МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ
Заведующий кафедрой
д.п.н., профессор
_____И.Г. Захарова
_____2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

НА ОСНОВЕ АНАЛИЗА ЗАВИСИМОСТЕЙ ПОКАЗАТЕЛЕЙ ОДНОТИПНЫХ ОБЪЕКТОВ

02.04.03. Математическое обеспечение и администрирование информационных систем
Магистерская программа «Высокопроизводительные вычислительные системы»

Выполнила работу
Студентка 2 курса
очной формы
обучения
Богданюк
Юлия
Викторовна

Научный
руководитель
к.т.н., доцент
Воробьева
Марина
Сергеевна

Рецензент
к.т.н., доцент

Григорьев Михаил Викторович

Тюмень 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время все более неотъемлемой частью продвижения в интернете является web-аналитика. Необходимость детального анализа показателей эффективности продвижения обусловлена увеличением количества активных пользователей интернета и конкурентов, продвигающих свои продукты и услуги в интернете. Одним из методов web-аналитики является анализ рекламных кампаний конкурентов.

При разработке стратегии продвижения, в частности контекстной рекламы, используется множество инструментов. Одним из инструментов является аналитика стратегий продвижения конкурентов, то есть рекламных кампаний с однотипными параметрами.

Аналитика стратегий включает в себя выявление зависимостей и периодичности показателей рекламных кампаний с пересекающимися параметрами. Такая аналитика позволяет разработать стратегию продвижения, которая будет наиболее эффективна при вводе внешних ограничений, таких как бюджет рекламной кампании и количество посетителей сайта, и будет учитывать стратегии продвижения конкурентов.

Данная тема является актуальной, так как ежегодно наблюдается рост оборота глобального рынка интернет-рекламы. Например, «в 2017 году по оценке исследовательской компании Magna, объем рынка интернет-рекламы составил 41% мирового рекламного рынка, при оценке объема телевизионной рекламы в 35%. Ожидается, что к 2020 году интернет-реклама составит 50% мирового рынка рекламы» [1].

Целью исследования является проектирование и разработка системы поддержки принятия решений для анализа зависимостей и периодичности показателей объектов с пересекающимися параметрами.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить методы анализа данных, методы анализа зависимостей параметров объектов;
- проанализировать источники данных и подготовить dataset для системы;
- применить алгоритмы обучения и выполнить оценку качества моделей обучения;
- разработать систему поддержки принятия решений, предлагающую смоделированные варианты состояний аукциона;
- выполнить апробацию предлагаемых решений системы на реальных данных.

Научно-практическая новизна исследовательской работы заключается в применении результатов анализа зависимостей и периодичности показателей рекламных кампаний в процессе моделирования возможных решений

для разработки наиболее эффективной стратегии размещения рекламных объявлений в поисковой выдаче.

По итогам исследования планируется внедрение разработанной системы в работу организации, специализирующейся на разработке стратегий интернет-рекламы.

ГЛАВА 1. ОПРЕДЕЛЕНИЕ ЦЕЛЕЙ И МЕТОДОЛОГИИ ИССЛЕДОВАНИЯ

§ 1.1. Принципы работы аукциона в поисковой системе

Объявления контекстной рекламы отображаются в результатах выдачи поисковой системы при каждом запросе пользователя поисковой системы. После ввода запроса проводится аукцион в режиме реального времени. По результатам аукциона объявления всех рекламодателей размещаются на позициях рекламного блока.

Пример отображения рекламного блока поисковой выдачи и позиций конкурентов по результатам аукциона представлен на рисунке 1.1.1.

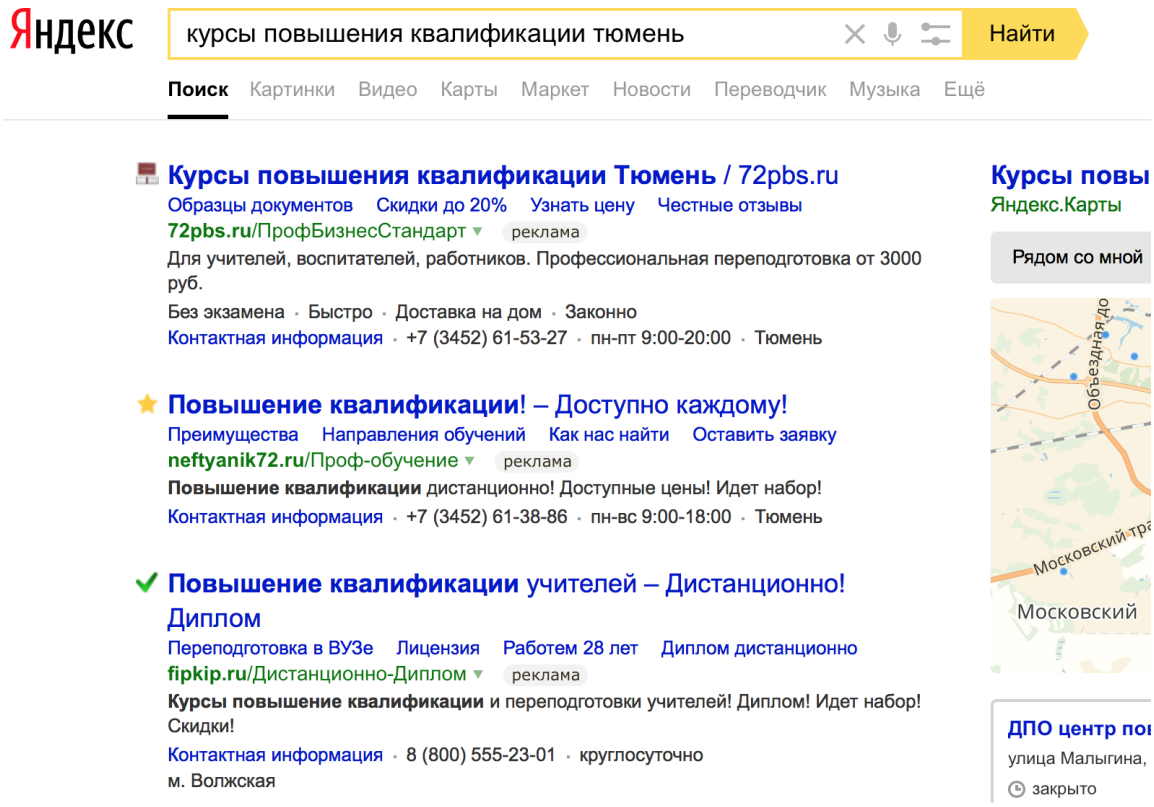


Рисунок 1.1.1. Пример рекламного блока поисковой выдачи по запросу «курсы повышения квалификации Тюмень»

Позиция, на которой будет отображаться объявление, зависит от 3 параметров: ставка, которую рекламодатель готов заплатить за размещение в аукционе, кликабельность (отношение кликов к показам) объявлений и показатель эффективности, который рассчитывается поисковой системой [12].

При разработке рекламной кампании маркетологи настраивают параметры показа объявлений, и в первую очередь настраивается ставка. Однако с ростом числа рекламодателей и конкуренции все чаще используются инструменты оптимизации бюджета кампании. Например, настраивается временной таргетинг показа объявлений: день недели, числа месяца, часы суток. Также возможна настройка показа объявлений исключительно на определенной позиции, то есть, при невозможности размещаться, например, на первой строчке рекламного блока, объявление исключается из показа.

Так как аукцион проводится каждый раз, когда пользователи выполняют запрос, невозможно без автоматизированных систем отследить результаты поисковой выдачи и возможные стратегии размещения конкурентов (всех рекламодателей, объявления которых отображаются при вводе одного и того же запроса).

§ 1.2. Обзор существующих инструментов анализа стратегий

Анализ стратегии размещения конкурентов в поисковой выдаче является одним из первых этапов разработки рекламной кампании. На данный момент существует 2 подхода к решению задачи анализа стратегий - мониторинг

поисковой выдачи и использование специальных сервисов для анализа стратегий конкурентов.

При ручном мониторинге поисковой выдачи специалисты анализируют список размещающихся в выдаче конкурентов и их позиции. Мониторинг производится с определенной периодичностью и по всем поисковым запросам, которые включены в рекламную кампанию. С помощью специальных сервисов возможно получение всех поисковых запросов, по которым отображаются объявления определенного конкурента. К таким сервисам относятся SimilarWeb, СайтРепорт.

Сервис SimilarWeb позволяет получать сводную информацию об источниках трафика, ключевых словах, ссылках, присутствии в социальных сетях. Для того чтобы получить сводную информацию, необходимо ввести адрес сайта конкурента.

Сервис СайтРепорт позволяет для указанного сайта получить информацию о рекламной активности – количестве поисковых запросов, по которым отображаются объявления владельца сайта.

Специальные сервисы для анализа стратегий размещения объявлений в поисковой выдаче позволяют получить сводную информацию по домену, такую как средняя позиция, процент времени размещения на каждой позиции, количество показов на каждой позиции. К таким сервисам относятся «Инклюзив Медиа» и SpyWords.

Сервис «Инклюзив Медиа» позволяет выгрузить отчет по видимости конкурентов с поисковой выдаче, получить подробную информацию о позициях рекламных объявлений сайтов конкурентов.

Сервис SpyWords позволяет выгрузить информацию о количестве и показателе CTR ключевых слов сайтов конкурентов.

Стоит отметить, что для мониторинга сводной информации по конкурентам приведенные сервисы в качестве источников данных используют собственную накопленную базу данных, а не открытые источники.

§ 1.3. Применение методологии CRISP-DM в работе

Для достижения цели разработки системы поддержки принятия решений необходимо провести анализ данных, включающий предварительную обработку исходных данных и анализ источников данных. Опишем процесс проведения научного исследования по методологии CRISP-DM (Cross Industry Standart Process for Data Mining) [26]. Согласно данной методологии разделим исследование на этапы (см. Рисунок 1.3.1):

1. Бизнес-анализ;
2. Анализ данных;
3. Подготовка данных;
4. Моделирование;
5. Оценка результата;
6. Внедрение.

Рисунок 1.3.1. Схема этапов методологии CRISP-DM

Рассмотрим задачи, которые включены в каждый пункт детально и определим, какая работа в рамках каждого из этапов будет проведена.

На этапе бизнес-планирования определяются бизнес-цели проекта, выполняется оценка текущей ситуации, определяются цели аналитики и выполняется подготовка плана проекта [23]. В рамках проведенного исследования в этап бизнес-планирования входят следующие задачи: постановка задачи исследования, формулировка решаемой проблемы, анализ возможных источников данных, выполняется выбор метрик для оценки точности результата моделирования, определяются критерии успешности модели, формулируются требования к разрабатываемой системе и разрабатывается архитектура системы.

«Этап анализа данных включает реализацию сбора данных, описание, изучение и проверку качества данных». В ходе проведенного исследования на этапе анализа данных

реализуются следующие задачи: автоматизация сбора данных, описание данных, первичное исследование данных и выявление зависимостей между параметрами, обеспечение высокого качества данных за счет автоматизации сбора данных.

«Этап подготовки данных включает выборку, очистку, генерацию, интеграцию и форматирование данных». В рамках проведенного исследования в этап подготовки данных входят следующие задачи: отбор атрибутов для каждого из источников данных, генерация новых атрибутов, интеграция данных из двух источников данных в один дата-сет, проектирование хранилища данных.

«Этап моделирования включает выбор алгоритмов обучения, подготовку плана тестирования, обучение моделей и оценку качества моделей». В ходе проведенного исследования на этапе моделирования реализуются следующие задачи: выбор алгоритмов обучения модели, анализ наилучшего разделения выборки данных на обучающую и тестовую выборки, обучение моделей, оценка качества моделей с помощью выбранных метрик.

«Этап оценки решения включает оценку результатов, оценку процесса и определение следующих шагов». В рамках проведенного исследования на этапе оценки решения проводится оценка полученных результатов модели, выявление новых знаний, которые получает пользователь системы.

«Этап внедрения включает внедрение решения, планирование мониторинга и поддержки, подготовку отчета и превью проекта». Результатом проведенного исследования является обученная модель прогнозирования состояния

аукциона. На этапе внедрения в ходе проведенного исследования определяется, каким образом будет внедрен модуль анализа данных в систему и интегрирован с web-приложением, реализован модуль визуализации данных.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ «COMPETITOR»

§ 2.1. Постановка задачи

Анализ стратегий продвижения рекламных кампаний с однотипными параметрами включает в себя отслеживание текущих показателей и позиций размещения всех объектов кампаний, выявление зависимостей между показателями и позициями как объектов одной рекламной кампании, так и между объектами разных кампаний.

Знание стратегий и зависимостей показателей однотипных кампаний позволяет для новой рекламной кампании разработать такую стратегию, по которой владелец сайта будет получать наибольшее количество переходов посетителей на сайт с учетом ограничений на бюджет кампании.

Мониторинг показателей однотипных кампаний пользователем является невыполнимой задачей, так как поисковые системы обновляют показатели с частотой до 1 минуты. Ручная обработка, анализ зависимостей постоянно обновляющихся и неструктурированных данных требует автоматизации процесса аналитики данных. Поэтому целью исследования является проектирование системы поддержки принятия решений на основе анализа зависимостей и периодичности однотипных объектов.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить методы анализа данных, методы анализа зависимостей параметров объектов;
- определить требования к разрабатываемой системе;

- автоматизировать сбор данных;
- разработать алгоритм поддержки средств регулирования нагрузки на сервер;
- спроектировать и разработать хранилище данных системы;
- выполнить анализ и подготовку данных для обучения;
- применить алгоритмы обучения и выполнить оценку качества моделей обучения;
- разработать систему поддержки принятия решений, предлагающую смоделированные варианты состояний аукциона;
- с целью апробации проверить применимость предлагаемых решений системы на реальных данных.

Разрабатываемое приложение должно удовлетворять следующим требованиям:

- приложение должно быть написано с использованием web-технологий для обеспечения возможности доступа к системе с любого устройства;
- приложение должно поддерживать средства регулирования нагрузки на сервер рекламной системы;
- необходимо автоматизировать сбор данных из рекламной системы и поисковой системы: сбор цен позиций из рекламной системы и позиций конкурентов из поисковой системы;
- в качестве входных данных система должна принимать данные рекламных кампаний посредством подключения к пользовательскому кабинету рекламной системы.

Разрабатываемое приложение должно предоставлять следующий функционал:

- визуализация поисковой выдачи рекламных объявлений с указанием цен позиций;
- выявление стратегий размещения объявлений рекламных кампаний конкурентов;
- генерация предложений возможных стратегий размещения объявлений конкурентов.

Работа системы поддержки принятия решений должна быть разделена на следующие этапы:

1) формирование из набора данных о ценах позиций и результатах выдачи поисковой системы таблиц данных по каждому поисковому запросу;
2) предобработка данных для обеспечения высокой точности результатов прогнозирования;
3) проведение по каждому конкуренту анализа данных о позициях объектов кампаний и ценах на позиции поисковой выдачи на наличие зависимости, поиск функции для прогнозирования позиции;
4) обучение модели прогнозирования и статистический анализ полученных результатов;
5) моделирование состояния поисковой выдачи.

Сбор данных из рекламной системы и поисковой выдачи должен осуществляться автоматически с указанным интервалом времени.

Для того чтобы построить прогноз позиции, на которой будет размещаться объект кампании, проанализируем зависимость признаков объекта друг от друга.

Пусть дано множество объектов $A=\left\{a_1, a_2, \ldots, a_n \middle| n \in N\right\}$, где $a_i=\left\{a_i^1, a_i^2, \ldots, a_i^m\right\}, где i=1..n, m \in N$. Для каждого объекта a_i дано множество наблюдаемых состояний $S_i=\left\{s_i^1, s_i^2, \ldots, s_i^c\right\}, где c$ –

порядковый номер последнего замера состояния объектов, то есть $s_i^j = a_{ij} = \left(a_{ij}^1, a_{ij}^2, \ldots, a_{ij}^m\right), j = 1..c$.

