МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка рекомендательной системы музыкальных произведений с использованием машинного обучения»

Студент В.Д. Пантилеев
(И.О. Фамилия) (личная подпись)

Руководитель Доцент, к. тех. н., В.С. Климов
(ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Название бакалаврской работы – Разработка рекомендательной системы музыкальных произведений с использованием машинного обучения.

Целью бакалаврской работы является реализация персонализированной рекомендательной системы, где пользователь на основе своего списка понравившихся музыкальных треков может получить список рекомендаций.

Объектом исследования является алгоритм рекомендательной системы, основанной на содержимом предметов рекомендаций.

В первой главе выявляются характеристики персонализированных рекомендательных систем. Так же был проведен анализ некоторых алгоритмов персонализированных рекомендательных систем, представлено описание предметов рекомендаций и выделены требования к рекомендательной системе.

Во второй главе описываются методы предварительной обработки данных. После чего, представлены алгоритмы машинного обучения, используемые в рекомендательной системе.

Третья глава посвящена реализации и тестированию рекомендательной системы. Были описаны технологии, используемые при разработке и представлен программный код рекомендательной системы. Тестирование производительности разработанного алгоритма производилось при разных входных условиях – сравнивалась производительность при разном количестве кластеров кластеризации и при различных метриках предварительной обработки данных. Для тестирования достоверности рекомендаций использовались метрики полноты, точности и F-меры.

Abstract

The title of the graduation work is *Development of recommender system for musical compositions using machine learning*.

This graduation work is devoted to implementing the personalized recommender system, with the help of which a user can get a list of recommendations based on the list of their favourite music tracks.

When working on the research, the following objectives have to be attained:

– to select the method of making recommendations,

– to determine a set of the music tracks,

– to describe the preliminary data processing,

– to construct a clustering model,

– to implement a nucleus of the recommender system based on cosine similarity and KNN algorithm,

– to test the recommender system.

The graduation work may be divided into several logically connected parts, dedicated to analyzing the subject and design, as well as developing and testing the recommender system of musical compositions.

The first chapter presents an analysis of the existing methods for making recommendations of the personalized recommender systems. This chapter also dwells on the music tracks data and sets the objective to develop a recommender system.

The second chapter represents the methods of preliminary data processing. To solve the problem of clustering the musical compositions, the machine learning algorithm K Means is chosen. We also discuss the method for making recommendations based on the KNN algorithm and cosine similarity.

The third chapter features the selected development tools and the programme code of the solution.

Performance testing is carried out under different initial conditions. Then, we select the most optimal ones for the algorithm to work.

The accuracy of the algorithm shows than the developed recommender system offers music tracks that correspond to the user interests.

The developed recommender system has to be used in the server part of the application because the calculations can load the device.

Оглавление

Введение 6

Глава 1 Анализ состояния вопроса 9

1.1 Характеристики рекомендательных систем 9

1.2 Анализ алгоритмов рекомендательных систем 10

1.3 Анализ данных музыкальных произведений и постановка требований к рекомендательной системе 16

Глава 2 Проектирование рекомендательной системы музыкальных произведений 20

2.1 Масштабирование данных музыкальных произведений 20

2.2 Использование алгоритма K-Means для кластеризации музыкальных произведений 22

2.3 Алгоритм KNN для нахождения сходства между предметами рекомендаций 27

2.4 Алгоритм рекомендательной системы 30

Глава 3 Реализация рекомендательной системы музыкальных произведений 33

3.1 Выбор программных средств 33

3.2 Программная реализация рекомендательной системы 34

3.3 Тестирование 40

Заключение 47

Список используемой литературы и список используемых источников 48

Введение

Рекомендательные системы – это системы машинного обучения, которые помогают пользователям находить новые продукты и услуги. Вместо того, чтобы предлагать потенциальному клиенту или пользователю опыт, в котором он сам ищет продукт или интересующую услугу, рекомендательные системы определяют рекомендации на основе других объектов или пользователей со схожими интересами. Каждый раз, когда пользователь делает покупки в интернете, система его направляет к более предпочтительному товару [3].

Сегодня рекомендательные системы являются одной из важных особенностей современного мира, так как пользователи часто перегружены выбором товаров и услуг и нуждаются в более удобном способе поиска того, что они ищут [3].

В сфере Data Science, рекомендательные системы являются одной из самых популярных тем на сегодняшний день. Они используются для предсказания рейтинга или предпочтения, которое пользователь дал бы предлагаемому элементу. Почти каждая крупная компания применяла рекомендательные системы в той или иной сфере. В простейшем виде, персонализированные рекомендательные системы представлены в виде ранжированных списков элементов. При выполнении этого ранжирования, рекомендательная система пытается предсказать, какие элементы наиболее важны для пользователя.

Элемент – это общий термин, используемый для описания того, что система рекомендует пользователям. Обычно рекомендательные системы фокусируют на конкретных типах элементов и, соответственно, его атрибутах, дизайне, графическом пользовательском интерфейсе, и основная методика рекомендаций, которая используется для генерации рекомендаций. Методика рекомендаций в каждом отдельном случае настраивается таким

образом, чтобы обеспечить максимальную полезность и эффективность предложения для конкретного типа элементов [3].

Данная бакалаврская работа направлена на реализацию рекомендательной системы музыкальных произведений с использованием алгоритмов машинного обучения.

Объектом исследования ВКР является алгоритм рекомендательной системы музыкальных произведений.

Предметом исследования ВКР являются персональные рекомендации музыкальных треков на основанные на интересах пользователя.

Целью работы является реализация персональной рекомендательной системы, где пользователь на основе его списка понравившихся элементов (музыкальных треков) может получить другие предпочтительные.

Для достижения поставленной цели необходимо:

– определить характеристики рекомендательной системы и провести анализ алгоритмов вынесения рекомендаций;
– выбрать алгоритм вынесения рекомендаций;
– выбрать средства проектирования и разработки;
– определить набор данных для рекомендательной системы;
– спроектировать рекомендательную систему;
– разработать программный модуль вынесения рекомендаций;
– оценить достоверность рекомендаций разработанного алгоритма.

В первой главе выполнялись следующие задачи:

– описаны характеристики и цели рекомендательной системы;
– описаны методы вынесения рекомендаций;
– произведен обзор входных данных музыкальных произведений;
– описаны требования к рекомендательной системе.

Во второй главе производилось:

– описание методов предварительной обработки данных;
– описание алгоритма кластеризации K Means для кластеризации музыкальных произведений и их жанров;

– описание алгоритма машинного обучения K-ближайших соседей для нахождения сходств с использованием метрики косинусного сходства.

В третьей главе:

– были представлены технологии для реализации рекомендательной системы;

– произведено подключение зависимостей и входных данных;

– представлен программный код рекомендательной системы;

– произведена оценка производительности и достоверности рекомендательной системы.

В ходе работы, была разработана рекомендательная система музыкальных произведений, реализующая персональные рекомендации, основанные на контенте (на атрибутах рекомендуемых элементов).

Разработанная рекомендательная система, позволит пользователю получать рекомендации на основе своих музыкальных предпочтений.

Глава 1 Анализ состояния вопроса

1.1 Характеристики рекомендательных систем

Задача рекомендательной системы – информировать конечного пользователя о интересующих его элементах в некоторый момент времени. В зависимости от цели и бизнес модели рекомендаций, рекомендательная система может быть, как основой какого-либо сервиса, так и его дополнением.

Все рекомендательные системы можно описать следующими характеристиками:

– предмет рекомендаций,
– цель рекомендаций,
– контекст рекомендаций,
– источник рекомендаций,
– степень персонализации,
– прозрачность,
– формат рекомендаций,
– алгоритм.

Предметом рекомендации называют рекомендуемые системой элементы. Целью рекомендации называют то, зачем пользователю предоставляются рекомендации (например, покупка или информирование). Контекст рекомендации – это то, что в данный момент времени пользователь делает (он может смотреть товары, слушать музыку или общаться с людьми). Источником рекомендаций называют то, что предоставляет рекомендации (другие пользователи, аудитория).

По степени персонализации рекомендательные системы бывают персонализированными и не персонализированными. Персонализированные рекомендательные системы используют доступную информацию о пользователе, например, историю его покупок для предоставления ему

рекомендаций. Более продвинутые варианты персонализированных рекомендательных систем могут использовать данные текущей сессии пользователя. Не персонализированные рекомендательные системы – это рекомендательные системы, которые хотя и могут учитывать время и регион пользователя, но предлагают ему тоже самое, что и другим.

Прозрачность рекомендательных систем может затрагивать тему надежности и репутации системы. Если пользователь понимает, как получил те или иные рекомендации, он больше доверяет им. Существуют рекомендательные системы, которые могут рекомендовать проплаченный товар. Такие рекомендации называют недобросовестными. Манипуляции с рекомендательными системами могут быть и непреднамеренными, например, когда рекомендательная система опирается на средний рейтинг определенного товара и этот рейтинг может быть накручен.

Форматом рекомендаций называют то, как конечный пользователь их получает. В интернет-магазинах часто можно увидеть рекомендации в виде списка товаров или в виде всплывающего окна.

1.2 Анализ алгоритмов рекомендательных систем

Несмотря на множество существующих алгоритмов, все они сводятся к нескольким базовым подходам, которые будут описаны далее. Классические алгоритмы машинного обучения в рекомендательных системах обычно подразделяются на:

– методы фильтрации, основанные на содержимом,
– методы совместной фильтрации,
– методы, основанные на матричном разложении.

Метод фильтрации, основанный на содержимом использует метрики схожести атрибутов элементов данных (рисунок 1). Определяя схожесть объектов рекомендаций, он сопоставляет их с имеющимися, относя их к

какому-либо классу. После чего, основываясь на некоторой характеристике, например, расстоянии до элемента, генерирует рекомендации.

