Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки /специальность 09.04.04 – Программная инженерия
Направленность (профиль) образовательной программы Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«____»_____2023 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Использование технологии больших данных в области санитарно-эпидемиологического контроля

Исполнитель
студент группы 157-ом _____ Н.Ю. Копылов
(подпись, дата)

Руководитель
доцент, канд. техн. наук _____ С.Г. Самохвалова
(подпись, дата)

Руководитель научного
содержания программы
магистратуры
профессор, доктор техн. наук _____ И.Е. Ерёмин
(подпись, дата)

Нормоконтроль
доцент, канд. техн.наук _____ Л.В. Никифорова
(подпись, дата)

Рецензент _____ _____
(подпись, дата)

Благовещенск 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____А.В. Бушманов
«_____»_____2023 г

З А Д А Н И Е

К выпускной квалификационной работе студента Копылова Никиты Юрьевича

1. Тема выпускной работы: Использование технологии больших данных в области санитарно-эпидемиологического контроля

(утверждена приказом от _____№ _____)

2. Срок сдачи студентом законченной работы 23.06.2023 г.

3. Исходные данные к выпускной квалификационной работы: предметная область, отчеты по практической подготовке, результаты выступления на научной конференции

4. Содержание выпускной квалификационной работы: описание предметной области, описание алгоритма сбора данных, описание алгоритма машинного обучения, описание разработанного программного обеспечения

5. Дата выдачи задания: 31.01.2023 г.

Руководитель выпускной квалификационной работы: Самохвалова Светлана Геннадьевна, доцент, канд. техн. наук

Задание принял к исполнению (31.01.2023): _____

РЕФЕРАТ

Магистерская диссертация содержит 71 с., 30 рисунков, 3 таблицы, 52 источника.

БОЛЬШИЕ ДАННЫЕ, АЛГОРИТМ СБОРА ДАННЫХ, МАШИННОЕ ОБУЧЕНИЕ, ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ, РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА, ТЕСТИРОВАНИЕ, АНАЛИЗ ПРАКТИЧЕСКОЙ ЗНАЧИМОСТИ

Цель работы: разработать программу, позволяющую осуществлять сбор данных по заболеваемости COVID-19 и с помощью методов машинного обучения выполнять прогнозирование динамики заболеваемости.

В данной работе описывается процесс разработки программного обеспечения для осуществления анализа и прогнозирования заболеваемости COVID-19 на территории Российской Федерации с использованием технологии больших данных, а также машинного обучения. Подробно описаны алгоритмы сбора данных и их обработки, а также алгоритм подготовки модели машинного обучения. Создан удобный пользовательский графический интерфейс для работы с программными ресурсами продукта.

Задачи выпускной работы:

- рассмотреть совокупность уже существующих способов и методик использования технологии больших данных в сфере эпидемиологического контроля;
- определить проблемы, существующие на данный момент в этой сфере;
- определить возможности и методы решения данных проблем;
- разработать соответствующее программное обеспечение, которое бы позволило реализовать результаты исследования на практике.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время в сфере информационных технологий весьма актуальным остаётся вопрос применения методов и технологий анализа больших данных. Большие данные – это сложные наборы данных очень большого размера, собираемые из разнообразных нестандартных источников. Обработка таких данных программами традиционного типа весьма затруднительна, т.к. объём и скорость поступления больших данных достигают очень больших значений, при этом продолжая постоянно возрастать. Это приводит к необходимости применения особых методов анализа, как, например: использование искусственных нейронных сетей, имитационное моделирование, пространственный анализ, статистический анализ и т.д. Анализ больших данных позволяет получить всю необходимую информацию для выработки управленческих решений в различных сферах деятельности, как, например: розничная торговля, финансовая отрасль, энергетика, транспорт и т.д.

Применение технологии Big Data весьма эффективно в сфере эпидемиологического контроля, т.к. позволяет осуществлять сбор огромных массивов данных для оперативного отслеживания количества заболевших, выздоровевших, вакцинированных, определения групп населения, наиболее подверженных заражению и т.д. Использование данной технологии, а также методов машинного обучения позволяет выполнять прогнозирование динамики заболеваемости, выявляя не известные ранее закономерности. На основании всего вышесказанного, можно сделать вывод, что тема магистерской диссертации является актуальной.

Цель работы: разработать программу, позволяющую осуществлять сбор данных по заболеваемости COVID-19 в Российской Федерации и с помощью методов машинного обучения выполнять прогнозирование динамики заболеваемости.

Задачи исследования: рассмотреть совокупность уже существующих способов и методик использования технологии больших данных в сфере эпидемиологического надзора; определить проблемы, существующие на данный момент в этой сфере; определить возможности и методы решения данных проблем и разработать соответствующее программное обеспечение, которое бы позволило реализовать результаты исследования на практике.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЪЕКТА ИССЛЕДОВАНИЯ

1.1 Основные теоретические сведения о больших данных

Существует множество различных определений понятия «большие данные». Самое первое определение было сформулировано следующим образом: «Большие данные – это данные, сбор, управление и обработку которых невозможно осуществить с помощью наиболее часто используемых аппаратных сред и программных инструментов в течение допустимого для пользователя времени» [24].

Чаще всего такие данные описываются с помощью пяти основных характеристик, обозначаемых аббревиатурой 5V (рисунок 1) [52]:

- volume (объём);
- velocity (скорость);
- variety (разнообразие);
- veracity (точность);
- value (ценность).

Характеристика объём (volume) используется для обозначения количества и размеров получаемых и анализируемых данных. В случае Big Data размер всего массива данных может составлять десятки, сотни терабайт или даже петабайт. При этом не весь объём получаемых данных является полезным для решения выбранной задачи. Но ценность данных возможно определить лишь только после проведения их анализа.

Большие данные характеризуются очень большой скоростью поступления и обновления (характеристика velocity (скорость)). При этом обновление чаще всего подразумевает рост объёма данных.

Разнообразие (variety) больших данных обусловлено наличием огромного количества различных форматов, которые они могут иметь. Многообразие форматов, в свою очередь, обусловлено тем, что источники таких данных

так же весьма разнообразны. Так, данные могут быть неструктурированными, полуструктурированными или структурированными.

Характеристика точность (veracity) используется для обозначения достоверности и правдивости полученных данных. Собранные данные не всегда могут обладать необходимой для проведения дальнейшей с ними работы точностью. Некоторые фрагменты данных могут отсутствовать или быть недостоверными и не давать по-настоящему ценной информации.

Ценность (value) данных определяется количеством полезной информации, которую можно из них извлечь. В зависимости от специфики решаемой задачи полезность одних и тех же данных может различаться.

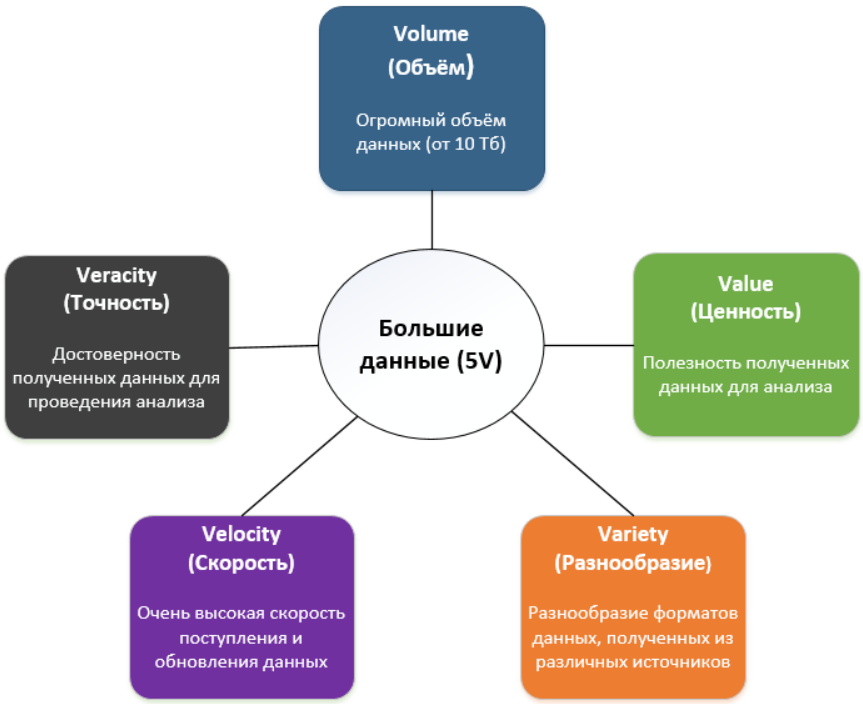


Рисунок 1 – Характеристики больших данных

Источники больших данных достаточно разнообразны, но среди них можно выделить три основных типа источников, генерирующих значительную часть всех таких данных [50]:

- источники машинных данных;
- источники социальных данных;
- источники данных о транзакциях.

Машинные данные – это автоматически генерируемые данные, получаемые из таких источников, как умные сенсоры, SIEM-логи, дорожные камеры, медицинские устройства и браслеты, мобильные телефоны и т.д. Генерация таких данных происходит либо по установленному расписанию, либо по определённому событию.

К социальным данным можно отнести данные получаемые из социальных медиа: комментарии, лайки и дизлайки, размещаемые посты на страницах в социальных сетях, загружаемые видео на видео-хостингах и т.д.

Транзакционные данные содержат информацию о всевозможных онлайн и офлайн транзакциях, включая время транзакции, методы оплаты, место проведения транзакции, список оплаченных продуктов и т.д. Источниками таких данных являются платёжные поручения, счета, электронные квитанции и т.д.

На рисунке 2 показаны другие примеры источников больших данных.

Рисунок 2 – Примеры источников больших данных

В настоящее время активное применение технологии Big Data обусловлено тем, что качественный анализ наборов данных больших объёмов позволяет выявлять скрытые в них корреляции и паттерны, знание которых позволяет весьма эффективно проводить планирование и принимать решения в различных областях человеческой жизнедеятельности. Так, например, сбор социальных больших данных и их анализ позволяет определять интересы и предпочтения различных пользователей тех или иных социальных медиа и предлагать им персонализированную рекламу тех товаров, которые могут быть им интересны [31].

В области бизнеса и маркетинга анализ больших данных позволяет организациям более эффективно принимать решения, оптимизировать бизнес-процессы и повышать эффективность маркетинговых кампаний. Также в данных областях технология больших данных позволяет анализировать поведение потребителей, проводить сегментацию аудитории, предсказывать спрос, улучшать персонализацию и проводить A/B-тестирование.

В медицинской сфере анализ больших данных может помочь в выявлении паттернов и тенденций в медицинских записях, исследованиях и клинических испытаниях. Это, в свою очередь, позволит более точно ставить диагнозы, предсказывать риски и оптимизировать процессы лечения.

В транспортной сфере применение данной технологии позволяет осуществлять планирование маршрутов, выполнять контроль трафика, прогнозировать области с высоким риском возникновения аварий.

В банковской сфере технология Big Data может применяться для осуществления управления рисками, выявления подозрительных операций с банковскими картами клиентов и т.д.

Также весьма эффективно применение данной технологии в области информационной безопасности. Многие существующие кибер–атаки имеют нечто общее: они разработаны для блокировки «шумов», вызванных оповещениями IDS/IPS, скрывая себя среди огромного объема информации, генериру-

мой повседневными операциями компаний–жертв атаки. Применение технологии больших данных позволяет распознавать сигнатуры таких «аномалий» и своевременно обнаруживать вторжения.

На рисунке 3 показаны другие примеры областей применения технологии Big Data:

Рисунок 3 – Сферы применения технологии больших данных

1.2 Технологии обработки больших данных

Программно-аппаратные средства для работы с большими данными должны обладать такими свойствами как: масштабируемость, распределённость, поддержка параллельных вычислений [2]. Необходимость наличия данных свойств обусловлена тем, что объёмы больших данных непрерывно возрастают с большой скоростью.

Чаще всего в качестве аппаратной платформы для систем обработки больших данных применяются многопроцессорные вычислительные системы

без совместного использования ресурсов (Shared Nothing Architecture), позволяющие обеспечить параллельную обработку данных, масштабируемую без деградации на сотни и тысячи узлов.

Аппаратно-программные решения, используемые для обработки больших данных, представляют собой телекоммуникационные шкафы, предназначенные для установки в центр обработки данных. Данные системы содержат кластер серверов и управляющее программное обеспечение для массово-параллельной обработки. Наиболее известными аппаратно-программными комплексами для обработки больших данных являются Aster MapReduce Appliance корпорации Teradata, Big Data Appliance корпорации Oracle, Greenplum Appliance корпорации EMC. Существуют также комплексы для аналитической обработки в оперативной памяти: система Hana компании SAP и система Exalytics корпорации Oracle, построенная на основе реляционной СУБД TimesTen и многомерной СУБД Essbase.

К основным технологиям обработки больших данных можно отнести:

- NoSQL;
- MapReduce;
- Hadoop;
- параллельные СУБД.

Термин NoSQL (Not Only SQL — не только SQL) используется для обозначения нереляционных типов баз данных, в которых используется модель представления данных, отличная от традиционной табличной, характерной для реляционных баз данных. Применимость таких систем для обработки больших данных обусловлена тем, что они проектируется с расчётом на неограниченное горизонтальное масштабирование, т.е. добавление или удаление узлов в кластере не будет влиять на работоспособности системы.

В NoSQL-системах отсутствует поддержка транзакций с ACID-свойствами (Atomicity — атомарность, Consistency — согласованность, Isolation — изолированность, Durability — долговременность), поэтому они не могут ис-

пользоваться в приложениях, где указанные свойства являются необходимостью, например, в системах обслуживания бирж и банков. В связи с этим в NoSQL-системах, обрабатывающих Большие Данные, согласованностью данных жертвуют ради достижения двух других свойств: доступность данных и устойчивость к разбиению данных по узлам вычислительной системы. Подобная особенность распределенных вычислений — возможность обеспечить не более двух свойств из трех, указанных выше, — известна как теорема CAP (аббревиатура от англоязычных названий свойств: Consistency, Availability, Partition tolerance).

Для работы с большими данными также весьма применима модель распределенных вычислений MapReduce, разработанная корпорацией Google в 2004 году. Данная модель ориентирована на параллельные вычисления в распределенных кластерах.

Кратко принцип работы MapReduce можно описать как последовательность трёх шагов:

- разделение информационного массива на части;
- параллельная обработка каждой части на отдельном узле;
- финальное объединение всех результатов.

Обработка данных выполняется в виде последовательности из двух шагов: Map и Reduce. На шаге Map узел-мастер получает входные данные и распределяет их по узлам-рабочим. На шаге Reduce узел-мастер выполняет свертку данных, предварительно обработанных рабочими, и отправку конечного результата пользователю. На рисунке 4 показана общая схема работы MapReduce.

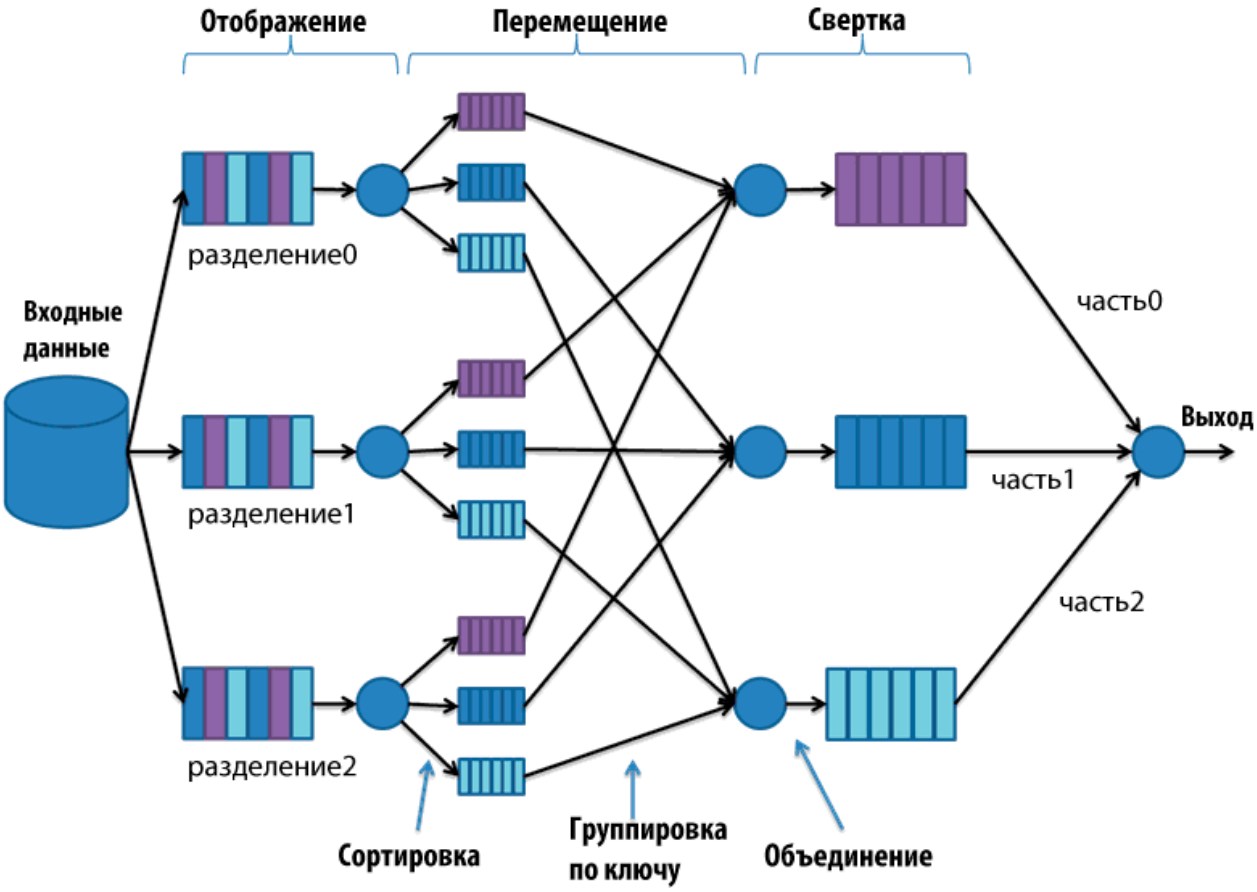


Рисунок 4 – Схема работы MapReduce

В рамках парадигмы MapReduce был разработан Hadoop — проект фонда Apache Software Foundation, который представляет собой фреймворк для разработки и выполнения распределенных программ, работающих на кластерах из сотен и тысяч узлов.