Схема объектов и их состояний представлена на рисунке 2.1.1.

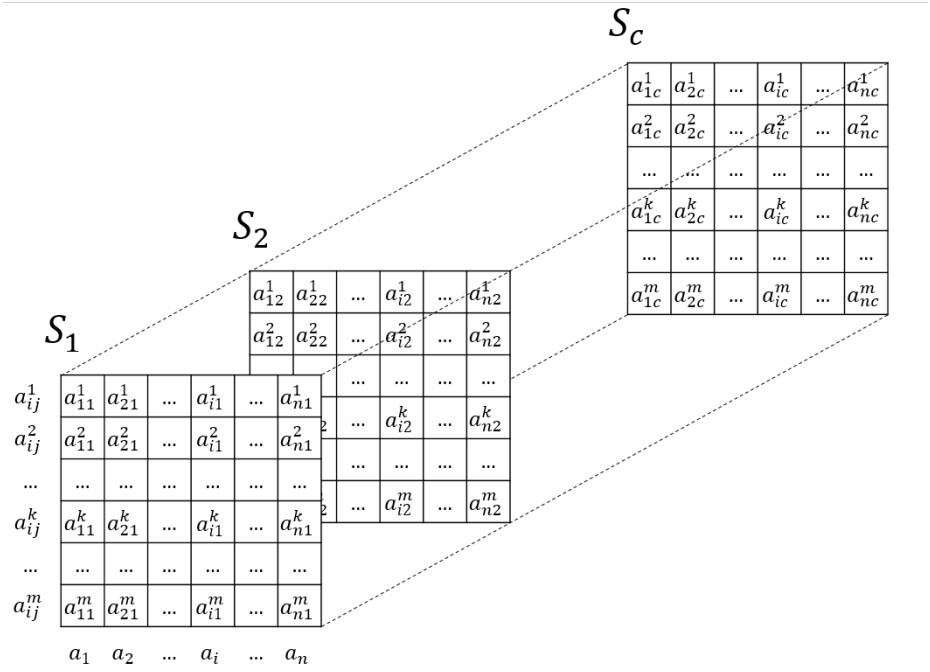


Рисунок 2.1.1. Схема объектов и их состояний

Необходимо для каждого a_i, имеющего множество состояний S_i и множество соответствующих состояниям меток позиции $a_{ij}^m \in M$, найти отображение $y : S_i \to a_{ij}^m$ такое, что для неизвестного состояния s_i^{c+1} можно найти метку $label \in M$.

§ 2.2. Алгоритмы и подходы, используемые в системе «Competitor»

2.2.1. Средства регулирования нагрузки на сервер

Сервер API Яндекс.Директ имеет несколько средств регулирования нагрузки. Есть два обязательных средства для поддержки в приложении.

Во-первых, сервер API Яндекс.Директ накладывает ограничения на количество объектов, которое возвращает сервер в ответ на запрос, и ограничение на количество объектов, которое может быть отправлено серверу в одном запросе.

Во-вторых, в качестве основного средства регулирования нагрузки на сервер применяются баллы. Нехватка баллов не позволяет выполнять запросы к серверу API.

Баллы начисляются на счет аккаунта Яндекса и списываются при каждом запросе к серверу от имени этого аккаунта. В ответ на каждый запрос к API в HTTP-заголовке запроса возвращается параметр, в котором указано количество баллов:

- израсходовано баллов при выполнении запроса;
- доступный остаток баллов;
- суточный лимит баллов аккаунта.

Каждому аккаунту предоставляется индивидуальный суточный лимит баллов. Причем лимит зависит от активности и эффективности рекламных кампаний аккаунта – количества кликов и показов и израсходованного бюджета кампаний и всего аккаунта.

Частые и объемные запросы к API считаются нерациональной нагрузкой на серверы API в тех случаях, когда количество кликов и показов кампаний аккаунта растет незначительно или когда на счете кампаний или аккаунта осталось немного средств. Яндекс использует сетку бюджетных порогов для расчета индивидуальных суточных лимитов. Эта сетка была разработана с учетом статистики кампаний Яндекса разных типов и тематик.

Индивидуальный суточный лимит баллов разделен на 24 часовых интервала и предоставляется по принципу скользящего окна (см. Рисунок 2.2.1). На счет аккаунта в начале каждого интервала начисляется 1/24 суточного лимита. В текущем интервале от имени аккаунта можно использовать 1/24 суточного лимита плюс баллы, начисленные, но не использованные за последние прошедшие 23 часа.

Рисунок 2.2.1. Описание средства регулирования нагрузки на сервер

При этом, время начала временного интервала может отличаться для разных аккаунтов и не совпадать с началом астрономического часа.

Сервером API Яндекс.Директ списываются баллы в зависимости от произведенного действия:

1. успешный вызов метода;
2. вызов метода, завершившийся ошибкой;
3. получение объекта;
4. успешную операцию создания или редактирования объекта;

5. ошибку выполнения операции создания или редактирования объекта.

Пункты 4 и 5 не рассматриваются в данной работе, так как не входят в список задач исследования и разработки.

Важность поддержки баллов как средства регулирования нагрузки на сервер обусловлена тем, что за ошибки при выполнении запросов к серверу списывается в разы большее количество баллов, чем за успешную отправку запросов.

2.2.2. Автоматизированный сбор данных

Сбор из рекламной системы происходит посредством взаимодействия с сервером системы через API с помощью POST-запросов в формате JSON (см. Рисунок 2.2.2) и включает следующие этапы:

1. Формирование запроса, включающего группу идентификаторов ключевых слов, и позволяющего получить значения цен позиций по каждому из включенных ключевых слов;
2. Отправка необходимого количества запросов для получения цен всех ключевых слов кампании в соответствии с алгоритмом поддержки регулирования нагрузки на сервер и получение ответа в формате JSON;
3. Десериализация полученных данных в структуру для хранения исторических данных, включающую поля значений времени замера состояния, идентификатора ключевого слова, цен всех позиций аукциона, и запись в локальное хранилище данных.

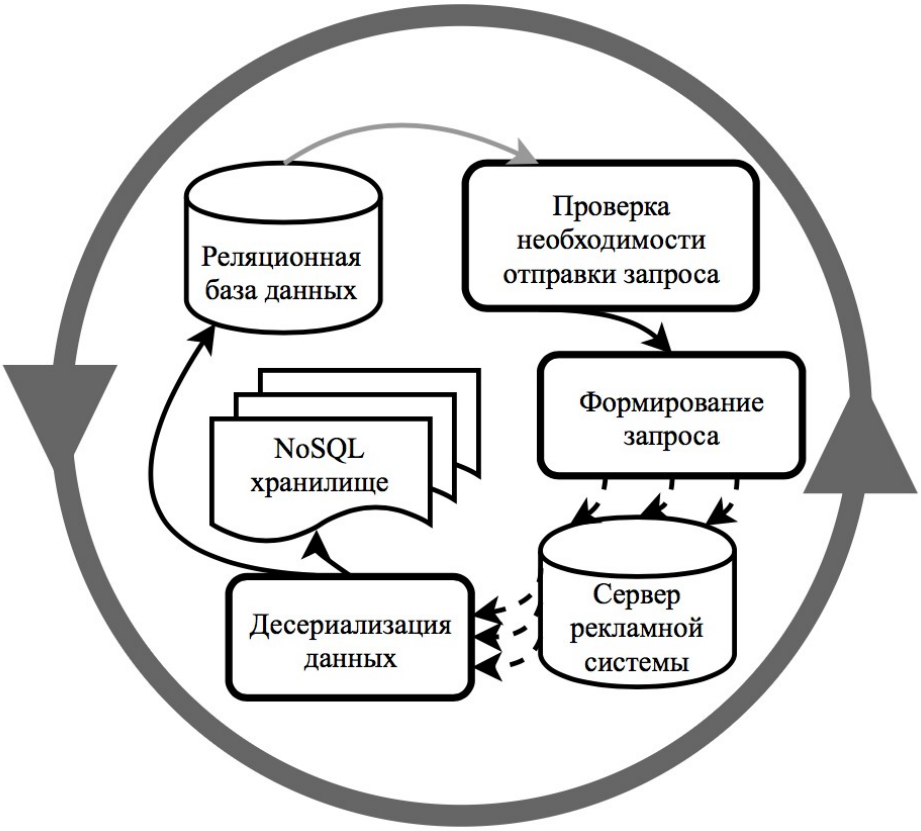


Рисунок 2.2.2. Схема процесса выгрузки данных с сервера рекламной системы

Сбор данных о позициях размещения объявлений производится с помощью синтаксического анализа HTML-кода страницы поисковой выдачи (см. Рис. 2.2.3). Процесс сбора можно разделить на следующие этапы:

1. Формирование строки URL, включающей ключевое слово, для отправки get-запроса на адрес поисковой системы;
2. Отправка запроса и получение ответа в виде HTML-кода страницы поисковой выдачи;
3. Синтаксический анализ HTML-кода страницы поисковой выдачи для извлечения данных объектов рекламного блока;
4. Запись параметров извлеченных объектов в локальное хранилище данных.

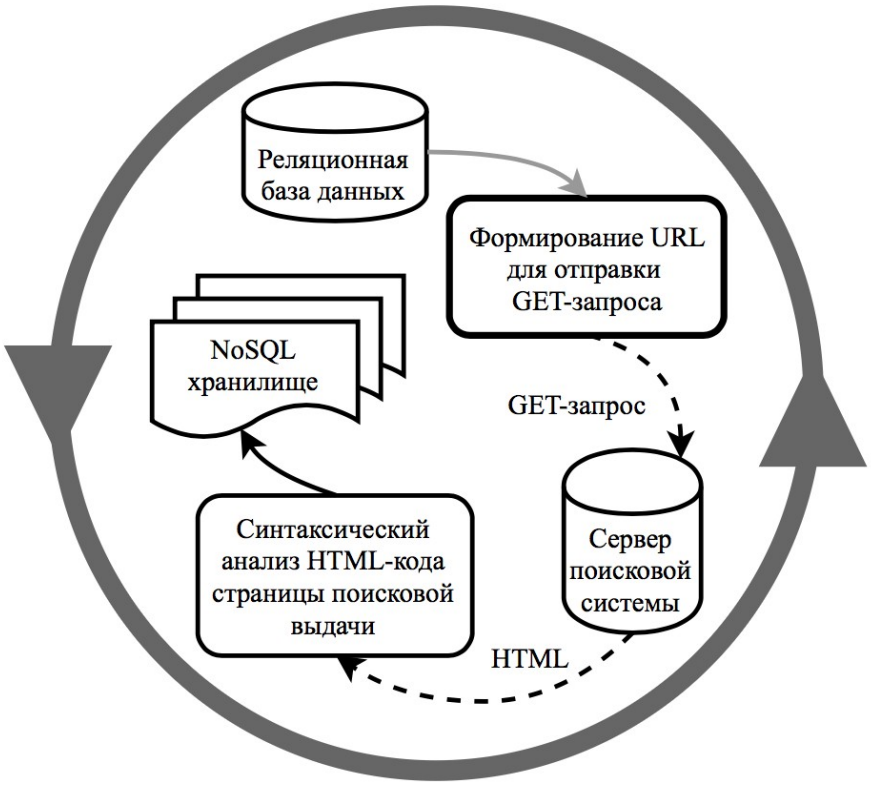


Рисунок 2.2.3. Схема процесса сбора данных о позициях размещения объявлений

2.2.3. Алгоритм поддержки средств регулирования нагрузки на сервер

Необходимо разработать метод, который будет определять временные интервалы обновления параметров аккаунтов, так как за ошибочные запросы к серверу API списывается увеличенное количество баллов по сравнению с успешными запросами. При этом, временные интервалы должны рассчитываться таким образом, чтобы обмен данными между приложением и сервером был равномерно распределен внутри интервалов средства регулирования нагрузки на сервер.

Разработанный метод состоит из двух частей. Во-первых, данные о баллах обновляются и пересчитываются при каждом запросе приложения к серверу API. Во-вторых, каждую минуту происходит пересчет текущих показателей и

проверка необходимости отправки запроса на сервер для обновления данных аккаунта.

Для реализации метода необходимо хранить следующие данные в локальном хранилище приложения:

- SM_i - количество баллов, списываемое за вызов i-го метода API, и SO_i - количество баллов, списываемое за объект при вызове i-го метода API;
- SD - суточный лимит баллов аккаунта;
- SC - текущий лимит баллов аккаунта;
- SMin - количество баллов, необходимое для выгрузки всех объектов аккаунта;
- UB - дата и время последней выгрузки всех объектов аккаунта;
- UA - дата и время последнего пополнения баллов аккаунта.

Алгоритм обновления данных о баллах, который выполняется при каждом запросе от приложения к серверу API (см. Рис. 2.2.4):

Шаг 1) Считывание из HTTP-заголовка запроса значений SD и SC - показателей баллов аккаунта;

Шаг 2) Обращение к локальному хранилищу приложения и получение SD и SC - текущих показателей баллов аккаунта;

Шаг 3) Если новое значение SC больше текущего значения SC, которое записано в локальном хранилище приложения, то обновление UA в хранилище на текущее значение даты и времени; Обновление в локальном хранилище значения SC - текущего доступного лимита баллов аккаунта;

Шаг 4) Обращение к локальному хранилищу приложения и вычисление OC – количества объектов, параметры которых необходимо обновить, и QC – количества запросов, которые

необходимо отправить на сервер API для выгрузки всех дочерних объектов и параметров аккаунта, при условии включения в каждый запрос максимально возможного количества объектов;

Шаг 5) Обращение к локальному хранилищу и получение значений SM и SO;

Шаг 6) Вычисление количества баллов, необходимого для выгрузки всех объектов аккаунта, по формуле (2.1):

$$SMin = \sum_{i=1}^{n} OC * SO_i + QC * SM_i, \quad (2.1)$$

где i=1..n - номер типа запроса.

Шаг 7) Обновление в локальном хранилище приложения значений SM_i и SO_i.

