МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2023 г.

Разработка приложения для анализа географически-распределенных данных на платформе PySyft

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2023.308-278.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.,
доцент
_____ Г.И. Радченко

Автор работы,
студент группы КЭ-403
_____ В.А. Дегтярев

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2023 г.

Челябинск, 2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
06.02.2023 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-403
Дегтяреву Владимиру Андреевичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 25.04.2023 г. № 753-13/12) Разработка приложения для анализа географически-распределенных данных на платформе PySyft.
2. **Срок сдачи студентом законченной работы:** 05.06.2023 г.
3. **Исходные данные к работе**

3.1. Официальный сайт OpenMined. [Электронный ресурс] URL: https://www.openmined.org/ (дата обращения: 13.02.2023 г.).
3.2. Официальный сайт PySyft. [Электронный ресурс] URL: https://github.com/OpenMined/PySyft (дата обращения 13.02.2023 г.).
3.3. Официальный сайт TenserFlow Federated. [Электронный ресурс] URL: https://www.tensorflow.org/federated?hl=ru (дата обращения 13.02.2023 г.).
3.4. Ziller A., Trask A., Lopardo A., Szymkow B., Wagner B., Bluemke E., Nounahon J.-M., Passerat-Palmbach J., Plakash K., Rose N., Ryffel T., Reza Z.N., Kaissis G. PySyft. A Library for Easy Federated Learning. // Part of the Studies in Computation Intelligence book series, 2021. – pp. 111–139.

4. **Перечень подлежащих разработке вопросов**

4.1. Выполнить обзор литературы.
4.2. Выполнить анализ аналогичных проектов.
4.3. Определить функциональные и нефункциональные требования к системе.

4.4. Спроектировать систему для анализа географически-распределенных данных на платформе PySyft.
4.5. Реализовать систему для анализа географически-распределенных данных на платформе PySyft.
4.6. Провести тестирование системы для анализа географически-распределенных данных на платформе PySyft.
5. Дата выдачи задания: 06.02.2023 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н., доцент Г.И. Радченко

Задание принял к исполнению В.А. Дегтярев

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В настоящее время в мире существует огромное количество информации. А также большое количество людей, которые хотят использовать эту информацию в своих научных интересах, для нахождения закономерностей, обучения математических моделей, чтобы получать ответы на важные вопросы.

Машинное обучение и анализ данных уже активно применяется в медицине, финансах, промышленности [1]. Однако эти технологии еще не могут уверенно отвечать на некоторые глобальные и сложные вопросы из-за отсутствия доступа у разработчиков и ученых к большому количеству информации. Основные причины этого выражаются в виде защиты персональных данных, сохранения приватности конкретных данных, а также раздробленности этих данных среди огромного количества организаций.

Эти проблемы можно решить с помощью систем географически-распределенного и конфиденциального машинного анализа.

Постановка задачи

Целью выпускной квалификационной работы является реализация системы для анализа географически-распределенных данных на платформе PySyft. Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить обзор литературы;
2) выполнить анализ аналогичных проектов;
3) определить функциональные и нефункциональные требования к системе;
4) спроектировать систему для анализа географически-распределенных данных на платформе PySyft;
5) реализовать систему для анализа географически-распределенных данных на платформе PySyft;

6) провести тестирование системы для анализа географически-распределенных данных на платформе PySyft.

Структура и содержание работы

Работа состоит из введения, пяти глав, заключения, списка литературы и приложения. Объем работы составляет 35 страниц, объем списка литературы – 20 источников.

В первой главе описываются предметная область и аналогичные проекты.

Вторая глава посвящена определению функциональных и нефункциональных требований к системе и проектированию ее архитектуры.

Третья глава содержит в себе подробности и особенности реализации системы для анализа географически-распределенных данных на платформе PySyft.

В четвертой главе описывается практическое применение системы решением задачи по анализу данных.

В пятой главе описывается процесс тестирования работы системы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

Целью данной работы является разработка приложения для анализа географически-распределенных данных на платформе PySyft. Анализ данных представляет собой область математики и информатики, которая занимается построением и исследованием наиболее общих математических методов и вычислительных алгоритмов извлечения знаний из экспериментальных данных с целью получения полезной информации и принятия решений [2].

Существует множество методов анализа данных. Анализ статистических данных используется для изучения свойств распределения данных [3]. Классификация данных используется для разделения данных на группы на основе их характеристик. Многомерный анализ данных используется для изучения связей между различными переменными. Регрессионный анализ используется для определения связи между зависимой переменной и одной или несколькими независимыми переменными. Кластерный анализ используется для разделения данных на группы на основе их сходства [4].

Особенностью машинного анализа географически-распределенных данных является то, что информация, которую анализирует система, распределена между различными независимыми устройствами.

Для обеспечения географически-распределенного анализа и конфиденциальности данных современные решения используют такие методы, как федеративное обучение, дифференциальная приватность, гомоморфное шифрование [5].

Федеративное обучение представляет собой метод машинного обучения, который позволяет коллективно обучать алгоритм на нескольких устройствах без централизации всех исходных данных на одном сервере. Системы федеративного обучения совершенствуют единую общую модель. Источники данных никогда не перемещаются и не объединяются, но каждое

устройство вносит свой вклад в обучение и повышение качества общей модели [6].

Дифференциальная приватность – это область, изучающая методы, которые обеспечивают максимально точные результаты статистических запросов в базу данных при минимизации возможности идентификации отдельных записей в ней. Для каждого человека, чьи данные входят в анализируемый набор, дифференциальная приватность гарантирует, что результат анализа на дифференциальную приватность будет практически неотличим вне зависимости от того, есть ли данные этого конкретного человека в наборе или нет [7]. Дифференциальная приватность основана на введении случайности в данные. Чем больше случайности добавляется, тем сильнее сохраняется приватность, однако получаются более неточные результаты. Также на точность результатов влияет размер выборки, чем больше выборка, тем точнее результаты [8].

Гомоморфное шифрование – это форма шифрования, позволяющая производить определённые математические действия с зашифрованным текстом и получать зашифрованный результат, который соответствует результату операций, выполненных с открытым текстом [9]. Использование гомоморфного шифрования открывает множество перспектив при обработке конфиденциальных данных в среде, участники которой не доверяют друг другу. Оно позволяет осуществлять индексацию, фильтрацию спама, обработку платежей и другие действия без расшифровки самих сообщений и может применятся в облачных вычислениях, децентрализованных системах, электронном голосовании [10].

1.2. Анализ существующих платформ распределенного анализа данных

1.2.1. PySyft

В 2019 году была создана библиотека PySyft сообществом OpenMined. Это люди, объединенные темой конфиденциальности в машинном обучении. PySyft представляет собой обертку над PyTorch, Tensorflow или Keras для приватного машинного обучения [8].

Основная задача, стоящая перед сообществом OpenMined, заключалась в том, чтобы создать программное обеспечение, которое бы позволяла одному человеку получать ответы на свои вопросы, используя данные, принадлежащие другому человеку без необходимости просмотра и создания копии этих данных [11].

Проект предоставляет удаленный вызов процедур, что позволяет разработчику отправлять математическую модель к пользователям, где она локально обучается на их данных, после чего возвращается с обновленными весами обратно разработчику (рисунок 1). Данный процесс может происходить одновременно на разных устройствах, тем самым происходит параллельное обучение модели [8].

