МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)
Физико-технический факультет

Кафедра теоретической физики и компьютерных технологий

К У Р С О В О Й П Р О Е К Т

АНАЛИЗ СРЕДСТВ ДЛЯ РАБОТЫ С БОЛЬШИМИ ДАННЫМИ

Работу выполнил _____ Локтев Евгений Сергеевич
Курс 3
Направление 09.03.02 Информационные системы и технологии
Научный руководитель
канд. биолог. наук, доцент _____ Н. Н. Куликова

Нормоконтролер ст. преподаватель _____ Г. Д. Цой

Краснодар 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Большие данные. Едва ли кто не слышал это словосочетание. Оно словно заклинание, и представители крупных компаний, кажется, вынуждены использовать его в своей речи, дабы бизнес оставался на плаву. Излишне активны, впрочем, маркетологи: вешая ярлык с этим термином везде и всюду, они всё вокруг делают модным и отлично продающимся. Увы, в большинстве случаев термин «большие данные» употребляется неверно. Существует необходимость разобраться в этом понятии, уметь использовать средства для работы с большими данными, чтобы вся та огромная лавина информации, генерируемая человечеством с несоизмеримой скоростью, была под чутким контролем.

Для оценки актуальности данной темы достаточно взглянуть на рисунок 1, где представлен график частоты употребления термина «big data» в запросах в одной из наиболее популярных поисковых систем ко времени.

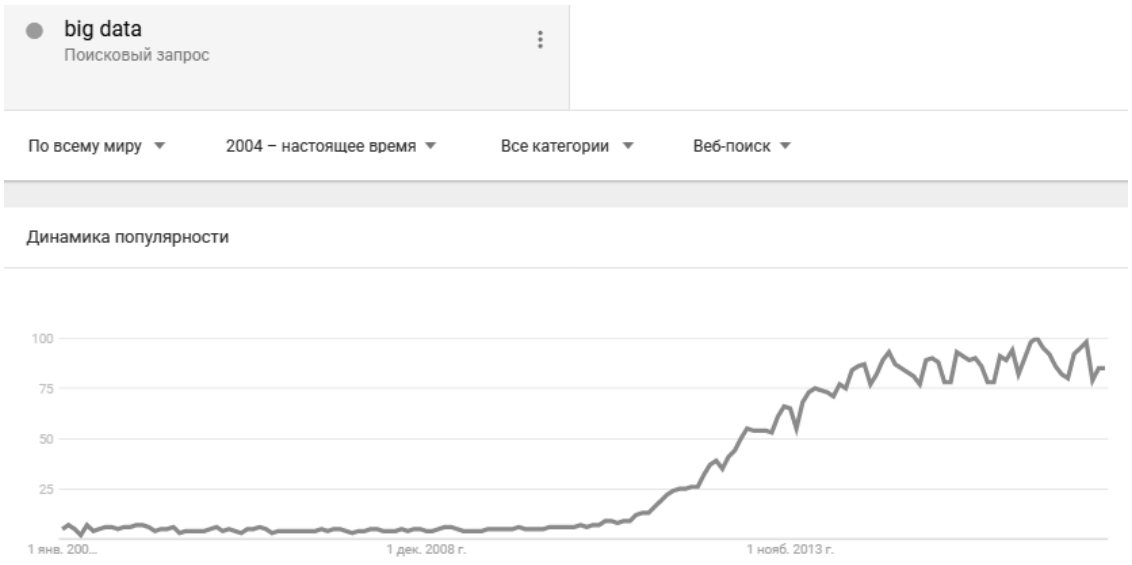

Рисунок 1 – Динамика популярности термина «Big Data» в одной из поисковых систем

Целями данного курсового проекта являются обзор и анализ основных средств, применяемых в настоящее время для работы с большими данными, определение перспектив в данной области и написании небольшой нейронной сети, оценивающей риск возникновения психических заболеваний у работников технических специальностей.

Для достижения поставленной цели необходимо реализовать следующие задачи: разобрать основные положения, связанные с большими данными, изучить парадигму Map Reduce и применить её в написании программы, ознакомиться с инструментарием Apache Hadoop и выяснить его преимущества, недостатки и ограничения, разобрать основные принципы и понятия нейронных сетей, провести анализ существующих средств и сделать выбор в пользу одного из них, провести детальный разбор выбранного метода и создать приложение, основанное на выбранном методе.

1 Общие сведения о Big Data

Первым делом, необходимо определить лексическое значение термина Big Data. Бытуют различные определения. Одни считают большими данными всё, что превышает объем в 1ТБ, иные – данные, которые невозможно обработать на одном компьютере. В рамках данного курсового проекта решено придерживаться следующего определения.

Большие данные - 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 [1].

Таким образом, под Big Data будем понимать не какой-то конкретный объём данных и даже не сами данные, а методы их обработки, которые позволяют распределённо обрабатывать информацию. Эти методы можно применить как к огромным массивам данных (таким как содержание всех страниц в интернете), так и к относительно небольшим (таким как содержимое этого курсового проекта).

Приведем несколько примеров того, что может быть источником данных, для которых необходимы методы работы с большими данными:

-журналы поведения пользователей в интернете;

-GPS-сигналы от автомобилей для транспортной компании;

-данные, снимаемые с датчиков в Большом Адронном Коллайдере;

-оцифрованные книги в Российской Государственной Библиотеке;

-информация о транзакциях всех клиентов банка;

-информация о всех покупках в крупной ритейл сети.

Количество источников данных стремительно растёт, а значит технологии их обработки становятся всё более востребованными.

1.1 Принципы работы с большими данными

Исходя из определения Big Data, можно сформулировать основные принципы работы с такими данными:

Горизонтальная масштабируемость. Поскольку данных может быть сколь угодно много – любая система, которая подразумевает обработку больших данных, должна быть расширяемой. В 2 раза вырос объём данных – в 2 раза увеличили количество «железа» в кластере, и всё продолжило работать.

Отказоустойчивость. Принцип горизонтальной масштабируемости подразумевает, что машин в кластере может быть много. Например, Hadoop-кластер Yahoo имеет более 42000 машин. Это означает, что часть этих машин будет гарантированно выходить из строя. Методы работы с большими данными должны учитывать возможность таких сбоев и переживать их без каких-либо значимых последствий.

