НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Анализ данных»

Домашнее задание на тему:
«Лабораторная работа №13»

Выполнил: Осипов Лев,
студент группы 301ПИ (1).

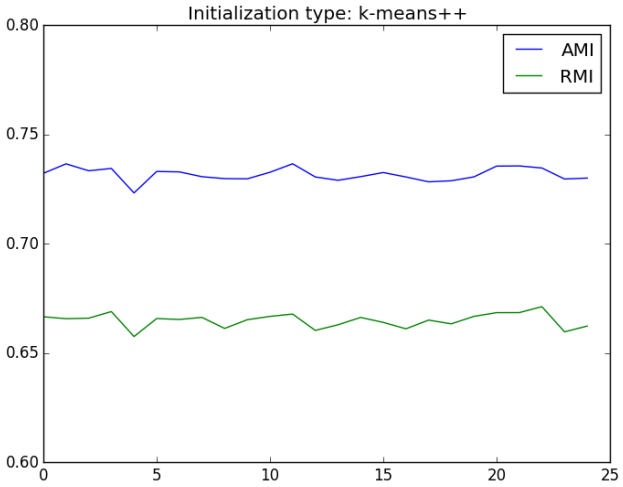
Москва, 2015 г.

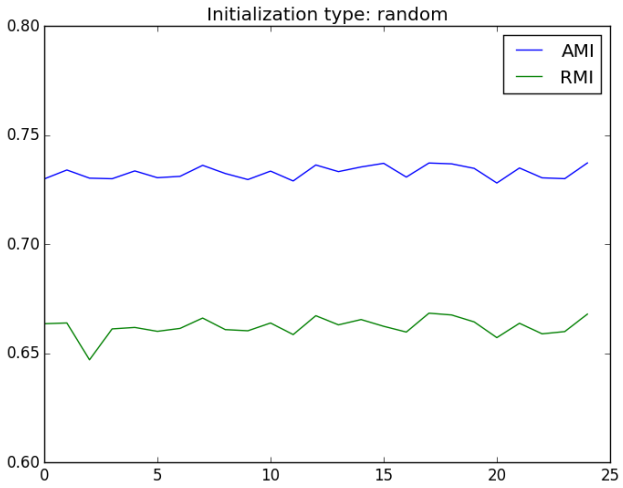
СОДЕРЖАНИЕ

ПРАКТИЧЕСКАЯ ЧАСТЬ

ЗАДАНИЕ 1

Для начала были посчитаны динамики двух метрик: Adjusted Mutual Information (AMI) и Adjusted Rand Index (RMI), при двух разных типах начального приближения (k-means++ и random), на 25 экспериментах:

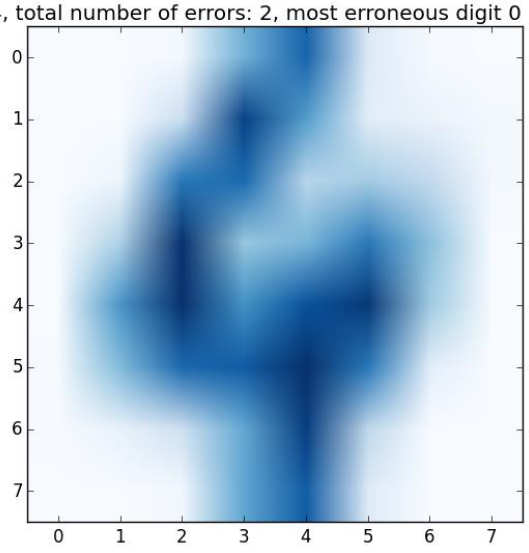




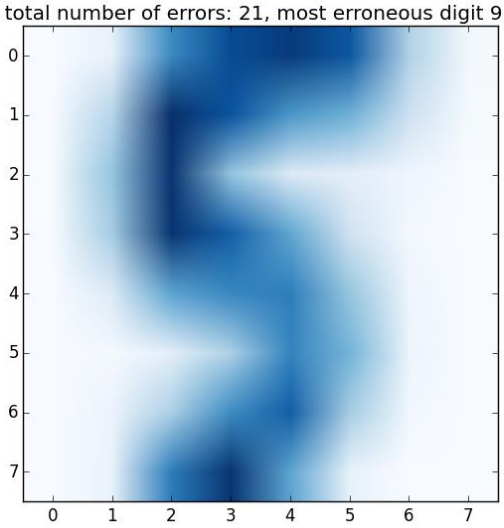
Как видно по графикам динамики, в целом метрики между собой согласуются неплохо.

Далее, была проведена очередная кластеризация. При этом было определено соответствие меток и кластеров, было посчитано общее количество ошибочных объектов в каждом кластере, также было посчитано, какая метка в каждом кластере наиболее ошибочна. Данные были визуализированы:

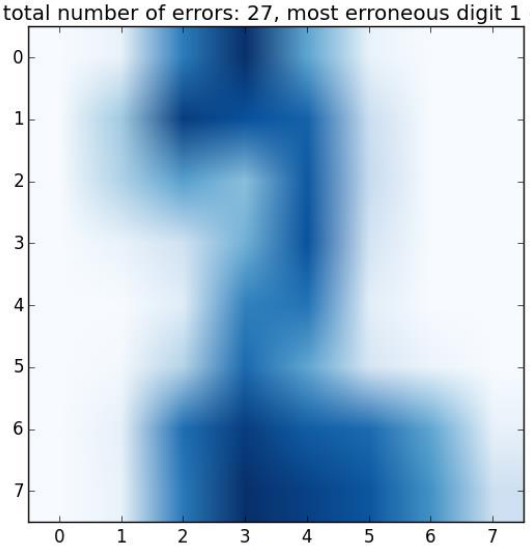
Digit: 4, total number of errors: 2, most erroneous digit 0 (1 errors)