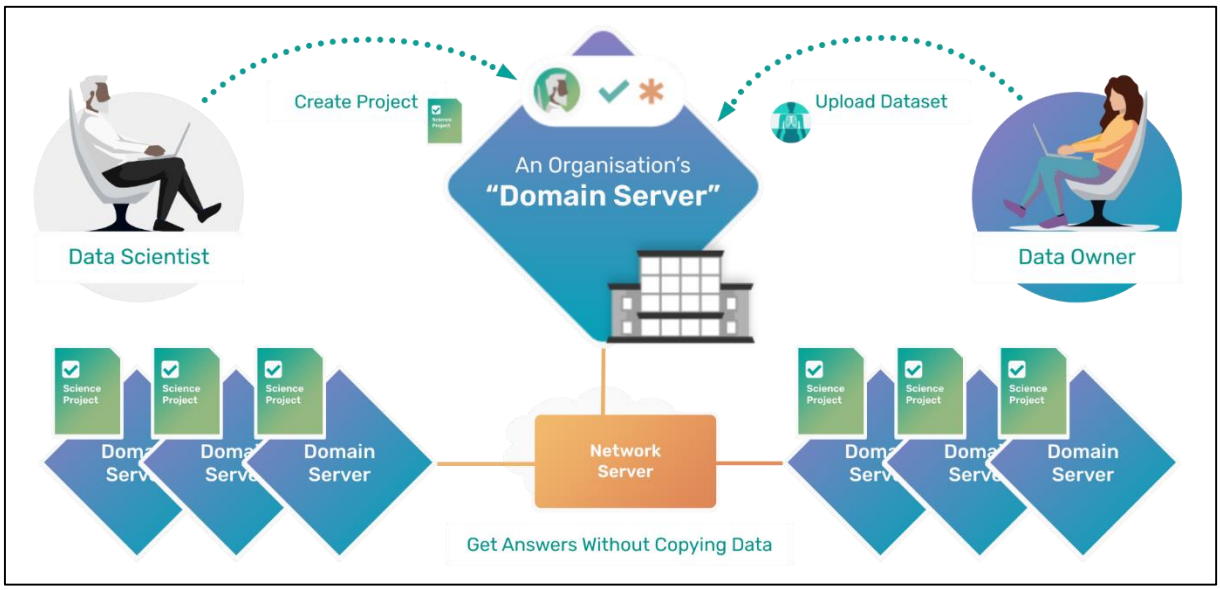

Рисунок 1 – Взаимодействие разработчика с пользователями [12]

Также ключевой особенностью PySyft является использование дифференциальной приватности [11]. По измененным весам модели нейронной сети можно догадаться, какие данные были у пользователя. Чтобы это предотвратить, к данным, которые хранятся на вычислительной машине пользователя, добавляется шум. Дифференциальная приватность представляет собой методы, которые описывают добавление шума.

1.2.2. TensorFlow Federated

Компания Google давно занимается сбором некоторой информации с устройств пользователей в единое защищенное хранилище, на котором тренируют свои нейросети. В 2017 году в Google Research был предложен инновационный подход под названием федеративное машинное обучение (рисунок 2). Он позволяет всем устройствам, которые участвуют в машинном обучении, делить на всех единую модель для прогнозирования, но при этом не делиться первичными данными для обучения модели. Система федеративного обучения работает по принципу совершенствования единой общей модели нейросети [13].

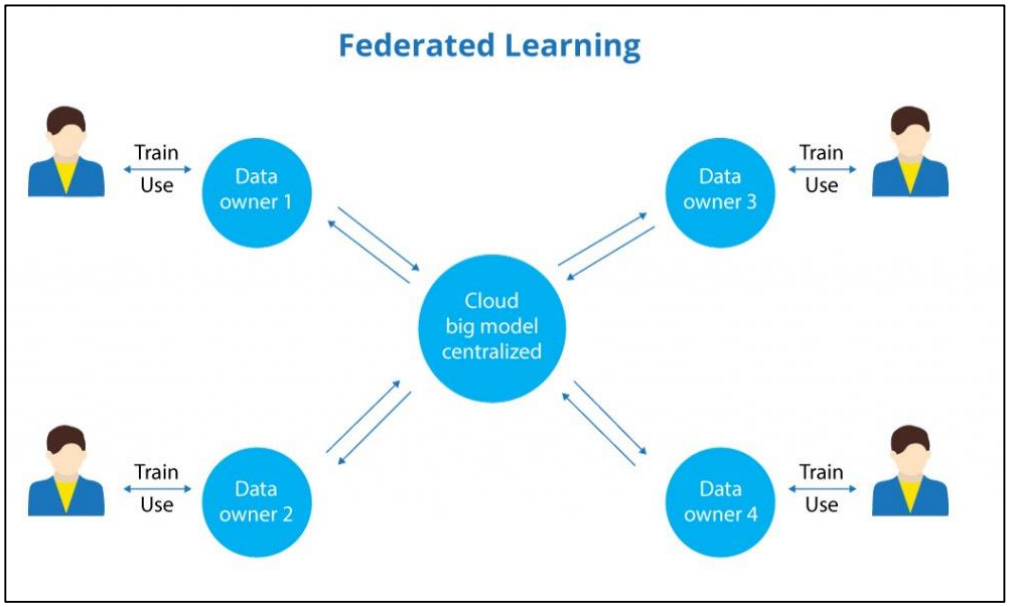

Рисунок 2 – Федеративное машинное обучение [14]

Для проверки системы федеративного обучения на больших объемах данных компанией Google был реализовал этот алгоритм в мобильном приложении клавиатуры Gboard [13]. Основной задачей являлось прогнозирование слов и выражений, которые пользователь предположительно использовал бы следующими во время печатания текста. Система федеративного обучения не отправляет текст, который печатает пользователь, на сервер компании Google, она отправляет На устройстве каждого пользователя производится анализ текста, который он использовал. Затем результаты анализа отправляются в компанию Google, где они будут объединены с другими результатами анализа для улучшения общей модели набора текста. Тем самым, каждый пользователь улучшает опыт использования Gboard каждому пользователю.

Компанией Google была создана платформа TensorFlow Federated с открытым исходным кодом для машинного обучения на децентрализованных данных. TensorFlow Federated был разработан для облегчения исследований и экспериментов с федеративным обучением [14].

Выводы по первой главе

В данной главе был проведен обзор предметной области и анализ существующих решений и проектов в области машинного анализа географически-распределенных данных.

2. ПРОЕКТИРОВАНИЕ

Целью данной работы является разработка системы для анализа географически-распределенных данных на платформе PySyft. Система представляет собой децентрализованное приложение, которое будет предоставлять одноранговую сеть для владельцев данных и аналитиков данных.

Владелец данных с помощью системы сможет создавать узлы внутри сети, загружать данные и управлять созданными узлами.

Аналитик данных с помощью приложения сможет подключаться к различным узлам внутри сети и проводить аналитические операции на основе данных, расположенным на этих узлах.

Логика работы проектируемой системы показана на рисунке 3.

