МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

*Применение технологий Hadoop для анализа "больших данных" на примере задач геоаналитики и машинного обучения*

ВКР *Магистерская диссертация*

*09.04.02* *Информационные системы и технологии*

*Технологии разработки информационных систем*

Допущено к защите в ГЭК

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_*Н.А. Тюкачев*, *к.ф.-м.н., доцент* \_\_.\_\_.2016

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_*А.Г. Чурсин, 2 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ *Н.А. Тюкачев, к.ф.-м.н., доцент*

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_ *Д.И. Соломатин*, *ст. преподаватель*

Воронеж2016

МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Факультет компьютерных наук

Кафедра программирования и информационных технологий

**УТВЕРЖДАЮ**

заведующий кафедрой

\_\_\_\_\_\_\_\_ Н.А. Тюкачев

*подпись, расшифровка подписи*

\_\_.\_\_.2016

**ЗАДАНИЕ**

**НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ ОБУЧАЮЩЕГОСЯ** Чурсина Андрея Георгиевича

1. Тема работы «Применение технологий Hadoop для анализа "больших данных" на примере задач геоаналитики и машинного обучения», утверждена решением ученого совета факультета компьютерных наук от \_\_.\_\_.2016
2. Направление подготовки 09.04.02 Информационные системы и технологии
3. Срок сдачи студентом законченной работы \_\_\_.\_\_. 2016
4. Календарный план:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Структура ВКР | Сроки выполнения | Примечание |
|  | Введение | 04.09.15-09.09.15 |  |
| 1 | Постановка задачи | 10.09.15-15.09.15 |  |
| 2.1 | Актуальность выбора технологий | 16.09.15-31.10.15 |  |
| 2.2 | Анализ требований и подходов к проекту | 01.11.15-14.11.15 |  |
| 2.3 | Анализ существующих решений. Подсистемы Hadoop | 15.11.15-23.11.15 |  |
| 2.4 | Сравнение исполняемых сред | 24.11.15-30.11.15 |  |
| 2.5 | Языки запросов к данным | 01.12.15-07.12.15 |  |
| 2.6 | Анализ организации хранения данных в системе | 08.12.15-15.12.15 |  |
| 2.7 | Анализ требований к отчетным формам задачи геоаналитики | 16.12.15-20.12.15 |  |
| 2.8 | Справочник базовых станций | 21.01.16-05.01.16 |  |
| 2.9 | Справочник покрытия БС | 06.12.16-11.01.16 |  |
| 2.10 | Справочник соседних БС | 12.01.16-16.01.16 |  |
| 2.11 | Справочник сетки 500 на 500 | 17.01.16-25.01.16 |  |
| 2.12 | Анализ источников входных данных | 26.01.16-30.01.16 |  |
| 2.13 | Анализ архитектуры приложения | 31.01.16-10.02.16 |  |
| 3 | Аппаратные и программные требования | 11.02.16-20.02.16 |  |
| 3.1 | Программные средства для тестового окружения | 21.02.16-25.02.16 |  |
| 3.2 | Программные средства реализации | 26.02.16-29.02.16 |  |
| 4.1 | Реализация модуля формирования справочника уникальных базовых станций | 01.03.16-16.03.16 |  |
| 4.2 | Реализация модуля формирования справочников покрытия, соседних базовых станций и отчета по сетке 500 на 500 | 17.03.16-31.03.16 |  |
| 4.3 | Реализация результирующего отчета «Монобренд» | 01.04.16-15.04.16 |  |
| 4.4 | Реализация модуля по определению пола абонента | 16.04.16-30.04.16 |  |
| 4.5 | Автоматизация модулей по формированию отчетов геоаналитики | 01.05.16-06.05.16 |  |
| 4.6 | Анализ полученных результатов | 07.05.16-15.05.16 |  |
|  | Заключение | 16.05.16-20.05.16 |  |
|  | Список использованных источников | 21.05.16-23.05.16 |  |
|  | Приложения | 24.05.16-27.05.16 |  |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_ Чурсин А.Г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_ Тюкачев Н.А.

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

Выпускная квалификационная работа представлена на кафедру \_\_.\_\_.20\_\_

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*должность, ученая степень, ученое звание*

Выпускная квалификационная работа на тему «Применение технологий Hadoop для анализа "больших данных" на примере задач геоаналитики и машинного обучения»

допущена к защите в ГЭК \_\_.\_\_.20\_\_

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_.\_\_.20\_\_

*подпись, расшифровка подписи*

Реферат

Текстовый документ 92 с., 32 рисунка, 10 таблиц, 34 источника, 7 приложений.

ГЕОАНАЛИТИКА, МАШИННОЕ ОБУЧЕНИЕ, МЕТОД ОПОРНЫХ ВЕКТОРОВ, ТЕЛЕКОММУНИКАЦИИ, ХРАНИЛИЩЕ ДАННЫХ, BIG DATA, HADOOP, MAP-REDUCE.

Объектом исследования являются методики геоаналитики и алгоритмы машинного обучения применительно к задачам оптимального расположения офисов продаж поставщика услуг сотовой связи и определения пола абонентов соответственно.

Цель работы – исследование путей наилучшего развития бизнеса телекоммуникационной компании, разработка программных продуктов для выгрузки показателей для единой системы отчетности.

В процессе выполнения работы проводится анализ предметной области, исследование подходов к решению задачи, выбор технических средств реализации, разрабатываются алгоритмы формирования вспомогательных справочников и результирующего отчета, разрабатывается предиктивная модель на основе метода опорных векторов для задачи машинного обучения, реализуются программные модули для решения поставленных задач.

В результате исследования для задачи геоаналитики реализованы алгоритмы получения справочников уникальных базовых станций (их соседей и покрытия) для крупных городов России. По задаче машинного обучения реализована предиктивная модель определения пола абонента для активной абонентской базы оператора. На основе предложенных алгоритмов разработано программное обеспечение.

Степень внедрения – результаты включены в релиз проекта, как продуктивные задачи для автоматизированной работы на промышленном кластере.

Эффективность работы – разработанное программное обеспечение позволяет производить расчеты в указанные промежутки времени. Полученные результаты соответствую критериям качества со стороны экспертной группой лаборатории BigData в ПАО «ВымпелКом».

**Содержание**

[Введение 8](#_Toc454371427)

[1 Постановка задачи 11](#_Toc454371428)

[2 Анализ задачи 14](#_Toc454371429)

[2.1 Актуальность выбора технологий 14](#_Toc454371430)

[2.2 Анализ требований и подходов к проекту 17](#_Toc454371431)

[2.3 Анализ существующих решений. Подсистемы Hadoop 18](#_Toc454371432)

[2.4 Сравнение исполняемых сред 23](#_Toc454371433)

[2.5 Языки запросов к данным 27](#_Toc454371434)

[2.6 Анализ организации хранения данных в системе 27](#_Toc454371435)

[2.7 Анализ требований к отчетным формам задачи геоаналитики 28](#_Toc454371436)

[2.8 Справочник базовых станций 29](#_Toc454371437)

[2.9 Справочник покрытия БС 31](#_Toc454371438)

[2.10 Справочник соседних БС 34](#_Toc454371439)

[2.11 Справочник сетки 500 на 500 36](#_Toc454371440)

[2.12 Анализ источников входных данных 38](#_Toc454371441)

[2.13 Анализ задачи машинного обучения 40](#_Toc454371442)

[3 Аппаратные и программные требования 41](#_Toc454371443)

[3.1 Программные средства для тестового окружения 42](#_Toc454371444)

[3.2 Программные средства реализации 44](#_Toc454371445)

[4 Реализация 45](#_Toc454371446)

[4.1 Реализация модуля формирования справочника уникальных базовых станций 45](#_Toc454371447)

[4.2 Реализация модуля формирования справочников покрытия, соседних базовых станций и отчета по сетке 500 на 500 49](#_Toc454371448)

[4.3 Реализация результирующего отчета «Монобренд» 54](#_Toc454371449)

[4.4 Реализация модуля по определению пола абонента 59](#_Toc454371450)

[4.5 Автоматизация модулей по формированию отчетов геоаналитики 67](#_Toc454371451)

[4.6 Анализ полученных результатов геоаналитики 67](#_Toc454371452)

[Заключение 72](#_Toc454371453)

[Список использованных источников 73](#_Toc454371454)

[Приложение А. HiveQL-скрипт для выгрузки эталонных данных 77](#_Toc454371455)

[Приложение В. Содержимое oozie-файла coordinator.xml 82](#_Toc454371456)

[Приложение Г. Содержимое исполняемого файла startup\_coordinator.sh 87](#_Toc454371457)

[Приложение Д. Содержимое файла Nizhniy\_NovgorodGrid.prj 90](#_Toc454371458)

[Приложение Е. Содержимое файла Nizhniy\_NovgorodGrid.qpj 91](#_Toc454371459)

[Приложение Ж. Содержимое файла модели gender\_model.model 92](#_Toc454371460)

Введение

За последние несколько лет было создано более 90% объема мировой информации. Данные появляются из различных источников. Не так давно получил распространение термин «большие данные», обозначивший новую прикладную область – поиск способов автоматического быстрого анализа огромных объёмов разнородной информации.

Термин Big Data появился сравнительно недавно. Google Trends показывает начало активного роста употребления словосочетания начиная с 2011 года [1].

В общем смысле «большие данные» (англ. Big Data) – серия подходов, инструментов и методов обработки структурированных и неструктурированных данных огромных объёмов и значительного многообразия для получения воспринимаемых человеком результатов, эффективных в условиях непрерывного прироста, распределения по многочисленным узлам вычислительной сети, сформировавшихся в конце 2000-х годов, альтернативных традиционным системам управления базами данных и решениям класса Business Intelligence [2][3].

Бурный рост данных диктует пересмотр традиционных подходов к хранению и обработки информации. Большое количество информации об абонентах, пользователях услуг сотовой связи, находится в информационных системах сотового оператора и социальных сетях. Новые технологии Big Data позволяют уменьшить стоимость хранения данных, а также проводить обогащение информации о клиентах социально-демографическими показателями, геолокационными данными и т.п. Разработка специализированного программного обеспечения для хранения и обработки данных об абонентах сотовых операторов является актуальным и значимым вопросом проведения исследований в области анализа больших данных. В результате уникальной стоимости бизнеса спрос на навыки в области больших данных растет быстрыми темпами.

Data Mining в отличие от Big Data – это класс программ для анализа данных, который работает только со структурированными данными [4].

Big Data – это стек технологий и архитектурный принцип, который решает задачи ETL (Extract, Transform, Load) неструктурированных данных и в дальнейшем может включать и анализ этих данных. Хотя ничего не мешает Big Data после ETL направить данные уже в структурированном виде в Data Mining [4].

Строго говоря, Big Data – это обработка огромных массивов данных, но, например, работа с базой данных под управлением системы управления базами данных (СУБД) Oracle на 20 Гб или 4 Пб – это ещё не Big Data, это просто высоконагруженная база данных. Ключевое отличие Big Data от обычных высоконагруженных систем состоит в возможности строить гибкие запросы. Реляционная база данных в силу своей архитектуры предназначена для коротких быстрых запросов, идущих однотипным потоком. Если разработчик решит выйти за пределы таких запросов и собрать новый сложный, то базу придётся переписывать – или же она не справится под нагрузкой [5].

Рассмотрим несколько классов задач, решение которых использует технологии Big Data.

Оператор связи может знать местоположение абонентов. Например, известно, что в какой-то зоне часто появляются люди с iPhone 6, возможно они пользуются стандартом связи 4G – значит, нужно построить в этой соте LTE-базовую станцию. В другом месте хватит 2G [6].

На основе загруженных детальных данных о звонках из добилингового трафика в режиме близком к реальному времени возможно определение местонахождения абонента. Затем со стороны маркетингового отдела следует предложение ему подключить какую-либо услугу или предложить услуги компаний-партнеров при условии, что абонент в данную минуту находится рядом с точками продаж партнера. Например, абонент находится в некотором крупном городе, тогда ему приходит сообщение с предложением посетить некоторое заведение и получить скидку. С использованием технологий Big Data решение такой задачи упрощается на несколько порядков.