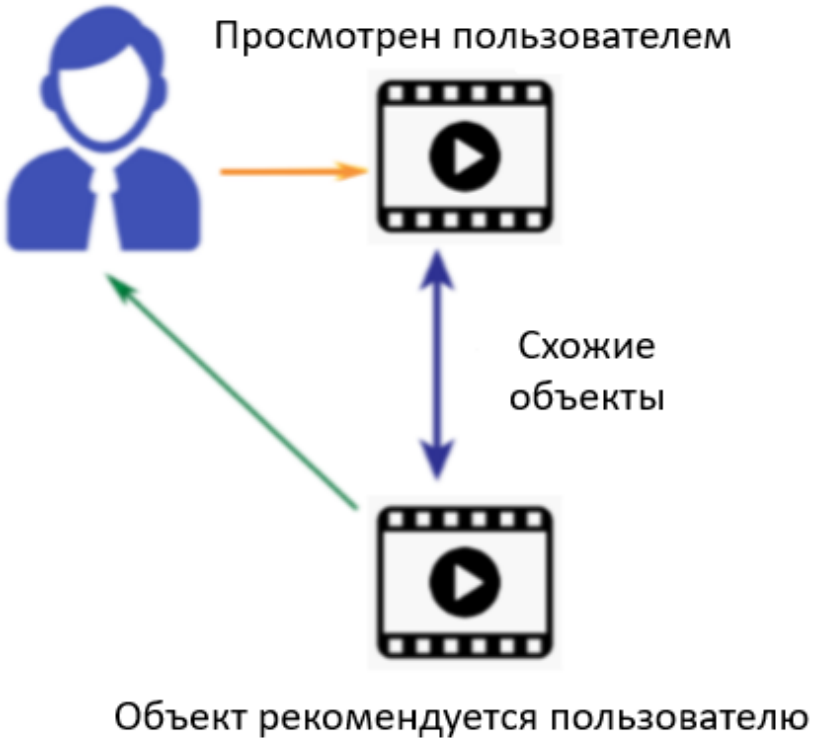

Рисунок 1 – Фильтрация, основанная на содержимом

Данный тип фильтрации не задействует других пользователей, кроме пользователя, которому представляются рекомендации. В зависимости от того, что нравится пользователю, алгоритм выберет элементы с похожим содержимым.

Сходством называют меру близости двух векторов атрибутов предметов рекомендаций. Чаще всего в качестве метрики сходства используют косинусное расстояние (1):

$$sim(A,B) = sim(\theta) = \frac{A*B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \quad (1)$$

где A и B – предметы рекомендаций, которые сравниваются между собой.

Второй способ найти схожесть двух векторов – это использовать корреляцию Пирсона (2):

$$sim(A,B) = sim(\theta) = \frac{\sum_i (A_i - \overline{A})(B_i - \overline{B})}{\sqrt{\sum_i (A_i - \overline{A})^2 \sum_i (B_i - \overline{B})^2}}, \quad (2)$$

где A и B – предметы рекомендаций, а $\overline{A}$ и $\overline{B}$ – их выборочное среднее (3).

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad (3)$$

где n – количество элементов, x – элемент.

Фильтрация, основанная на содержимом, предоставляет меньшее разнообразие элементов, чем другие алгоритмы, но отлично подходит, когда пользователь что-то просматривает или оценивает.

Одним из главных плюсов фильтрации, основанной на содержимом является стабильность. Это означает то, что рейтинги предметов рекомендаций не будут существенно меняться длительное время, в отличие от вкусов людей.

Недостатком этого метода является то, что, когда появляется новый пользователь с недостаточным количеством данных о его транзакциях, метод фильтрации, основанный на содержимом не может качественно делать рекомендации.

Этого недостатка лишен метод совместной фильтрации. Рекомендательный системы, основанные на совместной фильтрации, рассчитывают сходство на основе взаимодействий для рекомендаций (рисунок 2). Данный подход строит модель на основе прошлого поведения пользователя (ранее приобретенные или выбранные элементы, или присвоенные им оценки), а также аналогичных решений, принятых другими пользователями. Затем эта модель используется для прогнозирования элементов, которые могут заинтересовать пользователя, или рейтингов объектов [11].

Стандартный метод совместной фильтрации и фильтрации, основанной на содержимом известен как алгоритм K-ближайших соседей (KNN). Для каждого пользователя ищутся похожие на него в терминах предпочтений

другие пользователи или схожие с его интересами объекты, дополняется информация известными данными о соседях [17].

Рисунок 2 – Совместная фильтрация

У разных людей могут быть разные базовые показатели при выставлении оценок. Некоторые люди, как правило, дают высокие оценки в целом, некоторые довольно строгие, даже если они удовлетворены предметами. Чтобы избежать смещения, можно вычесть среднюю оценку каждого пользователя всех элементов при вычислении взвешенного среднего и добавить ее обратно для целевого пользователя. Для чего можно воспользоваться одной из метрик вычисления меры близости двух векторов – косинусным сходством (1) или корреляцией Пирсона (2).

К плюсам данного метода можно отнести разнообразие рекомендуемых элементов.

Однако, этот метод плохо справляется с разреженностью матриц, когда никто по соседству не оценил предмет именно так, как система пытается предсказать целевому пользователю. Кроме того, данный метод

малоэффективен с точки зрения вычислений при росте числа пользователей и предметов.

К методам, основанным на матричном разложении, можно отнести матричную факторизацию. Разреженность матриц является большой проблемой для совместной фильтрации. Для решения этой проблемы можно использовать матричное разложение. Задачей матричного разложения является аппроксимация значений ячеек матрицы, где в начальной таблице стоят нули.

Самые популярными способами разложения матрицы являются:

1. Метод главных компонент (Principal Component Analysis, PCA) – это метод уменьшения размерности матриц, который позволяет идентифицировать корреляции и закономерности в наборе данных, чтобы он мог быть преобразован в набор данных значительно меньшей размерности без какой-либо важной информации. При обнаружении сильной корреляции между различными переменными принимается окончательное решение об уменьшении размера данных таким образом, чтобы значимые данные сохранились. Для уменьшения размерности матрицы с помощью данного алгоритма нужно выполнить следующие шаги:

    1) произвести нормировку данных – масштабировать данные таким образом, чтобы их значения лежали водном диапазоне;

    2) вычислить ковариационную матрицу – вычислить матрицу, каждая запись которой представляет собой ковариацию соответствующих переменных;

    3) вычисление собственных векторов и собственных значений – вычисление математических конструкций из ковариационной матрицы;

    4) вычисление основных компонентов – формирование матрицы признаков, содержащей все значимые элементы данных, которые обладают максимальной информацией о данных;

5) уменьшение размера данных – умножение транспозиции исходного набора данных на транспозицию полученного вектора признаков.

2. Сингулярное разложение (Singular Values Decomposition, SVD) является удобным методом при работе с матрицами. Сингулярное разложение показывает геометрическую структуру матрицы, и позволяет наглядно представить имеющиеся данные. Сингулярное разложение используется при решении самых разных задач – от приближения методом наименьших квадратов и решения систем уравнений до сжатия распознаваемых изображений. Используются разные свойства сингулярного разложения, например, показывать ранг матрицы и приближать матрицы данного ранга. Общая формула сингулярного разложения представлена ниже:

$$M = U\Sigma V^t, \quad (4)$$

где M – исходная матрица; U – левая сингулярная матрица (столбцы левые сингулярные векторы); Σ – диагональная матрица, содержащая сингулярные (собственные) значения; V – правильная сингулярная матрица (столбцы – правильные сингулярные векторы).

3. Неотрицательное разложение матрицы (Non-Negative Matrix Factorization, NNMF) – алгоритм разложения исходной матрицы на другие малые матрицы, которые не имеют отрицательных элементов, что облегчает проверку полученных матриц.

Эти методы разлагают исходную разреженную матрицу на малоразмерные матрицы со скрытыми факторами и меньшей разреженностью (рисунок 3).

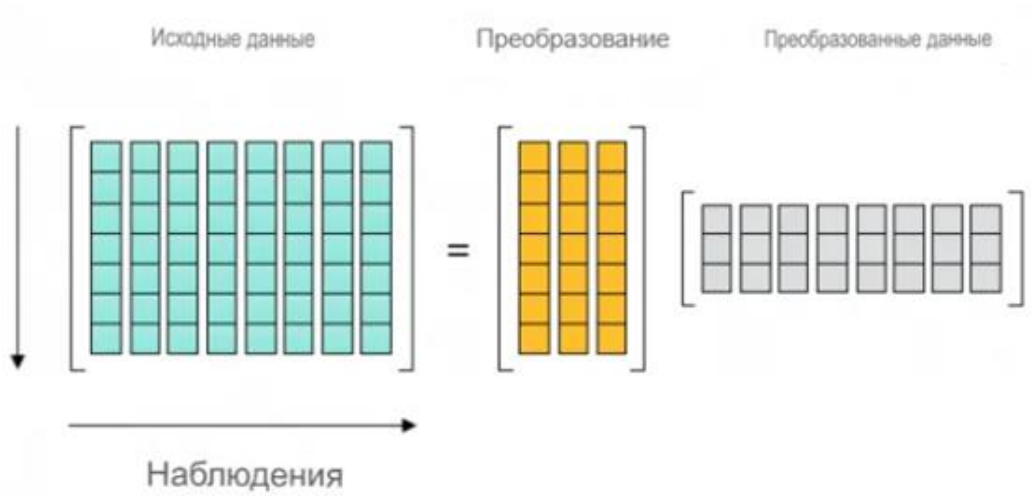

Рисунок 3 – Матричная факторизация разреженной матрицы

Разложение матрицы этими алгоритмами имеет общий вид:

$$V(m * n) = W(m * k) * H(k * n), \quad (5)$$

где V, W, H – исходная и преобразованные матрицы; m, n – размерность исходной матрицы; k – количество компонент.

Результатом матричной факторизации является новая реконструированная матрица, приближенная к исходной, где пустые значения обретают смысл путем выделения главных компонент. Чтобы ее получить, нужно выполнить обратное умножение матриц W и H.

1.3 Анализ данных музыкальных произведений и постановка требований к рекомендательной системе

Алгоритмы машинного обучения, включая алгоритмы, которые используют рекомендательные системы, работают с данными в виде набора атрибутов, представленными в числовом или категориальном виде [20].

Для реализации задачи рекомендаций музыкальных произведений, был выбран набор данных музыкальных произведений из открытого репозитория данных Kaggle. Этот массив данных содержит свыше 160 тысяч музыкальных произведений с числовыми и категориальными атрибутами:

– key – предполагаемый общий ключ трека;

– mode – тип шкалы, из которой извлекается мелодическое содержание трека.
– acoustiness – мера достоверности того, является ли дорожка акустической;
– danceability – описывает, насколько трек подходит для танцев, основанный на сочетании музыкальных элементов, включая темп, стабильность ритма, силу ритм и общую регулярность;
– energy – перцептивная мера интенсивности и активности трека;
– instrumentalness – шкала содержания вокальной составляющей в музыкальном треке;
– loudness – общая громкость в треке;
– valence – музыкальная позитивность, передаваемая треком;
– tempo – общий темп трека в ударах в минуту;
– popularity – популярность трека от 0 до 100.

Эти данные были извлечены из набора данных Spotify с помощью специального API и содержат метаданные треков и соответствующие им музыкальные функции.

Критерий корреляции Пирсона – это метод параметрической статистики, позволяющий определить наличие или отсутствие линейной связи между двумя показателями.