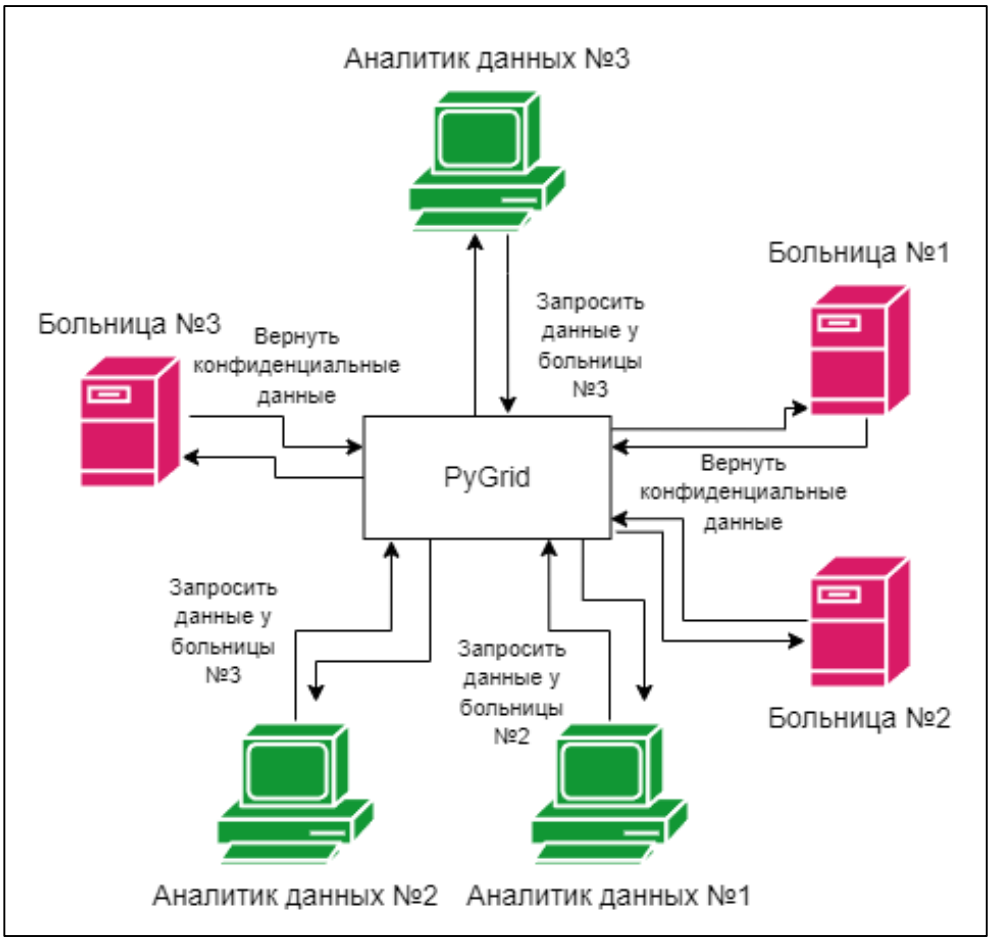

Рисунок 3 – Логика работы системы

В рамках данной работы будет создано по одному узлу с двух разных устройств, которые соединены между собой локальной сетью. На каждый узел будут загружены данные. По локальной сети к созданным узлам будет подключаться устройство, на котором будет происходить процесс получения данных с подключенных узлов и их анализа.

2.1. Функциональные требования к системе

Можно выделить следующий набор функциональных требований к системе.

1. Система должна предоставлять владельцу данных возможность запустить узел в одноранговой сети.
2. Система должна предоставлять владельцу данных возможность отключить созданный им узел в одноранговой сети.
3. Система должна предоставлять владельцу данных возможность загрузить данные на созданный им узел в одноранговой сети.
4. Система должна предоставлять аналитику данных возможность использовать данные, размещенные на любых других узлах одноранговой сети.

2.2. Нефункциональные требования к системе

Можно выделить следующие нефункциональные требования к системе.

1. Система должна образовывать целостную одноранговую сеть.
2. Система должна обеспечивать аналитику данных возможность получать данные из нескольких узлов одновременно.
3. Система должна быть написана на языке программирования Python.
4. Система должна быть разработана с использованием таких инструментов, как: PySyft, PyGrid, HaGrid.

2.3. Диаграмма вариантов использования системы

Для проектирования системы был использован язык графического описания для объектного моделирования UML.

На рисунке 4 представлена диаграмма вариантов использования.

Рисунок 4 – Диаграмма вариантов использования системы для анализа географически-распределенных данных

В системе определены следующие виды акторов.

1. *Владелец данных* – это пользователь приложения, который может запустить и отключить узел в одноранговой сети для размещения на него определенных данных.

2. *Аналитик данных* – пользователь приложения, который может использовать размещенные на узлах данные для аналитических операций.

Актору «Владелец данных» доступны следующие варианты использования системы.

1. Владелец данных может запустить узел для дальнейшего размещения на нем данных.

2. Владелец данных может отключить узел, после этого, все данные, которые были размещены на узле, станут недоступными для использования аналитиками данных.

3. Владелец данных может загрузить данные на созданный им узел, которые в дальнейшем будут аннотированы дифференциальной приватностью и станут доступны аналитикам данных.

4. Владелец данных может создать нового пользователя, который будет иметь доступ к созданному владельцем узлу.

Актор «Аналитик данных» может получить данные, которые размещены на узле.

2.4. Диаграмма компонентов

На рисунке 5 представлена диаграмма компонентов системы.

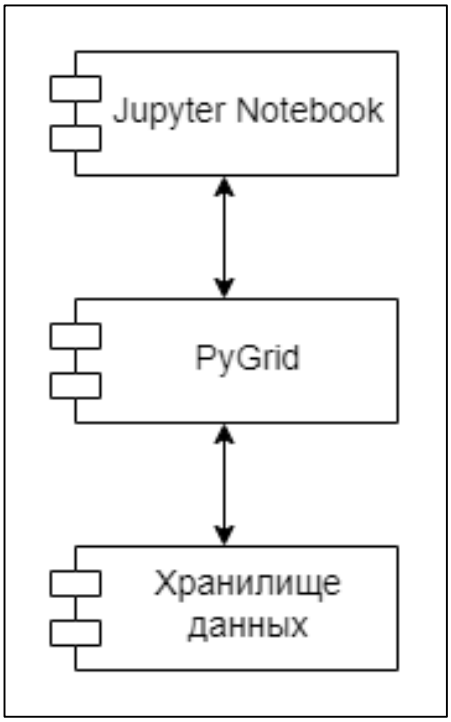

Рисунок 5 – Диаграмма компонентов системы

Система состоит из следующих компонентов.

1. Jupyter NoteBook – интерфейс, с которым взаимодействуют пользователи.

2. PyGrid – серверный компонент системы, который обрабатывает все запросы пользователей, взаимодействует с хранилищем данных узла.

3. Хранилище данных – компонент системы для хранения данных. Для каждого узла существует собственный компонент хранилища данных, в котором хранится информация об узле и загруженные владельцем данные.

2.5. Диаграммы деятельности

При проектировании системы были составлены диаграммы деятельности для прецедентов «Загрузить данные» и «Получить данные».

На рисунке 6 представлена диаграмма деятельности прецедента «Загрузить данные».

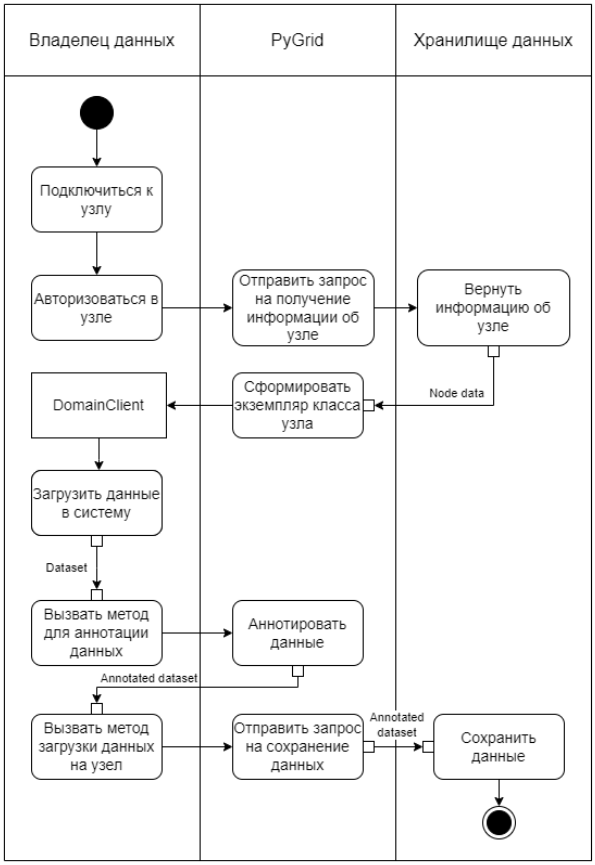

Рисунок 6 – Диаграмма деятельности прецедента

На рисунке 7 представлена диаграмма деятельности прецедента «Получить данные».