Предположим, что у конкретного лица имеются два устройства – телефон и планшет. Важно знать, что это один и тот же человек. Такая ситуация называется «мультидевайсом». Если две сим-карты постоянно регистрируются в одних и тех же секторах базовых станций – можно сделать вывод, что они принадлежат одному человеку, и предложить ему специальный тариф, который позволяет оплачивать услуги для разных сим-карт с одного счета. [6].

Данная работа посвящена анализу больших массивов данных на примере нескольких задач для крупной телекоммуникационной компании, поставщика услуг сотовой связи и широкополосного интернета, в рамках проекта Hadoop for BigData.

1. Постановка задачи

В данной работе рассматривается применение технологий Hadoop на для решения задач геоаналитики и машинного обучения.

Одним из пунктов, рассматриваемых в рамках данной работы, является разработка программного продукта, решающего следующую задачу. Необходимо сформировать единовременную выгрузку, определив количество уникальных абонентов, которые находились в течение рабочего дня в каждой ячейке зоны разбиения города Нижний Новгород за период с 14 по 20 сентября 2015 года для оптимального расположения офисов «Монобренда».

Отчет формируется относительно всех зон разбиения (zone id), в дальнейшем именуемых для краткости zid. В поле total записывается число абонентов, удовлетворяющих условию: существует принадлежащая рассматриваемой зоне локация (набор сот), в которой абонент находился некоторое отличное от 0 секунд время в течение рассматриваемого интервала. Если в течение рассматриваемого интервала найдутся несколько зон, для которых это условие выполнено, то абонент приписывается к каждой из них. Таким образом, отчет показывает количество уникальных абонентов, прошедшее через каждую зону в течение рассматриваемого периода. Глубина периода анализа – с 8 до 22 часов. Прототип будущего отчета приведен в таблице 1.

Таблица 1 – Прототип результирующего отчета по задаче геоаналитики

|  |  |  |
| --- | --- | --- |
| Измерение | Тип | Описание |
| zid | string | Идентификационный номер зоны разбиения. |
| interval | string | Временной интервал. |
| 14/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 14 сентября 2015 г. |
| Продолжение таблицы 1 | | |
|  |  |  |
| 15/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 15 сентября 2015 года. |
| 16/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 16 сентября 2015 года. |
| 17/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 17 сентября 2015 года. |
| 18/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 18 сентября 2015 года. |
| average\_weekdays | integer | Среднее количество уникальных абонентов в будние дни за 1 календарную неделю. |
| 19/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 19 сентября 2015 года. |
| 20/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 20 сентября 2015 года. |
| average\_weekend | integer | Среднее количество уникальных абонентов в выходные дни за 1 календарную неделю. |
| average\_week | integer | Среднее количество уникальных абонентов за 1 календарную неделю. |

Требованиями к решению поставленной задачи являются:

* расчет справочника по сетке деления г. Нижний Новгород по квадратам размером 500 на 500 метров за сентябрь;
* измерение zid должно соответствовать справочнику по сетке деления г. Нижний Новгород на квадраты 500 на 500 метров и содержать все зоны разбиения;
* показатели total, average\_weekdays, average\_weekend, average\_weekдолжны быть рассчитаны согласно алгоритму и содержать целое неотрицательное число не равное нулю.

Следующим пунктом в рамках постановки задачи является применение метода опорных векторов для решения задачи по определению пола абонента. Задача актуальна тем, что в настоящее время нет достоверных сведений о поле абонента, пользующегося телефонным номером. Для витрины в рамках отдельного проекта Data Eye необходимо сформировать показатель gender (пол) для каждого, кто находится в активной абонентской базе (ААБ) автоматически раз в месяц.

Система должна включать в себя:

* модель определения пола абонента по набору признаков;
* расчет точности определения пола для каждого абонента;
* автоматизацию определения пола ААБ по расписанию – раз в месяц (1 или 5 число каждого месяца);
* результат разметки должен сохраняться как отчет в виде файла с расширением «.csv», который располагается на кластере по адресу /user/tech\_arsspp\_bgd\_ms/spp/gender/merged\_gender.csv;
* предыдущий расчет нужно удалять.

1. Анализ задачи


5. 1. Актуальность выбора технологий

В 1995 году исследовательская компания Gartner предложила hype cycle – кривую зрелости технологии (рисунок 1), графически представляющую стадии, через которые проходит технологическое новшество в ходе своего становления [7]. Как можно видеть, кривая состоит из пяти фаз:

1. Запуск технологии – первая фаза цикла, которая может включать в себя включает технологический прорыв.
2. Пик завышенных ожиданий – общественный ажиотаж приводит к чрезмерному энтузиазму и нереалистичным ожиданиям.
3. Нижняя точка разочарования подразумевает, что технология не в состоянии соответствовать ожиданиям и быстро гасит энтузиазм.
4. Склон просвещения. На этой фазе технология заменяется смежной.
5. Плато производительности. Преимущества технологии становятся очевидными и признаются всеми.



Рисунок 1 – Цикл «зрелости» технологии (Gartner Hype Сycle)

На самом деле данная кривая и ее разные варианты взяты из классической теории управления. График схож с обычной функцией мотивации [8].

По результатам исследований аналитического агентства Gartner термин Big Data находился на пике популярности в 2013 году, о чем свидетельствует рисунок 2 [9].



Рисунок 2 – Кривая «зрелости» технологий по данным за 2013 год (Hype Cycle for Emerging Technologies, 2013)

В 2014 совокупность технологий Big Data уже находилась «склоне разочарований» [10] (рисунок 3).

Наконец, в 2015 года агентство Gartner заменило Big Data, и теперь там появился термин Machine Learning [11] (рисунок 4). Тем не менее, технология продолжает пользоваться популярностью и значимостью.



Рисунок 3 – Кривая «зрелости» технологий по данным за 2014 год (Hype Cycle for Emerging Technologies, 2014)



Рисунок 4 – Кривая «зрелости» технологий по данным за 2015 год (Hype Cycle for Emerging Technologies, 2015)

* 1. Анализ требований и подходов к проекту

У проекта Hadoop for BigData, предназначенного для решения определенного существующего круга задач, имеются три главные цели:

* поиск лучших решений для бизнеса;
* создание новых типов продуктов;
* совместное развитие экспертизы.

Отдельно стоит упомянуть три крупных цикла производства программного продукта, свойственных проекту:

* «пилотная» фаза;
* уточнение параметров;
* автоматизация.

На стадии «пилота» проверяются основные предположения относительно цели продукта, закладываются базовые параметры алгоритмам. В дальнейшем получившиеся данные становятся близки к реальным, но еще не в полной мере отражают желания бизнес-заказчика. На этой фазе допустимы различные эксперименты с данными и появление новых критериев.

В рамках фазы уточнения параметров по выделенным критериям проводятся дополнительные проверки, в ходе чего финализируется алгоритм. Полученные значения на данной ступени близки к совершенным, другими словами к 100% попаданию в цель. Все проверки осуществляются по заранее определенным наборам критериев.

На фазе автоматизации полученный в ходе первых двух стадий алгоритм включается в релиз и автоматизируется для бизнес-целей заказчика. Алгоритм может улучшаться в ходе продолжения исследований алгоритма.

* 1. Анализ существующих решений. Подсистемы Hadoop

Материалов по теории больших данных в специализированных журналах и на сайтах сегодня публикуется довольно много, но из теоретических публикаций далеко не всегда ясно, как можно использовать соответствующие технологии для решения конкретных практических задач.

Одним из самых известных и обсуждаемых проектов в области Big Data является Hadoop, разрабатываемый фондом Apache Software Foundation. Он представляет собой набор из свободно распространяемых утилит и библиотек для разработки и выполнения программ распределенных вычислений. Непосредственно сам проект Apache Hadoop, развиваемый сообществом разработчиков, обеспечивает и дополняет существующие технологии как модель платформы для консолидации, обработки и анализа больших объемов сложных данных. В состав проекта Hadoop входят следующие компоненты:

* Common – набор компонентов и интерфейсов для распределенных файловых систем и общего ввода-вывода;
* MapReduce – модель распределенных вычислений, предназначенная для параллельных вычислений над очень большими (до нескольких петабайт) объемами данных;
* Hadoop Distributed File System (далее HDFS) – распределенная файловая система.

Классическая конфигурация кластера Hadoop состоит из одного сервера имён (Name Node), одного планировщика MapReduce-заданий, так называемого JobTracker, а также набора рабочих машин, на каждой из которых одновременно находятся сервер данных (Data Node) и сервер заданий (Task Tracker). Пример такой конфигурации представлен на диаграмме развертывания, изображенной на рисунке 5.



Рисунок 5 – Диаграмма развертывания компонентов на dev-кластере

Узел NameNode представляет собой программный код, обычно выполняющийся на выделенной машине экземпляра HDFS. Этот узел отвечает за управление пространством имен файловой системы и за управление доступом со стороны внешних клиентов. Узел Name Node определяет соответствие между файлами и реплицированными блоками на узлах Data Node.

Узел Data Node также представляет собой программный код, обычно выполняющийся на выделенной машине экземпляра HDFS. Кластер Hadoop содержит один узел типа Name Node и сотни или тысячи узлов типа Data Node. Узлы Data Node обрабатывают запросы на чтение и запись, поступающие от клиентов файловой системы HDFS. Также они отвечают на команды создания, удаления и реплицирования блоков, полученные от узла NameNode.[13].

Ранее в Hadoop входили другие подпроекты (рисунок 6), которые теперь являются самостоятельными продуктами Apache Software Foundation [14]:

* Avro – система сериализации для выполненных межъязыковых вызовов RPC и долгосрочного хранения данных;
* Pig – язык управления потоком данных и исполнительная среда для анализа больших объемов данных;
* Hive – распределенное хранилище данных; оно управляет данными, хранимыми в HDFS, и предоставляет язык запросов на базе SQL для работы с этими данными;
* HBase – нереляционная распределенная база данных;
* ZooKeeper – распределенный координационный сервис, который предоставляет примитивы для построения распределенных приложений;
* Sqoop – инструмент для пересылки данных между структурированными хранилищами и HDFS;
* Oozie – сервис для записи и планировки заданий Hadoop.



Рисунок 6 – Набор компонентов, входящих в «экосистему» Hadoop

Сегодня Hadoop представляет собой сложную систему, состоящую из большого числа компонентов. Установить и настроить такую систему самостоятельно – весьма непростая задача. Поэтому многие компании сегодня предлагают готовые дистрибутивы Hadoop, включающие инструменты развертывания, администрирования и мониторинга.

Дистрибутивы Hadoop распространяются как под коммерческими (продукты таких компаний, как Intel, IBM, EMC, Oracle), так и под свободными (продукты компаний Cloudera, Hortonworks и MapR) лицензиями.

Cloudera Hadoop представляет собой полностью открытый дистрибутив. Он распространяется как в бесплатном, так и в платном варианте, известном под названием Cloudera Enterprise.

Cloudera Hadoop включает следующие основные компоненты (рисунок 7):

* Cloudera Hadoop (CDH) – собственно дистрибутив Hadoop;
* Cloudera Manager – инструмент для развертывания, мониторинга и управления кластером Hadoop.



Рисунок 7 – Набор компонентов, предоставляемых дистрибутивами от Cloudera

Начиная со второй версии Hadoop, исполняемую среду MapReduce, обеспечивающую управление ресурсами и обработку данных, заменили на MapReduce 2 – YARN (Yet Another Negotiation) – еще один посредник между HDFS и инструментами для работы с данными. Различия между версиями приведены на рисунке 8.



Рисунок 8 – Различия версий Hadoop 1.0 и 2.0

* 1. Сравнение исполняемых сред

MapReduce – это модель распределенной обработки данных, предложенная компанией Google для обработки больших объёмов данных на компьютерных кластерах [15]. MapReduce предполагает, что данные организованы в виде некоторых записей. Обработка данных происходит в 3 стадии:

1. Стадия Map. На этой стадии данные предобрабатываются при помощи функции map(), которую определяет пользователь. Работа этой стадии заключается в предобработке и фильтрации данных. Функция map() примененная к одной входной записи и выдаёт множество пар ключ-значение. Что будет находиться в ключе и в значении – решать пользователю, но ключ – очень важная вещь, так как данные с одним ключом в будущем попадут в один экземпляр функции reduce().
2. Стадия Shuffle. Проходит незаметно для пользователя. В этой стадии вывод функции map() разбирается по «корзинам». Каждая корзина соответствует одному ключу вывода стадии map(). В дальнейшем эти корзины послужат входом для reduce().
3. Стадия Reduce. Каждая «корзина» со значениями, сформированная на стадии shuffle, попадает на вход функции reduce(). Функция reduce() задаётся пользователем и вычисляет финальный результат для отдельной «корзины». Множество всех значений, возвращённых функцией reduce(), является финальным результатом MapReduce-задачи.