На рисунке 4 представлен график критерия корреляции Пирсона для представленных выше атрибутов и популярности. Формула расчета коэффициентов была представлена выше (2)

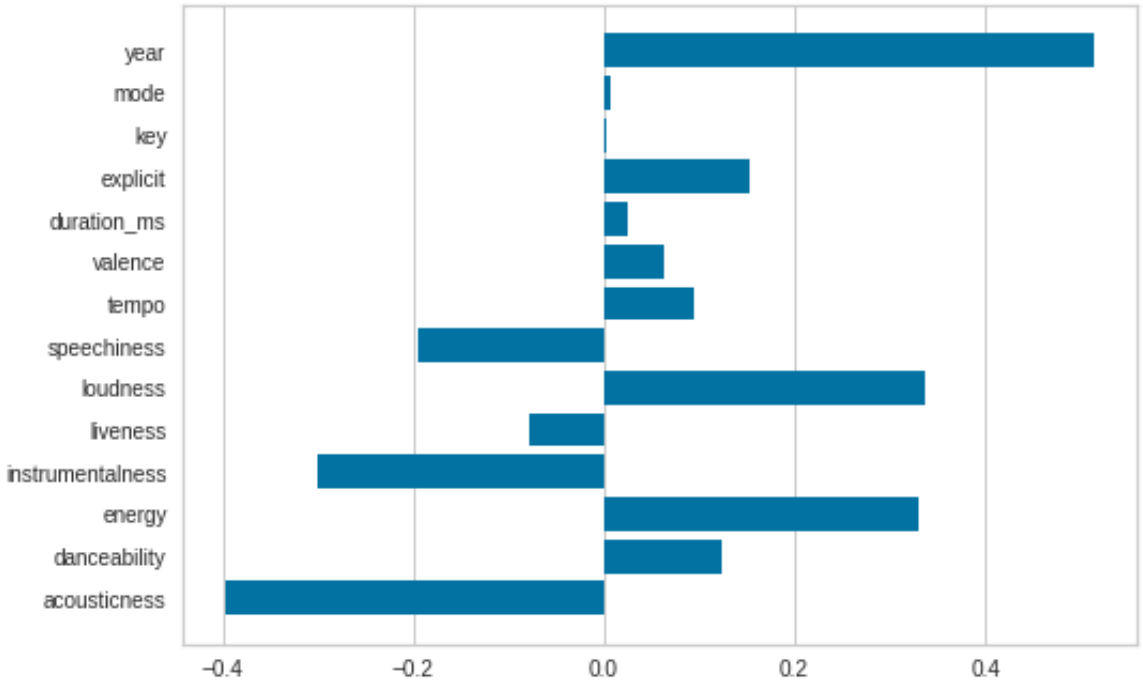

Рисунок 4 – Корреляция атрибутов по критерию Пирсона для предметов рекомендаций

Для оценки силы корреляционной связи обычно используют общепринятые критерии, согласно которым, при абсолютном значении менее 0.3 связь можно описать как слабую, при значении более 0.7 – сильную, в иных случаях – среднюю.

По графику видно, что на популярность влияют такие атрибуты как год, скорость соотношение произнесенных слов, инструментальность, энергичность, акустичность. Все остальные атрибуты предметов рекомендаций имеют слабую связь с атрибутом популярности.

Используя данные сгруппированные по годам можно понять, как изменялось звучание музыки за последние сто лет (рисунок 5-6).

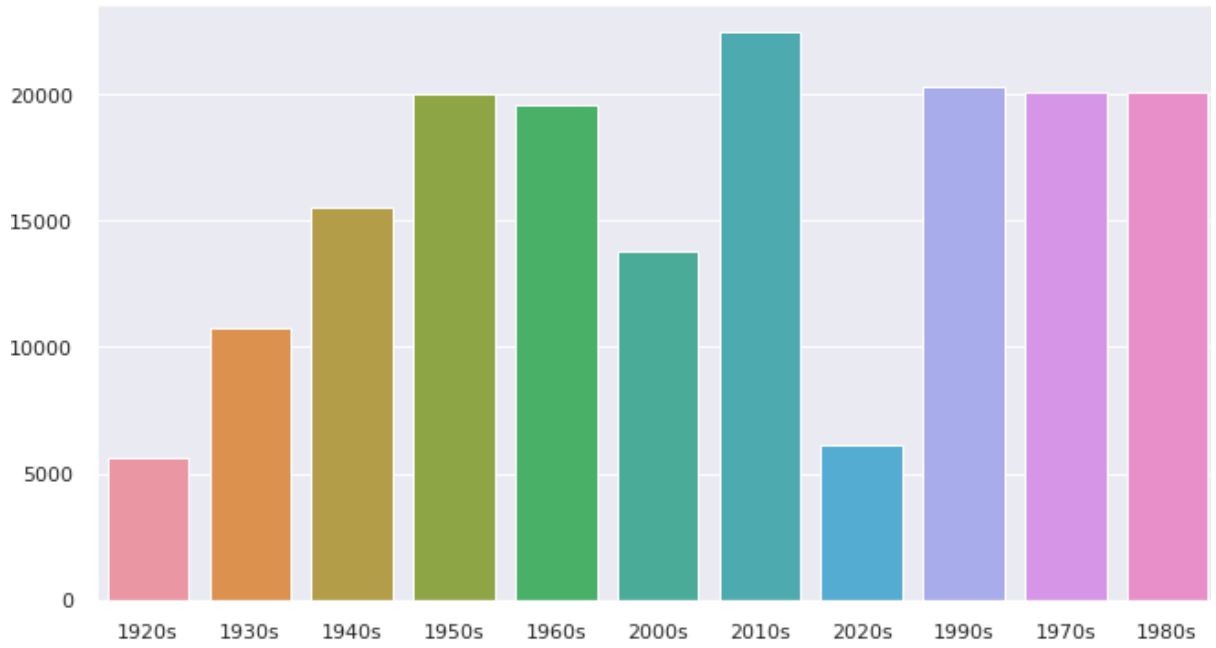

Рисунок 5 – Количество музыкальных произведений в каждом десятилетии

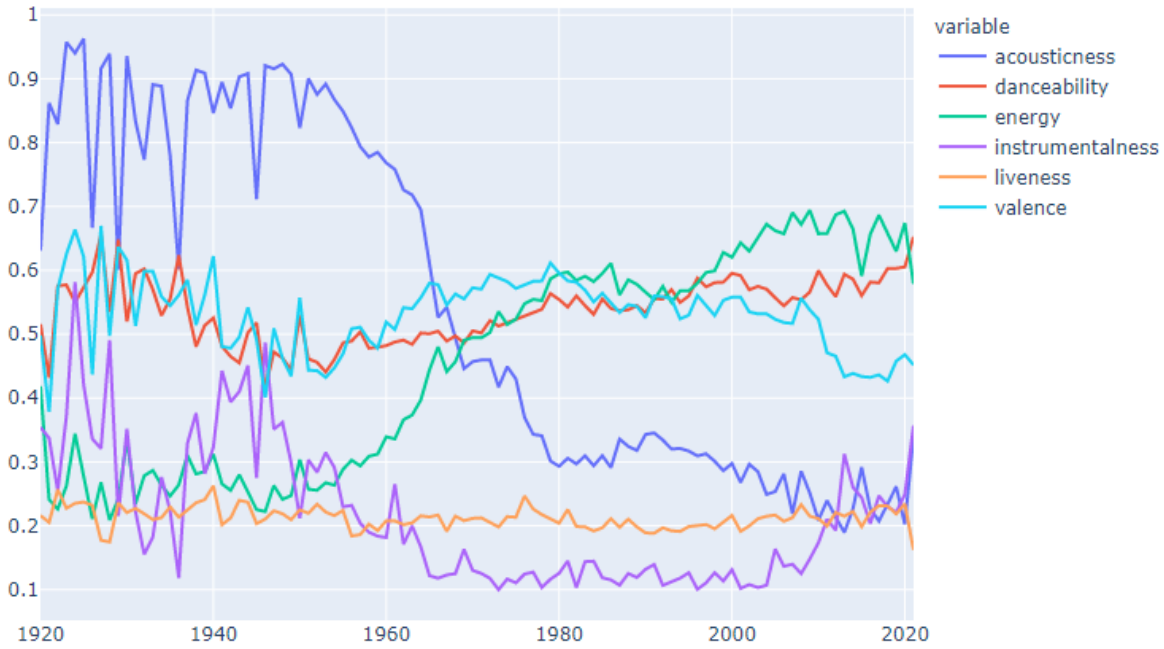

Рисунок 6 – Изменение значений числовых атрибутов с течением времени

Определив набор данных, можно вынести требования к разрабатываемой рекомендательной системе:

– объектом рекомендаций является музыкальные произведения, представленные в виде числовых и категориальных музыкальных характеристик;

– рекомендательная система должна быть персонализированной и независимой от других пользователей;
– рекомендательная система должна определять схожесть элементов основываясь на содержимом элементов рекомендаций;
– рекомендательная система должна уметь возвращать рекомендации пользователю в виде списка при получении в качестве входных данных просмотренных элементов пользователем на текущий момент времени.

Таким образом, необходимо построить персонализированную рекомендательную систему, основанную на содержимом (атрибутах) этих данных. Рекомендации будут предоставляться путем передачи в алгоритм некоторого набора произведений, которые просмотрел пользователь.

Вывод по главе 1

В данной главе были выполнены следующие задачи:
– определены характеристики рекомендательных систем;
– представлены методы вынесения рекомендаций, используемые в рекомендательных системах
– описан входной набор данных музыкальных произведений, взятый из открытого репозитория данных Kaggle;
– описаны основные требования к разрабатываемой рекомендательной системе музыкальных произведений.

В виду простоты реализации, а также высокой степени персонализации, был сделан выбор в пользу метода вынесения рекомендаций, основанном на содержимом.

Глава 2 Проектирование рекомендательной системы музыкальных произведений

2.1 Масштабирование данных музыкальных произведений

Некоторые алгоритмы машинного обучения чувствительны к масштабированию объектов, в то время как другие практически инвариантны к нему.

Алгоритмы машинного обучения, такие как линейная регрессия, логистическая регрессия или нейронные сети используют градиентный спуск в качестве метода оптимизации и требуют масштабирования данных. Разница в диапазонах атрибутов приведет к различным размерам шагов для каждой функции. Чтобы гарантировать, что градиентный спуск плавно движется к минимумам, и что шаги градиентного спуска обновляются с одинаковой скоростью для всех объектов, требуется масштабировать данные перед подачей в модель.

Алгоритмы расстояния, такие как K-средних (K Means) или K ближайших соседей наиболее подвержены влиянию диапазона значений. Поскольку при вычислении расстояния между двумя признаками, они могут находиться в разном диапазоне, существует вероятность того, больший вес будет передан признакам с большей величиной. Это повлияет на производительность алгоритма.

