РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

Дисциплина: Интеллектуальный анализ данных

Студент: Николаев Александр Викторович

Группа: НФИбд-01-17

Москва 2020

Вариант №9

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

1. Считаем из заданного набора данных (http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data) репозитария UCI значения двух признаков (Length - столбец с индексом 1 и Diameter - столбец с индексом 2) и метки класса (Sex - столбец с индексом 0).

In [2]:
```
#read data
data = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data',
                   header=None,
                   usecols=[0, 1, 2],
                   names=['sex', 'length', 'diameter'])
#transform category to int
data['sex'] = data['sex'].astype('category').cat.codes
match_class = {
    2: 'Male',
    0: 'Female',
    1: 'Infant'
}
data.head()
```

Out[2]:

	sex	length	diameter
0	2	0.455	0.365
1	2	0.350	0.265
2	0	0.530	0.420
3	2	0.440	0.365
4	1	0.330	0.255

1. Проверим данные на наличие пропусков.

In [3]:
```
data.isna().sum()
```

Out[3]:
```
sex         0
length      0
diameter    0
dtype: int64
```

Пропусков нет.

1. Посмотрим количество меток класса и заодно их распределение.

In [4]:
```
data['sex'].value_counts(normalize=True)
```

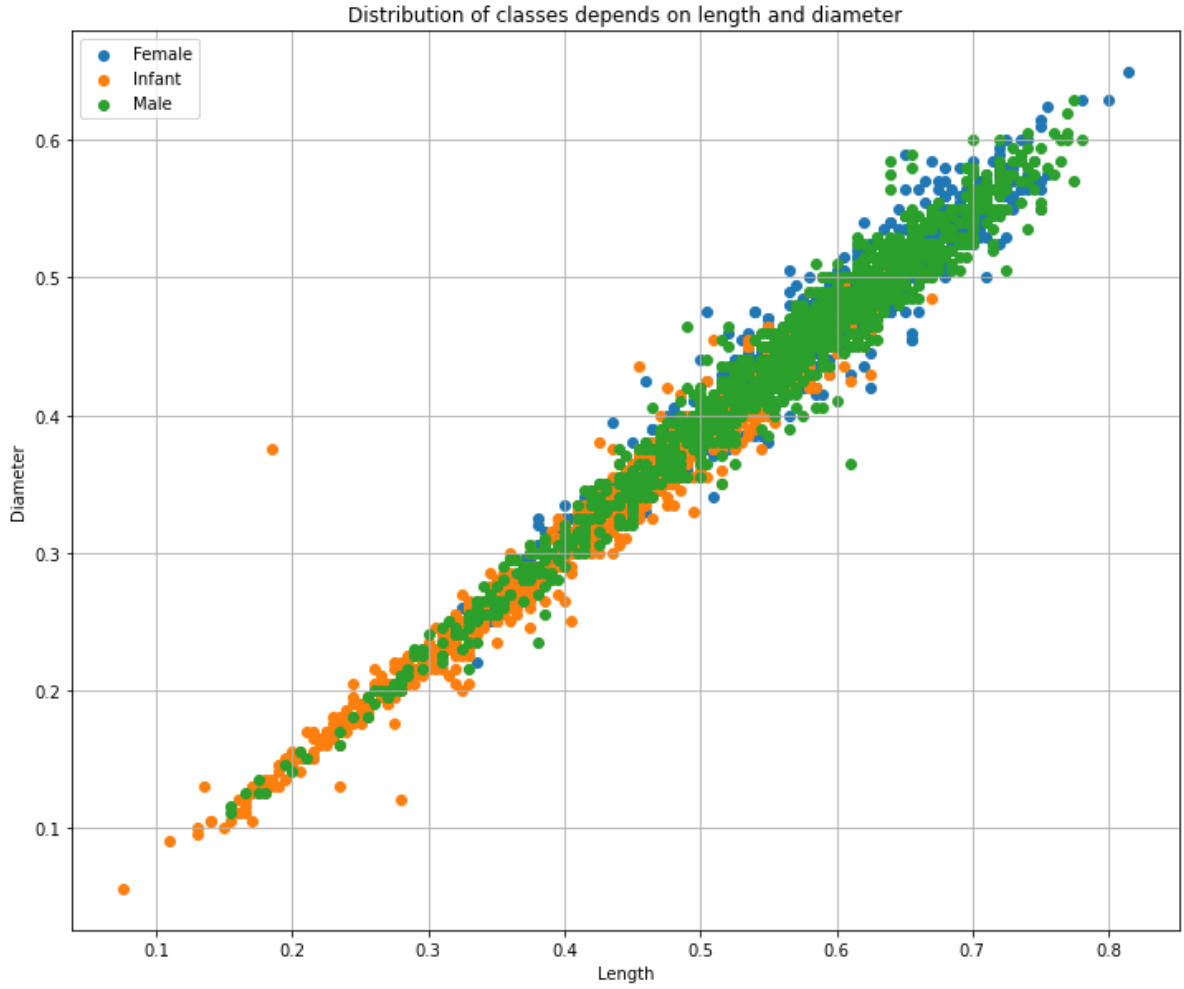
Out[4]:
```
2    0.365813
1    0.321283
0    0.312904
Name: sex, dtype: float64
```

