Санкт-Петербургский Государственный Университет
Математико-механический факультет

Руденко Дмитрий Андреевич 244 группа

Рекомендательная система музыки

Курсовая работа

Санкт-Петербург
2015

Оглавление

Введение

В век интернет-технологий каждый человек имеет доступ к терабайтам аудиофайлов. И каждый хочет найти именно ту музыку, которая ему подходит. Но вручную перебирать тысячи песен очень трудоемко. Поэтому хотелось бы автоматизировать этот процесс. Но как? Уже существуют рекомендательные системы: например, last.fm или vk.com. Но они основаны на коллоборативной фильтрации. Я же сделал систему, которая строит рекомендательную модель именно на анализе самих ауйдиофайлов, которые пользователь отметил как понравившиеся или не понравившиеся.

1. Постановка задачи

Целью данной работы является реализация приложения, которое по готовому набору пользовательской музыки формировало бы систему, способную определять, понравится ли конректный аудифайл пользователю. Необходимо реализовать следующие компоненты:

- Чтение пользовательского набора wav файлов
- Обработка начальных данных
- Построение рекомендательной модели на основе собранных данных
- Выборка подходящих wav файлов нового набора

Дополнительные задачи: разобраться в инструментах Data Maining и FFT на языке Python

2. Обзор существующих систем

Существуют несколько систем по рекомендации музыки: например, last.fm, vk.com, spotify. Имея неполный список предпочтений пользователя, их рекомендательные системы предсказывают, какая музыка понравится ему. Кроме того, существует дочернее предприятие Spotify The Echo Nest, которое предоставляет пользователям по всему миру огромную базу данных по анализу музыки. Но опять таки, в этой базе может не содержаться тех аудиофайлов, которые именно пользователь выбрал.

3. Решение задачи

Для работы был выбран язык Python, так как на нем реализованы библиотеки NumPy и Sklearn, про которые будет рассказано далее.

3.1. 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.

3.2. Обработка начальных данных

Анализируемый файл представляется в виде вектора из семплов, который делится на N снимков, каждый из которых приводится в АЧХ при помощи FFT-преобразования. То есть, из временного ряда мы получаем набор амплитут разных частот. FFT преобразоание выполняется с помощью библиотеки NumPy. Далее, для каждого такого снимка считаются следующие статистические величины: среднее значение, медиана, стандартное отклонение, скос (skewness) и крутизна (kurtosis). Соответственно, для каждой композиции после вычислений мы будем иметь 5 * N величин, которые формируют точки в 5N-мерном пространстве.

3.3. Построение рекомендательной модели

Далее нам требуется, чтобы система умела определять,к какому из классов (”true” или ”false”) принадлежит новая 5N - мерная точка. Это задача Data Maining: у нас имеется матрица ”object - features” и имеется вектор ответов(”1” или ”0”) для каждого из объектов, и по этим данным требуется построить функцию, максимально точно определяющую класс целевого объекта. Изначально была простая идея обобщающего прямоугольника: по каждой из координат у всех понравившихся пользователю композиций выбираются максимум и минимум, а далее проверяем, попадает ли каждая из координат у целевой композиции в

промежуток между максимом и минимумом. Минус очевиден: какая-либо композиция, сильно отличающаяся от всех остальных(так называемый "выброс"), очень сильно снижает точность определения класса. Поэтому дальнейший упор был сделан на алгоритмы Data Maining, которые реализованы в библиотеке Sklearn. Для исследования были выбраны следующие алгоритмы: K-means, Linear regression и Support vector clustering("SVC").

Задана обучающая выборка пар «объект-ответ» $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ $a(u)$ - искомая функция

- K-means (k ближайших соседей):Задана метрика $\rho(x, x')$. Для произвольного объекта u расположим объекты обучающей выборки x_i в порядке возрастания расстояний до u: $\rho(u, x_{1;u}) \leq \rho(u, x_{2;u}) \leq \cdots \leq \rho(u, x_{m;u})$, где через $x_{i;u}$ обозначается тот объект обучающей выборки, который является i-м соседом объекта u. Аналогичное обозначение введём и для ответа на i-м соседе: $y_{i;u}$. Т.о. $a(u) = \arg\max_{y \in Y} \sum_{i=1}^{m} \big[y(x_{i;u}) = y\big] w(i, u)$, где $w(i, u) = [i \leq k]$.