Нормировка – это метод, часто применяемый как часть подготовки данных для машинного обучения. Цель нормировки состоит в том, чтобы изменить значения числовых атрибутов в наборе данных на общую шкалу, не искажая различия в диапазоне значений. Для машинного обучения каждый набор данных не требует нормализации. Это требуется, когда объекты имеют различные диапазоны.

Суть нормировки состоит в том, что значения сдвигаются и масштабируются таким образом, что в конечном итоге они находятся в диапазоне от 0 до 1. Формула нормализации представлена ниже:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}, \quad (6)$$

где X_{max} и X_{min} – это максимальное и минимальное значение атрибутов соответственно.

Существует еще один метод масштабирования данных, называемый стандартизацией. Суть метода заключается в том, что значения центрируются вокруг среднего с единичным стандартным отклонением. Это означает что среднее значение атрибута становится равным нулю, а результирующее изменение имеет единичное стандартное отклонение. Формула стандартизации представлена ниже:

$$X' = \frac{X-\mu}{\sigma}, \quad (7)$$

где μ – среднее значений признаков, а σ – их стандартное отклонение.

Нормализацию следует использовать, когда распределение данных не соответствует гауссовкому распределению. Это может быть полезно, в алгоритмах, которые не предполагают никакого распределения данных.

Стандартизация, с другой стороны, полезна в тех случаях, когда данные следуют гауссовскому распределению. Кроме того, в отличие от нормализации, стандартизация не предполагает никакого ограничивающего диапазона. Стандартизация полезна во многих случаях. Ее применение может повысить численную стабильность модели и часто сокращает время обучения. Однако, стандартизация также может повредить производительности алгоритмов, если предполагается равная важность признаков. Если существуют внутренние различия между функциями, то, как правило, стандартизацию не рекомендуется проводить.

Чтобы определить, какой метод масштабирования лучше использовать для рекомендательной системы, требуется провести тесты производительности. Сравнение производительности методов будет представлено в следующей главе.

2.2 Использование алгоритма K-Means для кластеризации музыкальных произведений

Алгоритм K Means – это итеративный алгоритм, который разбивает набор данных на заранее определенные неперекрывающиеся различные подгруппы, где каждая точка данных принадлежит только одному кластеру.

Алгоритм кластеризации K Means относится к классу алгоритмов машинного обучения с обучением без учителя. Данный термин означает, что алгоритм обучается без вмешательства со стороны (без разметки данных).

Для работы алгоритму требуются следующие исходные данные:

1. Обучающая выборка X^m со всеми, заранее определенными атрибутами данных (8):

$$X^m = \{x_1, x_2, \dots, x_m\}, \quad (8)$$

где m – размер обучающей выборки; x_i – элемент обучающей выборки (9).

$$x_i^n = \{p_1, p_{2,} \dots, p_n\}, \quad (9)$$

где n – количество атрибутов элемента выборки; p – числовой или категориальный атрибут.

2. Количество кластеров, на которые будут делиться данные.

3. Метрика определения схожести элементов. Вместе с алгоритмом K means часто используются следующие метрики:

– Евклидово расстояние – расстояние между двумя точками в евклидовом пространстве (10);

$$d_E(x, x') = \sqrt{(p_1 - p_1')^2 \pm \dots \pm (p_n - p_n')^2}, \quad (10)$$

где x и x' – элемент обучающей выборки и центр кластера; p – атрибут элемента.

– расстояние Манхеттена – сумма абсолютной разности двух координат (элементов):

$$d_M(x, x') = \|(p_1 - p_1')\| \pm \dots \pm \|(p_n - p_n')\| \quad (11)$$

– расстояние Чебышева – это метрика расстояния, которая является максимальным абсолютным расстоянием в одном измерении двух N-мерных точек (12).

$$d_{\text{ч}}(x, x') = \max(\|(p_1 - p_1')\|, \dots, \|(p_n - p_n')\|), \quad (12)$$

где max – функция, определяющая абсолютное максимальное значение.

Одно из проблем алгоритма кластеризации K-Means является невозможность работы с категориальными атрибутами выборки. Так как категориальный признак является нечисловым атрибутом, то для расчета расстояний с помощью метрик требуется введение специального рода функций. Простейших из них является функция отличия (13).

$$f_{\text{отл}} = \begin{cases} 0, p_k = p'_k \\ 1, p_k \neq p'_k \end{cases} \quad (13)$$

Функция отличия возвращает ноль, если два категориальных атрибута между собой равны, иначе, возвращает единицу.

Работа алгоритма K Means состоит из следующих шагов:

1. Случайным образом выбирается k-объектов исходной выборки, которые будут начальными центрами кластеров.
2. Каждому объекту исходных данных определяется кластер, вычисляя расстояние между ними и центром кластера. Запись принадлежит тому кластеру, к центру которого она находится ближе всего. В качестве формулы сходства объектов можно использовать метрики, приведенные выше.
3. Как только для всех объектов будет определена принадлежность, производится смещение центров кластеров путем определения средних значений для каждого кластера (14).

$$M_i = (\frac{\sum_1^t p_1(t)}{t}, \dots, \frac{\sum_n^t p_n(t)}{t}), \quad (14)$$

где t – объекты, принадлежащие кластеру.

4. Шаги 2 и 3 должны повторяться пока не будет выполнен один из критериев остановки:

– сумма квадратов наименьших расстояний (15) достигнет достаточно малого значения:

$$E = \sum_{i=1}^{n} \sum_{p \in C_i} (p - m_i)^2, \quad (15)$$

где p – точка данных, которая принадлежит кластеру C_i, а m_i – является центром данного кластера;

– если границы кластеров остаются неизменными от итерации к итерации.

Набор данных музыкальных произведений содержит звуковые функции для разных песен, а также звуковые функции для разных жанров. На основе звуковых характеристик можно произвести кластеризацию музыки и жанров для определения средних значений кластеров (рисунок 7).

Рисунок 7 – Кластеризация музыкальных произведений

На рисунке 8 представлен результат кластеризации жанров музыки для десяти кластеров с использование метода главных компонент и алгоритма K Means.

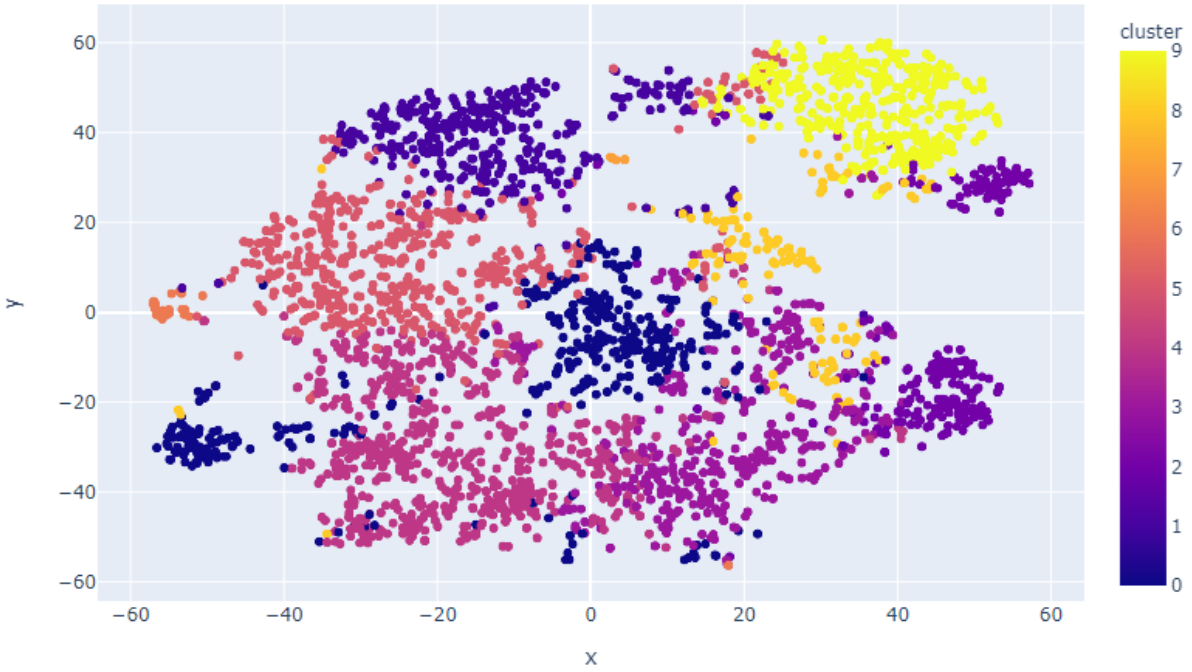

Рисунок 8 – Вывод результатов кластеризации жанров музыки

На рисунке 9 представлен результат кластеризации музыкальных произведений.

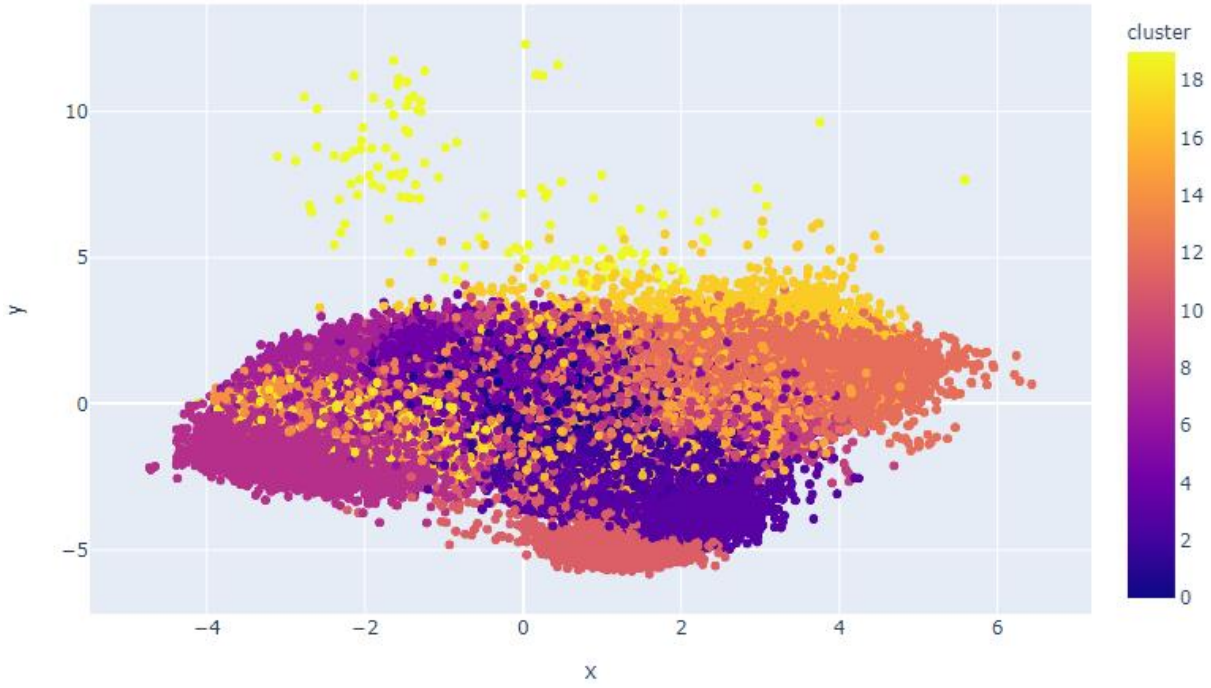

Рисунок 9 – Результат кластеризации музыкальных произведений

На рисунках 8, 9 видно, что подобные жанры, как правило, имеют точки данных, расположенные близко к друг другу, в то время, как похожие типы песен также сгруппированы вместе. Это наблюдение имеет смысл, так

как похожие жанры будут звучать одинаково и будут происходить из аналогичных периодов времени, как и музыка в этих жанрах.