Несколько дополнительных фактов про MapReduce.

* Все запуски функции map() работают независимо и могут работать параллельно, в том числе на разных машинах кластера;
* Shuffle внутри себя представляет параллельную сортировку, поэтому также может работать на разных машинах кластера. Пункты 1-3 позволяют выполнить принцип горизонтальной масштабируемости.
* Функция map(), как правило, применяется на той же машине, на которой хранятся данные. Это позволяет снизить передачу данных по сети (принцип локальности данных).
* MapReduce – это всегда полное сканирование данных, никаких индексов нет. Это означает, что MapReduce плохо применим, когда ответ требуется очень быстро.

Обработка данных подразделяется на следующие этапы (последовательность представлена на рисунке 9):

1. Запуск приложения – передача кода приложения на главный (master) и подчиненные узлы (workers).
2. Главный узел назначает конкретные задачи map() или reduce() и распределяет части входных данных по вычислительным узлам (workers).
3. Map-узлы читают назначенные им входные данные и начинают их обработку.
4. Map-узлы локально сохраняют промежуточные результаты (каждый узел сохраняет полученный результат на локальные диски);
5. Reduce-узлы читают промежуточные данные с map-узлов и выполняют reduce-обработку данных;
6. Reduce-узлы сохраняют итоговые результаты в выходные файлы, обычно в HDFS.

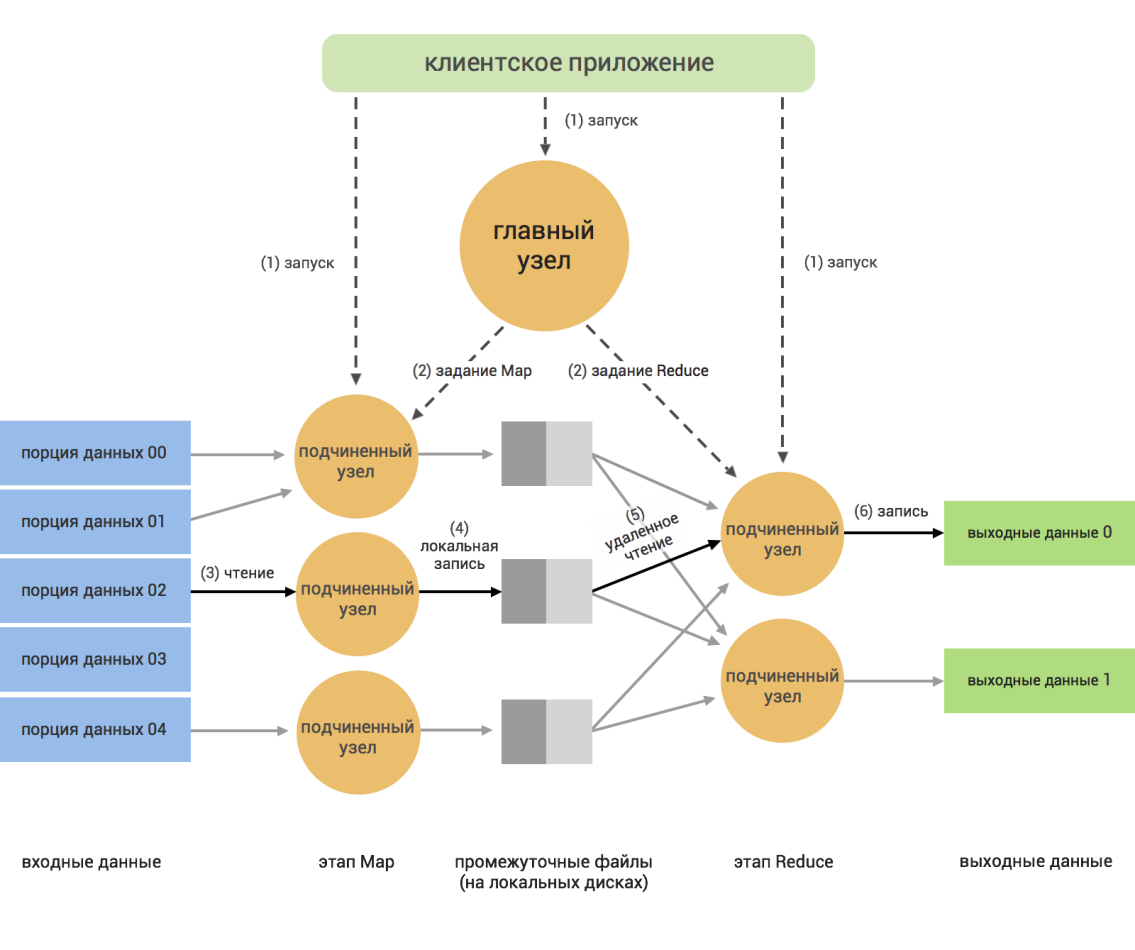


Рисунок 9 – Последовательность выполнения MapReduce-задания

Создание приложений для MapReduce достаточно трудоемко, написание всех функций, компилирование, тестирование и упаковка занимают много времени.

Apache Spark – это платформа параллельной обработки, которая поддерживает обработку в памяти и тем самым значительно повышает производительность аналитических приложений, работающих с большими данными. Spark работает для SQL, потока данных и машинного обучения.

Проект Apache Spark в отличие от MapReduce использует идею локальности данных, однако выносит большинство вычислений в память вместо диска [16]. Ниже приведён скрипт на Python (Spark поддерживает интерфейсы для Scala, Java и Python) для решения задачи фильтрации серверных сообщений из журналов [17]:

Листинг 1. Пример обработки текстовых файлов

# создаём контекст (SparkContext)

sc = ...

# создаём указатель на данные

rdd = sc.textFile("/path/to/server\_logs")

# разбираем строки и переводим их в удобный формат.

# Далее фильтруем записи без ошибок и сохраняем результаты на диск

rdd.map(parse\_line).filter(contains\_error).saveAsTextFile("/path/to/result")

В этом примере реальные вычисления начинаются только на последней строке. Spark видит, что нужно материализовать результаты, и для этого начинает применять операции к данным. При этом здесь нет никаких промежуточных стадий, каждая строчка кладется в память, разбирается, проверяется на признак ошибки в сообщении и, если такой признак есть, тут же записывается на диск. Такая модель оказалась настолько эффективной и удобной, что проекты из экосистемы Hadoop начали один за другим переводить свои вычисления на Spark, а над самим проектом сейчас работает больше людей, чем над устаревающим MapReduce.

Компания Hortonworks решила сделать упор на альтернативный движок Tez. Он представляет задачу в виде направленного ациклического графа – Directed Acyclic Graph (DAG) – компонентов-обработчиков. Планировщик запускает вычисление графа и при необходимости динамически переконфигурирует его, оптимизируя под данные. Это очень естественная модель для выполнения сложных запросов к данным, таких как SQL-подобные скрипты в Hive, где Tez приносит ускорение до 100 раз. Впрочем, кроме как в связке с Hive, Tez пока мало где используется, поэтому сказать, насколько он пригоден для более простых и распространённых задач, довольно сложно.

* 1. Языки запросов к данным

Hive – самая первая и до сих пор одна из самых популярных СУБД на платформе Hadoop. В качестве языка запросов используется HiveQL –диалект SQL, который, тем не менее, позволяет выполнять довольно сложные запросы над данными, хранимыми в HDFS.

Impala – продукт компании Cloudera и основной конкурент Hive. Impala активно использует кеширование часто используемых блоков данных и колоночные форматы хранения, что очень хорошо сказывается на производительности аналитических запросов.

Spark SQL – новая ветвь развития SQL на базе Spark. В Spark SQL нет отдельной консоли и своего хранилища метаданных, SQL-парсер пока довольно слабый, а партиции вовсе не поддерживаются.

* 1. Анализ организации хранения данных в системе

В основе инфраструктуры Hadoop расположена файловая система HDFS. HDFS имеет классическую Unix-подобную древовидную структуру директорий, пользователей с набором прав, а также схожий набор консольных команд [18]:

Листинг 2. Команды Hadoop для работы с файловой системой

# просмотреть корневую директорию: локально и на HDFS

ls /

hadoop fs -ls /

# оценить размер директории

du -sh mydata

hadoop fs -du -s -h mydata

# вывести на экран содержимое всех файлов в директории

cat mydata/\*

hadoop fs -cat mydata/\*

Источники для решения поставленных задач располагаются в HDFS. Они представляют собой сырой массив данных, которые извлекаются, преобразовываются, загружаются и хранятся в одном месте, на промышленном кластере. Это могут быть события сети (вызовы, обрывы, удержания звонков и т.п.), геолокационные данные, данные биллинга и так далее. Схема промышленного кластера с поступающими данными на отдельные его узлы приведена на рисунке 10.



Рисунок 10 – Структура промышленного кластера для продуктивных задач

* 1. Анализ требований к отчетным формам задачи геоаналитики

Для успешного решения поставленной задачи необходимо предварительно провести подробный анализ отчетных форм и определить, какие данные они должны содержать.

Разработку конечного программного продукта по решению задачи геоаналитики можно разделить на несколько взаимосвязанных этапов по формированию отдельных вспомогательных справочников:

* базовых станций;
* справочник покрытия БС;
* соседних базовых станций;
* сетки административного деления 500 на 500.

Входными источниками должны являться:

* справочник с координатами полигонов для Нижнего Новгорода;
* версионный справочник по секторам свечения базовых станций – DIM\_CRAMER\_OSS\_SECTORS – по регионам;
* версионный справочник позиций базовых станций – DIM\_BS\_POSITION – по регионам;
* справочник с сетками городов. Размер ячейки в сетке – 500 на 500.

Формат результирующих отчетов – текстовый файл в расширением «.сsv». В качестве разделителя принят символ «;». Данный выбор обусловлен дальнейшим использованием файлов редактором электронных таблиц Microsoft Excel для небольших по объему файлов или современным текстовым редактором EmEditor, поддерживающим открытие и редактирование файлов очень больших объемов, которые не удается открыть с помощью Excel.

* 1. Справочник базовых станций

Базовая станция (БС) – комплекс радиопередающей аппаратуры (ретрансляторы, приемопередатчики), осуществляющий связь с конечным абонентским устройством – сотовым телефоном. БС может быть описана при помощи комбинации двух параметров:

1. Location Area Code (далее LAC) – код зоны местоположения группы базовых станций.
2. Cell ID – идентификатор конкретной соты внутри Location Area.

Помимо наличия предварительно рассчитанного справочника БС для города Нижний Новгород необходимо, рассчитать аналогичный справочник для каждого города из списка: Волгоград, Москва, Московская область, Воронеж, Казань, Красноярск, Нижний Новгород, Новосибирск, Омск, Ростов, Санкт-Петербург, Уфа, Самара. Для этого необходимо объединить файлы справочника свечения по секторам (DIM\_CRAMER\_OSS\_SECTORS) со справочником позиций базовых станций (DIM\_BS\_POSITION) по значению атрибута POSITION\_CODE, который представлен для каждого из городов в таблице 2.

Таблица 2 – Соответствие значения поля POSITION\_CODE регионам

| GEOREGION\_NAME | POSITION\_CODE | Наименование латиницей |
| --- | --- | --- |
| Волгоград | 7264 | Volgograd |
| Воронеж | 7171 | Voronezh |
| Казань | 8911 | Kazan |
| Красноярск | 8668,307733, 307735 | Krasnoyarsk |
| Нижний Новгород | 5971 | nizhniy\_novgorod |
| Новосибирск | 5837 | Novosibirsk |
| Омск | 5802 | Omsk |
| Ростов-на-Дону | 5593 | rostov\_na\_donu |
| Санкт-Петербург | 599,194844 | saint\_petersburg |
| Уфа | 11152 | Ufa |
| Самара | 5497 | Samara |
| Москва и МО | 12 | moscow\_mo |

В результате сформируются справочники БС с названием CITY\_DIM\_BASE\_STATIONS.csv, где CITY – название города, по которому произведен расчет из таблицы 2. Описание полей приведено в таблице 3. Разделитель для показателей – символ табуляции.

