Федеральное государственное бюджетное образовательное учреждение высшего образования
ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Институт (НОЦ) технических систем и информационных технологий (ИТСИТ)

Кафедра систем обработки информации, моделирования и управления (СОИМУ)

Программная инженерия (ПИ)

К ЗАЩИТЕ ДОПУСТИТЬ
И.о. заведующий кафедрой СОИМУ
к.т.н.,

_____/В.М. Татьянкин /
(подпись)

_____
(дата)

СИСТЕМА ПОИСКА АБИТУРИЕНТОВ НА БАЗЕ ТЕХНОЛОГИЙ NLP И МАШИННОГО ОБУЧЕНИЯ

Выпускная квалификационная работа бакалавра

Студент гр. (1541б) _____/А.М. Захаров/
(подпись)

Руководитель _____/В.В. Бурлуцкий/
(подпись)

Ханты-Мансийск – 2018

Реферат

Выпускная квалификационная работа, 59 с., 13 рис., 7 табл., 18 источников, 14 прилож.

Ключевые слова: Python, NLP, база данных, машинное обучение, веб-приложение.

Объектом разработки является проектирование и разработка алгоритма для альтернативного метода привлечения абитуриентов, в связи с существующей конкуренцией между высшими учебными заведениями.

Цель работы – проектирование и разработка алгоритма для системы поиска абитуриентов на базе технологий NLP и машинного обучения.

В процессе работы производилось определение требований к сервису и его реализация.

Экономическая эффективность и значимость работы обусловлена тем, что разрабатывается алгоритм для альтернативного метода привлечения абитуриентов, требующий меньше трудозатрат по сравнению с традиционными методами.

Результат: в результате разработки был спроектирован и протестирован алгоритм привлечения абитуриентов, получен работающая система поиска абитуриентов.

Оглавление

Список сокращений и специальных терминов

NLP – Natural language processing – общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Применительно к искусственному интеллекту анализ означает понимание языка, а синтез – генерацию грамотного текста. [12]

Таргетинг – рекламный механизм, позволяющий выделить из всей имеющейся аудитории только ту часть, которая удовлетворяет заданным критериям (целевую аудиторию), и показать рекламу именно ей [12].

Датасет – идентифицированная совокупность физических записей, организованная одним из установленных в системе обработки данных способов и представляющая файлы или части файлов в среде хранения [12].

вуз – высшее учебное заведение [9]

Парсер (от англ. *parser*) или **граббер** (от англ. *grabber*) – программное обеспечение, предназначенное для анализа и разбора исходных данных, с целью их обработки и дальнейшей использования в требуемом виде. [9]

Парсинг – автоматический анализ и сбор данных по заданному признаку. [10]

Целевая группа, целевая аудитория – термин, используемый в маркетинге или рекламе для обозначения группы людей, объединенных общими признаками, или объединенной ради какой-либо цели или задачи. [10]

БД – база данных – совокупность взаимосвязанных данных, организованных в соответствии со схемой базы данных таким образом, чтобы с ними мог работать пользователь. [10, 4]

СУБД – Система управления базами данных - совокупность программ и языковых средств, предназначенных для управления данными в базе данных,

ведения базы данных и обеспечения взаимодействия ее с прикладными программами. [10, 4]

Система – Система поиска абитуриентов на базе технологий NLP и машинного обучения.

ВКР – выпускная квалификационная работа.

API – Application Programming Interface – это интерфейс программирования, интерфейс создания приложений [13, 10].

Вконтакте – социальная сеть.

ПО – программное обеспечение.

Введение

В 2008 году ученые из университета Хэриот-Уотт (Эдинбург) во главе с профессором Адрианом Нортом, главой кафедры прикладной психологии, решили проверить, связаны ли музыкальные предпочтения с интеллектом и характером слушателей [2]. Их исследования выявили очень интересную особенность: самые высокие результаты IQ-тестов – у любителей тяжелой музыки и рока. На основе данных о том, что музыка взаимосвязана с уровнем интеллекта, вывели гипотезу, что она так же взаимосвязана с предрасположенностью к какому-либо направлению обучения. Для подтверждения данной гипотезы велась разработка программного обеспечение, которое на основе выборки студентов по разным направлениям, находит зависимости между ними.

В настоящее время резко стоит проблема качества поступающих абитуриентов. Для повышения рейтинга передовые вузы переходят от традиционных методов привлечения абитуриентов к новым, прогрессивно развивающимся методам, таким как привлечение через социальные сети. Выездные встречи со школьниками, дни открытых дверей, проведение олимпиад на базе университета, конференции не решают полностью проблему и слишком трудозатратны. «Худший формат дня открытых дверей, который можно придумать, – посадить в фойе тетушек пенсионного возраста с буклетами вуза. Это будет первый и последний день, когда абитуриент переступит порог этого института», – рассказывает психолог, карьерный консультант, программный директор Школы Осознанного Развития Елена Рагозина. «Современные дети не понимают ту информацию, которую им доносят на днях открытых дверей, проходящих по классической схеме. Поэтому побеждают ВУЗы, распространяющие информацию о себе заблаговременно и не в традиционных направлениях» [5].

Метод поиска абитуриентов через социальные сети позволит университету выходить за границы своего расположения и активно искать, и привлекать талантливые умы на территориях соседних регионов без существенных финансовых издержек.

Поэтому цель данной работы заключается в разработке алгоритма для системы поиска абитуриентов на базе технологий NLP и машинного обучения.

Для достижения поставленной цели необходимо решить список задач:

1. Разработать алгоритм для Системы на основе машинного обучения.
2. Подготовить датасет для обучения.
3. Обучить модель на собранном датасете.
4. Спроектировать Систему.
5. Разработать Систему.

Данные этапы подробно расписаны в главах представленной работы.

1. Обзор предметной области поиска абитуриентов на базе технологий NLP и машинного обучения

1.1 Описание предметной области поиска абитуриентов на базе технологий NLP и машинного обучения

Между университетами происходит конкуренция за лучшие умы. На сегодняшний день финансирование вузов зависит от результатов приемной компании. Для вуза важен каждый студент, но если этот студент учится на направлении, к которому у него нет предрасположенности, то вероятность его отчисления из вуза велика. Поэтому для создания списка первоочередных абитуриентов нужен их анализ, который мы будем брать из социальной сети Вконтакте, на основе музыки, которую он слушает.

Многие учебные заведения, не исключая Югорский Государственный Университет (далее ЮГУ) используют традиционные методы привлечения абитуриентов: олимпиады на базе университета, конференции, дни открытых дверей, выездные встречи со школьниками. Данные методы не эффективны и трудозатратны.

Для повышения рейтинга ЮГУ и его конкурентоспособности на рынке учебно-научной деятельности необходимо переходить на развивающиеся методы привлечения абитуриентов.

В качестве источника возможных абитуриентов будем использовать социальные сети, в них коммуницирует большая часть молодежи. Поиск абитуриентов менее затратен и расширяет территорию привлечения.

Стратегия метода проста: выявить школьников с предрасположенностью к определенному направлению и пригласить его на обучение.

На основе описания предметной области были сформулированы требования к Системе, описанные в следующем пункте.

1.2 Формулировка требований к системе поиска абитуриентов на базе технологий NLP и машинного обучения

Разрабатываемая Система должна соответствовать следующим функциональным требованиям заказчика:

1. Составлять классификацию абитуриентов
2. Собирать данные из социально сети Вконтакте
3. Строить визуализацию модели обучения
4. Обучение модели на первичных данных
5. Переобучение модели на новых данных
6. Обработка названий музыкальных групп технологиями NLP
7. Загрузка идентификаторов пользователей из файла в формате JSON

Так же были составлены нефункциональные требования Системы:

1. Работоспособность Системы на оборудовании:
   1.1. ОЗУ – 6 Гб
   1.2. ОС – Windows 10
   1.3. Разрядность системы - x64
   1.4. Частота процессора – 2 ГГц
   1.5. Количество ядер процессора – 2
2. Требование к персоналу:
   2.1. Уровень пользования ПК – уверенный
3. Требования к пользовательскому интерфейсу:
   3.1. Минималистичный интерфейс
   3.2. Интерфейс с использованием микро-фреймворка flask

После составления требований Системы, был произведен выбор социальной сети, описанный в следующем разделе.

1.3 Выбор социальной сети для анализа

По данным ВЦИОМ (всероссийский центр изучения общественного мнения) самой популярной социальной сетью в России является ВКонтакте. Около 42% российских интернет-пользователей ежедневно пользуются Вконтакте из них: 78% среди 18-24-летних, 54% среди 25-34-летних. 27% интернет-пользователей используют - «Одноклассники» из них большая часть 60-летние и старше – 40%. [14] Ежемесячная аудитория «ВКонтакте» составляет более 97 000 000 человек. Так как для поиска абитуриентов нужны выпускники вуза и сами поступающие (17-24 года), то на основе среднего удовлетворяющего возраста будем использовать Вконтакте.

На основе выбранной социальной сети проведем обзор аналогов Системы, представленный в следующем пункте работы.

1.4 Обзор аналогов системы поиска абитуриентов на базе технологий NLP и машинного обучения

Выявлен один прямой аналог Национального исследовательского Томского Государственного университета, но данный продукт строит целевую модель поступающего не только по музыке, но и по «следам», оставленным в социальной сети Вконтакте, подробности не разглашаются. Косвенными аналогами Системы являются программы для таргетинга. Так как был выявлен только один прямой аналог, то был произведен поиск косвенных аналогов, занимающихся схожим функционалом.