Для визуализации графиков было произведено матричное разложение для уменьшения размерности с помощью метода главных компонент таким образом, что в результирующей матрице находится две компоненты – атрибута [18].

2.3 Алгоритм KNN для нахождения сходства между предметами рекомендаций

Алгоритм K-ближайших соседей (K Nearest Neighbors) – это непараметрический ленивый алгоритм машинного обучения. Непараметрическими алгоритмами машинного обучения, называют такие алгоритмы, которые не имеют предположения об изначальном распределении данных обучающей выборки. То есть, структура определяется на основе набора данных. Ленивым алгоритм называют потому, что он не использует обучающий данные для каких-либо обобщений.

Алгоритм высчитывает расстояния между точками (векторами данных), а затем присваивает точки, которые еще не были классифицированы, классу k-ближайших соседей.

Исходные данные алгоритма k-ближайших соседей:

1. Обучающая выборка, состоящая из объектов одинакового набора атрибутов. Для каждого объекта должны быть известны все значения атрибутов и принадлежность к одному из классов. Атрибуты объектов могут быть числовыми и категориальными.
2. В качестве метрики расчета расстояния между объектами часто используются формулы расчета расстояний Евклида (10), Манхеттена (11) или Чебышева (12). Как уже говорилось ранее, для рекомендательной системы, основанной на содержимом предметов рекомендаций, такой метрикой может служить косинусной расстояние (1) и критерий Пирсона (2).

3. Объекты, для которых необходимо провести тестирования. На выходе алгоритма этим объектам будут присвоены классы.

Алгоритм KNN состоит и следующих шагов:

1. Определить количество ближайших точек данных – K. K может быть любым целым числом;
2. Вычислить расстояния от классифицируемого (неизвестного) объекта до объектов обучающей выборки;
3. Отсортировать данные в порядке возрастания на основе значений расстояний;
4. Выбрать K – количество объектов из отсортированного списка, сходство которых наиболее высоко.

К плюсам алгоритма KNN можно отнести:

– простоту реализации и возможность внедрения различных модификаций алгоритма,
– универсальность – алгоритм полезен для решения задач регрессии и классификации,
– хорошо понятную логику работы алгоритма в любых предметных областях.

У алгоритма KNN существуют следующие недостатки:

– понижение точности работы алгоритма при наличии погрешностей в исходных данных,
– точность работы зависит от количества данных в обучающей выборке,
– необходимость в хранении обучающей выборки целиком, что приводит к неэффективному расходу памяти и усложнению решающего правила,
– необходимость сравнивания объекта со всеми объектами обучающей выборки, что может сказаться на производительности при наличии больших выборок.

Алгоритм K ближайших соседей будет использоваться для решения задачи определения сходства входного набора музыки пользователя со всеми музыкальными произведениями (рисунок 10). В качестве обучающей

выборки алгоритм может использовать результаты кластеризации музыкальных произведений алгоритма K Means. Таким образом, полученные классы, присвоенные в ходе кластеризации, будут определены для набора музыки пользователя, после чего останется лишь отсортировать результаты по уровню сходства.

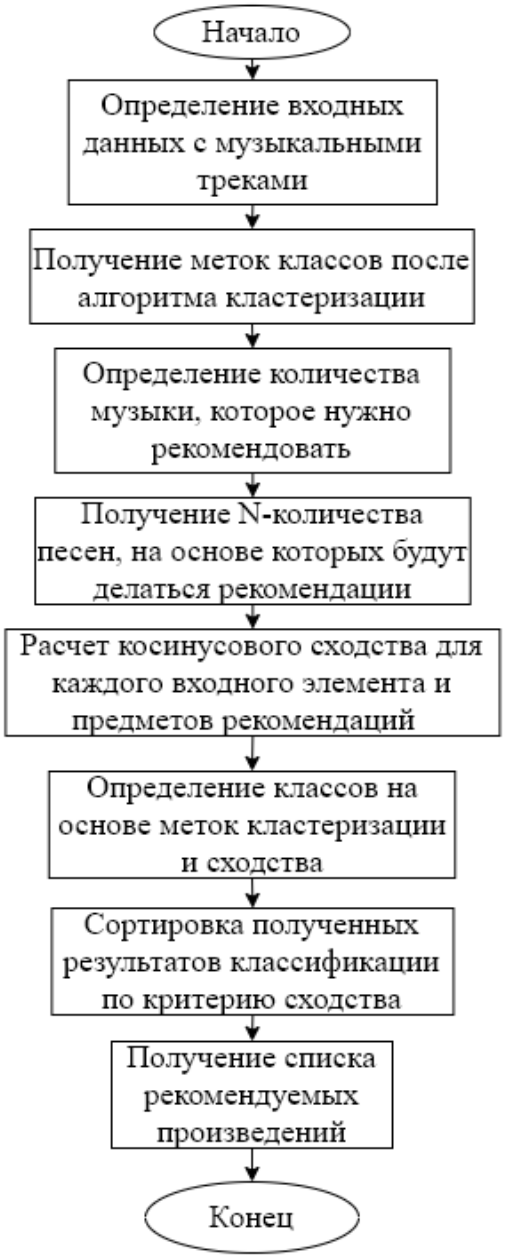

Рисунок 10 – Алгоритм K-ближайших соседей для вынесения рекомендаций пользователю

В качестве обучающей выборки алгоритма будут использоваться данные, описанные в предыдущей главе. Следует произвести

масштабирование данных, после чего, определить количество рекомендаций на выходе из алгоритма.

2.4 Алгоритм рекомендательной системы

Рекомендательная система будет основана на представленных выше алгоритмах K Means и KNN с проведением предварительной обработки данных при подаче в модель (рисунок 11).

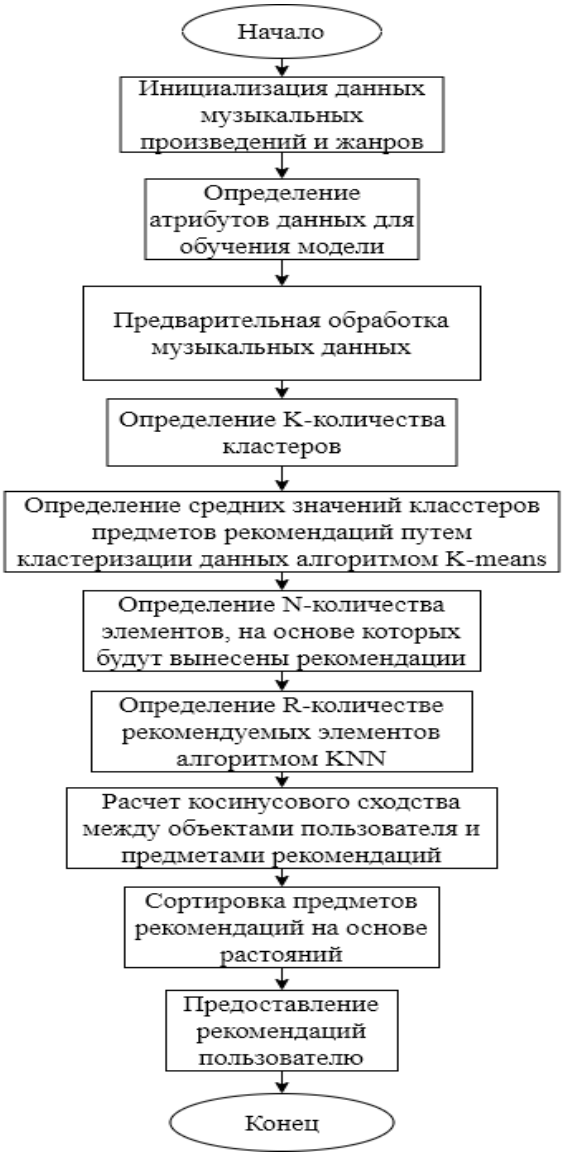

Рисунок 11 – Алгоритм рекомендательной системы, основанной на содержимом

После инициализации данных музыкальных произведений, следует исключить некоторые атрибуты из выборки, такие, как «название» и

«авторы», так как эти атрибуты не должны влиять на предоставление рекомендаций. Цель рекомендательной системы предложить, как можно больше различных рекомендаций, не основанных на авторах и названиях музыки.

Рекомендательная система проводит кластеризацию музыкальных произведений стараясь отделить разные жанры музыки друг от друга. Полученные от пользователя данные сравниваются с точками данных кластеров, после чего сортируются по параметру схожести и рекомендуют пользователю определенное количество музыкальных рекомендаций.

На первом шаге работы алгоритма следует произвести масштабирование данных музыкальных произведений и данных, полученных от пользователя с использованием одной из метрик масштабирования.

Далее, с помощью задачи кластеризации, различные жанры музыкальных произведений будут отделены друг от друга. На основе кластерного анализа (рисунок 8-9), можно сделать вывод, что существует различие между разными жанрами музыкальных произведений, чем можно воспользоваться при вынесении рекомендаций пользователю.

На основе полученных результатов кластеризации и косинусного сходства между точками данных и треками, полученными от пользователя, будет определяться схожесть предметов рекомендаций. Таким образом, отсортировав результаты классификации по параметру схожести и исключив треки, которые были получены пользователя, можно вернуть пользователю список музыки, рекомендуемой ему.

Вывод по главе 2

Во второй главе производилось проектирование рекомендательной системы:

– было представлено описание методов предварительной обработки данных;

– описан алгоритм K Means для кластеризации данных музыкальных произведений с целью отделения жанров друг от друга;

– представлен кластерный анализ музыкальных данных;

– описан алгоритм KNN для определения похожих треков с использованием метрики косинусного сходства.

При большом объеме данных 1 итерация может занимать много времени, поэтому при реализации алгоритма, потребуется определить оптимальное количество кластеров, а также выбрать метод обработки данных. Входные параметры алгоритмов K Means и KNN могут сильно влиять на производительность, что будет более подробно описано в 3 главе.