Таблица 3 – Атрибуты выходных данных справочника БС

|  |  |  |  |
| --- | --- | --- | --- |
| Показатель | Описание измерения | Формат данных | Алгоритм расчета |
| POSITION\_LATITUDE | Географическая широта | Вещественное число | Определяется из справочника позиций базовых станций |
| POSITION\_LONGITUDE | Географическая долгота |
| CELL | ID соты (номер вышки). | Целое неотрицательно число | Определяется из справочника свечения по секторам |
| AZIMUTH | Направление свечения вышки | Вещественное число |
| SECTOR\_ANGLE | Угол свечения вышки |
| RANGE | Cтандарт связи. Диапазон (900, 1800, 2100 МГц) | Целое неотрицательное число |

* 1. Справочник покрытия БС

Определяем радиус свечения для БС из справочника CITY\_DIM\_BASE\_STATIONS.csv:

* если стандарт связи составляет 900 МГц, то радиус равен 1 км;
* если стандарт связи – 1800 МГц, то радиус – 0,75 км. Если стандарт связи – 2100, то радиус – 0,25 км;
* если стандарт связи принимает пустое значение или -99, то радиус равен 1 км.

На следующем этапе необходимо определить площадь свечения БС. Наносим координаты каждой БС на карту. Координаты БС (значения полей POSITION\_LATITUDE*,* POSITION\_LONGITUDE) берутся из справочника CITY\_DIM\_BASE\_STATIONS.csv.

От получившейся точки откладываем прямую по направлению свечения БС (значение поля AZIMUTH из справочника CITY\_DIM\_BASE\_STATIONS.csv). Затем от получившейся прямой откладываем угол, равный углу свечения БС (значение поля SECTOR\_ANGLE из справочника CITY\_DIM\_BASE\_STATIONS.csv). В получившийся угол вписываем окружность с радиусом равным значению, соответствующему стандарту связи.

Если для БС угол свечения строго больше 180 градусов, то около вышки описываем окружность радиусом один километр, затем данную окружность вписываем в треугольник. По получившейся окружности определяем основание треугольника.

Определяем дальность свечения БС. Дальность свечения БС равна высоте треугольника. Если в справочнике по секторам БС – DIM\_CRAMER\_OSS\_SECTORS – отсутствуют азимут и угол свечения, то вместо площади свечения будет использоваться круг.

Получаем справочники секторов с названием CITY\_DIM\_TRIANGLES\_WKT.csv, где CITY – название города (кроме Москвы и МО по условию задачи), по которому необходим расчет (в нашем случае – это Нижний Новгород). Разделитель в справочнике – символ «;».Таблица 4 – Атрибуты выходных данных справочника покрытия БС

|  |  |  |  |
| --- | --- | --- | --- |
| Показатель | Описание показателя | Формат данных | Алгоритм расчета |
| LAC | Номер группы вышек. | Целое неотрицательное число | Определяется из справочника по секторам базовых станций |
| CELL | ID соты (номер вышки). |
| LATITUDE\_LEFT | Географическая широта | Вещественное неотрицательное число | – |
| LONGITUDE\_LEFT | Географическая долгота |
| LATITUDE\_RIGHT | Географическая широта |
| LONGITUDE\_RIGHT | Географическая долгота |
| LATITUDE\_BS | Координата (географическая широта) вершины треугольника, где установлена БС |
| LONGITUDE\_BS | Координата (географическая долгота) вершины треугольника, где установлена БС |
| Продолжение таблицы 4 | | | |
| LATITUDE\_CENTER | Координата (географическая широта) центр треугольника, где установлена БС | Вещественное неотрицательное число | – |
| LONGITUDE\_CENTER | Координата (географическая широта) центр треугольника, где установлена БС |
| AZIMUTH | Направление свечения вышки | Определяется из справочника свечения по секторам |
| ANGLE | Угол свечения вышки |
| RANGE | Диапазон рабочих частот | Целое неотрицательное число |
| HEIGHT | Высота треугольника | Вещественное неотрицательное число | – |
| the\_geom\_EPSG:3576 | Точки треугольника в формате WKT и системе координат EPSG:3576 | – |

* 1. Справочник соседних БС

На сетку из справочника с сетками городов наносим покрытие БС (в виде треугольников), сформированное в результате расчета предыдущих вычислений DIM\_TRIANGLE\_WKT.csv. Если площади треугольников вышек пересекаются, то они являются соседними.

Листинг 3. Вид справочника соседних БС

Lac#cell\_id <разделитель> Lac1#cell\_id1, Lac2#cell\_id2, Lac3#cell\_id3, …

//Lac#cell\_id – вышка, для которой мы определяем соседей; //Lac1#cell\_id1, Lac2#cell\_id2, Lac3#cell\_id3 – соседние БС, //для вышки Lac#cell\_id.

Название справочника – CITY\_DIM\_NEIGHBORS.csv, где CITY – название города, по которому возможен расчет из таблицы 2.

Далее приведена таблица 5 с описанием атрибутов выходных данных справочника соседних БС.

Таблица 5 – Атрибуты выходных данных справочника соседних БС

|  |  |  |  |
| --- | --- | --- | --- |
| Показатель | Описание показателя | Формат данных | Алгоритм расчета |
| LAC | Номер группы вышек. | Вещественное число | Определяется из справочника по секторам базовых станций |
| CELL | ID соты (номер вышки) |
| LAC\_NEIGHBOUR | Номер группы соседних вышек. | Вещественное число | Определяется из справочника с площадью секторов, построенного по алгоритму определения площади свечения вышки на ячейки сетки |
|  | | | |
| Продолжение таблицы 5 | | | |
| CELL\_NEIGHBOUR | ID соседней соты (номер вышки). | Вещественное число | Определяется из справочника, построенного по алгоритму определения площади свечения вышки на ячейки сетки |

* 1. Справочник сетки 500 на 500

Используем заготовленные сетки административного деления по 12 городам в виде файлов CITY\_GRID\_500.shp, где CITY – название города из таблицы 2.

На сетку из справочника города наносим покрытие БС в виде треугольника, получившееся в результате расчета DIM\_TRIANGLE\_WKT.csv. Далее определяем на какие ячейки сетки светит данная БС. На рисунке 11 приведен пример расположения БС и ее свечения на соседние квадраты сетки города.



Рисунок 11 – Расположение БС и засветы на квадраты

Рассчитываем площадь свечения вышки в каждой ячейке сетки (квадраты 2, 3, 4, 6, 7, 8, 10, 11, 12 на рисунке 11) и заносим в справочник DIM\_GRID\_500.csv.

Таблица 6 – Выходной формат данных справочника DIM\_GRID\_500.csv

| Показатель | Описание показателя | Формат данных | Алгоритм расчета |
| --- | --- | --- | --- |
| ID\_SQUARE | Идентификатор квадрата, на которую светит данная вышка | Целое неотрицательно число | Определяется из справочника с координатами полигонов |
| LAC | Номер группы вышек. | Целое неотрицательно число | Определяется из справочника по секторам базовых станций |
| CELL\_ID | ID соты (номер вышки). |
| AREA\_GLARE | Площадь засвета квадрата, получившегося в результате пересечения квадрата и сектора вышки | Вещественное число | – |
| AREA\_COVERAGE | Площадь фигуры полученной в результате пересечения треугольника (покрытия БС) и квадрата (ячейки сетки) |

* 1. Анализ источников входных данных

Наиболее важными при решении поставленных задач являются источники голосового и интернет-трафика – Anritzu и Polystar. Трафик Anritzu cодержит информацию о разобранных событиях сигнальной сети, включая следующие данные: звонки, смс, геолокацию БС. Информация хранится в сжатых файлах в формате Sequence File [19] с целью наилучшего ваианта хранения в HDFS. Источник содержит информацию по транзакциям исключительно в сетях 2G и 3G. Периодичность и время появления данных – каждые 15 мин. Средний объем поступающих данных в день – 12 гигабайт (Гб).

Источник с траффиком Polystar схож по содержанию с Anritsu за исключением того, что включает в себя больше событий, больше регионов, а также ошибки сети. Средний объем поступающих данных в день – 55 Гб.

Ниже приведена таблица 7 с указанием номеров полей в каждом из источников.

Таблица 7 – Индексы полей источников Anritzu и Polystar

| Название поля | Номер поля в источнике Anritzu (ADR) | Номер поля в источнике Anritzu (TDR) |
| --- | --- | --- |
| DATE\_TIME | 2 | 2 |
| IMEI | 53 | 70 |
| IMSI | 16 | 68 |
| TMSI | 17 | 71 |
| LAC | 18 | 25 |
| CELL\_ID | 19 | 84 |
| EVENT | 1 | 1 |
| CAUSES | 33 | 32 |
| STANDART | 2G | 3G |
| NUM2 | 06 или 07 | 73 или 74 |
| OPC\_DPC | 4 или 5 | 8 или 9 |
| CAUSES\_CALLS | 37 | 54 |
| LOC\_UPD\_TYPE | 56 | 43 |
| WAIT\_TIME | 24 | 47 |
| CALL\_TYPE | 6 | 51 |
| DISCONNECTION\_CAUSE | 49 | 52 |
| IMEI | 15 | 69 |
| TMSI\_OLD | 54 | 72 |
| CONV\_TIME | 25 | 77 |
| LAC\_OLD | 46 | 82 |
| RANAP\_MESSAGE\_FLAG1 | – | 86 |

* 1. Анализ задачи машинного обучения

Для витрины Data Eye необходимо определить пол абонента. В рамки задачи входит:

* создание алгоритма, определяющего пол;
* постановка расчета показателей в виртуальной очереди на продуктивном кластере (должен осуществляться раз в месяц для активной абонентской базы);
* разработка процесса оценки точности алгоритма (при возможности автоматизация процесса оценки точности).

К задаче уже приступали, наработки приложены. Алгоритм опирался в том числе на динамику СМС в марте-феврале. Была неплохая точность, но невозможно было применить разработанный подход к абонентам, которые появились после марта. Решать задачу вновь предложено с помощью метода опорных векторов.

Метод опорных векторов (SVM) ***–*** набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа.

Основная идея метода ***–*** перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей [20].

Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора. Ближайшие к параллельным гиперплоскостям точки называются опорными векторами.

1. Аппаратные и программные требования

В блоге Cloudera приводится список аппаратных конфигураций для различных вариантов загрузки [23]. В рамках данной работы возможно использование QuickStart VM от Cloudera с SingleNode-конфигурацией. Обязательным условием для разработки и тестирования является наличие всех необходимых справочников и источников, предоставляемых на вход создаваемым программам, анализ которых приведен в главе «Анализ задачи».

К аппаратному оснащению, а именно процессору, оперативной памяти (ОЗУ) и ПЗУ с учетом использования технологии виртуализации предъявляется следующее требования:

* двухъядерный либо четырехъядерный 64-битный процессор на базе Intel с поддержкой аппаратной виртуализации VT (Virtualization Technology) или AMD-V;
* не менее 20 Гб (гигабайт) свободного места на жестком диске;
* минимальная рекомендуемая оперативная память для работы с помощью средств виртуализации в зависимости от версии CDH отражена в таблице 8.

Таблица 8 – Аппаратные требования дистрибутивов CDH

|  |  |
| --- | --- |
| Версия CDH и Cloudera Manager | Требуемое количество оперативной памяти основной ОС для обеспечения виртуализации |
| CDH 5 (по умолчанию) | 4 Гб и более |
| Cloudera Express | 8 Гб и более |
| Cloudera Enterprise (пробный период) | 10 Гб и более |

1. 1. Программные средства для тестового окружения

Для обеспечения стабильной работы используется только 64-разрядная основная операционная система с поддержкой средств виртуализации. Это могут быть:

* Microsoft Windows 7/8/10;
* Mac OS X 10.10 или выше;
* Ubuntu 14.04 и выше;
* CentOS 6 и выше.

