данных: Synthetic Control Chart Time Series.

**Ход работы**

Набор данных содержит 600 примеров контрольных диаграмм, синтетически генерируемых процессом в Alcock and Manolopoulos (1999). Выборку исходных данных требуется разделить на классы контрольных диаграмм.

*Подключение библиотек*

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

from sklearn.model\_selection import train\_test\_split

from sklearn import preprocessing

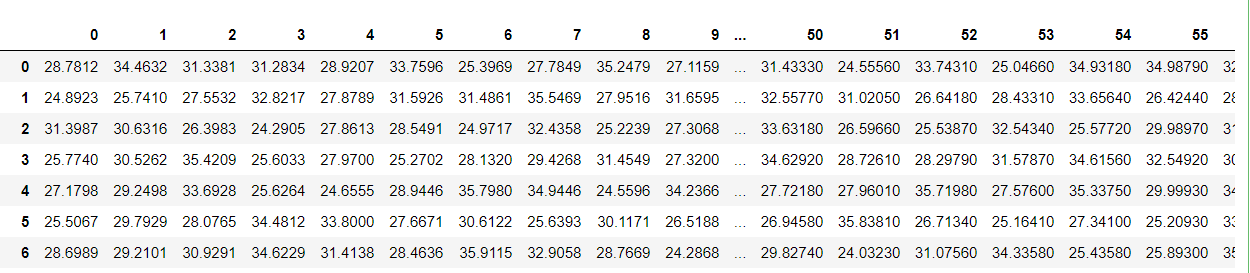
from sklearn.cluster import MiniBatchKMeans, KMeans

*Подготовка данных*

Считывание данных из файла для обучения:

data = pd.read\_csv('synthetic\_control.csv', sep='\s+', header = None)

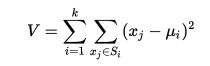
Фрагмент данных для обучения:



**Построение модели K-Means**

Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров.



Перед началом использования модели необходимо отмасштабировать признаки. Масштабирование преобразует функции, масштабируя каждую функцию до заданного диапазона. Эта оценка масштабирует и переводит каждую функцию по отдельности так, чтобы она находилась в заданном диапазоне на обучающем наборе, то есть между нулем и одним.

x\_train = preprocessing.minmax\_scale(x\_train)

# Compute clustering with Means

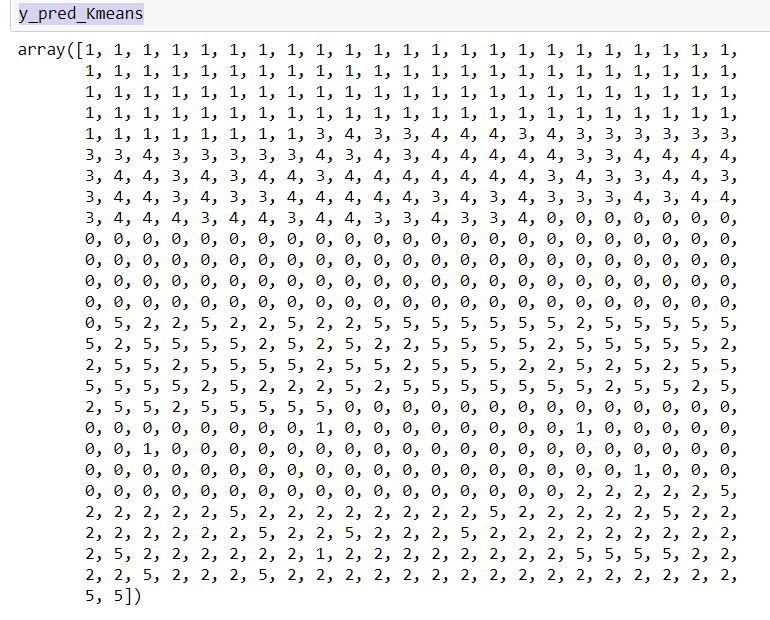
k\_means = KMeans(init='k-means++', n\_clusters=6, n\_init=10)

k\_means.fit(x\_train)

y\_pred\_Kmeans = k\_means.predict(x\_train)

y\_pred\_Kmeans

Получаем результат:



**Построение модели Mini Batch K-Means**

MiniBatchKMeans - это вариант алгоритма K-Means, который использует мини-партии для сокращения времени вычисления, при этом все еще пытается оптимизировать одну и ту же целевую функцию.

