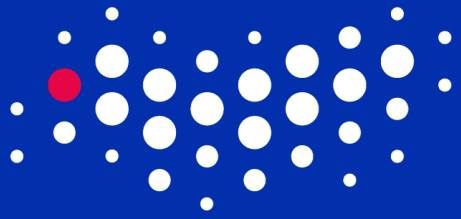


УНИВЕРСИТЕТ ИТМО

Параллельные алгоритмы кластеризации в Python

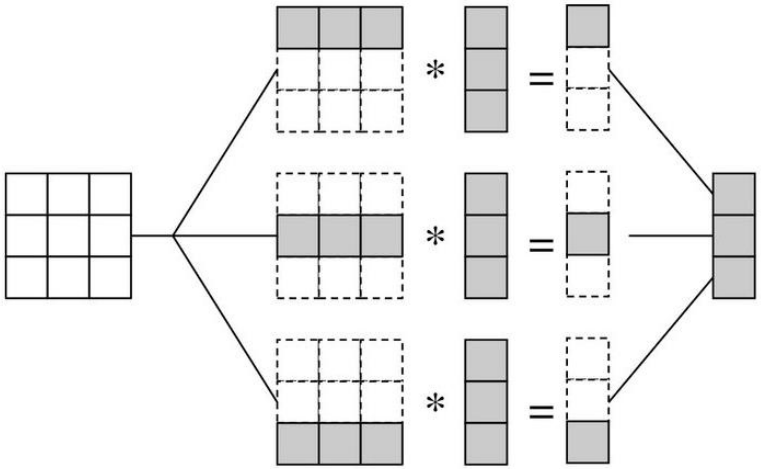
Докладчик: Михаил Сарафанов

Санкт-Петербург, 2019

Почему именно Python?

- Высокоуровневый язык
- На Python’е легко писать и читать код
- Содержит большое количество библиотек для анализа данных
- Большое сообщество разработчиков

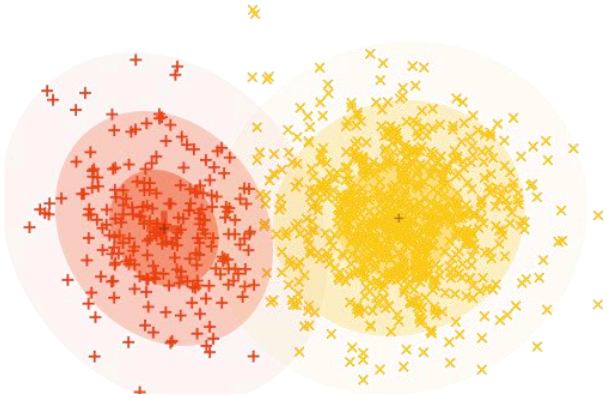
Зачем использовать параллельные вычисления?



- Python по скорости работы уступает некоторым другим языкам программирования, например, C++, C и т.д.
- Алгоритмы машинного обучения могут работать очень долго в однопоточном режиме
- Временная сложность многих алгоритмов кластеризации - O(n^2)