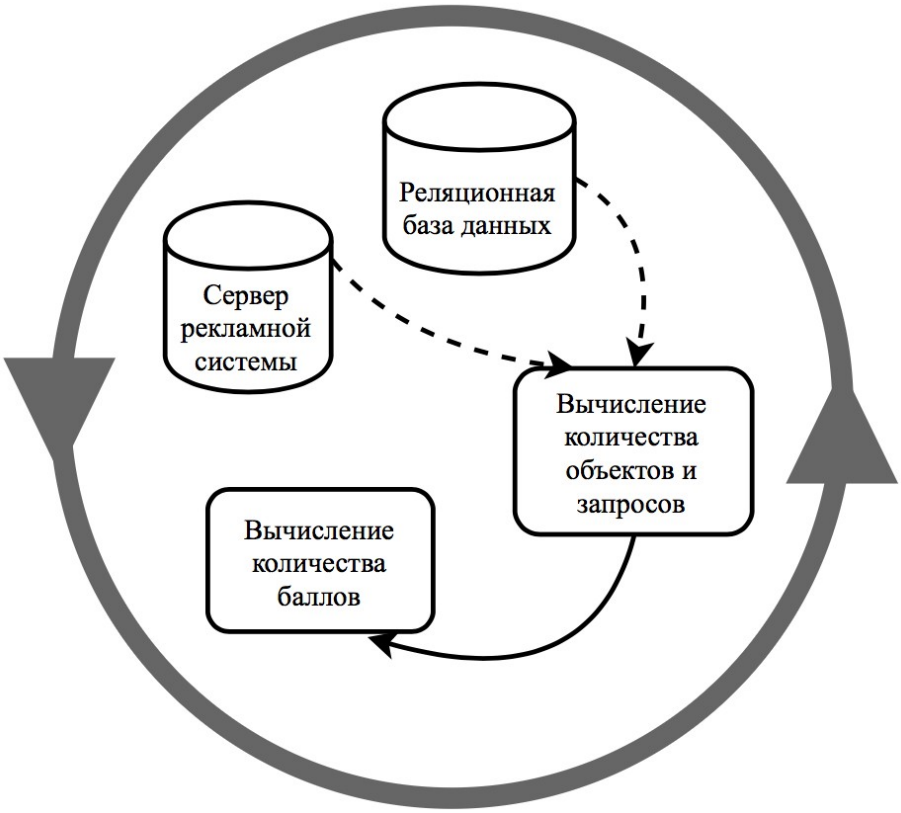


Рисунок 2.2.4. Схема алгоритма обновления данных о баллах

Алгоритм пересчета текущих показателей и проверки необходимости отправки запроса на сервер API для обновления данных аккаунта выполняется каждую минуту (см. Рис. 2.2.5):

Шаг 1) Обращение к локальному хранилищу приложения и получение текущих значений SMin, SC, UA и UB;

Шаг 2) Вычисление количества выгрузок данных, которое может быть выполнено до следующего пополнения лимита баллов аккаунта, по формуле:

$$P = SC / SMin \tag{2.2}$$

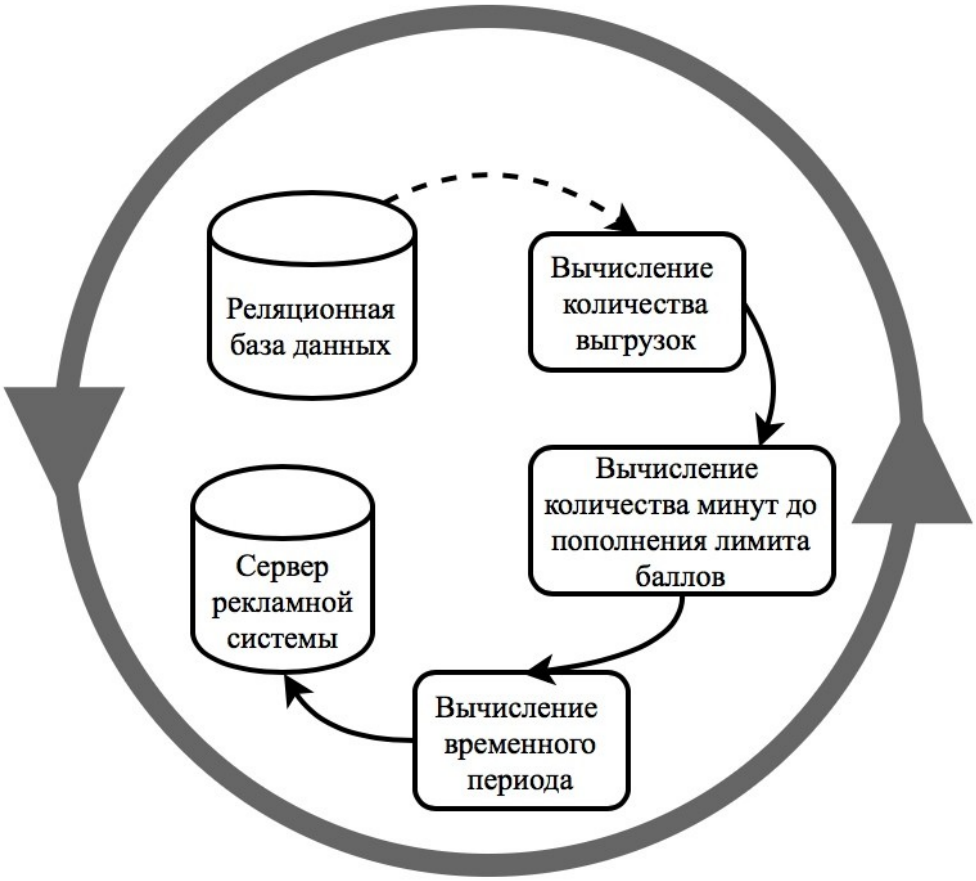


Рисунок 2.2.5. Алгоритм пересчета текущих показателей и проверки необходимости отправки запроса

Шаг 3) Вычисление количества минут, через которое будет пополнен лимит баллов аккаунта, по формуле:

$$M = 60 - |CT - UA|, \quad (2.3)$$

где CT – значение текущего времени суток;

Шаг 4) Вычисление временного периода, через который возможна равномерная выгрузка параметров всех объектов аккаунта в текущем интервале суточного лимита баллов аккаунта, по формуле:

$$PM = M / P \quad (2.4)$$

Шаг 5) Если $UB \geq PM$, то приложение формирует и отправляет запрос на сервер для выгрузки параметров объектов аккаунта и значение UB обновляется в локальном хранилище приложения.

ГЛАВА 3. ПРОГНОЗИРОВАНИЕ СОСТОЯНИЯ АУКЦИОНА

§ 3.1. Подготовка данных

Этап подготовки данных включается в себя отбор данных (признаков и объектов), очистку данных, генерацию, интеграцию и форматирование.

Система поддержки принятия решений содержит модуль автоматизированного сбора данных, что позволяет не выполнять отбор и очистку данных. Автоматизированный сбор данных исключает ситуации, когда у объектов могут быть пропущенные значения атрибутов, ошибки в данных или несоответствующая кодировка.

На этапе генерации данных необходимо проанализировать исходные атрибуты объектов и спроектировать дополнительные атрибуты для проверки гипотез о повышении точности модели обучения при добавлении дополнительных признаков.

Сбор данных происходит из более, чем одного источника, поэтому необходимо выполнять интеграцию данных для подготовки обучающей выборки.

Данные о ценах позиций объекта a_i, получены от сервера поисковой системы и представлены упорядоченным множеством $S_i^1 = \{s_i^{1,1}, s_i^{1,2}, \dots, s_i^{1,c}\}$ состояний, определенных в пункте 2.1, с соответствующим каждому элементу значением времени замера состояния, где c – порядковый номер последнего замера состояния объекта a_i.

Данные о позициях размещения рекламных объявлений объекта a_i, полученные из поисковой выдачи представлены упорядоченным множеством $S_i^2 = \{s_i^{2,1}, s_i^{2,2}, \dots, s_i^{2,c}\}$ состоянийс

соответствующим каждому элементу значениемвремени замера состояния.

При этом $\left|S_i^1\right|=¿S_i^2 \vee ¿$. Объединим множества S_i^1 и S_i^2 состояний в множество S_i так, что $s_i^j=s_i^{1,j} \cup s_i^{2,j}$. Примем за метку класса выборки значение a_i^m позиции размещения рекламного объявления, а множество $a_i\{a¿_i^m$ за множество признаков объекта.

Выборка содержит значения времени замера состояния, поэтому необходимо провести форматирование данных таким образом, что элементы множества были упорядочены по возрастанию по значению времени замера состояния.

Данные о ценах позиций, получаемые посредством POST-запроса от сервера рекламной системы, десериализуются в объект типа динамический список структуры AuctionBids.

Данные о позициях размещения конкурентов, получаемые в формате HTML в ответе на GET-запрос, извлекаются посредством синтаксического анализа в объект типа динамический список структуры SearchAd.

Так как при настройке рекламных кампаний используются такие параметры таргетинга, как время, то выдвинем гипотезу о наличии зависимости между позициями объявлений поисковой выдачи и временем и проверим ее. Для этого выполним генерацию дополнительных признаков состояний аукциона: месяц, день недели, суточный временной интервал (по 3 часа), час, минута. Данные признаки будем генерировать из значения поля TimeStamp объекта структуры AuctionBids.

Обучение модели прогнозирования состояния аукциона по каждому поисковому запросу будет проводиться по

матрице данных, которая формируется путем интеграции данных о ценах позиций, данных о позициях размещения конкурентов и данных о времени замера состояния. Под интеграцией данных понимается «горизонтальное» соединение признаков. Схема процесса интеграции данных представлена на Рисунке 3.1.1.

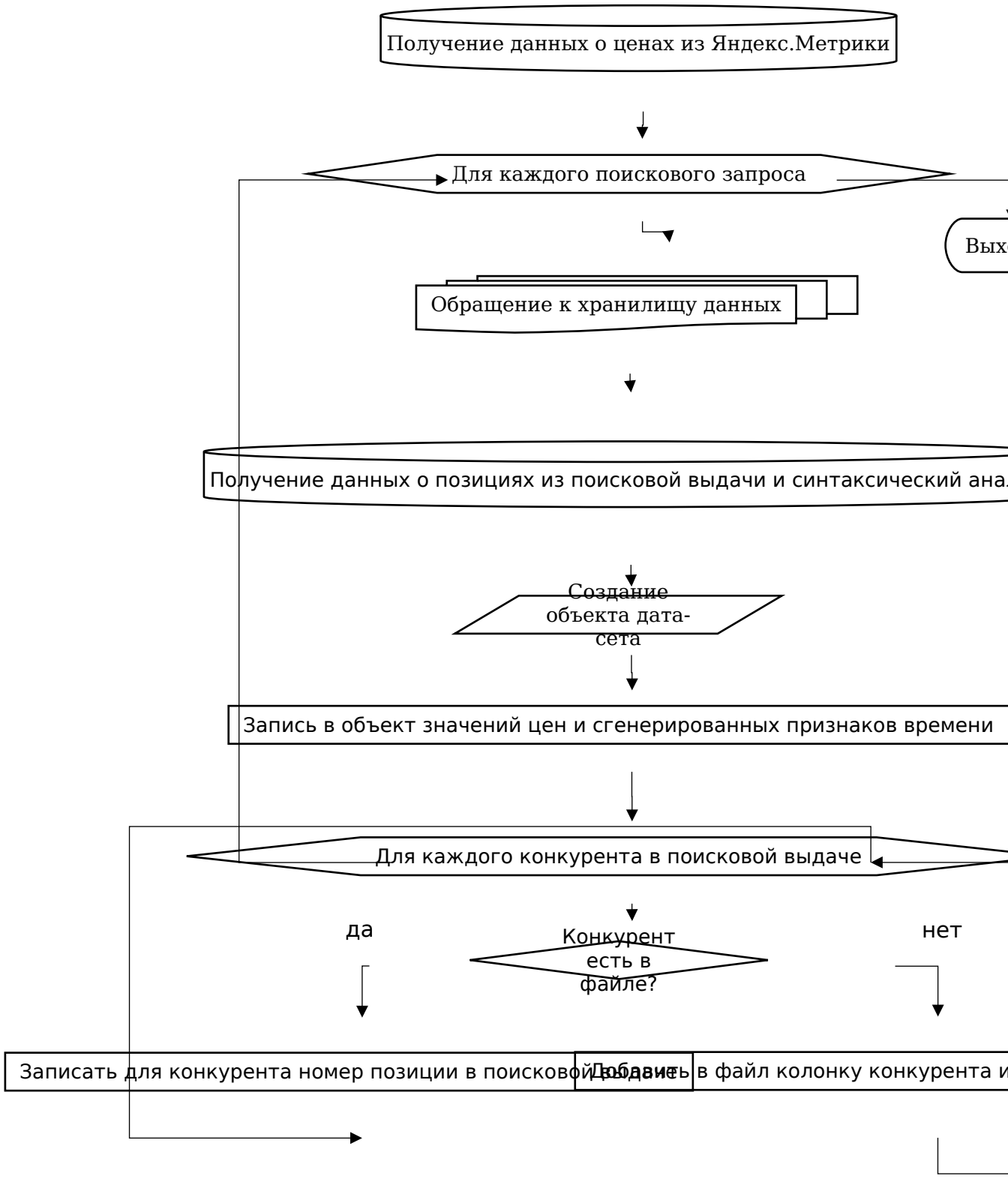


Рисунок 3.1.1. Схема процесса интеграции данных

Таким образом, для каждого поискового запроса рекламной кампании данные (признаки), выгружаемые из двух источников, интегрируются в один объект дата-сета – объект структуры CompetitorDatasetItem (см. Таблица 3.1.1).

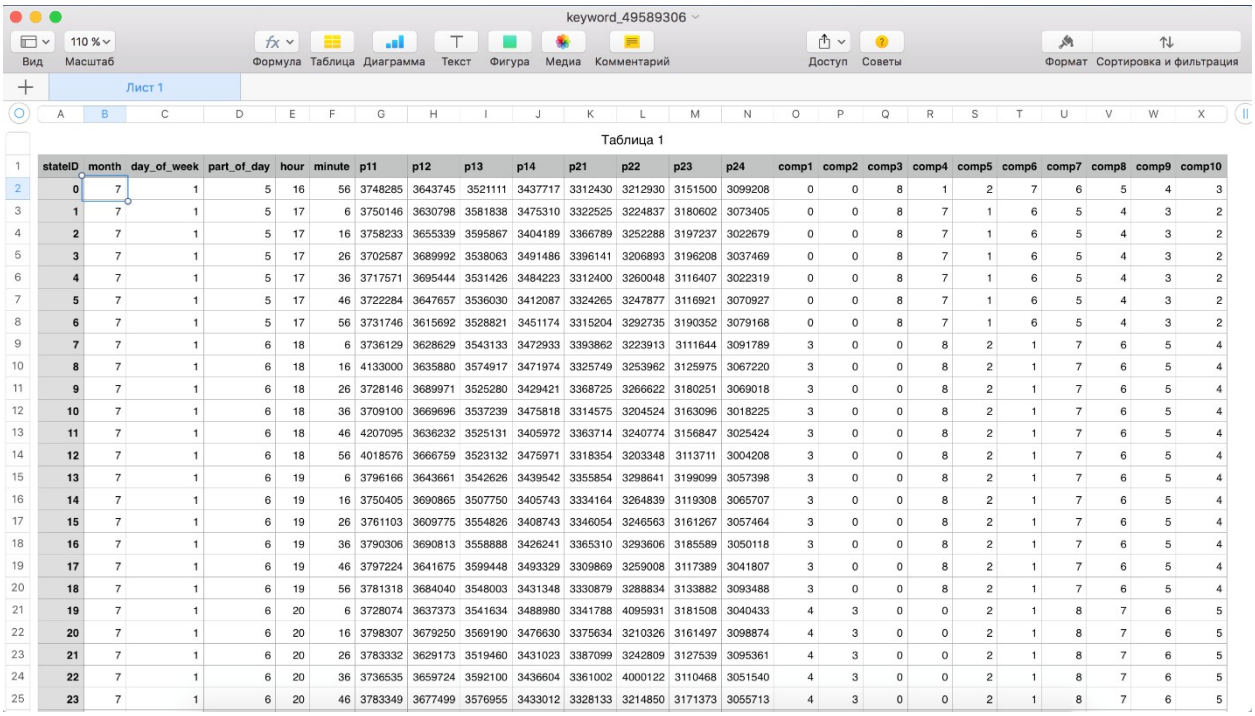
Таблица 3.1.1.Признаки объекта дата-сета

№	Название поля	Тип	Описание	Примечание
1	stateID	int	Идентификатор состояния	
2	p_i	long	Цены размещения на i-ой позиции	Более одного поля
3	$comp_j$	int	Позиция j-го конкурента	Более одного конкурента
4	month	int	Номер месяца	
5	day_of_week	int	Номер дня недели, где понедельник = 1	
6	part_of_day	int	Номер части дня, где $part_i=[hour/3]$	
7	hour	int	Значение часа, в которые был сделан замер состояния	
8	minute	int	Значение минуты, в которую был сделан замер состояния	

После интеграции признаков объект структуры CompetitorDatasetItem преобразовывается в строковую переменную, соответствующую формату CSV-файла для обрабатываемого поискового запроса, и записывается в конец этого файла.

Для обучения модели прогнозирования состояния аукциона по определенному поисковому запросу в качестве выборки данных (дата-сета) используются данные, хранящиеся в соответствующем этому запросу CSV-файле.