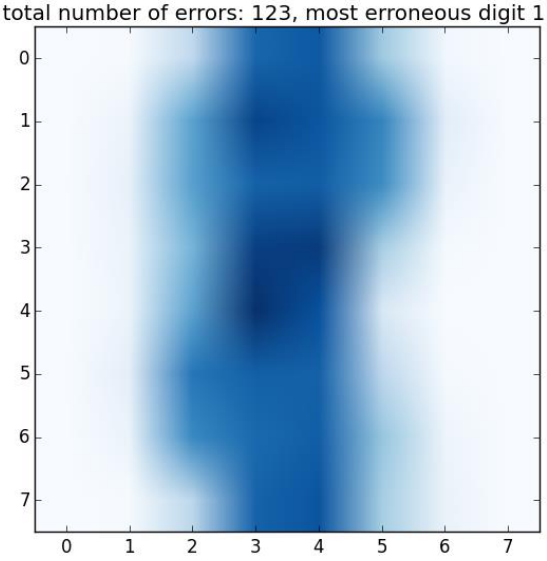
Digit: 5, total number of errors: 21, most erroneous digit 9 (7 errors)



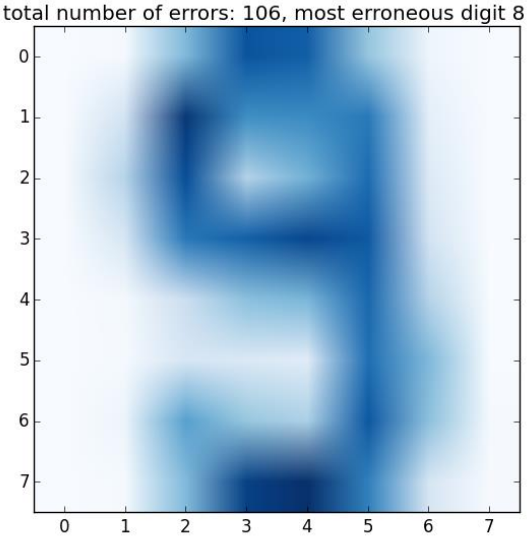
Digit: 2, total number of errors: 27, most erroneous digit 1 (24 errors)



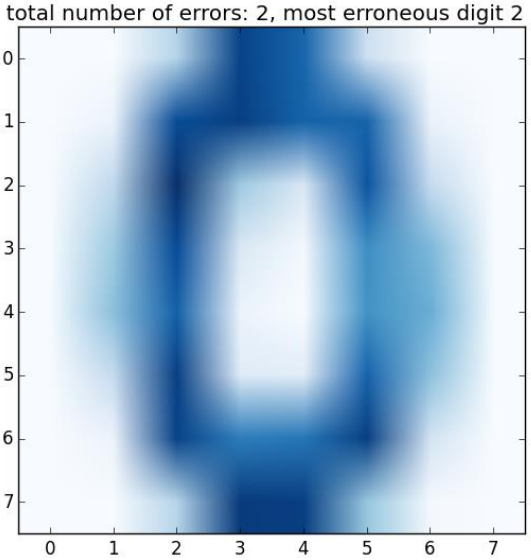
Digit: 8, total number of errors: 123, most erroneous digit 1 (99 errors)



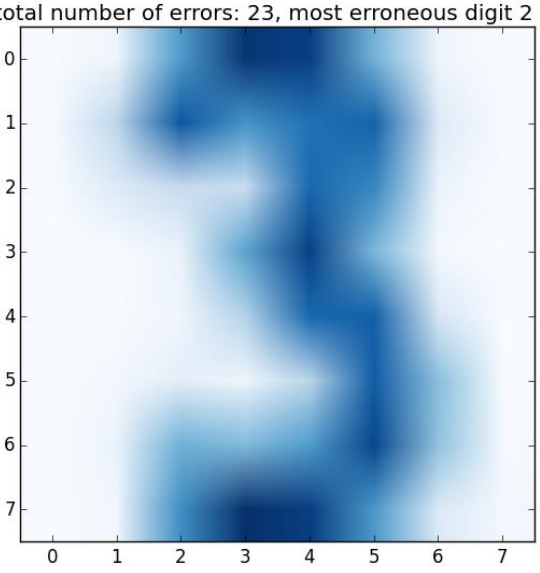
Digit: 9, total number of errors: 106, most erroneous digit 8 (50 errors)



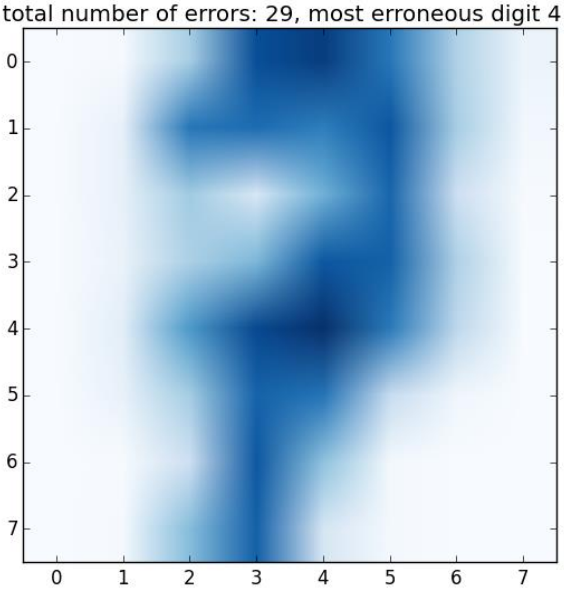
Digit: 0, total number of errors: 2, most erroneous digit 2 (1 errors)