В качестве виртуальной машины возможно использование одного из нижеперечисленных программных продуктов:

* VirtualBox – программный продукт виртуализации для операционных систем Microsoft Windows, Linux, Mac OS X и других.
* VMware WorkStation версии 8 (и выше) с предустановленными VMware Player версии 4 (и выше) для Windows и Linux, а также VMware Fusion 4 (и выше) для Mac OS X. Нужно учесть, что VMware Fusion работает лишь под управлением Intel-архитекутры, таким образом ранние компьютеры с Mac OS с процессорами PowerPC не запустят предлагаемые дистрибутивы QuickStart VM [23].
* KVM – программное решение, обеспечивающее виртуализацию в среде Linux на платформе x86, которая поддерживает аппаратную виртуализацию на базе Intel VT (Virtualization Technology) либо AMD SVM (Secure Virtual Machine).

Исходя из вышеперечисленных требований, предъявляемых со стороны средств виртуализации и имеющейся установленной операционной системы Microsoft Windows 7 в рамках данной работы для тестов была использована виртуальная машина от Oracle – VirtualBox, а вместе с ней файл VMDK-образа Cloudera QuickStart VM версии 5.4.2.

Самый простой способ переноса данных в кластер – это копирование файлов с помощью стандартных утилит hadoop. Эта операция осуществляется при помощи следующей команды (листинг 4), выполняемой с любого сервера, входящего в HDFS-кластер.

Листинг 4. Команда put для копирования файлов HDFS

hadoop fs -put file\_for\_hadoop /path/to/put/file/in/HDFS/

Для загрузки большого количества файлов или файлов большого объема предпочтительнее копировать файлы в HDFS с помощью веб-интерфейса файлового менеджера в панели управления Hue. Hue – это веб-приложение, используемое для взаимодействия с кластером. Оно интуитивно понятно и не требует дополнительных пояснений. Пример пользовательского веб-интерфейса приведен на рисунке 12.



Рисунок 12 – Веб-интерфейс Hue для доступа к ресурсам кластера

* 1. Программные средства реализации

С учетом всех факторов, в качестве средств реализации требуемых программных модулей для решения поставленных задач были выбраны следующие программные продукты и технологии:

* Язык программирования Java (JDK 1.7).
* Apache Maven – фреймворк для сборки проектов.
* HiveQL – диалект SQL, который позволяет выполнять запросы над данными, хранимыми в HDFS.
* Apache Oozie – планировщик потоков задач. Помимо объединения отдельных MapReduce задач в единый конвейер и запуска их по расписанию дополнительно может выполнять Hive, Java и консольные действия.
* Dell Toad for Apache Hadoop – инструмент для настройки и получения доступа к кластеру.

1. Реализация

В ходе анализа была изучена предметная область, рассмотрена архитектура будущей системы, были установлены все требования, предъявляемые к разрабатываемым модулям, а также выбраны способы и средства их реализации. Данный раздел описывает основные особенности их непосредственной разработки.

1. 1. Реализация модуля формирования справочника уникальных базовых станций

Разработку модуля условно следует реализуется поэтапно (по количеству stage-пакетов):

* фильтрация данных источника Anritsu (пакет stage1);
* фильтрация данных источника Polystar (пакет stage2);
* соединение справочников DIM\_BS\_POSITION и DIM\_CRAMER\_OSS\_SECTORS (пакет stage3).

Для первого этапа разработаны классы MImsiTmsi, RImsiTmsi, DImsiTmsi. Диаграмма классов представлена на рисунке 13:

Рисунок 13 – Диаграмма классов пакета stage1 модуля unique-bs

Класс-mapper – MImsiTmsi – реализуют абстрактный класс Mapper из библиотеки org.apache.hadoop.mapreduce.Mapper. Класс-reducer – RImsiTmsi – наследует класс Reducer из библиотеки org.apache.hadoop.mapreduce.Reducer.

На втором этапе задействованы классы MPolystar, RPolystar, DPolystar, представленные на рисунке 14.



Рисунок 14 – Диаграмма классов пакета stage2 модуля unique-bs

На третьем этапе созданы классы BsPosition, CramerOssSectors, описывающие спецификацию источников, а также 2 mapper-класса – MDimBsPosition, MDimCramerOssSectors, и класс-reducer – RJoinDim. Разработанные классы приведены в соответствии со спецификацией UML 2.0 на следующем рисунке 15.



Рисунок 15 – Диаграмма классов пакета stage3 модуля unique-bs

* 1. Реализация модуля формирования справочников покрытия, соседних базовых станций и отчета по сетке «500 на 500»

Далее приведены диаграммы основных классов реализуемого модуля на рисунке 16.

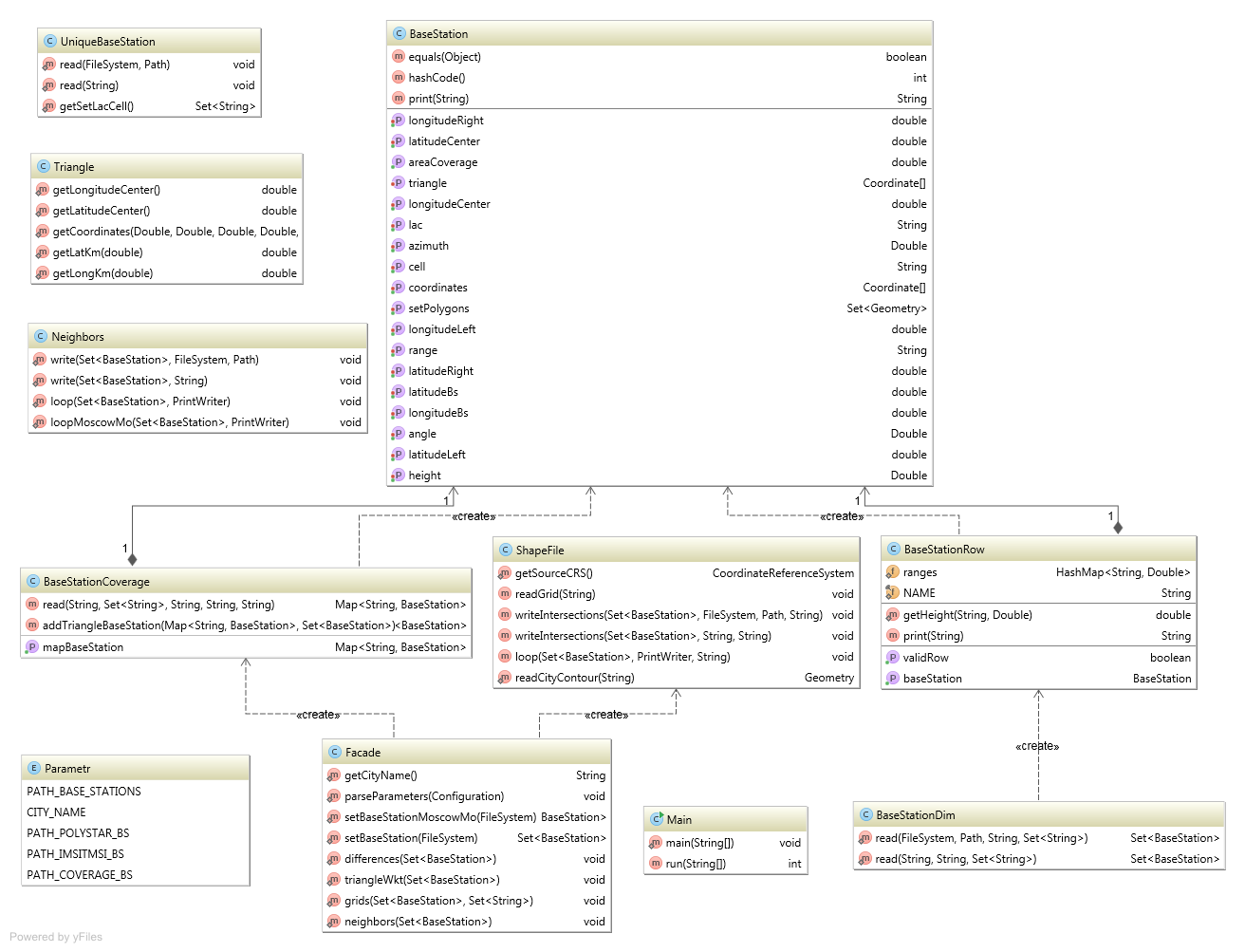


Рисунок 16 – Диаграмма классов модуля формирования справочников покрытия, соседних БС и отчета по сетке «500 на 500»

Далее приведены диаграммы последовательностей в нотации UML 2.0, демонстрирующие порядок формирования каждого из вспомогательных справочников.

На рисунке 17 изображена общая диаграмма последовательностей, описывающая все взаимосвязанные этапы расчета.



Рисунок 17 – Общая диаграмма последовательности формирования вспомогательных справочников покрытия, соседних БС и отчета по сетке 500 на 500

На рисунке 18 более детально представлен этап чтения справочника уникальных базовых станций, сформированного при помощи программного модуля описанного в предыдущем разделе.

 Рисунок 18 – Диаграмма последовательности формирования справочника БС

На рисунке 19 изображена последовательность формирования справочника покрытия уникальных БС.

 Рисунок 19 – Диаграмма последовательности формирования справочника покрытий уникальных БС

На следующем рисунке 20 изображен следующий этап – получение справочника покрытия соседних БС.

 Рисунок 20 – Диаграмма последовательности формирования справочника покрытий соседних БС

На рисунке 21 изображен заключительный этап формирования вспомогательных справочников – расчет «сетки 500 на 500».



Рисунок 21 – Диаграмма последовательности формирования справочника «сетка 500 на 500»

* 1. Реализация результирующего отчета «Монобренд»

В данном разделе приведены диаграммы классов основных пакетов модуля системы (реализованного на языке Java) на рисунках 22, 23, 24, 25. В приложении А приведен исходный код для выгрузки эталонных данных на языке HiveQL для сверки в дальнейшем результатов отчета, полученных двумя разными способами.



Рисунок 22 – Диаграмма классов пакета polyimsi модуля формирования результирующего отчета «Монобренд»



Рисунок 23 – Диаграмма классов пакета grid модуля формирования результирующего отчета «Монобренд»

Рисунок 24 – Диаграмма классов пакета specs модуля формирования результирующего отчета «Монобренд»



Рисунок 25 – Диаграмма вспомогательных классов пакета specs модуля формирования результирующего отчета «Монобренд»

* 1. Реализация модуля по определению пола абонента

Для того, чтобы реализовать задачу по определению пола абонента для начала требуется построить предиктивную модель.

Диаграмма классов разработанного модуля построения модели с использованием метода SVM представлена на рисунке 26.



Рисунок 26 – Диаграмма классов модуля gender-model

Агрегат реализуется MapReduce-задачей на Java. Задача запускается по расписанию через oozie в соответствии с функциональными требованиями (1-5 числа месяца).

Помимо стандартных параметров MapRreduce-задач допустимы следующие, которые приведены в таблице 9.

Таблица 9 – Описание параметров для задачи определения пола

| Имя | Значение |
| --- | --- |
| MONTH | расчетный месяц в формате YYYYMM (по умолчанию –предыдущий месяц) в соотв. С ФТ т.к. отчет ежемесячный. |
| STAGE={1|2} | 1 – агрегация признаков, в таблицах которых нет сост авного ключа (DIM\_BAN, AGG\_SUBS\_GENDER) – join c ААБ т.е. с FCT\_RTC\_MONTHLY, 2 – получение финального набора признаков и применение модели. Другие значения параметра приводят к немедленному завершению задачи. |
| MODEL\_PATH | Файл модели |
| NORM\_COEF\_PATH | Файл нормировочных коэффициентов |
| MODE={prod|test} | prod: Выдается отчет по абонентам.  test: Вместо отчета по абонентам выдается одно число - процент точности предсказания в методе cleanup редьюсера по данным из PDE (для чего в конфигурации должно быть указано, что запускается 1 редьюсер). Можно выдавать доп. столбцы – число абонентов c PDE info, число правильных и неправильных ответов модели, precision, recall.  По умолчанию prod. |
| OUTPUT\_PATH | Целевой каталог. В нем создаются подкаталоги, именованные по MONTH. В них – подкаталоги STG1 и STG2 для стадий. |
| AGG\_SUBS\_GENDER  DIM\_BAN  FCT\_PDE\_SUBS\_ATTR\_DAILY  FCT\_RTC\_MONTHLY  FCT\_USAGE\_POSTPAID  FCT\_USAGE\_PREP\_CHA  FCT\_USAGE\_PREP\_PSA  FCT\_USAGE\_PREP\_OGPRS\_N | Пути до Hive таблиц. |

В таблице 10 приведено описание названия необходимых входных источников.