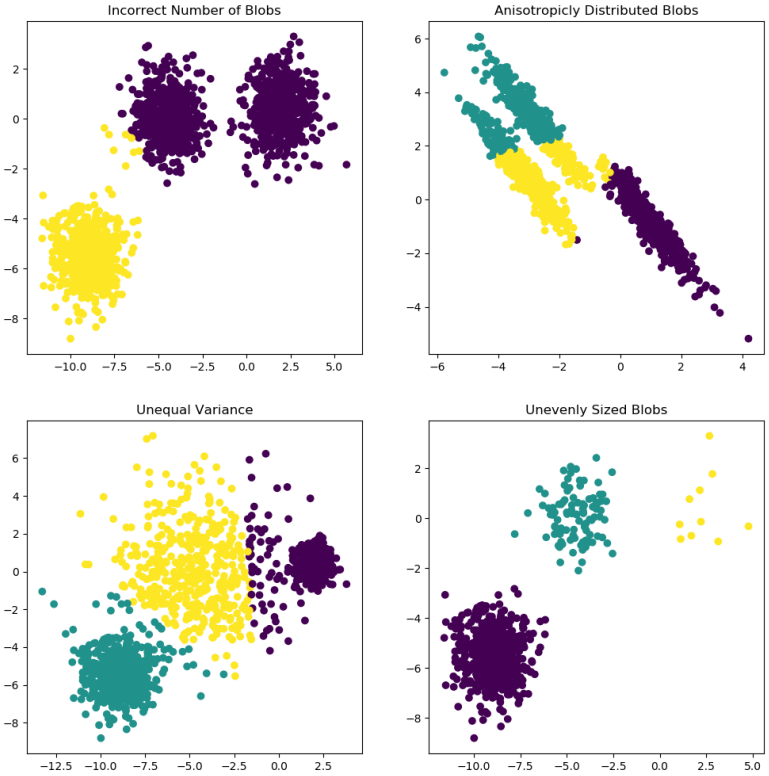
Алгоритмы кластеризации

- K-средних (K-Means)
- Агломеративная кластеризация (Agglomerative clustering)
- Плотностной алгоритм пространственной кластеризации с присутствием шума (Density-based spatial clustering of applications with noise DBSCAN)
- Гауссова смесь распределений (Gaussian mixtures)



K-средних (K-Means)

- Необходимо задать точное количество кластеров
- Алгоритм стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров
- Один из самых простых, но при этом достаточно эффективный алгоритм кластеризации, при этом имеет интересные модификации

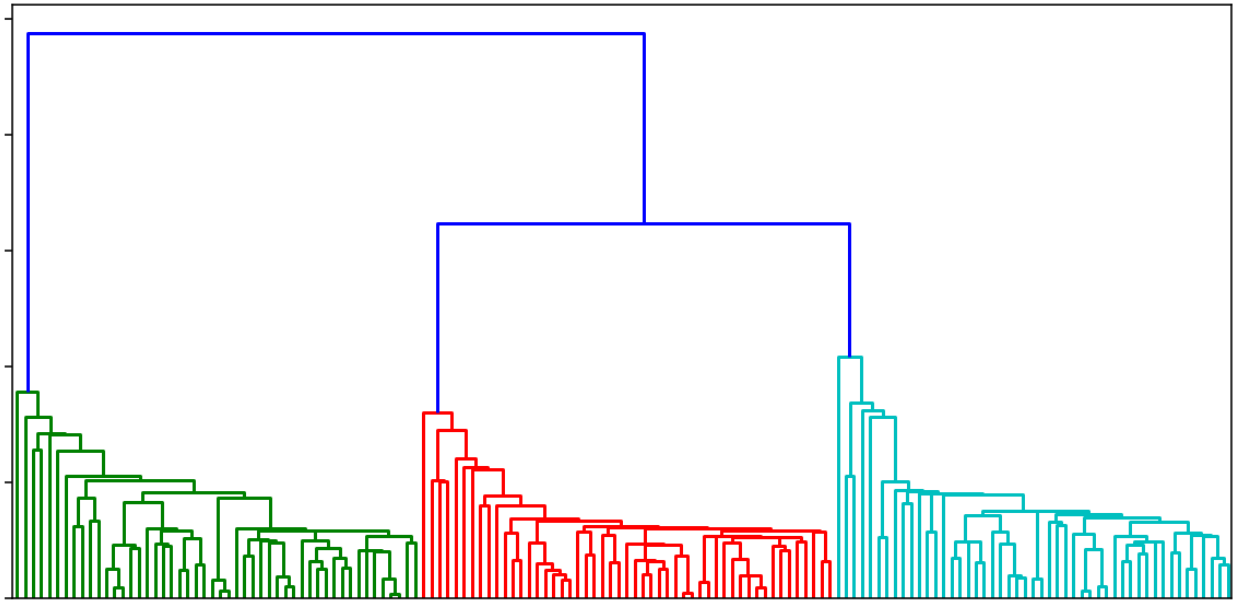


$$V = \sum_{i-1}^{k} \sum_{x \in S_i} (x - \mu_i)^2$$

где k — число кластеров, S_i — полученные кластеры, $i = 1, 2, \ldots, k$, а μ_i — центры масс всех векторов x из кластера S_i.