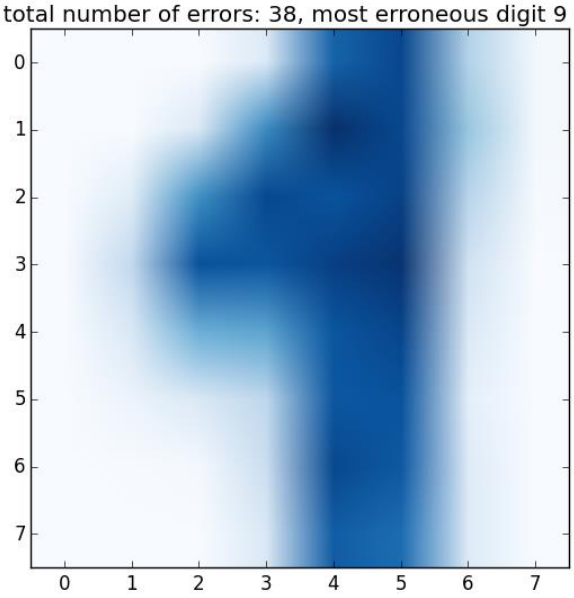
Digit: 3, total number of errors: 23, most erroneous digit 2 (13 errors)

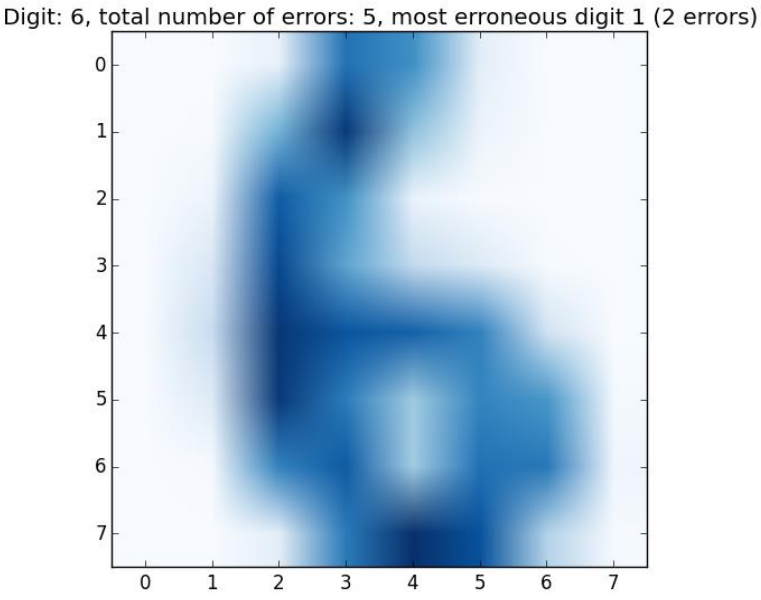


Digit: 7, total number of errors: 29, most erroneous digit 4 (9 errors)

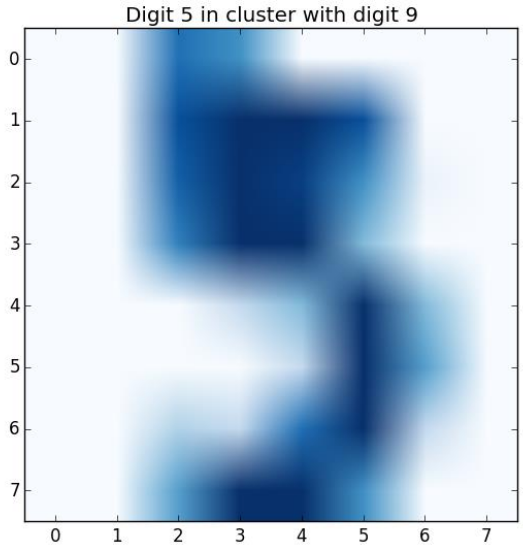


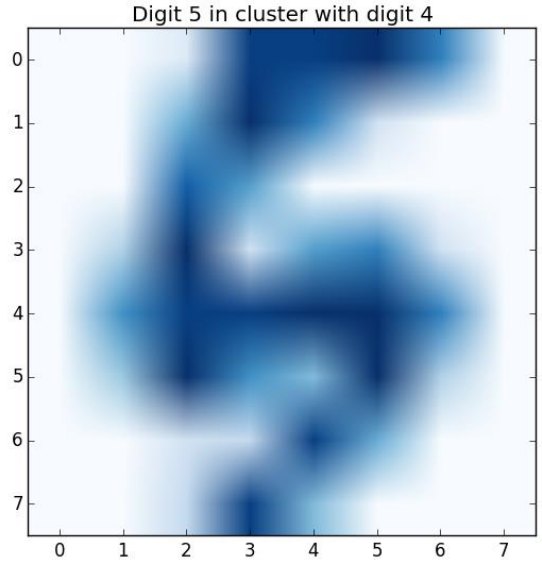
Digit: 1, total number of errors: 38, most erroneous digit 9 (20 errors)

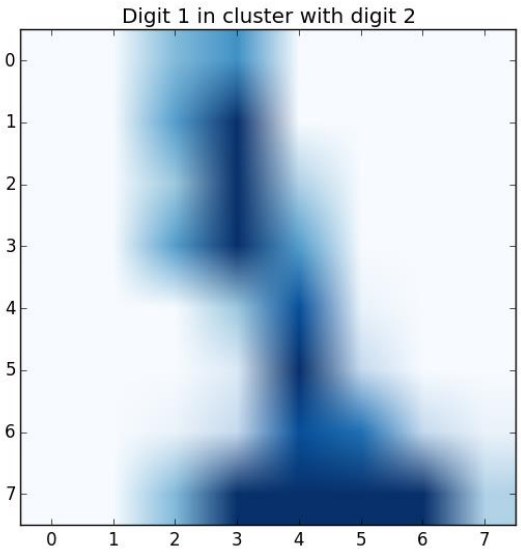




Также были высчитаны и визуализированы примеры ошибок:





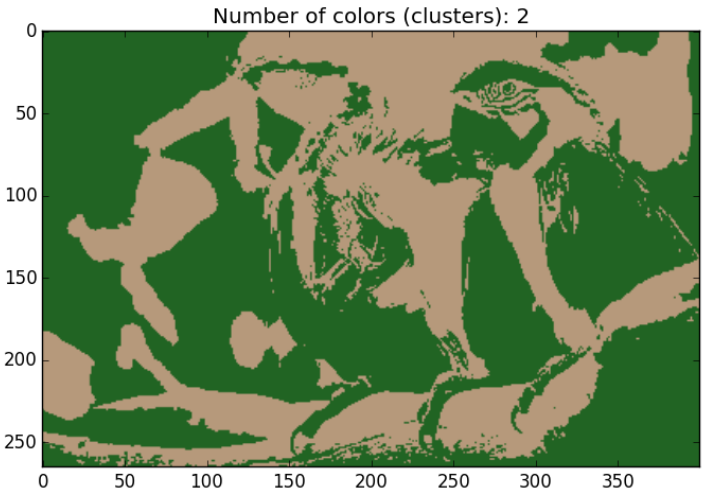


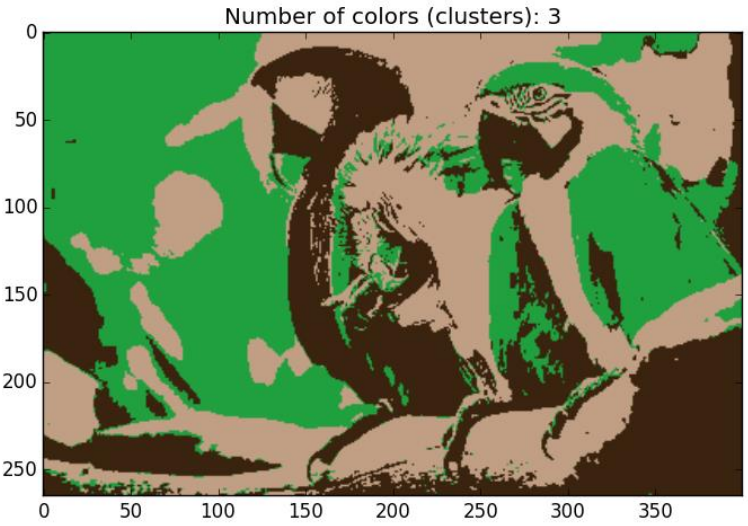
В целом можно сказать, что на основе большинства результатов кластеры выделились верно и охватили все метки цифр. Но заметны и крупные недочеты, например, с восьмеркой. Видно, что ввиду ее строения для нее построился достаточно нечеткий центр кластера, из-за которого возникло множество ошибок, особенно с единицей, которая зачастую была отнесена к этому кластеру.

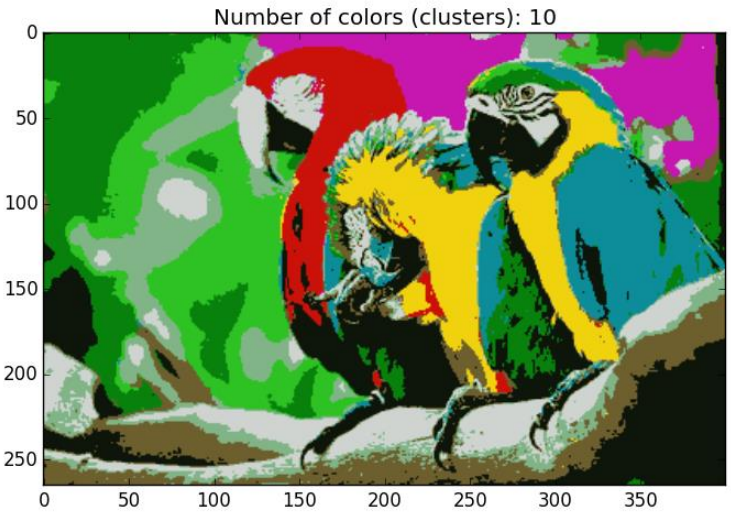
Из-за «рукописных» цифр зачастую возникали весьма закономерные ошибки, как, например, на картинке с примером ошибки, где пятерка была ошибочно воспринята за четверку. Здесь уже нельзя винить алгоритм, так как на вид мог бы ошибиться даже человек.

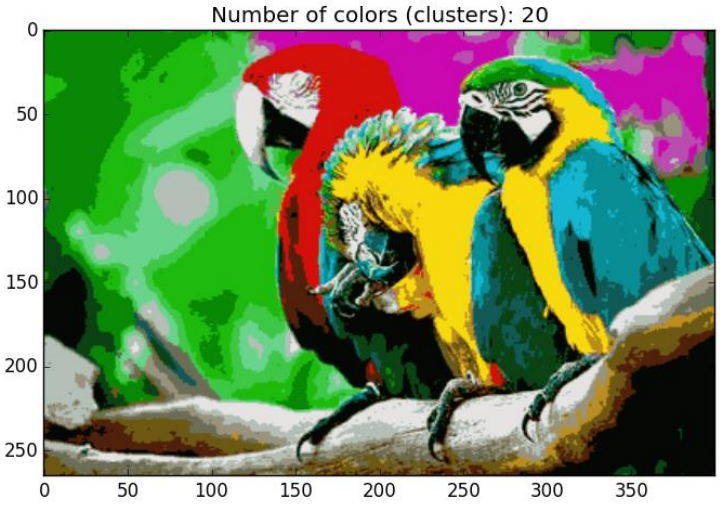
ЗАДАНИЕ 2

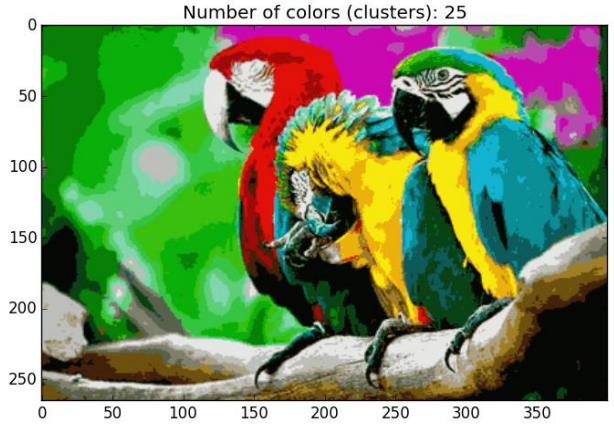
Были произведены и визуализированы кластеризации для картинки с попугаями с различным количеством кластеров (в данном случае – цветов):

