Федеральное государственное бюджетное образовательное учреждение
высшего образования «Санкт-Петербургский государственный
университет»
Математико-механический факультет

Кафедра информационно-аналитических систем

Руденко Дмитрий Андреевич

Рекомендательная система музыки

Курсовая работа

Научный руководитель:
доцент Графеева Н.Г.

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 коллаборативной фильтрации. Я же сделал систему, которая строит рекомендательную модель именно на анализе самих аудиофайлов, 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 конкретный аудиофайл пользователю. Необходимо реализовать следующие компоненты:

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

Дополнительные задачи: разобраться в инструментах Machine Learning и FFT на языке Python.

2. Обзор существующих систем

Существуют несколько систем по рекомендации музыки: например, last.fm, vk.com, spotify[2]. 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 временного ряда, 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 преобразование 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 Machine Learning[4]: у нас имеется матрица ”object - features”, и имеется вектор ответов (”-1” или ”1”) 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 Machine Learning, которые реализованы в библиотеке Sklearn[3]. Для исследования были выбраны следующие алгоритмы: KNeighbors Classification (”KNC”), Logistic regression, Support vector classification (”SVC”) и Gradient Boosting Classification (”GBC”).

Задана обучающая выборка пар «объект-ответ»

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

$a(x)$ - искомая функция

- KNeighbors Classification (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: $y_{i;u}$. Таким образом,

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

  $$w(i, u) = [i \leq k]$$

  .

- Logistic regression (логистическая регрессия):

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

  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:

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

  .

- Support vector classification (метод опорных векторов): Основная идея метода — перевод исходных векторов(точек) в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором(расстоянием от гиперплоскости до ближайшейшей точки) в этом пространстве. Постановка задачи аналогична logistic regression, отличается лишь задача нахождения оптимального вектора весов w и порога вхождения w_0:

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

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

Если выборка линейно неразделима(нельзя построить гиперплоскость, которая не проходила бы ни через одну точку выборки), то скалярное произведение в приведённой выше формуле заменяется нелинейной функцией ядра (скалярным произведением в пространстве с большей размерностью). В этом пространстве уже может существовать оптимальная разделяющая гиперплоскость. Библиотека Scikit-learn предоставляет следущие стандартные функции ядра ($K(x, x')$): "linear" - $\langle x, x' \rangle$; "polynomial" - $(\gamma\langle x, x' \rangle + r)^d$; "rbf" - $\exp(-\gamma|x - x'|^2)$; "sigmoid" - $\tanh(\gamma\langle x, x' \rangle + r)$ (r, d, γ - определяются пользователем).

- Gradient boosting classification[1] (градиентный бустинг): Суть бустинга - построения композиции алгоритмов Mashine Learning, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов. Наша цель - построить итоговую модель в виде

$$a(x) = \sum_{m=1}^{T} \gamma_m a_m(x)$$

, где $a_m(x)$ - базовые классификаторы. Вид базовых классификаторов определяется пользователем. Алгоритм градиентного бустинга схож с алгоритмом градиентного спуска(откуда и пошло его название): на каждой итерации алгоритм строит новую модель, максимально аппроксимирующую антиградиент функции потерь, и добавляет её к результирующей модели. Количество итераций равно количеству базовых классификаторов(определяется пользователем). Формально:

$$a_m = \arg\min_a \sum_{j=1}^{N} \left(\frac{\partial L(y_j, a_{m-1}(x_j))}{\partial a_{m-1}(x_j)} - a(x_j)\right)^2$$

$$\gamma_m = \arg\min_\gamma \sum_{j=1}^{N} L(y_j, a_{m-1}(x_j) + \gamma * a_m(x_j))$$

Здесь $L(y, a(x))$ - функция ошибки(определяется пользователем). В библиотеке Scikit-learn GBC в качестве базовых классификаторов использует дерево решений, а в качестве стандартных функций ошибок предлагает пользователю "deviance" - $log(1 + exp(-2ya(x)))$ или "exponential" - $exp(-ya(x))$

3.4. Выборка подходящих аудиофайлов 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 ("1"), 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. В данной работе представлены результаты проверки только тех параметров, которые сильно влияют в данном контексте на результат. Для остальных параметров, которые влияют на результат малозначительно, оставлены стандартные показатели.

Здесь и далее N - количество снимков с аудиофайла.