Пример файла для поискового запроса № 49589305 представлен на Рисунке 3.1.2. Каждая строка представляет запись состояния аукциона, каждый столбец – признак состояния. Файл содержит более 12 000 строк и 24 столбца, данные взяты за 3 месяца с интервалом 10 минут.



keyword_49589306

Вид | Масштаб | Формула | Таблица | Диаграмма | Текст | Фигура | Медиа | Комментарий | Доступ | Советы | Формат | Сортировка и фильтрация

Лист 1

Таблица 1

	stateID	month	day_of_week	part_of_day	hour	minute	p11	p12	p13	p14	p21	p22	p23	p24	comp1	comp2	comp3	comp4	comp5	comp6	comp7	comp8	comp9	comp10
2	0	7	1	5	16	56	3748285	3643745	3521111	3437717	3312430	3212930	3151500	3099208	0	0	8	1	2	7	6	5	4	3
3	1	7	1	5	17	6	3750146	3630798	3581838	3475310	3322525	3224837	3180602	3073405	0	0	8	7	1	6	5	4	3	2
4	2	7	1	5	17	16	3758233	3655339	3595867	3404189	3366789	3252288	3197237	3022679	0	0	8	7	1	6	5	4	3	2
5	3	7	1	5	17	26	3702587	3689992	3538063	3491486	3396141	3206893	3196208	3037469	0	0	8	7	1	6	5	4	3	2
6	4	7	1	5	17	36	3717571	3695444	3531426	3484223	3312400	3260048	3116407	3022319	0	0	8	7	1	6	5	4	3	2
7	5	7	1	5	17	46	3722284	3647657	3536030	3412087	3324265	3247877	3116921	3070927	0	0	8	7	1	6	5	4	3	2
8	6	7	1	5	17	56	3731746	3615692	3528821	3451174	3315204	3292735	3190352	3079168	0	0	8	7	1	6	5	4	3	2
9	7	7	1	6	18	6	3736129	3628629	3543133	3472933	3393862	3223913	3111644	3091789	3	0	0	8	2	1	7	6	5	4
10	8	7	1	6	18	16	4133000	3635880	3574917	3471974	3325749	3253962	3125975	3067220	3	0	0	8	2	1	7	6	5	4
11	9	7	1	6	18	26	3728146	3689971	3525280	3429421	3368725	3266622	3180251	3069018	3	0	0	8	2	1	7	6	5	4
12	10	7	1	6	18	36	3709100	3669696	3537239	3475818	3314575	3204524	3163096	3018225	3	0	0	8	2	1	7	6	5	4
13	11	7	1	6	18	46	4207095	3636232	3525131	3405972	3363714	3240774	3156847	3025424	3	0	0	8	2	1	7	6	5	4
14	12	7	1	6	18	56	4018576	3666759	3523132	3475971	3318354	3203348	3113711	3004208	3	0	0	8	2	1	7	6	5	4
15	13	7	1	6	19	6	3796166	3643661	3542626	3439542	3355854	3298641	3199099	3057398	3	0	0	8	2	1	7	6	5	4
16	14	7	1	6	19	16	3750405	3690865	3507750	3405743	3334164	3264839	3119308	3065707	3	0	0	8	2	1	7	6	5	4
17	15	7	1	6	19	26	3761103	3609775	3554826	3408743	3346054	3246563	3161267	3057464	3	0	0	8	2	1	7	6	5	4
18	16	7	1	6	19	36	3790306	3690813	3558888	3426241	3365310	3293606	3185589	3050118	3	0	0	8	2	1	7	6	5	4
19	17	7	1	6	19	46	3797224	3641675	3599448	3493329	3309869	3259008	3117389	3041807	3	0	0	8	2	1	7	6	5	4
20	18	7	1	6	19	56	3781318	3684040	3548003	3431348	3330879	3288834	3133882	3093488	3	0	0	8	2	1	7	6	5	4
21	19	7	1	6	20	6	3728074	3637373	3541634	3488980	3341788	4095931	3181508	3040433	4	3	0	0	2	1	8	7	6	5
22	20	7	1	6	20	16	3798307	3679250	3569190	3476630	3375634	3210326	3161497	3098874	4	3	0	0	2	1	8	7	6	5
23	21	7	1	6	20	26	3783332	3629173	3519460	3431023	3387099	3242809	3127539	3095361	4	3	0	0	2	1	8	7	6	5
24	22	7	1	6	20	36	3736535	3659724	3592100	3436604	3361002	4000122	3110468	3051540	4	3	0	0	2	1	8	7	6	5
25	23	7	1	6	20	46	3783349	3677499	3576955	3433012	3328133	3214850	3171373	3055713	4	3	0	0	2	1	8	7	6	5

Рисунок 3.1.2. Пример CSV-файла поискового запроса

Автоматизированный сбор данных для обучения модели гарантирует, что однозначно исключить из выборки можно только первый признак – идентификатор состояния аукциона (значения этого признака являются уникальными для каждого объекта выборки).

§ 3.2. Выбор алгоритмов и оценка качества моделей обучения

Необходимо спрогнозировать позиции каждого из конкурентов. В качестве исходных данных для прогнозирования позиции возьмем признаки $p_1, p_2, \ldots, p_z$ (z – количество позиций рекламного блока поисковой

выдачи),month, day_of_week, part_of_day, hour, minute, определенные в пункте 3.1.

Поставленную задачу можно рассматривать как задачу многоклассовой классификации. Для решения задачи применим методы машинного обучения, рекомендованные документацией библиотеки scikit-learn в разделе «Подбор подходящей оценочной функции» [25]. Воспользуемся методами: Стохастического градиентного спуска (SGD), Метод опорных векторов (SVM) с линейным, радиальным базисным, полиномиальным ядрами и k-ближайших соседей. Также применим такие классические методы классификации, как Дерево решений и Байесовский классификатор.

В статье [13] Алексея Натекина, члена сообщества «Open Data Science», говорится о том, что в задачах ранжирования выдачи поисковых систем используется метод Градиентного бустинга, поэтому применим этот метод для прогнозирования позиции конкурента в поисковой выдаче.

Для оценки качества алгоритмов классификации воспользуемся такой метрикой, как матрица ошибок для C классов (см. Таблица 3.2.1). Пусть дана выборка x_i ($i=1,\dots,N$, y_i – метка класса -го объекта, $y_i \in \{1,2,\dots,C\}$), каждый объект которой относится к одному из C классов и классификатор a, который эти классы предсказывает. Матрицей ошибок для такого классификатора называется следующая матрица:

$$M=\{m_{ij}\}_{i,j=0}^{C}, m_{ij}=\sum_{k=0}^{N}\left[a(x_k)=j\right]\left[y_k=i\right] \quad (3.1)$$

Значение m_{ij} означает, сколько объектов класса j были распознаны как объекты класса i.

Таблица 3.2.1. Матрица ошибок классификатора

	y_1	y_2	…	y_i	…	y_C

y_1	m_{11}	m_{12}	…	m_{ij}	…	m_{ij}
y_2	m_{21}	m_{22}	…	m_{2i}	…	m_{2C}
…	…	…	…	…	…	…
y_i	m_{i1}	m_{i2}	…	m_{ii}	…	m_{iC}
…	…	…	…	…	…	…
y_C	m_{C1}	m_{C2}	…	m_{Ci}	…	m_{CC}

Матрица ошибок позволит понять, какие классы определяются алгоритмом (классификатором) неверно относительно друг друга в большинстве случаев. «В многоклассовых задачах, сводят подсчет качества к вычислению одной из двухклассовых метрик. Выделяют два подхода к такому сведению: микро- и макро-усреднение» [11].

Введем следующие обозначения:

- TP_i – число корректно предсказанных элементов класса i;
- TN_i – число элементов, не относящихся к классу i, предсказанных как не относящиеся к классу i;
- FN_i – число элементов, которые были некорректно предсказаны как элементы класса i;
- FP_i – число элементов класса i, которые были некорректно предсказаны как элементы других классов.

«При микро-усреднении сначала эти характеристики усредняются по всем классам, а затем вычисляется итоговая метрика». Рассчитаем для каждой модели микро-меры для учета наиболее крупных классов. Рассчитаем точность - долю истинно положительных примеров среди примеров, предсказанных как положительные, по формуле:

$$Precision_{\mu}=\frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{K}(TP_i+FP_i)} \quad (3.2)$$

Рассчитаем долю истинно положительных примеров среди фактически положительных примеров, то есть полноту, по формуле:

$$Recall_{\mu}=\frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{K}(TP_i+FN_i)} \quad (3.3)$$

Рассчитаем F-меру, которая представляет гармоническое среднее между точностью и полнотой, по формуле:

$$F_1-score_{\mu}=\frac{2*Precision_{\mu}*Recall_{\mu}}{Precision_{\mu}+Recall_{\mu}} \quad (3.4)$$

«При макро-усреднении сначала вычисляется итоговая метрика для каждого класса, а затем результаты усредняются по всем классам». Рассчитаем для учета всех классов с одинаковым весом без учета их размеров для каждой модели макро-меры – точность по формуле (3.5), полноту по формуле (3.6) и F-меру по формуле (3.7):

$$Precision_M=\frac{\sum_{i=1}^{K}\frac{TP_i}{TP_i+FP_i}}{K} \quad (3.5)$$

$$Recall_M=\frac{\sum_{i=1}^{K}\frac{TP_i}{TP_i+FN_i}}{K} \quad (3.6)$$

$$F_1-score_M=\frac{2*Precision_M*Recall_M}{Precision_M+Recall_M} \quad (3.7)$$

Если классы отличаются по мощности, то при микро-усреднении они практически никак не будут влиять на результат, поскольку их вклад в средние TP, FP, FN и TN

будет незначителен. В случае же с макро-вариантом усреднение проводится для величин, которые уже не чувствительны к соотношению размеров классов, и поэтому каждый класс внесет равный вклад в итоговую метрику.

ГЛАВА 4. ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА

§ 4.1. Используемые технологии и архитектура системы

Модули сбора и визуализации системы «Competitor» были разработаны в среде Microsoft Visual Studio 2017 Comunity на языке программирования C# с использованием технологии ASP.NET Core и библиотеки JavaScript, FluentScheduler. В качестве СУБД использовалась СУБД PostgreSQL 9.6.

ASP.NET Core – кроссплатформенная, высокопроизводительная среда от Microsoft с открытым исходным кодом для разработки web-приложений и web-сервисов.

JavaScript – объектно-ориентированный сценарный язык программирования. Наиболее широко применяется в браузерах как язык сценариев для придания интерактивности webстраницам.

FluentScheduler – библиотека для выполнения программного кода приложения по расписанию.

PostgreSQL 9.6 – реляционная система управления базами данных с открытым исходным кодом.

Взаимодействие с сервером API Яндекс.Директ осуществлялось с использованием 5-й версии API (интерфейса прикладного программирования) сервера Яндекс.Директ.

Модуль синтаксического анализа поисковой выдачи реализован с помощью библиотеки AngleSharp 0.9.

Модуль анализа данных системы «Competitor» был разработан в среде Spyder дистрибутива Anaconda на языке программирования Python 3.6.

Схема архитектуры системы поддержки принятия решений представлена на Рисунке 4.1.1.

Система поддержки принятия решений включает в себя 2 основных программных блока – web-приложение и модуль анализа данных. Web-приложение включает следующие модули:

- модуль расчета нагрузки и отправки запросов на сервер рекламной системы;
- модуль синтаксического анализа HTML-страницы поисковой выдачи;
- модуль интеграции данных;
- модуль интерпретации и визуализации данных.

Web-приложение взаимодействует с реляционной базой данных, сервером рекламной системы посредством API и поисковой системой по HTTP-протоколу. Модуль интеграции данных осуществляет запись данных в NoSQL хранилище данных, которое представлено каталогом CSV-файлов.

Модуль анализа данных взаимодействует с NoSQL хранилищем данных.

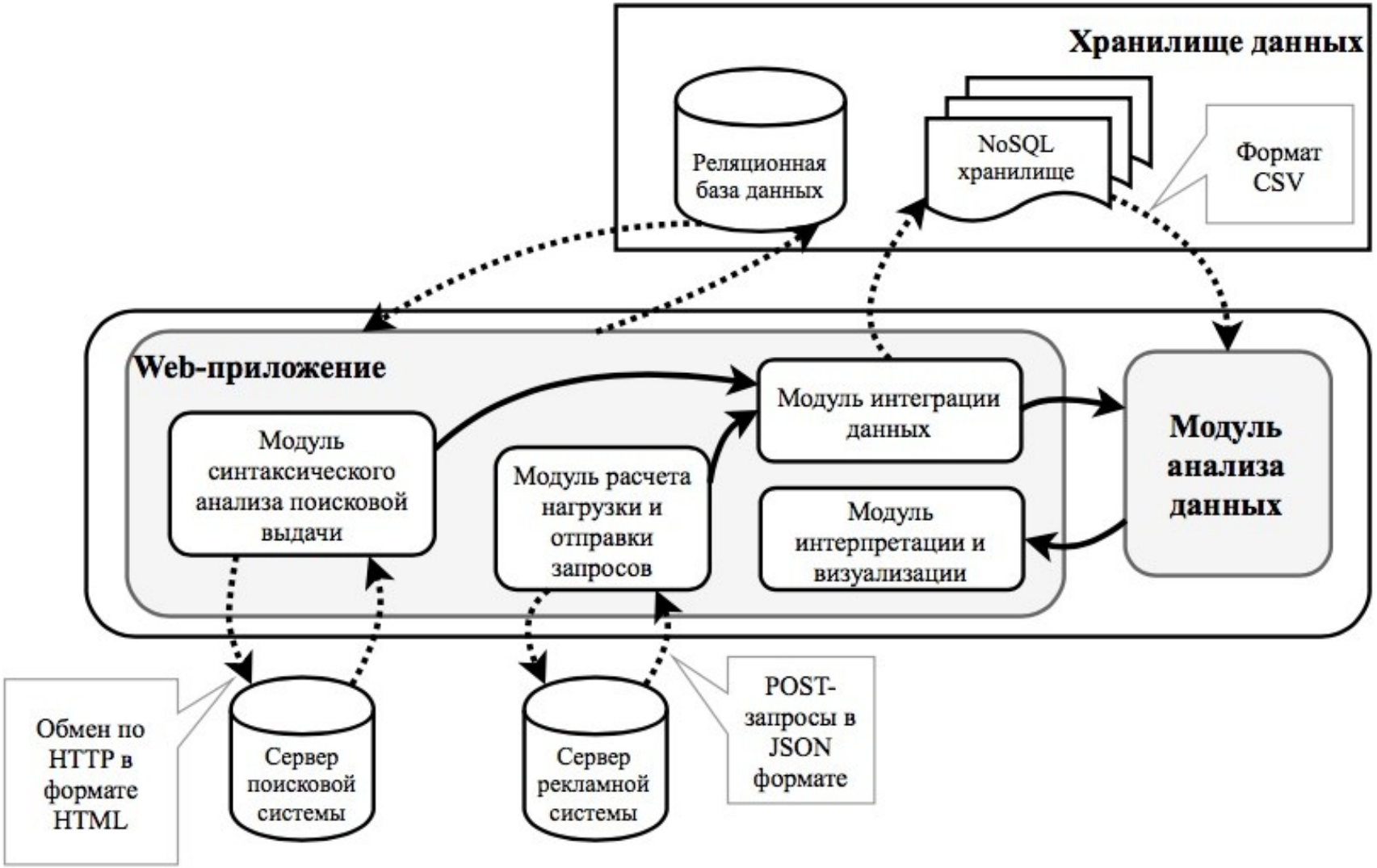


Рисунок 4.1.1. Архитектура системы поддержки принятия решений

§ 4.2. Реализация автоматизированного сбора данных

Для сбора данных о ценах позиций и получения данных рекламных кампаний реализован модуль взаимодействия с API рекламной системы Яндекс.Директ.

Запросы к API Яндекс.Директ выполняются по протоколу HTTPS методом POST.