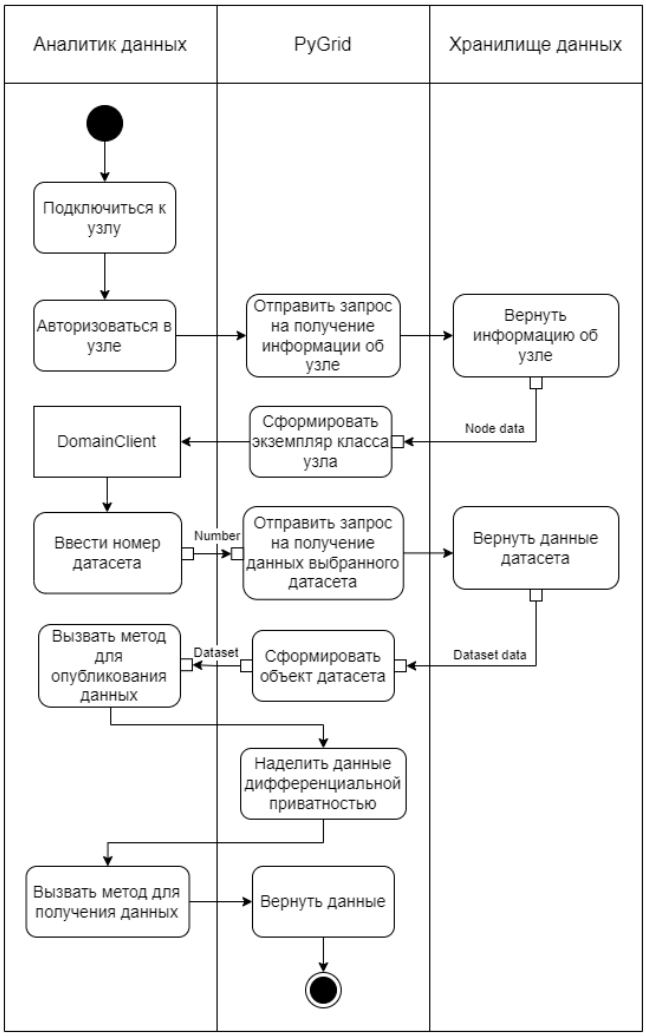

Рисунок 7 – Диаграмма деятельности прецедента

Выводы по второй главе

В данной главе были выделены функциональные и нефункциональные требования к системе, составлена диаграмма вариантов использования системы и разработаны диаграммы компонентов системы.

3. РЕАЛИЗАЦИЯ

Для реализации системы для анализа географически-распределенных данных на платформе PySyft был выбран высокоуровневый язык программирования Python 3 [15], библиотека PySyft [5], инструмент командной строки HaGrid [16].

Для преобразования и анализа данных были использованы библиотеки NumPy [17], pandas [18], scikit-learn [19].

3.1. Настройка окружения

Запуск узла домена владельца данных был произведен с помощью инструмента HaGrid. HaGrid – это инструмент командной строки, который ускоряет развертывание PyGrid, программного обеспечения, обеспечивающего одноранговую сеть владельцев данных и аналитиков данных, которые могут совместно анализировать данные и обучать модели с помощью PySyft [16]. Процесс запуска узла домена показан на рисунке 8.

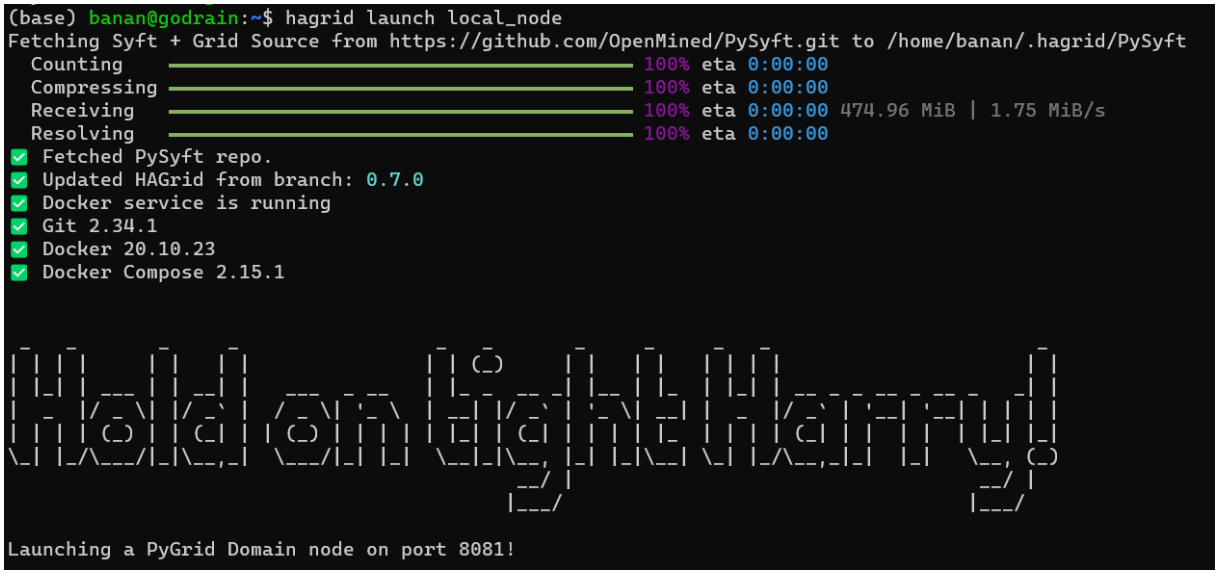

Рисунок 8 – Запуска узла домена владельца данных

Аналогичным образом был запущен узел на втором устройстве.

Узел домена занимает порт 8081. При развертывании узла домена создаются и запускаются «Docker» контейнеры. Данный процесс показан на рисунке 9.