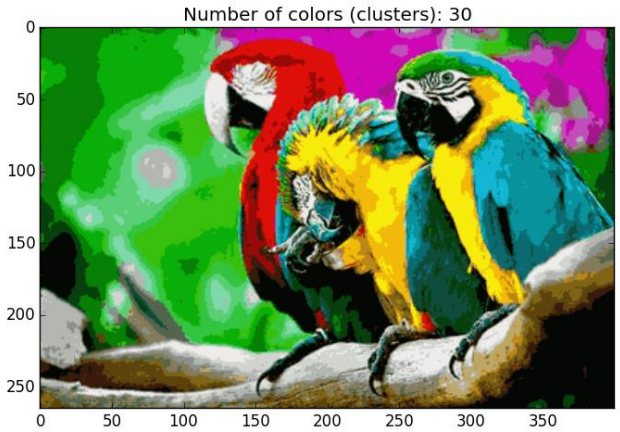


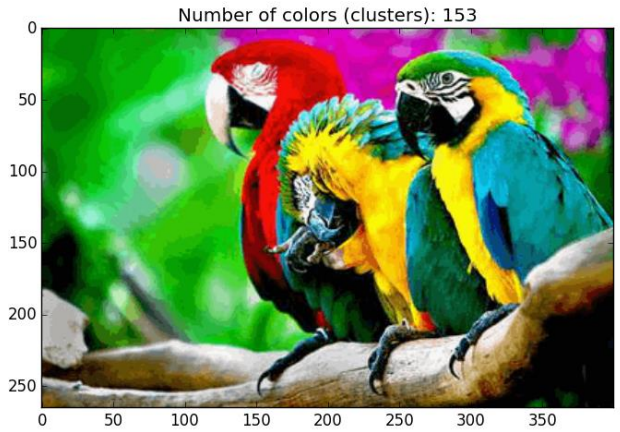

Number of colors (clusters): 3
0
50
100
150
200
250
0
50
100
150
200
250
300
350


Number of colors (clusters): 10
0
50
100
150
200
250
0
50
100
150
200
250
300
350


Number of colors (clusters): 20
0
50
100
150
200
250
0
50
100
150
200
250
300
350



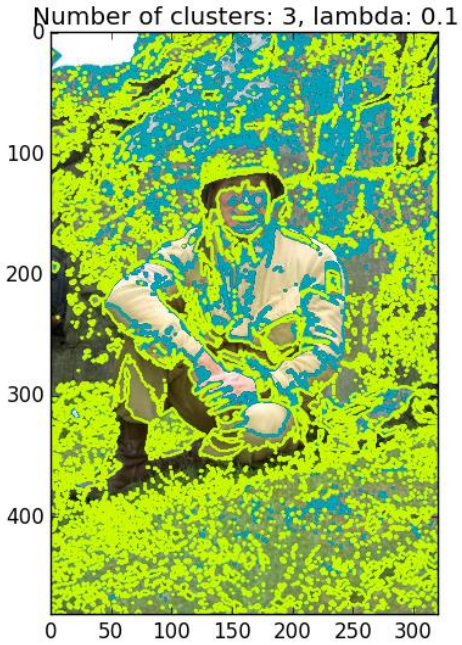


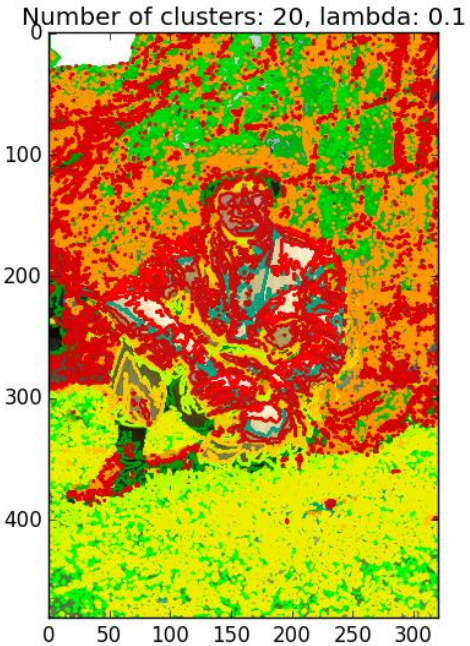


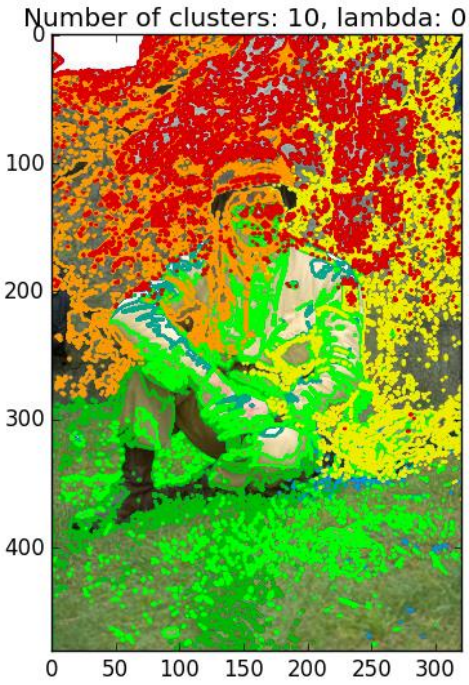
Более-менее приемлемым изображение становится, начиная с 20 кластеров (цветов). С увеличением количества цветов, что очевидно, качество изображения улучшается, что иллюстрирует пример с количеством кластеров, равным 153. Но все же искажение можно заметить.