Для сравнения аналогов были выявлены следующие критерии:

1. Цена

2. Работа с API VK
3. Тип ПО
4. Скорость обработки
5. Машинное обучение (Анализ данных)

На основе исследования аналогов с учетом требований были выявлены следующие:

1. Проект: «Поиск и привлечение абитуриентов с высоким потенциалом через социальную сеть» Национального исследовательского Томского Государственного университета (далее Проект)
2. Сервис поиска целевой аудитории ВКонтакте «vk.barkov.net»
3. «Vk Audio Parser v2.2»
4. «Яндекс.Музыка»
5. «Pepper.ninja»

В ходе сравнения аналогов, была составлена сравнительная таблица 1:

Таблица 1. – Сравнение аналогов

Критерии/ПО	Цена (руб/мес.)	Работа с музыкой	Тип ПО	Скорость обработки (пользователей/мин.)	Анализ данных
Проект	–	–	Неизвестен	Неизвестна	✓
vk.barkov.net	390	✓,– (Только со страницы профиля)	Web-сервис	3000	–
Vk Audio Parser v2.2	1200	✓	Настольное ПО	150	–
Pepper.ninja	690	–	Web-сервис	2000	–
Яндекс.Музыка	100	✓	Web-сервис	Персонально	✓

Данные таблицы актуальный на 15 июня 2018 г.

Одним из главных критериев был анализ данных, как видно из таблицы сравнения данному критерию удовлетворяют «Яндекс.Музыка» и «Проект».

Из описания Проекта, реализуется схожая функция – поиск абитуриентов по мотивации к изучению предметов определенного направления (гуманитарные, точные и естественные науки).

«Яндекс.Музыка» на базе машинного обучения анализирует список музыки пользователя, его настроение и строит ежедневный трек лист. Так же Яндекс проводил исследование «У каждого поколения – своя музыка. Или нет», в котором анализировалась музыка для каждого поколения. В результате была выявлена зависимость, подробная информация в источниках литературы [15].

Настольное программное обеспечение «Vk Audio Parser v2.2» работал с музыкой через API VK, на данный момент функционал API VK AUDIO закрыт, программа перестала работать. Интерфейс программы оставляет желать лучшего. Для работоспособности программы не нужен сервер.

«vk.barkov.net» и «Pepper.ninja» имеют приятный интерфейс и быструю скорость обработки. Скорость можно поднять, за счет разных тарифных планов.

Проанализировав аналоги было принято решение разрабатывать web-сервис.

Закончив с обзором предметной области, переходим к следующей главе, проектированию Системы.

2. Проектирование системы поиск абитуриентов на базе технологий NLP и машинного обучения

Проектирование Системы является неотъемлемым этапом в жизненном цикле программного обеспечения. На данном этапе происходит проектирование алгоритмов для системы, структуры базы данных, пользовательских интерфейсов, архитектурного решения системы, определения технологий и инструментов для разработки. Первым шагом обоснуем выбор жизненного цикла Системы.

2.1 Выбор жизненного цикла системы поиска абитуриентов на базе технологий NLP и машинного обучения

Разработка начинается с выбора модели жизненного цикла программного обеспечения. Воспользовавшись рекомендациями Института качества программного обеспечения SQI были заполнены таблицы, на основе которых был сделан вывод о необходимой модели, Приложение Б.1-Б.4 [6].

На основании результатов, представленных в таблицах в Приложении Б.1-Б.4 для проектируемой Системы преимущественно подходит инкрементная модель жизненного цикла.

В инкрементной модели полные требования делятся на различные версии ПО. Каждая версия – это определенный этап сборки общего ПО. Каждый модуль проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования. Процесс циклов будет продолжаться до тех пор, пока не будут добавлены все необходимые функции и ПО не будет полностью готово. Схема инкрементной модели представлена на рисунке 2.1.

Рисунок 2.1 – Инкрементная модель

Разработка программного обеспечения подразумевает под собой 3 фазы со своими подзадачами:

- Работа с базой данных
    1. Проектирование БД
    2. Первоначальная разработка парсера
    3. Сбор датасета
    4. Очистка данных
- Построение модели ПО
    1. Проектирование модели ПО
    2. Доработка парсера
    3. Доработка очистки данных
    4. Машинное обучение
- Готовый сервис
    1. Сбор всех модулей в один сервис

После выбора жизненного цикла Системы переходим к проектированию алгоритмов, описанных в следующем пункте.

2.2 Проектирование алгоритма системы поиска абитуриентов на базе технологий NLP и машинного обучения

В ходе проектирования было выявлено два модуля системы: Парсер (загрузка информации из социальной сети «Вконтакте») и Классификации (модуль классификации предрасположенности абитуриентов). Для классификации абитуриентов был составлен общий алгоритм Системы, рисунок 2.2:

1. Загрузка файл с идентификационными номерами пользователей Вконтакте.
2. Работа парсера, для сбора данных из Вконтакте.
3. Если стоит задача обучить систему, то запускается алгоритм построения модели на новых данных занесенных данных.
4. Если стоит задача поиска, то используется уже готовое дерево принятия решений для классификации.
5. Получение результата работы программы.

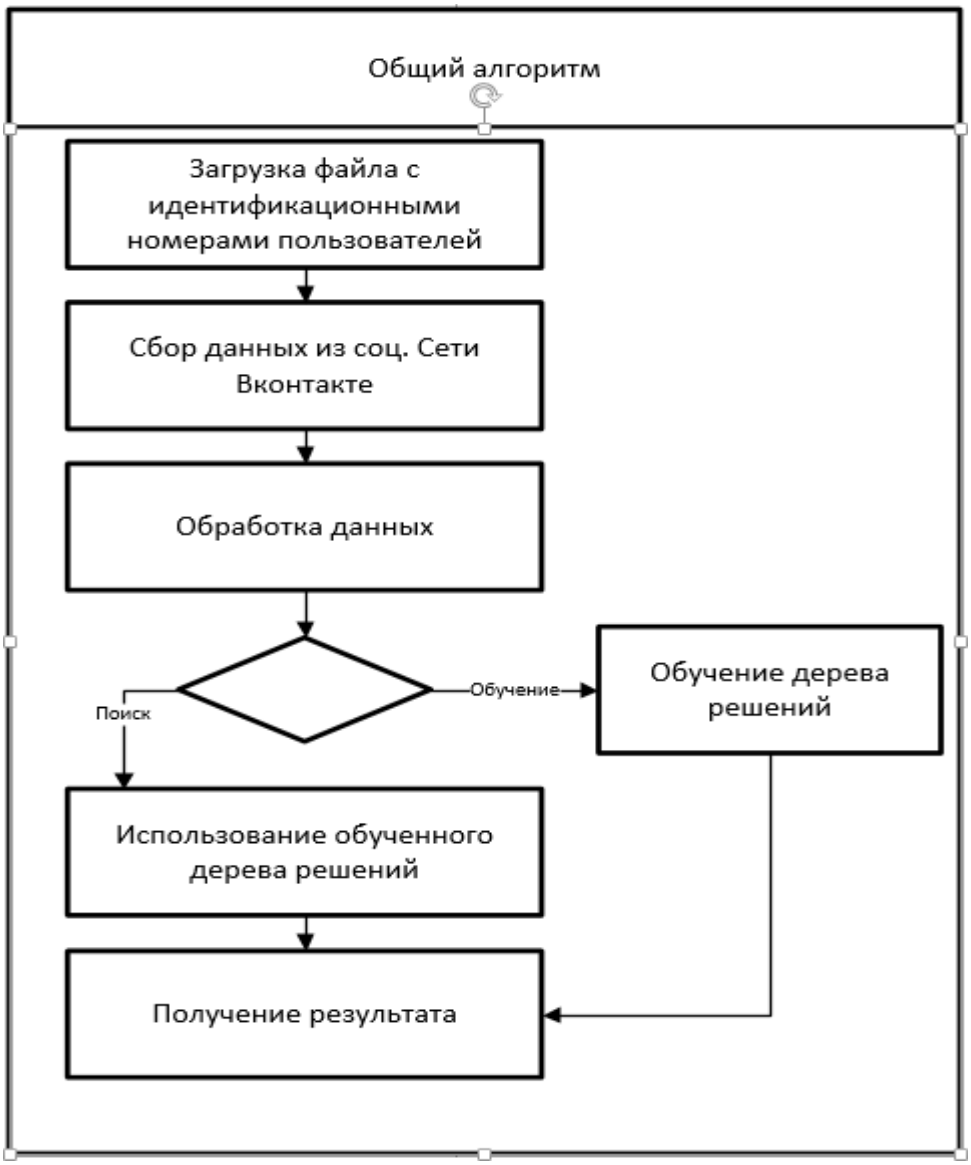

Рисунок 2.2 - Общий алгоритм работы программы

Для модуля Парсер спроектирован алгоритм подготовки датасета, представленный на рисунке 2.3:

1. Сбор данных аудио исполнителей у пользователей социальной сети Вконтакте.
2. Визуальный обзор собранных данных, для выявления стоп-слов.
3. Составление списка стоп-слов
4. Предварительная очистка данных при помощи регулярных выражений.

5. Классификация данных по лексемам
6. Очистка данных алгоритмом Левенштейна [12], если разница в один изменяемый символ.
7. Очистка по лексемам, по схожести
8. Занесение очищенных данных в базу.
9. Построение новой связи аудио исполнителя связь с пользователем.