Запрос должен содержать HTTP-заголовок Authorization с OAuth-токеном пользователя, от имени которого осуществляется запрос к API, а также может содержать другие заголовки, необходимые для обработки запроса. Ответ содержит заголовок RequestId – уникальный идентификатор запроса.

Входные и выходные структуры данных передаются в теле запроса и ответа. Взаимодействие с API производится методом запросов в формате JSON. Преимущество JSON заключается в большей компактности по сравнению с SOAP/XML, а также в скорости анализа запросов на стороне сервера рекламной системы.

Для извлечения данных о ценах позиций использовался сервис Bids и метод get, который возвращает цены для ключевых фраз по позициям показа. «После отправки POST-запроса на сервер Яндекс.Директа приложению возвращается объект или массив объектов, представленный в формате JSON. Результирующие данные в формате JSON десериализуются в объект соответствующего класса, после чего информация, необходимая для дальнейшей работы, сохраняется в локальное хранилище данных приложения» [22].

Возвращаемый объект десериализуется в объект типа динамический список структуры AuctionBids (см. Рисунок 4.2.1), для которого Position - номер позиции объявлений в формате *«PNM»* (N – номер блока выдачи, M – номер позиции в блоке), Bid - ставка рекламодателя и Price - цена размещения на позиции.

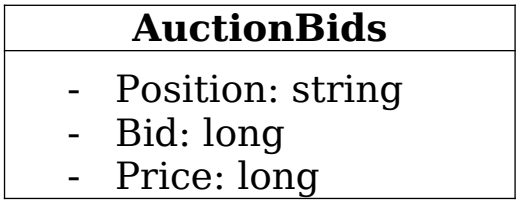


Рисунок 3.2.1. Описание структуры AuctionBids

Для получения данных о позициях размещения конкурентов в результатах поисковой выдачи на языке программирования C# реализован метод GetTop (см. Листинг в Приложении 1), для которого входным параметром является значение поискового запроса строкового типа, и возвращаемый результат – список объектов структуры SearchAd (см. Рисунок 4.2.2), где pos – номер позиции объявления, site – ссылка на сайт конкурента.

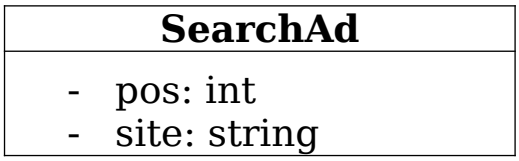


Рисунок 4.2.2. Описание структуры SearchAd

Перед отправкой запроса на адрес поисковой системы необходимо для поискового запроса провести кодирование символов согласно таблице кодирования URL, то есть привести строку формата

$$«word1\,word2[wordN]» к виду «word1\%20word2[\%20wordN]»$$

По HTTP-протоколу отправляется запрос методом Getна URL-адрес «http://yandex.ru/search/» поисковой системы и в качестве параметра указывается кодированный поисковый запрос, для которого необходимо получить результаты выдачи.

Результат запроса возвращается в формате HTML-кода страницы поисковой выдачи. С использованием библиотеки AngleSharp из кода страницы извлекаются в список все блоки типа «div», для которых атрибут «class» имеет значение «serp-adv-item».

Для каждого извлеченного блока создается объект структуры SearchAd, в который записывается номер блока, где номер блока соответствует позиции в поисковой выдаче, и ссылка на сайт, которая извлекается из кода блока как элемент типа «a» со значением атрибута «class» содержащим значение «path__item».

§ 4.3. Реализация информационного обеспечения

Для реализации системы поддержки принятия решений было спроектировано хранилище данных, которое включает реляционную базу данных и NoSQL хранилище данных.

Реляционная база данных хранит данные пользователей системы, рекламных кампаний и информацию о баллах для поддержки средства регулирования нагрузки на сервер рекламной системы. Схема реляционной базы данных представлена на рисунке 4.3.1. База состоит из 12 таблиц: AspNetUsers, Tokens, YandexUsers, YandexCampaigns, YandexAdGroups, YandexKeywords,

YandexKeywordsCompetitors, YandexCompetitors, YandexDirectUpdates, YandexDirectScores, YandexDirectServices, YandexDirectScoresDictionary.

Сущность AspNetUsers содержит основную информацию о пользователях системы (см. Приложение 2). Сущность Tokens (см. Приложение 2) содержит информацию об авторизационных OAuth-токенах пользователей рекламной системы. Токен указывается в заголовке HTTP-запроса при каждом обращении к серверу.

YandexUser, YandexCampaigns, YandexAdGroups, YandexKeywords – сущности для десериализации данных рекламного аккаунта и хранения информации о кампаниях и их составляющих.

Сущность YandexUser (см. Приложение 2) содержит основную информацию аккаунтов рекламной системы. Одному пользователю системы может соответствовать несколько аккаунтов рекламной системы. Сущность YandexCampaigns (см. Приложение 2) хранит информацию о рекламных кампаниях. Сущность YandexAdGroups (см. Приложение 2) содержит информацию о группах объявлений кампании. Сущность YandexKeywords (см. Приложение 2) содержит информацию о поисковых запросах кампании.

Сущность YandexCompetitors (см. Приложение 2) содержит информацию о конкурентах. Сущность YandexKeywordsCompetitors (см. Приложение 2) предназначена для реализации связи многие-ко-многим между сущностями YandexKeywords и YandexCompetitors и хранения дополнительной информации для корректного чтения данных о позициях конкурентов из NoSQL хранилища.

Сущность YandexDirectUpdates (см. Приложение 2) содержит информацию о времени обновления данных объектов. Сущность YandexMinScores (см. Приложение 2) содержит информацию о текущих баллах и минимальном количестве баллов для получения информации об объектах.

YandexDirectScoresDictionary, YandexDirectServices – справочники для хранения статической информации API Яндекс.Директ. Сущность YandexDirectScoresDictionary (см. Приложение 2) содержит информацию о количестве списываемых баллов за объект для каждого метода каждого сервиса API рекламной системы. Сущность YandexDirectServices (см. Приложение 2) содержит информацию о сервисах API рекламной системы.

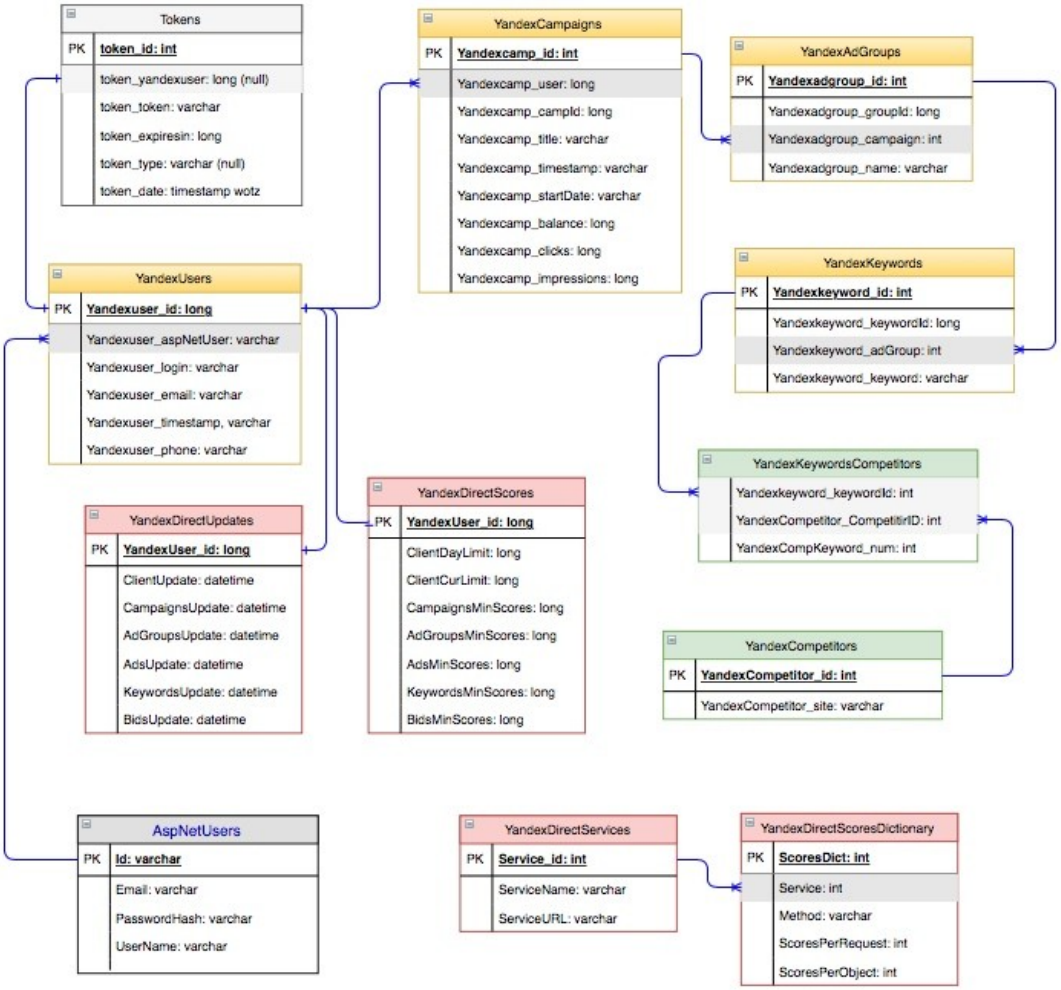


Рисунок 4.3.1. Схема реляционной базы данных

NoSQL хранилище представляет собой каталог файлов модели «ключ-значение» формата CSV с наименованиями «*keyword_[N].csv*», где [N] - идентификатор поискового запроса. Название файла является ключом, содержание файла – значением.

Каждый файл *keyword_[N].csv* хранит данные о состояниях аукциона для поискового запроса с идентификатором [N]. Для каждого файла строка – это объект структуры CompetitorDatasetItem, а столбцы соответствуют атрибутам объекта (см. Рисунок 3.1.2).

§ 4.4. Программная реализация web-приложения «Competitor»

Веб-приложение состоит из 17 классов.

Классы ClientsGetQuery, CampaignsGetQuery, AdGroupsGetQuery, AdsGetQuery, KeywordsGetQuery, BidsGetQuery необходимы для формирования структуры запросов на получение данных с сервера рекламной системы и сериализации структуры в JSON-строку.

Классы ClientsGet, CampaignsGet, AdGroupsGet, AdsGet, KeywordsGet, BidsGet необходимы для десериализации JSON-строк, получаемых от сервера рекламной системы в объекты приложения.

Класс APIYandexDirectRequestвключает методы для формирования, отправки запроса и получения ответа от сервера рекламной системы (см. Таблицу 4.4.1).

Таблица 4.4.1. Описание методов класса APIYandexDirectRequest

№	Имя метода	Результат	Назначение
1	SendAPIRequest()	void	Выполняется сериализация отправляемого объекта в JSON-формат, отправка запроса, получение и десериализация ответа от сервера рекламной системы
2	GetAPIResult()	string	Возвращает ответ сервера рекламной системы в строковом формате
3	UpdateUsersScores(List<long> units, object model)	void	Обновление в базе данных значений текущего и суточного лимита баллов аккаунта

Класс YandexDirectScores включает методы для пересчета баллов аккаунта рекламной системы (см. Таблицу 4.4.2).

Таблица 4.4.2. Описание методов класса YandexDirectScores

№	Имя метода	Результат	Назначение
1	UpdateUsersUnits(List<long>units)	void	Обновление в базе данных значений текущего и суточного лимита баллов аккаунта
2	UpdateUsersScores()	void	Пересчет минимального количества баллов для получения всех данных и сохранение в базу данных
3	CountScoresForObjectsType(YandexDirectServices service, intobjectsCount)	int	Подсчет количества баллов, необходимого для получения данных определенного сервиса
4	GetBidsPeriod()	int	Расчет интервала отправки запроса на сервер рекламной системы

Класс CompetitorDataset необходим для формирования структуры объекта дата-сета для обучения модели прогнозирования состояния аукциона. Класс SearchAd необходим для формирования структуры объекта, который представляет результат поисковой выдачи в виде списка конкурентов и их позиций.

Класс YandexDirectUpdater реализует интерфейс IJob и включает методы для автоматизации сбора данных из двух источников - рекламной системы и поисковой выдачи (см. Таблицу 4.4.3).

Таблица 4.4.3. Описание методов класса YandexDirectUpdater

№	Имя метода	Результат	Назначение
1	Execute()	void	Реализует метод интерфйеса IJob, для каждого аккаунта выполняет проверку возможности отправки запроса, получает данные о ценах позиций для поисковых запросов кампаний
2	CompetitorAddData(BidsGetbidsGetResult)	void	Для каждого поискового запроса получает результаты поисковой выдачи и интегрирует данные с последующим сохранением в хранилище данных
3	GetTop(string keyword)	List<SearchAd>	Отправляет GET-запрос и получает данные поисковой выдачи в HTML-формате, производит синтаксический анализ результата и преобразовывает в объект поисковой выдачи

Интерфейс страницы рекламной кампании с отображением прогноза аукциона по наиболее высокочастотному поисковому запросу представлен на рисунке 4.4.1.

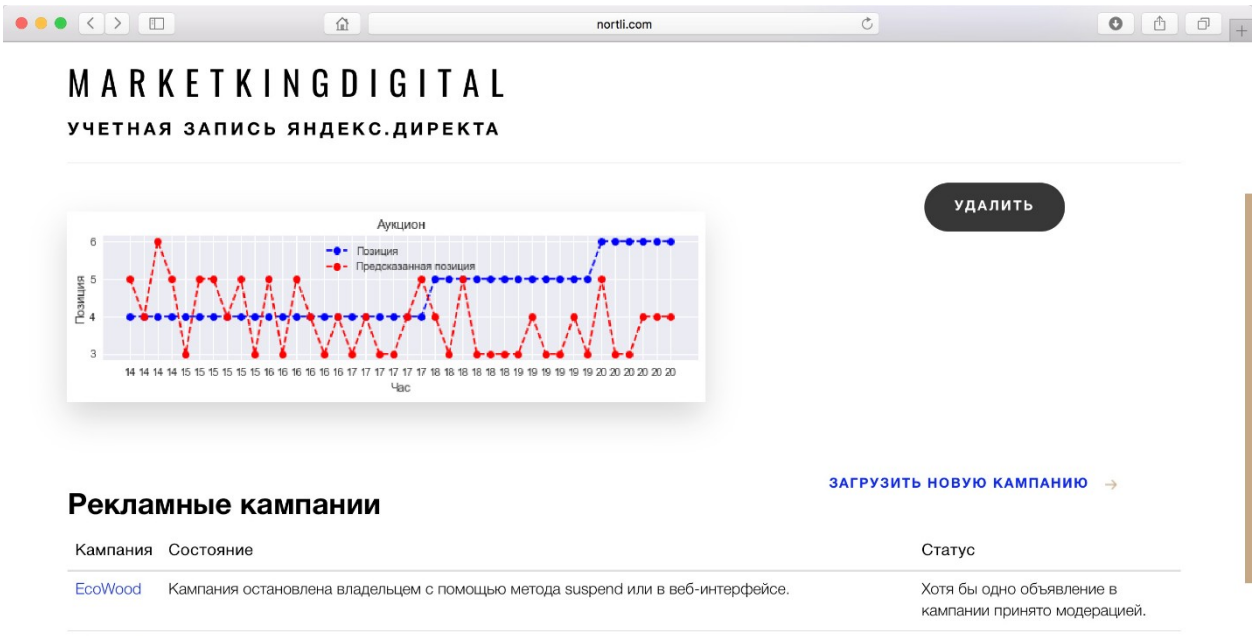


Рисунок 4.4.1. Интерфейс страницы системы

§ 4.5. Программная реализация модуля анализа данных и обучения

Модуль анализа данных реализован на языке программирования Python 3.6 в среде Spyder.