- Linear regression(линейная регрессия): $a(x) = \text{sign}(\sum_{j=1}^{n} w_j f_j(x) - w_0)$, где w_j - вес j- ого признака, w_0 - порог принятия решения, $w = (w_1, \ldots, w_n)$ - вектор весов. Поиск вектора весов по выборке X^m и есть задача алгоритма. В linear regression она решается следующим образом:$\left\| (\sum_{j=1}^{n} w_j f_j(x)) - w_0 - y \right\|^2 \to \min_w$

- Logistic regression (логистическая регрессия): $a(x) = \text{sign}(\sum_{j=1}^{n} w_j f_j(x) - w_0)$, где w_j - вес j- ого признака, w_0 - порог принятия решения, $w = (w_1, \ldots, w_n)$ - вектор весов. Поиск вектора весов по выборке X^m и есть задача алгоритма. В logistic regression она решается следующим образом: $\sum_{i=1}^{m} \ln(1 + \exp(-y_i \langle x_i, w)) \to \min_w$.

- Support vector clustering(метод опорных векторов): Постановка задачи аналогична logistic regression, отличается лишь задача нахождения оптимального вектора весов w и порога вхождения w_0:

$\sum_{i=1}^{m}(1-M_i(x,w_0))_+ + \frac{1}{2C}(\|w\|)^2 \to \min_{w;w_0}$, где $M_i(x,w_0) = y_i(\langle x_i, w\rangle - w_0)$ - отступ объекта x

3.4. Выборка подходящих wav файлов 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

4. Тестирование моделей

Для тестирование моделей был взят начальный набор, содержащий 73 wav файла и целевой набор, содержащий 33 wav файла. Тестирование проводилось с помощью перекрестной проверки(cross validation), которая также реализована в библиотеке Sklearn. Так как перекрестная проверка основывается только на начальном наборе данных, было дополнительно проведено тестирование работы алгоритмов на целевом наборе данных. Здесь и далее N - количество снимков с аудиофайла.