Классов меньше 7, поэтому объединять ничего не будем. Классов три: мужчина, женщина и младенец. Распределение близко к равномерному.

1. Визуализируем набор данных

In [5]:
```
def plot_classes(X, y):
    plt.figure(figsize=(12,10))
    clusters = np.unique(y)
    for cluster in clusters:
        cluster_idx = np.where(y == cluster)
        plt.scatter(X[cluster_idx, 0], X[cluster_idx, 1], label=f'{
match_class[cluster]}')
    plt.grid(True)
    plt.title('Distribution of classes depends on length and diamet
er')
    plt.legend()
    plt.xlabel('Length')
    plt.ylabel('Diameter')
    plt.show()
```

In [6]:
```
X = data.drop(columns='sex').values
y = data['sex'].values
plot_classes(X, y)
```



По полученной визуализации можно сделать вывод о том, что удачно кластеризовать данные с помощью данных двух признаков не получится, т.к. классы по этим признакам сильно перемешены между собой, и нет поверхности (на глаз), которая их хорошо бы могла отделить :)

1. Проведем кластеризацию набора данных с помощью следующих алгоритмов: K-means, Agglomerative Clustering, DBSCAN, Gaussian Mixture Model. Количество кластеров = количество кластеров.

In [7]:
```
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.mixture import GaussianMixture
```

In [8]:
```
n_clusters = np.unique(y).size

kmeans = KMeans(n_clusters=n_clusters)
kmeans_preds = kmeans.fit_predict(X)

ac = AgglomerativeClustering(n_clusters=n_clusters)
ac_preds = ac.fit_predict(X)

dbscan = DBSCAN() # нельзя задать n_clusters, зададим min_samples
dbscan_preds = dbscan.fit_predict(X)

gm = GaussianMixture(n_components=n_clusters)
gm_preds = gm.fit_predict(X)
```

1. Для каждого из алгоритмов кластеризации построим матрицу сопряженности и найдём значения мер качества кластеризации: условная энтропия, TP, TN, FP, FN и индекс Rand

In [9]:
```
from sklearn.metrics.cluster import contingency_matrix
```

Реализуем функции подсчета мер качеств.

