Рекомендательная система музыки

курсовая работа
студента 244 группы
Руденко Дмитрия Андреевича

Введение

Как из огромного набора аудиофайлов выбрать подходящие?

Перебор вручную

Очень трудоемко и долго

Автоматизировать процесс с помощью рекомендательной системы

Быстрее и совсем без трудовых затрат. Вот наш выбор!

Обзор существующих систем

Коллоборативная фильтрация

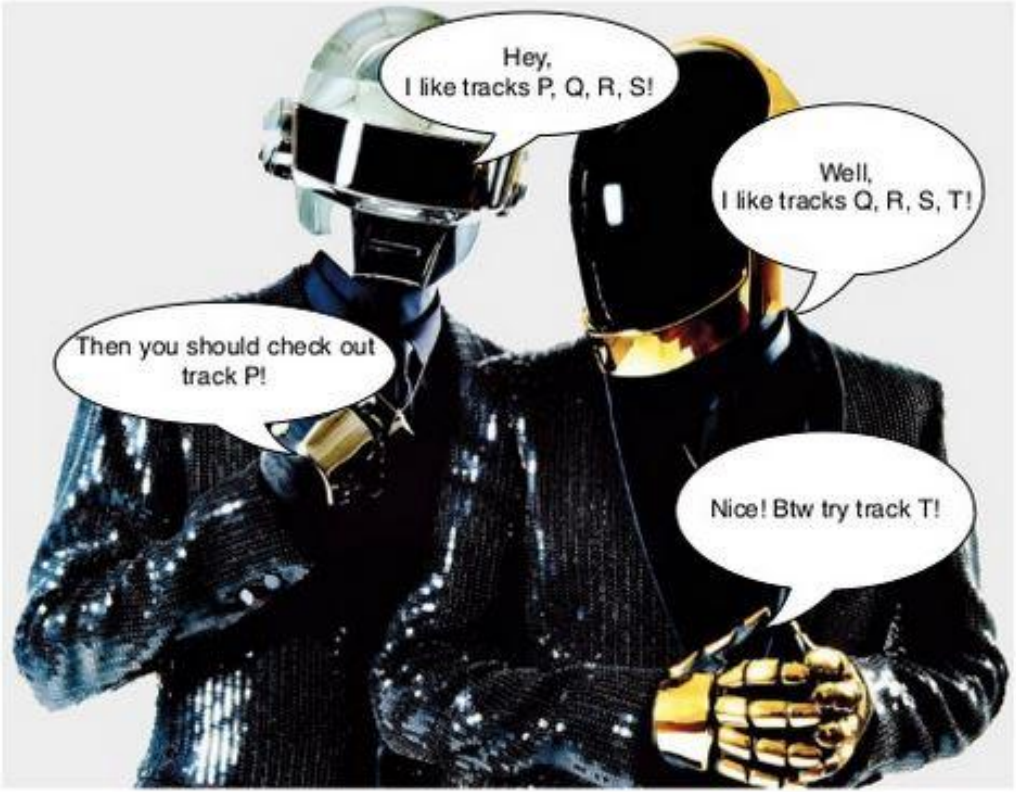
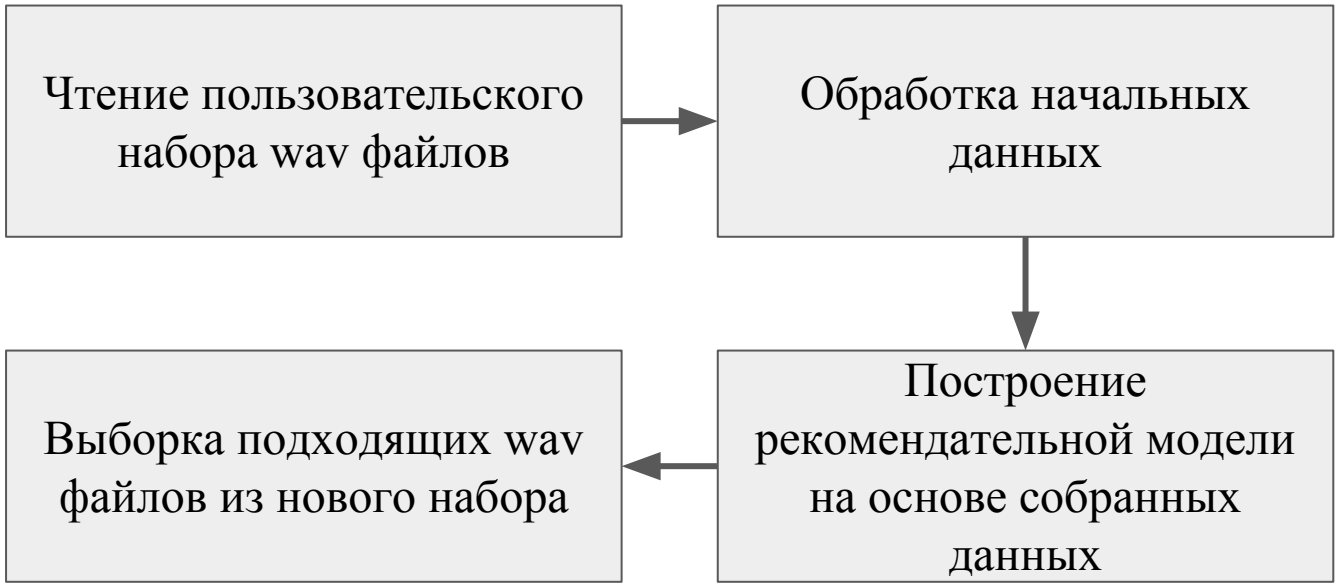


Image via Erik Bernhardsson

Постановка задачи

Целью данной работы является реализация приложения, которое по готовому набору пользовательской музыки формировало бы систему, способную определять, понравится ли конкретный аудиофайл пользователю.