Агломеративная кластеризация

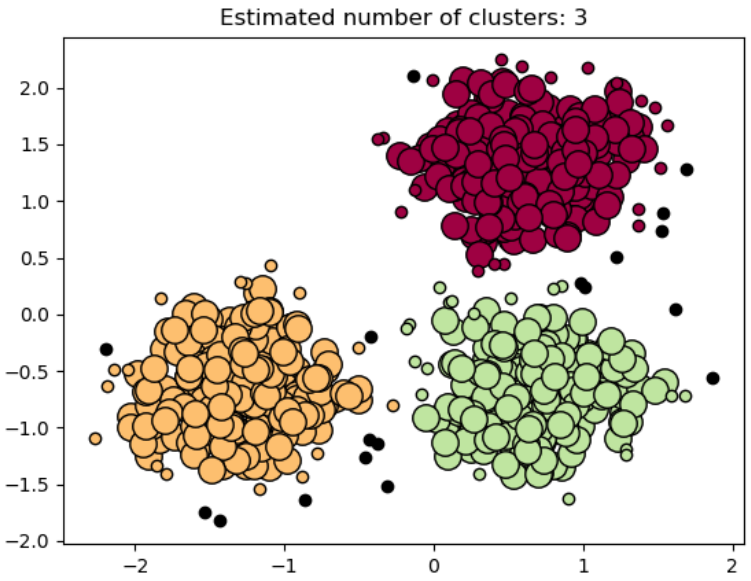
- Иерархические алгоритмы кластеризации, называемые также алгоритмами таксономии, строят не одно разбиение выборки на непересекающиеся классы, а систему вложенных разбиений. Результат таксономии обычно представляется в виде таксономического дерева — дендрограммы

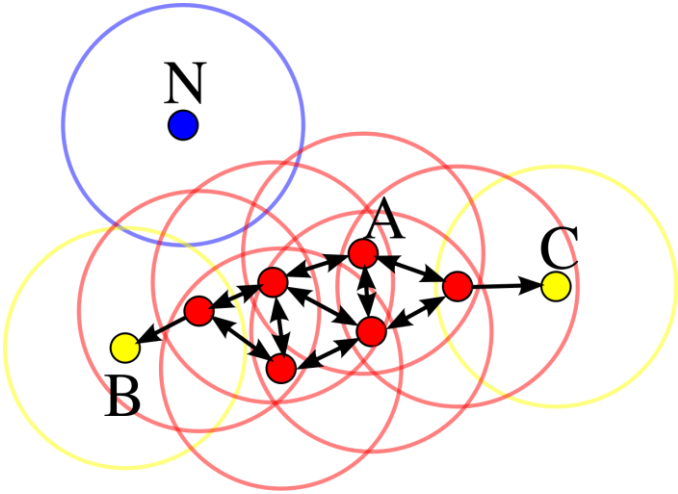


DBSCAN

- Алгоритм группирует вместе точки, которые тесно расположены (точки со многими близкими соседями)

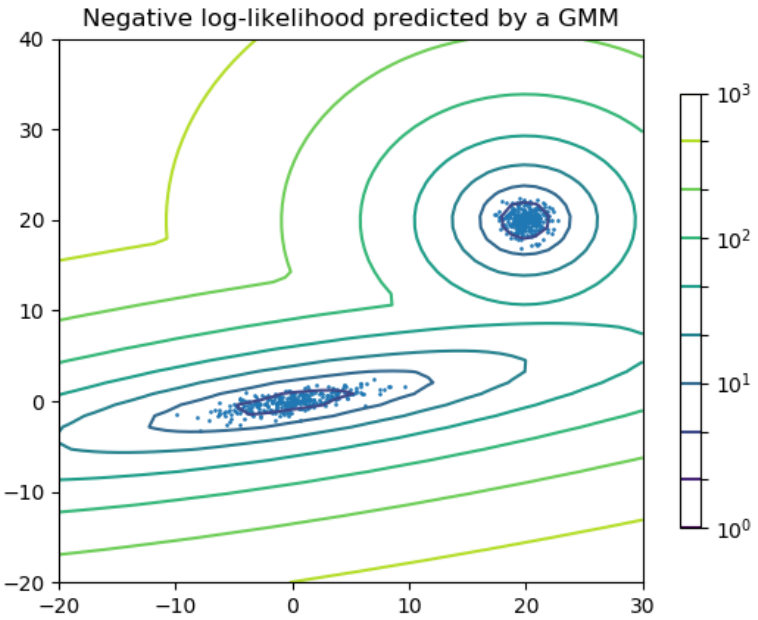
- Выбросами считаются точки, которые располагаются обособленно в областях с малой плотностью (ближайшие соседи которых лежат далеко)