In [10]:
```
def calc_cond_entropy(matrix):
    cond_entropy = 0
    n = matrix.sum()
    for j in range(matrix.shape[1]):
        entropy = 0
        nj = matrix[:, j].sum()
        for i in range(matrix.shape[0]):
            if matrix[i, j] == 0:
                continue
            p = matrix[i, j] / nj
            entropy -= p * np.log2(p)
        cond_entropy += nj / n * entropy
    return cond_entropy


def calc_pair_metrics(matrix):
    TP = (np.sum(matrix ** 2) - matrix.sum()) // 2
    FN = (np.sum(test.sum(axis=1) ** 2) - np.sum(test ** 2)) // 2
    FP = (np.sum(test.sum(axis=0) ** 2) - np.sum(test ** 2)) // 2
    N = test.sum() * (test.sum() - 1) // 2
    TN = N - TP - FN - FP
    return TP, FN, FP, TN


def calc_rand_index(matrix):
    TP, FN, FP, TN = calc_pair_metrics(matrix)
    return (TP + TN) / (TP + FN + FP + TN)
```

Проверим правильность реализации на примере из лекции (для ирисов)

In [11]:
```
test = np.array([
    [50, 0, 0],
    [0, 1, 49],
    [0, 36, 14]
])
entropy = calc_cond_entropy(test)
TP, FN, FP, TN = calc_pair_metrics(test)
rand_index = calc_rand_index(test)
print(entropy, TP, FN, FP, TN, rand_index, sep='\n')
```

```
0.36518238924260643
3122
553
722
6778
0.8859060402684564
```

Всё верно, посчитаем меры для наших алгоритмов.

In [12]:
```
algo_preds = {
    'K-means': kmeans_preds,
    'Agglomerative Clustering': ac_preds,
    'DBSCAN': dbscan_preds,
    'Gaussian Mixture': gm_preds
}
for key, value in algo_preds.items():
    print(f'Algorithm: {key}')
    matrix = contingency_matrix(y, value)
    entropy = calc_cond_entropy(matrix)
    TP, FN, FP, TN = calc_pair_metrics(matrix)
    rand_index = calc_rand_index(matrix)
    print(f'Conditional etropy = {entropy}')
    print(f'TP = {TP}')
    print(f'FN = {FN}')
    print(f'FP = {FP}')
    print(f'TN = {TN}')
    print(f'Rand = {rand_index}\n----------------------------------
------\n')
```

```
Algorithm: K-means
Conditional etropy = 1.3521481807765976
TP = 1349781
FN = 553
FP = 722
TN = -1339881
Rand = 0.8859060402684564
---------------------------------------

Algorithm: Agglomerative Clustering
Conditional etropy = 1.3777794773362055
TP = 1388788
FN = 553
FP = 722
TN = -1378888
Rand = 0.8859060402684564
---------------------------------------

Algorithm: DBSCAN
Conditional etropy = 1.5815100477524742
TP = 2919910
FN = 553
FP = 722
TN = -2910010
Rand = 0.8859060402684564
---------------------------------------

Algorithm: Gaussian Mixture
Conditional etropy = 1.3568180665453002
TP = 1404801
FN = 553
FP = 722
TN = -1394901
Rand = 0.8859060402684564
---------------------------------------
```

Заметим, что DBSCAN показал себя очень плохо с точки зрения условной энтропии (главная мера качества в моём идз), оно и понятно, т.к. DBSCAN определил количество кластеров = 1