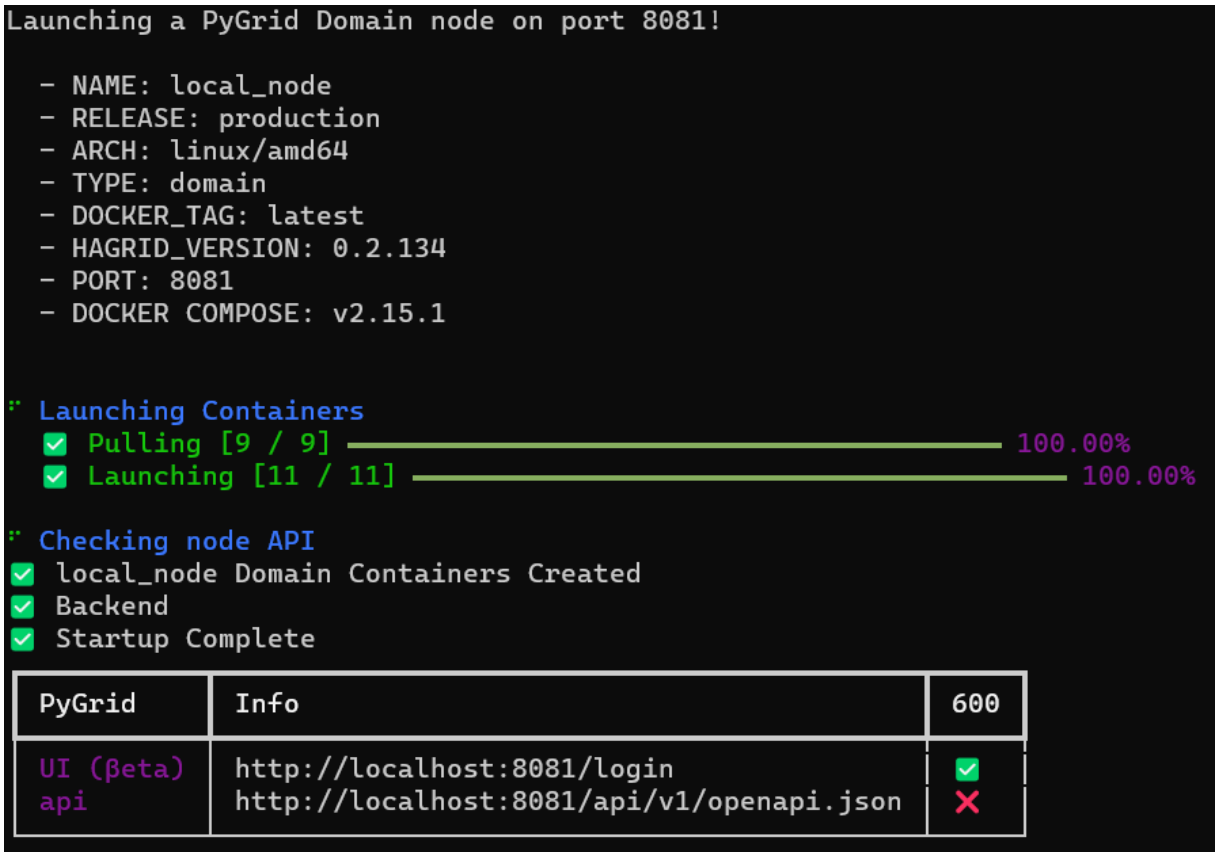

Рисунок 9 – Запуска узла домена владельца данных

3.3. Процесс загрузки данных на узел

Процесс подключения к узлу в качестве владельца данных продемонстрирован в листинге 1.

Листинг 1 – Подключение к узлу

```
try:
    domain_client = sy.login(
       port=8081,
       email="info@openmined.org",
       password="20052005"
    )
except Exception as e:
    print("Unable to login. Please check your domain is up with `!hagrid
check localhost:8081 --timeout=120`")
```

Исходные данные были обработаны. Был удален пустой столбец «Unnamed». Исходный набор данных был разделен на два набора данных, так как, для того, чтобы эти данные имели смысл, нужно наделить дифференциальной приватностью все столбцы, кроме столбца, обозначающего тип клетки.

Данные были аннотированы с дифференциальной приватностью с помощью метода `annotate_with_dp_metadata` класса `Tensor`. В параметрах данного метода были указаны нижняя и верхняя границы предельно допустимых значений для конкретного набора данных. Для каждого набора данных эти значения будут различаться, например, на основе всего набора данных, для данных о среднем расстоянии от центра до точек на периметре ядра клетки (`radius_mean`) нижней границей является 0, а верхней границей является 30.

Были выделены субъекты набора данных, то есть сущности, информация о которых не будет никому предоставляться. Для данного набора данных субъектом является идентификатор (`id`), то есть у любой строки таблицы будет отсутствовать идентификатор (`id`).

Процесс аннотации представлен в листинге 2.

Листинг 2 – Аннотирование данных

```
final_dataset = dict()

data_subjects = sy.DataSubjectArray.from_objs(dataset1["id"])

final_dataset["radius_mean"] = sy.Tensor(dataset1["radius_mean"]).anno-
tate_with_dp_metadata(
   lower_bound=0, upper_bound=30, data_subject=data_subjects
)
```

Аналогично были аннотированы все столбцы набора данных. Все аннотированные столбцы были добавлены в словарь `final_dataset`. Исходный код аннотации всего набора данных представлен в приложении.

Для загрузки данных на узел был использован метод `load_dataset` класса `DomainClient` (листинг 3). В параметрах метода были указаны данные, которые будут храниться в датасете, его имя и описание.

Листинг 3 – Загрузка данных характеристик

```
domain_client.load_dataset(
   name="testDataset_v3",
   assets=final_dataset,
   description="Our dataset contains real-valued features for cell nu-
cleus of a breast mass. There are 31 columns and 250 rows in our da-
taset.")
```

Также на узел был загружен второй набор данных, обозначающий тип клетки (листинг 4).

Листинг 4 – Загрузка данных типа клетки

```
domain_client.load_dataset(
   name="testDataset_v3_diagnosis",
   assets={
       "diagnosis": diagnosis,
   },
   description="Dataset contains diagnosis result. There are 1 column and
250 rows in our dataset.")
```

Для того, чтобы проверить загрузку данных на узел, выведем объект `datasets` класса `DomainClient`. Вывод объекта представлен на рисунке 10.

Idx	Name	Description	Assets	Id
[0]	testDataset_v3	Our dataset contains test data. There are 31 columns and 250 rows in our dataset.	["radius_mean"] -> Tensor ["texture_mean"] -> Tensor ["perimeter_mean"] -> Tensor ...	a4614973-9288-4a94-9019-cb06f42d7bdf
[1]	testDataset_v3_diagnosis	Dataset contains diagnosis result. There are 1 column and 250 rows in our dataset.	["diagnosis"] -> Tensor	a62e69d7-0ecb-4e49-a168-e2c94764f050

Рисунок 10 – Загруженные на узел данные

Также с профиля владельца данных был создан новый пользователь с ролью аналитика данных при помощи метода `create` объекта `users`. В параметрах метода были указаны имя пользователя, почта, пароль и

количество бюджета конфиденциальности. Вызов метода представлен в листинге 5.

Листинг 5 – Создание нового пользователя

```
domain_client.users.create({
        "name": "Vladimir Fake",
        "email": "vova.degtyarev.01@mail.ru",
        "password": "20012002",
        "budget": 5000000})
```

Для того, чтобы посмотреть всех пользователей узла, выведем объект users (рисунок 11). В рамках данного узла существуют два пользователя. Пользователь «Jane Doe» является владельцем данных, а пользователь «Vladimir Fake» является аналитиком данных.

	id	email	name	budget	verify_key	role
0	1	info@openmined.org	Jane Doe	5.55	950ae3021cdc01ebbf27f6d20cd7b7b72cfcd8a98ac24a...	Owner
1	3	vova.degtyarev.01@mail.ru	Vladimir Fake	5000000.00	9fb07148374b827d6d64854dd3bea5956e9893d0bdc0f2...	Data Scientist

Рисунок 11 – Вывод объекта о пользователях

Аналогичным образом были загружены данные на узел на втором устройстве.

3.4. Процесс получения данных из узла

Процесс подключения к первому узлу в качестве аналитика данных продемонстрирован в листинге 6.

Листинг 6 – Подключение к первому узлу

```
try:
    domain = sy.login(
       email="vova.degtyarev.01@mail.ru",
       password="20012002",
       url="192.168.9.251",
       port=8081
    )
except Exception as e:
    print("Unable to login. Please check the domain you are connecting
to`")
```

Аналитик данных может вывести любой размещенный на узле объект данных для получения демонстрационных данных. Например, вывод объекта, представляющего собой тензор `radius_mean` датасета с идентификатором 0 представлен на рисунке 12.