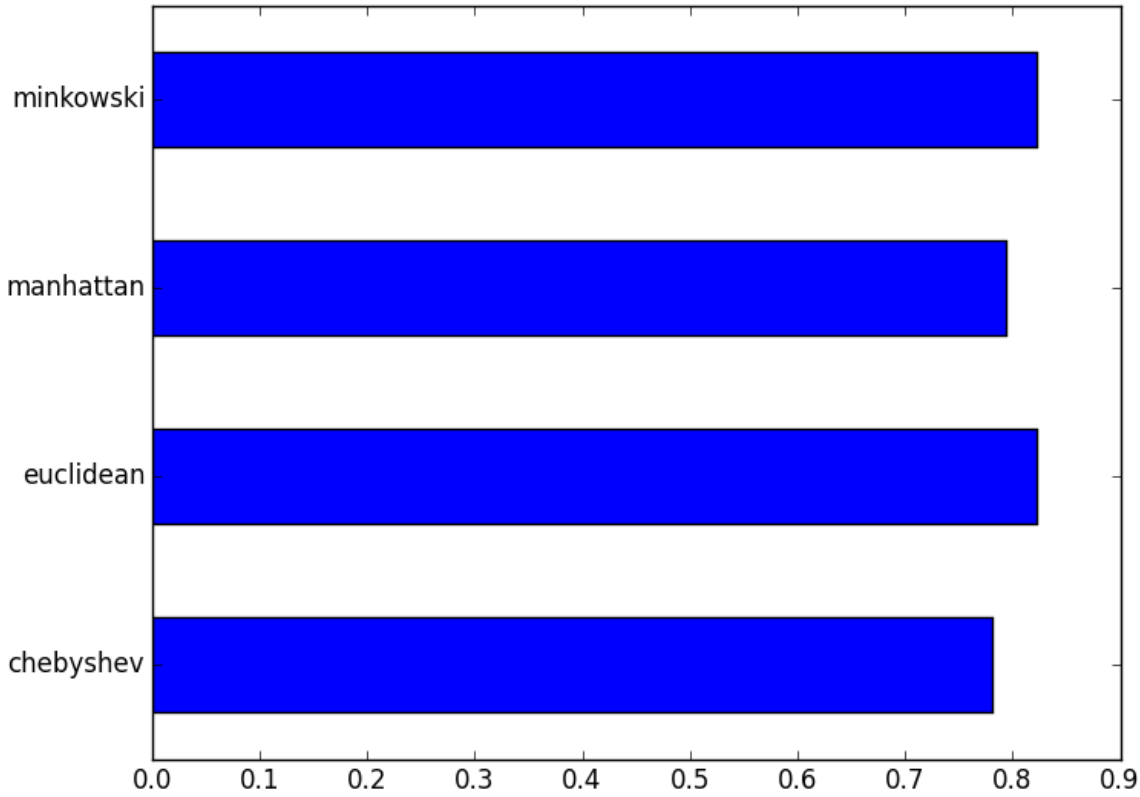

Рис. 1: Тестирование метрик KNN. N = 100

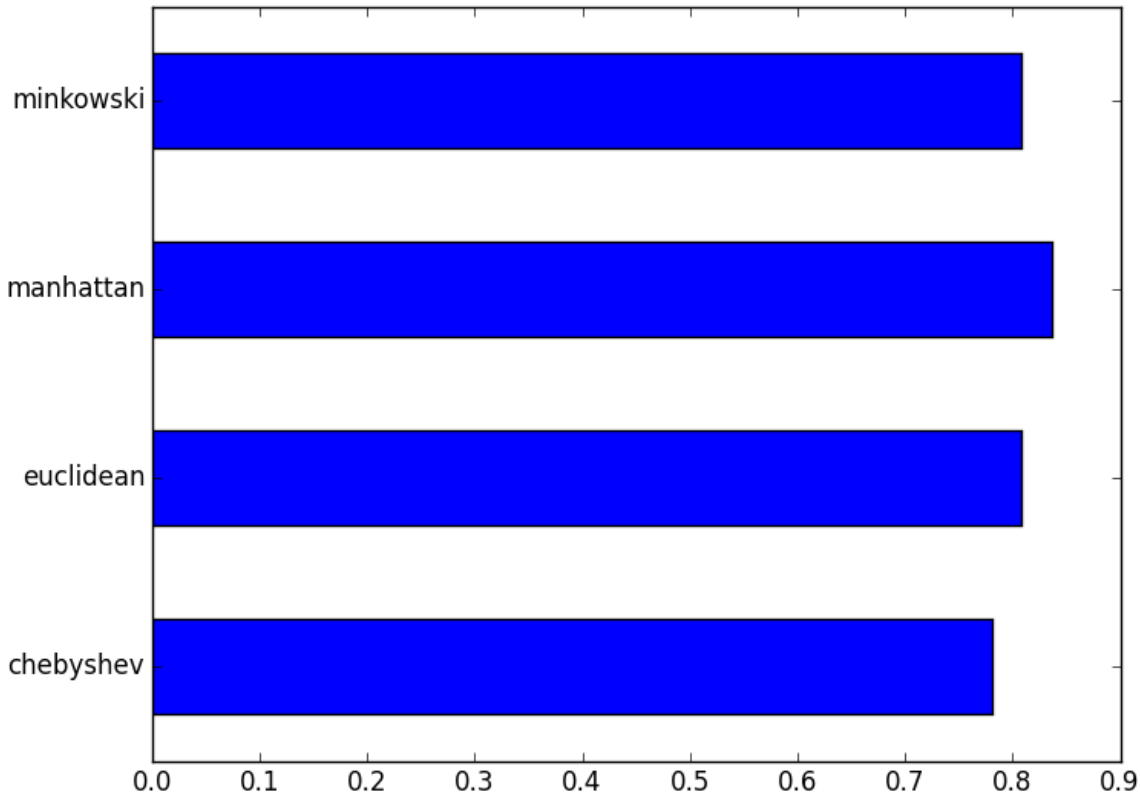

Рис. 2: Тестирование метрик KNN. N = 1000