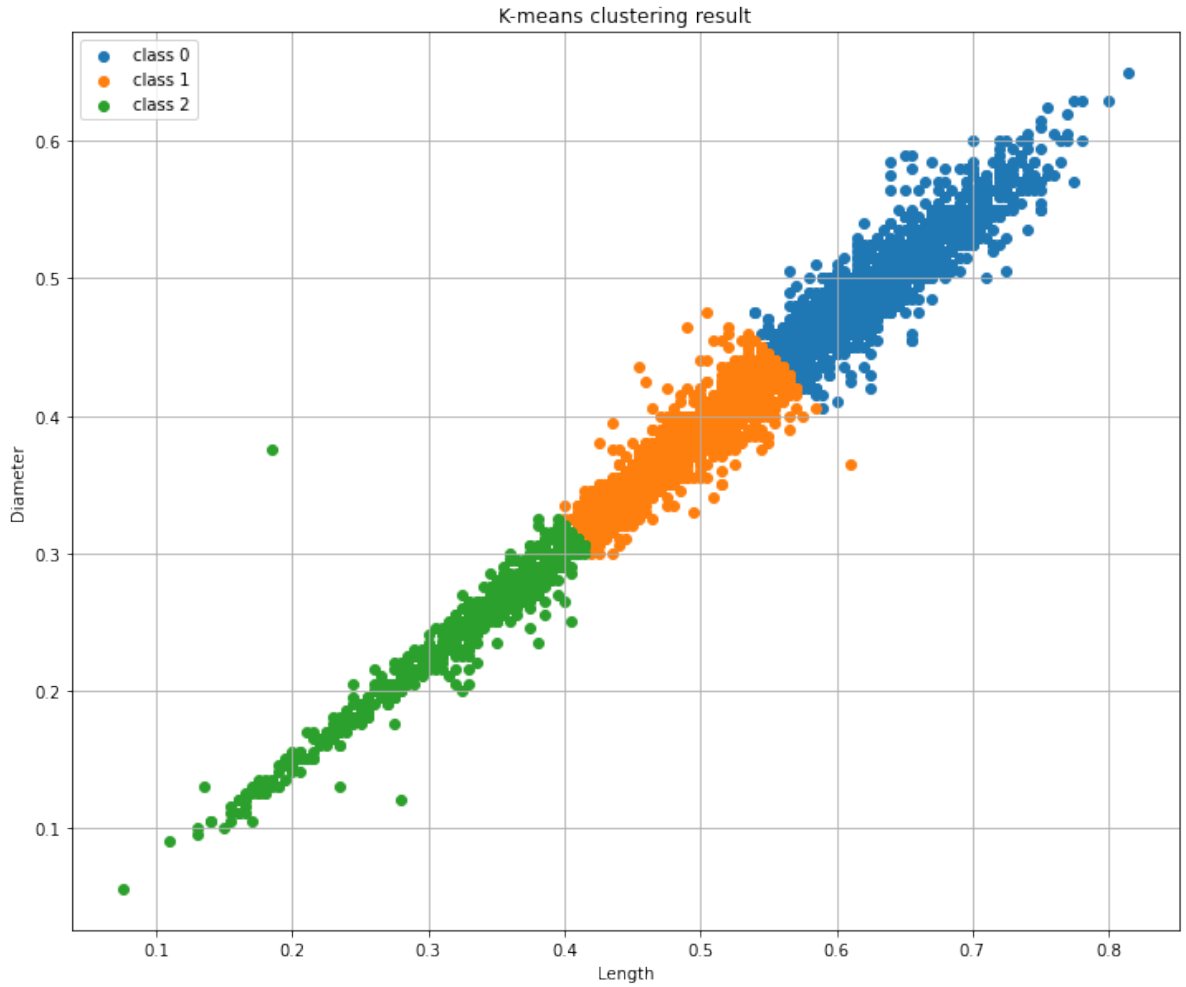
1. Определим алгоритм кластеризации, оптимальный с точки зрения условной энтропии.

Оптимальный алгоритм - K-means, т.к. именно он имеет наименьшую условную энтропию (см. вывод для алгоритмов сверху)

1. Визуализируем кластеризацию, полученную с помощью K-means.

In [13]:
```
def plot_clusters(X, y):
    plt.figure(figsize=(12,10))
    clusters = np.unique(y)
    for cluster in clusters:
        cluster_idx = np.where(y == cluster)
        plt.scatter(X[cluster_idx, 0], X[cluster_idx, 1], label=f'c
lass {cluster}')
    plt.grid(True)
    plt.title('K-means clustering result')
    plt.legend()
    plt.xlabel('Length')
    plt.ylabel('Diameter')
    plt.show()
```

In [14]:
```
plot_clusters(X, kmeans_preds)
```



Довольно неплохо, с определенной точки зрения.