Hadoop состоит из четырёх основных модулей:

- Hadoop Common – набор инфраструктурных программных библиотек и утилит;

- HDFS – распределенная файловая система;

- YARN – система для планирования заданий и управления кластером;

- Hadoop MapReduce – платформа программирования и выполнения распределенных MapReduce вычислений.

С Hadoop ассоциирован ряд проектов и технологий, многие из которых изначально развивались в рамках этого проекта, но впоследствии стали самостоятельными.

Параллельные СУБД, в противовес NoSQL-решениям, не отказываются от реляционной модели данных. Базовой концепцией параллельных СУБД является фрагментный параллелизм, который подразумевает горизонтальную фрагментацию каждой таблицы базы данных по дискам кластерной системы. Способ фрагментации определяется функцией фрагментации, которая для каждой записи таблицы вычисляет номер вычислительного узла кластера, где должна храниться данная запись.

На каждом узле кластерной системы запускается параллельный агент (ядро СУБД), обрабатывающий запросы пользователей. Один и тот же запрос параллельно выполняется каждым агентом над «своими» фрагментами таблиц базы данных, и затем полученные частичные результаты сливаются в результирующую таблицу. Несмотря на независимую обработку агентами «своих» фрагментов базы данных, для получения корректного результата необходимо выполнять пересылки записей во время выполнения запроса (при выполнении операции соединения двух таблиц по общей колонке).

Для организации таких пересылок выполняется распараллеливание последовательного плана запроса: в дополнение к обычным реляционным операциям (соединение, выборка, проекция и др.) в нужные места плана запроса вставляется специальная операция exchange, которая обеспечивает пересылку «чужих» записей соответствующим параллельным агентам и прием «своих» записей от них, не нарушая хода выполнения запроса. В силу того, что различные фрагменты таблицы могут иметь существенно различные размеры (такая ситуация носит название «Перекос данных»), в параллельных СУБД для балансировки загрузки параллельных агентов используется техника частичной репликации базы данных. К классу параллельных СУБД можно отнести системы Greenplum, Netezza, EXASOL и др.

1.3 Машинное обучение

Машинное обучение (Machine Learning) - это подраздел искусственного интеллекта, относящийся к разработке алгоритмов и моделей, которые позволяют компьютерным системам обучаться выполнению тех или иных операций на основе получаемых ими данных [19]. Чаще всего задача машинного обучения заключается в разработке определённой предсказательной модели, которая могла бы применяться для осуществления предсказания тех или иных результатов на основе новых поступающих данных. Например, это могут быть модели, позволяющие с той или иной вероятностью определять, является ли полученное на электронную почту письмо спамом или нет, прогнозировать результаты спортивных событий, предсказывать поведение покупателя и т.д.

Существуют различные типы машинного обучения. На рисунке 5 показаны три основных типа машинного обучения с примерами их применения для решения различных задач

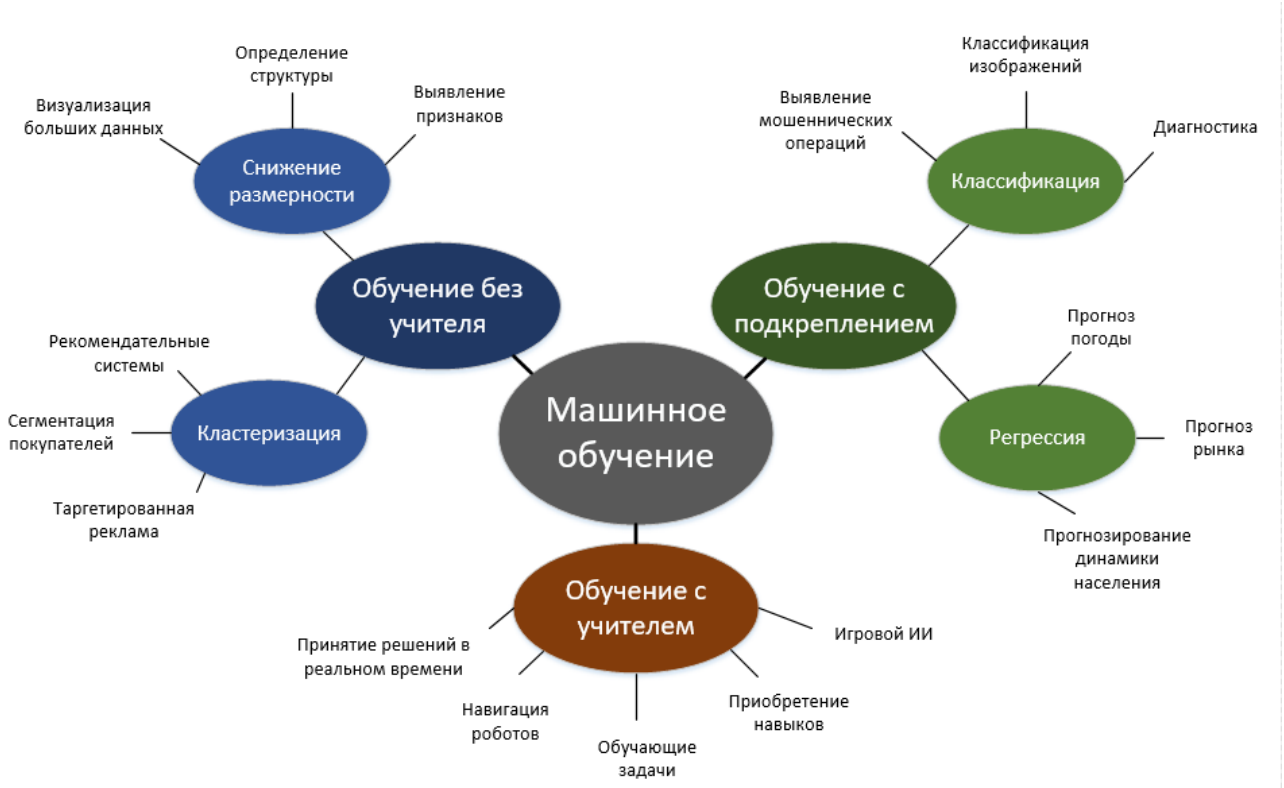


Рисунок 5 – Типы машинного обучения

Машинное обучение с учителем.

В случае применения данного типа машинного обучения модель обучается на основе помеченных данных, которые содержат входные примеры и соответствующие им правильные ответы или метки. В основе данного подхода лежит задача нахождения зависимости между входными данными и целевыми значениями с целью предсказания этих значений для новых, ранее не встречавшихся данных [20].

Процесс обучения включает в себя следующие шаги:

- подготовка данных;
- выбор модели;
- обучение модели;
- оценка модели;
- применение модели.

В процессе подготовки данные разделяются на две части - обучающую выборку и тестовую выборку. Обучающая выборка используется для обучения модели, а тестовая выборка - для оценки ее производительности и обобщающей способности. Затем выбирается алгоритм или модель, которая будет использоваться для обучения. В ходе обучения модель подстраивается под обучающую выборку путем настройки внутренних параметров на основе входных данных и соответствующих меток. После завершения обучения производится оценка модели: модель оценивается на тестовой выборке для оценки ее способности обобщать знания на новые данные. По завершении успешного обучения и оценки модель может быть использована для предсказания значений на новых, ранее неизвестных данных.

Машинное обучение с учителем используется для решения двух основных классов задач: классификации и регрессии. Регрессия относится к задачам предсказания непрерывных или числовых значений. Цель регрессии заключается в том, чтобы построить функциональную зависимость между входными признаками и выходными значениями, чтобы можно было предсказать численное значение на основе новых входных данных. Например, предсказание

цены дома на основе его характеристик, прогнозирование температуры, предсказание дохода на основе образования и опыта работы и т.д.

Классификация, с другой стороны, относится к задачам, где требуется отнести объекты к определенным категориям или классам на основе их характеристик. В этом случае, модель обучается на основе помеченных данных и стремится построить границы или правила разделения между различными классами. Примеры классификации включают определение, является ли электронное письмо спамом или не спамом, распознавание изображений (например, определение, содержит ли изображение кошку или собаку), идентификацию заболеваний на основе медицинских симптомов и т.д.

Машинное обучение без учителя.

В данном случае модель обучается на непомеченных данных, то есть данных, не имеющих предопределенных меток или правильных ответов. Вместо того, чтобы искать зависимости между входными данными и целевыми значениями, модель ищет скрытые структуры или паттерны в данных.

Основные задачи машинного обучения без учителя включают:

- кластеризацию (Clustering): это процесс разделения набора данных на группы или кластеры, где объекты внутри каждого кластера схожи между собой, а объекты разных кластеров различаются, кластеризация помогает выявить внутренние структуры данных и сегментировать их на группы схожих объектов;

- снижение размерности (Dimensionality Reduction): это процесс уменьшения количества признаков или переменных в наборе данных, позволяющий представить данные в более компактной форме, удаляя ненужные или коррелирующие признаки, упростить анализ данных, улучшить производительность моделей и помочь визуализировать данные в пространстве меньшей размерности;

- обнаружение аномалий (Anomaly Detection): это задача выявления редких, необычных или аномальных образцов в наборе данных, выполняемая путём обучения на нормальных или типичных данных и последующей идентификации объектов, которые существенно отличаются от остальных;

- ассоциативное извлечение (Association Rule Learning): это задача выявления связей, зависимостей или ассоциаций между переменными или признаками в наборе данных, позволяющая идентифицировать часто встречающиеся комбинации признаков или правила, что может быть полезно для рекомендательных систем, анализа покупательского поведения и других приложений.

Машинное обучение без учителя играет важную роль в анализе данных и помогает выявить структуры, паттерны и информацию, которую сложно обнаружить вручную.

Машинное обучение с подкреплением.

В данном случае агент (компьютерная программа или система) обучается принимать решения в интерактивной среде с целью максимизации некоторой численной награды или кумулятивного вознаграждения.

В машинном обучении с подкреплением агент действует в окружении, в котором он может воспринимать состояния окружающей среды, выполнять определенные действия и получать обратную связь в виде вознаграждений или штрафов. Цель агента - научиться принимать решения, которые максимизируют накопленное вознаграждение на протяжении времени.

Основные компоненты машинного обучения с подкреплением:

- агент (agent): это сущность, принимающая решения и взаимодействующая с окружающей средой, она может воспринимать текущее состояние окружения, выбирать действия для выполнения и получать обратную связь в виде вознаграждений;

- среда (environment): это контекст, в котором действует агент, среда может быть физической, виртуальной или абстрактной, и она определяет правила взаимодействия и динамику состояний и вознаграждений;

- действия (actions): это возможные варианты, из которых агент может выбирать для взаимодействия с окружающей средой, они могут быть дискретными (например, выбор из нескольких предопределенных вариантов) или непрерывными (например, изменение значений параметров).

- состояния (states): это описание текущего состояния окружения, которое воспринимает агент, они могут быть полными (включающими всю информацию о текущем состоянии) или частичными (ограниченными набором доступной информации).

- вознаграждения (rewards): это численная обратная связь, которую агент получает от среды в ответ на выполненные действия, вознаграждения определяются с помощью заранее заданных правил или функций, и агент стремится максимизировать накопленное вознаграждение в долгосрочной перспективе.

Агент в машинном обучении с подкреплением обучается через множество взаимодействий с окружающей средой. Он исследует различные стратегии и методы принятия решений для достижения наилучших результатов в виде максимального накопленного вознаграждения. Машинное обучение с подкреплением широко применяется в таких областях, как автономная навигация, управление роботами, игры и финансовые торги.

1.4 Существующие решения

В настоящее время существует достаточно большое количество моделей искусственного интеллекта, разработанных для прогнозирования развития тяжёлых форм, летальных исходов и диагностики COVID-19 в целом.

Исследователями Нью-Йоркского университета была разработана одна из первых моделей регулирования принятия врачебных решений, предназначенная для проведения сортировки пациентов с подозрением на COVID-19. Данная модель показала высокую точность (80%) ля формирования прогноза развития ОРДС у пациентов с COVID-19 без явных клинических признаков тяжелого течения заболевания.

Существуют различные мобильные приложения для проведения диагностики COVID-19 на основании симптомов заболевания, прогнозирования тяжёлого течения и летальных исходов. В майском номере журнала Nature Machine Intelligence была опубликована статья, в которой был представлен новый, легко интерпретируемый алгоритм оценки риска смерти больных COVID-19 по количеству лимфоцитов, а также уровням лактатдегидрогеназы (ЛДГ) и С-реактивного белка (СРБ).

D. Bertsimas et al. на основе обработки данных 3 927 пациентов с COVID-19 из 6 независимых центров, включающих 33 различных больницы, разработали COVID-19 Mortality Risk (CMR) - инструмент с использованием алгоритма XGBoost для прогнозирования смертности [5]. Модель CMR использует машинное обучение для создания точных прогнозов смертности с использованием общедоступных клинических функций: пол, возраст, температура тела, ЧСС, SpO2, лабораторные значения (аминотрансферазы, креатинин, мочевина, калий, натрий, глюкоза, гемоглобин, лейкоциты, тромбоциты, средний объем эритроцитов, протромбиновое время). Это первая оценка риска, разработанная и проверенная на когорте пациентов с COVID-19 из Европы и США.

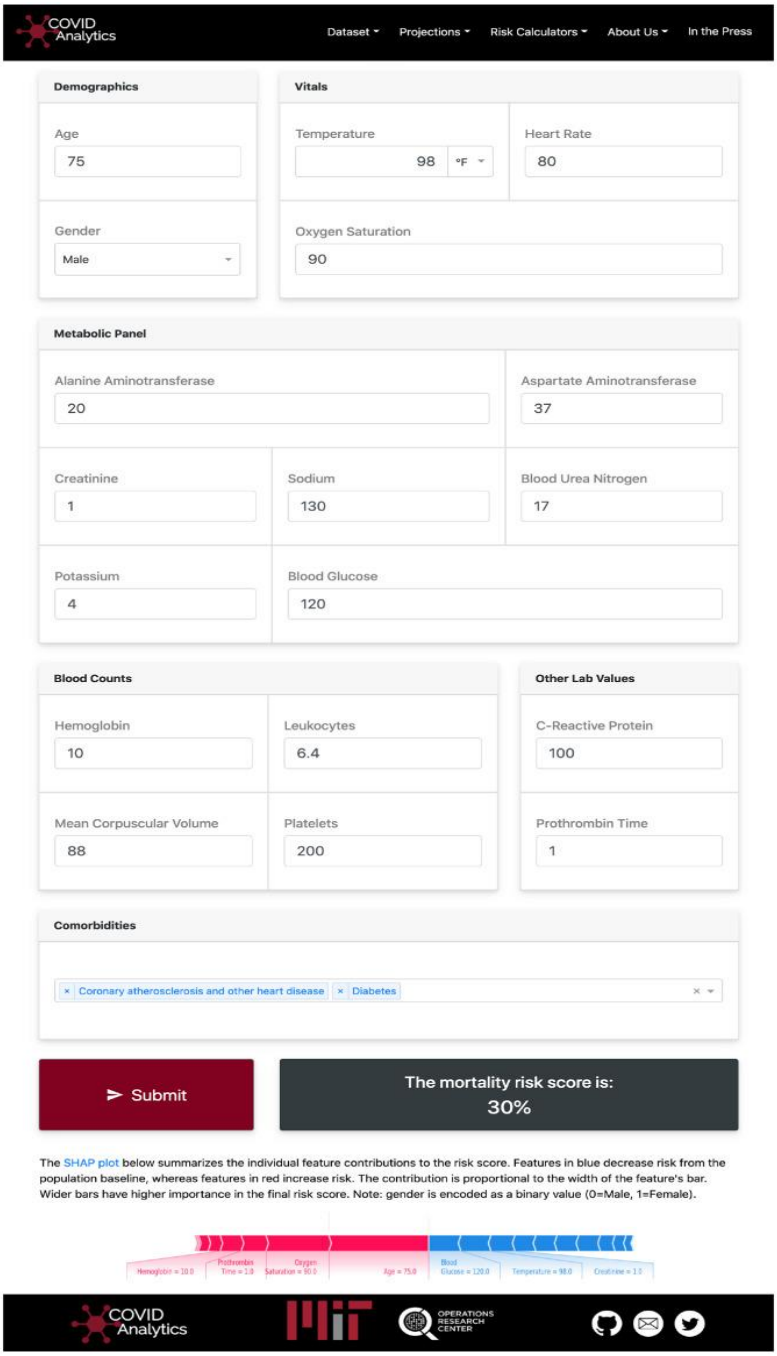


Рисунок 6 – COVID-19 Mortality Risk

H. Estiri et al. создали прогностическую модель для осуществления прогнозирования риска смертности и изучения факторов риска смерти в разных возрастных группах. Данные 16 709 пациентов с COVID-19 для обучения модели были получены из медицинских записей Бостонской некоммерческой сети больниц и врачей. Применяя вычислительный алгоритм Minimize Sparsity Maximize Relevance (MSMR), дополненный клинической экспертизой, авторы

построили кластеры предикторов смертности на основе медицинских данных для пациентов с COVID-19. На основе этих кластеров был разработан набор обобщенных линейных моделей (GLM) для прогнозирования смертности и понимания взаимосвязи различных демографических характеристик и заболеваний со смертностью от COVID-19. Ключевыми факторами, ухудшающими прогноз при COVID-19, в данном исследовании были названы возраст, наличие у пациента в анамнезе пневмонии, хронической болезни почек, СД 2-го типа с осложнениями, артериальной гипертензии и сердечной недостаточности. Среди 65–85-летних пациентов возраст оставался самым неблагоприятным прогностическим фактором. Также в этом возрасте большое влияние на прогноз оказывали респираторные заболевания, которые увеличивали риск смертности от COVID-19. Они включали не только пневмонии в анамнезе, но и хронические обструктивные болезни легких, рак легких, тромбоэмболию легочной артерии, интерстициальные легочные болезни и курение. Несмотря на то, что все эти заболевания связаны с дыхательной системой, каждое из них в отдельности вносило свой вклад в повышение риска смерти среди пациентов с COVID-19.