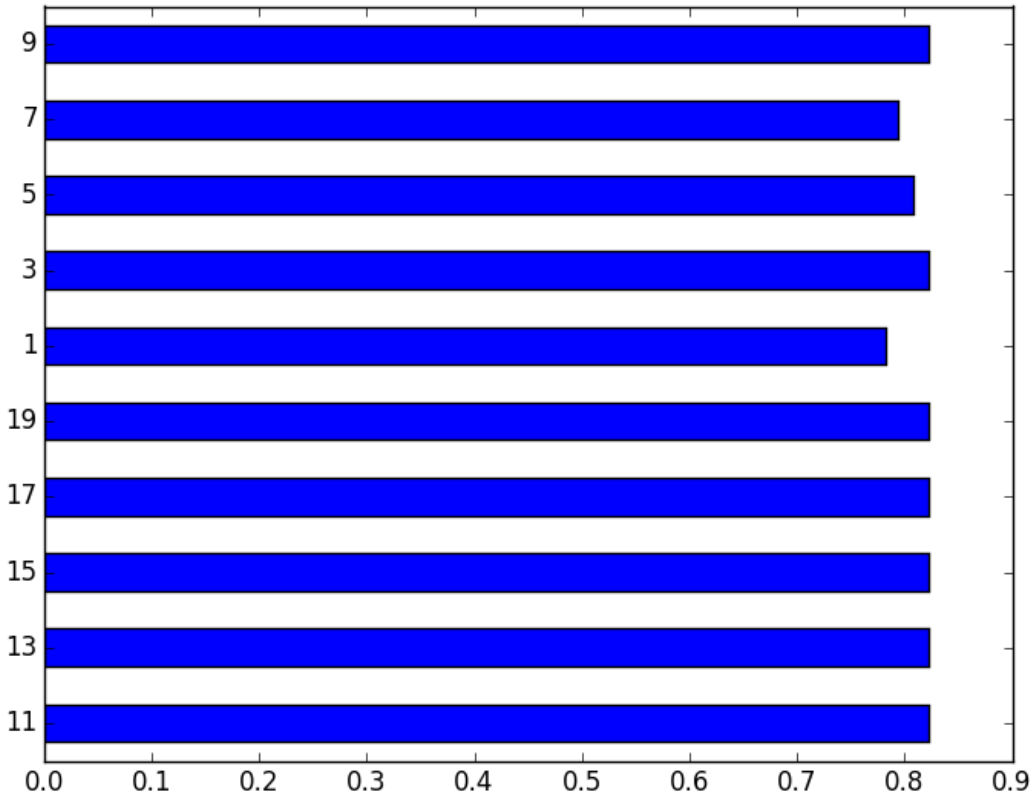

Рис. 3: Тестирование количества соседей KNN. N = 100

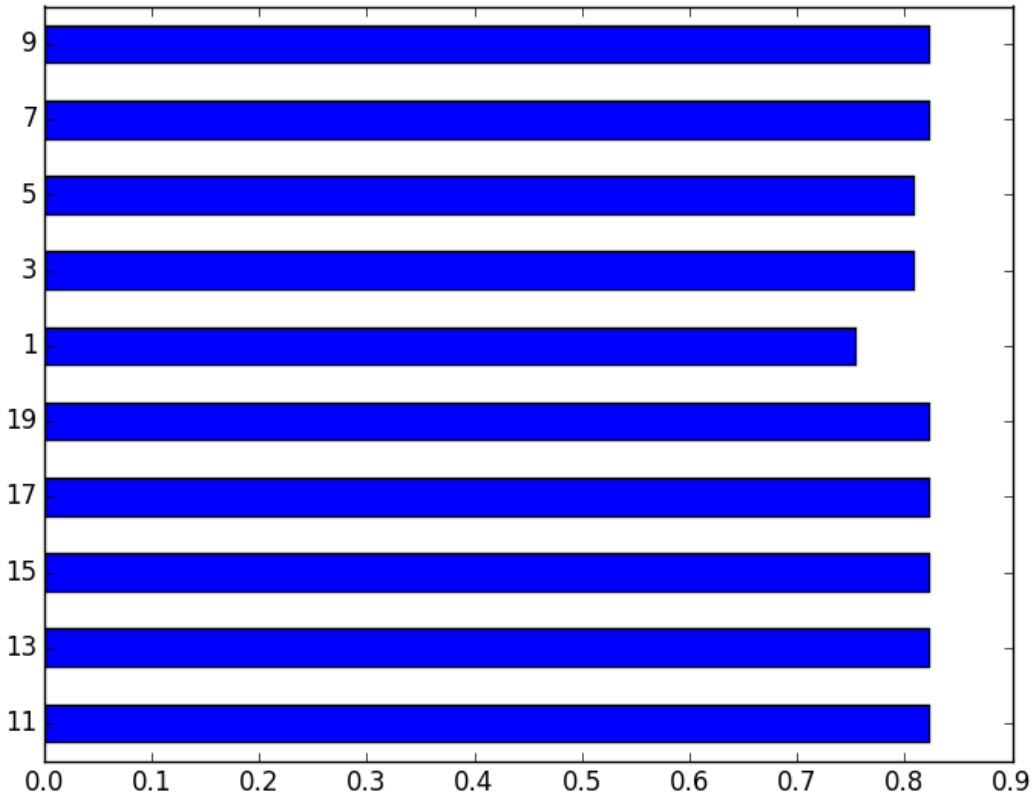

Рис. 4: Тестирование количества соседей KNN. N = 1000

