# Московский авиационный институт (Национальный исследовательский университет)

## Факультет: «Информационные технологии и прикладная математика» Кафедра: 806 «Вычислительная математика и программирование» Дисциплина: «Искусственный интеллект»

**Лабораторная работа № 2**

Тема: Алгоритмы машинного обучения

## Студент: Бурлаков Д.А. Группа: М80-406Б

Преподаватель: Ахмед Самир Халид

**Москва, 2019**

### Постановка задачи

Требуется реализовать класс на выбранном языке программирования, который реализует один из алгоритмов машинного обучения. Обязательным является наличия в классе двух методов fit, predict. Необходимо проверить работу вашего алгоритма на ваших данных (на таблице и на текстовых данных), произведя необходимую подготовку данных. Также необходимо реализовать алгоритм полиномиальной регрессии, для предсказания значений в таблице. Сравнить результаты с стандартной реализацией sklearn, определить в чем сходство и различие ваших алгоритмов. Замерить время работы алгоритмов.

Вариант: 5 % 6 + 1 = 6 (Метод К-средних)

### Метод К-средних

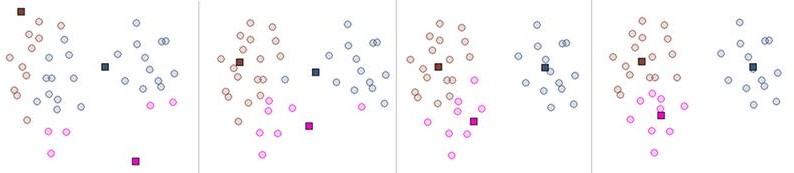
Метод k-средних ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) k-means) - наиболее популярный метод [кластеризации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F). лгоритму широко отдается предпочтение из-за его простоты реализации, большой скорости (а это очень важно при работе с видео).

Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров. В простонародье говоря, это итеративный алгоритм, который делит данное множество пикселей на k кластеров точки, которых являются максимально приближенными к их центрам, а сама кластеризация происходит за счет смещения этих же центров. Такой себе принцип разделяй и властвуй. Также следует оговорить то, что метод к-средних очень чувствительный к шуму, который может существенно исказить результаты кластеризации.Так что в идеале, перед кластеризацией, нужно прогнать кадры через фильтры предназначиные для его уменьшения. Вот собственно сам принцип простейшей кластеризации методом к-средних:

1. Надо выбрать из множества k пикселей те пиксели, которые будут центроидами соответствующих k кластеров.

Выборка начальных центроидов может быть как рандомной так и по определенному алгоритму.

1. Входим в цикл, который продолжается до тех пор, пока центроиды кластеров не перестанут изменять свое положение.
2. Обходим каждый пиксель и смотрим, к какому центроиду какого кластера он является близлежащим.
3. Нашли близлежащий центроид? Привязываем пиксель к кластеру этого центроида.
4. Перебрали все пиксели? Теперь нужно высчитать новые координаты центроидов k кластеров.
5. Теперь проверяем координаты новых центроидов. Если они соответственно равны предыдущим центроидам — выходим из цикла, если нет возвращаемся к пункту 3.



Вот картинка, которая приблизительно демонстрируют работу алгоритма: Источник: <https://habr.com/ru/post/165087/>

### Листинг программы

import numpy as np import pandas as pd import pylab as plt

import sklearn.cluster as km

class MyKMeans:

""" Класс реализующий алгоритм К-Средних """ def init (self, n\_clusters=2):

self.n\_clusters = n\_clusters self.labels = None self.cluster\_centers = None self.tol = 0.0001

def fit(self, X):

# Случайные центры

self.cluster\_centers = X[np.random.choice(range(X.shape[0]), self.n\_clusters)].copy()

samples = X.shape[0]

self.labels = np.zeros(shape=(samples), dtype=np.uint8) min\_dist = np.zeros(shape=(samples), dtype=np.float64)

while True:

# Перераспределение точек по кластерам for i in range(samples):

min\_dist[i] = np.linalg.norm(X[i] - self.cluster\_centers[0]) self.labels[i] = 0

for clust in range(1, self.n\_clusters): for i in range(samples):

dist = np.linalg.norm(X[i] - self.cluster\_centers[clust]) if dist < min\_dist[i]:

min\_dist[i], self.labels[i] = dist, clust

# Пересчет центров

new\_centers = np.array([X[self.labels == i].sum(axis=0) / X[self.labels == i].shape[0] for i in range(self.n\_clusters)])

if (np.abs(new\_centers - self.cluster\_centers) < self.tol).all(): break

# print(min\_dist, self.labels, self.cluster\_centers, new\_centers, sep='\n') # print(" \n")

self.cluster\_centers = new\_centers.copy() return self

def predict(self, X): samples = X.shape[0]

min\_dist = np.zeros(shape=(samples), dtype=np.float64) labels = np.zeros(shape=(samples), dtype=np.uint8)

for i in range(samples):

min\_dist[i] = np.linalg.norm(X[i] - self.cluster\_centers[0]) labels[i] = 0

for clust in range(1, self.n\_clusters): for i in range(samples):

dist = np.linalg.norm(X[i] - self.cluster\_centers[clust]) if dist < min\_dist[i]:

min\_dist[i], labels[i] = dist, clust return labels

if name == ' main ':

data = pd.read\_csv('MSFT.csv', index\_col=0) # Нормировка Volume

data['Volume'] = (data['Volume'] - data['Volume'].mean()) / data['Volume'].std() print("Первые 10 записей MSFT.csv: ")

print(data[:10])

sklearn\_kmeans = km.KMeans(3).fit(data.values) print("Центры, которые вычислила библиотека sklearn:") print(sklearn\_kmeans.cluster\_centers\_)

my\_kmeans = MyKMeans(3).fit(data.values) print("Центры, которые вычислила MyKMeans:") print(my\_kmeans.cluster\_centers)

plt.grid() plt.xlabel('High') plt.ylabel('Low')

plt.scatter(data['High'],data['Low'], c=my\_kmeans.labels) plt.show()

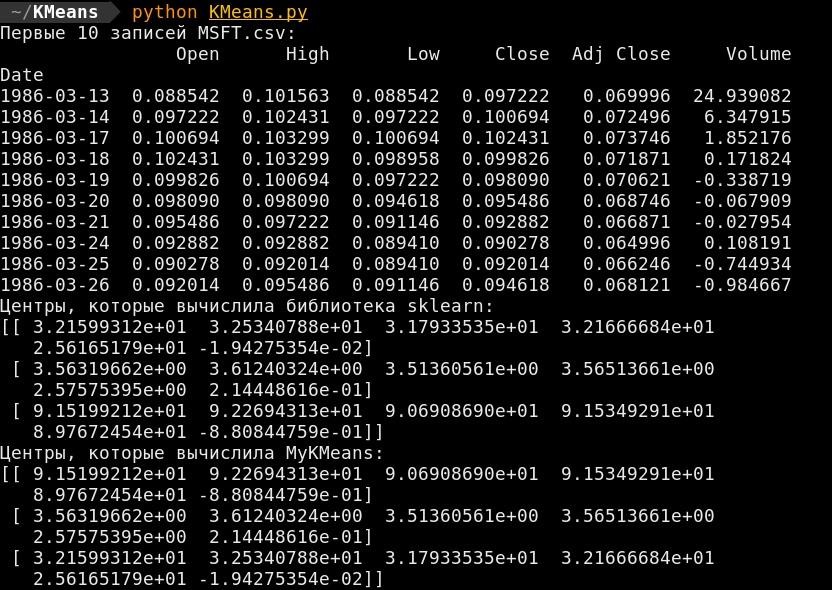
print("Случайные 20 записей для пресказания: ")