4.1. Тестирование количества соседей KNC

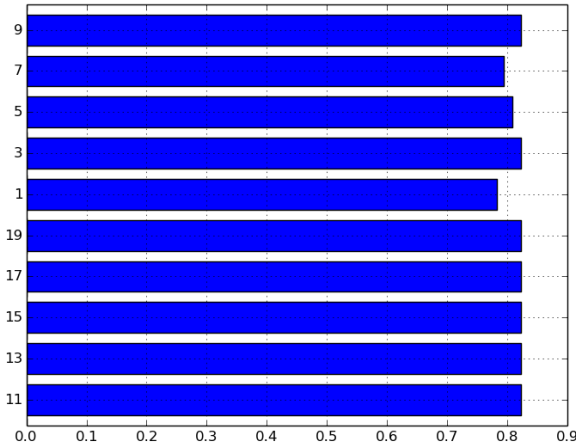

Рис. 1: N = 100

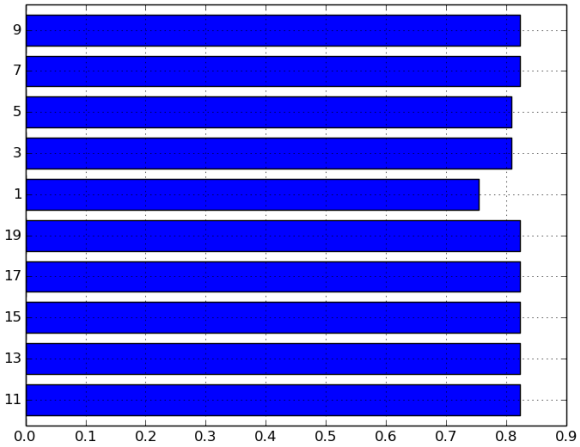

Рис. 2: N = 1000

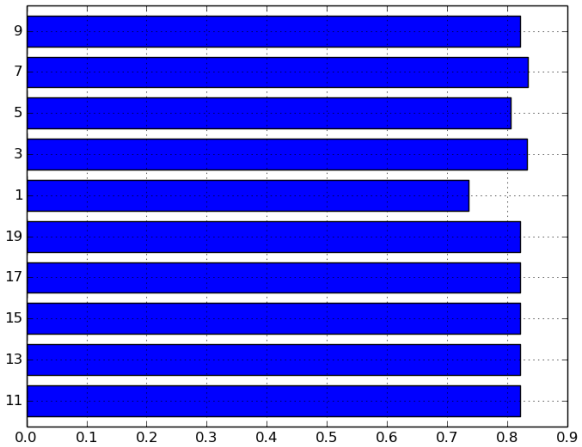

Рис. 3: N = 10000

4.2. Тестирование метрик KNC

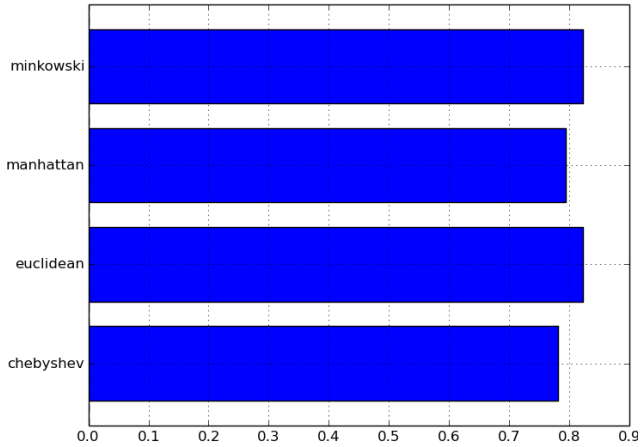

Рис. 4: N = 100

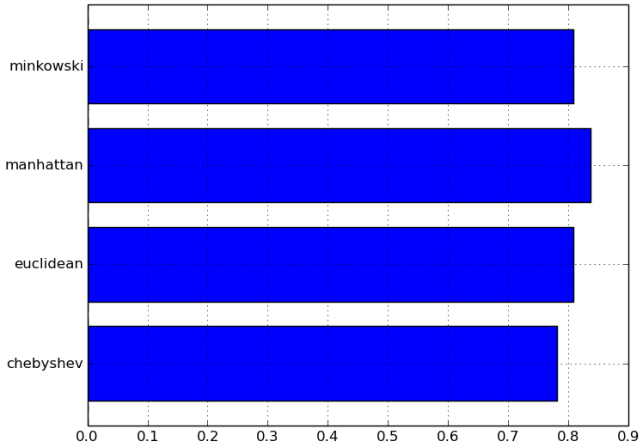

Рис. 5: N = 1000

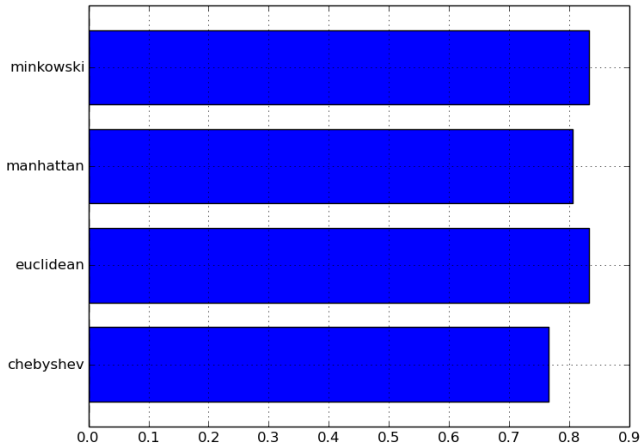

Рис. 6: N = 10000

4.3. Тестирование ядер SVC

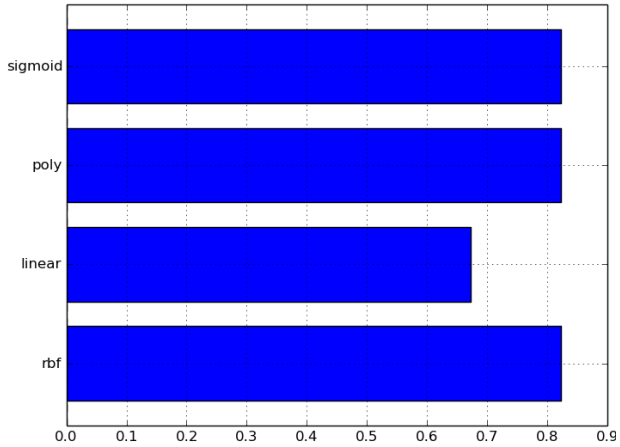

Рис. 7: N = 100

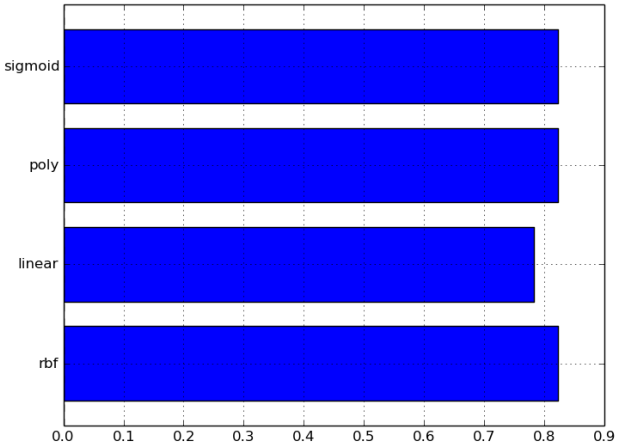

Рис. 8: N = 1000

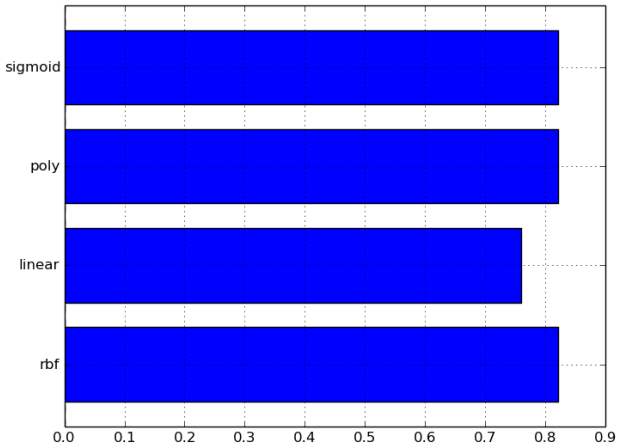

Рис. 9: N = 10000