Для работы с dataset использовалась табличная структура данных DataFrame с помощью библиотеки pandas. Каждый столбец структуры данных DataFrame является объектом Series. Таким образом, набор признаков объектов выборки для задачи классификации представлен табличной структурой данных DataFrame, и последовательность меток классов для объектов выборки представлена объектом Series. Для построения матрицы парных коэффициентов корреляции использовался метод corr() библиотеки pandas.

Реализации методов машинного обучения для решения задачи многоклассовой классификации производилась с использованием библиотек scikit-learnи xgboost. В таблице 4.5.1 приведены импортируемые классы библиотек для

реализации классификаторов с вариантами передаваемых параметров.

В качестве вспомогательных инструментов для анализа результатов обучения классификаторов использовались дополнительные методы библиотеки scikit-learn (см. Табл. 4.5.2).

Методы библиотеки matplotlib использовались для вывода графиков анализируемых данных.

Для нормализации значений признаков цен использовался класс sklearn.preprocessing.StandardScaler библиотеки scikit-learn.

Таблица 4.5.1. Классы библиотек для реализации классификаторов

№	Библиотека	Классификатор	Класс библиотеки	Параметры
1	scikit-learn	Стохастический градиентный спуск	sklearn.linear_model.SGDClassifier	loss="hinge", penalty="l2"
				loss="modified_huber", penalty="l2"
				loss="log", penalty="l2"
				loss="hinge", penalty="l1"
				loss="modified_huber", penalty="l1"
				loss="log", penalty="l1"
				loss="hinge", penalty="elasticnet"
				loss="modified_huber", penalty="elasticnet"
				loss="log", penalty="elasticnet"
2	scikit-learn	Метод опорных векторов	sklearn.svm.SVC	kernel='linear', C=1,gamma='auto'
				kernel='rbf', C=1,gamma='auto'
				kernel='poly', C=1,gamma='auto'
3	scikit-learn	Дерево решений	sklearn.tree.DecisionTreeClassifier	random_state=1
4	scikit-learn	Мультиномиальный Байесовский классификатор	sklearn.naive_bayes.MultinomialNB	
5	scikit-learn	K-ближайших	sklearn.neighbors.K	n_neighbors = 10

№	Библиотека	Классификатор	Класс библиотеки	Параметры
		соседей	Neighbors Classifier	
6	xgboost	Градиентный бустинг	xgboost.XGBClassifier	max_depth=3, n_estimators=300, learning_rate=0.05

Таблица 4.5.2. Дополнительные методы библиотеки для анализа данных

№	Метод	Назначение
1	sklearn.cross_validation.cross_val_score	Выполнение кросс-валидации, возвращает массив оценок модели для каждого запуска
2	sklearn.metrics.confusion_matrix	Возвращает матрицу ошибок для классификатора
3	sklearn.metrics.classification_report	Возвращает значения метрик классификатора: точность, полнота, F-мера

§ 4.6. Апробация работы системы

Для апробации работы системы были созданы рекламные кампании 10 конкурентов, и для каждого из них была задана стратегия размещения объявлений кампании в поисковой выдаче. Апробация проводилась на данных за 3 месяца на более 12 000 записей состояний аукциона, взятых с интервалом 10 минут. Поисковая выдача содержит 8 позиций.

Пример файла для поискового запроса №49589305 (см. Рис. 3.1.2). Каждая строка представляет запись состояния аукциона, каждый столбец – признак состояния. Файл содержит более 12 000 строк и 24 столбца. В каждой строке

для двух из десяти конкурентов значение позиции равно 0, так как поисковая выдача содержит всего 8 позиций, то есть, только 8 конкурентов отображаются в выдаче, а оставшиеся 2 в выдачу не попадают.

Цель проведения апробации – сравнить состояния аукциона, спрогнозированные с помощью обученной модели, и реальные значения состояния аукциона.

Пример заданных стратегий для каждого конкурента см. в Таблице 4.6.1.

Таблица 4.6.1. Описание стратегий конкурентов

№ конкурента	**Повышенная ставка**	**Ограниченный период времени**	**Ставка по умолчанию**
1	120 руб./клик	с 18:00 до 20:00	10 руб./клик.
	110 руб./клик	с 20:00 до 22:00	
2	120 руб./клик	с 20:00 до 21:00	20 руб./клик.
	130 руб./клик	с 21:00 до 22:00	
3	120 руб./клик	выходные	40 руб./клик.
4	130 руб./клик	в будние дни в период с 12:00 до 16:00	40 руб./клик.
5	120 руб./клик	в будние дни в период с 09:00 до 22:00	50 руб./клик.
6	120 руб./клик	в будние дни в период с 09:00 до 12:00 и с 18:00 до 22:00	50 руб./клик.
7			70 руб./клик.
8			80 руб./клик.
9			90 руб./клик.
10			100 руб./клик.

Матрица корреляции для наиболее зависимых между собой признаков по результатам расчета представлена в таблице 4.6.2.

Таблица 4.6.2. Матрица корреляции наибольших коэффициентов

	hour	**part_of_day**	**comp1**	**comp5**	**comp7**
hour	1,00	0,99	0,62	-0,53	0,70
part_of_day	0,99	1,00	0,64	-0,54	0,71
comp1	0,62	0,64	1,00	-0,45	0,78
comp5	-0,53	-0,54	-0,45	1,00	-0,70
comp7	0,70	0,71	0,78	-0,70	1,00

Высокий коэффициент корреляции между значениями часа и позиции конкурента 1 объясняется стратегией конкурента на повышение ставки в течение четырех часов с 18:00 до 22:00.

Для пары значений часа и позиции конкурента 5 коэффициент корреляции отрицательный, так как при увеличении часа, то есть, к концу дня, ставки начинают повышаться у конкурента 1 и 2. При этом позиция конкурента 5 снижается, то есть между часом и позицией конкурента 5 есть обратная зависимость.

Коэффициент корреляции между значениями часа и позицией конкурента 7 выше, чем между значением часа и позицией конкурента 1, так как повышение позиции конкурента 1 происходит только в течение 4 часов (с 18:00 до 22:00), а для конкурента 7 позиция повышается в течение всего дня за счет более высокой фиксированной ставки (70 руб./клик).

Матрица парных корреляционных коэффициентов всей выборки приведена в Приложении 3.

Построим графики зависимости позиции от времени для конкурентов 1 (см. Диаграмму 4.6.1) и конкурента 7 (см. Диаграмму 4.6.2). По оси абсцисс отложены значения часа, по оси ординат - позиция конкурента.

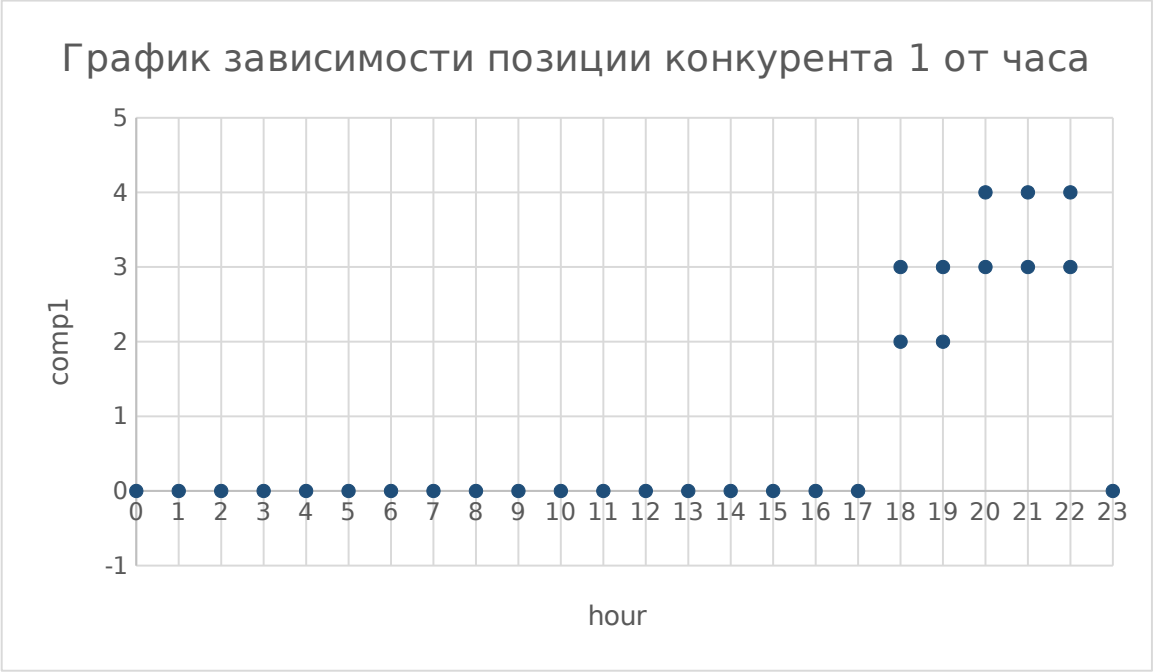


Диаграмма 4.6.1. График зависимости позиции конкурента 1 от часа

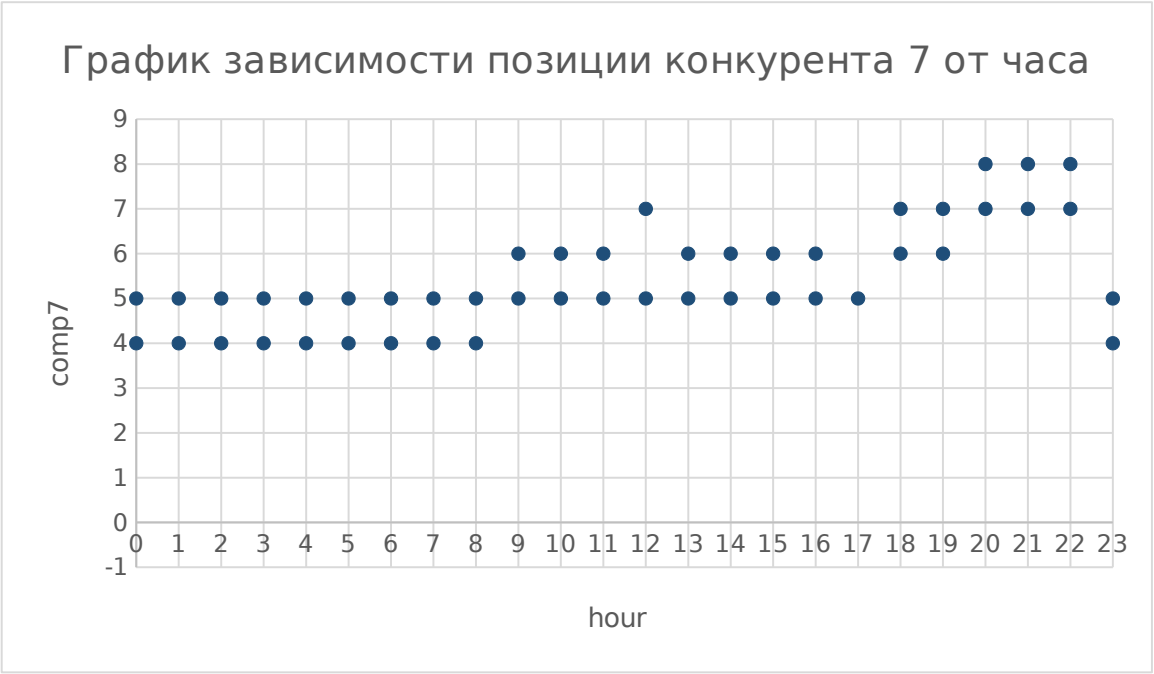


Диаграмма 4.6.2. График зависимости позиции конкурента 7 от часа

Для проведения кросс-валидации выборка была разделена на 5 равных частей. Выборка данных была разделена на обучающую и тестовую в соотношении 80% и 20% соответственно.

Показатели кросс-валидации выбранных методов классификации для всех конкурентов приведены в таблице 4.6.3. По результатам тестирования наибольшую точность показал метод k-ближайших соседей по метрике кросс-валидации. Метрики оценки метода k-ближайших соседей для прогнозирования позиций всех десяти конкурентов представлены в таблице 4.6.4.

Таблица 4.6.3. Показатели кросс-валидации методов

Метод	Стохастический градиентный спуск									Метод опорных векторов			Дерево решений	Мультиномиальный	К-ближайших	Градиентный бустинг
Параметры	l2" loss="hinge"	_ loss="modified_h	lty="l2" loss="log"	="l1" loss="hinge"	_ loss="modified_h	lty="l1" loss="log"	asticn loss="hinge"	_ loss="modified_h	elasticn loss="log"	aut kernel='linear'	a='aut kernel='rbf'	='aut kernel='poly'			n_neighbors = 10	=300 max_depth=3
Конкурент 1	0,37	0,65	0,79	0,65	0,65	0,51	0,79	0,79	0,79	0,63	0,61	0,60	0,38	0,39	0,79	0,60
Конкурент 2	0,71	0,87	0,87	0,70	0,71	0,67	0,38	0,87	0,71	0,67	0,68	0,65	0,52	0,60	0,87	0,72
Конкурент 3	0,50	0,50	0,25	0,38	0,46	0,39	0,27	0,34	0,39	0,37	0,35	0,36	0,40	0,43	0,50	0,48
Конкурент 4	0,31	0,28	0,36	0,29	0,32	0,38	0,31	0,31	0,20	0,29	0,30	0,29	0,39	0,45	0,34	0,30
Конкурент	0,30	0,19	0,19	0,25	0,23	0,22	0,18	0,21	0,21	0,25	0,24	0,26	0,18	0,14	0,30	0,30

Метод	Стохастический градиентный спуск									Метод опорных векторов			Дерево решений	Мультиномиальный	й К-ближайших	Градиентный бустинг
Параметры	l2" loss="hinge"	_ loss="modified_h	lty="l2" loss="log"	="l1" loss="hinge"	_ loss="modified_h	lty="l1" loss="log"	astic n loss="hinge"	_ loss="modified_h	elastic n loss="log"	aut kernel='linear'	a='aut kernel='poly	='aut kernel='rbf			n_neighbors = 10	=300 max_depth=3
5																
Конкурент 6	0,17	0,18	0,22	0,21	0,26	0,26	0,21	0,21	0,20	0,25	0,23	0,22	0,30	0,32	0,25	0,26
Конкурент 7	0,15	0,25	0,24	0,29	0,25	0,21	0,19	0,27	0,27	0,27	0,28	0,29	0,30	0,35	0,26	0,26
Конкурент 8	0,18	0,22	0,25	0,23	0,20	0,25	0,25	0,20	0,18	0,23	0,22	0,22	0,38	0,39	0,26	0,25
Конкурент 9	0,28	0,22	0,26	0,20	0,22	0,25	0,18	0,20	0,27	0,25	0,23	0,24	0,38	0,39	0,26	0,25
Конкурент	0,24	0,24	0,22	0,27	0,26	0,23	0,23	0,22	0,19	0,26	0,25	0,23	0,38	0,35	0,26	0,25

Метод	Стохастический градиентный спуск									Метод опорных векторов			Дерево решений	Мультиномиальный	й	К-ближайших	Градиентный бустинг
Параметры	l2"	loss="hinge"	_	lty="l2"	="l1"	_	lty="l1"	asticn	_	elasticn	aut	kernel='linear'	a='aut	='aut			n_neighbors = 10	=300	max_depth=3
10																			

Таблица 4.6.4. Метрики оценки метода k-ближайших соседей

	Оценка по кросс-валидации	Точность (Микро-мера)	Полнота (Микро-мера)	F-мера (Микро-мера)	Точность (Макро-мера)	Полнота (Макро-мера)	F-мера (Макро-мера)
Конкурент 1	0,79	0,79	0,79	0,79	0,38	0,25	0,22

Конкурент 2	0,87	0,87	0,87	0,87	0,22	0,25	0,23
Конкурент 3	0,50	0,59	0,59	0,59	0,39	0,32	0,32
Конкурент 4	0,34	0,48	0,48	0,48	0,43	0,38	0,38
Конкурент 5	0,30	0,43	0,43	0,43	0,30	0,22	0,21
Конкурент 6	0,25	0,40	0,40	0,40	0,37	0,28	0,27
Конкурент 7	0,27	0,26	0,26	0,26	0,19	0,19	0,17
Конкурент 8	0,26	0,26	0,26	0,26	0,19	0,19	0,17
Конкурент 9	0,26	0,26	0,26	0,26	0,19	0,19	0,17
Конкурент 10	0,26	0,26	0,26	0,26	0,19	0,19	0,17

После обучения модели классификации с помощью метода k-ближайших соседей самым точным оказался прогноз позиций конкурента 2 равный 87%, а самым низким оказался прогноз позиции конкурента 5 равный 0,25%.