Для пациентов, госпитализированных с вирусной пневмонией, Bender et al. разработали простой инструмент прогнозирования общей смертности, который был полезен для прогнозирования течения заболевания вновь поступивших больных и получения результатов их анализов. В исследовании было отмечено, что оценка иммунного статуса важна при мониторинге общего состояния пациента при COVID-19. В исследовании все подтипы Т-лимфоцитов были снижены в группе с высоким риском летального исхода, что характеризует дефицит адаптивного иммунного ответа в данной группе. Предыдущие исследования при вирусной инфекции показали, что адаптивные Т-клетки обеспечивают более широкий и более продолжительный перекрестно-реактивный клеточный иммунитет с меньшими штамм-специфическими ограничениями, особенно CD8+ Т-клеток. Кроме того, документально подтверждено, что более высокий уровень провоспалительных цитокинов связан с тяжестью

заболевания и повреждением легких. Соответственно, ИЛ-2R и ИЛ-6, которые, как оказалось, в значительной степени коррелировали с тяжестью заболевания, дополняли функцию CD8+ Т-клеток

J. Gong et al. создали модель, основанную на 189 пациентах с COVID-19, в которую были включены биомаркеры для определения вероятности тяжелого заболевания. Эта модель имела AUC ROC 0,85 (95% ДИ: 0,79–0,92).

2 АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ

2.1 Алгоритм сбора данных

Для проведения машинного обучения необходимо предварительно осуществить сбор соответствующих данных, а именно статистики по заболеваемости COVID-19 в РФ. Получить такие данные возможно на специализированный сайтах, таких как: стопкоронавирус.рф, coronavirus-monitor.info, coronavirus-control.ru и т.д. Было решено использовать сайт coronavirus-monitor.info, т.к. на нём можно получить данные по заболеваемости COVID-19 во всех регионах РФ не только на сегодняшний день, но и за прошедшие месяцы и годы, что необходимо для проведения машинного обучения.

Рассмотрим алгоритм сбора данных. Исходной ссылкой для парсинга будет являться https://coronavirus-control.ru/latest-news/moskva/ (рисунок 7). Данная ссылка перенаправляет на страницу, где можно получить ссылки на другие страницы со статистикой по заболеваемости за различные даты. Вначале необходимо направить GET-запрос на получение HTML-кода исходной страницы. После успешной отправки запроса полученный HTML-код будет записан в отдельную переменную для дальнейшей работы с ним.

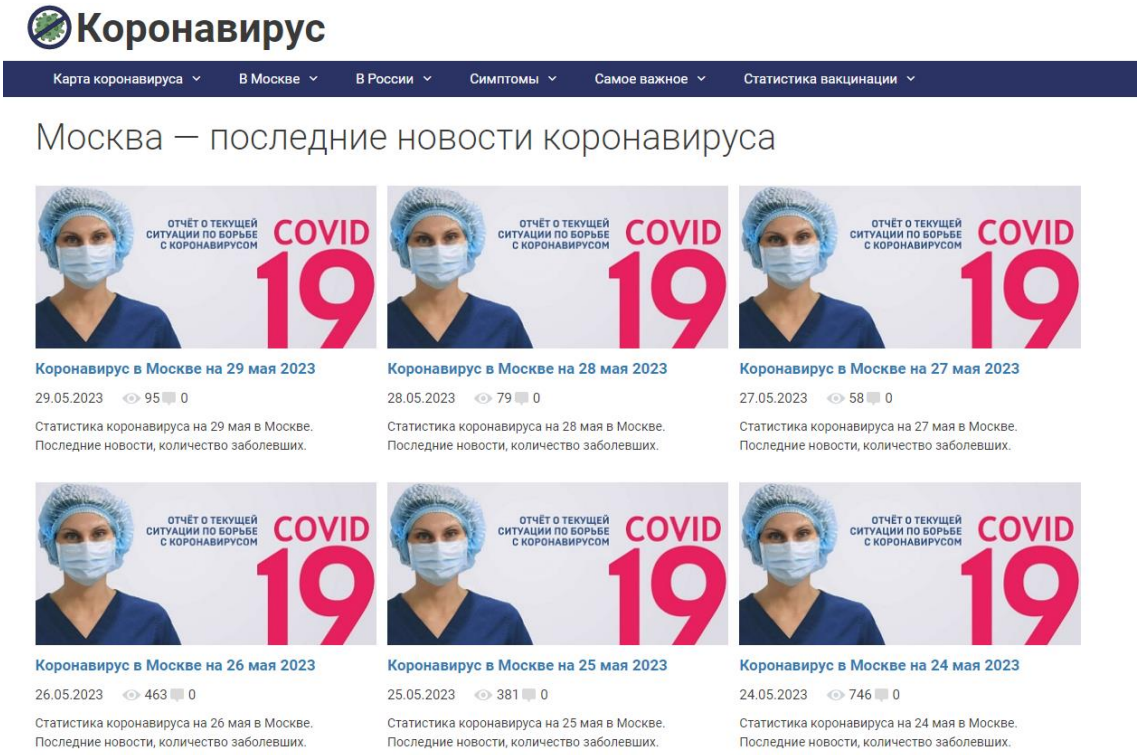


Рисунок 7 – Исходная страница для парсинга

Далее из полученного HTML-кода необходимо извлечь ссылки на страницы с самой статистикой. Необходимая информация содержится в тегах , имеющих класс «news-element». Каждый такой элемент содержит гиперссылку на страницу с данными по заболеваемости за выбранную дату. После извлечения всех описанных ранее тегов необходимо произвести отбор данных за необходимый период. Информация о дате содержится в теге . Отобрав только элементы за интересующие даты, наконец необходимо осуществить извлечение ссылок на страницы со статистикой, содержащиеся в атрибуте href тега <a>. Т.к. на одной странице отображается лишь 15 блоков, содержащих ссылки на статистику за последние 15 дней соответственно, то для получения ссылок на более поздние данные необходимо осуществлять переход на другие страницы. Исходная веб-страница содержит кнопки для перехода на другие страницы. Ссылка на следующую страницу содержится в атрибуте href тега <a>, имеющего класс «next page-numbers». Таким образом, если существует необходимость получения данных за более поздние периоды, то выполняется поиск тега <a class=’next-page’>, и, если такой тег удаётся найти,

то из атрибута href извлекается ссылка на следующую страницу, и затем выполняется парсинг ссылок на статистику с этой страницы по схеме, описанной ранее.

После того, как список ссылок получен, можно начинать парсинг данных по заболеваемости. На рассматриваемых страницах статистика по COVID-19 представлена в виде HTML-таблицы, пример которой представлен на рисунке 8.

Статистика коронавируса по регионам России на 29 мая 2023

Область	Заражено	Активные случаи	Погибло	Выздоровело	Летальность	Восстановление
Россия	22 915 494	172 673	398 893	22 343 928	1.74%	97.51%
Москва	3 515 236	138 433	48 829	3 327 974	1.39%	94.67%
Санкт-Петербург	1 944 541	2 491	37 744	1 904 306	1.94%	97.93%
Московская область	1 151 687	2 339	15 477	1 133 871	1.34%	98.45%
Свердловская область	591 140	2 092	10 941	578 107	1.85%	97.80%
Самарская область	541 297	373	8 298	532 626	1.53%	98.40%
Нижегородская область	522 626	434	11 866	510 326	2.27%	97.65%
Воронежская область	505 331	904	8 661	495 766	1.71%	98.11%
Пермский край	473 722	382	9 340	464 000	1.97%	97.95%
Ростовская область	456 907	316	11 084	445 507	2.43%	97.50%
Красноярский край	439 322	674	11 349	427 299	2.58%	97.26%
Челябинская область	412 322	356	8 396	403 570	2.04%	97.88%
Иркутская область	372 913	644	8 752	363 517	2.35%	97.48%

Рисунок 8 – Пример таблицы со статистикой на сайте

Для начала необходимо так же, как и в случае с исходной ссылкой, отправить GET-запрос на получение HTML-кода страницы. После этого из кода извлекается фрагмент с таблицей (тег <table>), и полученная таблица записывается в отдельный объект. Затем необходимо выбрать из таблицы строку, содержащую информацию по интересующему региону. В найденной строке выбираются нужные столбцы («Область» и «Заражено»), а также добавляется дополнительный столбец, содержащий дату. В итоге, после выполнения парсинга данных по каждой ссылке из списка, формируется таблица, содержащая

динамику заболеваемости COVID-19 за выбранный период времени по определённой области. Пример такой таблицы показан на рисунке 9:

	Дата	Область	Заражено
0	2023-05-25	Москва	3 515 236
1	2023-05-24	Москва	3 515 236
2	2023-05-23	Москва	3 512 963
3	2023-05-22	Москва	3 512 963
4	2023-05-21	Москва	3 512 963
5	2023-05-19	Москва	3 512 963
6	2023-05-18	Москва	3 512 963
7	2023-05-17	Москва	3 512 963
8	2023-05-16	Москва	3 512 963
9	2023-05-15	Москва	3 512 720
10	2023-05-14	Москва	3 512 720
11	2023-05-13	Москва	3 512 267
12	2023-05-12	Москва	3 511 825
13	2023-05-11	Москва	3 511 223
14	2023-05-10	Москва	3 510 958
15	2023-05-09	Москва	3 510 657
16	2023-05-08	Москва	3 510 386
17	2023-05-07	Москва	3 510 013
18	2023-05-06	Москва	3 509 635
19	2023-05-05	Москва	3 509 147
20	2023-05-04	Москва	3 508 570
21	2023-05-03	Москва	3 507 807

Рисунок 9 – Пример таблицы с полученными данными

После того, как данные получены, они записываются в файл формата CSV. Полный алгоритм сбора данных представлен на рисунке 10 в виде блок-схемы

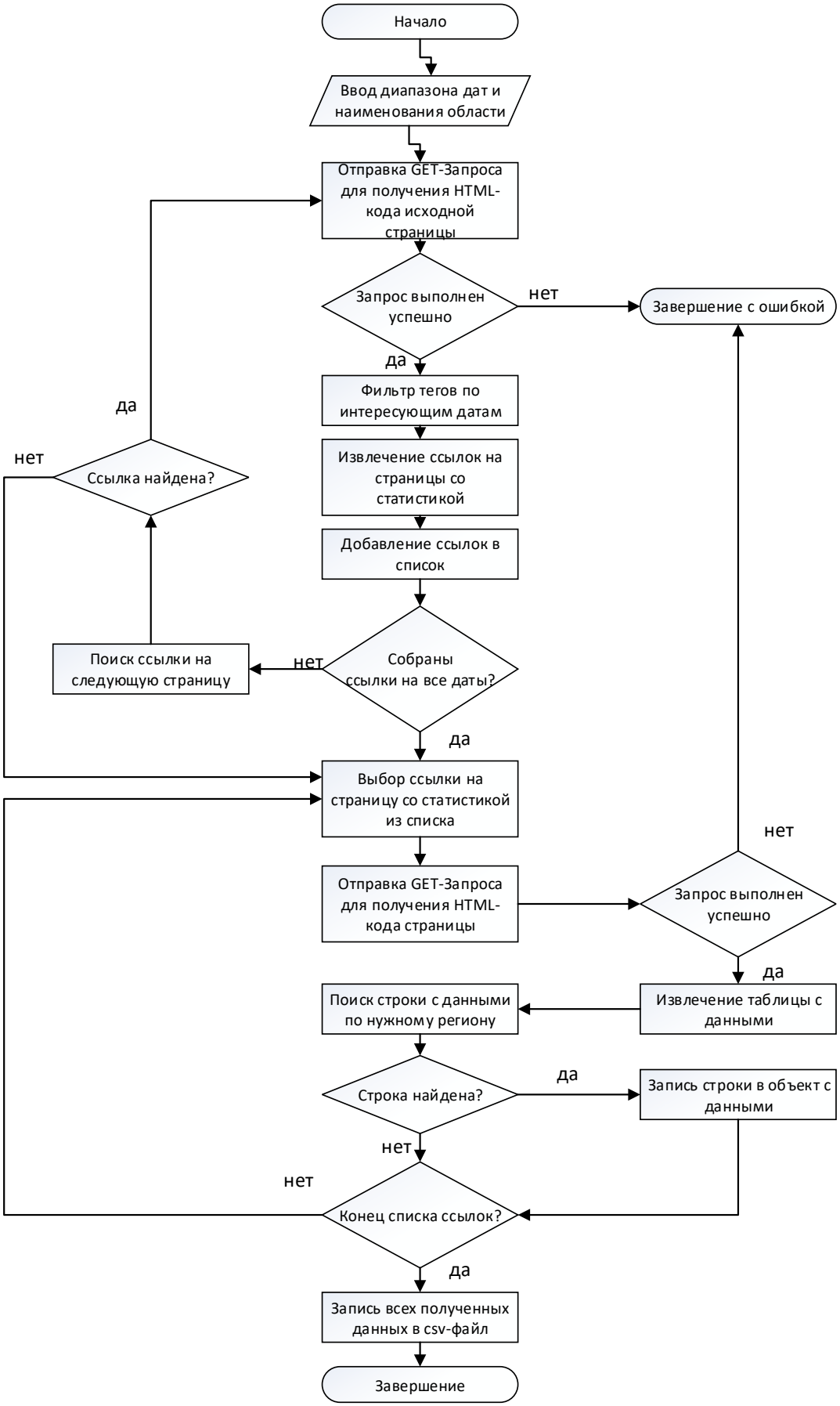


Рисунок 10 – Алгоритм сбора данных

2.2 Алгоритм подготовки данных

После того, как данные собраны, необходимо преобразовать их в форму, пригодную для осуществления машинного обучения.

Во-первых, необходимо провести очистку данных для исправления ошибок в полученных значениях и заполнения недостающих данных. Всё это позволит повысить качество собранных данных, что, в свою очередь, повысит эффективность машинного обучения. В целом, очистка включает в себя:

- просмотр всех полученных данных и исключение столбцов с отсутствующими данными;

- выбор столбцов, которые будут использоваться при непосредственном обучении;

- исключении строк с отсутствующими данными в оставшихся столбцах;

- исправление опечаток и объединение эквивалентных значений;

- исключение строк, находящихся вне нужного диапазона.

После выполнения очистки необходимо осуществить преобразование данных в читаемый формат. Данный процесс может включать изменение форматов полей, изменение соглашений об именовании, а также корректировку значений и единиц измерения для обеспечения их соответствия.

Следующим шагом необходимо провести маркировку данных. Под маркировкой данных понимают процесс идентификации необработанных данных и добавления одной или нескольких значимых информативных меток для обеспечения контекста, что позволяет модели машинного обучения учиться на этих данных

2.3 Алгоритм подготовки модели машинного обучения

Для выполнения прогнозирования должна быть подготовлена соответствующая модель машинного обучения. Для начала нужно получить данные для обучения. Если данные были собраны ранее, то они будут храниться в CSV файле, и их нужно будет лишь считать. Если же сбор не осуществлялся, то нужно будет его провести согласно алгоритму, описанному в пункте 1.1.

После того, как данные получены, необходимо провести их разбиение на отдельные наборы. Один набор будет использоваться для тестирования модели, а все остальные - для обучения модели. Данный процесс (обучение-тестирование) будет продолжаться k раз, где k – количество наборов данных, таким образом, чтобы каждый набор был использован для проверки ровно один раз. Данная методика называется кросс-валидацией или перекрёстной проверкой. Такая проверка позволяет снизить риск недообучения или переобучения модели, выбрав наиболее эффективную модель путём вычисления ошибки на тестовом наборе данных, который не используется для обучения. В качестве показателя эффективности обучения используется средняя ошибка кросс-валидации. Недостатком такой методики может являться то, что осуществление проверки может занимать достаточно много времени, т.к. необходимо многократно обучать модель. На рисунке 11 показана схема 4-х кратной кросс-валидации.