Локальность данных. В больших распределённых системах данные распределены по большому количеству машин. Если данные физически находятся на одном сервере, а обрабатываются на другом – расходы на передачу данных могут превысить расходы на саму обработку. Поэтому одним из важнейших принципов проектирования Big Data решений является принцип локальности данных – по возможности обрабатываем данные на той же машине, на которой их храним.

Все современные средства работы с большими данными так или иначе следуют этим трём принципам. Для того чтобы им следовать – необходимо придумывать какие-то методы, способы и парадигмы разработки средств разработки данных.

1.2 MapReduce

Первым делом давайте рассмотрим MapReduce.

MapReduce – это модель распределенной обработки данных, предложенная компанией Google для обработки больших объёмов данных на компьютерных кластерах [2]. MapReduce иллюстрируется рисунком 2.

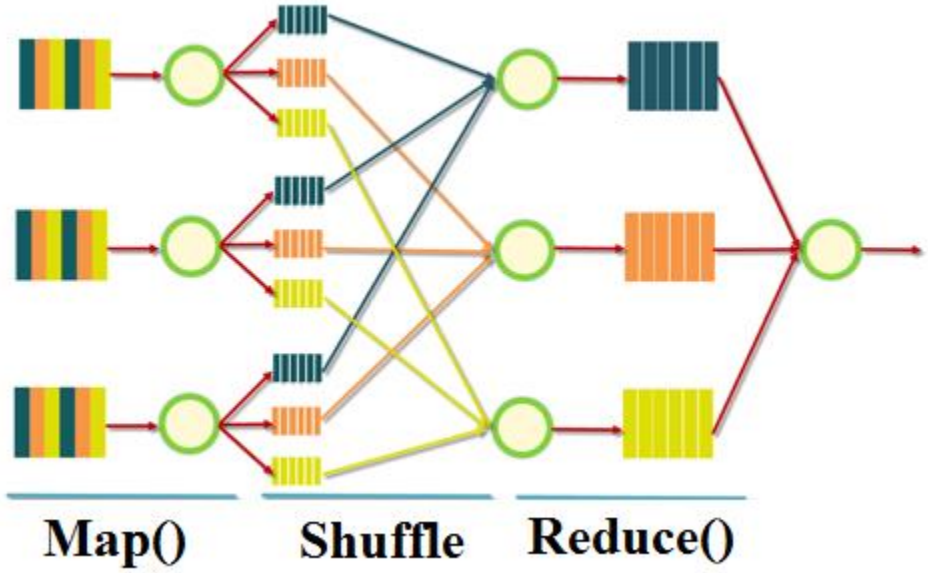

Рисунок 2 – Иллюстрация парадигмы MapReduce

MapReduce предполагает, что данные организованы в виде некоторых записей. Обработка данных происходит в 3 стадии:

Стадия Map. На этой стадии данные предобрабатываются при помощи функции map(), которую определяет пользователь. Работа этой стадии заключается в предобработке и фильтрации данных. Работа очень похожа на операцию map в функциональных языках программирования – пользовательская функция применяется к каждой входной записи. Функция map() примененная к одной входной записи и выдаёт множество пар ключ-значение. Множество – т.е. может выдать только одну запись, может не

выдать ничего, а может выдать несколько 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.

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.

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.

Все запуски функции reduce 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.

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 [3].

В рамках данного курсового проекта была написана простая программа для подсчета слов на языке Python, принцип работы которой основан на MapReduce, её код и результат работы можно видеть на листингах 1 и 2, соответственно.

```
def iter_group(queue):
    buf = []
    prev_key = None
    for val in queue:
        cur_key, cur_val = val
        if cur_key == prev_key or prev_key is None:
            buf.append(cur_val)
        else:
            yield prev_key, buf
            buf = []
            buf.append(cur_val)
        prev_key = cur_key
    if buf:
        yield cur_key, buf
class MapReduce:
    def __init__(self):
        self.queue = []
    def send(self, a,b):
        self.queue.append((a,b))
    def count(self):
        return len(self.queue)
    def __iter__(self):
        return iter_group(sorted(self.queue))
x = MapReduce()
for word in "ура ура Физтех ура КубГУ".split():
    x.send(word, 1)
for word, ones in x:
    print word, sum(ones)
```

Листинг 1 – Код программы для подсчета слов

```
    RESTART: C:\Users\ficus\Google Drive\Курсовой проект
3\LoktevKP3_mapreduce.py
    КубГУ 1
    Физтех 1
    ура 3
    >>>
```

Листинг 2 – Результат выполнения программы для подсчета слов

1.3 Apache Hadoop

Теперь необходимо изучить один из самых популярных инструментариев для работы с Big Data – Apache Hadoop.

Hadoop – это проект с открытым исходным кодом, находящийся под управлением Apache Software Foundation. Hadoop используется для надежных, масштабируемых и распределенных вычислений, но может также применяться и как хранилище файлов общего назначения, способное вместить петабайты данных. Многие компании используют Hadoop в исследовательских и производственных целях [4].

Изначально Hadoop был, в первую очередь, инструментом для хранения данных и запуска MapReduce-задач, сейчас же Hadoop представляет собой большой стек технологий, так или иначе связанных с обработкой больших данных (не только при помощи MapReduce) [5].

Основными (core) компонентами Hadoop являются:

-Hadoop Distributed File System (HDFS) – распределённая файловая система, позволяющая хранить информацию практически неограниченного объёма;

-Hadoop YARN – фреймворк для управления ресурсами кластера и менеджмента задач, в том числе включает фреймворк MapReduce;

-Hadoop common – библиотеки управления файловыми системами.

Также существует большое количество проектов непосредственно связанных с Hadoop, но не входящих в Hadoop core:

-Hive – инструмент для SQL-like запросов над большими данными (превращает SQL-запросы в серию MapReduce–задач);

-Pig – язык программирования для анализа данных на высоком уровне. Одна строчка кода на этом языке может превратиться в последовательность MapReduce-задач;

-Hbase – колоночная база данных, реализующая парадигму BigTable;

-Cassandra – высокопроизводительная распределенная key-value база данных;

-ZooKeeper – сервис для распределённого хранения конфигурации и синхронизации изменений этой конфигурации;

-Mahout – библиотека и движок машинного обучения на больших данных.