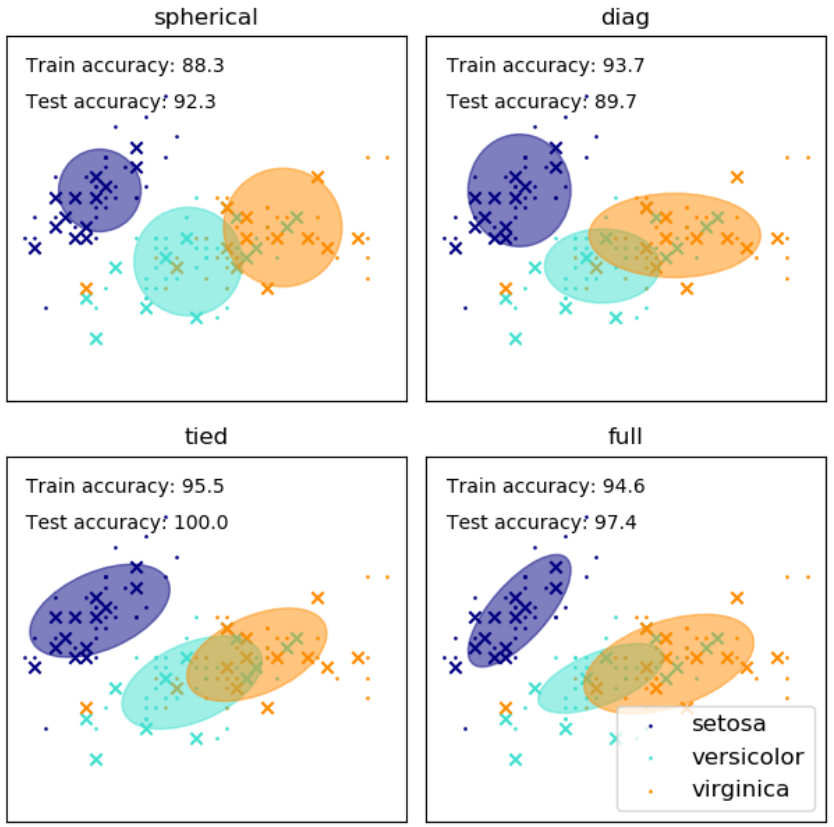




Гауссова смесь распределений

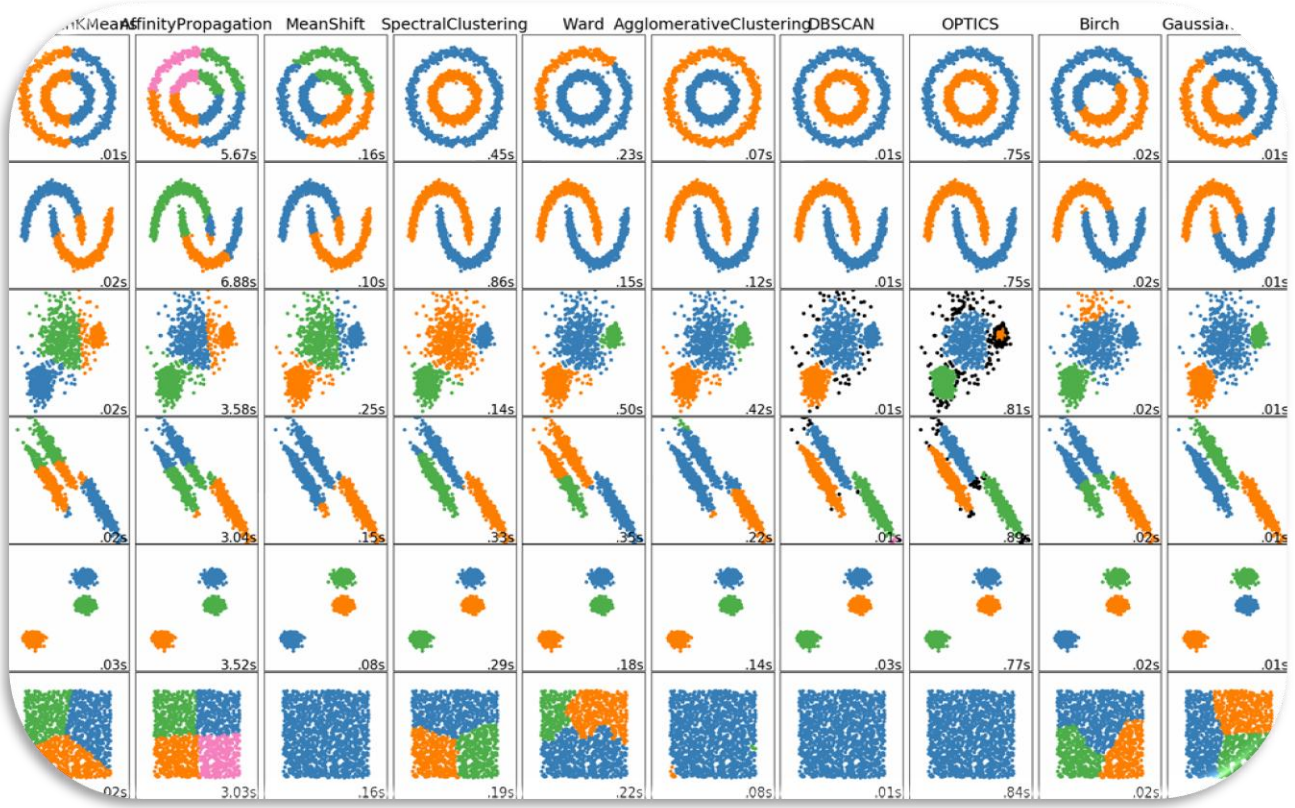
- Предполагается, что данные представляют собой смесь многомерных распределений Гаусса с определёнными параметрами





Реализация

- Scikit-learn



- Имеется отличная документация с различными иллюстрациями , математическими выкладками и примерами кода

Реализация кластеризации на основе Гауссовой смеси распределений

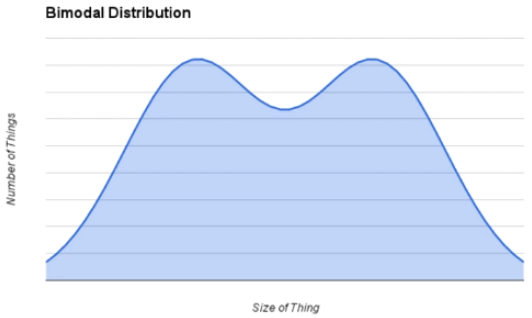
- Данный вероятностный алгоритм даёт нам приближённое значение распределения вероятностей наших данных