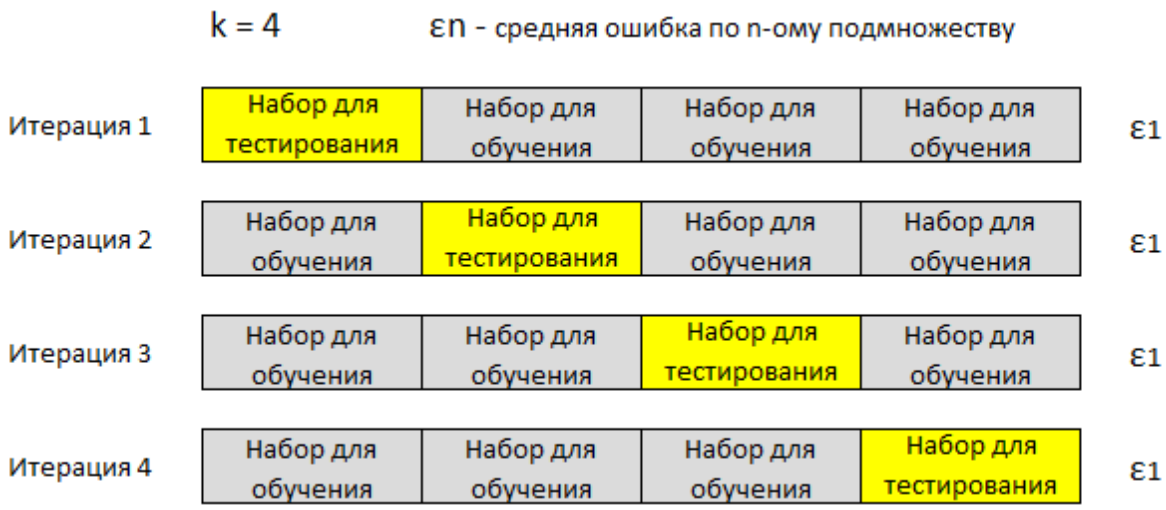


Рисунок 11 – Схема 4-х кратной кросс-валидации

После того, как было проведено обучение и тестирование модели, необходимо сохранить её для дальнейшего использования. Полный алгоритм обучения модели представлен в виде блок-схемы на рисунке 7.

После того, как было проведено обучение и тестирование модели, необходимо сохранить её для дальнейшего использования. Полный алгоритм обучения модели представлен в виде блок-схемы на рисунке 12.

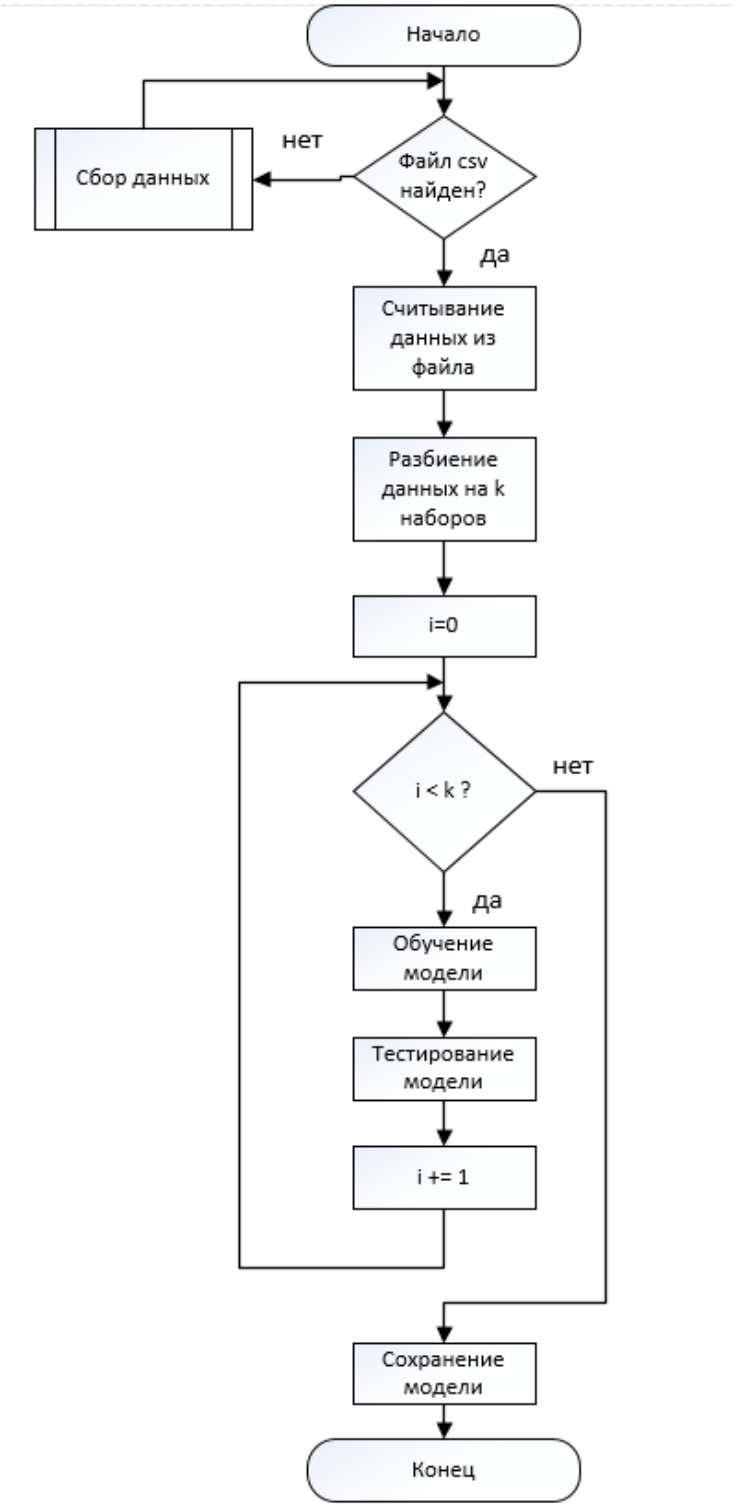


Рисунок 12 – Алгоритм подготовки модели машинного обучения

2.4 Обоснование выбора программно-технического обеспечения

2.4.1 Выбор языка программирования

В качестве языка программирования был выбран Python 3.9. Данный выбор обусловлен тем, что Python имеет все необходимые библиотеки для сбора, обработки и анализа данных, а также выполнения машинного обучения.

Если же говорить об языке Python в целом, то можно выделить следующие его ключевые особенности:

- динамическая типизация (при написании кода не нужно объявлять типы переменных, т.к. Python сам определяет их во время выполнения, что значительно ускоряет написание программы);

- является языком высокого уровня (сильная близость Python к естественным языкам облегчает написание сложных алгоритмов и портирование программ на другие платформы);

- является интерпретируемым языком (код выполняется построчно, и, в случае возникновения ошибки, её достаточно легко обнаружить по ссылке на соответствующую строку);

- является объектно-ориентированным языком (программа строится из совокупности взаимодействующих друг с другом объектов, каждый из которых является экземпляром некоторого класса, который может входить в определённую иерархию наследования);

- простота (в Python используются наименования, похожие на слова английского языка, а также вместо фигурных скобок используются отступы);

- большое количество библиотек (широкий выбор библиотек позволяет использовать Python для решения огромного количества разнообразных задач, таких как: машинное обучение, веб-разработка, автоматизация тестирования ПО и т.д.).

Python применяется для разработки большого количества различных типов приложений. Так, например, он может использоваться в веб-разработке, предлагая разработчику на выбор:

- фреймворки, такие как Django и Pyramid;

- микро-фреймворки, такие как Flask и Bottle;

- продвинутые системы управления содержимым, такие как Plone и Django CMS.

Также стандартные библиотеки Python поддерживают множество интернет-протоколов:

- HTML и XML;

- JSON;

- E-mail processing;

- FTP, IMAP и др.;

- простой в использовании программный интерфейс.

Python Package Index (каталог пакетов Python) предоставляет доступ к дополнительным библиотекам для веб-разработки:

- Requests;

- Beautiful Soup – HTML-парсер, позволяющий осуществлять обработку всех видов HTML;

- Feedparser для парсинга RSS/Atom;

- Paramiko, реализующая SSH2-протокол;

- Twisted Python – фреймворк для асинхронного сетевого программирования.

Python широко используется для выполнения научных и числовых вычислений, предлагая разработчикам большой набор соответствующих библиотек:

- SciPy – набор пакетов для выполнения математических, научных и инженерных вычислений;

- Pandas;

- IPython – высокоэффективная интерактивная оболочка, позволяющая осуществлять простое редактирование и запись рабочего сеанса, а также поддерживающая визуализацию и параллельные вычисления.

Ещё стоит упомянуть, что Python возможно применять для создания графических пользовательских интерфейсов, используя соответствующий инструментарий:

- Tkinter – библиотека, для создания приложений с графическими пользовательскими интерфейсами;

- wxWidgets – кроссплатформенная библиотека инструментов с открытым исходным кодом для написания настольных или мобильных приложений с графическим интерфейсом;

- Kivy – разработка мультитач-приложений;

- использование Qt через библиотеки pypt или pyside.

Также среди библиотек Python, используемых для реализации графического пользовательского интерфейса, можно выделить PyQt5 - библиотеку, которая, как и PyPT и PySide, предоставляет возможность привязки к Qt-фреймворку. Qt представляет собой фреймворк с открытым кодом, используемый для разработки приложений с графическим интерфейсом пользователя (GUI) [37]. Для реализации пользовательского интерфейса в разрабатываемом программном обеспечении предполагается использовать именно PyQt5. В целом, можно выделить следующие основные особенности данной библиотеки:

- возможность создания виджетов и компонентов (кнопки, текстовые поля, таблицы, списки, флажки и т.д.);

- поддержка обработки событий (возможность определять реакцию программы в ответ на те или иные действия пользователя);

- наличие макетов и возможности компоновки (упорядочивание и размещения виджетов в окне приложения);

- взаимодействие с веб-контентом (библиотека PyQt5 включает модуль QtWebEngine, позволяющий встраивать веб-контент в разрабатываемое приложение);

- возможность визуального проектирование с Qt Designer (Qt Designer позволяет создавать интерфейсы с помощью графического интерфейса, а затем экспортировать их в код PyQt5);

- возможность работы с базами данных (библиотека PyQt5 предоставляет модуль Qt SQL, который позволяет работать с базами данных прямо из разработанного приложения);

- расширяемость и интеграция: PyQt5 предоставляет возможность расширения функциональности фреймворка, создавая собственные пользовательские виджеты и компоненты с помощью языка Python.

Python часто используется в качестве дополнительного языка разработчиками программного обеспечения для контроля и управления сборкой, тестирования и многих других целей:

- SCons – управление сборками программных проектов;

- Buildbot и Apache Gump - для автоматической непрерывной компиляции и тестирования;

- Roundup или Trac для отслеживания ошибок и управления проектами.

2.4.2 Выбор среды разработки

В качестве среды для разработки ПО была выбрана PyCharm. PyCharm представляет собой кросплатформенную интегрированную среду для разработки приложений на языке программирования Python и других языках [6]. Данная среда была разработана компанией JetBrains на основе InteliJ IDEA. PyCharm предоставляет пользователю комплекс средств, позволяющих сделать процессы написания и отладки кода более удобными, а сам процесс разработки максимально продуктивным.

В целом, если говорить об упрощении процесса написания кода, то можно выделить следующие возможности редактора PyCharm:

- помощь при редактировании (оперативное обнаружение ошибок, автодополнение, быстрые исправления);

- подсветка синтаксиса (в PyCharm код читается легко благодаря возможностям настройки цветов подсветки синтаксиса Python и шаблонов Django);

- автоматическая расстановка отступов и форматирование (отступы добавляются автоматически в начале новой строки; проверка корректности отступов и автоматическое переформатирование производятся в соответствии с настройками стиля кода проекта);

- настраиваемый стиль кода;

- автодополнение (PyCharm предлагает варианты автодополнения для ключевых слов, классов и переменных);

- выделение фрагментов кода и комментарии;

- форматирование кода;

- сниппеты (ускорение процесса разработки за счёт использования настраиваемых шаблонов фрагментов кода (live templates) и сниппетов);

- сворачивание кода (сворачивание блоков кода, автоматическая расстановка скобок и кавычек, подсветка парных скобок и т. д.);

- подсветка ошибок на лету;

- многокурсорность (в режиме многокурсорного редактирования имеется возможность внесения изменений в файл сразу в нескольких местах);

- анализ кода (многочисленные инспекции проверяют код прямо в режиме редактирования, а также позволяют проанализировать весь проект на наличие ошибок и проблем в структуре кода);

- быстрые исправления (для большинства инспекций доступны быстрые исправления, которые позволяют откорректировать код мгновенно);

- удобная навигация (функция Search everywhere (рисунок 13) даёт возможность быстро найти любой класс, метод или файл, используемый в проекте, а операция Go to class / file / symbol позволяет осуществить быстрый переход к данным элементам);

- осуществление безопасных и быстрых рефакторингов (быстрая реорганизация кода с применением методов Extract, Delete, Rename и т.д., учитывающих особенности того языка, на котором ведётся разработка).

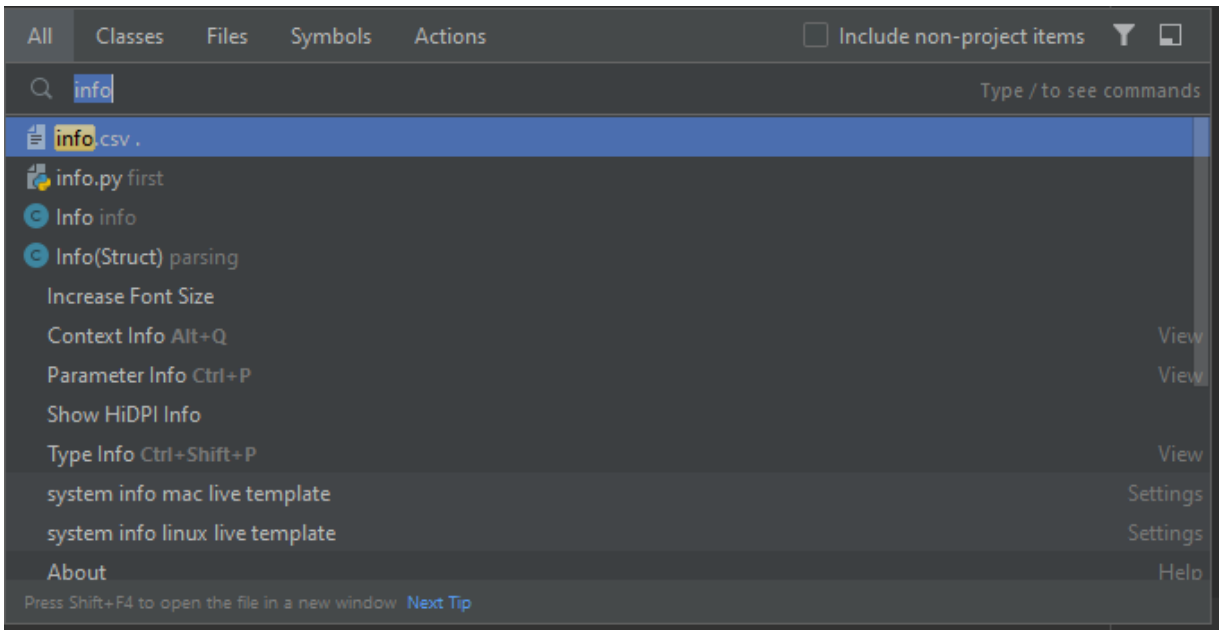


Рисунок 13 – Окно функции Search everywhere

Кроме того, PyCharm предоставляет широкий набор встроенных инструментов для разработчиков:

- визуальный отладчик кода (возможность запускать тесты и просматривать результаты в удобном графическом интерфейсе);

- инструменты для работы с базами данных (возможность осуществлять редактирование SQL-кода и самих баз данных, находящихся под управлением таких систем, как MySQL, Oracle, SQL Server и т.д.);

- инструменты для осуществления контроля версий (поддержка таких систем контроля версий, как Git (рисунок 14), Mercurial, SVN и т.д., позволяет управлять локальными изменениями в проекте).

Стоит также отметить, что PyCharm предоставляет широкие возможности в плане веб-разработки:

- поддержка различных веб-фреймворков (Flask, Pyramid, Google App Engine, Django и т.д.) и платформ для разработки;

- поддержка таких языков, как JavaScript, HTML, CSS, Node.js, TypeScript, Coffee Script и т.д.;

- функция Live Editing Preview позволяет осуществлять одновременное открытие редактора для оперативного отслеживания внесённых в код изменений на веб-странице.