Основные концепции Hadoop MapReduce можно сформулировать так:

-обработка/вычисление больших объемов данных;

-масштабируемость;

-автоматическое распараллеливание заданий;

-работа на ненадежном оборудовании;

-автоматическая обработка отказов выполнения заданий.

Работу Hadoop MapReduce можно условно поделить на этапы, они рассмотрены ниже.

Input read: входные данные делятся на блоки данных предопределенного размера (от 16 Мб до 128 Мб) – сплиты (от англ. split). MapReduce Framework закрепляет за каждой функцией Map определенный сплит;

Map. Каждая функция map получает на вход список пар «ключ/значение» <k,v>, обрабатывает их и на выходе получает ноль или более пар <k',v'>, являющихся промежуточным результатом. map(k, v) -> [(k', v')]где k' - в общем случае, произвольный ключ, не совпадающий с k.

Все операции map() выполняются параллельно и не зависят от результатов работы друг друга. Каждая функция map() получает на вход свой уникальный набор данных, не повторяющийся ни для какой другой функции map().

Partition/combine: целью этапа partition (разделение) является распределение промежуточных результатов, полученных на этапе map, по reduce-заданиям. (k', reducers_count) -> reducer_idгде reducers_count - количество узлов, на которых запускается операция свертки; reducer_id - идентификатор целевого узла.

В простейшем случае,reducer_id = hash(k') mod reducers_count. Основная цель этапа partition – это балансировка нагрузки. Некорректно реализованная функция partition может привести к неравномерному распределению данных между reduce-узлами. Функция combine запускается после map-фазы. В ней происходит промежуточная свертка, локальных по отношению к функции map, значений.[(k', v')] -> (k', [v']). Основное значение функции combine – комбинирование промежуточных данных, что в свою очередь ведет, к уменьшению объема передаваемой между узлами информации.

Copy/Compare/Merge. Схематично этап представлен на рисунке 3.

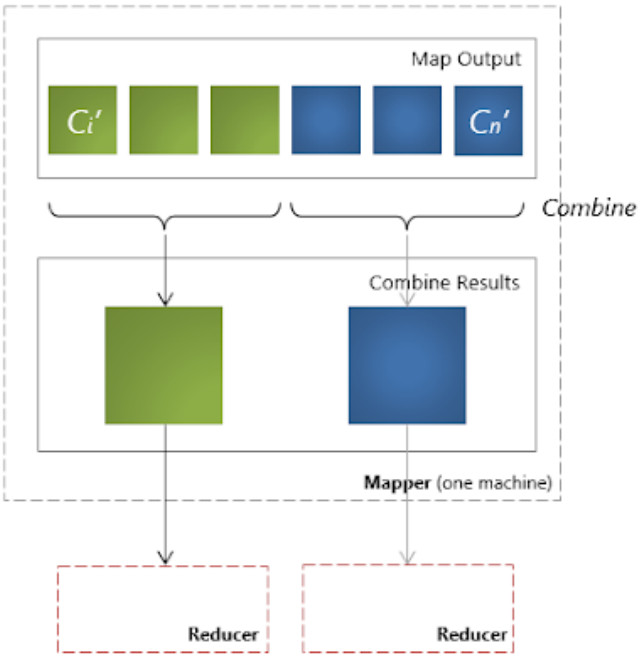

Рисунок 3 – Этап Copy/Compare/Merge

На этом этапе происходит:

-copy: копирование результатов, полученных в результате работы функций map и combine (если такая была определена), с map-узлов на reduce-узлы;

-compare (или Sort): сортировка, группировка по ключу k полученных в результате операции copy промежуточных значений на reduce-узле.compare(k'_n, k'_{n+1}) -> {-1, 0, +1};

-merge: «слияние» данных, полученных от разных узлов, для операции свёртки.

Reduce. Framework вызывает функцию reduce для каждого уникального ключа k' в отсортированном списке значений. reduce(k', [v']) -> [v'']Все операции reduce() выполняются параллельно и не зависят от результатов работы друг друга. Таким образом, результаты работы каждой функции reduce() пишутся в отдельный выходной поток.

Output write. Результаты, полученные на этапе reduce, записываются в выходной поток (в общем случае, файловые блоки в HDFS). Каждый reduce-узел пишет в собственный выходной поток.

Общий алгоритм работы сервиса представлен на рисунке 4.