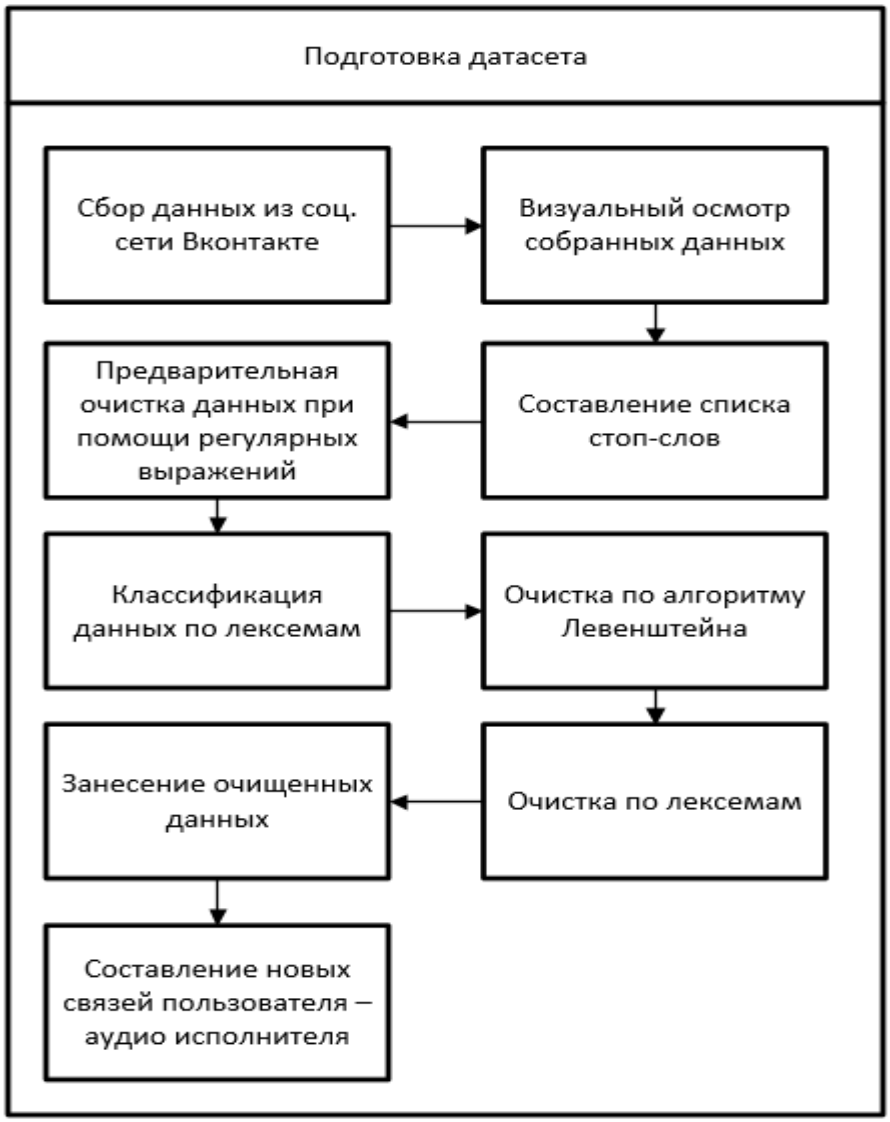

Рисунок 2.3 – Алгоритм подготовки датасета

Для модуля Классификации был спроектирован алгоритм создания таблицы для обучения, рисунок 2.4:

1. Получаем данные из БД по всем пользователям для обучения у которых категория факультета удовлетворяет запросу.
2. Получаем данные из БД по всем музыкальным исполнителям, с условием что они встречаются больше чем у 10 пользователей и менее чем у 1000.
3. Записываем данные в файл для классификации.
4. Получение из БД для каждого пользователя личного списка музыки.
5. Проверка пересечения музыки пользователя со всей музыкой.
6. Если встречается, то заменяем поле на 1 в строке пользователя.
7. Если у пользователя нет ни одного аудио исполнителя из общей выборки, то не записываем его в таблицу.
8. В последний столбец таблицы вставляем категорию пользователя.
9. Запись полученной таблицы в файл для дальнейшего использования.

Рисунок 2.4 – Алгоритм составления таблицы

Так же для модуля Классификации был спроектирован алгоритм обучения дерева решений, рисунок 2.5:

1. Чтение по частям файл с таблицей для обучения.
2. Получение значений для обучения и категории пользователей.
3. Разделение выборки на обучение и тестирование.
4. Перекрестной проверкой (Кросс-валидацией) разбиваем данные на 5 перемешанных частей [17].
5. Обучаем дерево решений с глубиной до 30.
6. Получаем численную оценку качества модели, а именно долю по которым модель приняла правильное решение (Accuracy) [11].
7. Получаем F-меру – средневзвешенное между полнотой и точностью [11].
8. Запись обученного дерева в файл для дальнейшего использования.

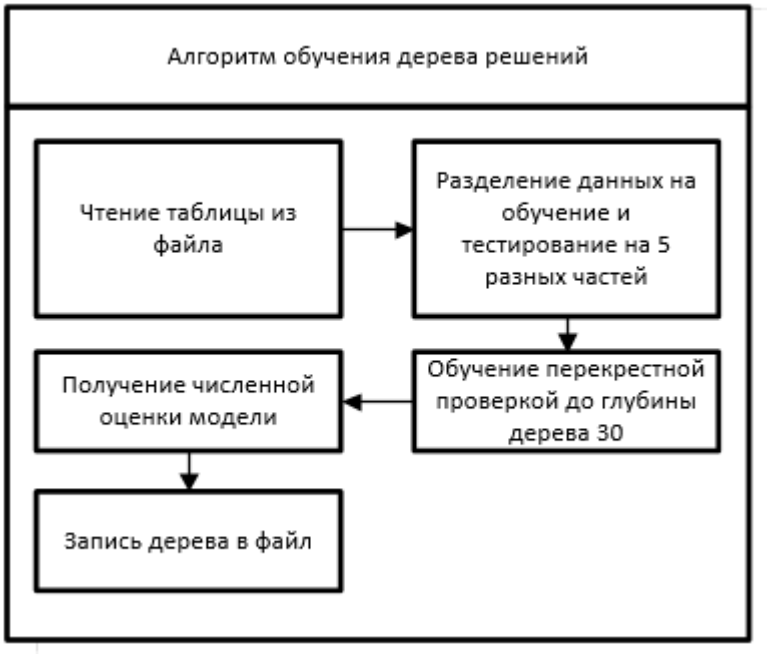

Рисунок 2.5 – Алгоритм обучения дерева решений

После проектирования алгоритмов Системы перешли к проектированию базы данных.

2.3 Проектирование базы данных сервиса поиска абитуриентов на базе технологий NLP и машинного обучения

Модель сущность-связь (англ. entity-relationship model, ERM, ER-модель) позволяет описывать концептуальные схемы предметной области [4].

ER-модель используется для высокоуровневого проектирования баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

ER-диаграмма представляет графическую структуру данных проектируемой БД. Сущности отображаются при помощи прямоугольников, таблиц, содержащих имя сущности – таблицы БД. Взаимосвязи сущностей отображаются в виде линий, соединяющих отдельные сущности.

Взаимосвязь показывает, что данные одной сущности ссылаются или связаны с данными другой.

В ходе проектирования базы данных была составлена ER - модель, рисунок 2.6 описание сущностей описано ниже.

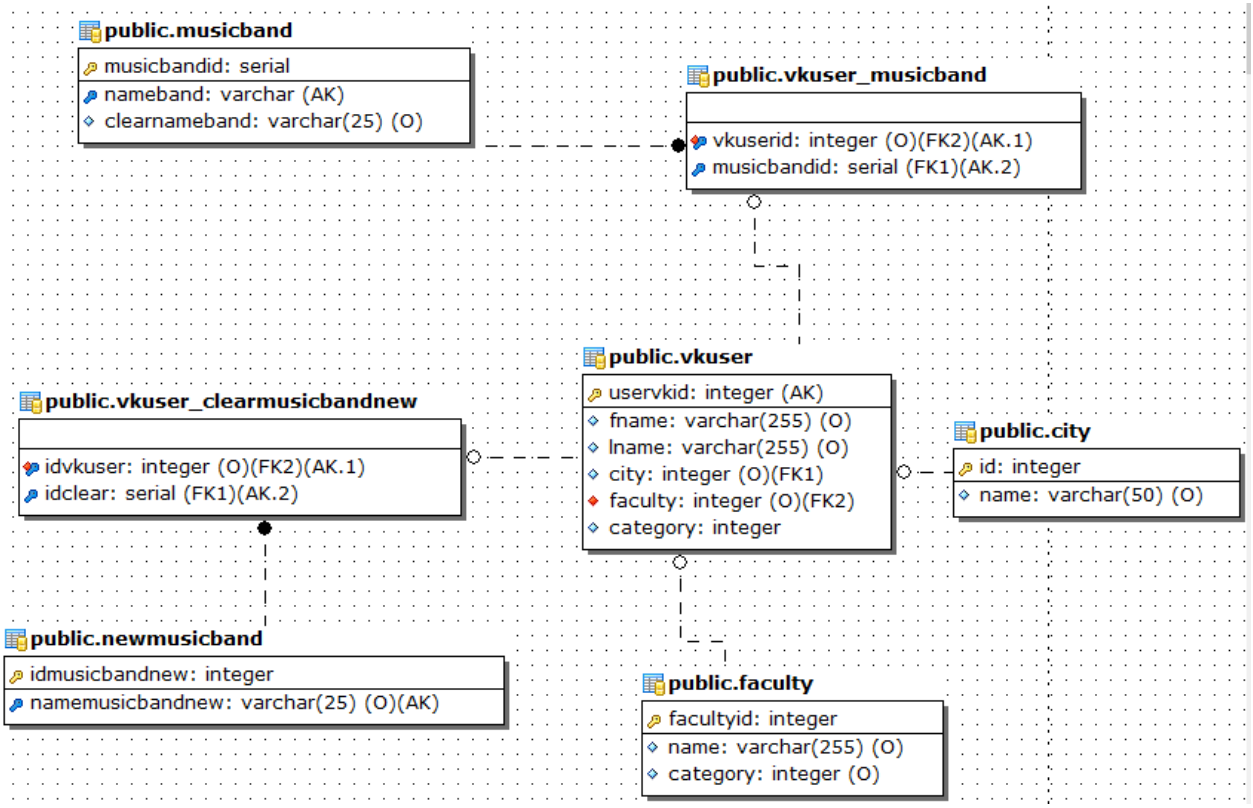

Рисунок 2.6 – ER - модель.