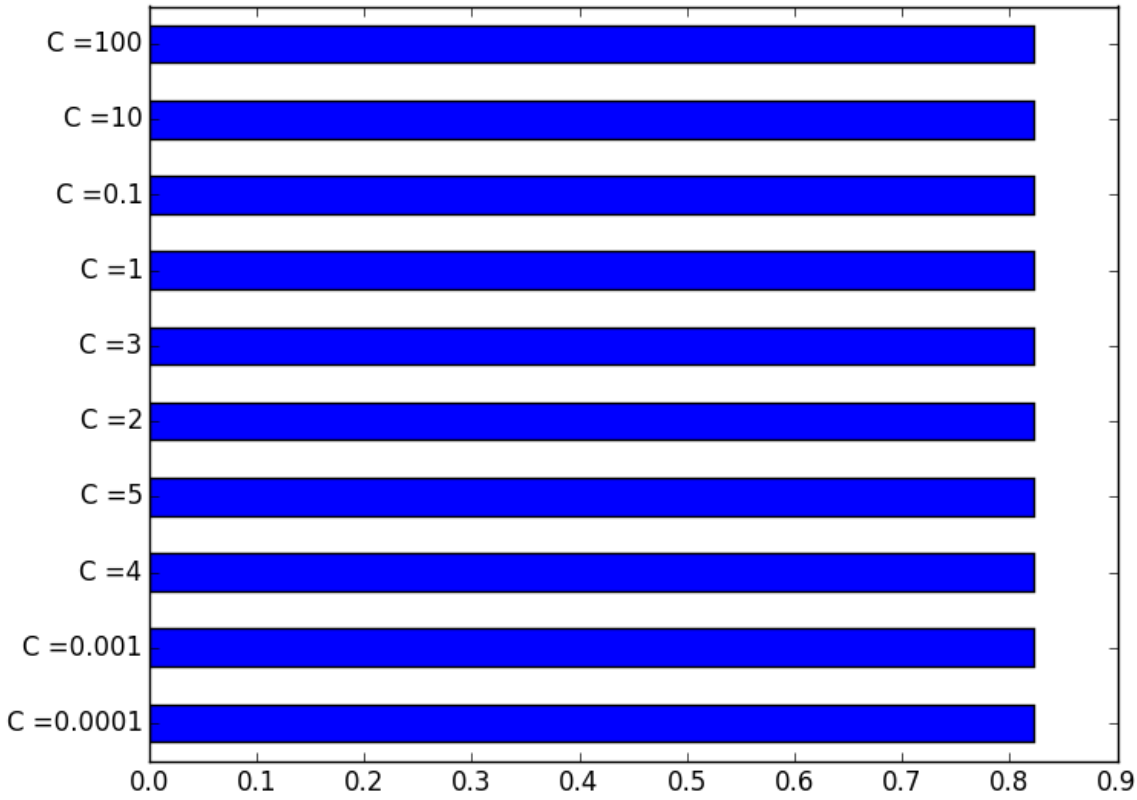

Рис. 5: Тестирование константы C SVC. N = 100

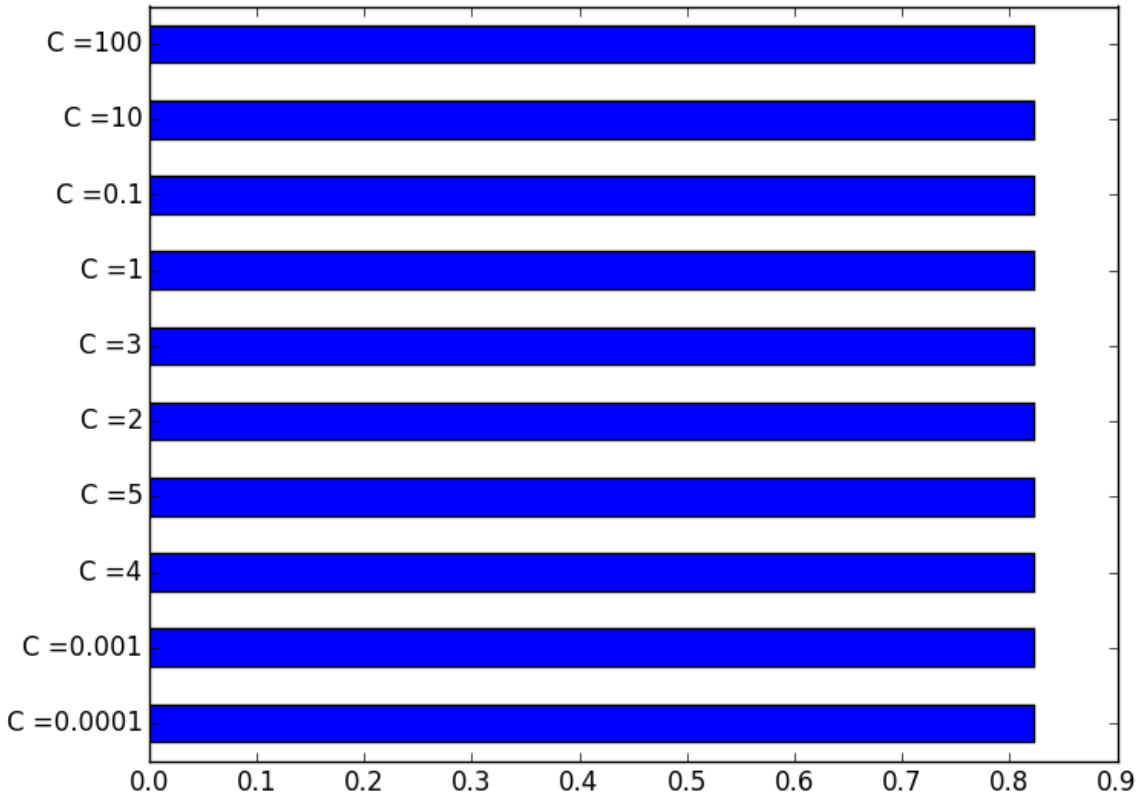

Рис. 6: Тестирование константы C SVC. N = 1000