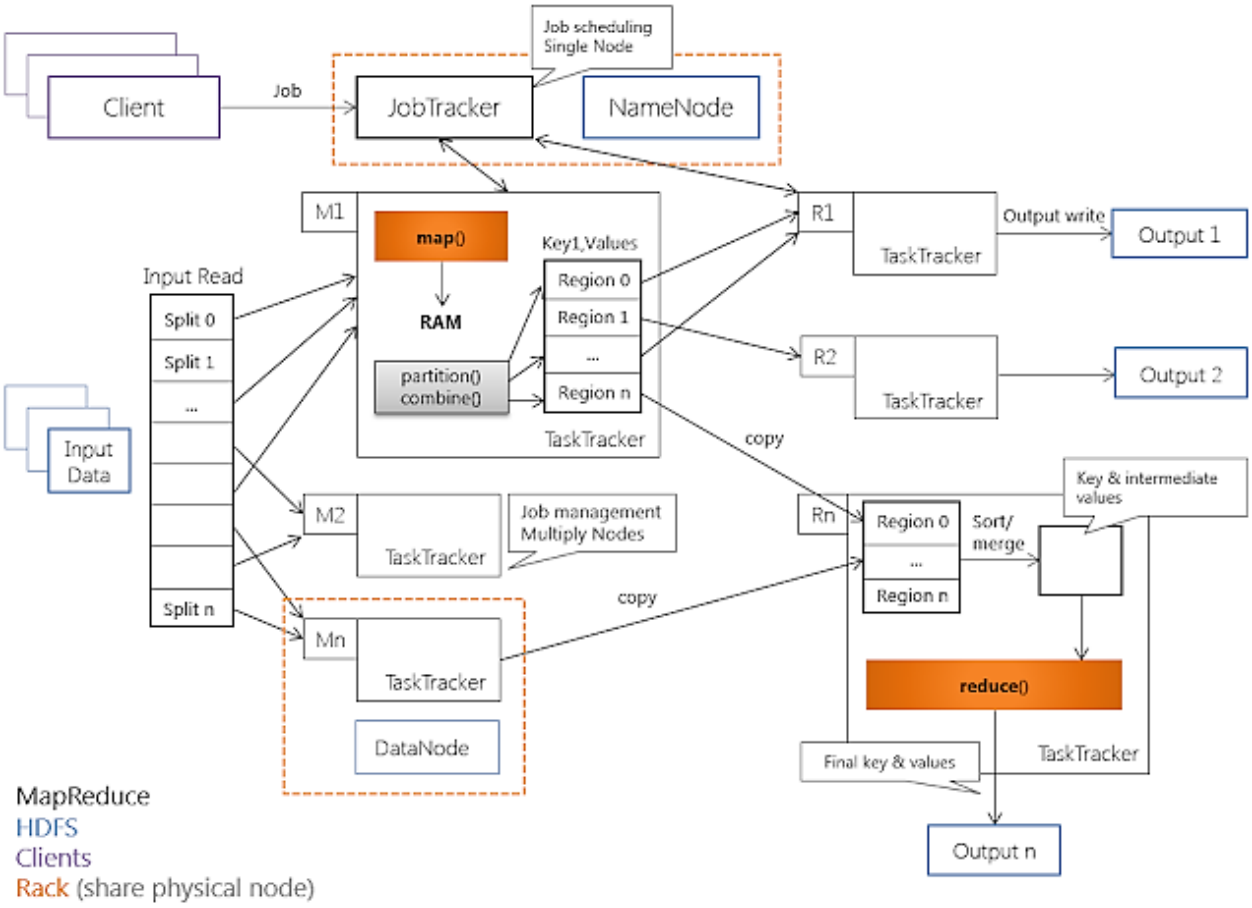

Рисунок 4 – Алгоритм работы сервиса Hadoop MapReduce

Разработчику приложения для Hadoop MapReduce необходимо реализовать базовый обработчик, который на каждом вычислительном узле кластера обеспечит преобразование исходных пар «ключ/значение» в промежуточный набор пар «ключ/значение» (класс, реализующий интерфейс Mapper), и обработчик, сводящий промежуточный набор пар в окончательный, сокращённый набор (класс, реализующий интерфейс Reducer).

Все остальные фазы выполняются программной моделью MapReduce без дополнительного кодирования со стороны разработчика. Кроме того, среда выполнения Hadoop MapReduce выполняет следующие функции:

-планирование заданий;

-распараллеливание заданий;

-перенос заданий к данным;

-синхронизация выполнения заданий;

-перехват «проваленных» заданий;

-обработка отказов выполнения и перезапуск проваленных заданий;

-оптимизация сетевых взаимодействий.

Hadoop MapReduce использует архитектуру «master-worker», где master – единственныйэкземпляр управляющего процесса (JobTracker), как правило, запущенный на отдельной машине (вычислительном узле). Worker-процессы – это произвольное множество процессов TaskTracker, исполняющихся на DataNode.

JobTracker и TaskTracker «лежат» над уровнем хранения HDFS, и запускаются в соответствии со следующими правилами:

-экземпляр JobTracker исполняется на NameNode-узле HDFS;

-экземпляры TaskTracker исполняются на DataNode-узле;

-TaskTracker исполняются в соответствии с принципом «данные близко», т.е. процесс TaskTracker располагается топологически максимально близко с узлом DataNode, данные которого обрабатываются.

Вышеописанные принципы расположения JobTracker- и TaskTracker-процессов позволяют существенно сократить объемы передаваемых по сети данных и сетевые задержки, связанные с передачей этих данных – основные «узкие места» производительности в современных распределенных системах.

JobTracker является единственным узлом, на котором выполняется приложение MapReduce, вызываемое программным клиентом. JobTracker выполняет следующие функции:

-планирование индивидуальных (по отношению к DataNode) заданий map и reduce, промежуточных свёрток;

-координация заданий;

-мониторинг выполнения заданий;

-переназначение завершившихся неудачей заданий другим узлам TaskTracker.

В свою очередь, TaskTracker выполняет следующие функции:

-исполнение map- и reduce-заданий;

-управление исполнением заданий;

-отправка сообщений о статусе задачи и завершении работы узлу JobTracker;

-отправка диагностических heartbeat-сообщений узлу JobTracker.

Взаимодействие TaskTracker-узлов с узлом JobTracker идет посредством RPC-вызовов, причем вызовы идут только от TaskTracker. Аналогичный принцип взаимодействия реализован в HDFS – между узлами DataNode и NameNode-узлом. Такое решение уменьшает зависимость управляющего процесса JobTracker от процессов TaskTracker.

Взаимодействие JobTracker-узла с клиентом (программным) проходит по следующей схеме: JobTracker принимает задание (Job) от клиента и разбивает задание на множество M map-задач и множество R reduce-задач. Узел JobTracker использует информацию о файловых блоках (количество блоков и их месторасположение), расположенную в узле NameNode, находящемуся локально, чтобы решить, сколько подчиненных задач необходимо создать на узлах типа TaskTracker. TaskTracker получает от JobTracker список задач (тасков), загружает код и выполняет его. Периодично TaskTracker отсылает JobTracker статус выполнения задачи.

Взаимодействия TaskTracker-узлов с программным клиентом отсутствуют.

По аналогии с архитектурой HDFS, где NameNode является единичной точкой отказа (Single point of failure), JobTracker также является таковой. Принцип восстановления в узлах JobTracker и TaskTracker описан ниже.

При сбое TaskTracker-узла JobTracker-узел переназначает задания неисправного узла другому узлу TaskTracker. В случае неисправности JobTracker-узла, для продолжения исполнения MapReduce-приложения, необходим перезапуск JobTracker-узла. При перезапуске узел JobTracker читает из специального журнала данные, о последней успешной контрольной точке (checkpoint), вос-

станавливает свое состояние на момент записи checkpoint и продолжает работу с места последней контрольной точки.