Описание таблиц БД:

vkuser - таблица пользователя Вконтакте:

- uservkid – идентификационный номер пользователя Вконтакте
- fname – имя пользователя Вконтакет
- lname – фамилия пользователя Вконтакте
- city- - идентификационный номер города пользователя Вконтакте
- faculty - идентификационный номер факультета пользователя Вконтакте
- category – категория пользователя – обучающая или поисковая.

musicband – таблица аудио исполнителей из Вконтате:

- musicbandid - идентификационный номер аудио исполнителей Вконтакте
- nameband - название очищенного аудио исполнителя Вконтакте
- clearnameband – пред очищенные данные аудио исполнителя Вконтакте

vkuser_clearmusicband – таблица реализующая связь многие ко многим:

- vkuserid – идентификационный номер пользователя Вконтакте
- musicbandid – идентификационный номер аудио исполнителей Вконтакте

newmusicband – таблица очищенных аудио исполнителей:

- idmusicband - идентификационный номер очищенного аудио исполнителей Вконтакте
- namemusicbandnew - название очищенного аудио исполнителя Вконтакте
- tag – поле для жанра музыки исполнителя из сервиса last.fm

vkuser_clearmusicbandnew – таблица реализующая связь многие ко многим:

- idvkuser – идентификационный номер пользователя Вконтакте
- idclear – идентификационный номер очищенного аудио исполнителей Вконтакте

city - таблица описания городов:

- id – идентификационный номер города
- name – наименование города

facukty - таблица факультетов:

- facultyid - идентификационный номер факультета пользователя Вконтакте
- name - название факультета
- category – категория распределения факультета

Закончив с проектированием базы данных, спроектировали физическое представление Системы, описание в следующем разделе.

2.4 Проектирование физического представления сервиса поиска абитуриентов на базе технологий NLP и машинного обучения

Форма физического представления программной системы – это диаграмма развертывания (размещения). Необходима для полного физического представления системы. Диаграмма развертывания представлена на рисунке 2.7

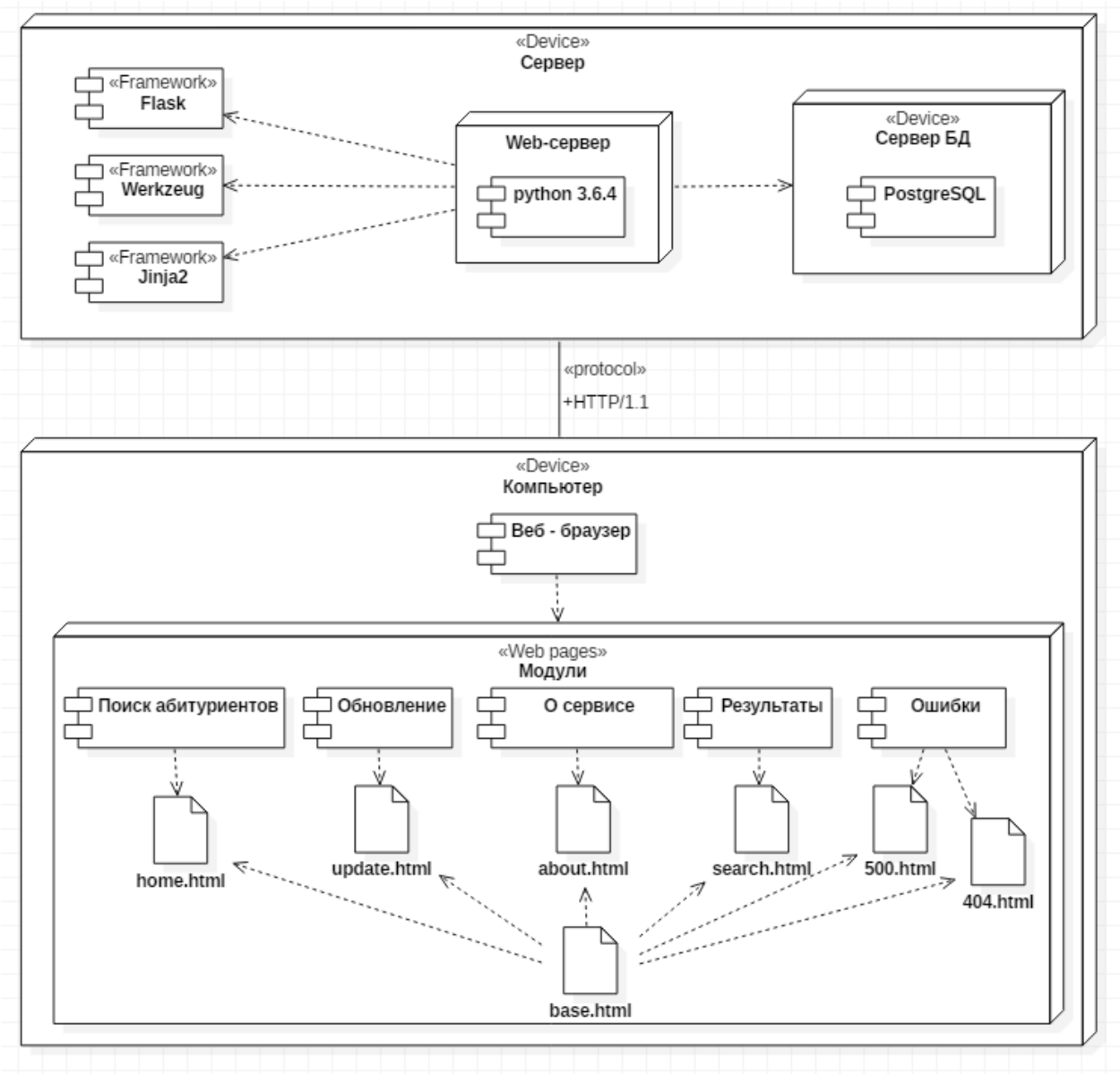

Рисунок 2.7 – Диаграмма развертывания.

Диаграмма развертывания применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений – маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы [10].

Клиент-серверная архитектура разделяет всю работающую систему на три различные части, определенным образом взаимодействующие между собой:

- клиентская приложение – это программа, работающая на компьютере пользователя и обеспечивающая интерактивное взаимодействие системы поиска абитуриентов с пользователем;
- серверная python – обеспечивает взаимодействие между пользователями и системой управления базами данных в клиент-серверном варианте работы.
- сервер базы данных – специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных.

Программа, работающая у пользователя, (клиентское часть) взаимодействует с сервером python, а она, при необходимости, обращается к серверу баз данных.

При этом физически сервер python и сервер баз данных могут располагаться как на одном компьютере, так и на разных. Это позволяет администратору при необходимости распределять нагрузку между серверами.

Для передачи данных между клиентскими приложениями и сервером, используется HTTP/1.1.

Закончив с проектированием переходим к следующей главе, разработке.

3. Разработка сервиса поиска абитуриентов на базе технологий NLP и машинного обучения

3.1 Выбор языка программирования системы поиска абитуриентов на базе технологий NLP и машинного обучения

Одной из задач ВКР является выявление закономерностей среди абитуриентов при помощи машинного обучения. На сегодняшний день, это направление отлично развито в высокоуровневом языке программирования (далее ЯП) Python. Данный ЯП имеет много вспоминающих библиотек для машинного обучения, поддерживается как front-end, так и back-end разработка. Для выполнения поставленных задач python наилучшее решение.

Для создания веб-приложения был использован микро-фреймворк Flask. Быстрый, простой, не имеет ни чего лишнего, нагружающего серверное оборудование, использует шаблонизатор Jinja2 и набор инструментов библиотеки Werkzeug для HTTP запросов.

Так как API VK AUDIO закрыты, то необходим сборщик данных. Для данного был выбран популярный инструмент для автоматизации браузера - Selenium.

Преимущества всего стека-технологий – бесплатно распространяемое ПО, большой объем информации в сети Интернет.

Выбрав стек технологий, переходим к выбору СУБД, описание представлено в следующем пункте.

3.2 Выбор СУБД для системы поиска абитуриентов на базе технологий NLP и машинного обучения

Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте, а столбцы таблицы содержат различные характеристики этих объектов – атрибуты [4]

Выбор системы управления баз данных (СУБД) представляет сложную задачу. Один из важнейших этапов разработки баз данных. Выбранный программный продукт должен удовлетворять многим критериям: текущим потребностям, будущим потребностям, финансовым затратам на приобретение оборудования, программного обеспечения и его использования.

При разработке тестировались схожие СУБД: MySQL, Oracle, PostgreSQL.

Преимущества и недостатки MySQL: К преимуществам относится: высокая безопасность, гибкость и простота в обращении, много готовых решений. К недостаткам относится: частые сбои при работе с одновременными операциями, малый функционал обработки текста.

Oracle и PostgreSQL профессиональные СУБД, с меньшим количеством недостатков, чем MySQL, способные расширяться за счет написания своих функций, огромный список возможностей, огромные сообщества в сети Интернет. Главный минус Oracle – это цена. Поэтому в целях написания ВКР использовалась PostgreSQL.

Сервис поиска абитуриентов будет работать на СУБД PostgerSQL. Поэтому в рамках выпускной квалификационной работы будет использоваться она. Графические оболочки для работы с СУБД PostreSQL: HeidiSQL и pgAdmin 4. Данный выбор заключается в том, что они хорошо дополняют друг друга. PgAdmin 4 хорош для создания самой базы данных и ее настройки. HeidiSQL удобен для резервирования БД, просмотра и редактирования данных. Так же был

рассмотрена графическая оболочка dbForge Studio Express for PostgreSQL. Данный продукт находится на стадии разработки и функционал продукта очень скуден, на момент написания ВКР был возможет лишь просмотр данных. Проектирование БД проводилось в сервисе dbdesigner.net и SQL Manager for PostgreSQL.

После выбора СУБД, был произведен выбор модели машинного обучения, описанный в следующем пункте.

3.3 Выбор модели для обучения системы поиска абитуриентов на базе технологий NLP и машинного обучения

Разрабатываемая Система предполагает классификацию пользователей по нескольким признакам. Деревья принятия решений – одна из моделей выполняющая поставленную задачу [3,7]. Цель дерева состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной путем изучения простых правил принятия решений, полученных из данных. Деревья имеют рад преимуществ: контролируемый метод обучения, простота в использовании, возможность визуализации, поддержка категориальных и числовых признаков. Для построения дерева решений использовалась библиотека Python для машинного обучения scikit-learn [3].