Постановка задачи

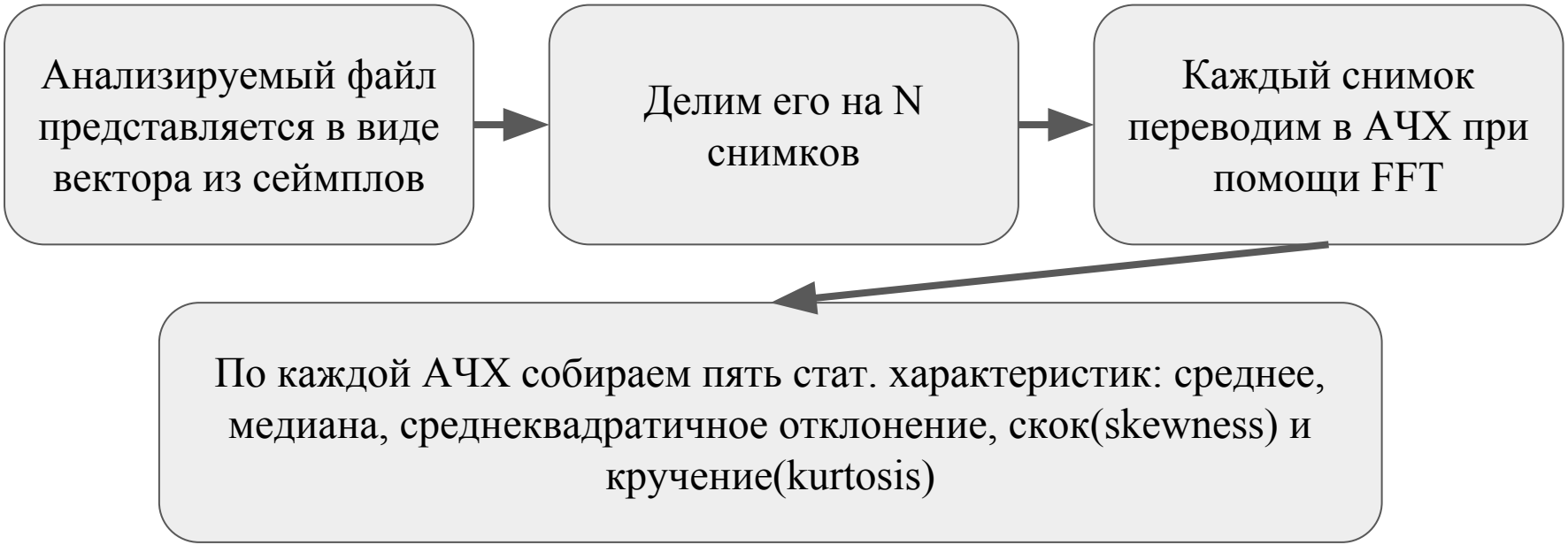


Чтение пользовательского набора

Составляется два набора данных: аудиозаписи, которые нравятся пользователю и которые нет. Чтение реализовано с помощью стандартных библиотек Python*: wave, struct и os.

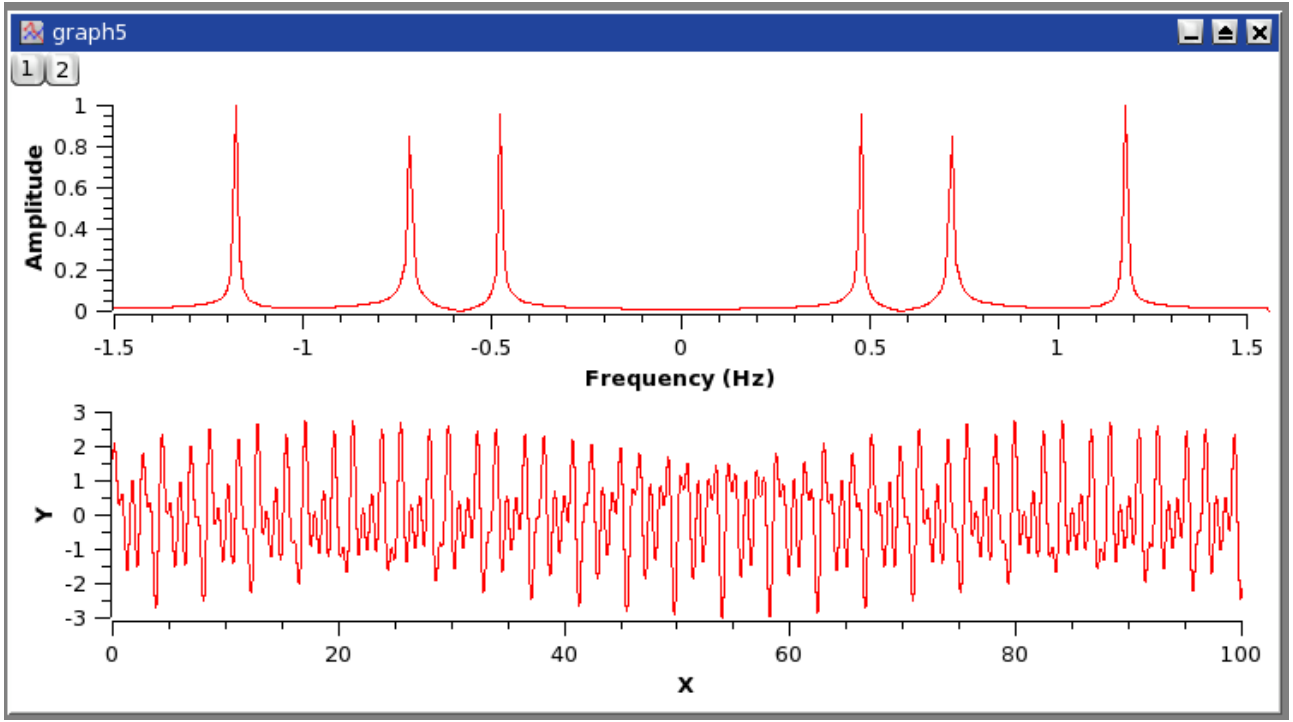
** Для проекта был выбран язык Python, так как на нем реализованы такие простые в обращении библиотеки, как NumPy и Sklearn, про которые будет рассказано далее.*

Обработка начальных данных



Таким образом, каждая композиция после обработки образует точку в 5N - мерном пространстве

FFT



FFT реализован в библитеке NumPy

Построение рекомендательной модели

Идеи:

1.Построить обобщающий прямоугольник: считаем максимум и минимум по всем координатам и далее считаем новую композицию подходящей, если каждая из ее координат входит в промежуток между максимумом и минимум. Очевидный минус: любой значительный выброс ломает модель

2. Обратится к алгоритмам классификации Data Maining.