При создании архитектуры/разработке программных систем с использованием Hadoop MapReduce следует учитывать следующие аспекты использования MapReduce:

Преимущества:

-эффективная работа с большим (от 100 Гб) объемом данных;

-масштабируемость;

-отказоустойчивость;

-унифицированность подхода;

-предоставление разработчику сравнительно «чистой» абстракции;

-снижение требований к квалификации разработчика, в том числе его знаний и опыта по написанию многопоточного кода;

-дешевизна лицензирования (Open Source).

Ограничения:

-смешение ответственности для Reducer (сортировка и агрегация данных). Таким образом, Reducer – это все, что «не map»;

-отсутствие контроля над потоком данных у разработчика (поток данных управляется фреймворком Hadoop MapReduce автоматически);

-как следствие предыдущего пункта, невозможность простыми средствами организовать взаимодействие между параллельно выполняющимися потоками.

Недостатки:

-применение MapReduce по производительности менее эффективно, чем специализированные решения;

-эффективность применение MapReduce снижается при малом количество машин в кластере (высоки издержки на взаимодействие, а степень распараллеливания невелика);

-невозможно предсказать окончание стадии map;

-этап свертки не начинается до окончания стадии map;

-как следствие предыдущего пункта, задержки в исполнении любого запущенного map-задания ведут к задержке выполнения задачи целиком;

-низкая утилизация ресурсов вследствие жесткого деления ресурсов кластера на map- и reduce-слоты.

-сбой узла JobTracker приводит к простою всего кластера.

1.4 Нейронные сети

Нейронные сети – принципиально новый подход в решении задач, которые решались алгоритмическим программированием. Нет необходимости строить алгоритмы под все возможные случаи развития системы или процессов в системе и стремиться предугадать все варианты и описать логику для них. Нейронные сети позволяют на основе большого числа накопленных данных, самостоятельно найти закономерности и связи в заранее не явных аспектах и использовать эту информацию для дальнейшего прогнозирования, классификации и управления данными и процессами [6].

Примитивно, можно примерно так описать процесс работы нейронных сетей: нейронная сеть на входе получает большой объем информации. Затем данная информация анализируется, нейронная сеть обучается (machine learning) на основе положительных и отрицательных примеров. В процессе обучения формируется структура нейронной сети, которая в дальнейшем может решать задачи идентификации, классификации, прогнозирования [7].

Нейронные сети целесообразно использовать в бизнес-задачах, в таких ситуациях:

-накопленное огромное количество различных данных;

-пока не существует рабочих методов по обработке и систематизации этих данных;

-данные искажены, повреждены, неполны или не систематизированы;

-существует большое разнообразие данных и на первый взгляд сложно установить между ними связи и закономерности.

Варианты и примеры возможного применения нейронных сетей и машинного обучения для бизнес-задач:

-прогнозирование, оценка рисков. (Прогнозирование спроса, объема продаж, среднего чека, частоты продаж, загрузки оборудования для оптимизации количества наличных денежных средств, складских мест и прочих ресурсов);

-поиск трендов, корреляций, тенденций. Прогнозирование дальнейшего развития системы и предсказание возможных изменений;

-распознавание фото, видео, аудио контента. Различные сервисы и онлайн приложения с применением технология распознавания;

-машинное обучение для ведения диалогов компьютерными системами. Для автоматизации деятельности операторов в онлайн-чатах, телефонных операторов и мессенджеров. Разработка чат-ботов.

2 Практическая реализация нейронной сети

Проведя анализ средств для работы с Big Data было решено для работы с оными использовать нейронные сети. В рамках данной главы мы рассмотрим основные python-библиотеки для работы с нейронными сетями, а также напишем собственную небольшую нейросеть, открывающую широкие перспективы для Data Science.

2.1 Библиотеки языка Python для работы с нейронными сетями

PyBrain — одна из популярных Python-библиотек для изучения и реализации большого количества разнообразных алгоритмов, связанных с нейронными сетями. Являет собой удачный пример совмещения компактного синтаксиса Python с хорошей реализацией большого набора различных алгоритмов из области машинного интеллекта [8].

PyBrain представляет собой модульную библиотеку, предназначенную для реализации различных алгоритмов машинного обучения на языке Python. Основной его целью является предоставление исследователю гибких, простых в использовании, но в то же время мощных инструментов для реализации задач из области машинного обучения, тестирования и сравнения эффективности различных алгоритмов.

Название PyBrain является аббревиатурой от английского: Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library.

Библиотека построена по модульному принципу, что позволяет использовать её как студентам для обучения основам, так и исследователям, нуждающимся в реализации более сложных алгоритмов. Общая структура процедуры её использования приведена на рисунке 5:

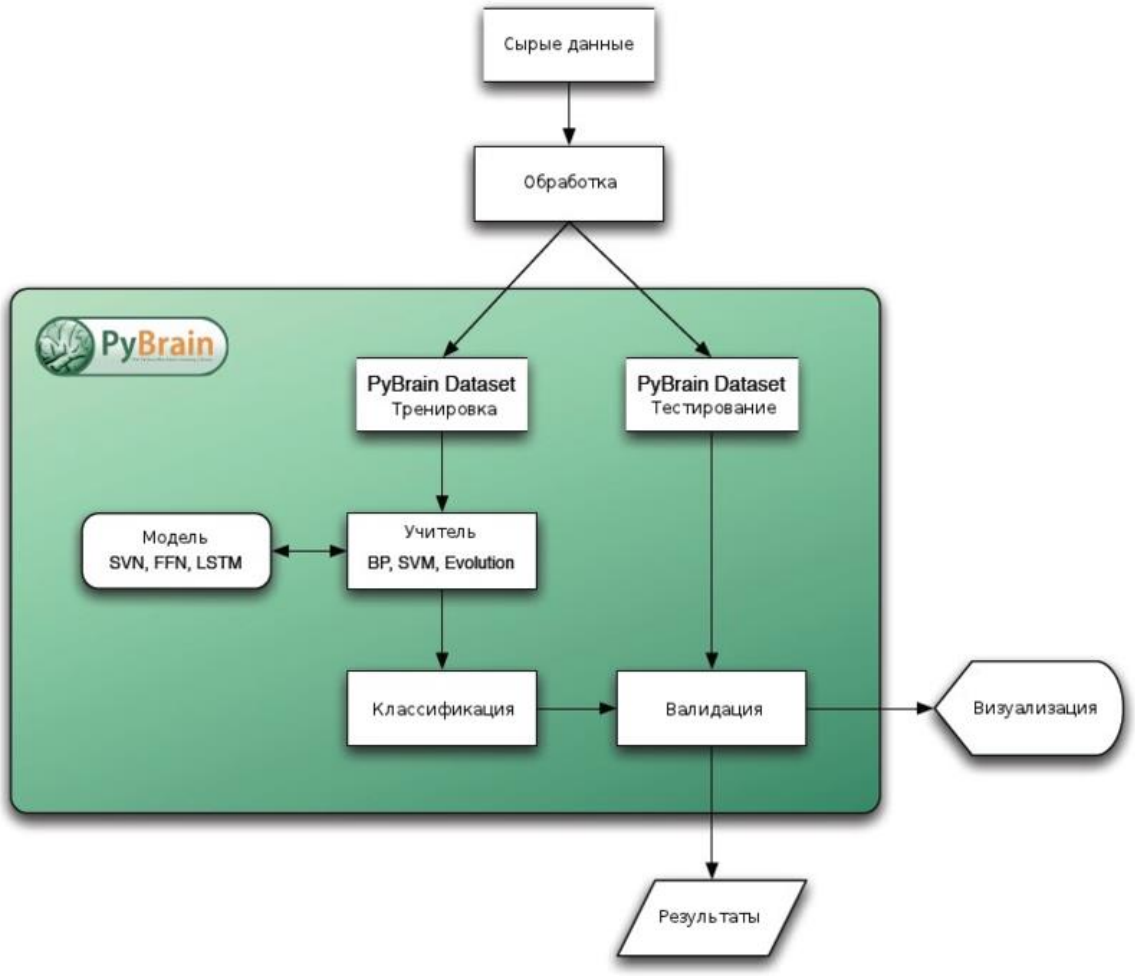

Рисунок 5 – Структура работы нейросети с использованием PyBrain

Сама библиотека является продуктом с открытым исходным кодом и бесплатна для использования в любом проекте.

PyBrain оперирует сетевыми структурами, которые могут быть использованы для построения практически всех поддерживаемых библиотекой сложных алгоритмов. В качестве примера можно привести:

-сети прямого распространения, включая Deep Belief Networks и Restricted Boltzmann Machines (RBM);

-рекуррентные нейронные сети, включая архитектуру Long Short-Term Memory (LSTM)

-multi-Dimensional Recurrent Networks;

-сети Кохонена;

-нейронная сеть Коско;

-создание топологий собственной структуры.

Дополнительно присутствуют программные инструменты, позволяющие реализовывать сопутствующие задачи:

-построение и визуализация графиков;

-поддержка netCDF;

-запись и чтение XML.

Благодаря модульной структуре, PyBrain можно без особого труда настроить для решения многих задач. Данная библиотека рассматривается как наиболее предпочтительная в случае необходимости написания нейросети с использованием языка Python.

Несмотря на очевидные преимущества PyBrain, есть еще множество библиотек, позволяющих комфортно работать с нейронными сетями. Ниже приведены некоторые из них, наиболее функциональные и популярные.

Самая фундаментальная библиотека - NumPy. Она позволяет выполнять основные операции над n-мерными массивами и матрицами: сложение, вычитание, деление, умножение, транспонирование, вычисление определителя и т. д. Благодаря механизму векторизации, NumPy повышает производительность и, соответственно, ускоряет выполнение операций [9].

SciPy – еще одна фундаментальная библиотека. Она содержит модули для линейной алгебры, оптимизации, интеграции и статистики. SciPy работает совместно с NumPy, что позволяет ей значительно расширить функциональность.

Pandas — это пакет, предназначенный для простой и интуитивно понятной работы с «помеченными» и «реляционными» данными. Тоже работает в связке с NumPy, и помимо математических вычислений обеспечивает их агрегацию и визуализацию.

SciKit — инструмент для обработки изображений и имитации искусственного интеллекта. Он основывается на библиотеке SciPy и отвечает за реализацию алгоритмов машинного обучения. SciKit, как и его математическая ос-

нова, демонстрирует высокую производительность, и имеет качественную документацию.

Theano — одна из самых мощных библиотек в данном перечне. Вот несколько причин:

-тесная интеграция с NumPy;

-использование CPU и GPU для повышения производительности;

-встроенные механизмы оптимизации кода;

-расширения для юнит-тестирования и самопроверки.

Theano используется там, где необходимо произвести вычисления с большой точностью максимально быстро.

TensorFlow - Библиотека от Google, была разработана специально для обучения нейронных сетей. Библиотека использует многоуровневую систему узлов для обработки большого количества данных, что расширяет сферу её использования далеко за научную область.

Keras. Данная библиотека использует возможности TensorFlow и Theano в качестве компонентов. Минималистичный подход в дизайне и невероятная расширяемость позволяет быстро начать работу с библиотекой, и не менять её для серьёзного моделирования. Keras также используется в построении и обучении нейронных сетей, а также при решении задачи распознавания устной речи [10].

2.2 Нейронная сеть, определяющая риск возникновения психических заболеваний

В рамках данного курсового проекта была написана нейронная сеть медицинского назначения, определяющая риск возникновения психических заболеваний на основе ответов человека на вопросы небольшой анкеты, у работников технической сферы. Ввиду очень активного внедрения информационных технологий в медицину в последние годы, данная тема была сочтена крайне актуальной.

Для реализации нейронной сети было задействовано ПО Anaconda Navigator 1.7.0. Работа была разделена на следующие этапы:

-загрузка библиотек и данных;

-очистка данных;

-кодирование данных;

-создание ковариационной матрицы;

-построение графиков;

-масштабирование;

-настройка;

-оценка моделей;

-создание предсказаний на основе тестовых данных.