Для проверки дерева решений используют перекрестную проверку(Кросс-валидацию). Кросс-валидация - метод оценки аналитической модели и её поведения на независимых данных [17]. Использовалась кросс-валидация по K блокам (K-fold cross-validation). В этом случае исходный набор данных разбивается на K одинаковых по размеру блока. Из K блоков один оставляется для тестирования модели, а остающиеся K-1 блока используются как тренировочный набор. Процесс повторяется K раз, и каждый из блоков используется один раз как тестовый набор. Получаются K результатов, по

одному на каждый блок, они усредняются. Преимущество такого способа перед случайным сэмплированием (random subsampling) в том, что все наблюдения используются и для тренировки, и для тестирования модели, и каждое наблюдение используется для тестирования в точности один раз.

Закончив с выбором модели машинного обучения, переходим к разработке визуального интерфейса, описанного в следующем пункте.

3.4 Разработка интерфейса сервиса поиска абитуриентов на базе технологий NLP и машинного обучения

В ходе написания выпускной квалификационной работы был разработан минимальный интерфейс сервиса при помощи микро-фреймворка – Flask.

В сервисе присутствует четыре вкладки, «Главная» (Поиск абитуриентов), «Результаты», «О сайте», «Обновить». Интерфейс представлен на рисунках 3.1-3.4 ниже. Код базовой страницы для шаблона представлен в Приложении А.

На «Главной» и «Обновить» страницах расположены две кнопки – запуск поиска и выбор файла из файловой системы пользователя.

На странице «О сайте» расположено описание с сервиса, и возможная работа с ним.

На странице «Результаты» представлены собранные результаты, визуальная модель дерева решений и таблица классификации.

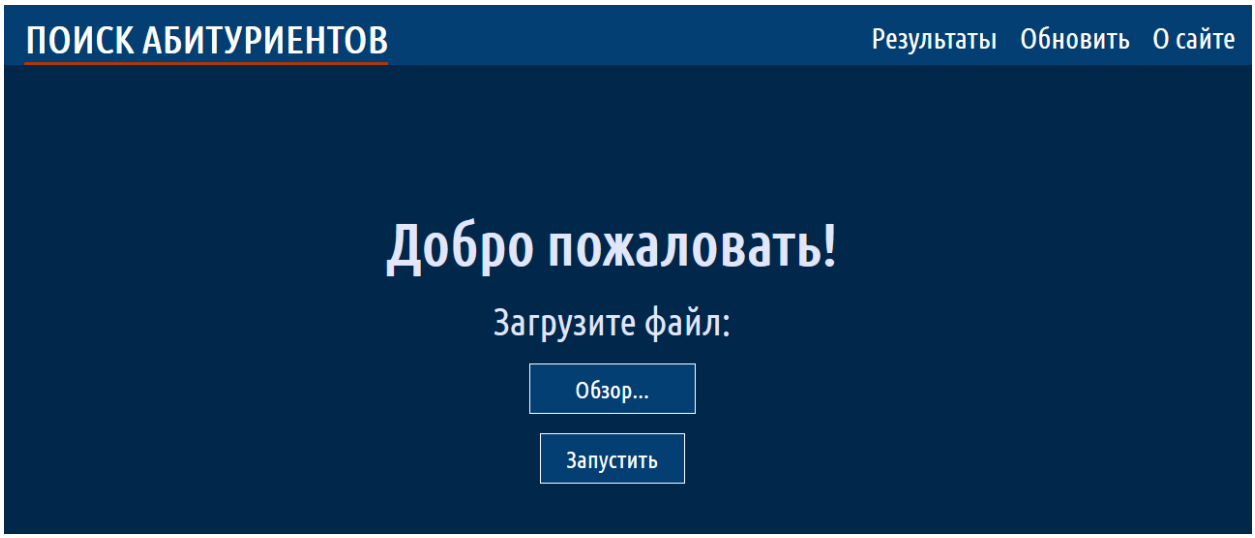

Рисунок 3.1 – Главная страница

Рисунок 3.2 – Страница о сервисе

Рисунок 3.3 – Страница о результатов

Рисунок 3.4 – Страница результатов с таблицей

Закончив с разработкой пользовательского минимального интерфейса, переходим к разработке подели обучения

3.5 Разработка серверной части системы поиска абитуриентов на мазе технологий NLP и машинного обучения

При разработке модели обучения использовался ранее указанный алгоритм обучения. Листинг кода представлен в Приложении В.1.

При разработке сборщика таблицы для обучения использовался ранее указанный алгоритм сбора таблицы для обучения. Листинг кода представлен в Приложении В.2.

Листинг кода для разработки парсера пользовательских данных представлен в Приложении В.3.

Для очистки музыкальных исполнителей использовался код представленный в Приложении В.4.

Для распределения факультетов на категории использовался код представленный в Приложении В.5.

Для реализации взаимодействий пользователя с Системой был написан код Представленный в Приложении В.6

Для очистки музыкальных аудио исполнителей по алгоритму Левенштейна использовался следующий код, представленный в Приложении В.7.

Закончив с разработкой серверной части приложения, переходим к следующей главе, тестированию.

4. Тестирование сервиса поиска абитуриентов на базе технологий NLP и машинного обучения

4.1 Тестирование интерфейса сервиса поиска абитуриентов на базе технологий NLP и машинного обучения фокус-группой

Для фокус-группы из 20 человек была составлена анкета с вопросами, Приложение В.1 и прототип тестируемого интерфейса (рисунки ранее 3.1-3.4). Фокус-группа состояла из разновозрастных участников с разным родом деятельности. Результаты тестирования представлены в Приложении В.2.

Исходя из результатов анализа анкетирования, программный продукт был хорошо принят фокус группой. На основе оценок и пожеланий анкет выявлены проблемные участки сервиса, которые были исправлены.

4.2 Тестирование модели системы поиска абитуриентов на базе технологий NLP и машинного обучения фокус-группой

В ходе написания ВКР было произведено тестирование обучающей модели, для выявления оптимальных параметров дерева решений, при которых оно не будет переобучаться и будет высокий процент качества классификации.

В ходе тестирования было выявлено, что на собранных данных наилучшее обучение достигается при глубине дерева не более 30. Количество разбиений для перекрестного обучения 5.

Была произведена численная оценка качества алгоритма [11]:

1. Меткость – доля элементов по которым было принято правильное решение к общему количеству элементов: Accuracy = P/N, где N – общее количеств элементов, P – количество элементов по которым было принято правильное решение. Такая оценка не является точной, так как может быть

спрогнозировано, верно, по нескольким классам и совсем не спрогнозировано по другим.

2. Точность и полнота. Точность – доля элементов, принадлежащих данному классу относительно всех элементов, которые система отнесла к этому классу. Полнота – доля найденных элементов, принадлежащих классу относительно всех элементов данного класса в тестовой выборке. Формулы для расчета представлены на рисунке 4.1.

- TP — истино-положительное решение;
- TN — истино-отрицательное решение;
- FP — ложно-положительное решение;
- FN — ложно-отрицательное решение.

Тогда, точность и полнота определяются следующим образом:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Рисунок 4.1 – Формулы расчета точности и полноты

3. F-мера – гармоническое среднее между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю. Формулы для расчета представлены на рисунке 3.13

$$F = 2\frac{Precision \times Recall}{Precision + Recall}$$

Рисунок 4.2 – F-мера.

Библиотека scikit-learn позволяет произвести все вычисления численных оценок, представленных выше. Но так как F-мера является неким средним

результатом.

При первоначальном тестировании алгоритма использовалась таблица для обучения по всем аудио исполнителям(195000), пользователи (2000). Процент меткости при данной выборке 20%, в последствии была изменена выборка аудио исполнителей по популярности в разных направлениях музыки. При данной выборке (460) процент меткости алгоритма вырос до 27%. Дальнейшая выборка была произведена с условием что музыкальные исполнители встречаются у минимум 10 человек и не более чем у 1000. В результате обучения получили 61% меткости модели – это наилучший результат, полученный в ходе выполнения работы. Лучшая F-мера для класса гуманитарных наук получили 75%, для технических 39%.

Для более точной классификации нужно собрать больше данных.

Заключение

В ходе выполнения выпускной квалификационной работы были рассмотрены различные методы для привлечения потенциальных абитуриентов. Выяснено, что социальные сети являются мощным инструментом для поиска и привлечения абитуриентов и могут использоваться при проведении приемных компаний.

В основе предлагаемого алгоритма лежит предположение о том, что музыкальные предпочтения потенциального абитуриента может совпадать с интересами действующих студентов старших и выпускных курсов.

Разработка и тестирование алгоритма подтверждает его работоспособность, но предложенный алгоритм выявления абитуриентов нуждается в доработке. Для развития алгоритма нужно больше собранных данных из социальной сети Вконтакте.

Из-за низкой эффективности работы алгоритма был предложен альтернативный метод для поиска потенциальных абитуриентов по целевому направлению подготовки. Данный метод заключается в расширении собираемых данных пользователя социальной сети Вконтакте, такие как фотографии пользователя, видеозаписи пользователя, текстовая информация со страницы пользователя, посты пользователя. Главным недостатком данного метода является его громоздкость и неполнота собираемых данных.

Разработанный алгоритм метод может быть хорошим дополнениям к традиционным способам поиска абитуриентов, поскольку является менее затратным, чем проведение очных встреч, также пропадают территориальные ограничения.