- Гауссова смесь – это сумма всех взвешенных гауссиан. Для представления весовых коэффициентов вводится новый символ π. Например, πk означает вероятность того, что x принадлежит k-той гауссиане:

$$p(x) = \pi_1 N(\mu_1 \sum_1) + \pi_2 N(\mu_2 \sum_2) + \pi_3 N(\mu_3 \sum_3) + \ldots$$

- Обучение гауссовой смеси распределений проходит в два этапа:

1) расчёт принадлежностей

2) перерасчёт всех параметров гауссиан



Статья

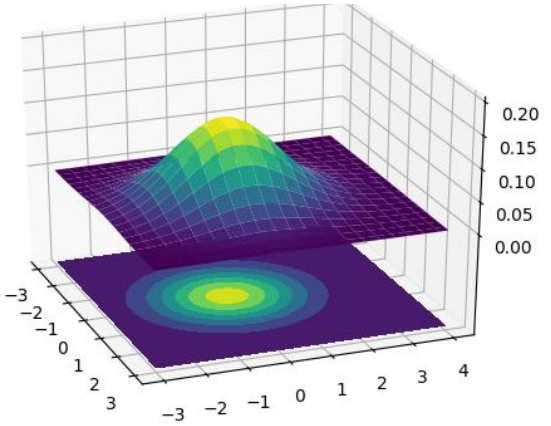
Реализация кластеризации на основе Гауссовой смеси распределений