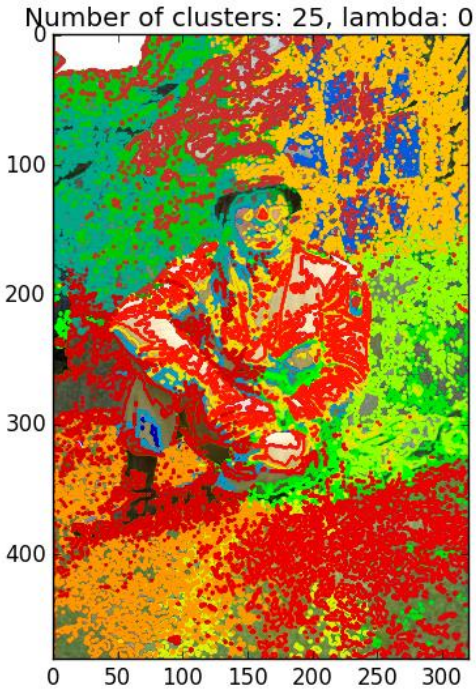
ЗАДАНИЕ 3

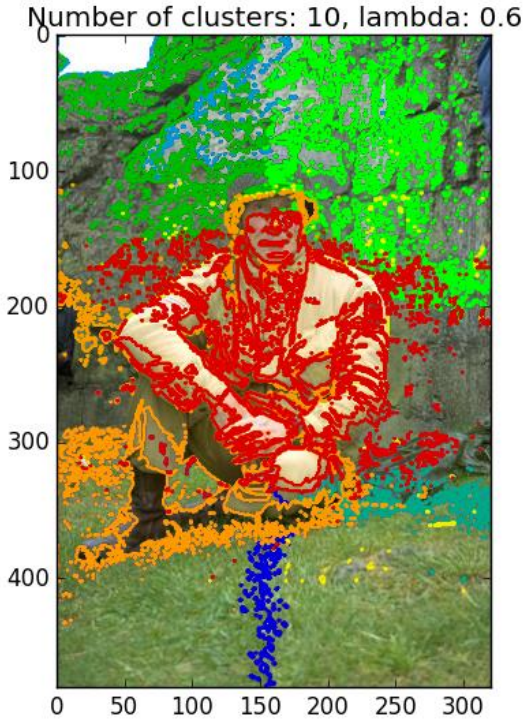
Были произведены и визуализированы сегментации (на основе кластеризации) с различными количествами кластеров и взятого параметра лямбда:

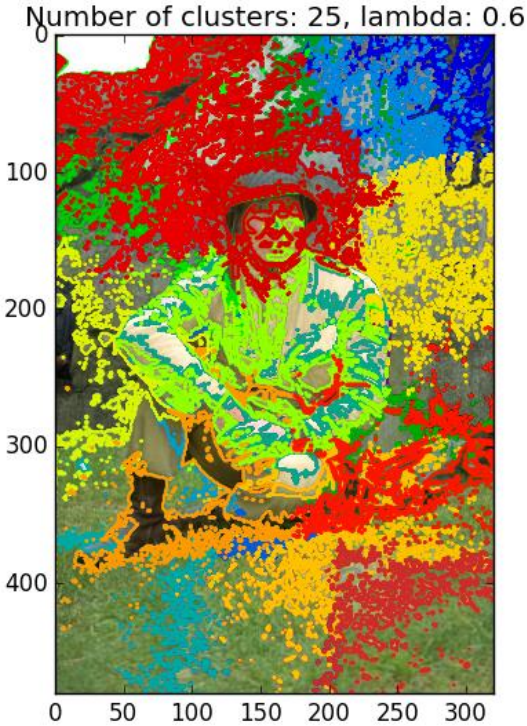


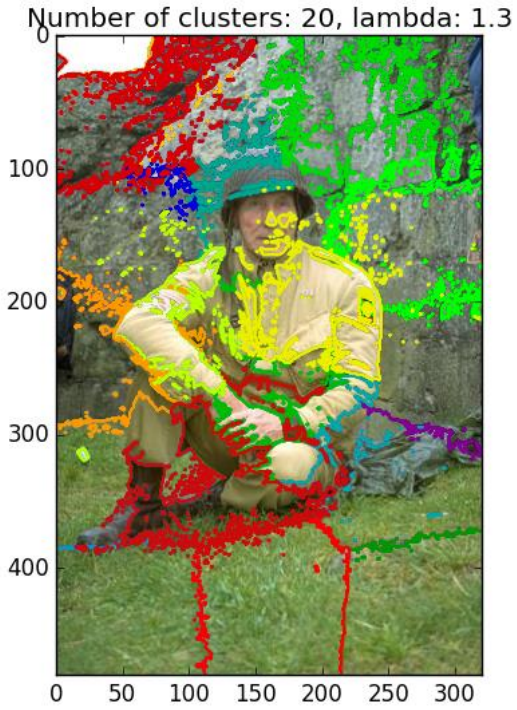


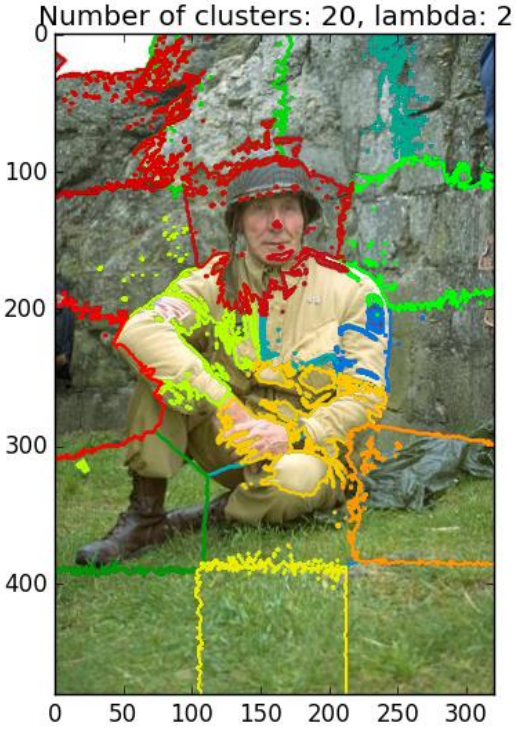

Number of clusters: 10, lambda: 0.3
0
100
200
300
400
0
50
100
150
200
250
300

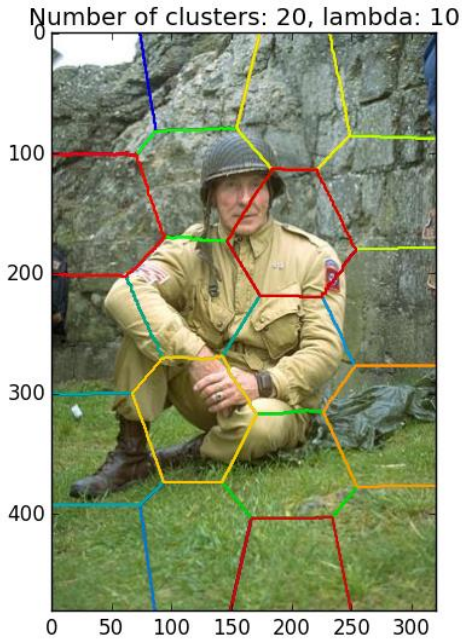

Number of clusters: 25, lambda: 0.3
0
100
200
300
400
0
50
100
150
200
250
300

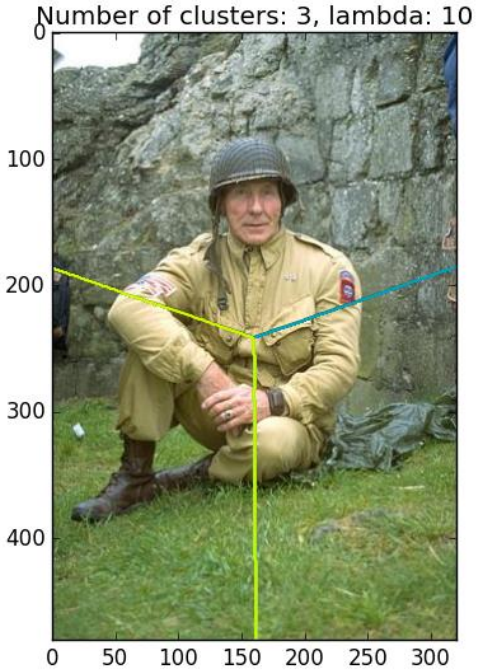

Number of clusters: 10, lambda: 0.6
0
100
200
300
400
0
50
100
150
200
250
300


Number of clusters: 25, lambda: 0.6
0
100
200
300
400
0
50
100
150
200
250
300


Number of clusters: 20, lambda: 1.3
0
100
200
300
400
0
50
100
150
200
250
300


Number of clusters: 20, lambda: 2
0
100
200
300
400
0
50
100
150
200
250
300





Количество кластеров, очевидно, влияет на количество выделенных сегментов. Что касается лямбда, при совсем малых значениях контур достаточно сложно воспринимать, а при высоких значениях контуры имеют вид диаграммы Вороного и малоинформативны на взгляд. Поэтому для наглядности и информативности лучше выбирать средние значения.

СПИСОК ЛИТЕРАТУРЫ

1) **Анализ данных (Программная инженерия)** – http://wiki.cs.hse.ru/Анализ_данных_(Программная_инженерия)