Список использованных источников

1. 8 Surprising Ways Music Affects and Benefits our Brains – blog.bufferapp.com [Электронный ресурс] Режим доступа: https://blog.bufferapp.com/music-and-the-brain Дата последнего обращения: 18.05.2018
2. Adrian North and David Hargreaves, The Social and Applied Psychology of Music. OUP Oxford 2008 г – 488 с [Книжное издание]
3. Decision Trees. Scikit-learn Machine Learning in Python [Электронный ресурс] Режим доступа: http://scikit-learn.org/stable/modules/tree.html Дата последнего обращения: 15.06.2018
4. Академия Microsoft: Технология Microsoft ADO .NET. [Электронный ресурс] Режим доступа: https://www.intuit.ru/studies/courses/1163/199/info Дата последнего обращения 8.06.2018
5. Борьба за абитуриентов. Интервью от 03.03.2017. [Электронный ресурс] Режим доступа: http://msk.mr7.ru/borba-za-abiturientov/ Дата последнего обращения 8.06.2018
6. В. В. Бахтизин, Л. А. Глухова Технология разработки программного обеспечения. Учебное пособие. – Минск : БГУИР, 2010. – 267 с. : ил. [Книжное издание]
7. Деревья решений – общие принципы работы. BaseGroup Labs – Технологии анализа данных [Электронный ресурс] Режим доступа: https://basegroup.ru/community/articles/description Дата последнего обращения: 15.06.2018
8. Документация к Postgres Pro 9.5.13.1 [Электронный ресурс] Режим доступа: https://postgrespro.ru/docs/postgrespro/9.5/datatype-textsearch Дата последнего обращения: 8.06.2018
9. Научно-информационный «Орфографический академический ресурс АКАДЕМОС» Института русского языка им. В. В. Виноградова РАН. http://orfo.ruslang.ru/search/word

10. ООО «Техническая документация» разработка техдокументации. [Электронный ресурс] Режим доступа: http://tdocs.su/search?searchid=2292527&text=%D0%A1%D0%A3%D0%91%D0%94&web=0# Дата последнего обращения: 8.06.2018
11. Оценка классификатора (точность, полнота, F-мера). bazhenov.me [Электронный ресурс] Режим доступа: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html / Дата последнего обращения: 19.05.2018
12. Словари и энциклопедии на Академике. [Электронный ресурс] Режим доступа: https://xzsad.academic.ru/dic.nsf/ruwiki/43819 Дата последнего обращения 8.06.2018
13. Сообщество разработчиков Вконтакте. [Электронный ресурс] Режим доступа: https://vk.com/dev/ Дата последнего обращения: 8.06.2018
14. Социальные сети: кто туда ходит и зачем? Всероссийский центр изучения общественного мнения (ВЦИОМ) [Электронный ресурс] Режим доступа: https://wciom.ru/index.php?id=236&uid=116254 Дата последнего обращения: 01.04.2018
15. У каждого поколения – своя музыка. Или нет. Яндекс блог [Электронный ресурс] Режим доступа: https://yandex.ru/company/researches/2016/ya_music_and_age Дата последнего обращения 8.05.2018
16. Учителя слушают классику, а охранники – шансон. Исследовательский центр портала Superjob.ru [Электронный ресурс] Режим доступа: https://www.superjob.ru/community/life/61506/ Дата последнего обращения: 09.05.2018
17. Что такое кросс-валидация. Data Scientist # 1 [Электронный ресурс] Режим доступа: http://datascientist.one/cross-validation/ Дата последнего обращения: 15.06.2018
18. Электронный фонд правовой и нормативно-технической документации. ГОСТ 20886-85 Организация данных в системе обработки данных.

[Электронный ресурс] Режим доступа: http://docs.cntd.ru/document/1200015708 Дата последнего обращения 8.06.2018

ПРИЛОЖЕНИЕ А

Листинг базового шаблона интерфейса

```
<html lang="ru">
   <head>
      <meta charset="utf-8">
      <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }} ">
      <link href="https://fonts.googleapis.com/css?family=Amatic+SC:700&amp;subset=cyrillic"
rel="stylesheet">
      <link
href="https://fonts.googleapis.com/css?family=Ubuntu+Condensed&amp;subset=cyrillic"
rel="stylesheet">
      <script
        src="https://code.jquery.com/jquery-3.3.1.js"
        integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
        crossorigin="anonymous"></script>
      <title>ПОИСК</title>
   </head>
   <body>
    <header>
      <div class="header" >
        <p>
          <a href="/index">Поиск абитуриентов</a>
        </p>
      </div>
      <nav>
        <div class="topnav" id="myTopnav">
          <a href="/search">Результаты</a>
          <a href="#">Обновить</a>
          <a href="/about">О сайте</a>
        </div>
      </nav>
       <script type="text/javascript">
        function changeLink()
        {
          document.getElementById('mass').innerHTML="Начался процесс вычисления,
ждите.";
          document.getElementById('circle').style.display="block";
          document.getElementById('circle1').style.display="block";
          document.getElementById('text').style.display="block";
        }
       </script>
    </header>
    <main>
      {% block content %}{% endblock %}
    </main>
   </body>
</html>
```

ПРИЛОЖЕНИЕ Б.1

Таблица – Выбор модели жизненного цикла на основе характеристик требований

Номер критерия	Критерии категории требований	Ответ на критерий
1.	Являются ли требования к проекту легко определимыми и реализуемыми?	Нет
2.	Могут ли требования быть сформулированы в начале ЖЦ?	Да
3.	Часто ли будут изменяться требования на протяжении ЖЦ?	Нет
4.	Нужно ли демонстрировать требования с целью их определения?	Да
5.	Требуется ли проверка концепции программного средства или системы?	Нет
6.	Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ?	Да
7.	Нужно ли реализовать основные требования на ранних этапах разработки?	Да

ПРИЛОЖЕНИЕ Б.2

Таблица – Выбор модели жизненного цикла на основе характеристик команды разработчиков

Номер критерия	Критерии категории команды разработчиков проекта	Ответ на критерий
1.	Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Да
2.	Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков?	Да
3.	Изменяются ли роли участников проекта на протяжении ЖЦ?	Нет
4.	Является ли структура процесса разработки более значимой для разработчиков, чем гибкость?	Нет
5.	Важна ли легкость распределения человеческих ресурсов проекта?	Да
6.	Приемлет ли команда разработчиков оценки, проверки, стадии разработки?	Да

ПРИЛОЖЕНИЕ Б.3

Таблица – Выбор модели жизненного цикла на основе характеристик коллектива пользователей

Номер критерия	Критерии категории коллектива пользователей	Ответ на критерий
1.	Будет ли присутствие пользователей ограничено в ЖЦ разработки?	Да
2.	Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки?	Да
3.	Будут ли пользователи вовлечены во все фазы ЖЦ разработки?	Нет
4.	Будет ли заказчик отслеживать ход выполнения проекта?	Да

ПРИЛОЖЕНИЕ Б.4

Таблица – Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

Номер критерия	Критерии категории коллектива пользователей	Ответ на критерий
1.	Разрабатывается ли в проекте продукт нового для организации направления?	Да
2.	Будет ли проект являться расширением существующей системы?	Да
3.	Будет ли проект крупно- или среднемасштабным?	Да
4.	Ожидается ли длительная эксплуатация продукта?	Да
5.	Необходим ли высокий уровень надежности продукта проекта?	Да
6.	Предполагается ли эволюция продукта проекта в течение ЖЦ?	Да
7.	Велика ли вероятность изменения системы (продукта) на этапе сопровождения?	Да
8.	Является ли график сжатым?	Да
9.	Предполагается ли повторное использование компонентов?	Да
10.	Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	Нет

ПРИЛОЖЕНИЕ В.1

Листинг обучения модели

```
def treeLearn(param, trees):
    if param == True:
        mylist = []
        for chunk in pd.read_csv('files/output12.csv', chunksize=100, dtype='uint32',
low_memory=False):
            mylist.append(chunk)
        df = pd.concat(mylist, axis=0)
        q = len(df.columns) - 1
        df = df.astype(int)
        X = df.values[:, 1:q-1]
        Y = df.values[:, q-1]
        lastColum = df.values[:, q]
        kf = KFold(n_splits=5,shuffle=True)
        for train_index, test_index in kf.split(X):
            print("TRAIN:", train_index, "TEST:", test_index)
            X_train, X_test = X[train_index], X[test_index]
            Y_train, Y_test = Y[train_index], Y[test_index]
        trees.fit(X_train, Y_train)
        tree_pred = trees.predict(X_test)
        print(f1_score(Y_test, tree_pred, average='weighted'), ">>>>>>>>>>>>>>>> точность ")
        print(f1_score(Y_test, tree_pred, average=None), ">>>>>>>>>>>>>>>> точность weighted")
        tree_predUser = trees.predict(X)
        tree_predUser = np.column_stack((lastColum,tree_predUser))
        dfU = pd.DataFrame(tree_predUser)
        try:
            dfU.to_csv('files/outputUser.csv')
        except:
            print("Сломалось")
        with open('files/treesdump.pkl', 'wb') as output_file:
            pickle.dump(trees, output_file)
        #рисование дерева
        try:
            dot_data = tree.export_graphviz(trees, out_file=None, rounded=True, filled=True)
            graph = pydotplus.graph_from_dot_data(dot_data)
            graph = graphviz.Source(dot_data)
            graph.render("static/trees")
        except:
            print("рисование дерево не получилось")
    else:
        with open('files/treesdump.pkl', 'rb') as output_file:
            trees1 = pickle.load(output_file)
        df = pd.read_csv('files/output2.csv', dtype='uint32', low_memory=False)
        q = len(df.columns) - 1
        df = df.astype(int)
        X = df.values[:, 1:q-1]
        Y = df.values[:, q-1]
        trees1.predict(X)
```