- В python’е все вышесказанное умещается в несколько строк кода

```
# Загружаем набор данных
from sklearn import datasets
iris_df = datasets.load_iris()
# Готовимся кластеризовать
from sklearn.mixture import GaussianMixture

# Вызываем модель
gmm = GaussianMixture(n_components = 3)
# "Обучаем" её
gmm.fit(iris_df.data)
# Расставляем метки на всем наборе данных
gmm.predict(iris_df.data)
```

- Есть хорошая статья с примерами



Метрики качества кластеризации

- Внешние

Используют информацию об истинном разбиении на кластеры

- Adjusted Rand Index (ARI)
- Adjusted Mutual Information (AMI)
- Гомогенность, полнота, V-мера

- Внутренние

Основаны только на имеющемся неразмеченном наборе данных

- Силуэт

Все приведенные метрики реализованы в Python в sklearn.metrics

Подробнее метриках в кластеризации

Параллельные вычисления

- Threads (потоки): один вычислительный узел, несколько дочерних процессов, общий ресурс (память)
- Processes (процессы): несколько вычислительных узлов, несколько независимых процессов, память нужно разделить между процессами

Параллельные вычисления на Python

- Модули/библиотеки на Python, позволяющие производить параллельные вычисления:
- threading - Модуль threading на примерах
- multiprocessing - Многопоточность в одну строку
- joblib - Embarrassingly parallel for loops
- Dask - Почему каждый Data Scientist должен знать Dask
- Theano - Библиотеки для глубокого обучения Theano/Lasagne

Немного о каждом из модулей

- Threading

Документация

Рекомендации к применению: загрузка ресурсов из интернета или чтение файлов и папок на вашем компьютере, так как потоки в Python лучше всего работают с операциями I/O

- Multiprocessing

Документация

Рекомендации к применению: расчеты, включающие «тяжелые» математические операции, обработка больших объемов данных

Немного о каждой из библиотек

- Dask

Документация

Рекомендации к применению:

Объем данных, который необходимо анализировать, превышает RAM, библиотека позволяет работать с такими данными в привычном формате Dataframes и Arrays

- Theano

Документация

Рекомендации к применению:

Есть необходимость залезать в «серьезную математику», используются тензоры, оптимизируется работа нейросетей

Ну и конечно встроенное распараллеливание в Scikit-learn

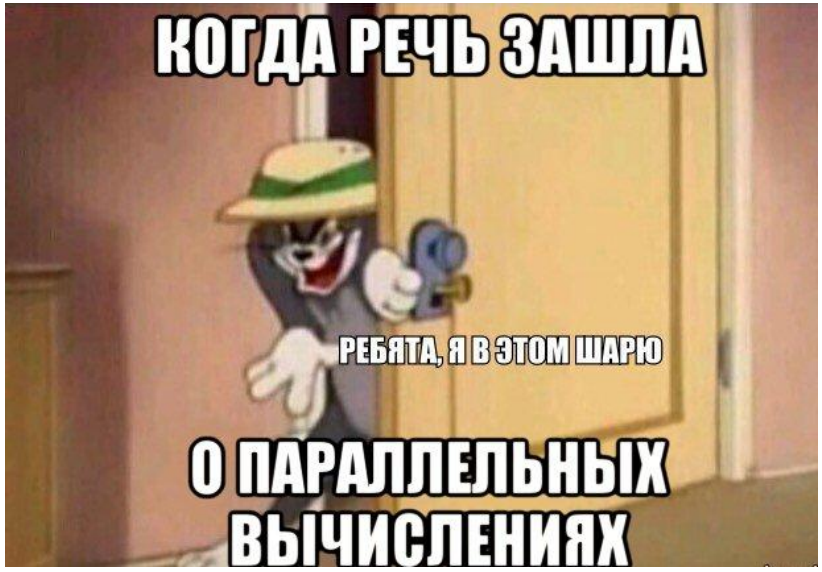
3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators='warn', criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None*) [source]

n_jobs : *int or None, optional (default=None)*

The number of jobs to run in parallel for both `fit` and `predict`. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

Все что нужно, указать параметр n_ jobs = «больше 1», - вы реализовали параллельные вычисления



УНИВЕРСИТЕТ ИТМО

Спасибо за внимание!

Давайте теперь попробуем все вышесказанное реализовать!

Санкт-Петербург, 2019