Скачайте двнные классификации листьев растений по ссылке. https://archive.ics.uci.edu/ml/machine-learning-databases/00241/ (https://archive.ics.uci.edu/ml/machine-learning-databases/00241/)

Загрузим файл `data_Mar_64.txt` .

In [ ]:
```
import pandas as pd
import numpy as np
from google.colab import files
uploder = files.upload()
```

Выбрать файлы Файл не выбран

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving data_Mar_64.txt to data_Mar_64.txt
```

In [ ]:
```
data = pd.read_csv('data_Mar_64.txt', header=None)
```

Первый столбец - ответ, положим его в отдельную переменную.

In [ ]:
```
X, y_name = np.array(data.iloc[:, 1:]), data.iloc[:, 0]
```

Целевая переменная принимает текстовое значение. С помощью `LabelEncoder` из `sklearn` закодируй тектовую переменную `y_name` и сохраните полученные значения в переменную `y` .

In [ ]:
```
## your code here
```

С помощью метода главных компонент снизьте размерность признакового пространства до двух. Зафиксируйте `random_state=0`

In [ ]:
```
## your code here
```

Выберите объекты, которые соответствуют значениям от 0 до 14 целевой переменной `y` . Изобразите выбранные объекты в двумерном пространстве признаков с помощью метода `scatter` из `matplotlib.pyplot` . Чтобы разным цветом отобразить объекты разных классов, в метод `scatter` передайте `c = y[y<15]` .

In [ ]:
```
## your code here
```

Проделайте тоже самое для метода TSNE.

In [ ]:
```
## your code here
```

1. Укажите координаты объекта с индексом 0 (`X[0]`) после применения метода TSNE. Округлите числа до сотых.

```
In [ ]: ## your code here
```

1. Укажите координаты объекта с индексом 0 (X[0]) после применения метода PCA. Округлите числа до сотых.

```
In [ ]: ## your code here
```

1. Какие выводы можно сделать из полученных изображений?

- С мпомощью метода главных компонет удалось визуализировать объекты на плоскоти и объекты разных класов визуально разделимы
- **С мпомощью метода TSNE удалось визуализировать объекты на плоскоти и объекты разных класов визуально разделимы**
- С мпомощью методов TSNE и PCA удалось визуализировать объекты на плоскоти и объекты разных класов визуально разделимы
- С мпомощью методов TSNE и PCA удалось визуализировать объекты на плоскоти и объекты разных класов визуально не разделимы

K_means

Реализуйте класс MyKMeans.

Класс должен соответствовать шаблону, который приведен ниже

В конструктор класса передаются:

- n_clusters - число кластеров, на которое будут разбиты данные
- n_iters - максимальное число итераций, может быть сделано в данном алгоритме

В методе `fit` :

- self.centers - центры кластеров, которые пересчитываются на каждой итерации. Изначально выбираются случайным образом с фиксированным seed.

далее в цикле по числу итераций вам необходимо реализовать:

- вычисление ближайшего центра кластера для каждого объекта
- пересчет центра каждого кластера(среднее каждой из координат всех объектов, отнесенных к этому кластеру) посчитанные новые центры кластеров положите в переменную `new_centers`

В методе `predict` :

вычисляются ближайшие центры кластеров для объектов `X`

In [ ]:
```
from sklearn.metrics import pairwise_distances_argmin
class MyKMeans():
  def __init__(self, n_clusters=3, n_iters = 100):
    self.n_clusters = n_clusters
    self.n_iters = n_iters

  def fit(self, X):
    np.random.seed(0)
    self.centers = np.random.uniform(low=X.min(axis = 0),
                                     high=X.max(axis = 0),
                                     size=(self.n_clusters, X.shape[1]))

    for it in range(self.n_iters):

      ## your code here

      if np.all(self.centers == new_centers):
          break

      self.centers = new_centers


  def predict(self, X):
    labels = pairwise_distances_argmin(X, self.centers)
    return labels
```

Сгенерируем данные для кластеризации

In [ ]:
```
from sklearn import datasets
n_samples = 1000

noisy_blobs = datasets.make_blobs(n_samples=n_samples,
                                  cluster_std=[1.0, 3.0, 0.5],
                                  random_state=0)
```

In [ ]:
```
X, y = noisy_blobs
```

1. Кластеризуйте объекты noisy_blobs с помощью `MyKMeans` , используйте гиперпараметры `n_clusters=3` , `n_iters=100` . Укажите ответ для объекта с индексом 1.

In [ ]:
```
## your code here
```

1. Кластеризуйте объекты noisy_blobs, используйте гиперпараметры n_clusters=3, n_iters = 5. Укажите ответ для объекта с индексом 1.

In [ ]:
```
## your code here
```

1. Вычислите у какого числа объектов изменилась метка предсказываемого кластера при изменении гиперпараметра n_iters с 5 до 100

In [ ]:
```
## your code here
```

1. Определите сколько за сколько итераций сошелся алгоритм на объектах объекты noisy_blobs?

In [ ]:
```
## your code here
```

DBSCAN

1. Кластеризуйте объекты noisy_blobs с помощью DBSCAN. Используйте реализацию DBSCAN из sklearn. Зафиксируйте гиперпараметр eps=0.5. Укажите ответ для объекта с индексом 1.

In [ ]:
```
## your code here
```

1. Укажите полученное число кластеров?

In [ ]:
```
## your code here
```

1. Сколько объектов было отнечено к выбросам (имеют метку -1)?

In [ ]:
```
## your code here
```