Мини-партии - это подмножества входных данных, случайным образом отбираемые на каждой итерации обучения. Эти мини-партии значительно сокращают объем вычислений, необходимых для сближения с локальным решением. В отличие от других алгоритмов, которые уменьшают время сходимости k-средних, мини-пакетное k-средство дает результаты, которые обычно немного хуже стандартного алгоритма.

# Compute clustering with MiniBatchKMeans

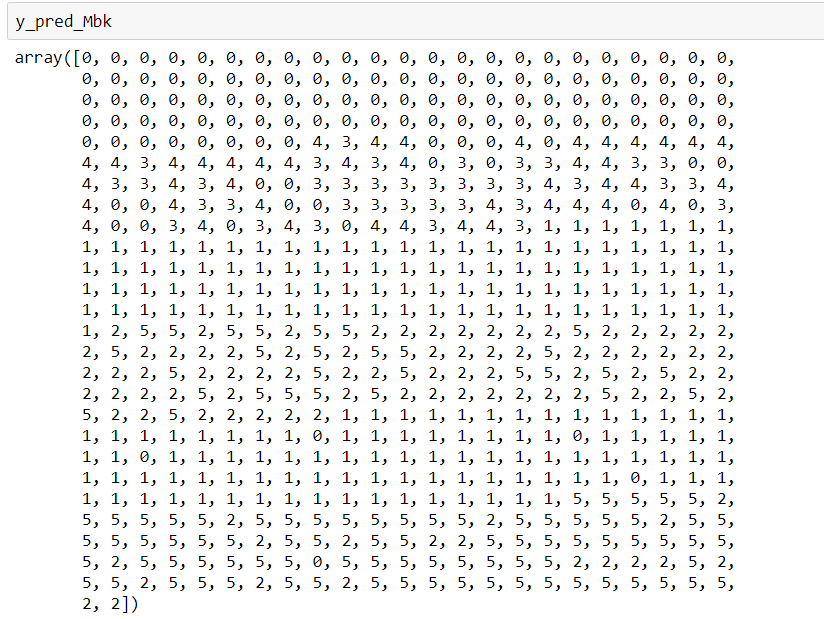
batch\_size = 45

mbk = MiniBatchKMeans(init='k-means++', n\_clusters=6, batch\_size=batch\_size, n\_init=10, max\_no\_improvement=10, verbose=0)

mbk.fit(x\_train)

y\_pred\_Mbk = mbk.predict(x\_train)

Получим результат:



Проанализировав полученные результаты, можно сделать вывод что кластеризация прошла корректно, так как элементы итогового принадлежат необходимым классам, допуская минимальную погрешность. Графическое представление результата невозможно реализовать, так как набор «иксов» более 50.

**Вывод**

Выполнив данную лабораторную работу, были улучшены навыки в программировании на языке Python. Были изучены библиотеки Python для построения K-Means и Mini Batch K-Means моделей, также были улучшены навыки в построении моделей для кластеризации данных.

**Исходный код**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

from sklearn.model\_selection import train\_test\_split

from sklearn import preprocessing

from sklearn.cluster import MiniBatchKMeans, KMeans

data = pd.read\_csv('synthetic\_control.csv', sep='\s+', header = None)

data

x\_train = data

x\_train

x\_train = preprocessing.minmax\_scale(x\_train)

# Compute clustering with Means

k\_means = KMeans(init='k-means++', n\_clusters=6, n\_init=10)

k\_means.fit(x\_train)

# Compute clustering with MiniBatchKMeans

batch\_size = 45

mbk = MiniBatchKMeans(init='k-means++', n\_clusters=6, batch\_size=batch\_size,

n\_init=10, max\_no\_improvement=10, verbose=0)

mbk.fit(x\_train)

y\_pred\_Kmeans = k\_means.predict(x\_train)

y\_pred\_Mbk = mbk.predict(x\_train)

y\_pred\_Kmeans

y\_pred\_Mbk