Как видно по результатам расчета матрицы ошибок для каждого конкурента, при увеличении количества занимаемых позиций конкурента соответственно уменьшается точность классификации. Матрица ошибок классификатора по методу k-ближайших соседей для конкурента 1 представлена на рисунке 4.6.2 и для конкурента 5 представлена на рисунке 4.6.3.

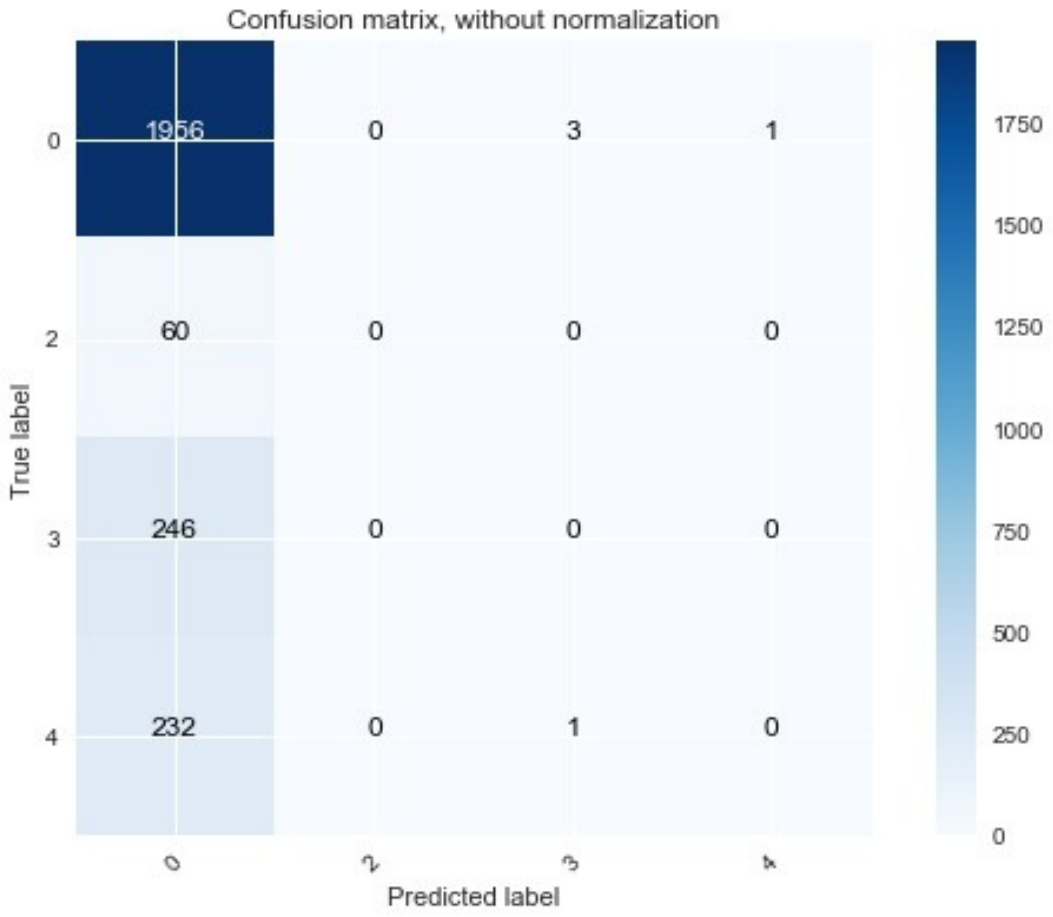


Рисунок 4.6.2. Матрица ошибок классификатора для конкурента 1

Например, по рисунку 4.7.2 можно увидеть, что конкурент 1 занимает всегда позицию 0, 2, 3 или 4.

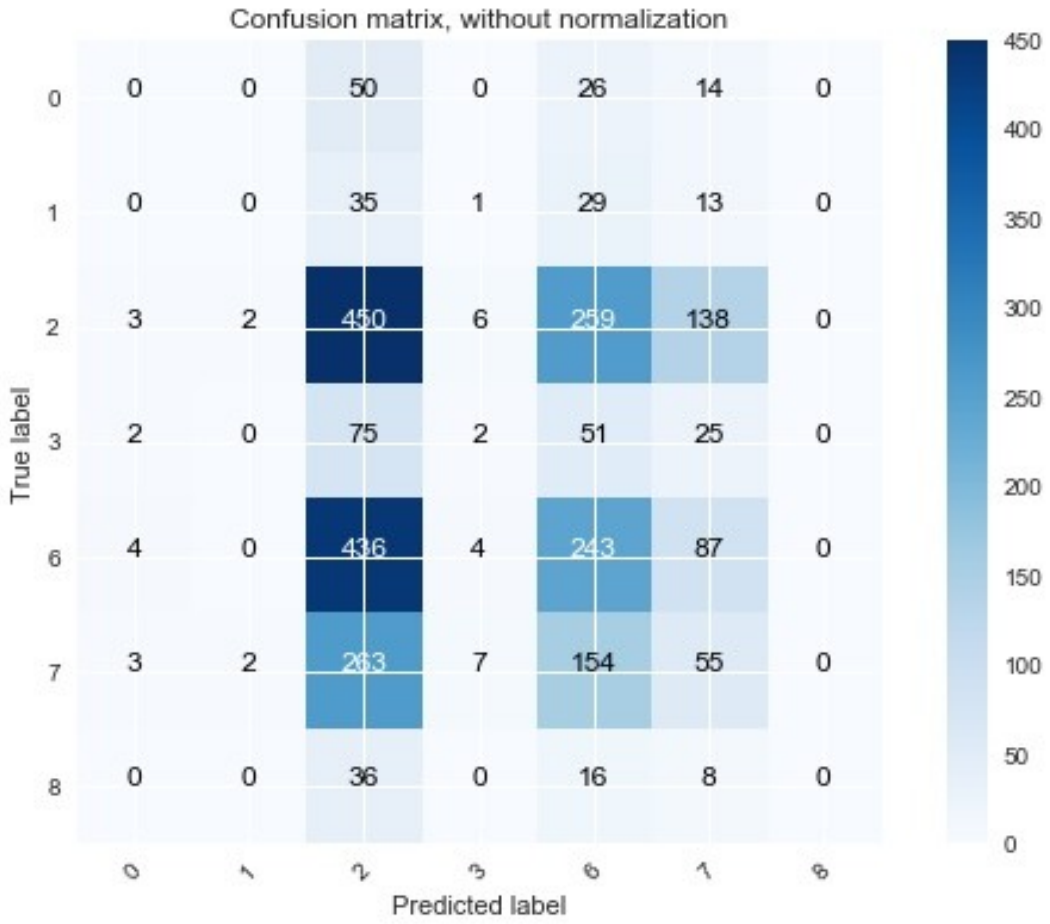


Рисунок 4.6.3. Матрица ошибок классификатора для конкурента 5

Например, по рисунку 4.7.3 можно увидеть, что конкурент 5 занимает любую позицию, кроме 4, что снижает точность прогноза позиции.

-

ЗАКЛЮЧЕНИЕ

В результате работы над магистерской диссертацией была разработана система «Competitor» поддержки принятия решений на основе анализа зависимостей показателей однотипных объектов. Система позволяет проводить мониторинг рекламных блоков поисковой выдачи и получать прогноз позиций размещения объявлений конкурентов.

Все поставленные задачи были достигнуты:

- изучены методы анализа данных, методы анализа зависимостей параметров объектов;
- определены требования к разрабатываемой системе;
- автоматизирован сбор данных;
- разработан алгоритм поддержки средств регулирования нагрузки на сервер;
- выполнен анализ и подготовка данных для обучения;
- применены алгоритмы машинного обучения и выполнена оценка качества моделей обучения;
- работа системы апробирована на реальных данных, получены достоверные результаты.

Знания позиций размещения объявлений конкурентов позволят специалисту по поисковой рекламе устанавливать дополнительные настройки показа объявлений с целью увеличения времени отображения объявлений на первых позициях поисковой выдачи.

В результате проведения исследования сделан вывод о том, что подготовительный этап, включающий анализ, сбор и подготовку данных, занимает около 60% времени всего

исследования, а сам процесс моделирования около 10-15% за счет применения открытых библиотек анализа данных.

Получено согласие со стороны ООО «Инкомтехнологии Групп» на внедрение разработанной системы «Competitor» в работу организации. Это позволит специалистам ООО «Инкомтехнологии Групп» не только более точно настраивать стратегии размещения объявлений своих клиентов, но и использовать наличие такой системы как конкурентное преимущество своей компании при проведении деловых переговоров.

Программный код модуля автоматизированного сбора данных был зарегистрирован (свидетельство о государственной регистрации программы для ЭВМ №2017664023).

СПИСОК ЛИТЕРАТУРЫ

1. 2017 was the year digital ad spending finally beat TV [Электронный ресурс]. URL: https://www.recode.net/2017/12/4/16733460/2017-digital-ad-spend-advertising-beat-tv (дата обращения: 22.06.2018).
2. Аукцион [Электронный ресурс]. URL: https://support.google.com/adwords/answer/142918?hl=ru&ref_topic=24937 (дата обращения: 20.02.2018).
3. Бабаев, А., Евдокимов, Н., Иванов, А. Контекстная реклама — СПб.: Питер, 2011. — 304 с.: ил. ISBN 978-5-459-00335-2.
4. Боганюк, Ю.В., Воробьева, М.С. Алгоритм взаимодействия с сервером контекстной рекламы для уменьшения нагрузки на сервер / МАТЕМАТИЧЕСКОЕ И ИНФОРМАЦИОННОЕ МОДЕЛИРОВАНИЕ: сборник научных трудов. Вып.15. Тюмень: Издательство Тюменского государственного университета, 2017. 534 с.
5. Боганюк, Ю.В., Воробьева, М.С. Разработка web-приложения для автоматизации управления контекстом интернет-страниц при выдаче поисковой системы / Издательство Тюменского государственного университета, 2016. 55 с.
6. Боганюк, Ю.В., Воробьев, А.М. Разработка алгоритма автоматического пересчета параметров контекста интернет-страниц при поисковой выдаче / ИННОВАЦИОННЫЕ ТЕХНОЛОГИИ НАУЧНОГО РАЗВИТИЯ: сборник статей Международной научно-

практической конференции (25 октября 2015 г. 2015 г., г.Пермь). / в 2 ч. Ч.2 – Уфа: АЭТЕРНА, 2015. – 286 с.

7. Боганюк, Ю.В. Разработка высокоуровневой архитектуры веб-приложения для автоматизации управления контекстными кампаниями с учетом ограничений / Материалы 54-й Международной научной студенческой конференции МНСК-2016: Информационные технологии / Новосиб. гос. ун-т. Новосибирск, 2016. 249 стр.
8. Виды поисковых запросов [Электронный ресурс]. URL: http://www.vsetyrabota.ru/semanticheskoe-yadro/184-vidy-poiskovykh-zaprosov (дата обращения: 15.02.2018).
9. Давыдов, Д.В., Измалков, С.Б., Смирнов, А.С. Рынки контекстной рекламы: эмпирические и экспериментальные работы / Журнал Новой экономической ассоциации, №4 (28), с. 56-73
10. Клиентский сценарий на веб-страницах ASP.NET [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/3hc29e2a(v=vs.100).aspx (дата обращения: 22.02.2018).
11. Высшая школа экономики. Многоклассовая классификация [Электронный ресурс]. URL: https://www.coursera.org/lecture/vvedenie-mashinnoe-obuchenie/mnoghoklassovaia-klassifikatsiia-P9Zun (дата обращения: 25.05.2018).
12. Нейский, И.М. Классификация и сравнение методов кластеризации [Электронный ресурс]. URL: it-claim.ru/Persons/Neyskiy/Article2_Neiskiy.pdf (дата обращения: 18.04.2018).

13. Открытый курс машинного обучения. Тема 10. Градиентный бустинг [Электронный ресурс]. URL: https://habr.com/company/ods/blog/327250/ (дата обращения: 20.03.2018).
14. Первоначальные сведения [Электронный ресурс]. URL: https://tech.yandex.ru/direct/doc/dg-v4/concepts/WhereToBegin-docpage/ (дата обращения: 20.02.2018).
15. Подробно о контекстной рекламе [Электронный ресурс]/icontext. – режим доступа: http://www.icontext.ru/context/context-in-depth/ (дата обращения: 15.04.2018).
16. Позиции на поиске [Электронный ресурс]. URL: https://yandex.ru/support/direct/general/positions.xml (дата обращения: 12.02.2018).
17. Смирнов, А.С. Рынки контекстной рекламы: подходы и теоретические модели / ЭКОНОМИКА И МАТЕМАТИЧЕСКИЕ МЕТОДЫ, 2015, том 51, № 4, с. 14–24
18. Соколов Е. Выбор моделей и критерии качества [Электронный ресурс]. URL: http://www.machinelearning.ru/wiki/images/1/1c/Sem06_metrics.pdf
19. Требования к приложению [Электронный ресурс]. URL: https://tech.yandex.ru/direct/doc/dg-v4/concepts/requirements-docpage/ (дата обращения: 08.02.2018).
20. Троелсен Э. Язык программирования C# и платформа .NET 4.5 [Текст] / Троелсэн Э.; перевод с

английского Артеменко Ю. М. – 6-е изд. – М.: ООО «И. Д. Вильямс», 2013. – 1312 с.

21. Трейдинг и машинное обучение с подкреплением [Электронный ресурс]. URL: http://ai-news.ru/2018/02/trejding_i_mashinnoe_obuchenie_s_podkreplениem.html (дата обращения: 24.04.2018).
22. Формат взаимодействия [Электронный ресурс]. URL: https://tech.yandex.ru/direct/doc/dg/concepts/format-docpage/
(дата обращения: 15.03.2018).
23. Формат JSON [Электронный ресурс]. URL: https://tech.yandex.ru/direct/doc/dg-v4/concepts/JSON-docpage/
(дата обращения: 08.01.2018).
24. Х.Бринк, Дж. Ричардс, М. Феверолф. Машинное обучение. – СПб.: Питер, 2017.
25. Choosing the right estimator [Электронный ресурс]. URL: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html (дата обращения: 05.04.2018).
26. CRISP-DM: проверенная методология для Data Scientist-ов [Электронный ресурс]. URL: http://www.pvsm.ru/algoritmy/255490 (дата обращения: 15.03.2018).
27. JavaScript [Электронный ресурс]. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 28.01.2018).
28. Microsoft. Руководство Microsoft по проектированию архитектуры приложений, 2е издание.

29. Боганюк, Ю.В., Воробьева, М.С. Свидетельство о государственной регистрации программы для ЭВМ № 2017664023 «Разработка модуля «SearchAdImport v2.2» для загрузки иерархических объектов поисковой выдачи» от 14.12.2017.