На листинге 3 приведен код, открывающий работу с нейросетью, являющий собой первый этап – загрузку библиотек и данных.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import randint
# подготовка
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.datasets import make_classification
from sklearn.preprocessing import binarize, LabelEncoder,
MinMaxScaler
# модели
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import RandomForestClassifier, Ex-
traTreesClassifier
# библиотеки валидации
from sklearn import metrics
from sklearn.metrics import accuracy_score,
mean_squared_error, precision_recall_curve
from sklearn.model_selection import cross_val_score
#нейронная сеть
from sklearn.neural_network import MLPClassifier
from sklearn.grid_search import RandomizedSearchCV
#дебаггинг
from sklearn.ensemble import BaggingClassifier, AdaBoost-
Classifier
from sklearn.neighbors import KNeighborsClassifier
#наивный байесовский классификатор
from sklearn.naive_bayes import GaussianNB
#стакинг
from mlxtend.classifier import StackingClassifier
#входные данные
from subprocess import check_output
print(check_output(["ls", "input"]).decode("utf8"))
#чтение входных данных
train_df = pd.read_csv('input/survey.csv')
#Pandas: счет данных, распределение и определение типов
print(train_df.shape)
print(train_df.describe())
print(train_df.info())
```

Листинг 3 – Фрагмент кода первого этапа работы с нейронной сетью

На втором этапе можно избавиться от некоторых данных с целью экономии вычислительных ресурсов ПК, а также привести обрабатываемые данные в вид, который будет удобен нейронной сети [11]. Предполагаем, что исключенные данные не окажут существенного влияния на результат работы сети. Фрагменты кода продемонстрированы на листингах 4 и 5.

В качестве тестовых данных были использованы находящиеся в свободном доступе результаты исследования «OSMI Mental Health in Tech Survey» [12].

```
train_df = train_df.drop(['comments'], axis= 1)
train_df = train_df.drop(['region'], axis= 1)
train_df = train_df.drop(['timestamp'], axis= 1)
```

Листинг 4 – Фрагмент кода с очисткой данных

```
#замена пустых полей с возрастом на усредненный возраст
train_df['Age'].fillna(train_df['Age'].median(),  inplace
= True)

#замена медианой значения возраста меньше 18 и больше 120
s = pd.Series(train_df['Age'])
s[s<18] = train_df['Age'].median()
train_df['Age'] = s
s = pd.Series(train_df['Age'])
s[s>120] = train_df['Age'].median()
train_df['Age'] = s
```

Листинг 5 – Фрагмент кода с оптимизацией данных

В результате выполнения следующего этапа создаются массивы вида «label_Gender ['female', 'male']», они лягут в основу дальнейших вычислений. На этапе «создание ковариационной матрицы» с использованием библиотек Matplotlib и Seaborn создаются две матрицы: общая, представлена на рисунке 6, и так называемая «матрица лечения», составленная по аналогичному принципу, но на основе иных полей.

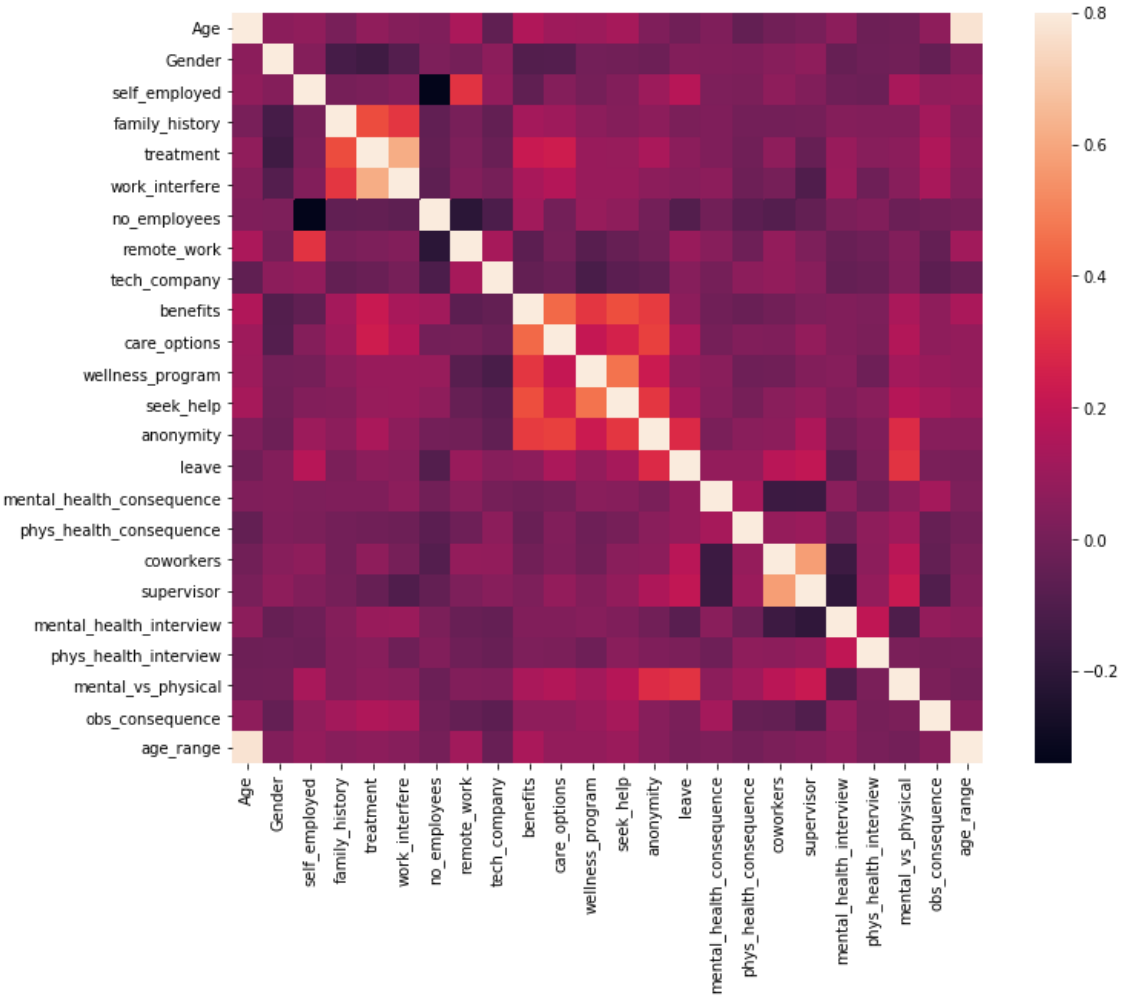

Рисунок 6 – Ковариационная матрица

Далее строятся некоторые графики, делающие данные из тестового набора данных нагляднее. Один из графиков представлен на рисунке 7.

Рисунок 7 – График на основе данных о возрасте

На этапе масштабирования на основе ковариационных матриц и графиков определяется «важность» тех или иных полей, это необходимо для адекватных прогнозов. На основе имеющихся данных было получено распределение, показанное на рисунке 8.

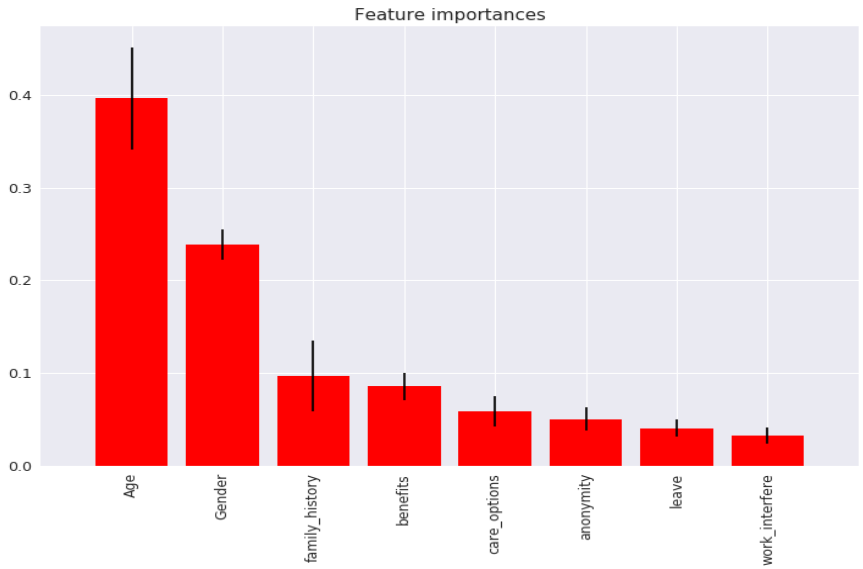

Рисунок 8 – Распределение «важности»

На последних этапах работы в дело вступают модели, выполняющие, непосредственно, вычисления, результаты которых станут основой для итога всей работы сети. Теперь созданная нейронная сеть способна делать предсказания. Для сего был создан отдельный файл с именем «results.csv», и код последнего этапа, а также часть результатов, представлены на рисунке 9.

[45]:
```
# Генерация предсказаний
clf = AdaBoostClassifier()
clf.fit(X, y)
dfTestPredictions = clf.predict(X_test)