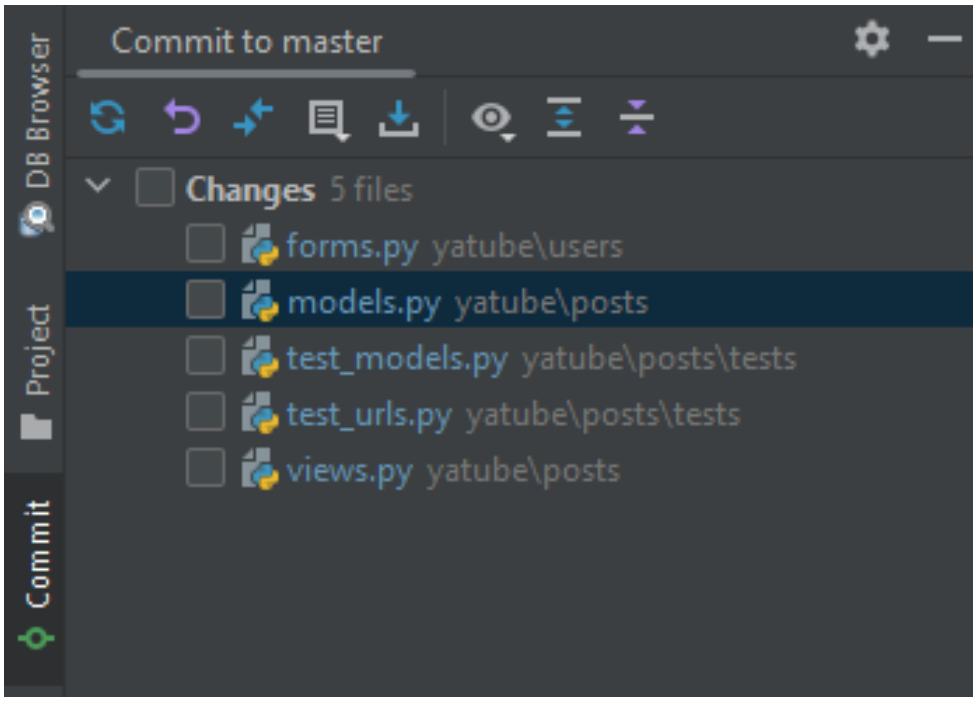


Рисунок 14 – Работа с Git

Также большой интерес представляют инструменты и возможности, предоставляемые PyCharm, для выполнения научных вычислений, в частности проведение визуализации и анализа данных:

- поддержка таких библиотек, как Matplotlib (визуализация данных), Numpy (поддержка больших многомерных массивов и матриц, а также доступ к высокоуровневым математическим функциям), Pandas (обработка и анализ данных);

- интерактивная консоль REPL (позволяет осуществлять автодополнение, проверку синтаксиса в режиме реального времени, сопоставление скобок и кавычек и т.д.;

- интеграция с Conda и Jupyter Notebook.

На рисунке 15 показано окно создания нового проекта. В левой части окна можно увидеть список различных фреймворков, доступных для выбора.

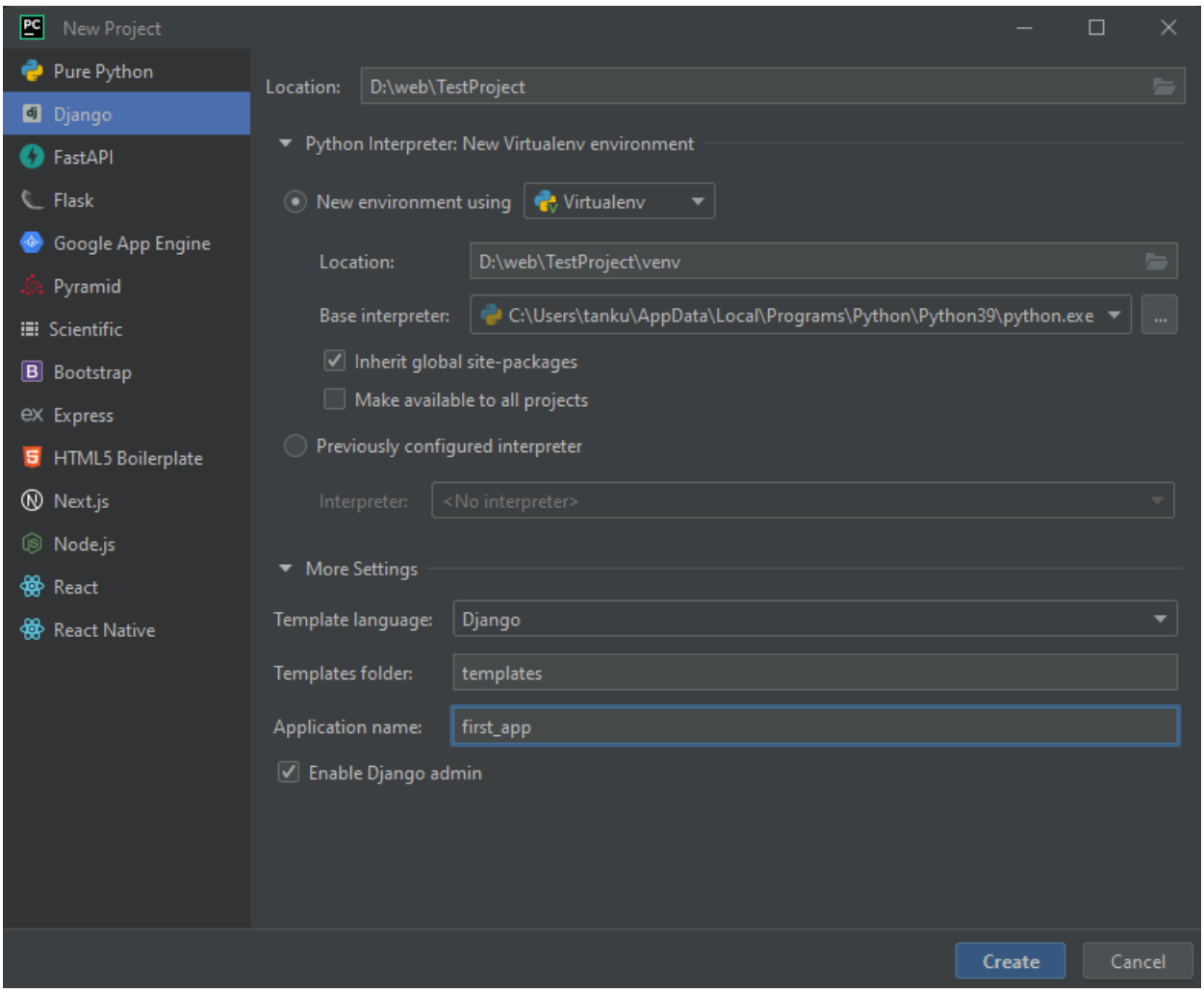


Рисунок 15 – Окно создания проекта

На рисунке 16 показано окно настройки виртуального окружения. Виртуальное окружение представляет собой изолированное окружение, создаваемое для каждого отдельного проекта и позволяющее изолированно устанавливать необходимые каждому приложению версии тех или иных программных библиотек. В разделе Python Interpreter можно настроить интерпретатор проекта и легко установить в виртуальное окружение проекта нужные версии библиотек.

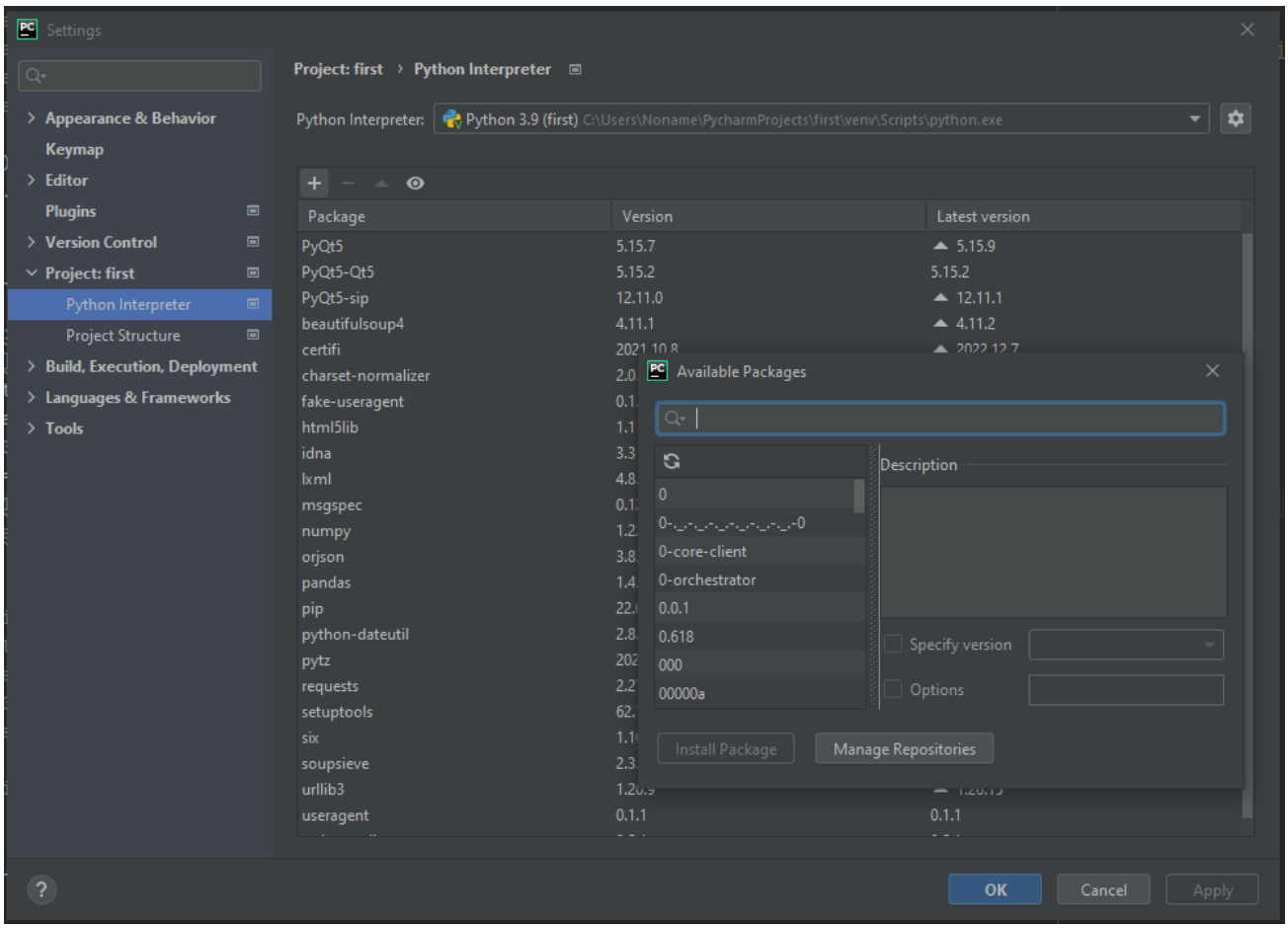


Рисунок 16 – Настройка виртуального окружения проекта

Также стоит отметить, что среда разработки Pycharm выпускается в двух редакциях: PyCharm Community Edition (бесплатная) и PyCharm Professional Edition (платная). Различие данных редакций состоит в предлагаемом функционале: PyCharm Professional Edition имеет более широкий набор функций. Выбор редакции для разработки зависит от специфики разрабатываемого продукта. Поскольку в случае разрабатываемой системы достаточно функционала PyCharm Community Edition, то использоваться будет именно эта редакция PyCharm. Полный список различий между двумя редакциями представлен в таблице 1

Таблица 1 – Сравнение двух различных редакций PyCharm

	PyCharm Professional Edition	PyCharm Community Edition
Функциональный редактор Python	+	+
Инструмент запуска тестов и графический отладчик	+	+
Навигация по коду и рефакторинги	+	+
Инспекции кода	+	+
Поддержка систем контроля версий	+	+
Инструменты для научных вычислений	+	
Веб-разработка	+	
Веб-фреймворки Python	+	
Python-профилировщик	+	
Возможности удаленной разработки	+	
Поддержка баз данных и SQL	+	

2.4.3 Выбор формата хранения данных

Так как основной функцией разрабатываемого программного обеспечения является сбор статистических данных по заболеваемости COVID-19 и дальнейшее их использование для обучения модели машинного обучения, необходимо организовать хранение получаемых данных для того, чтобы обеспечить возможность их повторного использования без постоянного выполнения выгрузок.

Выбор формата хранения данных зависит от размера данных, требований к производительности, безопасности, масштабируемости и других факторов. Среди различных форматов хранения можно выделить следующие одни из наиболее популярных [10]:

- базы данных (например, SQLite, MySQL, PostgreSQL);

- текстовые файлы (например, CSV или TSV);

- JSON (JavaScript Object Notation) – формат обмена данными, основанный на синтаксисе объектов JavaScript;

- XML (eXtensible Markup Language) – расширяемый язык разметки, который используется для хранения и передачи структурированных данных.

Использование баз данных целесообразно в случае, если данные имеют большой объем, сложную структуру и требуют сложных запросов. Базы данных обеспечивают эффективное хранение и доступ к данным, а также поддерживают множество возможностей для обработки и анализа данных.

Хранение данных в текстовых форматах актуально в случаях, если данные имеют простую структуру и существует необходимость выполнения простых операций над ними.

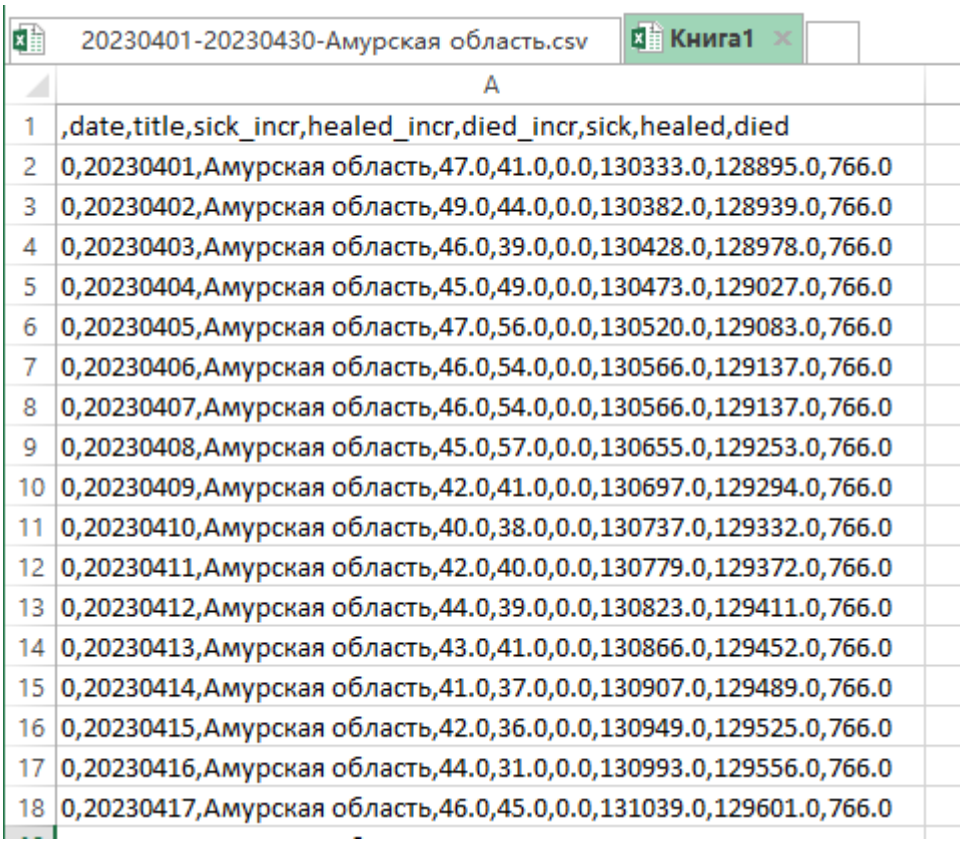
В случае, если данные имеют иерархическую или вложенную структуру, формат JSON может быть удобен для их хранения. JSON обеспечивает гибкость и легкость чтения/записи данных. Возможно сохранять данные в виде JSON-объектов или массивов в текстовом файле.

Использование формата XML актуально в случае, если данные имеют сложную структуру с вложенными элементами и атрибутами. XML-файлы могут быть созданы и обработаны с использованием различных инструментов и библиотек.

Поскольку собираемые данные не имеют сложной структуры и отсутствует необходимость выполнения сложных запросов к ним, то целесообразно использовать один из текстовых форматов. Поэтому в качестве формата хранения данных был выбран CSV (Comma-Separated Values) — текстовый фор-

мат, используемый для представления табличных данных. В CSV данные организованы в виде таблицы, где каждая строка представляет собой отдельную запись, а значения разделяются запятыми или другим разделителем. Каждое поле может содержать текст, числа или другие данные, и оно не обязательно должно быть одного типа. Обычно в CSV данные не содержат информацию о типе данных, поэтому при чтении данных из CSV-файла важно правильно интерпретировать значения. Формат CSV широко используется для обмена данными между различными программами, такими как электронные таблицы (например, Microsoft Excel, Google Sheets), базы данных, статистические пакеты и другие инструменты анализа данных. CSV-файлы легко создавать и читать с помощью текстовых редакторов или специализированных программных библиотек во многих языках программирования.

В случае решаемой задачи данные, получаемые после каждой выгрузки, формируются в одну таблицу, содержащую такие столбцы, как «Дата», «Регион», «Количество заражённых» и т.д. Пример файла CSV представлен на рисунке 17.