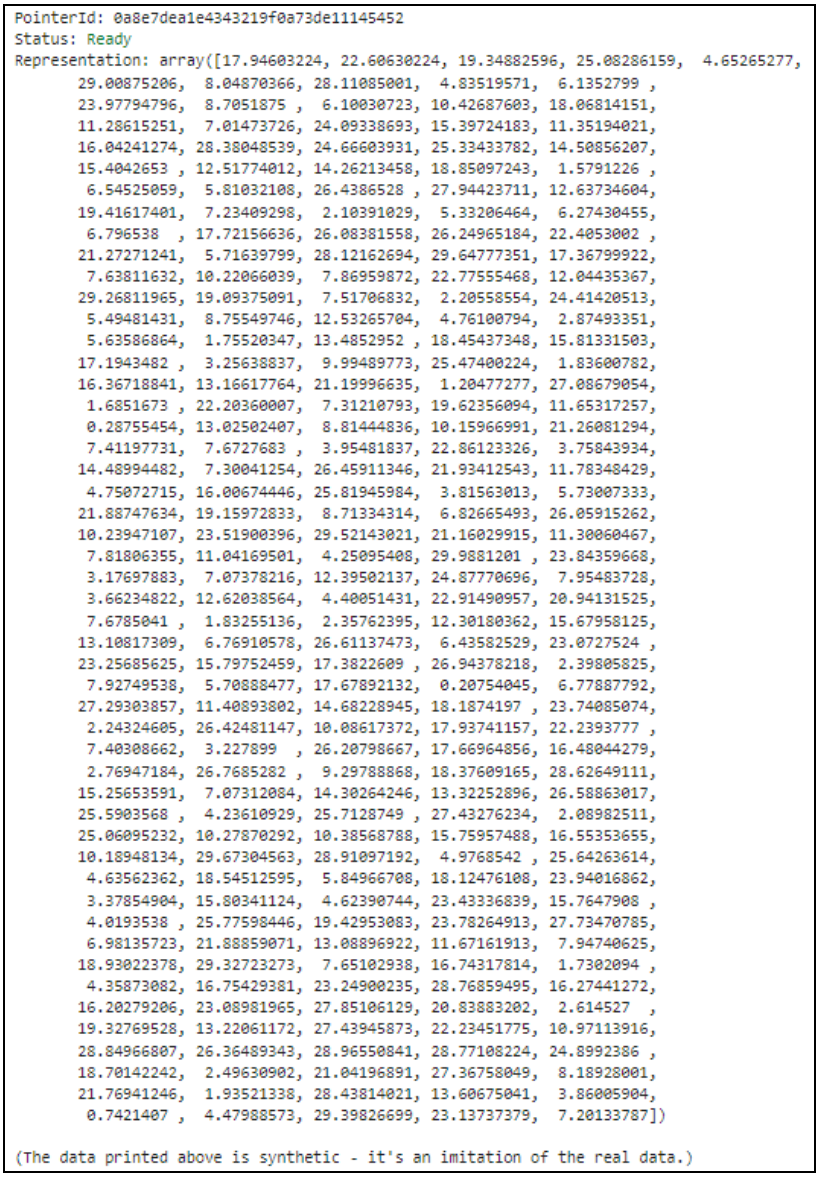

Рисунок 12 – Вывод тензора

Для того, чтобы аналитик данных смог получить данные с узла, ему нужно вызвать метод `publish` определенного объекта данных, чтобы узел наделил исходные данные этого объекта дифференциальной приватностью.

Процесс преобразования данных происходит на узле, к которому подключается аналитик данных. В параметрах этого метода указывается значение сигмы. Значение сигмы определяет насколько точны будут полученные данные. Сигма определяет максимальный разброс между значениями исходных и предоставляемых данных. Преобразование данных происходит так, что к исходным значениям данных добавляется случайная величина в диапазоне от отрицательного до положительного значения сигмы. Чем меньше значение сигмы, тем более точны будут полученные данные, и тем больше бюджета конфиденциальности будет потрачено у аналитика данных.

Для каждой из характеристик ядра клетки экспериментальным путем были подобраны оптимальные значения сигм.

После того, как узел преобразовал данные, аналитик данных может получить преобразованные данные вызвав метод `get`.

Процесс получения данных о характеристиках ядер клеткок аналитиком показан в листинге 7.

Листинг 7 – Получение данных

```
published_data = []
sigmas = [3, 4, 20, 100, 0.03, 0.01, 0.05, 0.03, 0.04, 0.01, 0.05, 0.15,
0.4, 5, 0.01, 0.04, 0.08, 0.01, 0.01, 0.01, 4, 5, 15, 80, 0.015, 0.01,
0.1, 0.1, 0.08, 0.02]
for asset_n, sigma_n in zip(assets, sigmas):
    feature = cancer_dataset[asset_n['name']]
    feature_public = feature.publish(sigma=sigma_n)
    feature_public.block_with_timeout(50)
    privacy_feature = feature_public.get(delete_obj=False)
    published_data.append(privacy_feature)
```

Полученные данные были обработаны и преобразованы в `DataFrame`. Данный процесс продемонстрирован в листинге 8.

Листинг 8 – Преобразование данных

```
published_data1 = np.asarray(published_data)
deboxed = np.array([item for item in published_data1.flatten()]).re-
shape(published_data1.shape)
data_df = pd.DataFrame(deboxed)
data_df = data_df.transpose()
```

Преобразованные данные продемонстрированы на рисунке 13.

	0	1	2	3	4	5	6	7	8	9	...	20	21
0	16.132178	8.434578	134.104971	1100.135552	0.087600	0.280242	0.298157	0.153714	0.226333	0.071672	...	34.266756	15.488959
1	17.078989	17.094781	127.665273	1272.185442	0.105969	0.089305	0.032212	0.048682	0.154750	0.063071	...	36.662754	23.940783
2	15.622685	15.774698	162.606136	1219.693821	0.103326	0.171460	0.152722	0.105217	0.220147	0.063774	...	18.326853	18.217385
3	10.051632	22.897328	55.129015	359.715497	0.160802	0.288066	0.231524	0.052108	0.190165	0.093511	...	12.677428	18.982321
4	20.652202	17.420749	126.831422	1330.350017	0.100585	0.117940	0.205592	0.097690	0.126316	0.048576	...	20.672949	19.742730
...	...	...	...	...	...	...	...	...	...	...	...	...	...
245	10.478143	17.744799	80.199168	281.126391	0.119582	0.058763	0.050669	-0.000533	0.211496	0.069394	...	7.003120	34.228511
246	13.277229	11.052537	120.697787	579.991444	0.045019	0.052506	0.023185	0.038215	0.102785	0.059122	...	20.335097	33.831021
247	13.609391	12.406311	89.819592	332.250218	0.133811	0.123512	0.107841	0.053611	0.173542	0.065869	...	12.548284	17.728222
248	9.921877	26.998829	88.430236	475.303414	0.091708	0.073301	-0.026388	0.005190	0.159686	0.054961	...	7.946641	27.365150
249	6.742087	19.200117	80.417527	363.016130	0.099238	0.067331	0.005977	0.022594	0.138886	0.057071	...	14.359081	23.946826

250 rows × 30 columns

Рисунок 13 – Преобразованные данные

Также аналитиком были получены данные о типах клеток, так как эти данные не были аннотированы дифференциальной приватностью, для их получения достаточно вызвать метод `get` (листинг 9).

Листинг 9 – Получение данных

```
diagnosis_data = diagnosis_dataset["diagnosis"].get(delete_obj=False)
diagnosis_data = diagnosis_data.child
deboxed_diagnosis = np.array([item for item in diagnosis_data.flat-
ten()]).reshape(diagnosis_data.shape)
data_df_diagnosis = pd.DataFrame(deboxed_diagnosis)
```

На операции получения данных аналитик данных потратил 311996 единиц конфиденциальности бюджета. Аналогичным образом аналитиком данных были получены данные с узла, размещенном на втором устройстве.

Выводы по третьей главе

В данной главе была реализована система для анализа географически-распределенных данных на платформе PySyft, было запущено два узла, на которые были загружены данные о ядрах клеток опухоли молочной железы, с другого устройства были получены эти данные, обладающие дифференциальной приватностью.

4. ЭКСПЕРИМЕНТЫ

В качестве примера работоспособности системы в рамках данной работы будет решена задачи классификации.