Таблица 10 – Описание источников для задачи определения пола

|  |  |
| --- | --- |
| Источник | Путь до источника |
| FCT\_USAGE\_PREP\_CHA\_N – существующий интерфейсный объект BIIS, содержащий информацию о трафике абонентов Prepaid. | – |
| FCT\_USAGE\_PREP\_PSA\_N – существующий интерфейсный объект BIIS, содержащий информацию об SMS-трафике абонентов Prepaid. | – |
| FCT\_USAGE\_POSTPAID – существующий интерфейсный объект BIIS, содержащий информацию о трафике абонентов Postpaid. | – |
| DIM\_BAN – существующий интерфейсный объект BIIS, содержащий информацию об абонентах. | – |
| AGG\_SUBS\_GENDER – существующий агрегат, содержащий информацию о предполагаемом поле абонентов, на основе смс-сообщений | /user/tech\_arsspp\_bgd\_ms/AGG\_SUBS\_GENDER |
| FCT\_PDE\_SUBS\_ATTR\_DAILY – существующий объект BIIS, содержащий информацию об абонентах. | Источник будет подключен в HDFS |
| FCT\_RTC\_MONTHLY – ежемесячный агрегат по активной абонентской базе (ААБ) | – |

Для именования приняты следующие требования. Префикс для названия модуля - package: ru.ts.beeline.bigdata.gender. Подкаталоги:

* value - значения VKey, VGender.
* stage1 – алгоритмы первой стадии.
* value – значения V1\*.
* stage2 – алгоритмы второй стадии.
* system – разбор параметров, main().
* utils – вспомогательный код.

Тип содержимого файлов с классами определяется первой буквой:

* (D)river – реализация метода run() для запуска стадии.
* (J)ob – конфигурирует один map/reduce job.
* (M)ap – mapper.
* (C)ombine – combiner.
* (P)artition – partitioner.
* (R)educe – reducer.
* (V)alue – класс для входного или выходного ключа или значения – использовать один универсальный класс VGender.

Также в названии класса описывается номер стадии, имя операции, подзадача (если их несколько), название параметра входного источника (Input sources) для мэпперов.

Пример: M1BiisGenderFctRtcMonthly.java:

* Mapper для stage1.
* Вычисление признака BIIS\_GENDER.
* Input source: FctRtcMonthly.

Unit-тесты реализуются в соответствии с Java соглашениями в классе с тем же именем и суффиксом Test: M1BiisGenderFctRtcMonthlyTest.java.

Листинг 5. VKey.java: ключ абонента

public class VKey implements WritableComparable<VKey> {

public long ban;

public long ctn;

//…

}

Сортировка ключей сначала сравнивает subsKey, потом banKey.

Листинг 6. VGender.java: показатели из AGG\_SUBS\_GENDER

public class VGender implements Writable {

// данные из AGG\_SUBS\_GENDER

public boolean aggSubsGender;

public boolean m1;

public boolean f1;

public boolean m2;

public boolean f2;

public boolean m3;

public boolean f3;

// данные из DIM\_BAN

public int dimBanGender;

// данные из FCT\_PDE\_SUBS\_ATTR\_DAILY

public int pdeGender;

// данные из FCT\_USAGE\_\* таблиц

public int feb23InV;

public int feb23OutV;

public int mar8InV;

public int mar8OutV;

public int marSatInV;

public int marSatOutV;

public int febSatInV;

public int febSatOutV;

public int feb23InS;

public int feb23OutS;

public int mar8InS;

public int mar8OutS;

public int marSatInS;

public int marSatOutS;

public int febSatInS;

public int febSatOutS;

public int gprs;

// признак присутствия абонента в FCT\_RTC\_MONTHLY

boolean aab;

…

}

Неопределенные показатели заполняются для данных из FCT\_USAGE\_\* таблиц – нулями, для FCT\_RTC\_MONTHLY – false, для данных из DIM\_BAN и FCT\_PDF\_SUBS\_ATTR\_DAILY – выделенным значением -1 (означает «неизвестно»). Для AGG\_SUBS\_GENDER aggSubsGender служит признаком наличия данных – если это поле равно false, то поля m1-f3 во внимание не принимаются. Такой формат просто комбинировать. Для компактности формат данных для write/readFields будет использовать вычислимые булевские признаки присутствия групп полей и не писать (не читать) отсутствующие группы.

Листинг 7. JobUtils.java: утилиты для конфигурации map/reduce задач

public class JobUtils {

public static List<ControlledJob> independentJob(Job job)

public static void addControlledJobs(JobControl control, List<Class<?>> jobClasses,final Configuration conf, final ProjectParams projectParams)

public static int waitForFinished(JobControl control, String projectName)

public static void enableCompression(Job job, final Configuration conf, CompressWhat what)

public enum CompressWhat {

Mapper, Reducer, MapperAndReducer

}

}

Задача в пакете stage1 (BIIS\_GENDER) нужна для реализации соединения по ban\_key, так как в DIM\_BAN нет поля subs\_key. Для одного ban\_key может найтись несколько subs\_key и их соединение должно выдать одну и ту же информацию о поле.

Листинг 8. Map Out Key: V1BiisGenderMapKey.java

public class V1AgeDataMapKey implements WritableComparable<V1BiisGenderMapKey> {

public Type type = Type.DimBan;

public final Text banKey = new Text();

public enum Type {

DimBan(DIM\_BAN),

FctRtcMonthly(FCT\_RTC\_MONTHLY);

}

}

Вторичная сортировка ключей дает записи для DIM\_BAN раньше записей для FctRtcMonthly. Map Out Value: имеет класс типа Text, Reduce Out Key – Vkey, Reduce Out Value - VGender.

Задача класса-маппера M1BiisGenderDimBan – пропустить записи с нечисловыми (невалидными) BAN\_KEY. Ключ на выходе работы маппера – значение banKey из поля BAN\_KEY таблицы DIM\_BAN. Выходным значением является содержимое поля CUSTOMER\_GENDER таблицы DIM\_BAN.

Задача маппера экземпляра класса M1BiisGenderFctRtcMonthly пропустить записи с нечисловыми (невалидными) значениями полей BAN\_KEY, SUBS\_KEY. Необходимо рассматривать только записи, в которых:

* RTC\_ACTIVE\_IND больше нуля;
* DW\_STATUS\_KEY принимает значение A или S;
* значение даты в поле SUBS\_ACTIVATION\_DATE\_KEY не должна быть неопределенным, то есть принимать значение по умолчанию – “1970-01-01 00:00:00.0”

В качестве параметра на входе принимается таблица по пути из параметра FCT\_RTC\_MONTHLY для всех подходящих дат (каждый месяц это partition в hive – каталог с файлами, в схеме это поле time\_key, формат пути к каталогу – «PYYYYMM», например P201511).

Задача partition-классаP1BiisGender – хэшировать поле banKey, чтобы для одинаковых banKey данные попадали только в один и тот же reducer.

Задача reducer-класса R1BiisGender заключается в следующем:

* если получено значение для ключа с type равное DimBan – запомнить banKey и value;
* если получено значение для ключа с type FctRtcMonthly – сравнить banKey с записанным, если они совпадают, то записать в output.

Выходным значением ключа является совокупность banKey из ключа и subsKey из значения при условии, что type принимает значение FctRtcMonthly.

Значение на выходе необходимо преобразовать в тип VGender с заполненными полями aab, равное true и dimBanGender равное 0, если пол мужской (М – выдать для русской и английской M); 1 – если пол женский (Ж – выдать для F и W); в противном случае показатель принимает значение -1 (минус единица).

В дальнейшем предполагается увеличить точность модели. Для это

требуется подключить к автоматизации определения пола абонентов источники:

* SMSProof – рассчитанные признаки (вектор длиной 6 из 0 и 1) есть в spp (даст точность до 89% на подмножестве приблизительно 7 миллиона ААБ);
* PDE – подключить источник в HDFS, использовать данные по полу для разметки (100% точность на подмножестве, размеченном PDE около 1 миллиона).

Диаграмма основных разработанных классов, реализующих логику подготовки признаков изображена на рисунке 27. Диаграмма вспомогательных классов пакета feautures представлена на рисунке 28.



Рисунок 27 – Диаграмма классов пакета prepare модуля gender-prediction



Рисунок 28 – Диаграмма классов пакета features модуля gender-prediction

Построенная предиктивная модель приведена в приложении Ж.

* 1. Автоматизация модулей по формированию отчетов геоаналитики

Скрипты автоматизации задачи с использованием системы планирования Oozie приведены в приложениях Б, В, Г.

* 1. Анализ полученных результатов геоаналитики

Для последующего нанесения полученных данных из выгрузки потребовалось программное обеспечение QGIS Desktop 2.8.4. [33]. Содержимое требуемых файлов для создания проекта в QGIS приведены в приложениях Д и Е.

На основе имеющихся shp-файлов строится карта города для визуального представления. Затем результирующий отчет привязывается к топологии по имеющимся координатам из полей вспомогательного справочника по сетке 500 на 500 метров. Таким образом, мы получаем тепловую карту с указанием распределения полученных количественных показателей. Параметры цветовой гаммы настраиваются внутри программы QGIS по процентилям [33].

На представленных ниже рисунках 29, 30, 31, 32 отражена ситуация за 14 сентября 2015 года, а именно количество активных абонентов в каждые из четырех отрезков времени. Для обозначения точек продаж разных мобильных операторов, включая компании конкурентов, приняты следующие условные обозначения:

* «Билайн» – красные треугольники;
* «Мегафон» – желтые квадраты;
* «МТС» – белые пятиугольники;
* «Теле2» – зеленые ромбы.



Рисунок 29 – Количество активных абонентов в городе Нижний Новгород в промежуток с 0 до 6 часов до полудня



Рисунок 30 – Количество активных абонентов в городе Нижний Новгород в промежуток с 6 до 9 часов до полудня



Рисунок 31 – Количество активных абонентов в городе Нижний Новгород в промежуток с 9 до 21 часов после полудня



Рисунок 32 – Количество активных абонентов в городе Нижний Новгород в промежуток времени с 21 до 00

В результате анализа полученных сведенийзаказчикомпринималось решение о наиболее оптимальном расположении точек обслуживания клиентов мобильного оператора в тех зонах административного разбиения, где наблюдалось наибольшее количество активных абонентов в определенные промежутки времени.

Заключение

В ходе работы над задачей геоаналитики были разработаны следующие модули:

* модуль формирования справочника уникальных базовых станций;
* модуль формирования справочников покрытия, соседних базовых станций и отчета по сетке 500 на 500;
* модуль формирования результирующего отчета «Монобренд».

Полученные результаты, а именно результирующий отчет и модули формирования вышеперечисленных справочников соответствуют критериям качества и приняты как продуктивные задачи для автоматизированной работы на промышленном кластере.

В ходе работы над задачей машинного обучения были получены следующие результаты:

* создана предиктивная модель определения пола абонента на основе метода опорных векторов (SVM);
* измерена точность обученной модели, показатель которой составляет не менее 70%;
* автоматизировано определение пола абонента на кластере (период обновления данных – 1 месяц);
* осуществлена проверка корректности предсказания с ранее полученными ответами из анкет респондентов с указанными телефонными номерами.