Глава 3 Реализация рекомендательной системы музыкальных произведений

3.1 Выбор программных средств

Рекомендательная система была реализована с использованием языка программирования Python версии 3.6 в среде разработки Visual Studio Code. Для начала работы требуется установить и подключить следующие зависимости (библиотеки):

– NumPy – библиотека, которая используется для научных вычислений. Содержит мощный n-мерный объект массива, а также полезен в линейной алгебре и может использоваться в качестве эффективного многомерного контейнера для общих данных.

– Pandas – библиотека, которая используется для анализа и очистки данных Она хорошо подходит для различных типов данных – таблиц, упорядоченных или неупорядоченных данных временных рядов, произвольных матричных данных с метками строк и столбцов, немаркированных данных, а также для любой другой формы наборов наблюдательных или статистических данных. Pandas будет использоваться для загрузки и хранения данных музыкальных произведений, а также для работы с ними.

– MatPlotLib/PyPlot – библиотека для построения 2-D графиков. Используется для визуализации при анализе данных.

– Seaborn – библиотека для сложной визуализации. Может использоваться для создания статистических графиков. В отличие от MatPlotLib может работать с фреймами данных, предоставляемыми Pandas.

– SciPy – библиотека, предоставляющая математические алгоритмы, включая алгоритмы определения расстояний между объектами и другие метрики.

– Scikit-learn – библиотека машинного обучения. Она включает в себя различные алгоритмы, включая алгоритмы кластеризации и классификации, а также поддерживает научные библиотеки, такие как, NumPy и SciPy. И данной библиотеки будут подключены следующие модули:

– StandardScaler, MinMaxScaler – модели масштабирования данных путем стандартизации (7) и нормировки объектов (6);

– KMeans представляет собой модель классификатора по алгоритму K средних;

– Pipeline используется для объединения нескольких моделей библиотеки в одну. Это полезно, так как часто существует определенная последовательность шагов при обработке данных, например, выбор объектов, нормализация и классификация.

Подключение всех библиотек представлено на рисунке 12.

```
 9  import numpy as np
10  import pandas as pd
11  import seaborn as sns
12  import matplotlib.pyplot as plt
13  from scipy.spatial.distance import cdist
14  from sklearn.cluster import KMeans
15  from sklearn.preprocessing import StandardScaler
16  from sklearn.preprocessing import MinMaxScaler
17  from sklearn.pipeline import Pipeline
```

Рисунок 12 – Подключение зависимостей в проект

После подключения всех зависимостей можно переходить к реализации рекомендательной системы.

3.2 Программная реализация рекомендательной системы

Для начала следует определить данные музыкальных произведений (рисунок 13).

```
46  import easygui
47  from easygui import *
48  input_file = easygui.fileopenbox(filetypes=["*.csv"])
49  data = pd.read_csv(input_file)
```

Рисунок 13 – Подключение музыкальных данных

Подключение данных выполняется с помощью библиотеки easygui. Подключенные данные хранятся в виде таблицы объектом типа DataFrame из библиотеки Pandas. Данные разделяются для определения корреляции атрибутов при визуализации по критерию Пирсона.

Модель кластеризации представлена ниже на рисунке 14.

```
129  song_cluster_pipeline = Pipeline([('scaler', var2),
130                                    ('kmeans', KMeans(n_clusters=int(var1),
131                                     verbose=2, n_jobs=4))],verbose=True)
```

Рисунок 14 – Модель кластеризации музыкальных данных

Конструктор класса Pipeline содержит следующие параметры:

– Кортеж шагов (steps), которые соединены в цепочку в том порядке, к котором должны выполниться. Содержит модели стандартизации и кластеризации. В данном случае будет произведена стандартизация, а затем, кластеризация данных.

– Verbose – вывод времени, затраченного на выполнение каждого шага в консоль, при истинном значении этого параметра.

Объект Pipeline может вернуть словарь с атрибутами доступа к каждому шагу, где ключи – имена шагов, а значения – их параметры.

Модель стандартизации принимает следующие опциональные параметры:

– Копирование не разреженной матрицы. Перед стандартизацией значений делает копию матрицы.

– Усреднение значений данных перед стандартизацией.

– Масштабирование данных до единичной дисперсии, что эквивалентно единичному стандартному отклонению.

Перечисленные параметры включены по умолчанию. Объект стандартизации может вернуть масштабирование данных по каждому объекту, среднее значение для каждой функции в обучающей выборке, дисперсию для каждого объекта и количество выборок стандартизации.

После стандартизации инициализируется модель кластеризации по алгоритму K Means. В объект были переданы следующие параметры:

– количество кластеров, а также количество их центров для генерации;

– вывод в консоль прогресса обучения модели;

– количество потоков обучения.

На рисунке 15 представлен программный код обучения модели.

```
133   X = data.select_dtypes(np.number)
134   number_cols = list(X.columns)
135   song_cluster_pipeline.fit(X)
136   song_cluster_labels = song_cluster_pipeline.predict(X)
137   data['cluster_label'] = song_cluster_labels
```

Рисунок 15 – Кластеризация

Из таблицы данных выбираются только числовые атрибуты. Затем, производится наполнение модели данных и выполняется кластеризация. Результатом кластеризации являются метки кластеров, которые добавляются в таблицу с данными.

Так как данные от пользователя будут приниматься в виде объекта, подобного JSON, необходимо реализовать функцию, переводящий массив объектов в одномерный словарь для удобной работы с данными, полученными от пользователя (рисунок 16).

```
203  def flatten_dict_list(dict_list):
204
205      flattened_dict = defaultdict()
206      for key in dict_list[0].keys():
207          flattened_dict[key] = []
208
209      for dictionary in dict_list:
210          for key, value in dictionary.items():
211              flattened_dict[key].append(value)
212
213      return flattened_dict
```

Рисунок 16 – Функция переводящая массив объектов в словарь

Для определения центра кластера полученных от пользователя музыкальных произведение производится расчет вектора средних значений (рисунок 17).

```
187  def get_mean_vector(song_list, spotify_data):
188
189      song_vectors = []
190
191      for song in song_list:
192          song_data = get_song_data(song, spotify_data)
193          if song_data is None:
194              print('Warning: {} does not exist in Spotify or in database'.format(song['name']))
195              continue
196          song_vector = song_data[number_cols].values
197          song_vectors.append(song_vector)
198
199      song_matrix = np.array(list(song_vectors))
200      return np.mean(song_matrix, axis=0)
```

Рисунок 17 – Расчет вектора средних значений

На первом этапе производится поиск атрибутов предметов рекомендаций среди данных музыкальных произведений. Объекты добавляются в список, после чего производится расчет средних значений.

Получение атрибутов музыкальных произведений производится с помощью следующей функции (рисунок 18).

```
176  def get_song_data(song, spotify_data):
177
178      try:
179          song_data = spotify_data[(spotify_data['name'] == song['name'])
180                                   & (spotify_data['year'] == song['year'])].iloc[0]
181          return song_data
182
183      except IndexError:
184          return find_song(song['name'], song['year'])
```

Рисунок 18 – Функция для получения атрибутов музыки

Сначала функция производит поиск среди выгруженных данных по значениям атрибутов названий и даты выпуска музыкального произведения. Если объект не был найден, то будет произведен поиск с помощью специального API Spotify.

На рисунке 19 представлена функция для поиска музыки среди данных Spotify.

```
155  def find_song(name, year):
156      song_data = defaultdict()
157      results = sp.search(q= 'track: {} year: {}'.format(name,year), limit=1)
158      if results['tracks']['items'] == []:
159          return None
160
161      results = results['tracks']['items'][0]
162      track_id = results['id']
163      audio_features = sp.audio_features(track_id)[0]
164
165      song_data['name'] = [name]
166      song_data['year'] = [year]
167      song_data['explicit'] = [int(results['explicit'])]
168      song_data['duration_ms'] = [results['duration_ms']]
169      song_data['popularity'] = [results['popularity']]
170
171      for key, value in audio_features.items():
172          song_data[key] = value
173
174      return pd.DataFrame(song_data)
```

Рисунок 19 – Поиск музыки на основе данных Spotify

Производится поиск по названию и дата выпуска, после чего, если поиск дал результаты создается таблица атрибутов трека.

Далее необходимо реализовать функцию для вынесения рекомендаций пользователю (рисунок 20).

```
216 def recommend_songs( song_list, spotify_data, n_songs=10):
217 
218     metadata_cols = ['name', 'year', 'artists', 'popularity']
219     song_dict = flatten_dict_list(song_list)
220 
221     song_center = get_mean_vector(song_list, spotify_data)
222     scaler = song_cluster_pipeline.steps[0][1]
223     scaled_data = scaler.transform(spotify_data[number_cols])
224     scaled_song_center = scaler.transform(song_center.reshape(1, -1))
225     distances = cdist(scaled_song_center, scaled_data, 'cosine')
226     index = list(np.argsort(distances)[:, :n_songs][0])
227 
228     rec_songs = spotify_data.iloc[index]
229     rec_songs = rec_songs[~rec_songs['name'].isin(song_dict['name'])]
230     return rec_songs[metadata_cols].to_dict(orient='records')
```

Рисунок 20 – Функция вынесения рекомендаций

Функция принимает параметры в следующей последовательности:

1. Полученные от пользователя объекты в виде списка, на основе которых будут выноситься рекомендации.
2. Данные предметов рекомендаций.
3. Количество музыкальных произведений, которые будут рекомендованы пользователю.

Функция реализует следующие шаги:

1. Полученный список произведений от пользователя преобразуется в плоский словарь;
2. Производится расчет центра данных на основе полученных данных и предметов рекомендаций;
3. Выполняется масштабирование данных музыкальных произведений и средних значений;
4. С помощью функции cdist производится расчет косинусного сходства между вектором средних значений и результатами кластеризации.

5. На основе полученных расстояний формируется список индексов, отсортированный по значению сходства

6. Производится поиск рекомендуемых элементов по индексу, исключая элементы, которые совпали с полученными от пользователя.

Главная функция системы рекомендаций представлена на рисунке 21.

```
232 def main():
233     input_file = easygui.fileopenbox(filetypes=["*.json"])
234     user_data = []
235     with open(input_file) as json_data:
236         user_data = json.load(json_data)
237         json_data.close()
238     input_data('Введите количество рекомендаций', 'Количество рекомендаций')
239     recommendations = recommend_songs(user_data,  data, int(var1))
240 
241     print('\n\nРекомендации\n\n')
242     for i in range(len(recommendations)):
243         print(i+1, '- Название:', recommendations[i]['name'],
244         '| Исполнители:', recommendations[i]['artists'],
245         '| Год выпуска:', recommendations[i]['year'],
246         '| Рейтинг:', recommendations[i]['popularity'])
```

Рисунок 21 – Функция вызова рекомендаций

Функция вызывает системные окна для выбора файла с расширением JSON, содержащего объекты пользователя, на основе которых будут выноситься рекомендации, после чего, предложит ввести количество рекомендаций на выходе. Таким образом, на выходе из алгоритма пользователь получит список рекомендуемых ему элементов.