ПРИЛОЖЕНИЕ В.2

Листинг кода построения таблицы для обучения

```
def tableLearn(categ):
    try:
        conn = psycopg2.connect("dbname='vk' user='postgres' host='127.0.0.1' password='1'")
    except:
        print("Не удалось подключиться к БД")
    # Создаем курсор для работы
    cur = conn.cursor(cursor_factory=psycopg2.extras.DictCursor)
    try:
        if categ == 0:
            cur.execute("SELECT uservkid, f.category FROM vkuser as v INNER JOIN faculty as f ON
f.facultyid = v.faculty WHERE v.category = "+str(categ)+" and f.category != 0 order by uservkid")
        else:
            cur.execute("SELECT uservkid, f.category FROM vkuser as v INNER JOIN faculty as f ON
f.facultyid = v.faculty WHERE v.category = " + str(categ) + " and f.category != 1 order by
uservkid")
        users = cur.fetchall()
        if categ == 0:
            cur.execute("SELECT idclear FROM vkuser_clearmusicbandnew GROUP BY idclear
HAVING count(*)> 10 and count(*)<1000 ORDER BY count(*) DESC")
            musicband = cur.fetchall()
            with open('files/musicbanddump.pkl', 'wb') as output_file:
                pickle.dump(musicband, output_file)
        else:
            with open('files/musicbanddump.pkl', 'rb') as output_file:
                musicband = pickle.load(output_file)
    except:
        print('Сломалася')
    svaz = [[5587], [5588]]
    dfaUser = np.zeros((len(users)), dtype=np.uint32)
    for indexU, j in enumerate(users):
        dfa = np.zeros((len(musicband) + 2), dtype=np.uint32)
        try:
            cur.execute("SELECT idclear FROM vkuser_clearmusicbandnew WHERE idvkuser =
"+str(j[0])+" order by idclear")
            svaz = cur.fetchall()
            print(svaz)
        except:
            print("запрос списка связи упал")
        try:
            cur.execute("SELECT f.category FROM vkuser as v INNER JOIN faculty as f ON
f.facultyid = v.faculty WHERE uservkid = "+str(j[0])+"")
            category = cur.fetchall()
        except:
            print("запрос списка связи упал")
        summaDrop = 0
        for indexS, k in enumerate(svaz):
            try:
```

```
            if musicband.index(k) != 0:
                print(musicband.index(k)
                dfa[musicband.index(k)] = 1
                summaDrop+=1
                print(summaDrop)
        except ValueError as e:
            print("ERROR > ", e)
    if summaDrop == 0:
        continue
    dfa[len(musicband)] = category[0][0]
    dfa[len(musicband)+1] = j[0]
    dfaUser[indexU] = j[0]
    dfaau = pd.DataFrame(dfaUser)
    dfa = np.expand_dims(dfa, axis=0)
    dfaOut = pd.DataFrame(dfa)
    @contextmanager
    def open_file(path, mode):
        file_to = open(path, mode)
        yield file_to
        file_to.close()
    try:
        if categ == 0:
            with open_file('files/output12.csv', 'r') as infile:
                dfaOut.to_csv('files/output12.csv', mode='a', header=False)
            dfaau.to_csv('files/output6User.csv')
        else:
            dfaau.to_csv('files/output1User.csv')
            dfaOut.to_csv('files/output1.csv', mode='a')
            print("ЗАПИСАНО 1")
    except:
        print("Упала запись в файл")
```

ПРИЛОЖЕНИЕ В.3

Листинг парсера пользовательских данных

```
def setUserInfo(userid):
    try:
        session = vk.Session(
access_token='fe34dc9f0660284aa0caced7b2dbe57579a378b45f24a99ba490b124f02c90958f7b7f7
df802fb7f80650')
        vk_api = vk.API(session, v='5.71')
        try:
            profiles = vk_api.users.get(user_id=userid,
fields='city,connections,education,interests,sex,universities')
        except VkAPIError as e:
            print("================VK================", e)
        for i in profiles:
            try:
                fname = i['first_name']
            except:
                fname = "null"
            try:
                lname = i['last_name']
            except:
                lname = "null"
            try:
                city = i['city']['id']
            except:
                city = "6666666"
            try:
                university = i['university']
                if university == 0:
                    universityArr = i['universities']
                    if universityArr != []:
                        for j in universityArr:
                            university = j['id']
                    else:
                        university = "6666666"
            except:
                university = "6666666"
            try:
                faculty = i['faculty']
                if faculty == 0:
                    facultyArr = i['universities']
                    if facultyArr != []:
                        for j in facultyArr:
                            faculty = j['faculty']
                    else:
                        faculty = "6666666"
            except:
                faculty = "6666666"
            try:
```

```
            faculty_name = i['faculty_name']
        except:
            faculty_name = "6666666"
    setCity(str(city))
    setFaculty(str(university), str(faculty), str(faculty_name))
    cur.execute("INSERT INTO public.vkuser (uservkid, fname, lname, city, faculty) VALUES ('"
+ str(
        userid) + "', '" + str(fname) + "', '" + str(lname) + "', '" + str(city) + "', '" + str(faculty) + "')")
        conn.commit()

    profiles = ""
except (psycopg2.DatabaseError, psycopg2.Error) as e:
    if conn:
        conn.rollback()
    print(e.pgcode, " КОД ОШИБКИ")
    print('Error %s' % e)
    if e.pgcode == "23505":
        try:
            cur.execute("UPDATE public.vkuser SET fname = '" + str(fname) + "', lname = '" + str(
                lname) + "', city = '" + str(city) + "', faculty = '" + str(
                faculty) + "' WHERE ""uservkid"" = " + str(userid) + "")
            conn.commit()
        except psycopg2.DatabaseError as e:
            if conn:
                conn.rollback()
            print("Запрос UPDATE пользователя не удался setInfoUser")
xpaths = {
    'usernameTxtBox': "//*[@id='index_email']",
    'passwordTxtBox': "//*[@id='index_pass']",
    'submitButton': "//*[@id='index_login_button']"
mydriver = webdriver.Firefox(executable_path=r'D:\parser\geckodriver.exe')
mydriver.get(baseurl)
mydriver.find_element_by_xpath(xpaths['usernameTxtBox']).clear()
mydriver.find_element_by_xpath(xpaths['usernameTxtBox']).send_keys(username)
mydriver.find_element_by_xpath(xpaths['passwordTxtBox']).clear()
mydriver.find_element_by_xpath(xpaths['passwordTxtBox']).send_keys(password)
mydriver.find_element_by_xpath(xpaths['submitButton']).click()
time.sleep(5)
# Разбираем список пользователей
for audio in audios:
    urlaudio = baseurl + 'audios' + audio
    try:
        mydriver.get(urlaudio)
        if urlaudio == mydriver.current_url:
            for i in range(100):
                mydriver.execute_script("window.scrollBy(0,1000)")
            setUser(audio)
            setUserInfo(audio)
            content = mydriver.find_elements_by_css_selector('a.audio_row__performer')
            viewMusic(content)
            time.sleep(0.5)
        else:
```

```
            print("Доступ к аудио закрыт id", audio)
            continue
    except:
        print("Не удалось обратиться к url")
        continue
    print("REFRESH страницы")
    mydriver.refresh()
try:
    cur.execute("SELECT musicbandid, nameband FROM musicband")
    results = cur.fetchall()
    for j in results:
        clearnameband = cleaningMusicBand(j[1])
        try:
            cur.execute("UPDATE musicband SET clearnameband = ltrim('" + str(clearnameband) + "')
WHERE musicbandid ='"+str(j[0])+"'")
            conn.commit()
        except:
            print("error")
```

ПРИЛОЖЕНИЕ В.4

Листинг очистки музыкальных исполнителей

```
def cleaningMusicBand(nameband):
    try:
        result = re.sub(r'[^a-zA-Za-яА-Я0-9( )ёЁ]', ' ', nameband)
        result = re.sub(r'OST', '', result)
        result = re.sub(r'[()_]', ' ', result)
        result = re.sub(r'\s+', ' ', result)
        resFind = result.find('FEAT')
        if resFind != -1:
            result = result[0:resFind:1]
        resFind = result.find('GLOBAL TUNING')
        if resFind != -1:
            result = result[resFind + 13:len(result):1]
        resFind = result.find('FT')
        if resFind != -1:
            if result.find("DAFT") != -1:
            elif result.find("DRIFTMOON") != -1:
            elif result.find("FIFTH HARMONY") != -1:
            else:
                result = result[0:resFind:1]
        resFind = result.find('HZ ')
        if resFind != -1:
            result = result[resFind + 3:len(result):1]
            resFind = result.find('HZ ')
            if resFind != -1:
                result = result[resFind + 3:len(result):1]
        resFind = result.find('CARMUSICKZ')
        if resFind != -1:
            result = result[resFind + 10:len(result):1]
        resFind = result.find('CLUB РАЙ')
        if resFind != -1:
            result = result[resFind + 8:len(result):1]
        resFind = result.find('145MUSIC')
        if resFind != -1:
            result = result[resFind + 8:len(result):1]
        result = re.sub(r'VKCOMSADINSTR', ' ', result)
        result = re.sub(r'VKHPNET', ' ', result)
        result = re.sub(r'МУЗЫКА', ' ', result)
        result = re.sub(r'НОВИНКА', ' ', result)
        result = re.sub(r'НОВИНКИ', ' ', result)
        result = re.sub(r'РИНГТОН', ' ', result)
        result = re.sub(r'ХИТ', ' ', result)
        result = re.sub(r'ХИТЫ', ' ', result)
        result = re.sub(r'EUROPA PLUS', ' ', result)
        result = re.sub(r'МЕСТО', ' ', result)
        result = re.sub(r'RADIO RECORD', ' ', result)
        result = re.sub(r'BOMBAFM', ' ', result)
        result = re.sub(r'CLUB', ' ', result)
```

```
        result = re.sub(r'DEEP HOUSE', ' ', result)
        result = re.sub(r'DFM', ' ', result)
        result = re.sub(r'EUROVISION', ' ', result)
        result = re.sub(r'FDM', ' ', result)
        result = re.sub(r'BPANMUSIC', ' ', result)
        result = re.sub(r'MUZMORU', ' ', result)
        result = re.sub(r'MS', ' ', result)
        result = re.sub(r'MUSIC', ' ', result)
        result = re.sub(r'[\d]', '', result)
        result = re.sub(r'\s+', ' ', result)
        if len(result) > 25:
            result = result[0:25:1]
        if len(result) < 2:
            result = ''
        else:
            return result
    except:
        print("очитска трека упала")
        return nameband
```