Список использованных источников

1. Динамика популярности поискового запроса // Google Trends : [сайт]. – URL: <http://www.google.ru/trends/explore#q=%2Fm%2F0bs2j8q/>
2. Big Data от А до Я. Часть 1: Принципы работы с большими данными, парадигма MapReduce // Блог компании DCA (Data-Centric Alliance) на Хабрахабр: [сайт]. – URL: <https://habrahabr.ru/company/dca/blog/267361>
3. Ward J. S. Undefined By Data: A Survey of Big Data Definitions / J.S. Ward, A. Barker // Cornell University Library : [сайт]. – URL: http://arxiv.org/abs/1309.5821
4. Big Data vs Data Mining // Хабрахабр: [сайт]. – URL: habrahabr.ru/post/267827
5. Stonebraker M. MapReduce and Parallel DBMSs: Friends or Foes? / M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, A. Rasin // Communications of the ACM : [сайт]. – URL: <http://gsl.azurewebsites.net/Portals/0/Users/dewitt/Papers/Hadoop/cacm2010.pdf>
6. Мифы и легенды про Big Data // Блог компании «ВымпелКом» : [сайт]. – URL: <https://habrahabr.ru/company/beeline/blog/218669/>
7. Некоторые специалисты по big data могут зарабатывать космические суммы // Rusbase : [сайт]. – URL: http://rusbase.com/longread/how-data
8. Цикл зрелости технологии (Hype Сycle) Gartner при внедрении

новой системы на предприятии // Хабрахабр: [сайт]. – URL: <https://habrahabr.ru/post/198506/>

1. Sheveleva N. Waterfall Theory of Motivation and Intelligence N. Sheveleva, P.Korsukov // Theory of Motivation by Nadezhda Sheveleva and Pavel Korsukov : [сайт]. – URL: <http://theoryofmotivation.info/ru/general/function.html/>
2. Gartner's 2013 Hype Cycle for Emerging Technologies Maps Out Evolving Relationship Between Humans and Machines // Gartner. : [сайт]. – URL: <https://www.gartner.com/newsroom/id/2575515>
3. Gartner's 2014 Hype Cycle for Emerging Technologies Maps the Journey to Digital Business // Gartner. : [сайт]. – URL: <http://www.gartner.com/newsroom/id/2819918>
4. Gartner's 2015 Hype Cycle for Emerging Technologies Identifies the Computing Innovations That Organizations Should Monitor // Gartner. : [сайт]. – URL: <http://www.gartner.com/newsroom/id/3114217>
5. M. Tim Jones. Distributed data processing with Hadoop, Part 1: Getting started Install and configure a simple cluster // IBM developerWorks: [сайт]. – URL: <http://www.ibm.com/developerworks/ru/library/l-hadoop/>
6. Что такое Hadoop в облаке? Знакомство с компонентами Hadoop в HDInsight для анализа больших данных. // Microsoft Azure : [сайт]. – URL: <https://azure.microsoft.com/ru-ru/documentation/articles/hdinsight-hadoop-introduction/>
7. Dean J. MapReduce: Simplified Data Processing on Large Clusters J. Dean and S. Ghemawat // Google, Inc : [сайт]. – URL: http://static.googleusercontent.com/media/research.google.com/ru//archive/mapreduce-osdi04.pdf (дата обращения: 29.05.2016)
8. Hadoop MapReduce vs. Apache Spark Who Wins the Battle? // DeZyre : [сайт]. – URL: <https://www.dezyre.com/article/hadoop-mapreduce-vs-apache-spark-who-wins-the-battle/83>
9. Hadoop: что, где и зачем // Хабрахабр : [сайт]. – URL: <https://habrahabr.ru/post/240405/>
10. Apache Hadoop. Module 2: The Hadoop Distributed File System // Yahoo! Developer Network : [сайт]. – URL: <https://developer.yahoo.com/hadoop/tutorial/module2.html>
11. Apache Hadoop. Module 4: MapReduce// Yahoo! Developer Network : [сайт]. – URL: <https://developer.yahoo.com/hadoop/tutorial/module4.html>
12. Andrew Ng. CS229 Lecture notes Part V Support Vector Machines : [сайт]. – URL: <http://cs229.stanford.edu/notes/cs229-notes3.pdf>
13. Hadoop, часть 1: развертывание кластера // Блог компании Селектел : [сайт]. – URL: <https://blog.selectel.ru/hadoop-chast-1-razvertyvanie-klastera/>
14. Cloudera QuickStart VM // Cloudera : [сайт]. – URL: http://www.cloudera.com/documentation/enterprise/latest/topics/cloudera\_quickstart\_vm.html
15. Apache Hadoop. Module 3: Getting Started With Hadoop // Yahoo! Developer Network: [сайт]. – URL: <https://developer.yahoo.com/hadoop/tutorial/module3.html>
16. White, T. Hadoop: The Definitive Guide, 2nd Edition / T. White. – Sebastopol : O’Reilly Media, 2010. – 628 p.
17. Лэм Ч. Hadoop в действии.: Пер. с англ. / Ч. Лэм. – Москва : ДМК Пресс, 2012. – 424 с.
18. Sammer E. Hadoop Operations, 1st Edition / E. Sammer. – Sebastopol : O’Reilly Media, 2012. – 283 p.
19. Miner D. MapReduce Design Patterns, 1st Edition / D. Miner, A. Shook. - Sebastopol : O’Reilly Media, 2012. – 233 p.
20. Capriolo E. Programming Hive, 1st Edition / E. Capriolo, D. Wampler, J. Rutherglen. - Sebastopol : O’Reilly Media, 2012. – 352 p.
21. Owens J.R. Hadoop Real-World Solutions Cookbook / J.R. Owens, B. Femiano, J. Lentz. – Birmingham : Packt Publishing, 2013. – 316 p.
22. Perera S. Hadoop MapReduce CookBook / S. Perera, T. Gunaranthne. – Birmingham : Packt Publishing, 2013. – 300 p.
23. Lublinsky B. Professional Hadoop Solutions / B. Lublinsky, K. T. Smith, A. Yakubovich. Indianapolis : John Wiley & Sons, Inc., 2013. – 506 p.
24. Venner J. Pro Hadoop / J. Venner. – Berkley : Apress, 2009. – 442 p.
25. Документация QGIS 2.0 [Электронный ресурс] : [сайт].– URL: : <http://docs.qgis.org/2.0/ru/docs/index.html>
26. How-To: Schedule Recurring Hadoop Jobs with Apache Oozie // Cloudera Engineering Blog : [сайт]. – URL: <http://blog.cloudera.com/blog/2013/01/how-to-schedule-recurring-hadoop-jobs-with-apache-oozie/>

Приложение А. HiveQL-скрипт для выгрузки эталонных данных

-- важно: если этого не сделать, будут ошибки при соединении с полями-массивами

set hive.mapjoin.optimized.hashtable=false;

-- загружаем справочник сеток

create table dim\_grid\_500 (

ZID int,

LAC int,

CELL\_ID int,

AREA\_GLARE float,

AREA\_COVERAGE float

) row format delimited fields terminated by '\;';

-- сворачиваем справочник до БС (массив zid)

create table dim\_grid\_500\_grp stored as orc as

select lac, cell\_id, collect\_set(zid) zids

from dim\_grid\_500 t

group by lac, cell\_id;

-- приемник xdr - соединяем все события из ADR, TDR и Polystar

create table nn\_xdr\_20150914\_20 stored as orc as

select imsi, to\_date(datetime) day\_dt, case

when hour(datetime) >= 0 and hour(datetime) <= 5 then 1

when hour(datetime) >= 6 and hour(datetime) <= 8 then 2

when hour(datetime) >= 9 and hour(datetime) <= 20 then 3

when hour(datetime) >= 21 and hour(datetime) <= 23 then 4

end part, lac, cell\_id from anritsu.anritsu\_adr a

where a.region = 'VLG' and a.city = 'nino' and a.date >= 20150914 and a.date <= 20150920

union all

select imsi, to\_date(datetime) day\_dt, case

when hour(datetime) >= 0 and hour(datetime) <= 5 then 1

when hour(datetime) >= 6 and hour(datetime) <= 8 then 2

when hour(datetime) >= 9 and hour(datetime) <= 20 then 3

when hour(datetime) >= 21 and hour(datetime) <= 23 then 4

end part, lac, cell\_id

from anritsu.anritsu\_tdr t

where t.region = 'VLG' and t.city = 'nino' and t.date >= 20150914 and t.date <= 20150920

union all

select imsi, to\_date(datetime\_val) day\_dt, case

when hour(datetime\_val) >= 0 and hour(datetime\_val) <= 5 then 1

when hour(datetime\_val) >= 6 and hour(datetime\_val) <= 8 then 2

when hour(datetime\_val) >= 9 and hour(datetime\_val) <= 20 then 3

when hour(datetime\_val) >= 21 and hour(datetime\_val) <= 23 then 4

end part, lac, cellid

from polystar.polystar\_data t

where t.region = 'VLG' and t.city = 'nino' and t.date >= 20150914 and t.date <= 20150920;

-- сворачиваем, чтобы убрать дубли

create table nn\_xdr\_20150914\_20\_rnd stored as orc as

select imsi, day\_dt, part, lac, cell\_id

from nn\_xdr\_20150914\_20 t

group by imsi, day\_dt, part, lac, cell\_id;

create table nn\_semifinal (

imsi\_cnt bigint,

day\_dt string,

day\_part int,

zids array<int>,

rnd\_zid int

) stored as orc;

with im as (

select count(distinct imsi) imsi\_cnt, day\_dt, part, lac, cell\_id

from nn\_xdr\_20150914\_20\_rnd

group by day\_dt, part, lac, cell\_id)

insert into table nn\_semifinal

select im.imsi\_cnt, im.day\_dt, im.part, gr.zids, gr.zids[cast ((rand() \* size(zids)) as int)] rnd\_zid

from im left join dim\_grid\_500\_grp gr on (gr.lac = im.lac and gr.cell\_id = im.cell\_id);

-- собираем данные по уникальным сочетаниям imsi - БС - день - интервал + пересечение со справочником

-- сетки. На этом этапе случайным образом выбираем zid для каждой БС.

-- Есть БС которых нет в справочнике - это БС области, в справочнике БС только из города.

create table NN\_XXXX stored as ORC as

with im as (

select distinct imsi, day\_dt, part, lac, cell\_id

from nn\_xdr\_20150914\_20\_rnd )

select im.imsi, im.day\_dt, im.part, gr.zids, gr.zids[cast ((rand() \* size(zids)) as int)] rnd\_zid

from im left join dim\_grid\_500\_grp gr on (gr.lac = im.lac and gr.cell\_id = im.cell\_id);

create table nn\_semifinal\_new (

imsi\_cnt bigint,

day\_dt string,

day\_part int,

rnd\_zid int

) stored as orc;

-- группируем данные для показателей по конкретным zid-ам

insert into table nn\_semifinal\_new

select count(imsi), day\_dt, part, rnd\_zid

from NN\_XXXX

group by day\_dt, part, rnd\_zid;

-- выгружаем данные в нужном нам формате

insert overwrite local directory '/export/home/agchursin/nino/results' row format delimited fields terminated by '\;'

select rnd\_zid,

case day\_part when 1 then '00:00:00-05:59:59' when 2 then '06:00:00-08:59:59' when 3 then '09:00:00-20:59:59' when 4 then '21:00:00-23:59:59' end as day\_interval,

sum(case when day\_dt = '2015-09-14' then imsi\_cnt else 0 end) as day20150914,

sum(case when day\_dt = '2015-09-15' then imsi\_cnt else 0 end) as day20150915,

sum(case when day\_dt = '2015-09-16' then imsi\_cnt else 0 end) as day20150916,

sum(case when day\_dt = '2015-09-17' then imsi\_cnt else 0 end) as day20150917,

sum(case when day\_dt = '2015-09-18' then imsi\_cnt else 0 end) as day20150918,

sum(case when day\_dt = '2015-09-19' then imsi\_cnt else 0 end) as day20150919,

sum(case when day\_dt = '2015-09-20' then imsi\_cnt else 0 end) as day20150920,

sum(case when day\_dt in ('2015-09-14', '2015-09-15', '2015-09-16', '2015-09-17', '2015-09-18') then imsi\_cnt else 0 end)/5 as average\_weekday,

sum(case when day\_dt in ('2015-09-19', '2015-09-20') then imsi\_cnt else 0 end)/2 as average\_weekend,

sum(case when day\_dt in ('2015-09-14', '2015-09-15', '2015-09-16', '2015-09-17', '2015-09-18', '2015-09-19', '2015-09-20') then imsi\_cnt else 0 end)/7 as average\_week

from nn\_semifinal\_new

--where zids is not null -- если надо отрезать БС из области

group by rnd\_zid, day\_part;

-- проверка общего количества по день/интервал чтобы оценить результаты.

select sum(imsi\_cnt), day\_dt, day\_part

from nn\_semifinal t

where zids is not null

group by day\_dt, day\_part

order by day\_dt, day\_part;