ТЕКСТ ПРОГРАММЫ

```
__author__ = 'Lev Osipov'

import numpy as np
import random
from sklearn.datasets import load_digits
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_mutual_info_score as
mutual_score
from sklearn.metrics import adjusted_rand_score as rand_score
import matplotlib.pyplot as plt
from skimage.io import imread


def swap(digs, i1, i2):
    temp = digs.data[i1]
    digs.data[i1] = digs.data[i2]
    digs.data[i2] = temp

    temp = digs.images[i1]
    digs.images[i1] = digs.images[i2]
    digs.images[i2] = temp

    temp = digs.target[i1]
    digs.target[i1] = digs.target[i2]
    digs.target[i2] = temp


def shuffle_digits(digs):
    indices = np.arange(len(digs))
    random.shuffle(indices)
    for i in xrange(len(indices)):
        swap(digs, i, indices[i])


def errors_analyzing(digs, kmns, n_cl):
    # Array for defining connection between labels and clusters
    cluster_digits = np.zeros(n_cl)
    # Array with numbers of errors and most erroneous number
    errors = np.zeros(shape=(n_cl, 3))
    # Array with error examples (for each cluster)
    err_examples = np.zeros(n_cl)
    # Defining cluster digits and counting errors
    for i in xrange(n_cl):
        # Array for true labels
        temp = np.zeros(n_cl)
        for j in xrange(len(kmns.labels_)):
            # Counting number of "votes"
            if kmns.labels_[j] == i:
                temp[digs.target[j]] += 1
        # Defining the digit
        index = np.argmax(temp, axis=0)
```

```
        cluster_digits[i] = index
        errors_number = 0
        # Counting errors
        max_errors = 0
        max_error_ind = 0
        for k in xrange(len(temp)):
            if k != index:
                errors_number += temp[k]
                if max_errors < temp[k]:
                    max_errors = temp[k]
                    max_error_ind = k
        errors[i, 0] = errors_number
        errors[i, 1] = max_errors
        errors[i, 2] = max_error_ind
        # Finding example
        for b in xrange(len(kmns.labels_)):
            if kmns.labels_[b] == float(i) and cluster_digits[i]
!= digs.target[b]:
                err_examples[i] = b
                break
    return cluster_digits, errors, err_examples


def visualization(digs, kmns, cl_digs, errs, err_exs):
    # Showing cluster centres
    for i in xrange(len(kmns.cluster_centers_)):
        Z = kmns.cluster_centers_[i, :]
        Z = Z.reshape([8, 8])
        plt.title("Digit: " + str(int(cl_digs[i])) + ", total
number of errors: " + str(int(errs[i, 0])) +
                  ", most erroneous digit " + str(int(errs[i,
2])) + " (" +
                  str(int(errs[i, 1])) + " errors)")
        plt.imshow(Z, cmap='Blues')
        plt.show()

    # Showing errors
    for i in xrange(len(err_exs)):
        Z = digs.data[err_exs[i]]
        Z = Z.reshape([8, 8])
        plt.title("Digit " + str(int(digs.target[err_exs[i]])) +
                  " in cluster with digit " +
str(int(cl_digs[kmns.labels_[err_exs[i]]])))
        plt.imshow(Z, cmap='Blues')
        plt.show()


def task1():

    # Data loading
    digits = load_digits()
    cluster_numbers = 10
```

```
    # Testing different methods of initialization in clustering
    n_tests = 25
    for init_type in ('k-means++', 'random'):
        results1 = np.zeros(n_tests)
        results2 = np.zeros(n_tests)
        for i in xrange(n_tests):
            # Data shuffling
            shuffle_digits(digits)

            kMeans = KMeans(n_clusters=cluster_numbers,
init=init_type)
            kMeans.fit(digits.data)
            predict = kMeans.predict(digits.data)
            results1[i] = mutual_score(digits.target, predict)
            results2[i] = rand_score(digits.target, predict)

        plt.title("Initialization type: " + init_type)
        plt.plot(results1, label='AMI')
        plt.plot(results2, label='RMI')
        plt.ylim(0.6, 0.8)
        plt.legend()
        plt.show()

    # Took k-means++
    shuffle_digits(digits)
    kMeans = KMeans(n_clusters=cluster_numbers)
    kMeans.fit(digits.data)

    # Defining clusters and finding errors
    cluster_digits, errors, err_examples =
errors_analyzing(digits, kMeans, cluster_numbers)
    visualization(digits, kMeans, cluster_digits, errors,
err_examples)


def cluster_parrots(im, n_cl, sh0, sh1, sh2):
    # Clustering
    kMeans = KMeans(n_clusters=n_cl)
    kMeans.fit(im)
    new_image = np.zeros(shape=(sh0, sh1, sh2))
    # Filling new image
    for i in xrange(len(kMeans.labels_)):
        ind1 = i / sh1
        ind2 = i % sh1
        new_image[ind1, ind2] =
kMeans.cluster_centers_[kMeans.labels_[i]]

    # Showing
    plt.title("Number of colors (clusters): " + str(n_cl))
    plt.imshow(new_image)
    plt.show()
```

```
def task2():
    # Data reading
    image = imread("parrots.jpg")
    image = np.array(image, dtype=np.float64) / 255
    shape0 = image.shape[0]
    shape1 = image.shape[1]
    shape2 = image.shape[2]
    image = image.reshape(shape0 * shape1, shape2)
    # Clustering tests
    cluster_parrots(image, 153, shape0, shape1, shape2)
    cluster_parrots(image, 3, shape0, shape1, shape2)
    cluster_parrots(image, 10, shape0, shape1, shape2)
    cluster_parrots(image, 20, shape0, shape1, shape2)
    cluster_parrots(image, 25, shape0, shape1, shape2)
    cluster_parrots(image, 30, shape0, shape1, shape2)
    cluster_parrots(image, 50, shape0, shape1, shape2)


def lambda_injection(im, sh0, sh1, sh2, lam):
    data = np.zeros(shape=(sh0, sh1, sh2 + 2))
    for i in xrange(sh0):
        for j in xrange(sh1):
            for k in xrange(sh2):
                data[i][j][k] = im[i][j][k]
            data[i][j][sh2] = lam * i
            data[i][j][sh2 + 1] = lam * j
    data = data.reshape(sh0 * sh1, sh2 + 2)
    return data


def cluster_segments(im, lam, n_cl):
    shape0 = im.shape[0]
    shape1 = im.shape[1]
    shape2 = im.shape[2]
    # Injecting lambdas
    data = lambda_injection(im, shape0, shape1, shape2, lam)
    # Clustering
    kMeans = KMeans(n_clusters=n_cl)
    kMeans.fit(data)
    # Showing
    plt.title("Number of clusters: " + str(n_cl) + ", lambda: "
+ str(lam))
    plt.imshow(im)
    labels = kMeans.labels_
    labels = labels.reshape(shape0, shape1)
    for i in xrange(n_cl):
        plt.contour(labels == i, contours=1,
colors=[plt.cm.spectral(i / float(n_cl)), ])
    plt.show()


def task3():
    # Data reading
```

```
    image = imread("grass.jpg")
    # Clustering segments
    cluster_segments(image, 0.1, 3)
    cluster_segments(image, 0.1, 20)
    cluster_segments(image, 0.3, 10)
    cluster_segments(image, 0.3, 25)
    cluster_segments(image, 0.6, 10)
    cluster_segments(image, 0.6, 25)
    cluster_segments(image, 1.3, 20)
    cluster_segments(image, 2, 20)
    cluster_segments(image, 10, 20)
    cluster_segments(image, 10, 3)


task1()
task2()
task3()
```