20230401-20230430-Амурская область.csv | Книга1

	A
1	,date,title,sick_incr,healed_incr,died_incr,sick,healed,died
2	0,20230401,Амурская область,47.0,41.0,0.0,130333.0,128895.0,766.0
3	0,20230402,Амурская область,49.0,44.0,0.0,130382.0,128939.0,766.0
4	0,20230403,Амурская область,46.0,39.0,0.0,130428.0,128978.0,766.0
5	0,20230404,Амурская область,45.0,49.0,0.0,130473.0,129027.0,766.0
6	0,20230405,Амурская область,47.0,56.0,0.0,130520.0,129083.0,766.0
7	0,20230406,Амурская область,46.0,54.0,0.0,130566.0,129137.0,766.0
8	0,20230407,Амурская область,46.0,54.0,0.0,130566.0,129137.0,766.0
9	0,20230408,Амурская область,45.0,57.0,0.0,130655.0,129253.0,766.0
10	0,20230409,Амурская область,42.0,41.0,0.0,130697.0,129294.0,766.0
11	0,20230410,Амурская область,40.0,38.0,0.0,130737.0,129332.0,766.0
12	0,20230411,Амурская область,42.0,40.0,0.0,130779.0,129372.0,766.0
13	0,20230412,Амурская область,44.0,39.0,0.0,130823.0,129411.0,766.0
14	0,20230413,Амурская область,43.0,41.0,0.0,130866.0,129452.0,766.0
15	0,20230414,Амурская область,41.0,37.0,0.0,130907.0,129489.0,766.0
16	0,20230415,Амурская область,42.0,36.0,0.0,130949.0,129525.0,766.0
17	0,20230416,Амурская область,44.0,31.0,0.0,130993.0,129556.0,766.0
18	0,20230417,Амурская область,46.0,45.0,0.0,131039.0,129601.0,766.0

Рисунок 17 – Содержимое CSV-файла со статистикой по Амурской области

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

3.1 Функциональное моделирование

Для того, чтобы более наглядно отразить процесс функционирования разработанной системы, было проведено построение функциональной модели с использованием нотации IDEF0. На рисунке 18 показана контекстная модель системы.

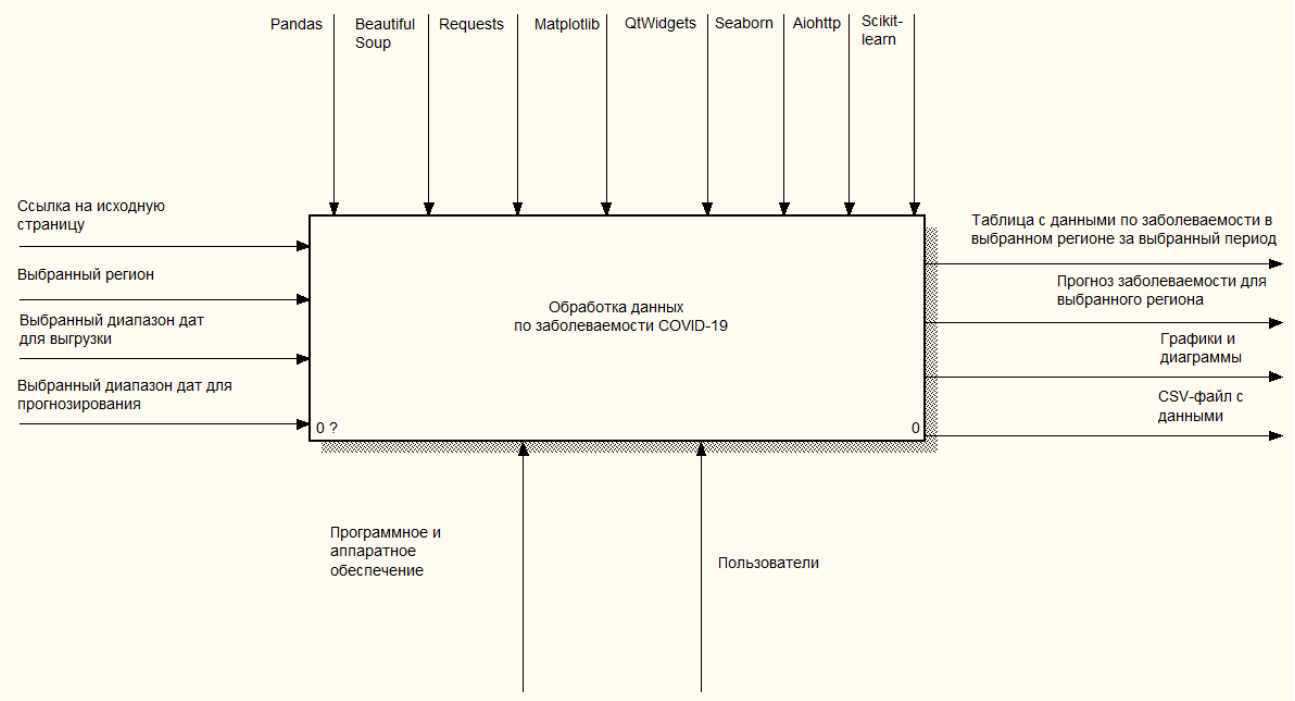


Рисунок 18 – Контекстная модель системы

Работа программы определяется набором библиотек и модулей языка Python:

- Requests и Aiohttp: выполнение HTTP-запросов;

- BeautifulSoup: анализ HTML и XML документов и извлечение из них данных;

- Pandas: сбор, обработка и анализ данных;

- Matplotlib и Seaborn: визуализация данных;

- QtWidgets: элементы интерфейса;

- Scikit-learn: машинное обучение.

Механизмами программы являются программное и аппаратное обеспечение и пользователи.

Для более подробного рассмотрения функций системы была выполнена декомпозиция контекстной модели (рисунок 19):

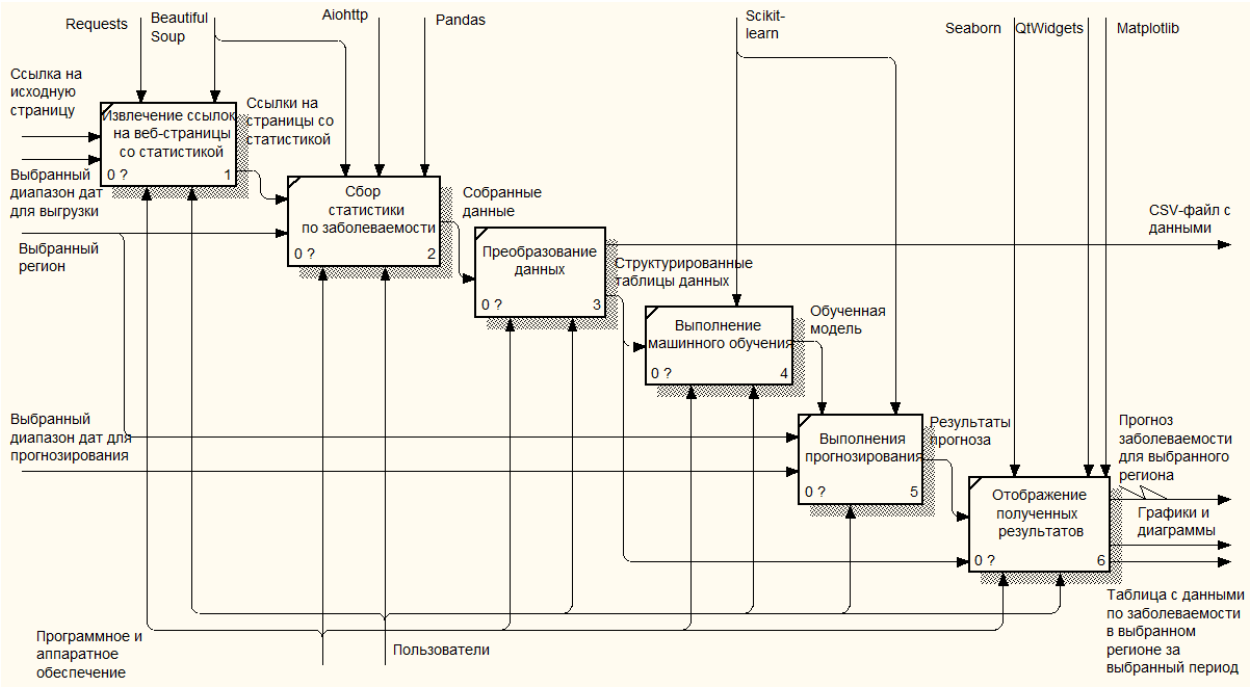


Рисунок 19 – Декомпозиция контекстной модели системы

Первым функциональным блоком на представленной выше декомпозиции контекстной модели является «Извлечение ссылок на веб-страницы со статистикой». На вход он получает ссылку на исходную страницу сайта, а также диапазон дат, за который нужно произвести выгрузку. На выходе же формируется список ссылок на веб-страницы со статистикой по заболеваемости.

Полученный список ссылок передаётся на вход вместе с наименованием выбранного региона, по которому нужно получить данные, блоку сбора статистики по заболеваемости. В результате проведения сбора данных по полученным ссылкам формируется набор данных по заболеваемости в выбранном регионе за выбранный период.

Сформированный набор данных передаётся блоку преобразования данных, в котором выполняются все необходимые операции по упорядочиванию

и форматированию. Результатом работы блока является структурированная таблица данных.

Полученная таблица данных может быть передана на вход блоку машинного обучения, если необходимо провести обучение и выполнить прогнозирование, либо блоку отображения полученных результатов, если необходимо просто проанализировать уже имеющиеся данные. В случае, если данные передаются на вход блоку машинного обучения, то будет произведено их разбиение на обучающую и тестовую выборки и выполнено обучение. Результатом работы блока является обученная модель.

Обученная модель машинного обучения регламентирует работу блока выполнения прогнозирования, который получает на вход диапазон дат для прогнозирования и наименование выбранного региона, для которого нужно провести данную операцию. На выходе блока формируется результат прогноза.

Блок отображения полученных результатов может принимать на вход таблицу с выгруженной статистикой либо результаты прогноза для выполнения их визуального отображения в виде таблиц, графиков или диаграмм.

3.2 Реализация алгоритма сбора данных

Ранее был рассмотрен алгоритм сбора данных (рисунок 20). Первым шагом алгоритма является получение списка ссылок на веб-страницы, хранящие статистику по заболеваемости COVID-19 за необходимый промежуток времени. Получить данный список возможно, проведя обработку HTML-кода исходной страницы. Для получения HTML-кода страницы необходимо отправить GET-запрос. Чтобы выполнить эту операцию, необходимо подобрать соответствующую библиотеку Python, которая позволила бы отправлять запросы к серверу. Для выполнения запросов к исходной странице была выбрана рассмотренная в предыдущей главе Requests, которая позволяет легко пересылать HTTP/1.1 запросы. Основные возможности, предоставляемые данной библиотекой:

- создание и отправка GET и POST запросов к серверу;

- возможность передачи параметров в URL (Requests позволяет передавать данные в строке запроса URL, добавляя нужные аргументы к строке в виде словаря);

- возможность прочитать содержимое ответа от сервера в различных форматах (Requests позволяет прочитать содержимое ответа от сервера в виде набора байтов, в формате JSON или в необработанном виде);

- добавление пользовательских заголовков в запрос (Requests предоставляет возможность добавления HTTP-заголовков в запрос путём передачи их в виде словаря в соответствующем параметре запроса).

Создание GET-запроса осуществляется с помощью функции get, обязательным аргументом которой является url запроса. В качестве необязательных параметров данной функции могут быть переданы:

- params представляет собой список кортежей, байтов или словарь для отправки в качестве параметров запроса (по умолчанию принимает значение None);

- allow_redirects – параметр функции с типом Boolean, используемый для определения того, разрешено ли выполнять переадресацию запроса (по умолчанию принимает значение True (означает, что переадресация запроса разрешена);

- auth представляет собой кортеж, передаётся для осуществления включения определенной HTTP-аутентификации (по умолчанию принимает значение None);

- cert – строка или кортеж для указания файла сертификата или ключа, используемых для проверки целостности и подлинности (по умолчанию принимает значение None);

- cookies представляет собой словарь, используемый для пересылки файлов куки на указанный URL-адрес (по умолчанию принимает значение None);

- headers – словарь, содержащий HTTP-заголовки и используемый для пересылки данных заголовков на указанный URL-адрес (по умолчанию принимает значение None);

Поскольку извлечение данных по заболеваемости будет выполняться путем анализа HTML-кода веб-страниц с таблицами со статистикой, то в случае, если будет необходимо получить данные за большой период времени, отправка GET-запросов к каждой странице с помощью библиотеки Requests может занять достаточно долгое время. Поэтому для отправки запросов к страницам со статистикой эффективнее будет использовать асинхронный HTTP-клиент/сервер, т.к. это позволит быстрее обмениваться данными с сервером. Реализовать данную возможность на языке Python позволяет Aiohttp – библиотека, предназначенная для создания сетевых приложений с использованием асинхронного программирования, которое позволяет эффективно обрабатывать большое количество одновременных запросов и подключений. Aiohttp основана на фреймворке Asyncio, встроенном в стандартную библиотеку Python, который обеспечивает асинхронное выполнение кода. Можно выделить следующие основные особенности данной библиотеки:

- асинхронный клиентский HTTP-запрос: Aiohttp позволяет выполнять асинхронные HTTP-запросы (методы get(), post(), put(), delete() и др.);

- работа с WebSocket: Aiohttp обеспечивает поддержку WebSocket-подключений, позволяя создавать асинхронные веб-сокеты, отправлять и принимать сообщения, устанавливать обработчики событий и контролировать состояние подключения;

- работа с cookie: библиотека предоставляет инструменты для работы с HTTP-cookie, позволяющие отправлять и получать cookie, устанавливать параметры cookie (срок действия, домен, путь) и выполнять другие операции с ними;

- обработка заголовков: предоставляется возможность управлять заголовками HTTP-запросов и ответов, включая добавление, изменение и удаление заголовков, а также чтение полученных заголовков ответа.

- многопоточность: поддержка работы с несколькими потоками выполнения, что позволяет эффективно обрабатывать множество одновременных запросов и подключений;

- асинхронный сервер: помимо клиентской части, Aiohttp также предлагает возможность создания асинхронных HTTP-серверов;

- поддержка прокси: Aiohttp позволяет отправлять HTTP-запросы через прокси-серверы, устанавливая соответствующие параметры.

После получения HTML-кода страницы необходимо произвести извлечение нужной информации, хранимой в HTML-тегах. Во время описания алгоритма было отмечено, что первым интересующим тегом будет являться тег , имеющий класс «news-element». Для выполнения операции извлечения данных из полученного HTML-кода необходимо также использовать соответствующую библиотеку языка Python, позволяющую осуществлять сбор данных с веб-страниц. Одной из самых популярных таких библиотек является BeautifulSoup, используемая для извлечения данных из HTML и XML файлов. Она предоставляет удобный способ парсинга и навигации по структуре веб-страницы, позволяя легко извлекать интересующие вас данные. BeautifulSoup позволяет разбирать HTML и XML документы и создавать древовидное представление структуры документа, которое можно просматривать и модифицировать с помощью простых методов и синтаксиса Python. Это упрощает работу с веб-страницами и позволяет извлекать нужные данные, такие как заголовки, ссылки, таблицы или другие элементы. Одна из основных особенностей BeautifulSoup - это её способность обрабатывать «грязный» или неправильно сформированный HTML-код, который может содержать ошибки и несоответствия стандартам.

Перед использованием BeautifulSoup необходимо выбрать тип парсера, который будет применяться. Существует четыре основных библиотеки для парсинга HTML, поддерживаемые BeautifulSoup [17]:

- html.parser – это стандартный парсер HTML в Python, который выполняет разбор HTML-кода и создает древовидное представление структуры документа;

- lxml – это парсер, используемый для разбора HTML и XML документов и основанный на библиотеке Lxml, которая является быстрой и эффективной библиотекой для обработки XML и HTML;

- html5lib – Python-реализация парсера HTML5, имеющая совместимость со стандартами HTML5.

В таблице 2 представлены достоинства и недостатки всех вышеперечисленных парсеров.

Таблица 2 – Сравнение парсеров

Парсер	Достоинства	Недостатки
html.parser	– не требует дополнительной установки; – приемлемая скорость; – нестрогий	– не настолько быстр, как lxml; – более строгий, чем html5lib
lxml – HTML парсер	– очень быстрый; – нестрогий	– внешняя зависимость от языка C
lxml – XML парсер	– очень быстрый; – единственный на данный момент поддерживаемый XML парсер	– внешняя зависимость от языка C
html5lib	– максимально нестрогий; – разбирает веб-страницы так же, как это делает браузер; – создаёт валидный HTML5	– очень медленный; – внешняя зависимость от языка Python

Основным критерием при выборе парсера в случае решаемой задачи является скорость. Поэтому в качестве парсера для BeautifulSoup будет использоваться lxml. Эта библиотека имеет специализированный модуль lxml-xml, который фокусируется только на обработке XML документов. Несмотря на это данный модуль всё-таки может быть использован для обработки кода HTML, но условием корректности такой обработки является отсутствие в анализируемом фрагменте синтаксических особенностей, не поддерживаемых XML. В рассматриваемом фрагменте такие особенности отсутствуют, поэтому его обработка с помощью lxml-xml производится корректно, и, так как по результатам тестов lxml-xml выполняет данную задачу быстрее, чем обычный lxml, то использоваться будет именно он.

Использование BeautifulSoup начинается с создания объекта одноименного класса. В качестве атрибутов данному объекту передаются HTML-код страницы, полученный ранее с помощью функции get, и наименование парсера (рисунок 20).

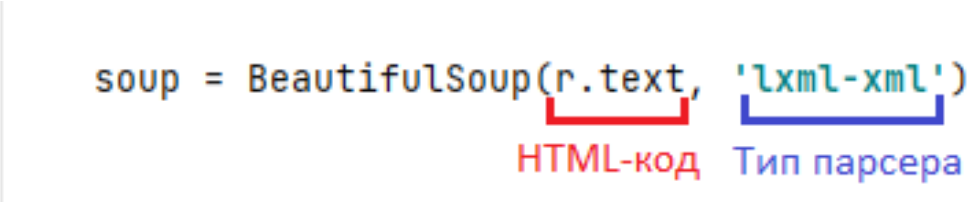


Рисунок 20 – Создание объекта класса BeautifulSoup

Для поиска элементов по тегам в BeautifulSoup используются методы find() и find_all(). Метод find() возвращает только первый найденный элемент, соответствующий заданным критериям, а метод find_all() – список всех найденных элементов, соответствующих заданным критериям. Оба метода в качестве аргументов принимают:

- тег элемента, который нужно найти;

- атрибуты элемента или словарь атрибутов, по которым осуществляется поиск;

- дополнительные параметры, такие как класс CSS, регулярные выражения и т. д.