3.3 Тестирование

Для того, чтобы рекомендательная система предоставляла высококачественные рекомендации, необходимо постоянно обновлять информацию об интересах пользователя. Но существует и другая проблема – пользователь всегда хочет минимизировать время, затрачиваемое на взаимодействие с системой и неохотно делится информацией для обратной связи. Это типичная проблема для всех рекомендательных систем, и

поскольку данная рекомендательная система основана исключительно на числовых атрибутах треков, то чем меньше информации пользователь предоставляет, тем менее соответствующий набор рекомендаций пользователь получит.

Рекомендательная система обладает следующими особенностями:

– возможность ввода количества жанров при кластеризации музыки;
– загрузка данных музыкальных произведений;
– загрузка предоставленных данных пользователем в виде JSON-объекта, чтобы эмулировать REST-запрос;
– возможность определения количества рекомендуемых треков.

На предоставленные рекомендации влияют следующие факторы:

– количество кластеров;
– значения атрибутов выборки;
– метрика расчета расстояний;
– количество объектов, полученных от пользователя.

Для начала, следует определить оптимальное K – количество кластеров (жанров) музыки при кластеризации данных (рисунок 22) [12, 16].

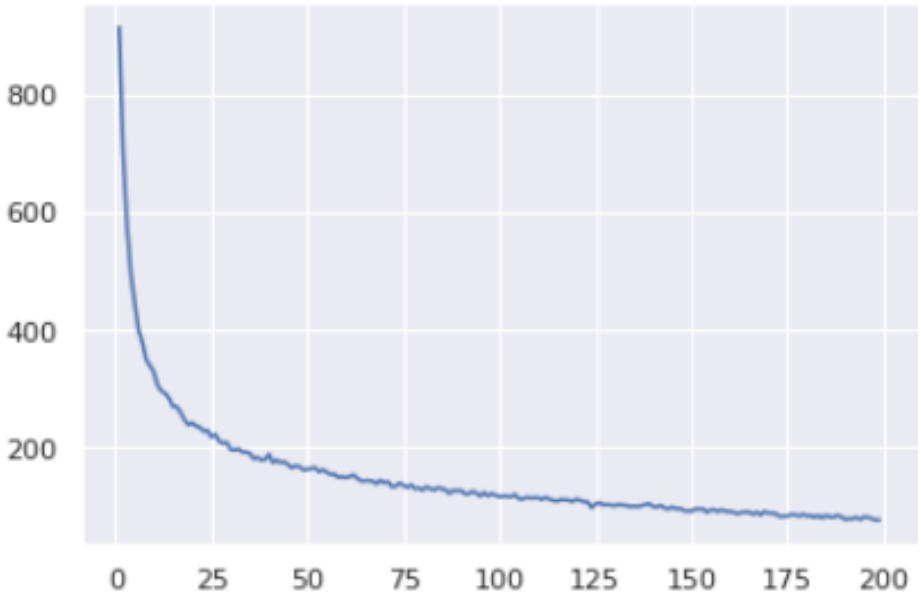

Рисунок 22 – Сумма квадратов ошибки

На оси абсцисс рисунка 22 представлено количество кластеров алгоритм, на оси ординат – значения суммы квадратов ошибок на каждой итерации алгоритма K Means (15). Увеличением количество кластеров, увеличивается время работы алгоритма (таблица 1), однако, также уменьшается ошибка, что положительно влияет на работу алгоритма.

Таблица 1 – Время работы алгоритма K-Means

Количество кластеров	**Время работы t (в секундах) при использовании стандартизации**	**Время работы t (в секундах) при использовании нормировки**
5	3.9	2.3
15	11	12.3
25	34.6	27.9
50	108	66
100	258	162
200	594	426

Несмотря на то, что кластеризация выполняется в 4 потоках, начиная с 25 кластеров, обработка данных будет занимать большое количество времени. Таким образом, для кластеризации музыки было выбрано количество кластеров равным 25. При таком количестве кластеров ошибка не так высока, а время кластеризации не занимает больше минуты.

На таблице 1 видно, что разные методы предварительной обработки данных перед передачей в модель кластеризации, также существенно влияют на производительность и могут повлиять на рекомендации, предлагаемые пользователю в результате. Так как время работы алгоритма с нормировкой данных меньше, для масштабирования данных лучше использовать этот метод.

Так как алгоритм выбирает изначальные центры кластеров случайным образом, то время работы алгоритма, представленное в таблице 1, может отличаться при каждой кластеризации данных.

В зависимости от выбранной метрики, предоставляемые рекомендации могут выглядеть по-разному. На рисунке 23 представлен вывод результатов работы алгоритма.

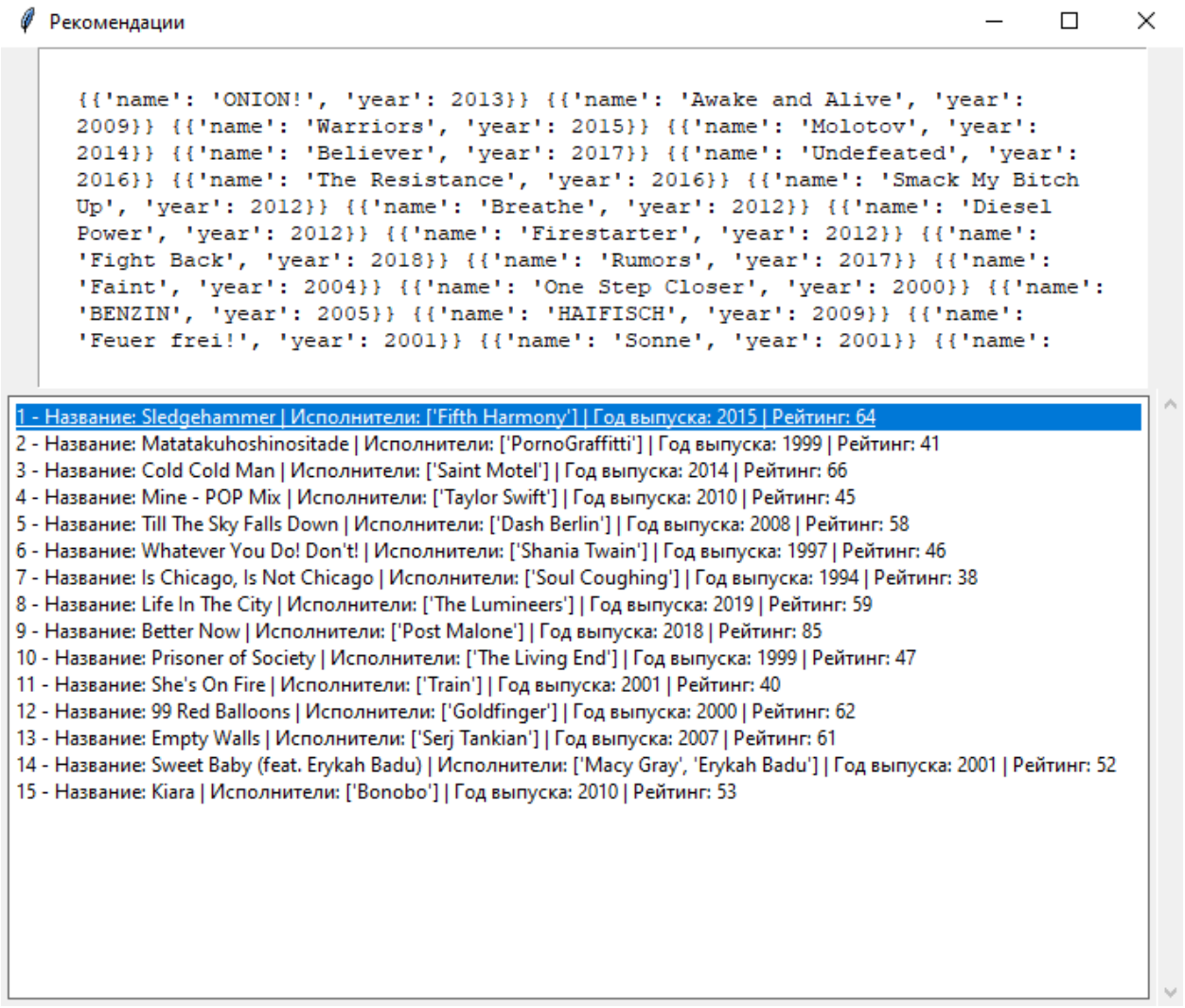

Рисунок 23 – Полученные рекомендации

При тестировании было выбрано 25 кластеров и нормировка в качестве метода предварительной обработки данных. На рисунке 20 также можно увидеть переданные от пользователя объекты, на основе которых вычисляются рекомендации.

В контексте рекомендательных систем разработчики заинтересованы в том, чтобы рекомендовать пользователю топ N элементов. Таким образом точность, полнота и f-мера будут зависеть от количества элементов,

рекомендуемых пользователю. В качестве порогового значения этих метрик будет установлен атрибут популярности объектов пользователя, который рассчитывается как среднее [4, 6, 7].

На рисунке 24 представлен график точности метрики Precision разработанного алгоритма в зависимости от количества предоставляемых рекомендаций.

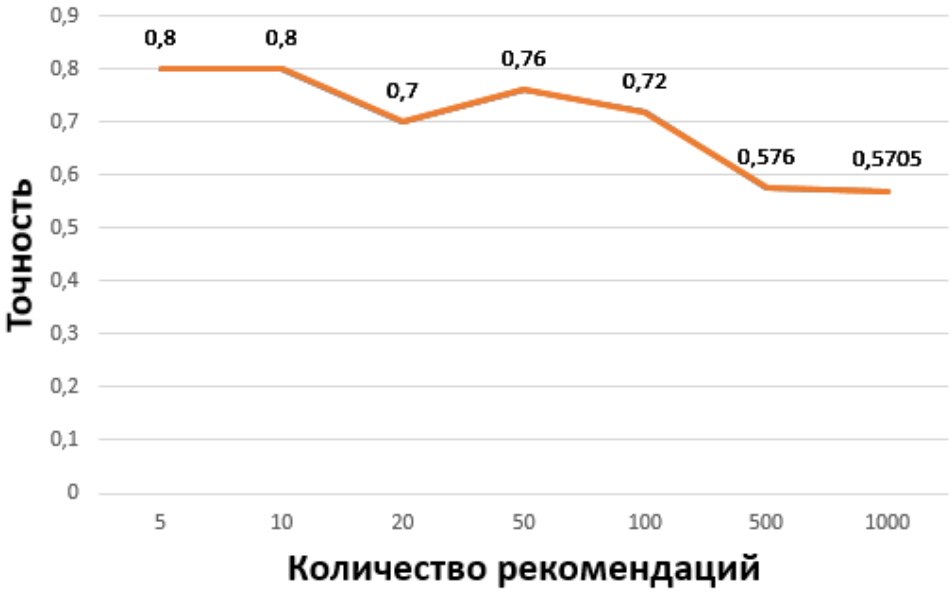

Рисунок 24 – Precision

Уменьшение значения точности с увеличением количества рекомендаций связано с тем, что при большем количестве рекомендаций появляются все более несоответствующие интересам пользователя элементы, рейтинг которых также можешь различаться.