Цель прикладной задачи заключается в том, чтобы на основе данных о ядрах клеток опухоли молочной железы выявить, злокачественной или доброкачественной является клетка опухоли.

Для решения данной задачи будет реализована модель дерева решений.

Данные о клетках молочной железы были взяты с сайта Kaggle [20]. Это набор данных, представляющий собой таблицу с 569 строками и 31 столбцом, каждый из этих столбцов обозначает определенную характеристику ядра клетки молочной железы и имеет вещественное значение, а также столбец, который обозначает тип клетки. Для первого узла были выделены первые 250 строк набора данных, для второго узла следующие 250 строк набора данных.

Для построение модели дерева решений был использован класс `DecisionTreeClassifier` библиотеки sklearn. В конструкторе класса был указан критерий энтропии, который вычисляет энтропию Шеннона возможных классов [19].

После разделения данных, полученных с первого узла, на обучающие и тестовые выборки, модель была обучена на обучающей выборке с помощью метода `fit`, где обучающими входными образцами являлись данные о характеристиках ядер клеткок, а целевым значением данные о типах клеток. Также на тестовых данных было подсчитана средняя точность модели с помощью метода `score`.

Перед обучением модели, она с помощью модуля pickle стандартной библиотеки python загружается из файла «model.pkl», а после обучения модели на данных, она сохраняется в этот же файл. Тем самым, модель совершенствуется после каждого обучения на новом наборе данных.

Модель была повторно обучена на данных, которые были получены со второго узла.

Процесс загрузки, обучения и сохранения модели продемонстрирован в листинге 10.

Листинг 10 – Загрузка, обучение и сохранение модели

```
from sklearn import tree
from sklearn.model_selection import train_test_split
import pickle
import os.path

X = data_df
y = data_df_diagnosis

if os.path.isfile('model.pkl'):
    with open("model.pkl", 'rb') as file:
    clf = pickle.load(file)
else:
    clf = tree.DecisionTreeClassifier(criterion='entropy')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

clf.fit(X_train, y_train)

accuracy = clf.score(X_test, y_test)
print("Accuracy:", accuracy)

pkl_filename = "model.pkl"
with open(pkl_filename, 'wb') as file:
    pickle.dump(clf, file)
```

Выводы по четвертой главе

В данной главе была поставлена задача по анализу данных для проверки работоспособности системы, был выбран набор данных, задача была решена построением модели дерева решений на основе полученных с узлов данных.

5. ТЕСТИРОВАНИЕ

Для тестирования системы для анализа географически-распределенных данных на платформе PySyft и функции системы наделять данные на узлах дифференциальной приватностью выведем данные, которые были загружены на узел (рисунок 14), а также данные, которые были получены аналитиком данных (рисунок 15).

radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...	...	...	...	...	...	...	...
10.48	19.86	66.72	337.7	0.10700	0.05971	0.04831	0.03070
13.20	17.43	84.13	541.6	0.07215	0.04524	0.04336	0.01105
12.89	14.11	84.95	512.2	0.08760	0.13460	0.13740	0.03980
10.65	25.22	68.01	347.0	0.09657	0.07234	0.02379	0.01615
11.52	14.93	73.87	406.3	0.10130	0.07808	0.04328	0.02929

Рисунок 14 – Загруженные на узел данные

0	1	2	3	4	5	6	7
16.132178	8.434578	134.104971	1100.135552	0.087600	0.280242	0.298157	0.153714
17.078989	17.094781	127.665273	1272.185442	0.105969	0.089305	0.032212	0.048682
15.622685	15.774698	162.606136	1219.693821	0.103326	0.171460	0.152722	0.105217
10.051632	22.897328	55.129015	359.715497	0.160802	0.288066	0.231524	0.052108
20.652202	17.420749	126.831422	1330.350017	0.100585	0.117940	0.205592	0.097690
...	...	...	...	...	...	...	...
10.478143	17.744799	80.199168	281.126391	0.119582	0.058763	0.050669	-0.000533
13.277229	11.052537	120.697787	579.991444	0.045019	0.052506	0.023185	0.038215
13.609391	12.406311	89.819592	332.250218	0.133811	0.123512	0.107841	0.053611
9.921877	26.998829	88.430236	475.303414	0.091708	0.073301	-0.026388	0.005190
6.742087	19.200117	80.417527	363.016130	0.099238	0.067331	0.005977	0.022594

Рисунок 15 – Полученные данные

После обучения модели на наборе данных, размещенном на первом узле, среднее значение точности модели было равно 0,88. После последующего обучения этой же модели на наборе данных, размещенном на втором узле, среднее значение точности модели было равно 0,94.

Выводы по пятой главе

В данной главе была протестирована система для анализа географически-распределенных данных на платформе PySyft на возможность загрузки данные на узел, получения этих данных, преобразованных с дифференциальной приватностью. Также были получены результаты точности построенной модели дерева решений после обучения на данных, размещенных на двух узлах.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была спроектирована и реализована система для анализа географически-распределенных данных на платформе PySyft. При этом были решены следующие задачи.

1. Выполнен обзор литературы.
2. Выполнен анализ аналогичных проектов.
3. Определены функциональные и нефункциональные требования к системе.
4. Спроектирована система для анализа географически-распределенных данных на платформе PySyft.
5. Реализована система для анализа географически-распределенных данных на платформе PySyft.
6. Проведено тестирование системы для анализа географически-распределенных данных на платформе PySyft.

ЛИТЕРАТУРА

1. 7 примеров применения машинного обучения в 5 отраслях бизнеса. [Электронный ресурс] URL: https://mcs.mail.ru/blog/17-primerov-mashinnogo-obucheniya (дата обращения: 13.02.2023 г.).

2. Анализ данных – основы и терминология. [Электронный ресурс] URL: https://habr.com/ru/post/352812/ (дата обращения: 13.02.2023 г.).

3. Что такое интеллектуальный анализ данных? [Электронный ресурс] URL: https://aws.amazon.com/ru/what-is/data-mining/ (дата обращения: 16.02.2023 г.).

4. What Is Data Analysis? [Электронный ресурс] URL: https://www.datapine.com/blog/data-analysis-methods-and-techniques/ (дата обращения: 16.02.2023 г.).

5. Официальный сайт PySyft. [Электронный ресурс] URL: https://github.com/OpenMined/PySyft/ (дата обращения: 13.02.2023 г.).

6. Масштабируемый подход к частично локальному федеративному обучению. [Электронный ресурс] URL: https://habr.com/ru/post/645783/ (дата обращения: 16.02.2023 г.).

7. Дифференциальная приватность — анализ данных с сохранением конфиденциальности. [Электронный ресурс] URL: https://habr.com/ru/company/domclick/blog/526724/ (дата обращения: 16.02.2023 г.).

8. Конфиденциальное машинное обучение. Библиотека PySyft. [Электронный ресурс] URL: https://habr.com/ru/post/500154/ (дата обращения: 16.02.2023 г.).

9. Методы обфускации трафика. Гомоморфное шифрование. [Электронный ресурс] URL: https://habr.com/ru/company/globalsign/blog/717482/ (дата обращения: 16.02.2023 г.).

10. Цифровые фиатные деньги, гомоморфное шифрование и другие перспективные направления криптографии. [Электронный ресурс] URL: https://habr.com/ru/company/kryptonite/blog/658113/ (дата обращения: 18.02.2023 г.).

11. Официальный сайт OpenMined. [Электронный ресурс] URL: https://www.openmined.org/ (дата обращения: 24.02.2023 г.).

12. Introduction to Remote Data Science. [Электронный ресурс] URL: https://habr.com/ru/post/402987/ (дата обращения: 24.02.2023 г.).

13. Google изобрела распределённый ИИ для миллиарда смартфонов. [Электронный ресурс] URL: https://habr.com/ru/post/402987/ (дата обращения: 24.02.2023 г.).