Алгоритмы

Для исследования были выбраны три алгоритма классификации:

1. K-nearest neighbors
2. Support Vectors
3. Logistic Regression

Все три алгоритма реализованы в библиотеке scikit-learn (сокр. sklearn)

Теория

$$X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$$

$$\rho(u, x_{1;u}) \le \rho(u, x_{2;u}) \le \dots \le \rho(u, x_{m;u})$$

$$a(u) = \arg\max_{y \in Y} \sum_{i=1}^{m} [x_{i;u} = y] w(i, u);$$

$w(i,u) = [i \le k]$ — метод k ближайших соседей;

K-nearest neighbors

$$a(x, w) = \text{sign}\left(\sum_{j=1}^{n} w_j f_j(x) - w_0 \right)$$

$$Q(w) = \sum_{i=1}^{m} \ln\left(1 + \exp(-y_i \langle x_i, w \rangle)\right) \to \min_w.$$

Logistic Regression

$$a(x, w) = \text{sign}\left(\sum_{j=1}^{n} w_j f_j(x) - w_0 \right)$$

$$\sum_{i=1}^{m} (1 - M_i(x, w_0))_+ + \frac{1}{2C}(\|w\|)^2 \to \min_{w; w_0}$$

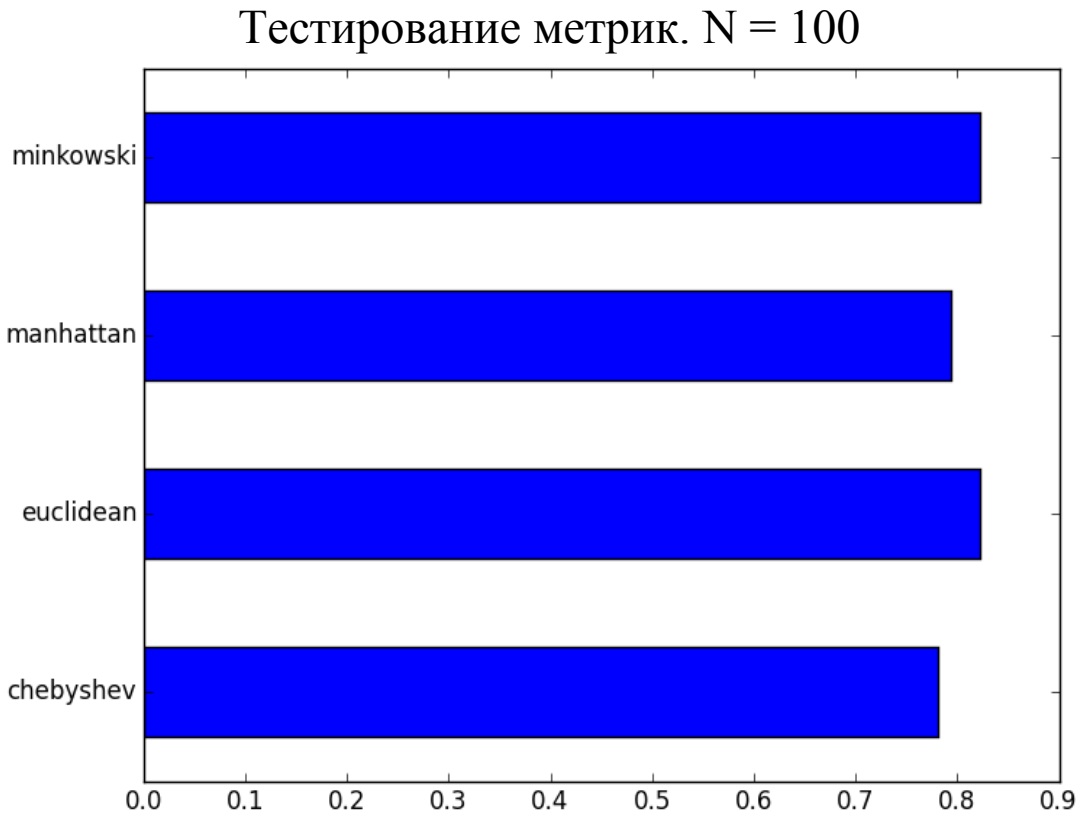
$$M_i(x, w_0) = y_i(\langle x_i, w \rangle - w_0)$$