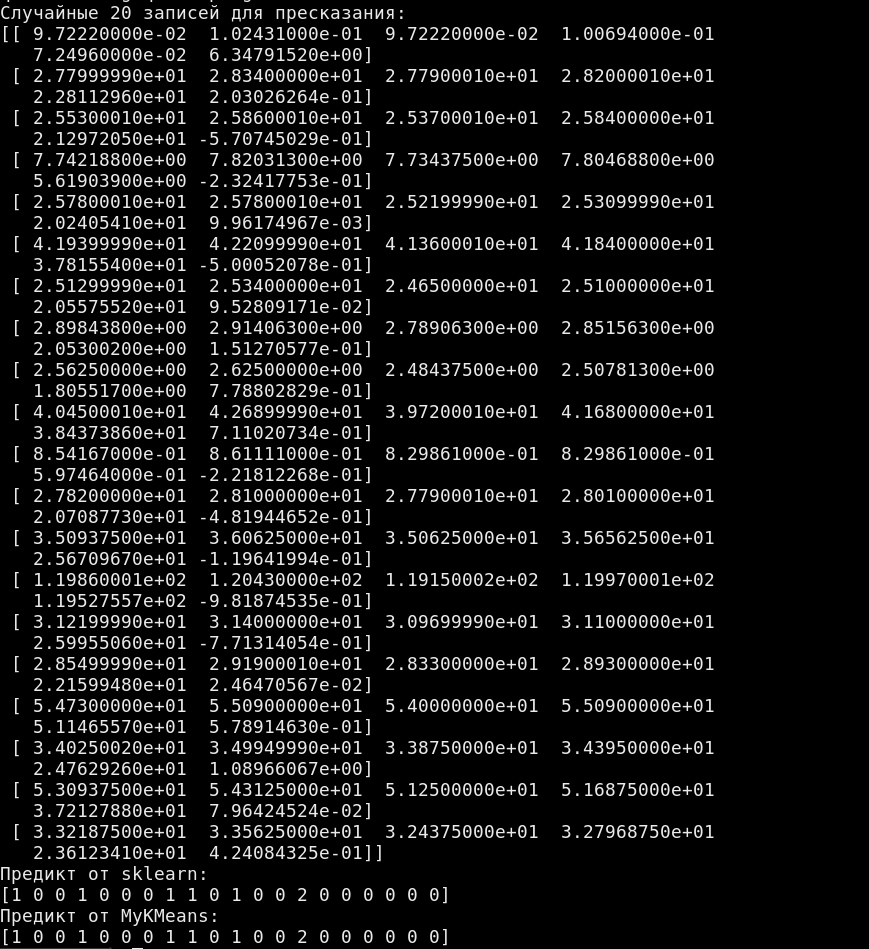
X = data.sample(20).values print(X)

print("Предикт от sklearn: ") print(sklearn\_kmeans.predict(X))

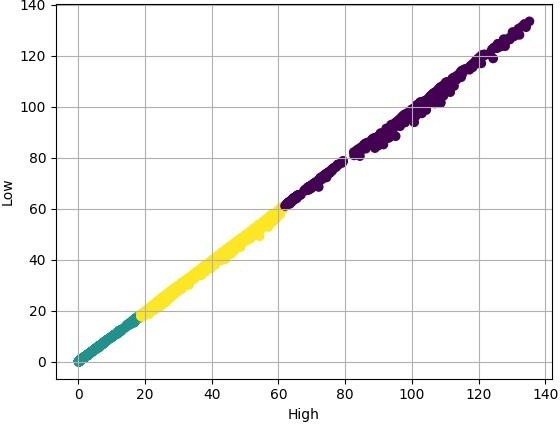
print("Предикт от MyKMeans: ") print(my\_kmeans.predict(X))

### Результат работы программы





Результат кластеризации



### Вывод

Благодаря проделанной работе, я познакомился с алгоритмом k-means и реализовал его. Была проверена работа моего алгоритма на данных.

Результаты работы обоих алгоритмов (собственной реализации и из библиотеки sklearn) совпадают с точностью до нумерации кластеров.