ПРИЛОЖЕНИЕ В.5

Листинг распределения факультетов на категории

```
def categoryFaculty(faculty):
    try:
        for j in faculty:
            j[1] = j[1].upper()
            if j[1].find('ЖУРНА') > -1 or j[1].find('УПРАВЕНИЕ') > -1 or j[1].find('СОЦ') > -1 or
j[1].find(
                    'ФИЛОЛ') > -1 or j[1].find('ЯЗЫК') > -1 or j[1].find('СПОРТ') > -1 or
j[1].find('ТУРИЗ') > -1 or \
                    j[1].find('ДИЗАЙ') > -1 or j[1].find('МУНИЦ') > -1 or j[1].find('ГУМАН') > -1 or
j[1].find(
                'ВОКАЛ') > -1 or j[1].find('ПСИХО') > -1 or j[1].find('ПЕДАГ') > -1 or j[1].find(
                'ФИЗИЧЕСКОЙ') > -1 or j[1].find('РУССК') > -1 or j[1].find('ЛИТЕРАТ') > -1 or
j[1].find('МЕНЕДЖ') > -1 or j[1].find('ФИНАН') > -1 or j[1].find('ЭКОНОМ') > -1 or
j[1].find('ПРАВО') > -1 or j[1].find('ЮРИД') > -1 or j[1].find('ПРАВА') > -1 or j[1].find(
                    'АРБИТ') > -1 or j[1].find('АДВОКА') > -1 or j[1].find('УГОЛОВ') > -1 or j[1].find(
                'ФИЛОС') > -1 or j[1].find('ИСТОР') > -1 or j[1].find('ЮРИС') > -1:
                print(j[1], "<========Гуманитарный")
                cur.execute("UPDATE faculty SET category = '1' WHERE facultyid ='" + str(j[0]) + "'")

            elif j[1].find('БИОЛ') > -1 or j[1].find('ЕСТЕСТ') > -1 or j[1].find('ГЕОЛ') > -1 or j[1].find(
                    'НЕФТ') > -1 or j[1].find('ГАЗА') > -1 or j[1].find('БУРЕ') > -1 or j[1].find('АВТОМ')
> -1 or \
                    j[1].find('ХИМИ') > -1 or j[1].find('ЭКОЛО') > -1 or j[1].find('ПРИРОДО') > -1 or
j[1].find(
                'ЭНЕРГЕ') > -1 or j[1].find('НЕФТЕ') > -1 or j[1].find('МАТЕМ') > -1 or
j[1].find('ТЕХН') > -1 or j[1].find('ЭЛЕКТ') > -1 or j[1].find(
                    'СИСТЕМ') > -1 or j[1].find('ВЫЧИСЛИ') > -1 or j[1].find('ИНЖЕН') > -1 or
j[1].find(
                'ИНФОРМ') > -1 or j[1].find('МОДЕЛИ') > -1 or j[1].find('СТРОИТ') > -1 or j[1].find(
                'ФИЗИК') > -1 or j[1].find('ТЕХНИЧ') > -1 or j[1].find('ТЕХНОЛОГ') > -1 or
j[1].find('ТРАНСПОРТ') > -1:
                print(j[1], "<========Техники")
                cur.execute("UPDATE faculty SET category = '2' WHERE facultyid ='" + str(j[0]) + "'")
            else:
                print('Неизвестное сочетание')
                cur.execute("UPDATE faculty SET category = '0' WHERE facultyid ='" + str(j[0]) + "'")
        conn.commit()
    except:
        print("факультет упал")
```

ПРИЛОЖЕНИЕ В.6

Листинг реализации взаимодействий пользователя с Системой

```
@app.route("/")
@app.route("/index")
def hello():
    return render_template("home.html", massage = 'Добро пожаловать!')
@app.route("/showTree")
def showTree():
    return render_template("search.html", tree='files/iris.pdf')
@app.route("/about")
def about():
    return render_template("about.html", massage = 'Добро пожаловать!')
UPLOAD_FOLDER = r'D:\uploadfile'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
#допустимые расширения файлов
ALLOWED_EXTENSIONS = set(['json'])
#функция проверки файлов
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS
@app.route("/search")
def search():
    try:
        conn = psycopg2.connect("dbname='vk' user='postgres' host='127.0.0.1' password='1'")
        print("connect OK")
    except:
        print("Не удалось подключиться к БД")
    cur = conn.cursor(cursor_factory=psycopg2.extras.DictCursor)
    df = pd.read_csv('files/outputUser.csv')
    df.drop(df.columns[[0]], axis=1, inplace=True)
    print(df.shape[0])
    print(df)
    a = []
    for i in range(df.shape[0]):
        a.append(df.values[i])
    userL = []
    userF = []
    # Создаем курсор для работы
    for i in a:
        print(i[0],'<<<<<')
        try:
            cur.execute("SELECT fname FROM vkuser WHERE uservkid = "+str(i[0])+"")
            usersF = cur.fetchone()
            userF.append(usersF[0])
            cur.execute("SELECT lname FROM vkuser WHERE uservkid = " + str(i[0]) + "")
            usersL = cur.fetchone()
            userL.append(usersL[0])
        except:
            print("ошибка")
```

```
    print(userF)
    print(userL)
    df.loc[:, 'Fname'] = pd.Series(userF)
    df.loc[:, 'Lname'] = pd.Series(userL)
    df.columns = ['ID пользователя',"Класс","Имя","Фамилия"]
    df.head()
    return render_template("search.html", massage='Ранее собранные данные',
id=HTML(df.to_html(max_rows=None,classes='myTable', justify='center')))
def pup(audio):
    subprocess.Popen(vksearch.program(audio))
@app.route("/uploadfile/", methods=['POST'])
def upload_file():
    try:
        if request.method == 'POST':
            f = request.files['filename']
            print(allowed_file(f.filename), '<<<<')
            print(f and allowed_file(f.filename), '>>>>>')
            if f and allowed_file(f.filename):
                filename = secure_filename(f.filename)
                print('работает')
                print(filename)
                f.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            else:
                return render_template("home.html", massage = 'Неверное расширение файла')
            try:
                with open(os.path.join(app.config['UPLOAD_FOLDER'], filename), 'r', encoding='utf-8')
as f:
                    str = f.read()
                    audio = json.loads(str)
                    f.close()
                with ThreadPoolExecutor(max_workers=2) as executor:
                    executor.submit(vksearch.program(audio), ())
                # p = Process(target=vksearch.program(audio))
                # p.start()
                # p.join()
                return render_template("search.html", massage='Расчеты произведены!')
            except Exception as e:
                print(e, ' <<<<<error')
                print('Загрузка файла не удалась')
                f.close()
                return render_template("home.html", massage='Загрузка файла провалилась')
    except:
        f = None
        print('лажа')
        return render_template("home.html", massage = 'Загрузка файла провалилась')
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html', massage = "Ошибка 404, страница не найдена"), 404
@app.errorhandler(500)
def internal_error(e):
    return render_template('500.html', massage = "Непредвиденная ошибка на сервере"), 500
```

ПРИЛОЖЕНИЕ В.7

Листинг алгоритма Левенштейна

```
def cleaningMusicClear(nameband):
    try:
        cur.execute("SELECT musicbandid, clearnameband, levenshtein('" + str(
            nameband) + "', clearnameband) as sml FROM musicband WHERE levenshtein('" + str(
            nameband) + "', clearnameband) = 1 ORDER BY  sml, clearnameband")
        results = cur.fetchall()
        print(results)
        if results == []:
            print('ПУСТО')
        else:
            for j in results:
                print(j[0], j[1], j[2])
                arrClean.append(j[0])
                cur.execute(
                    "UPDATE musicband SET clear = '" + str(j[1]) + "' WHERE musicbandid ='" +
str(j[0]) + "'")
                conn.commit()
    except:
        print("очитска трека упала")
```

ПРИЛОЖЕНИЕ Г.1

Таблица – Анкета для фокус-группы.

Вопрос	**Ответ**
1. Оцените читаемость текстов в программе по 10-балльной шкале. (*1-очень низкая, 10-очень высокая*)	
2. Оцените шрифт, используемый в программном продукте. (*Да/Нет*)	
3. Оцените цветовую гамму программного продукта по 10 –балльной шкале. (*1-очень низкая, 10-очень высокая*)	
4. Насколько интуитивно понятна навигация в программе? • *Совсем непонятна* • *Немного непонятна* • *Достаточно понятна* • *Хорошо понятна*	
5. Что бы вы поменяли в программном продукте? (*письменно*)	
6. Что нужно сделать, чтобы начать поиск абитуриентов? (*письменно*)	
7. Опишите не удобное расположение кнопок на формах (*письменно*)	
8. Что вам не нравится в нашем продукте? (*письменно*)	

ПРИЛОЖЕНИЕ Г.2

Таблица – Результаты оценки анкет фокус-группы.

№	Вопрос	Результат	Решение на основе результатов.
1	Оцените читаемость текстов в программе по 10-балльной шкале. (*1-очень низкая, 10-очень высокая*)	4,8	Требуются изменения.
2	Оцените шрифт, используемый в программном продукте. (*Да/Нет*)	Да	Изменений не требуется.
3	Оцените цветовую гамму программного продукта по 10 –балльной шкале. (*1-очень низкая, 10-очень высокая*)	8,2	Изменений не требуется.
4	Насколько интуитивно понятна навигация в программе? • *Совсем непонятна* • *Немного непонятна* • *Достаточно понятна* • *Хорошо понятна*	Хорошо понятна	Изменений не требуется.
5	Что бы вы поменяли в программном продукте? (*письменно*)	Изменить толщину шрифта, шрифт не серьезный.	
6	Что нужно сделать, чтобы начать поиск абитуриентов? (*письменно*)	1) Выбрать файл 2) Нажать кнопку запуска 3) Ждать завершения процесса.	
7	Опишите не удобное расположение кнопок на формах (*письменно*)	Кнопки расположены интуитивно, но слишком сжато друг к другу.	
8	Что вам не нравится в нашем продукте? (*письменно*)	1) Уменьшить пустое пространство 2) Толщину шрифта 3) Шрифт слишком сложный для восприятия 4) Пустое расстояние на главной странице 5) Расстояние между кнопками	