Support Vector

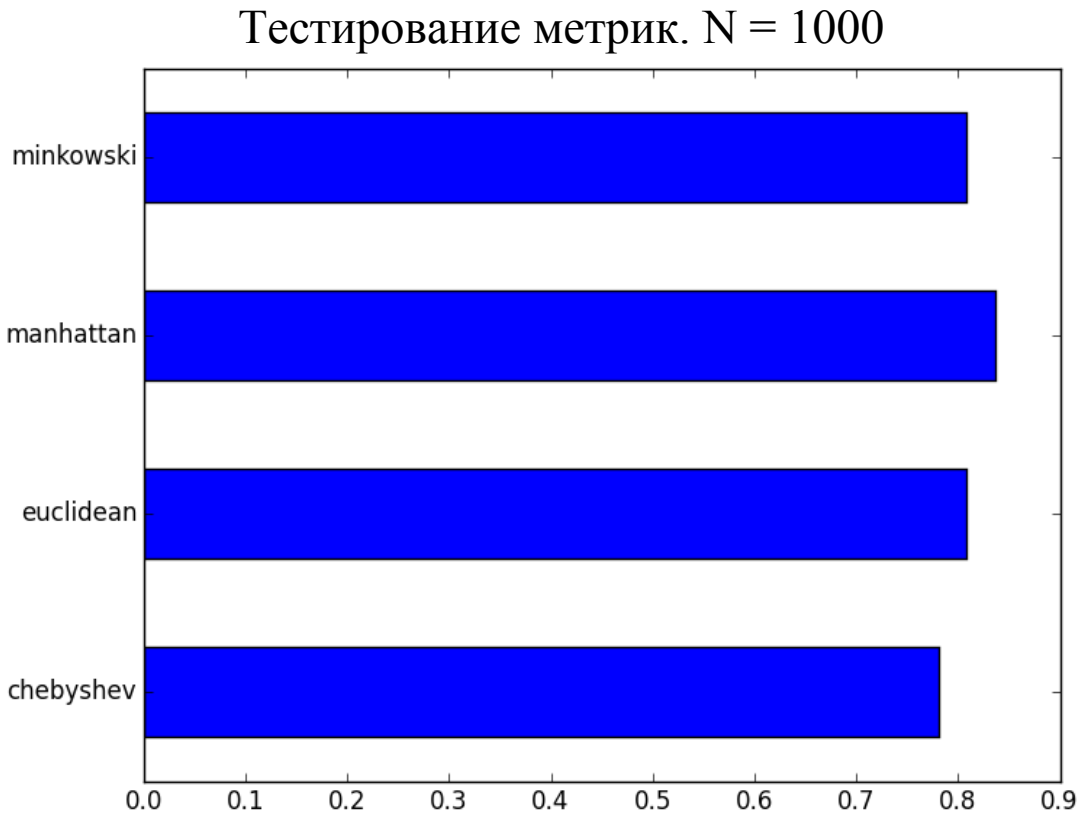
Выборка подходящих композиций из нового набора

Обрабатываем новый набор аналогично обработки начальных данных(собираем статистические данные) и для каждого из объектов запрашиваем у построенной модели ответ на вопрос: "К какому классу принадлежит этот объект?". Те из объектов, которые попадут в класс подходящих("True"), и будут искомые.