На рисунке 21 показан пример использования метода find_all() для получения всех элементов класса «news-element».

```
blocks = soup.find_all(attrs={'class': 'news-element'})
```

Название класса в качестве атрибута

Рисунок 21 – Использование функции find_all() для поиска всех элементов класса «news-element»

После получения данных элементов, необходимо извлечь из вложенных в них тегов значения дат и ссылки на страницы со статистикой по заболеваемости. Значение даты хранится в теге , а ссылка – как значение атрибута href (гипертекстовая ссылка) тега <a>. Поиск обоих тегов в каждом блоке «news-element» производится с помощью метода find() аналогичным образом, как и с родительским тегом. Для извлечения текста ссылки отдельно указывается название атрибута (рисунок 22)

```
url_list[date] = block.find("a")['href']
```

Искомый тег Искомый атрибут

Рисунок 22 – Извлечение значения атрибута href

После того, как на текущей странице были извлечены все имеющиеся ссылки на страницы со статистикой, в случае, если необходимы данные за более поздний период, чем последние 15 дней, выполняется обработка следующей страницы. Получение ссылки на следующую страницу выполняется путём извлечения значения атрибута href тега <a> с классом «next page-numbers» так же с использованием метода find(). Обработка всех последующих страниц осуществляется аналогичным способом, описанным ранее.

Следующим этапом является извлечение таблиц с данными по заболеваемости со всех веб-страниц, ссылки на которые были получены до этого. Вначале, как и в случае с исходными страницами, необходимо получить HTML-код страницы путём отправки GET-запроса, используя функцию get библиотеки requests. После получения кода начинается его непосредственная обработка. На рассматриваемых веб-страницах необходимые данные хранятся в виде HTML-таблиц, обозначаемых тегом <table>. Извлечение данных таблиц будет осуществлять с помощью библиотеки Pandas - высокоуровневой библиотека, построенная на основе более низкоуровневой библиотеки NumPy, позволяющая осуществлять анализ различных массивов данных. В целом, можно выделить следующие основные операции, которые могут быть проведены с данными при использовании Pandas:

- чтение и запись (поддержка самых популярных форматов хранения данных, таких как: csv, sql, html, буфер обмена, excel и т.д.);

- создание сводных таблиц (сводные таблицы являются достаточно эффективным инструментом обработки данных, позволяющих обобщить полученные данные и выделить из них только ту информацию, которая требуется в данный момент);

- проведение анализа временных рядов (Pandas предоставляет достаточно удобный инструментарий для выполнения анализа временных рядов, который включает в себя, например, инструмент под названием Resampling, позволяющий переформировать выборку так, как удобно пользователю);

- группировка и агрегирование данных (Pandas имеет встроенный метод groupby, позволяющий провести группировку данных по какому-либо признаку, и метод agg, используемый для агрегирования данных);

- визуализация данных (с помощью библиотеки matplotlib может быть проведён визуальный анализ данных путём построения графиков различных зависимостей между ними).

Главными и наиболее важными объектами Pandas являются Series и DataFrame. Объект Series можно представить как одномерный массива типа ndarray с наличием ассоциированных меток (индексов) у каждого его элемента. DataFrame, в свою очередь, представляет собой двумерную табличную изменяемую по размеру структуру потенциально разнородных табличных данных. Столбцами данной структуры являются объекты типа Series, строки которых и выступают непосредственными элементами DataFrame. Структура объекта DataFrame показана на рисунке 23.

Рисунок 23 – Структура объекта DataFrame

Использование Pandas для извлечения HTML-таблиц обусловлено наличием у данной библиотеки метода read_html(). Данный метод предназначен для чтения таблиц данных из веб-страниц в формате HTML и преобразования их в объект класса DataFrame. Метод read_html автоматически идентифицирует и извлекает все таблицы на странице и возвращает список объектов DataFrame, содержащий эти таблицы [18]. Данный метод имеет несколько параметров, которые можно использовать для настройки чтения таблицы, например:

- header: указывает, какой ряд в таблице использовать в качестве заголовков столбцов;

- flavor: указывает синтаксис парсинга HTML;

- attrs: указывает атрибуты таблицы для фильтрации.

Метод read_html удобен для быстрого чтения таблиц данных из HTML-страницы, особенно если необходимо извлечь данные из веб-таблицы и дальше анализировать их с помощью pandas.

Полученная таблица содержит данные по всем регионам Российской Федерации за тот или иной день. Для начала в таблицу нужно добавить столбец, содержащий значение даты, используя метод insert(). После этого в таблице необходимо найти строку, содержащую информацию по выбранному региону и извлечь необходимые столбцы с данными для этого региона (рисунок 24)

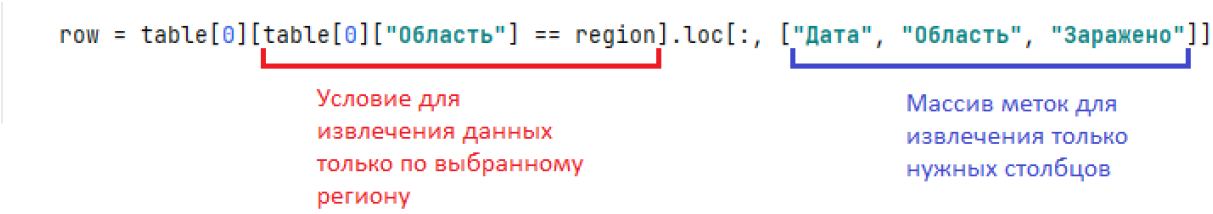


Рисунок 24 – Извлечение строки

Далее полученная строка добавляется к общей DataFrame-таблице с помощью метода concat(), используемого для объединения или конкатенации нескольких объектов DataFrame вдоль определенной оси (оси столбцов или оси строк) (рисунок 25).

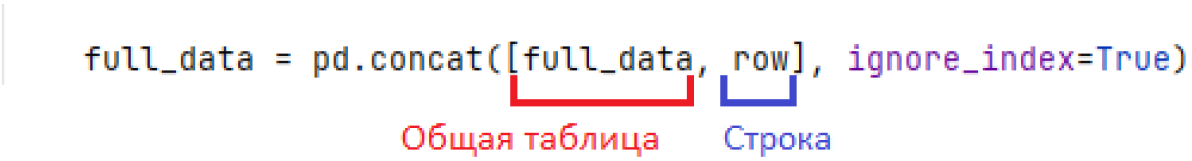


Рисунок 25 – Объединение строки с общей таблицей

Формат общего объекта DataFrame представлен на рисунке 26:

```
full_data = pd.DataFrame(
        {
            'Дата': [],
            'Область': [],
            'Заражено': []
        }
    )
```

Рисунок 26 – Формат общего DataFrame

Таким образом, с каждой страницы со статистикой из списка происходит извлечение одной строки, содержащей данные для выбранного региона за один определённый день, и все строки объединяются в общую таблицу, содержащую динамику заболеваемости COVID-19 в определённом регионе за определённый период времени.

3.3 Реализация алгоритма машинного обучения

Перед началом разработки модели машинного обучения необходимо выбрать тип обучения, который будет использоваться. Ранее были описаны три базовых таких типа:

- обучение с учителем;
- обучение без учителя;
- обучение с подкреплением.

В таблице 3 представлены достоинства и недостатки каждого из типов.

Таблица 3 – Сравнение типов машинного обучения

Тип машинного обучения	Достоинства	Недостатки
1	2	3
С учителем	- имеется доступ к размеченным данным; - можно использовать широкий спектр алгоритмов классификации и регрессии; - позволяет получать интерпретируемые результаты.	- требуется большой объем размеченных данных; - модель может оказаться чувствительной к качеству исходных данных или ошибкам в разметке; - не способно обнаруживать новые паттерны или информацию, которая не присутствует в обучающих данных.
Без учителя	- позволяет обрабатывать большие объемы неразмеченных данных, которые часто доступны в большом количестве; - может помочь обнаружить скрытые структуры, паттерны или группы в данных; - может использоваться для предобработки данных и снижения размерности.	- отсутствие явных меток; - интерпретация результатов может быть сложной; - обучение без учителя может потребовать большего объема вычислительных ресурсов и времени для обработки данных.

Продолжение таблицы 3

1	2	3
С подкреплением	- может использоваться для решения сложных задач, требующих последовательности действий и принятия решений в динамической среде; - позволяет агенту самостоятельно итеративно улучшать свое поведение, основываясь на обратной связи в виде награды или штрафа; может применяться в обучении роботов, играх и оптимизации управления системами.	- может потребовать большого количества времени для достижения хороших результатов; - необходимы хорошо определенные правила награды и штрафы, чтобы гарантировать, что агент будет обучаться желательному поведению; - проблема исследования-использования (exploration-exploitation) может быть сложной, когда агенту требуется исследовать новые действия, чтобы обнаружить оптимальную стратегию.

Поскольку в результате сбора формируется достаточный объём размеченных данных, и для решения поставленной задачи необходимы хорошо интерпретируемые результаты, то использоваться будет машинное обучение с учителем. Этот выбор также обусловлен тем, что данный вид обучения хорошо подходит для решения задач классификации и регрессии. В случае решаемой задачи имеет место быть именно задача регрессии: модель должна предсказать непрерывное числовое значение для данного входного примера.

Далее необходимо определить, какой алгоритм обучения будет использоваться. Машинное обучение с учителем использует различные методы и алгоритмы для обучения моделей на основе размеченных данных. Вот некоторые из наиболее распространенных методов:

- линейная регрессия: метод, используемый для решения задач регрессии путём построения линейной функции, которая наилучшим образом соответствует зависимости между входными и выходными данными;

- логистическая регрессия: метод, применяемый для решения бинарных или многоклассовых задач классификации и использующий логистическую функцию для предсказания вероятности принадлежности объекта к определенному классу;

- решающие деревья: метод, основанный на построении дерева решений, где каждый узел представляет условие на одном из признаков, а каждая ветвь соответствует возможному значению этого признака;

- метод опорных векторов (SVM): метод, который ищет оптимальную гиперплоскость, разделяющую объекты разных классов с наибольшим зазором между ними;

- наивный Байесовский классификатор: вероятностный метод, основанный на применении теоремы Байеса с предположением о независимости признаков, он строит модель, предсказывающую класс объекта на основе вероятностей его признаков;

- K-ближайших соседей (KNN): метод, основанный на идее, что близкие объекты имеют похожие выходные значения, KNN классифицирует объекты, основываясь на классе их ближайших соседей в пространстве признаков;

- градиентный бустинг: метод, который строит ансамбль слабых моделей (обычно деревьев) последовательно, где каждая следующая модель пытается исправить ошибки предыдущей модели;

- случайный лес: метод, который объединяет несколько решающих деревьев в ансамбль, каждое дерево обучается на случайной подвыборке данных,

а предсказание получается путем усреднения (в случае регрессии) или голосования (в случае классификации) результатов деревьев.

Для решения поставленной задачи был выбран алгоритм линейной регрессии. Для реализации данного алгоритма необходимо использовать одну из библиотек Python, предназначенную для проведения машинного обучения. Среди наиболее популярных таких библиотек можно выделить следующие:

- Scikit-learn;
- TensorFlow;
- PyTorch;
- Keras.

Scikit-learn (sklearn)

Данная библиотека предоставляет широкий набор алгоритмов, методов и инструментов для выполнения машинного обучения. Она позволяет осуществлять выбор и оценку моделей, подготавливать данные для обучения, а также проводить другие различные операции, связанные с процессом машинного обучения. Scikit-learn предлагает большой набор классов для реализации всех описанных выше моделей машинного обучения, а также многих других. Библиотека также предоставляет удобный и единообразный API для использования всех этих моделей.

TensorFlow

Данная библиотека, разработанная Google, предназначена для выполнения глубокого обучения. Она предоставляет широкий набор инструментов для работы с моделями глубокого обучения, такими как: нейронные сети, сверточные нейронные сети, рекуррентные нейронные сети и т. д. Одним из главных достоинств данной библиотеки является возможность контролировать каждую деталь модели и оптимизировать ее производительность

PyTorch

Данная библиотека, разработанная командой искусственного интеллекта Facebook, предоставляет гибкую и интуитивно понятную платформу для

создания и обучения моделей глубокого обучения. PyTorch основана на динамическом вычислении, что позволяет более гибко определять модели и легко выполнять операции над тензорами. Также она обладает удобным отладочным интерфейсом и обширной поддержкой сообщества. PyTorch активно используется в академической сфере и ведущими исследовательскими лабораториями для разработки и исследований в области глубокого обучения.

Keras

Данная библиотека предоставляет простой и интуитивно понятный API для создания и обучения моделей глубокого обучения, позволяя быстро прототипировать и тестировать различные архитектуры нейронных сетей. Она также обладает богатой функциональностью, включая поддержку множественных входов/выходов, автоматическое выравнивание размерности, а также встроенные функции для обработки изображений и текста.

Для реализации модели линейной регрессии была выбрана библиотека Scikit-learn. Она имеет модуль sklearn.linear_model, непосредственно используемый для создания различных моделей линейной регрессии. Данный модуль содержит класс LinearRegression, представляющей модель обычной линейной регрессии с одним или несколькими признаками. Основными методам класса являются:

- fit() – обучение модели;
- predict() – предсказание значений на основе обученной модели;
- score() – оценка точности модели.

Согласно алгоритму подготовки модели машинного обучения, представленному ранее на рисунке 10, собранные данные для применения их в процессе обучения должны быть разделены на обучающий и тестовый наборы. Для выполнения этой операции существует специальный функция модуля sklearn.model_selection библиотеки Scikit-learn под названием train_test_split(). Данная функция используется для разделения массива данных случайным образом на тестовый и обучающий наборы с заданным соотношением между

наборами. На рисунке 27 показан пример разбиения сформированного в результате сбора данных объекта DataFrame.

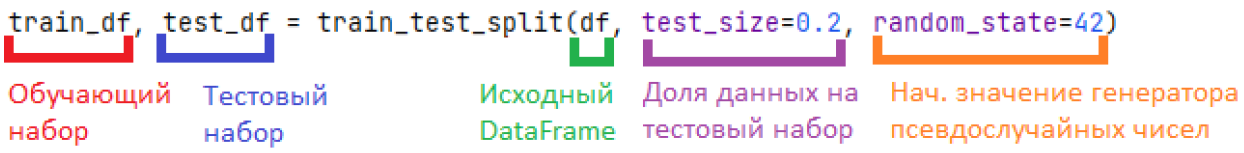


Рисунок 27 – Разбиения объекта DataFrame на обучающую и тестовую выборки

После того, как сформированы обучающий и тестовый наборы можно переходить непосредственно к самому обучению. Для начала должен быть создан объект класса LinearRegression(), соответствующий модели линейной регрессии. Обучение производится с помощью упомянутого ранее метода fit(), которому в качестве параметра передаётся обучающий набор данных. После этого работа модели проверяется на тестовых данных путём выполнения предсказания с помощью метода predict(). Далее производится оценка модели с помощью метода score(). Пример использования всех методов показан на рисунке 28.

Рисунок 28 – Работа с моделью машинного обучения

3.4 Выполнение тестирования программного обеспечения

После запуска программы пользователь может выбрать в меню необходимую операцию для выполнения. Выбрав пункт «Анализ данных», пользователь получит вкладку, на которой можно просмотреть статистику по заболеваемости для выбранного региона за выбранный период времени. Данные могут быть получены путём выполнения выгрузки с сайта либо путём открытия уже существующего файла. В случае, если данные необходимо выгрузить, пользователь должен выбрать дату начала и дату окончания выгрузки, а также регион. Если также поставить галочку напротив пункта «Сохранить в CSV файл», то будет открыто диалоговое окно для сохранения данных в формате CSV. Полученные данные отображаются в таблице снизу, а также в виде графиков и диаграмм в правой части окна. На рисунке показан пример отображения статистики по заболеваемости для Амурской области с 01.01.2023 по 30.01.2023.