14. Официальный сайт Tensorflow Federated. [Электронный ресурс] URL: https://www.tensorflow.org/federated/ (дата обращения: 24.02.2023 г.).

15. Официальная документация Python. [Электронный ресурс] URL: https://docs.python.org/ (дата обращения: 11.04.2023 г.).

16. Advanced Deployment: Introduction to HaGrid. [Электронный ресурс] URL: https://openmined.github.io/PySyft/deployment/ (дата обращения: 11.04.2023 г.).

17. Официальный сайт NumPy. [Электронный ресурс] URL: https://numpy.org/ (дата обращения: 11.04.2023 г.).

18. Официальный сайт pandas. [Электронный ресурс] URL: https://pandas.pydata.org/ (дата обращения: 11.04.2023 г.).

19. Официальный сайт scikit-learn. [Электронный ресурс] URL: https://scikit-learn.org/ (дата обращения: 11.04.2023 г.).

20. Breast Cancer Wisconsin (Diagnostic) Data Set. [Электронный ресурс] URL: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/ (дата обращения: 11.04.2023 г.).

ПРИЛОЖЕНИЕ. Аннотация набора данных

Листинг 1 – Аннотация набора данных

```
final_dataset = dict()

data_subjects = sy.DataSubjectArray.from_objs(dataset1["id"])

final_dataset["radius_mean"] = sy.Tensor(dataset1["radius_mean"]).anno-
tate_with_dp_metadata(
   lower_bound=0, upper_bound=30, data_subject=data_subjects
)
final_dataset["texture_mean"] = sy.Tensor(dataset1["texture_mean"]).anno-
tate_with_dp_metadata(
   lower_bound=8, upper_bound=42, data_subject=data_subjects
)
final_dataset["perimeter_mean"] = sy.Tensor(dataset1["perime-
ter_mean"]).annotate_with_dp_metadata(
   lower_bound=40, upper_bound=200, data_subject=data_subjects
)
final_dataset["area_mean"] = sy.Tensor(dataset1["area_mean"]).anno-
tate_with_dp_metadata(
   lower_bound=100, upper_bound=3000, data_subject=data_subjects
)
final_dataset["smoothness_mean"] = sy.Tensor(dataset1["smooth-
ness_mean"]).annotate_with_dp_metadata(
   lower_bound=0.03, upper_bound=0.20, data_subject=data_subjects
)
final_dataset["compactness_mean"] = sy.Tensor(dataset1["compact-
ness_mean"]).annotate_with_dp_metadata(
   lower_bound=0.01, upper_bound=0.45, data_subject=data_subjects
)
final_dataset["concavity_mean"] = sy.Tensor(dataset1["concav-
ity_mean"]).annotate_with_dp_metadata(
   lower_bound=0, upper_bound=0.5, data_subject=data_subjects
)
final_dataset["concave points_mean"] = sy.Tensor(dataset1["concave
points_mean"]).annotate_with_dp_metadata(
   lower_bound=0, upper_bound=0.3, data_subject=data_subjects
)
final_dataset["symmetry_mean"] = sy.Tensor(dataset1["symmetry_mean"]).an-
notate_with_dp_metadata(
   lower_bound=0.08, upper_bound=0.4, data_subject=data_subjects
)
final_dataset["fractal_dimension_mean"] = sy.Tensor(dataset1["fractal_di-
mension_mean"]).annotate_with_dp_metadata(
   lower_bound=0.02, upper_bound=0.15, data_subject=data_subjects
)
final_dataset["radius_se"] = sy.Tensor(dataset1["radius_se"]).anno-
tate_with_dp_metadata(
   lower_bound=0.08, upper_bound=3, data_subject=data_subjects
)
final_dataset["texture_se"] = sy.Tensor(dataset1["texture_se"]).anno-
tate_with_dp_metadata(
   lower_bound=0.2, upper_bound=6, data_subject=data_subjects
)
final_dataset["perimeter_se"] = sy.Tensor(dataset1["perimeter_se"]).anno-
tate_with_dp_metadata(
   lower_bound=0.5, upper_bound=30, data_subject=data_subjects
)
```

```
final_dataset["area_se"] = sy.Tensor(dataset1["area_se"]).anno-
tate_with_dp_metadata(
   lower_bound=5, upper_bound=650, data_subject=data_subjects
)
final_dataset["smoothness_se"] = sy.Tensor(dataset1["smoothness_se"]).an-
notate_with_dp_metadata(
   lower_bound=0, upper_bound=0.05, data_subject=data_subjects
)
final_dataset["compactness_se"] = sy.Tensor(dataset1["compact-
ness_se"]).annotate_with_dp_metadata(
   lower_bound=0, upper_bound=0.2, data_subject=data_subjects
)
final_dataset["concavity_se"] = sy.Tensor(dataset1["concavity_se"]).anno-
tate_with_dp_metadata(
   lower_bound=0, upper_bound=0.6, data_subject=data_subjects
)
final_dataset["concave points_se"] = sy.Tensor(dataset1["concave
points_se"]).annotate_with_dp_metadata(
   lower_bound=0, upper_bound=0.1, data_subject=data_subjects
)
final_dataset["symmetry_se"] = sy.Tensor(dataset1["symmetry_se"]).anno-
tate_with_dp_metadata(
   lower_bound=0, upper_bound=0.1, data_subject=data_subjects
)
final_dataset["fractal_dimension_se"] = sy.Tensor(dataset1["fractal_di-
mension_se"]).annotate_with_dp_metadata(
   lower_bound=0, upper_bound=0.05, data_subject=data_subjects
)
final_dataset["radius_worst"] = sy.Tensor(dataset1["radius_worst"]).anno-
tate_with_dp_metadata(
   lower_bound=5, upper_bound=50, data_subject=data_subjects
)
final_dataset["texture_worst"] = sy.Tensor(dataset1["texture_worst"]).an-
notate_with_dp_metadata(
   lower_bound=6, upper_bound=80, data_subject=data_subjects
)
final_dataset["perimeter_worst"] = sy.Tensor(dataset1["perime-
ter_worst"]).annotate_with_dp_metadata(
   lower_bound=25, upper_bound=400, data_subject=data_subjects
)
final_dataset["area_worst"] = sy.Tensor(dataset1["area_worst"]).anno-
tate_with_dp_metadata(
   lower_bound=100, upper_bound=8000, data_subject=data_subjects
)
final_dataset["smoothness_worst"] = sy.Tensor(dataset1["smooth-
ness_worst"]).annotate_with_dp_metadata(
   lower_bound=0.02, upper_bound=0.50, data_subject=data_subjects
)
final_dataset["compactness_worst"] = sy.Tensor(dataset1["compact-
ness_worst"]).annotate_with_dp_metadata(
   lower_bound=0.01, upper_bound=2, data_subject=data_subjects
)
final_dataset["concavity_worst"] = sy.Tensor(dataset1["concav-
ity_worst"]).annotate_with_dp_metadata(
   lower_bound=0, upper_bound=3, data_subject=data_subjects
)
final_dataset["concave points_worst"] = sy.Tensor(dataset1["concave
points_worst"]).annotate_with_dp_metadata(
   lower_bound=0, upper_bound=0.5, data_subject=data_subjects
```

```
)
final_dataset["symmetry_worst"] = sy.Tensor(dataset1["sym-
metry_worst"]).annotate_with_dp_metadata(
   lower_bound=0.1, upper_bound=1, data_subject=data_subjects
)
final_dataset["fractal_dimension_worst"] = sy.Tensor(dataset1["frac-
tal_dimension_worst"]).annotate_with_dp_metadata(
   lower_bound=0.02, upper_bound=0.5, data_subject=data_subjects
)
```