4.4. Тестирование количества базовых классификаторов GBC

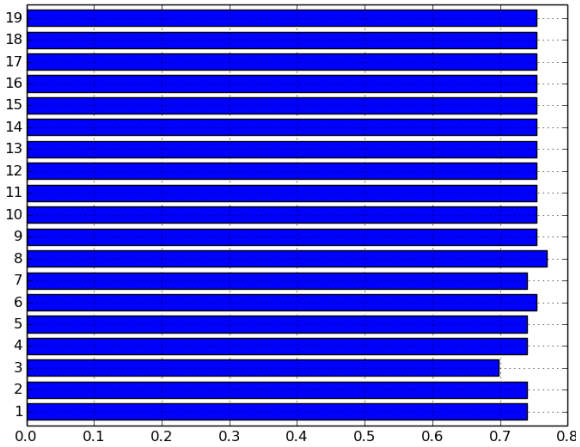

Рис. 10: N = 100

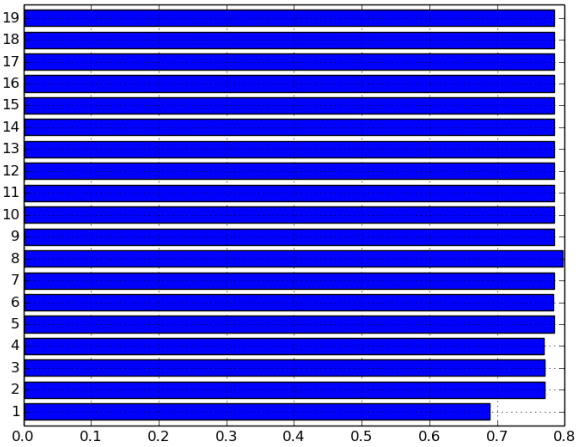

Рис. 11: N = 1000

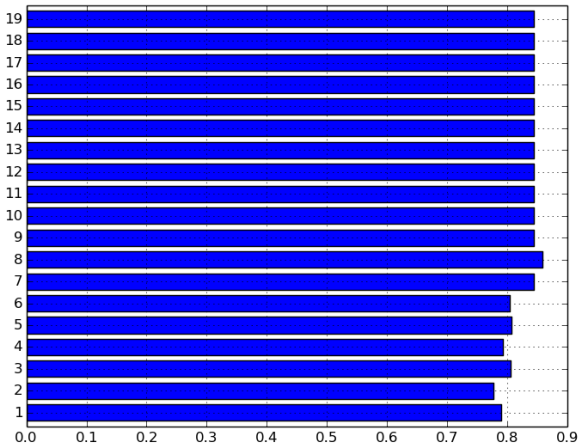

Рис. 12: N = 10000

4.5. Тестирование функции ошибок GBC

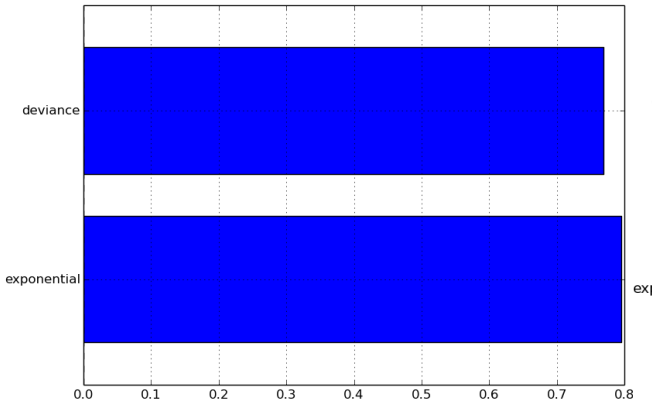

Рис. 13: N = 100

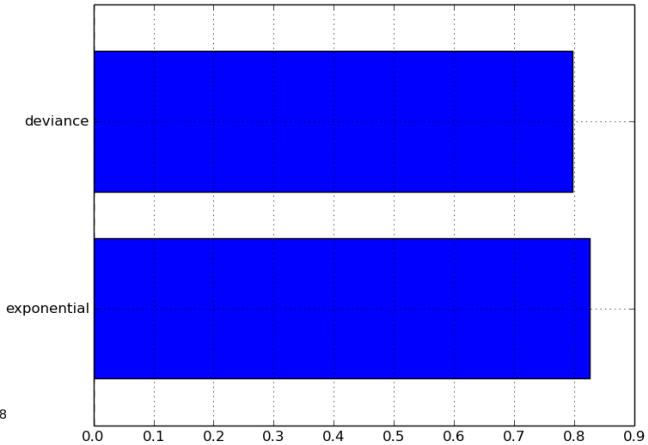

Рис. 14: N = 1000

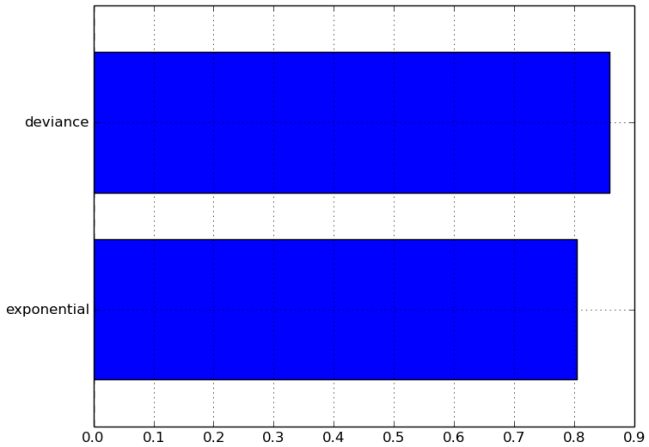

Рис. 15: N = 10000

4.6. Сравнительный тест алгоритмов

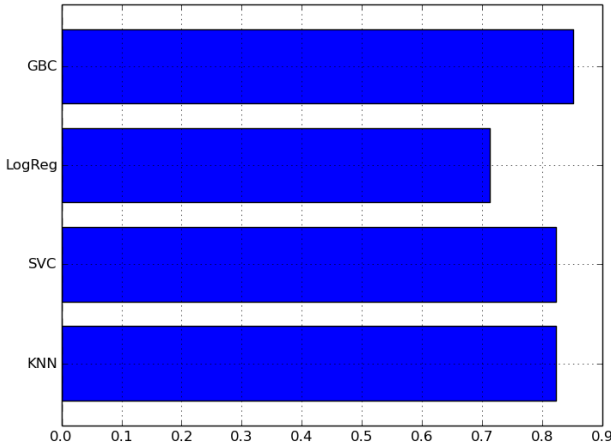

Рис. 16: N = 100

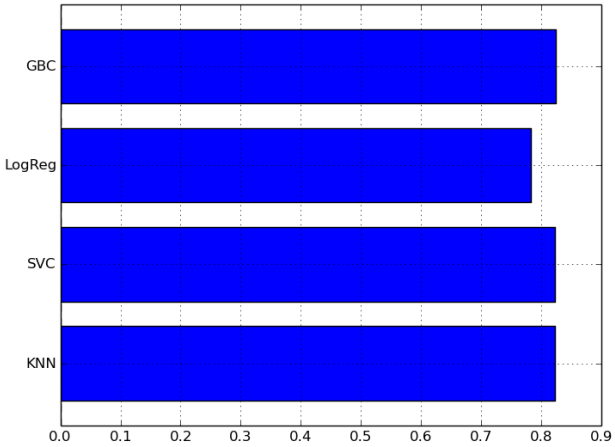

Рис. 17: N = 1000