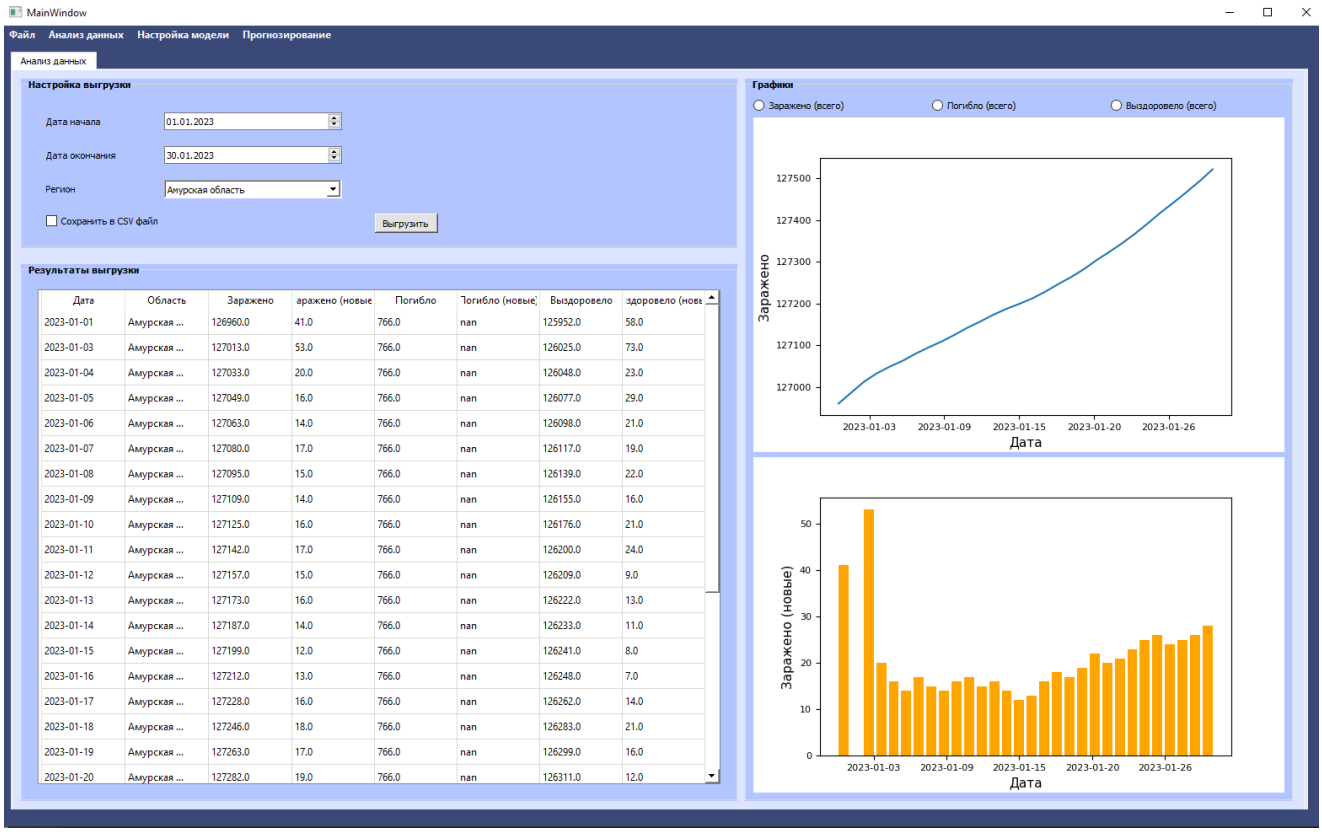


Рисунок 29 – Результаты выгрузки данных

Для выполнения прогнозирования необходимо выбрать соответствующий пункт меню. После этого будет открыта вкладка во многом аналогичная вкладке, представленной на предыдущем рисунке, и отличающаяся лишь тем, что вместо выбора начальной и конечной даты пользователю предлагается выбрать, на какое количество дней необходимо сформировать прогноз: 7, 14 или 30. Результаты прогноза будут представлены в виде таблицы и графиков и их так же можно выгрузить в файл формата CSV. На рисунке показан пример отображения результатов прогнозирования для Амурской области на 7 дней.

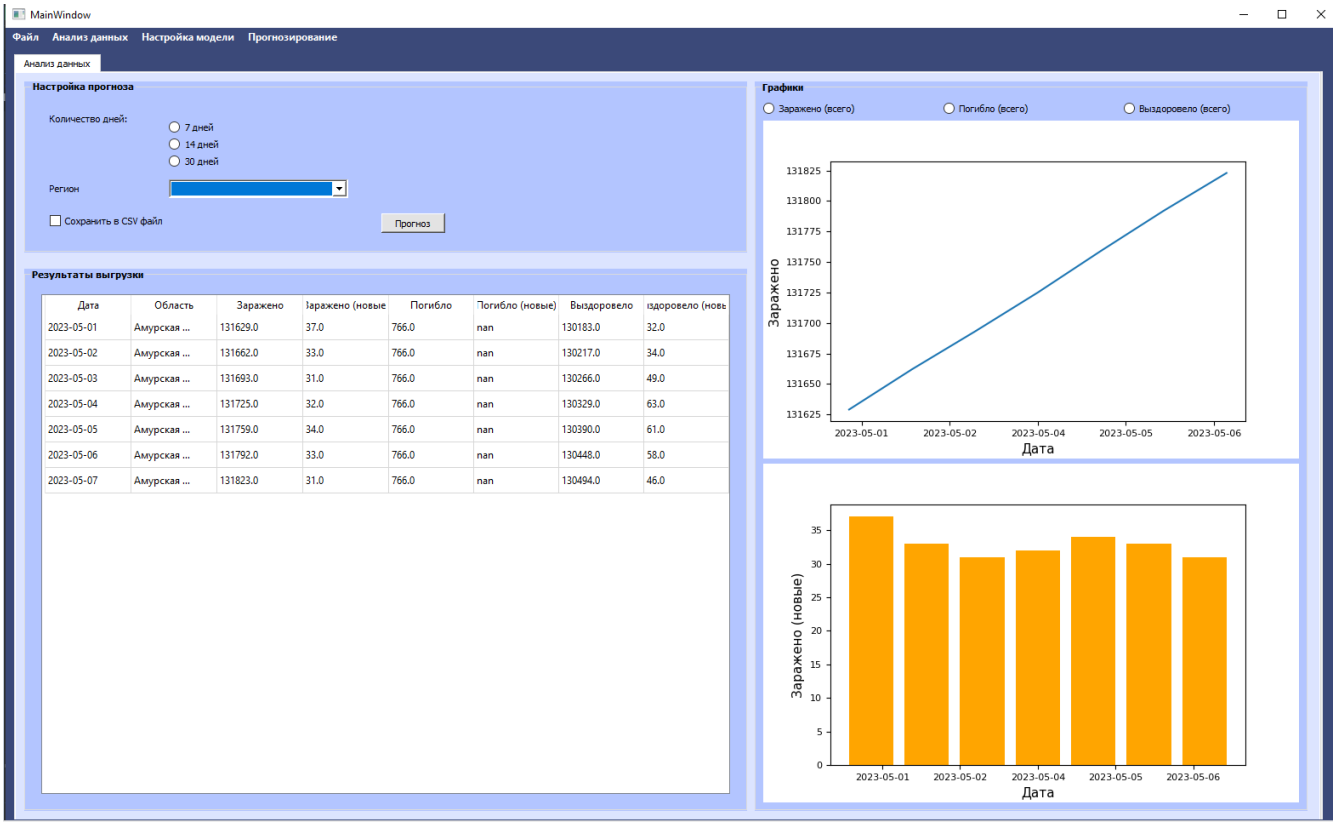


Рисунок 30 – Результаты прогноза

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было разработано программное обеспечение, позволяющее осуществлять сбор статистики по заболеваемости COVID-19 в регионах Российской Федерации и выполнять прогнозирование будущей заболеваемости.

На начальных этапах работы была проведена характеристика предметной области исследования, включая рассмотрение основных теоретических сведения о больших данных, возможностей их использования, а также существующих типов машинного обучения. Также были рассмотрены примеры уже существующих решений задачи

Далее был сформирован алгоритм сбора данных по заболеваемости и проведена его реализация с помощью языка программирования Python и его библиотек, таких как: BeautifulSoup, Pandas, Requests и т.д. Также был сформирован алгоритм подготовки модели машинного обучения, выбран тип машинного обучения и алгоритм обучения, а также определены средства программной реализации: библиотека Scikit-learn и её модули.

Полученные результаты исследования могут быть использованы в области санитарно-эпидемиологического контроля для разработки систем прогнозирования заболеваемости в периоды возможных будущих эпидемий тех или иных вирусных заболеваний. На основе разработанной системы прогнозирования COVID-19 могут быть созданы аналогичные системы, позволяющие осуществлять прогнозирование других вирусных заболеваний. Для этого необходимо лишь переработать существующую систему под сбор данных по другим заболеванием с учётом их специфики и специфики ресурса данных. Применённые методы и алгоритмы являются по сути универсальными, что создаёт возможность их использования в будущем в других подобных системах.

По результатам выполненной работы было опубликовано 2 статьи в сборниках материалов научных конференций.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Бейдер, Дэн Чистый Python. Тонкости программирования для профи / Д. Бейдер. – СПб. : Питер, 2018. – 288 с.

2 Большие данные (Big Data) [Электронный ресурс]. – Режим доступа: https://itelon.ru/solution/bolshie-dannye-big-data/ – 23.01.2023

3 Вандер Плас, Дж. Python для сложных задач: наука о данных и машинное обучение / Дж. Вандер Плас. – СПб. : Питер, 2018. – 576 с.

4 Введение в pandas: анализ данных на Python [Электронный ресурс]. – Режим доступа: https://khashtamov.com/ru/pandas-introduction/ – 01.11.2022

5 Возможности информационных систем в прогнозировании исходов новой коронавирусной инфекции COVID-19 [Электронный ресурс]. – Режим доступа: https://remedium.ru/doctor/pulmonology/vozmozhnosti-informatsionnykh- sistem-v-prognozirovanii-iskhodov-covid-19/ – 05.09.2022

6 Возможности PyCharm [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/features/ – 13.01.2023

7 Вьюгин, В. В. Математические основы машинного обучения и прогнозирования / В. В. Вьюгин. – М. : МЦНМО, 2015. – 387 с.

8 Грас, Джоэл Data Science. Наука о данных с нуля: пер. с англ. – 2-е изд., перераб. и доп. / Д. Грасс – СПб. : БХВ-Петербург, 2021. – 416 с.

9 Доусон, Майкл Программируем на Python / М. Доусон. – СПб. : Питер, 2022. – 416 с.

10 Какие бывают форматы файлов Big Data: row vs column [Электронный ресурс]. – Режим доступа: https://bigdataschool.ru/blog/row-vs-column-file-format-big-data.html – 11.09.2022

11 Копылов, Н. Ю. Использование технологии больших данных в области информационной безопасности / Н. Ю. Копылов // Молодежь XXI века: шаг в будущее : материалы конф. – Благовещенск: Изд-во БГПУ, 2021. – С. 764-765

12 Копылов, Н.Ю. Использование технологии больших данных в области санитарно-эпидемиологического контроля / Н.Ю. Копылов // Перспективные научные исследования: опыт, проблемы и перспективы развития : материалы конф. – Уфа: Изд-во НИЦ Вестник науки, 2023. – С. 176-179.

13 Копылов, Н.Ю. Разработка алгоритма автоматизированного сбора данных по заболеваемости COVID-19 / Н.Ю. Копылов // Актуальные проблемы науки и техники : материалы конф. – Уфа: Изд-во НИЦ Вестник науки, 2023. – С. 217-222.

14 Краткое руководство по библиотеке Python Requests [Электронный ресурс]. – Режим доступа: https://pythonru.com/biblioteki/kratkoe-rukovodstvo-po-biblioteke-python-requests – 12.04.2023

15 Лутц, Марк Изучаем Python, том 1 / М. Лутц. – 5-е изд., пер. с англ. – СПб. : ООО “Диалектика”, 2019. – 832 с.

16 Любанович, Билл Простой Python. Современный стиль программирования / Б. Любанович. – СПб. : Питер, 2021. – 592 с.

17 Модуль BeautifulSoup4 в Python, разбор HTML [Электронный ресурс]. – Режим доступа: https://docs-python.ru/packages/paket-beautifulsoup4-python/ – 05.12.2022

18 Модуль pandas [Электронный ресурс]. – Режим доступа: https://academy.yandex.ru/handbook/python/article/modul-pandas – 11.02.2023

19 Мюллер Андреас Введение в машинное обучение с помощью PYTHON / А. Мюллер, С. Гвидо. – М. : Вильямс, 2022. – 480 с.

20 Обучение с учителем (Supervised learning) [Электронный ресурс]. – Режим доступа: https://wiki.loginom.ru/articles/supervised-learning.html – 10.01.2023

21 Постолит, А. В. Основы искусственного интеллекта в примерах на Python. Самоучитель / А. В. Постолит. – СПб. : БХВ-Петербург, 2021. – 448 с.

22 Руководство по Qt Designer [Электронный ресурс]. – Режим доступа: http://doc.crossplatform.ru/qt/4.5.0/designer-manual.html – 01.05.2023

23 Силен, Дэви Основы Data Science и Big Data. Python и наука о данных / Д. Силен, А. Мейсман, М. Али. – СПб. : Питер, 2017. – 336 с.

24 Фрэнкс, Билл Укрощение больших данных: как извлекать знания из массивов информации с помощью глубокой аналитики / Б. Фрэнкс: пер. с англ. Андрея Баранова. – М. : Манн, Иванов и Фербер, 2018. – 352 с.

25 Что такое Scikit Learn - гайд по популярной библиотеке Python для начинающих [Электронный ресурс]. – Режим доступа: https://datastart.ru/blog/read/chto-takoe-scikit-learn-gayd-po-populyarnoy-biblioteke-python-dlya-nachinayuschih – 10.11.2022

26 Шолле, Франсуа Глубокое обучение на Python. 2-е межд. издание / Ф. Шолле. – СПб. : Питер, 2023. – 576 с.

27 Actually forecasting COVID-19 [Электронный ресурс]. – Режим доступа: https://towardsdatascience.com/how-to-actually-forecast-covid-19-778cce27b9d6 – 08.10.2022

28 An introduction to seaborn [Электронный ресурс]. – Режим доступа: https://seaborn.pydata.org/tutorial/introduction.html – 01.03.2023

29 Beautiful Soup: Build a Web Scraper With Python [Электронный ресурс]. – Режим доступа: https://realpython.com/beautiful-soup-web-scraper-python/ – 05.02.2023

30 Beautiful Soup Documentation [Электронный ресурс]. – Режим доступа: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ – 02.12.2022

31 Big Data: что такое большие данные и где они применяются [Электронный ресурс]. – Режим доступа: https://selectel.ru/blog/what-is-big-data/ – 10.09.2022

32 Big Data Analytics in COVID-19 [Электронный ресурс]. – Режим доступа: https://encyclopedia.pub/entry/9412 – 19.12.2022

33 Big Data, google, and infectious disease prediction: a statistical perspective [Электронный ресурс]. – Режим доступа: https://statistics.wharton.upenn.edu/research/seminars-conferences/previous-seminars/spring-2019/big-data-google-and-infectious-disease-prediction-a-statistical-perspective/ – 02.02.2023

34 Data science approaches to infectious disease surveillance [Электронный ресурс]. – Режим доступа: https://royalsocietypublishing.org/doi/10.1098/rsta.2021.0115 – 15.11.2022

35 Forecasting COVID-19 with predictive analytics Big Data tools [Электронный ресурс]. – Режим доступа: https://healthitanalytics.com/news/forecasting-covid-19-with-predictive-analytics-big-data-tools – 05.11.2022

36 Intelligent Data Analysis for Infection Spread Prediction [Электронный ресурс]. – Режим доступа: https://www.mdpi.com/2071-1050/14/4/1995 – 11.02.2023

37 Learn Python PyQt [Электронный ресурс]. – Режим доступа: https://pythonpyqt.com/ – 01.04.2023

38 Linear Models [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/modules/linear_model.html – 12.11.2022

39 Matplotlib: Научная графика в Python [Электронный ресурс]. – Режим доступа: https://pythonworld.ru/novosti-mira-python/scientific-graphics-in-python.html – 20.02.2023

40 Matplotlib: Visualization with Python [Электронный ресурс]. – Режим доступа: https://matplotlib.org/ – 15.02.2023

41 NumPy, часть 1: начало работы [Электронный ресурс]. – Режим доступа: https://pythonworld.ru/numpy/1.html – 14.04.2023

42 NumPy Introduction [Электронный ресурс]. – Режим доступа: https://www.w3schools.com/python/numpy/numpy_intro.asp – 14.04.2023

43 NumPy user guide [Электронный ресурс]. – Режим доступа: https://numpy.org/doc/stable/user/index.html#user – 11.04.2023

44 Pandas User Guide [Электронный ресурс]. – Режим доступа: https://pandas.pydata.org/docs/user_guide/index.html – 05.11.2022

45 PyCharm Community. Основы работы [Электронный ресурс]. – Режим доступа: https://younglinux.info/python/pycharm – 11.01.2023

46 Python GUI: создаём простое приложение с PyQt и Qt Designer [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/python-gui-pyqt/ – 10.04.2023

47 Qt Designer Manual [Электронный ресурс]. – Режим доступа: https://doc.qt.io/qt-6/qtdesigner-manual.html – 15.04.2023

48 Requests: HTTP for Humans [Электронный ресурс]. – Режим доступа: https://requests.readthedocs.io/en/latest/ – 02.12.2022

49 Seaborn для визуализации данных в Python [Электронный ресурс]. – Режим доступа: https://pythonru.com/biblioteki/seaborn-plot – 05.03.2023

50 Sources of Big Data: Where does it come from? [Электронный ресурс]. – Режим доступа: https://www.upgrad.com/blog/sources-of-big-data/ – 01.02.2023

51 Testing big data in a big crisis: Nowcasting under Covid-19 [Электронный ресурс] – Режим доступа: https://www.sciencedirect.com/science/article/pii/S0169207022001431 – 04.02.2022

52 What are the 5 V's of Big Data? [Электронный ресурс]. – Режим доступа: https://www.teradata.com/Glossary/What-are-the-5-V-s-of-Big-Data#:~:text=Big%20data%20is%20a%20collection,variety%2C%20velocity%2C%20and%20veracity – 23.01.2023