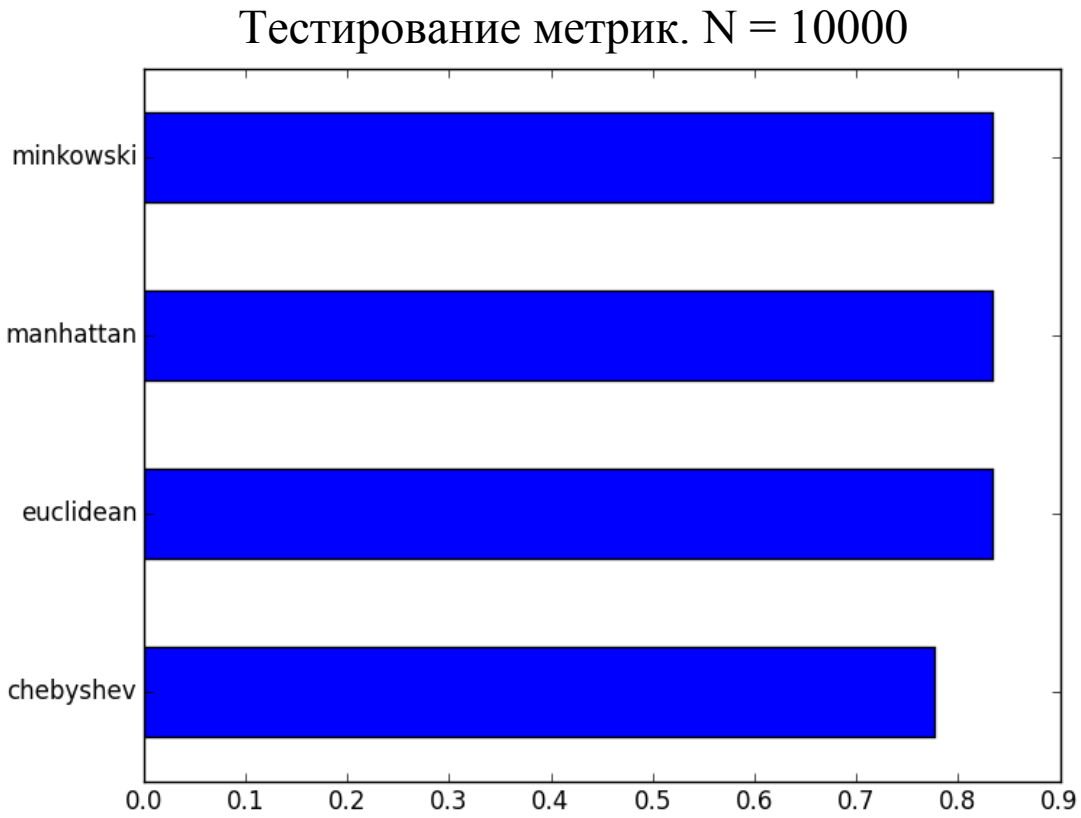
Тестирование



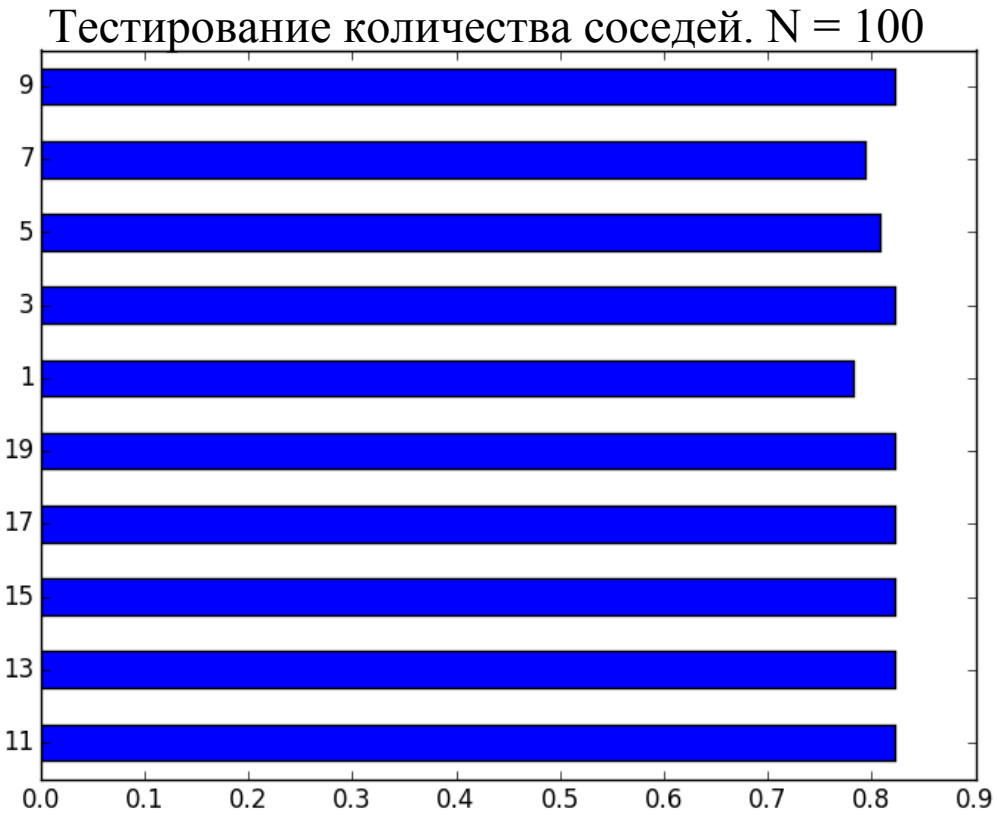
Тестирование



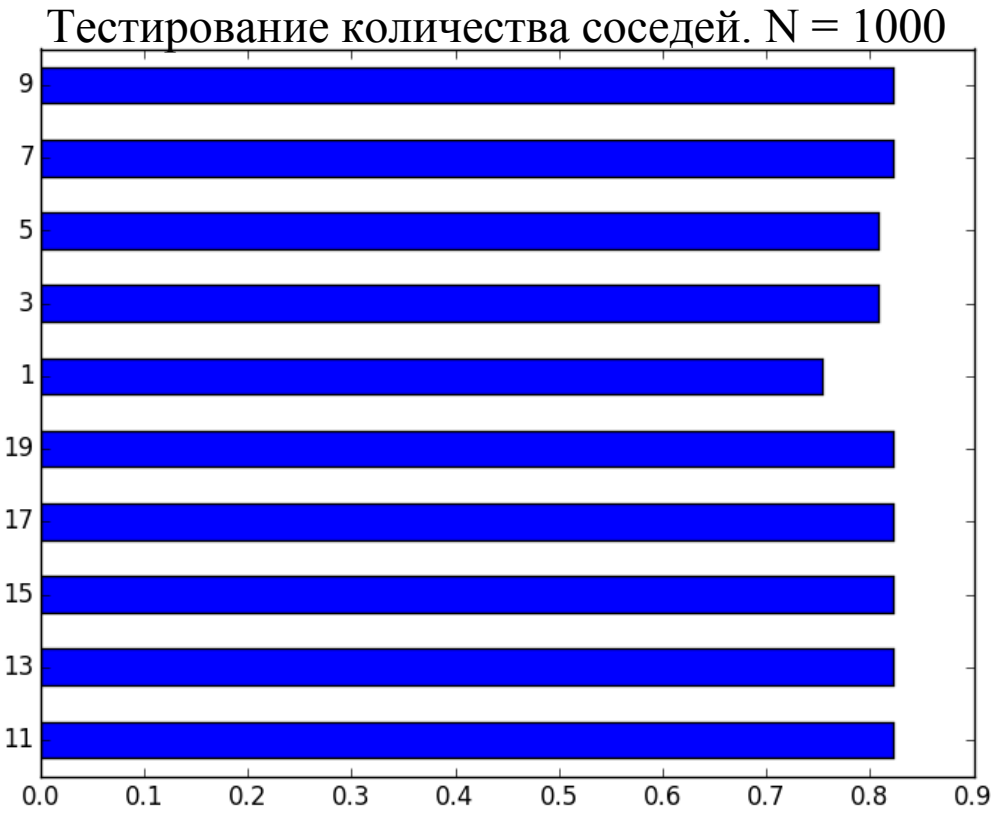
Тестирование



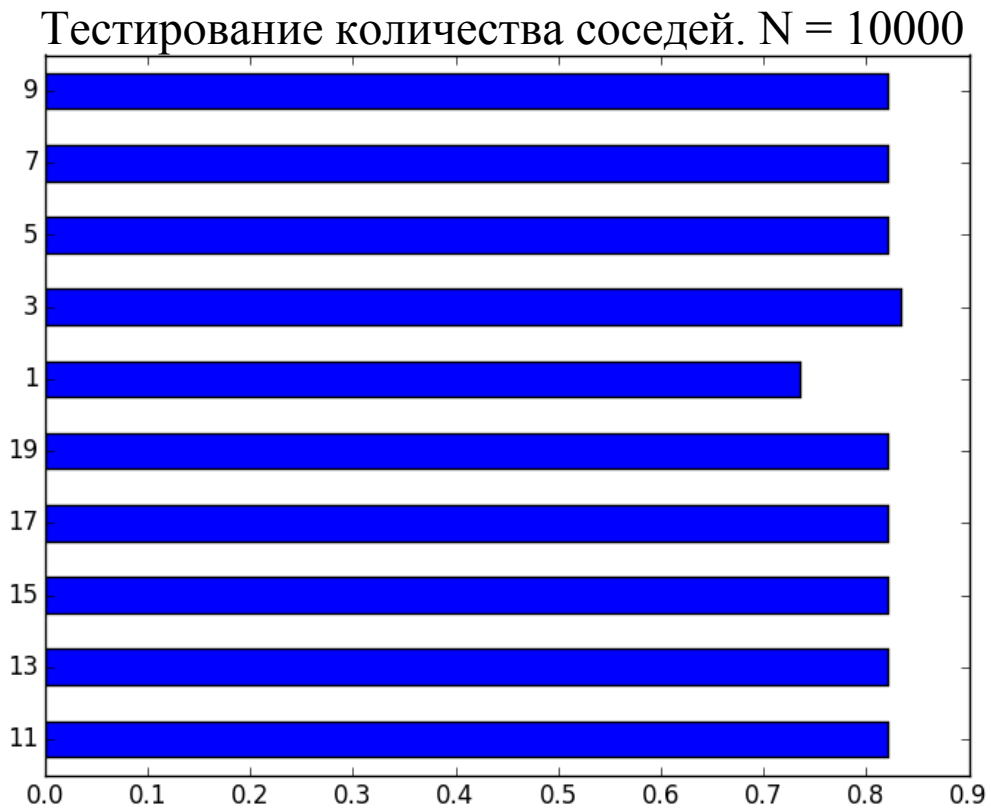
Тестирование



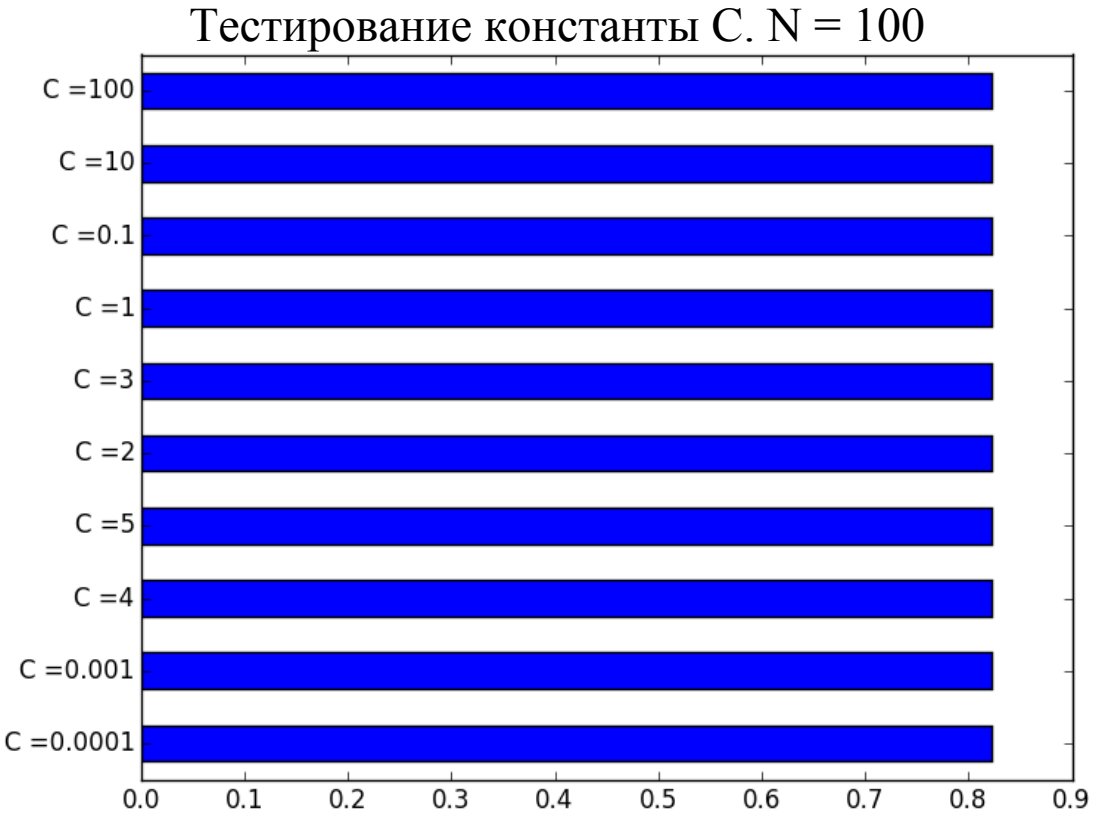
Тестирование



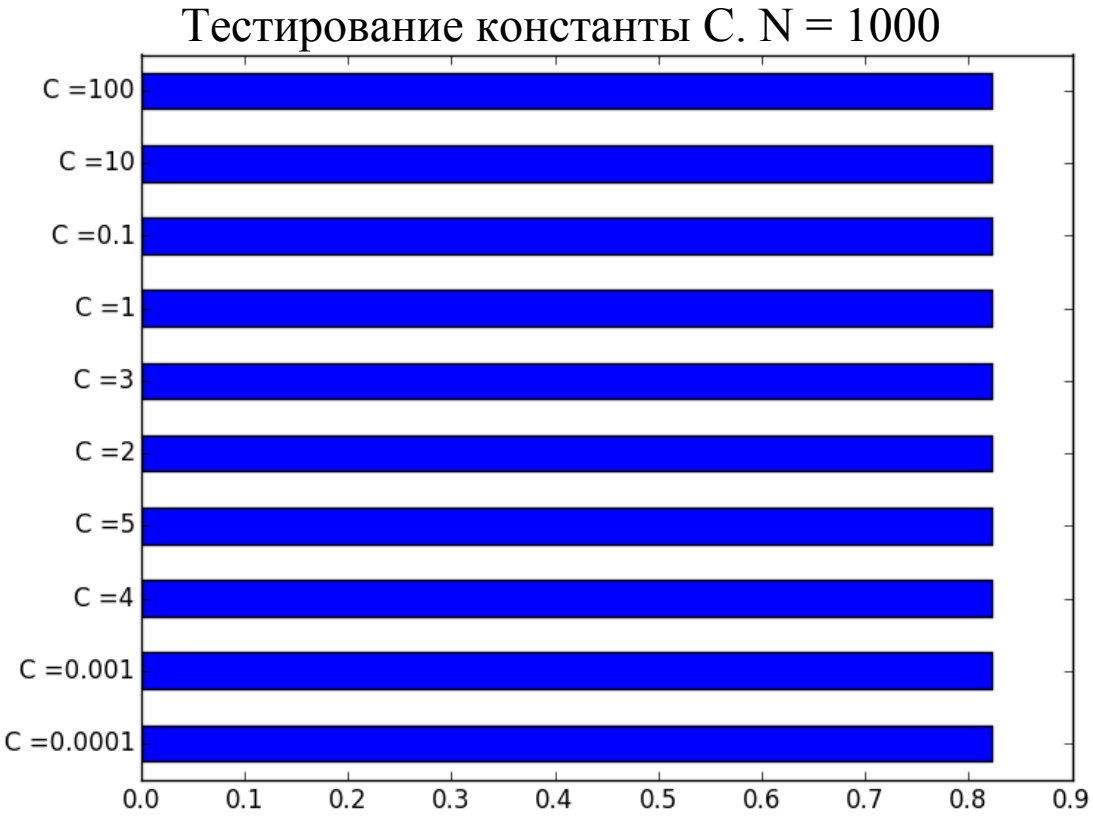