# Запись результатов в файл
results = pd.DataFrame({'Индекс': X_test.index, 'Лечение': dfTestPredictions})
results.to_csv('results.csv', index=False)
results.head()
```

	Индекс	Лечение
0	5	1
1	494	0
2	52	0
3	984	0
4	186	0

Рисунок 9 – Последний этап работы с нейросетью

В результате была получена полностью функционирующая нейросеть, способная обрабатывать данные и выдавать предсказания с высокой точностью. Исследования, результаты которых стали тестовыми данными, проводятся регулярно: количество данных для обучения увеличивается, что положительно сказывается на работе нейронной сети.

ЗАКЛЮЧЕНИЕ

В рамках выполнения данного курсового проекта были разобраны основные положения, связанные с большими данными, была изучена и применена в написании программы парадигма MapReduce, а также проведены детальный разбор и анализ средств, ныне используемых для работы с большими данными.

Также была написана и обучена нейросеть медицинского назначения, оценивающая риск возникновения психических заболеваний у работников технических специальностей на основе ответов на вопросы небольшой анкеты.

Все поставленные в ходе проекта цели и задачи были выполнены.

Более не нужны удивительные статистические графики для осознания того, сколь много информации человечество производит ежесекундно – это очевидно каждому. С каждым годом навыки работы с большими данными будут всё ценнее, а сама тема – актуальнее. Однако сейчас мы находимся именно на том переломном моменте, когда инструменты, которыми обходились прежде, перестают быть эффективными. Осваивать новые методы для работы с информацией крайне важно и нужно. В частности, отличным средством являются нейронные сети. Их перспективы поистине будоражат воображение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Марц Н. Большие данные. Принципы и практика построения масштабируемых систем обработки данных в реальном времени / Н. Марц, Д. Уоррен. – М.: Вильямс, 2017. – 368 с.

2 Янишевская А. Г. Способы хранения и обработки большого объема данных с использованием MapReduce и percona server / А. Г. Янишевская, М. А. Чурсин // Киберленинка. – 2015. - №36.

3 Шугуров И. С. Applying MapReduce to conformance / И. С. Шугуров, А. А. Митсюк // Киберленинка. – 2016. - №28.

4 Николаев Е. И. Базы данных в высокопроизводительных информационных системах / Е. И. Николаев. - Ставрополь: СКФУ, 2016. - 163 с.

5 Уайт Т. Hadoop. Подробное руководство / Т. Уайт. – СПб.: Питер, 2013. – 672 с.

6 Осовский С. Нейронные сети для обработки информации / Пер. с польского И. Д. Руднинского. – М.: Финансы и статистика, 2004. – 344 с.

7 Хайкин С. Нейронные сети. Полный курс / С. Хайкин. – М.: Вильямс, 2016. – 1104 с.

8 Рашид Т. Создаем нейронную сеть / Т. Рашид. – М.: Вильямс, 2017. – 272 с.

9 Жерон О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем / О. Жерон. – М.: Вильямс, 2018. – 688 с.

10 Тархов Д. Нейросетевые модели и алгоритмы. Справочник / Д. Тархов. – М.: Радиотехника, 2014. – 352 с.

11 Маккинни У. Python и анализ данных / У. Маккинни. – М.: ДМК-Пресс, 2015. – 2015. – 482 с.

12 Ed Finkler, OSMI Mental Health in Tech Survey // Open Sourcing Mental Illness. – Vol. 1, 2014. (Engl). – URL: https://osmihelp.org/research.html [21 June 2017].