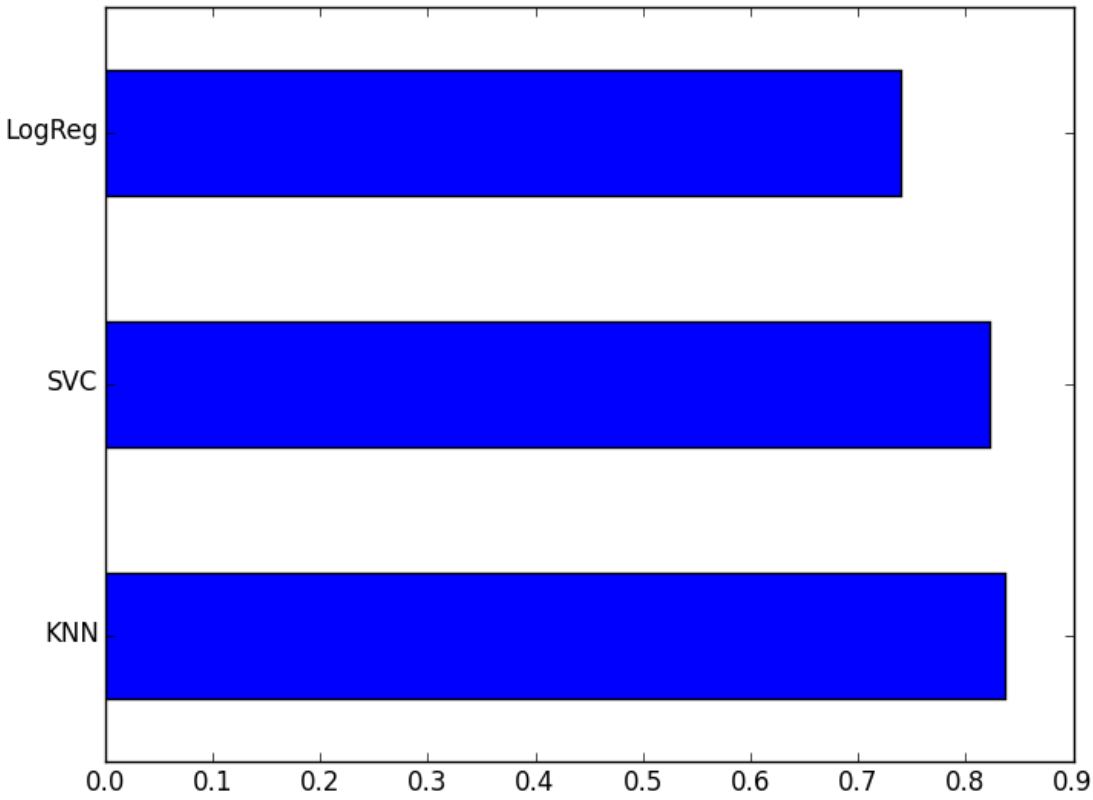

Рис. 7: Сравнительный тест итоговых алгоритмов. N = 100

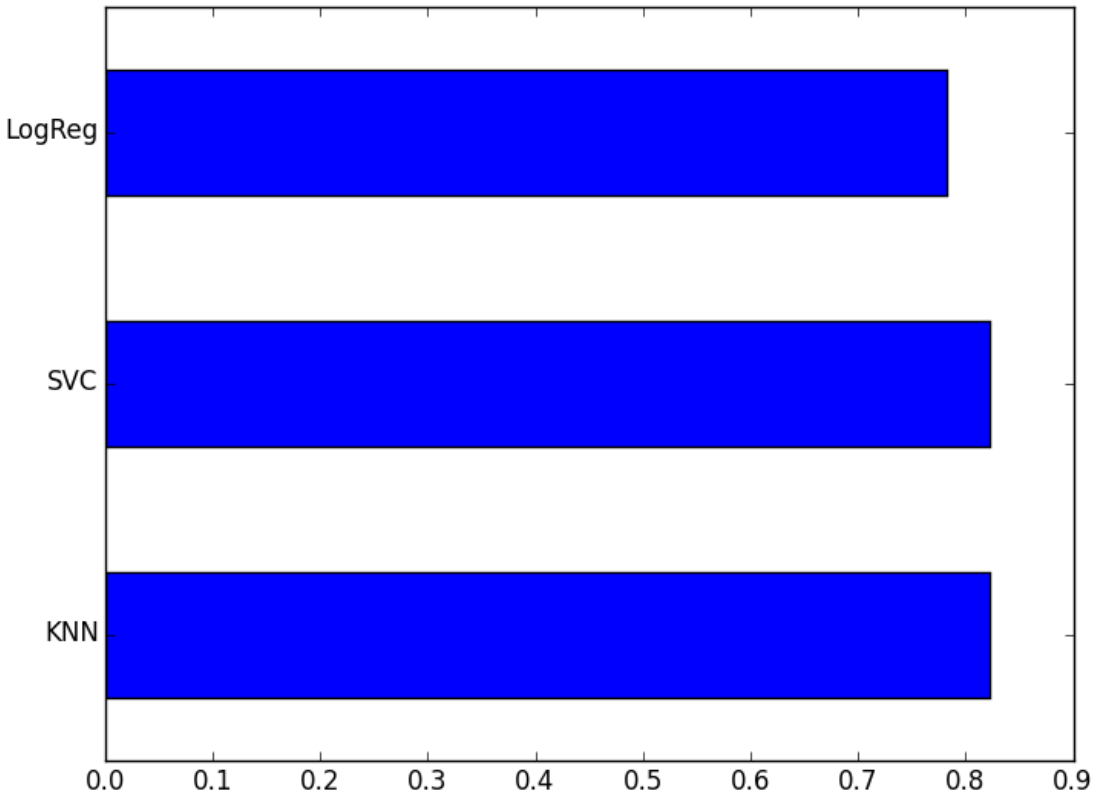

Рис. 8: Сравнительный тест итоговых алгоритмов. N = 1000

N	SVC	KNN	LogReg
100	75.8	78.8	63.6
1000	75.6	72.7	72.7

Рис. 9: Тестирование на целевом наборе

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