На рисунке 25 представлен график полноты метрики Recall.

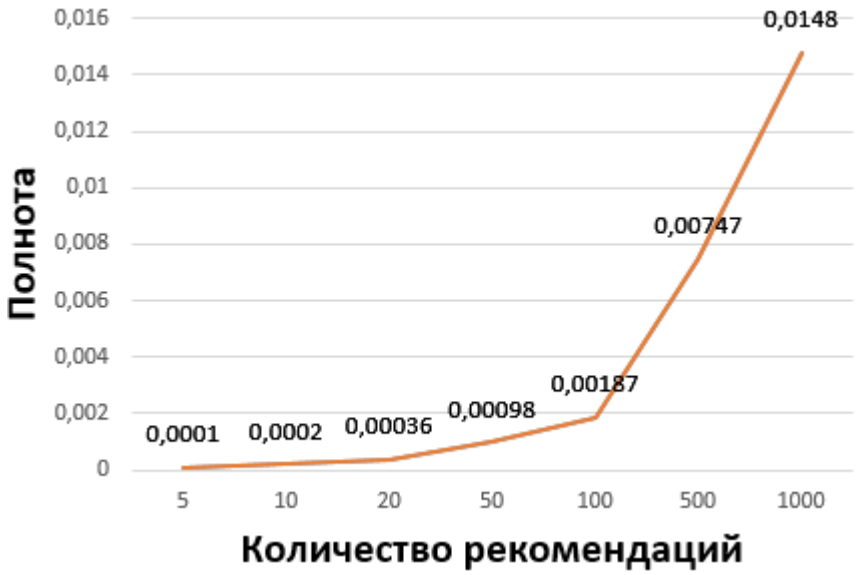

Рисунок 25 – Recall

На рисунке 26 представлен график f-меры.

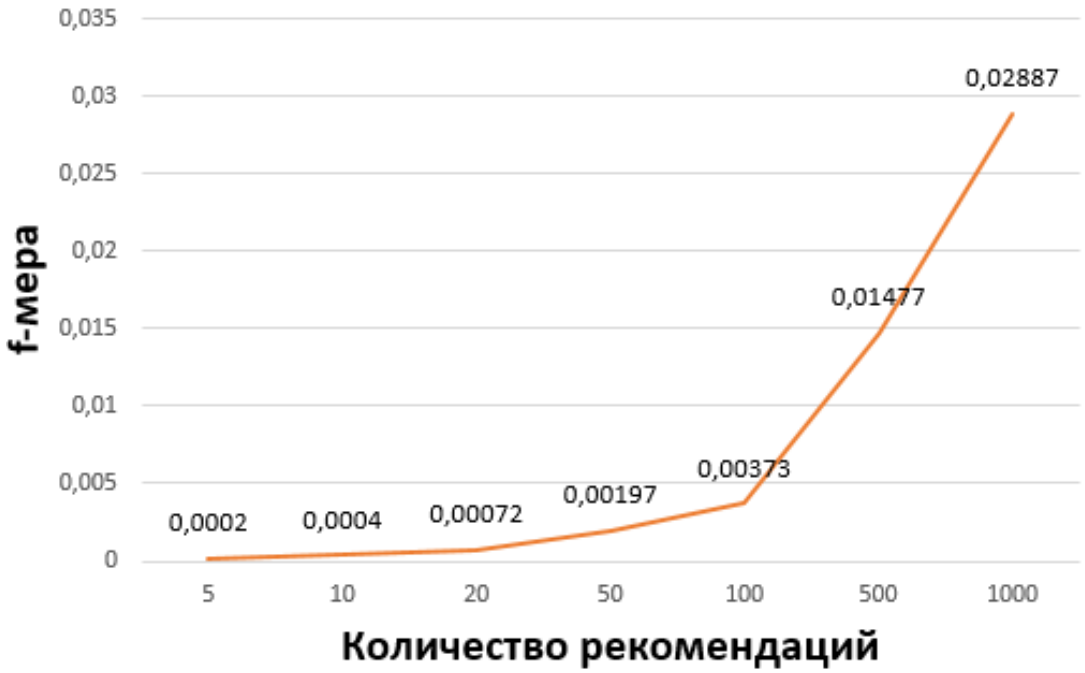

Рисунок 26 – F-мера

Значение метрики Recall будет всегда увеличиваться с уменьшением Precision и наоборот – стремиться к нулю при увеличении.

F-мера достигает максимума при полноте и точности, равным единице, и близка к нулю, если один из аргументов равен нулю.

Таким образом, можно сказать, что точность разработанной рекомендательной системы наиболее высока при количестве рекомендаций в

промежутке от 5 до 50 элементов и также зависит от качества предоставляемых пользователем объектов.

Вывод по главе 3

В третьей главе осуществлялась программная реализация и тестирование рекомендательной системы музыкальных произведений. За основу были взяты описанные в первой и второй главе алгоритмы и методы.

Основным инструментом реализации был выбран язык программирования Python и библиотеки сообщества для машинного обучения, визуализации, работы с данными, а также библиотеки реализующие математические функции.

Был протестирован алгоритм K Means для нахождения оптимального количества кластеров и метода масштабирования. Таковыми являются 25 кластеров и нормировка в качестве метода предварительной обработки данных.

Алгоритм KNN был реализован на основе метрики косинусного сходства.

В конце главы было произведено тестирование достоверности рекомендательной системы. В качестве метрик использовались Recall, Precision и F-мера.

Таким образом, в данной главе была реализована персонализированная рекомендательная система, основанная на контенте.

Заключение

Выпускная квалификационная работа посвящена реализации и анализу рекомендательной системы. Предметами рекомендаций являются треки музыкальных данных. Были выполнены следующие цели:

– были определены атрибуты объектов рекомендаций и описаны характеристики рекомендательных систем;

– был проведен анализ существующих алгоритмов вынесения рекомендаций;

– в качестве алгоритма музыкальной рекомендательной системы, был сделан выбор в сторону рекомендательной системы, основанной на содержимом;

– был представлен алгоритм музыкальной рекомендательной системы – в качестве основы были взяты такие алгоритмы машинного обучения, как K Means и KNN;

– была представлена программная реализация рекомендательной системы, после чего было произведено тестирование производительности и анализ влияния метрики расчета сходства на рекомендации.

Разработанную рекомендательную систему лучше всего использовать на сервере, так как серверной части приложения рекомендательная система может получать доступ к базе данных, хранящихся там музыкальных произведений и их атрибутам. Использование рекомендательной системы на клиентской части приложения не рекомендуется, так как вычисления могут занимать много времени и сильно нагружать устройство пользователя.

Рекомендательная система не рекомендует элементы, которые были получены от пользователя, и получает объекты, на основе которых будут выноситься рекомендации в виде JSON-объекта.

Список используемой литературы и список используемых источников

1. Бхаргава, А. Грокаем алгоритмы: Иллюстрированное пособие для программистов и любопытствующих. СПб.: Питер, 2017. 288 с.
2. Доусон, М. Программируем на Python. СПб.: Питер, 2017. 416 с.
3. Кацов И. Машинное обучение для бизнеса и маркетинга. СПб.: Питер, 2017. 512 с.
4. Ким Ф., Павлов Д.М. Рекомендательные системы на практике. М.: ДМК Пресс, 2020. 448 с.
5. Adomavicius G., Tuzhilin A. Context-aware recommender systems. In Recommender systems handbook. Springer, 2011. pp.: 217-253.
6. Charu C. Recommender Systems: The textbook. Aggarwal Springer, 2016. 519 p.
7. Desrosiers C., Karypis G. A comprehensive survey of neighborhood based recommendation methods // In Recommender systems handbook. Springer, 2011. pp.: 107-144.
8. Gantner Z., Rendle S., Schmidt-Thieme L. Factorization models for context-time-aware movie recommendations // In Proceedings of the Workshop on Context-Aware Movie Recommendation. ACM, 2010. pp.:14-19
9. Hidasi B., Tikk D. Fast als-based tensor factorization for context-aware recommendation from implicit feedback // In Machine Learning and Knowledge Discovery in Databases. Springer, 2012. pp.: 67-82.
10. Hu Y., Koren Y., Volinsky C. Collaborative filtering for implicit feedback dataset // In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. IEEE, 2008. pp.: 263-272.
11. Lemire D., Maclachlan A. Slope one predictors for online rating-based collaborative filtering // In SDM. SIAM. 2005, pp.: 1-5.
12. Liu, Y. An Improved Kernel K-means Clustering Algorithm // Yang Liu, Hong Peng Yin, Yi Chai // Proceedings of 2016 Chinese Intelligent Systems Conference. Springer Science Business Media Singapore, 2016. pp. 275-280

13. Lutz, M. Learning Python. O'Reilly Media, 2013. 1507 p.

14. Neil W. Artificial Intelligence: What You Need to Know About Machine Learning, Robotics, Deep Learning. Recommender Systems, Internet of Things, Neural Networks, Reinforcement Learning, and Our Future // Neil Wilkins – Independently Publisher, 2019. 104 p.

15. Pil´aszy I., Zibriczky D., Tikk D. Fast als-based matrix factorization for explicit and implicit feedback datasets // In Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010. pp.: 71-78.

16. Sangita, O. An Improved K-Means Clustering Approach for Teaching Evaluation // International Conference on 41 Advances in Computing, Communication and Control (ICAC3 2011). Springer 2011. pp. 108-115

17. Wang J., Vries A., Reinders M. Unifying user-based and item-based collaborative filtering approaches by similarity fusion // In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006. pp.: 501-508

18. Wu, J. Advances in K-means Clustering: A Data Mining Thinking. Springer-Verlag Berlin Heidelberg, 2012. 178 p.

19. Zhou Y., Wilkinson D., Schreiber R., Pan R. Large-scale parallel collaborative filtering for the Netflix prize // In Algorithmic Aspects in Information and Management. Springer, 2008, pp.: 337-348.

20. Воронцов К. В. Математические методы обучения по прецедентам (теория обучения машин) // URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf (дата обращения: 21.05.2021)