ПРИЛОЖЕНИЕ 1

ЛИСТИНГ КОДА АВТОМАТИЧЕСКОЙ ВЫГРУЗКИ ДАННЫХ

```
public List<SearchAd>GetTop(string keyword)
{
	stringkeyword_format = keyword.Replace(" ", "%20");
	stringurl = "http://yandex.ru/search/?lr=55&text="+keyword_format;
	List<SearchAd>searchAds = new List<SearchAd>();
	using (HttpClient client = new HttpClient())
	using (HttpResponseMessage response = client.GetAsync(url).Result)
	using (HttpContent content = response.Content)
	{
		string result = content.ReadAsStringAsync().Result;
		var parser = new HtmlParser();
		var document = parser.Parse(result);
		varadsItems = document.All.Where(m =>m.LocalName == "li"
		&&m.ClassList.Contains("serp-adv-item"));
		intpos = 1;
		foreach (var ad in adsItems)
		{
			varadParse = parser.Parse(ad.InnerHtml);
			varsiteNode = adParse.All.Where(a =>a.LocalName == "a"
			&&a.ClassList.Contains("path__item")).FirstOrDefault();
			if (siteNode != null)
			{
				varsiteNodeParse = parser.Parse(siteNode.InnerHtml);
				string site = siteNodeParse.All.Where(b =>b.LocalName
				== "b").First().InnerHtml;
				searchAds.Add(new SearchAd(pos, site));
				pos++;
			}
		}
	}
	returnsearchAds;
}
```

ПРИЛОЖЕНИЕ 2

ОПИСАНИЕ ТАБЛИЦ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Таблица 1. Сущность AspNetUsers

№	Имя поля	Тип	Назначение
1	Id (PK)	varchar(128)	Идентификатор пользователя
2	Email	varchar(255)	Электронный адрес пользователя
3	PasswordHash	varchar(255)	Хэш пароля пользователя
4	UserName	varchar(255)	Имя пользователя

Таблица 2. Сущность Tokens

№	Имя поля	Тип	Назначение
1	token_id (PK)	Int	Идентификатор токена
2	token_yandexuser(FK)	long	Идентификатор аккаунта Яндекс.Директ
3	token_token	varchar(255)	Значение токена
4	token_expiresin	long	Количество секунд до окончания действия токена
5	token_type	varchar(255)	Тип токена (тип определяется рекламной системой)
6	token_date	timestamp	Дата выдачи токена

Таблица 3. Сущность YandexUsers

№	Имя поля	Тип	Назначение
1	Yandexuser_Id (PK)	long	Идентификатор аккаунта в рекламной системе

№	Имя поля	Тип	Назначение
2	Yandexuser_aspNetUser (FK)	varchar(255)	Идентификатор пользователя приложения
3	Yandexuser_login	varchar(255)	Логин аккаунта в рекламной системе
4	Yandexuser_email	varchar(255)	Email пользователя аккаунта
5	Yandexuser_timestamp	varchar(255)	Дата и время последней проверки наличия изменений данных
6	Yandexuser_phone	varchar(255)	Телефон пользователя аккаунта

Таблица 4. Сущность Campaigns

№	Имя поля	Тип	Назначение
1	Yandexcamp_Id (PK)	int	Идентификатор кампании
2	Yandexcamp_user (FK)	long	Идентификатор аккаунта рекламной системы
3	Yandexcamp_campId	long	Идентификатор кампании в рекламной системе
4	Yandexcamp_title	varchar(255)	Название кампании
5	Yandexcamp_timestamp	varchar(255)	Дата и время последней проверки наличия изменений данных кампании
6	Yandexcamp_startDate	varchar(255)	Дата запуска кампании в рекламной системе
7	Yandexcamp_balance	long	Текущий баланс рекламной кампании
8	Yandexcamp_clicks	long	Количество кликов всех объявлений кампании
9	Yandexcamp_impressions	long	Количество показов всех объявлений кампании

Таблица 5. Сущность YandexAdGroups

№	Имя поля	Тип	Назначение
1	Yandexadgroup_Id (PK)	int	Идентификатор группы объявлений
2	Yandexadgroup_campaign	int	Идентификатор кампании, которой принадлежит группа объявлений
3	Yandexadgroup_group_ID	long	Идентификатор группы объявлений в рекламной системе
4	Name	varchar(255)	Заголовок группы объявлений

Таблица 6. Сущность YandexKeywords

№	Имя поля	Тип	Назначение
1	Yandexkeyword_Id (PK)	int	Идентификатор поискового запроса
2	Yandexkeyword_keywordId	long	Идентификатор поискового запросав рекламной системе
3	Yandexkeyword_adGroup	int	Идентификатор группы объявлений, которой принадлежит поисковый запрос
4	Yandexkeyword_keyword	varchar(255)	Значение поискового запроса

Таблица 7. СущностьYandexCompetitors

№	Имя поля	Тип	Назначение
1	YandexCompetitor_id (PK)	int	Идентификатор конкурента
2	YandexCompetitor_site	long	Адрес сайта конкурента

Таблица 8. СущностьYandexKeywordsCompetitors

№	Имя поля	Тип	Назначение
1	Yandexkeyword_keywordId (PK)	int	Идентификатор поискового запроса
2	YandexCompetitor_CompetitorId (PK)	long	Идентификатор конкурента
3	YandexCompKeyword_num	int	Номер конкурента в файле, соответствующем поисковому запросу

Таблица 9. Сущность YandexDirectUpdates

№	Имя поля	Тип	Назначение
1	YandexkUser_Id (PK)	long	Идентификатор аккаунта
2	ClientUpdate	datetime	Дата и время обновления баллов аккаунта
3	CampaignsUpdate	datetime	Дата и время обновления данных кампании
4	AdGroupsUpdate	datetime	Дата и время обновления данных групп объявлений
5	AdsUpdate	datetime	Дата и время обновления данных объявлений
6	KeywordsUpdate	datetime	Дата и время обновления данных поисковых запросов
7	BidsUpdate	datetime	Дата и время обновления данных ставок

Таблица 10. Сущность YandexMinScores

№	Имя поля	Тип	Назначение
1	YandexkUser_Id (PK)	long	Идентификатор аккаунта

№	Имя поля	Тип	Назначение
2	ClientDayLimit	long	Суточный лимит баллов аккаунта
3	ClientCurLimit	long	Текущий лимит баллов аккаунта
4	CampaignsMinScores	long	Минимальное количество баллов для получения данных рекламной кампании
5	AdGroupsMinScores	long	Минимальное количество баллов для получения данных групп объявлений
6	AdsMinScores	long	Минимальное количество баллов для получения данных объявлений
7	KeywordsMinScores	long	Минимальное количество баллов для получения данных поисковых запросов
8	BidsMinScores	long	Минимальное количество баллов для получения данных ставок

Таблица 11. Сущность YandexDirectScoresDictionary

№	Имя поля	Тип	Назначение
1	ScoresDict(PK)	int	Идентификатор записи
2	Service (FK)	int	Идентификатор сервиса API
3	Method	varchar(255)	Метод сервиса API
4	ScoresPerRequest	int	Количество баллов, списываемое за вызов метода API
5	ScoresPerObject	int	Количество баллов, списываемое за 1 объект

Таблица 12. Сущность YandexDirectServices

№	Имя поля	Тип	Назначение
1	ServiceId (PK)	int	Идентификатор сервиса API
2	ServiceName	int	Название сервиса API
3	ServiceURL	varchar(255)	URL-адрес для отправки запросов сервису API

ПРИЛОЖЕНИЕ 3

МАТРИЦА ПАРНЫХ КОЭФФИЦИЕНТОВ ПРИЗНАКОВ ВЫБОРКИ

Признак	month	day_of_we	part_of_da	hour	minute	p11	p12	p13	p14	p21	p22	p23	p24	comp1	comp2	comp3	comp4	comp5	comp6	comp7	comp8	comp9	comp10
month	**1,00**	0,03	0,00	0,00	0,00	0.02	0,00	0,00	-0,01	-0,01	0,00	0,00	0,00	0,00	0,00	-0,02	0,00	0,00	0,01	0,01	0,01	0,01	0,01
day_of_week	0,03	**1,00**	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	0,01	0,00	0,00	0,00	-0,06	-0,04	-0,51	0,15	0,37	0,38	-0,05	-0,05	-0,05	-0,05
part_of_day	0,00	0,00	**1,00**	0,99	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,64	0,46	-0,40	-0,55	-0,54	-0,19	0,72	0,70	0,70	0,70
hour	0,00	0,00	0,99	**1,00**	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,62	0,46	-0,39	-0,53	-0,53	-0,17	0,70	0,69	0,69	0,69
minute	0,00	0,00	0,00	0,00	**1,00**	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
p11	0.02	0,00	0,00	0,00	0,00	**1,00**	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
p12	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Признак	month	&day_of_we	part_of_da	hour	minute	p11	p12	p13	p14	p21	p22	p23	p24	comp1	comp2	comp3	comp4	comp5	comp6	comp7	comp8	comp9	comp10
p13	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**	0,00	0,00	0,00	0,01	-0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
p14	-0,01	-0,01	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
p21	-0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,01	**1,00**	0,00	0,00	0,00	0,00	0,00	-0,01	0,01	0,01	0,00	0,00	0,00	0,00	0,00
p22	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**1,00**	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
p23	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	**1,00**	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
p24	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	0,00	0,00	0,00	0,00	**1,00**	-0,01	-0,01	0,00	0,00	0,00	-0,01	-0,01	-0,01	-0,01	-0,01
comp1	0,00	-0,06	0,64	0,62	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	**1,00**	0,44	-0,42	-0,41	-0,45	-0,44	0,78	0,78	0,78	0,78
comp2	0,00	-0,04	0,46	0,46	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	0,44	**1,00**	-0,43	-0,57	-0,40	-0,31	0,63	0,63	0,63	0,63
comp3	-	-	-	-	0,0	0,0	0,0	0,0	0,0	-	0,0	0,0	0,0	-	-	**1,0**	0,0	-	0,0	-	-	-	-

Признак	month	&day_of_we	part_of_da	hour	minute	p11	p12	p13	p14	p21	p22	p23	p24	comp1	comp2	comp3	comp4	comp5	comp6	comp7	comp8	comp9	comp10
	0,02	0,51	0,40	0,39	0	0	0	0	0	0,01	0	0	0	0,42	0,43	**0**	4	0,11	9	0,49	0,49	0,49	0,49
comp4	0,00	0,15	-0,55	-0,53	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	-0,41	-0,57	0,04	**1,00**	0,55	-0,03	-0,60	-0,60	-0,60	-0,60
comp5	0,00	0,37	-0,54	-0,53	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	-0,45	-0,40	-0,11	0,55	**1,00**	0,35	-0,70	-0,70	-0,70	-0,70
comp6	0,01	0,38	-0,19	-0,17	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	-0,44	-0,31	0,09	-0,03	0,35	**1,00**	-0,47	-0,47	-0,47	-0,47
comp7	0,01	-0,05	0,71	0,70	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	0,78	0,63	-0,49	-0,60	-0,70	-0,47	**1,00**	1,00	1,00	1,00
comp8	0,01	-0,05	0,70	0,69	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	0,78	0,63	-0,49	-0,60	-0,70	-0,47	1,00	**1,00**	1,00	1,00
comp9	0,01	-0,05	0,70	0,69	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	0,78	0,63	-0,49	-0,60	-0,70	-0,47	1,00	1,00	**1,00**	1,00
comp10	0,01	-0,05	0,70	0,69	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,01	0,78	0,63	-0,49	-0,60	-0,70	-0,47	1,00	1,00	1,00	**1,00**

ПРИЛОЖЕНИЕ 4

ЛИСТИНГ КОДА ОБМЕНА ДАННЫМИ

```
public bool IsTimeToUpdateBids()
{
        bool isTime = false;
        using (Data.YandexDirectDbContext _context =
        Data.YandexDirectDbContext.YandexDirectDbContextCreate())
        {
                YandexDirectUpdates usersUpdates =
                _context.YandexDirectUpdates.FirstOrDefault(uu => uu.YandexuserId
                == YandexuserId);
                YandexDirectScores usersScores =
                _context.YandexDirectScores.FirstOrDefault(us => us.YandexuserId
                == YandexuserId);
                if (usersScores == null) return false;
                usersScores.UpdateUsersScores();
                if (usersUpdates.BidsUpdate == null) return false;
                TimeSpan timeSpan = DateTime.Now.Subtract((DateTime)
        (usersUpdates.BidsUpdate));
                isTime = (Convert.ToInt32(Math.Floor(timeSpan.TotalMinutes))) >=
                usersScores.GetBidsPeriod();
        }
        return isTime;
}
public class APIYandexDirectRequest
{
        YandexDirectDbContext _context;
        YandexDirectError apiResultError;
        string resultString;
        string units;
        private object apiModel;
        private string token;
        private string url;
        public APIYandexDirectRequest(object _apiModel, string _token, string _url)
        {
                apiModel = _apiModel;
                token = _token;
                url = _url;
        }
        public void SendAPIRequest()
        {
                string requestJson = JsonConvert.SerializeObject(apiModel);
                HttpClient client = new HttpClient();
                client.DefaultRequestHeaders.Clear();
                client.DefaultRequestHeaders.Add("Authorization", "Bearer " +
        token);
                client.DefaultRequestHeaders.Add("Accept-Language", "ru");
                var answer = client.PostAsync(url, new StringContent(requestJson,
                Encoding.UTF8, "application/json"));
                var resultJson =
                JsonConvert.DeserializeObject(answer.Result.Content.ReadAsStringA
                sync().Result);
```

```
        resultString = JsonConvert.SerializeObject(resultJson);
        apiResultError =
JsonConvert.DeserializeObject<YandexDirectError>(resultString);
        if (apiResultError.error == null)
                LogRequest("", resultString, "request");
        else
                LogRequest("", resultString, "error");
        KeyValuePair<string, IEnumerable<string>> unit =
        answer.Result.Headers.FirstOrDefault(h => h.Key == "Units");
        if (unit.Value == null) return;
        List<long> units = new List<long>();
        foreach (var u in unit.Value.First().Split('/'))
        {
                units.Add(Convert.ToInt64(u));
        }
        UpdateUsersScores(units, apiModel);
}
private void UpdateUsersScores(List<long> units, object model)
{
        using (_context =
YandexDirectDbContext.YandexDirectDbContextCreate())
        {
                Tokens yandexuser_token;
                using (VKDbContext vkContext =
        VKDbContext.VKDbContextCreate())
                {
                        yandexuser_token = vkContext.Tokens.FirstOrDefault(t
                        => t.TokenToken == token && t.TokenYandexuser !=
                        null);
                }
                if (yandexuser_token != null)
                {
                        YandexUsers user = _context.YandexUsers.First(u =>
                        u.YandexuserId == yandexuser_token.TokenYandexuser);
                        YandexDirectScores userScores =
                _context.YandexDirectScores
                        .FirstOrDefault(s => s.YandexuserId ==
                user.YandexuserId);
                        YandexDirectUpdates userUpdates =
                _context.YandexDirectUpdates
                        .FirstOrDefault(u => u.YandexuserId ==
                user.YandexuserId);
                        if (userScores == null)
                        {
                                userScores = new YandexDirectScores
                                {
                                        YandexuserId = user.YandexuserId
                                };
                                userScores.UpdateUsersUnits(units);
                                userScores.UpdateUsersScores();
                                userUpdates.CampaignsUpdate =
                                DateTime.Parse(user.YandexuserTimestamp, null,
                                System.Globalization.DateTimeStyles.RoundtripKin
                        d);
                                userUpdates.ClientUpdate = DateTime.Now;
                                _context.YandexDirectScores.Add(userScores);
```

```
                        _context.SaveChanges();
                    }
                    else
                    {
                        if (units[0] + units[1] > userScores.ClientCurLimit)
                            userUpdates.ClientUpdate = DateTime.Now;
                        userScores.UpdateUsersUnits(units);
                        userScores.UpdateUsersScores();
                        _context.SaveChanges();
                    }
                SimpleLogRequest(units, model, user.YandexuserId);
                }
            }
        }
}
```