Тестирование



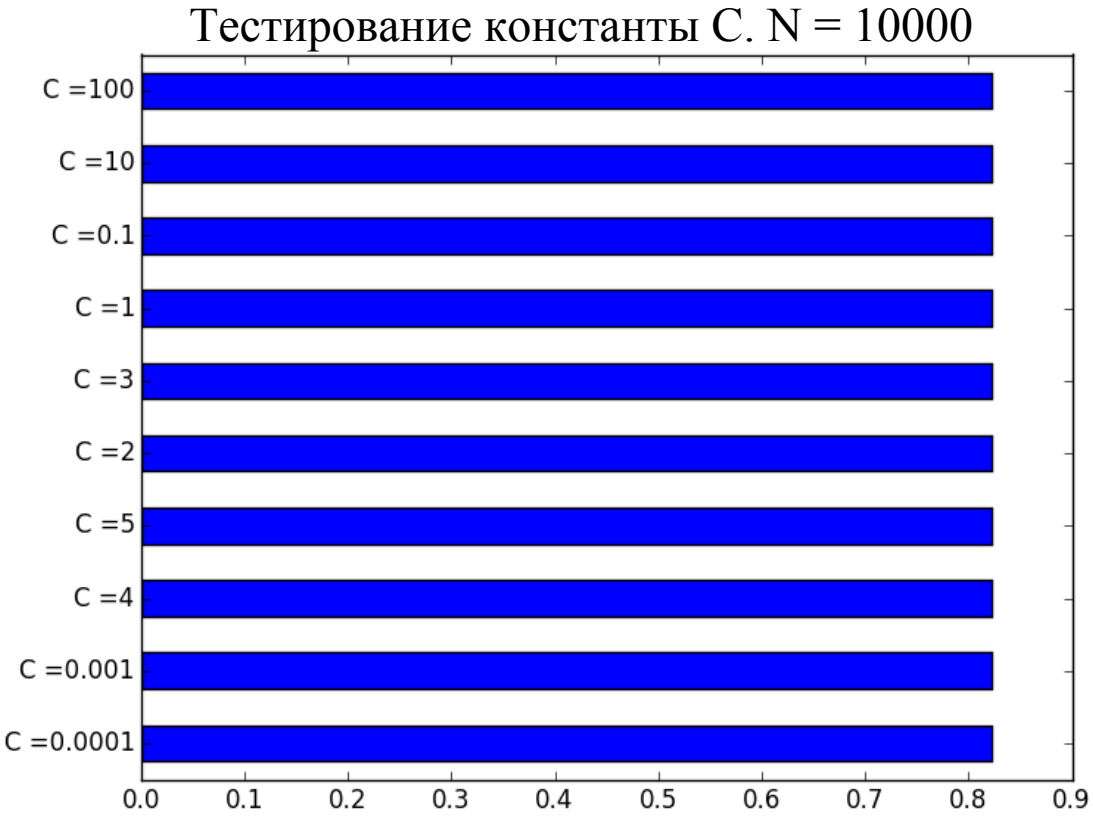
Тестирование



Тестирование

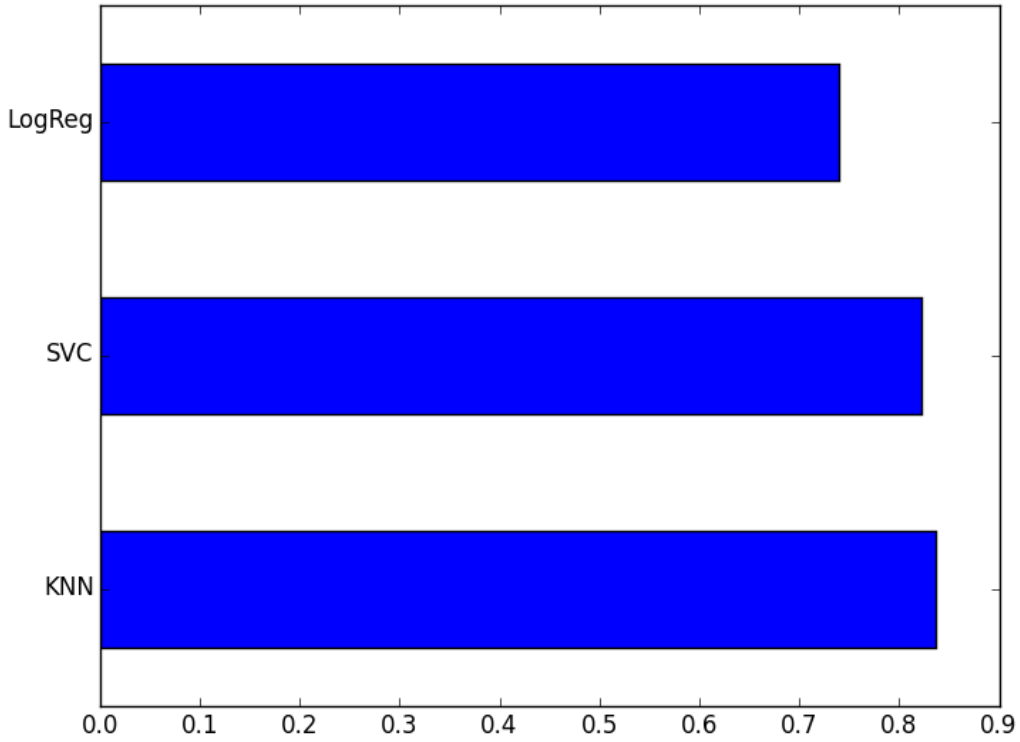


Тестирование



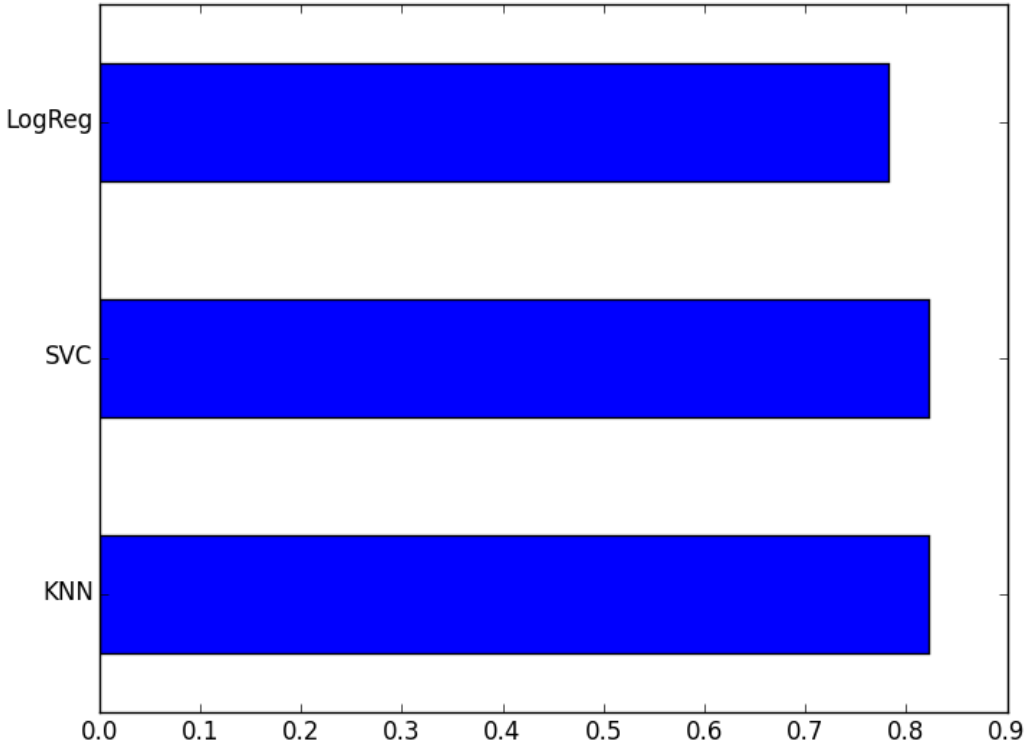
Тестирование

Сравнительный тест итоговых алгоритмов. N = 100



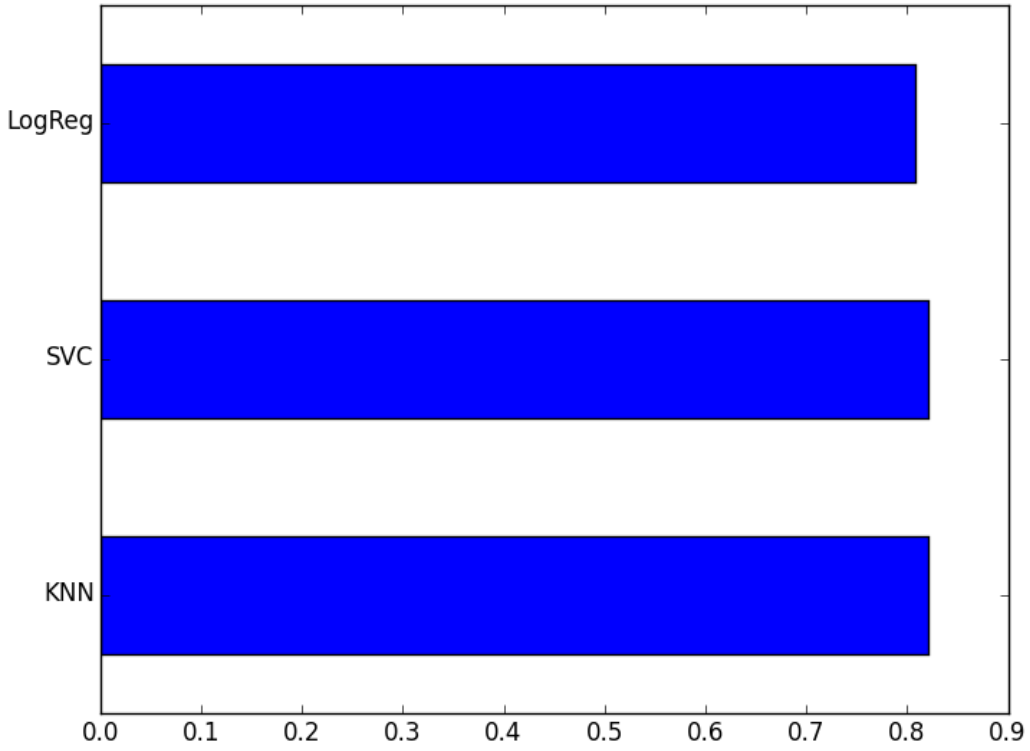
Тестирование

Сравнительный тест итоговых алгоритмов. N = 1000



Тестирование

Сравнительный тест итоговых алгоритмов. N = 10000



Тестирование

Тестирование на целевом наборе

N	KNN	SVC	LogReg
100	78.8	75.8	63.6
1000	72.7	75.6	72.7
10000	**81.8**	75.6	75.6

Заключение

В ходе работы были написаны методы для чтения и обработки wav файлов, хранения полученных данных, построения рекомендательной модели на языке Python с использованием библиотек NumPy и Sklearn.

Список литературы

1. http://www.machinelearning.ru/
2. http://scikit-learn.org/
3. http://www.numpy.org/
4. https://en.wikipedia.org/wiki/Kurtosis
5. https://en.wikipedia.org/wiki/Skewness
6. https://en.wikipedia.org/wiki/FFT