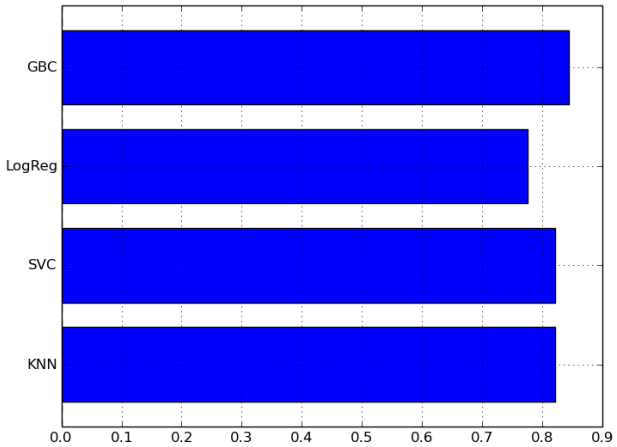

Рис. 18: N = 10000

4.7. Тестирование на целевом наборе

N	SVC	KNC	LogReg	GBC
100	75.8	78.8	63.6	72.7
1000	75.6	72.7	72.7	72.7
10000	75.6	81.8	75.6	84.8

Заключение

В ходе работы было реализовано приложение, которое по готовому набору пользовательской музыки формирует систему, способную определять, понравится ли конкретный аудиофайл пользователю. Написаны методы для чтения и обработки аудиофайлов, построения рекомендательной модели, выборки подходящих аудиофайлов из нового набора на языке Python с использованием библиотек NumPy и Sklearn.

Список литературы

[1] Friedman Jerome H. Greedy function approximation: a gradient boosting machine // Annals of statistics. — 2001. — P. 1189–1232.

[2] Johnson Chris. Algorithmic music discovery at Spotify // LinkedIn Corporation. — 2015. — URL: `http://www.slideshare.net/MrChrisJohnson/algorithmic-music-recommendations-at-spotify/` (online; accessed: 10.10.2015).

[3] Scikit-learn: Machine Learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort et al. // Journal of Machine Learning Research. — 2011. — Vol. 12. — P. 2825–2830.

[4] К.В Воронцов. Курс лекций по машинному обучению // ШАД Яндекс. — 2015. — URL: `http://www.machinelearning.ru/` (дата обращения: 12.11.2015).