Приложение Б. Содержимое файла coordinator.properties

nameNode=hdfs://nameservice1

jobTracker=hd-name004.vimpelcom.ru:8050

oozie.use.system.libpath=true

coord\_application\_path=${nameNode}${workingDir}/coordinator.xml

oozie.coord.application.path=${coord\_application\_path}

wf\_application\_path=hdfs://nameservice1${workingDir}

queueName=${queueName}

startTime=2015-09-03T03:00Z

endTime=2015-10-03T03:00Z

Приложение В. Содержимое oozie-файла coordinator.xml

<coordinator-app name="bdest-194-biis-7965-grid-references" frequency="${coord:months(1)}" start="${startTime}" end="${endTime}" timezone="UTC" xmlns="uri:oozie:coordinator:0.2">

<action>

<workflow>

<app-path>${wf\_application\_path}</app-path>

<configuration>

<property>

<name>jobTracker</name>

<value>${jobTracker}</value>

</property>

<property>

<name>nameNode</name>

<value>${nameNode}</value>

</property>

<property>

<name>queueName</name>

<value>${queueName}</value>

</property>

<property>

<name>projectName</name>

<value>${projectName}</value>

</property>

<property>

<name>numReduceTasks</name>

<value>${numReduceTasks}</value>

</property>

<property>

<name>HCAT\_URI</name>

<value>${HCAT\_URI}</value>

</property>

<property>

<name>KRB\_PRINCIPAL</name>

<value>${KRB\_PRINCIPAL}</value>

</property>

<property>

<name>START\_DATE</name>

<value>${coord:formatTime(coord:dateOffset(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), -2, 'DAY'), 'yyyy-MM-dd')}</value>

</property>

<property>

<name>END\_DATE</name>

<value>${coord:formatTime(coord:dateOffset(coord:nominalTime(), -3, 'DAY'), 'yyyy-MM-dd')}</value>

</property>

<property>

<name>OUTPUT\_PATH</name>

<value>${OUTPUT\_PATH}/${coord:formatTime(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), 'yyyyMM')}</value>

</property>

<property>

<name>IMSI\_TMSI</name>

<value>${IMSI\_TMSI}/${coord:formatTime(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), 'yyyyMM')}</value>

</property>

<property>

<name>POLYSTAR</name>

<value>${POLYSTAR}/${coord:formatTime(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), 'yyyyMM')}</value>

</property>

<property>

<name>COVERAGE\_PATH</name>

<value>${COVERAGE\_PATH}/${coord:formatTime(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), 'yyyyMM')}</value>

</property>

<property>

<name>SHAPE\_FILES\_PATH</name>

<value>${SHAPE\_FILES\_PATH}</value>

</property>

<property>

<name>DIM\_BS\_POSITION</name>

<value>${DIM\_BS\_POSITION}</value>

</property>

<property>

<name>DIM\_CRAMER\_OSS\_SECTORS</name>

<value>${DIM\_CRAMER\_OSS\_SECTORS}</value>

</property>

<property>

<name>SAINT\_PETERSBURG</name>

<value>${SAINT\_PETERSBURG}</value>

</property>

<property>

<name>NIZHNIY\_NOVGOROD</name>

<value>${NIZHNIY\_NOVGOROD}</value>

</property>

<property>

<name>ROSTOV\_NA\_DONU</name>

<value>${ROSTOV\_NA\_DONU}</value>

</property>

<property>

<name>KRASNOYARSK</name>

<value>${KRASNOYARSK}</value>

</property>

<property>

<name>NOVOSIBIRSK</name>

<value>${NOVOSIBIRSK}</value>

</property>

<property>

<name>VOLGOGRAD</name>

<value>${VOLGOGRAD}</value>

</property>

<property>

<name>VORONEZH</name>

<value>${VORONEZH}</value>

</property>

<property>

<name>MOSCOW\_MO</name>

<value>${MOSCOW\_MO}</value>

</property>

<property>

<name>MOSCOW</name>

<value>${MOSCOW}</value>

</property>

<property>

<name>SAMARA</name>

<value>${SAMARA}</value>

</property>

<property>

<name>KAZAN</name>

<value>${KAZAN}</value>

</property>

<property>

<name>OMSK</name>

<value>${OMSK}</value>

</property>

<property>

<name>UFA</name>

<value>${UFA}</value>

</property>

<property>

<name>SAINT\_PETERSBURG\_ID</name>

<value>${SAINT\_PETERSBURG\_ID}</value>

</property>

<property>

<name>NIZHNIY\_NOVGOROD\_ID</name>

<value>${NIZHNIY\_NOVGOROD\_ID}</value>

</property>

<property>

<name>ROSTOV\_NA\_DONU\_ID</name>

<value>${ROSTOV\_NA\_DONU\_ID}</value>

</property>

<property>

<name>KRASNOYARSK\_ID</name>

<value>${KRASNOYARSK\_ID}</value>

</property>

<property>

<name>NOVOSIBIRSK\_ID</name>

<value>${NOVOSIBIRSK\_ID}</value>

</property>

<property>

<name>VOLGOGRAD\_ID</name>

<value>${VOLGOGRAD\_ID}</value>

</property>

<property>

<name>VORONEZH\_ID</name>

<value>${VORONEZH\_ID}</value>

</property>

<property>

<name>MOSCOW\_MO\_ID</name>

<value>${MOSCOW\_MO\_ID}</value>

</property>

<property>

<name>MOSCOW\_ID</name>

<value>${MOSCOW\_ID}</value>

</property>

<property>

<name>SAMARA\_ID</name>

<value>${SAMARA\_ID}</value>

</property>

<property>

<name>KAZAN\_ID</name>

<value>${KAZAN\_ID}</value>

</property>

<property>

<name>OMSK\_ID</name>

<value>${OMSK\_ID}</value>

</property>

<property>

<name>UFA\_ID</name>

<value>${UFA\_ID}</value>

</property>

</configuration>

</workflow>

</action>

</coordinator-app>

Приложение Г. Содержимое исполняемого файла startup\_coordinator.sh

#!/usr/bin/env bash

saint\_petersburg="saint\_petersburg"

nizhniy\_novgorod="nizhniy\_novgorod"

rostov\_na\_donu="rostov\_na\_donu"

krasnoyarsk="krasnoyarsk"

novosibirsk="novosibirsk"

volgograd="volgograd"

voronezh="voronezh"

moscow\_mo="moscow\_mo"

moscow="moscow"

samara="samara"

kazan="kazan"

omsk="omsk"

ufa="ufa"

saint\_petersburg\_id="599,194844"

nizhniy\_novgorod\_id=5971

rostov\_na\_donu\_id=5593

krasnoyarsk\_id="8668,307733,307735"

novosibirsk\_id=5837

volgograd\_id=7264

voronezh\_id=7171

moscow\_mo\_id=12

moscow\_id="601,186351,6512168,186347,367230,186349"

samara\_id=5497

kazan\_id=8911

omsk\_id=5802

ufa\_id=11152

OOZIE\_RR=http://hd-name003.vimpelcom.ru:11000/oozie

queueName="prod"

projectDir=/user/$USER/GRIDS

#Путь к покрытиям БС (Shape Files)

#coverage\_path=/user/$USER/bd-general/

coverage\_path=/user/achursin/adhoc/$USER/bd-general-shp/

dimCramerOssSectors=

/apps/hive/warehouse/biis.db/dim\_cramer\_oss\_sectors

dimBsPosition=/apps/hive/warehouse/biis.db/dim\_bs\_position

imsi\_tmsi\_msk\_path=/user/anritsu/data/\*DR/MSK/moscow/

polystar\_msk\_path=/user/polystar/data/MSK/moscow/\*/

numReduceTasks=100

projectName="bdest-194-biis-7965-grid-references"

projectPath=$HOME/Tasks/${projectName}

workingDir=${projectDir}/work

shapeFileDir=${projectDir}/shp

outputPath=${projectDir}

hadoop fs -rm -R $workingDir

hadoop fs -mkdir -p $workingDir

hadoop fs -mkdir -p $shapeFileDir

hadoop fs -put $HOME/Tasks/$projectName/\*.xml $workingDir

hadoop fs -put $HOME/Tasks/$projectName/\*.properties $workingDir

hadoop fs -put $HOME/Tasks/$projectName/shp/\* $shapeFileDir

hadoop fs -mkdir -p ${workingDir}/lib

hadoop fs -put ${projectPath}/lib/\*.jar ${workingDir}/lib

oozie job -auth KERBEROS \

-config /export/home/$USER/Tasks/$projectName/coordinator.properties \

-D projectName=${projectName} \

-D queueName=${queueName} \

-D numReduceTasks=${numReduceTasks} \

-D workingDir=${workingDir} \

-D KRB\_PRINCIPAL=hive/hd-name004.vimpelcom.ru@BEE.VIMPELCOM.RU \

-D HCAT\_URI=thrift://hd-name004.vimpelcom.ru:9083 \

-D DIM\_BS\_POSITION=${dimBsPosition} \

-D DIM\_CRAMER\_OSS\_SECTORS=${dimCramerOssSectors} \

-D COVERAGE\_PATH=${coverage\_path} \

-D IMSI\_TMSI=${imsi\_tmsi\_msk\_path} \

-D POLYSTAR=${polystar\_msk\_path} \

-D OUTPUT\_PATH=${outputPath} \

-D SHAPE\_FILES\_PATH=${shapeFileDir} \

-D SAINT\_PETERSBURG=${saint\_petersburg} \

-D NIZHNIY\_NOVGOROD=${nizh niy\_novgorod} \

-D ROSTOV\_NA\_DONU=${rostov\_na\_donu} \

-D KRASNOYARSK=${krasnoyarsk} \

-D NOVOSIBIRSK=${novosibirsk} \

-D VOLGOGRAD=${volgograd} \

-D VORONEZH=${voronezh} \

-D MOSCOW\_MO=${moscow\_mo} \

-D MOSCOW=${moscow} \

-D SAMARA=${samara} \

-D KAZAN=${kazan} \

-D OMSK=${omsk} \

-D UFA=${ufa} \

-D SAINT\_PETERSBURG\_ID=${saint\_petersburg\_id} \

-D NIZHNIY\_NOVGOROD\_ID=${nizhniy\_novgorod\_id} \

-D ROSTOV\_NA\_DONU\_ID=${rostov\_na\_donu\_id} \

-D KRASNOYARSK\_ID=${krasnoyarsk\_id} \

-D NOVOSIBIRSK\_ID=${novosibirsk\_id} \

-D VOLGOGRAD\_ID=${volgograd\_id} \

-D VORONEZH\_ID=${voronezh\_id} \

-D MOSCOW\_MO\_ID=${moscow\_mo\_id} \

-D MOSCOW\_ID=${moscow\_id} \

-D SAMARA\_ID=${samara\_id} \

-D KAZAN\_ID=${kazan\_id} \

-D OMSK\_ID=${omsk\_id} \

-D UFA\_ID=${ufa\_id} \

-run \

-oozie $OOZIE\_RR

Приложение Д. Содержимое файла Nizhniy\_NovgorodGrid.prj

PROJCS["WGS\_84\_North\_Pole\_LAEA\_Russia",GEOGCS["GCS\_WGS\_1984",DATUM["D\_WGS\_1984",SPHEROID["WGS\_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Lambert\_Azimuthal\_Equal\_Area"],PARAMETER["latitude\_of\_origin",90],PARAMETER["central\_meridian",90],PARAMETER["false\_easting",0],PARAMETER["false\_northing",0],UNIT["Meter",1]]

Приложение Е. Содержимое файла Nizhniy\_NovgorodGrid.qpj

PROJCS["WGS 84 / North Pole LAEA Russia",GEOGCS["WGS 84",DATUM["WGS\_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]],PROJECTION["Lambert\_Azimuthal\_Equal\_Area"],PARAMETER["latitude\_of\_center",90],PARAMETER["longitude\_of\_center",90],PARAMETER["false\_easting",0],PARAMETER["false\_northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["X",EAST],AXIS["Y",NORTH],AUTHORITY["EPSG","3576"]]

Приложение Ж. Содержимое файла модели gender\_model.model

solver\_type L1R\_LR

nr\_class 2

label 0 1

nr\_feature 7

bias 0.000000000000000

w

-0.09339977157324555

0.1325172774150834

0.001349625080676907

0.01743943478474989

-0.003943468510724177

-0.02552200094724406

0

0

w - это веса, которые мы нашли после обучения