

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БАШКИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОМПЬЮТЕРНОЙ
МАТЕМАТИКИ**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ПО ПРОГРАММЕ МАГИСТРАТУРЫ**

ЗАГИРОВ РОБЕРТ АЛЬБЕРТОВИЧ

**РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ
НЕЙРОСЕТЕВЫХ РЕШЕНИЙ НА ОСНОВЕ PYTHON 3 И МИКРОФРЕЙМВОРКА
FLASK**

Выполнил:
Студент(ка) 2 курса очной формы обучения
Направление подготовки (специальность)
01.04.02 «Прикладная математика
и информатика»
Направленность (профиль) Механика
жидкости, газа и плазмы

Допущено к защите в ГЭК и проверено
на объем заимствования:

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

Руководитель
к.т.н., доц.

_____ / А.М. Болотнов

_____ / Д.В. Полупанов

УФА – 2021

Оглавление

Введение.....	3
1. Постановка задачи.....	6
1.1 Формализация работы системы.....	6
1.2 Модель системы.....	6
2. Инструментарий.....	8
2.1 Сериализация.....	8
2.2 Docker.....	10
2.3 СУБД.....	14
3. Описание работы системы-прототипа.....	15
3.1 Получение рейтинга.....	16
3.2 Пример задачи.....	17
Заключение.....	29
Список литературы.....	31
Приложение А.....	33
db.sql.....	33
app.py.....	35
Приложение Б.....	38
Dockerfile.....	38

Введение

С момента появления компьютеров в обиходе постоянного пользования и эволюции сети интернет из ARPANET до того вида, которым мы пользуемся сегодня каждый день, одним из самых масштабных способов их применения был в сфере улучшения процесса обучения. Одним из таких средств является Онлайн-судья. Это система для тестирования программ в соревнованиях по программированию. Они также используются для тренировок на таких соревнованиях. Система может компилировать и выполнять код, а также тестировать код с заранее созданными данными. Отправленный код может запускаться с ограничениями, включая ограничение по времени, лимит памяти, ограничение безопасности и так далее. Вывод кода будет зафиксирован системой и сравнен со стандартным выводом. Затем система вернет результат. Любые ошибки в коде должны быть исправлены и отправлены на повторное рассмотрение. В настоящее время одними из самых популярных платформ для автоматического тестирования кода являются Kaggle, HackerRank, Яндекс.Контест и др.

Переход к онлайн формам обучения во время коронавирусной инфекции также дал понять, что онлайн-инфраструктура для работы и обучения в высших учебных заведениях — это не роскошь, а необходимость, особенно на специальностях, тесно связанных с компьютерными технологиями, а таких сейчас большинство. Проверка студенческих работ преподавателем имеет недостатки и без смены режима его работы[1], стресс и изменение обстановки может ухудшить этот и без того трудоемкий процесс[4].

К тому же раздел разработки программного обеспечения, связанного с искусственным интеллектом ещё достаточно молод, чтобы обрести какие-то методологии развертывания программного кода из-за чего многие решения требуют подготовки достаточно нетривиальной инфраструктуры, требующей опыта и знаний специалиста в области DevOps[5], для начала работы. Курсы, нацеленные на обучение работы с машинным обучением и искусственным интеллектом или любую другую учебную деятельность, связанную с практической частью компьютерных наук должны обеспечивать всю необходимую рабочую среду в виде удобного и простого для использования веб-клиента для студентов и преподавателей, исключая лишнюю сложность,

связанную с взаимодействием с экосистемой Linux[3].

Интерес к искусственному интеллекту возрос за последние несколько лет, что подтверждает заявление Президента России В.В. Путина на конференции по искусственному интеллекту Artificial Intelligence Journey (AI Journey 2020) на тему «Искусственный интеллект — главная технология XXI века»[2]: «Нужно научиться управлять искусственным интеллектом. Когда я говорил о том, что люди будут контролировать машины, я именно это имел в виду. Необходимо подчинить себе одну из величайших технологий, когда-либо созданных человечеством, и для этого нам нужно самим быть смелыми, компетентными и смотреть в будущее.». Это высказывание отражает ситуацию преподавания учебных дисциплин, связанных с искусственным интеллектом в БашГУ, что в связи с отсутствием открытого и свободно распространяемого решения по автоматизации проверки нейросетевых задач делает очевидным тот факт, что разработка такой системы, является необходимым решением для продвижения прогресса в преподавании прикладного машинного обучения и искусственного интеллекта в целом.

Исходя из выше изложенного **целью** выпускной квалификационной работы является создание дизайна архитектуры приложения для автоматической оценки решений с применением нейросетей и алгоритмов искусственного интеллекта, реализованную с помощью стека выбранного автором стека технологий.

Перечислим задачи, решение которых способствует достижению указанной цели:

Создать дизайн системы, позволяющую оценивать проекты пользователей в курсе обучения нейронным сетям

1. Создать дизайн системы, позволяющую оценивать проекты пользователей в курсе обучения нейронным сетям
2. Разработать программную систему, реализующую с наличием узла-клиент, узла-процессора, а также их связь с узлом-хранилищем
3. Продемонстрировать работоспособность системы осуществив решение задачи «Титаник» и получение рейтинга в системе

Выпускная квалификационная работа состоит из введения, заключения, списка используемой литературы и приложения. Первая глава носит

теоретический характер и рассматривает инструментарий и абстрактную модель системы, а также различные средства сериализации моделей искусственного интеллекта. Во второй главе описывается реализация и использование онлайн-платформы для автоматического тестирования нейронных сетей. В заключении приводятся основные результаты и выводы. Приложение А содержит схему базы данных и код реализации узла-процессора, приложение Б – конфигурационный файл для Docker-образа.

1. Постановка задачи

1.1 Формализация работы системы

Пусть X, V, u исходный набор данных для обучения, проверочный набор данных и подлинный набор данных соответственно. Работа системы подразумевает операцию сравнения наборов данных $C(u, u')$ и выставление рейтинга R , т.е. $C(u, u') = R$, где u' является результатом работы нейросетевой модели, предоставленной пользователем на наборе данных V , т.е. $M(V) = u'$. За разработчиком остается выбор способа хранения и подачи на проверку наборов данных, а также получение и хранения рейтинга. Проверка кода подразумевает его запуск в безопасной среде с невозможностью навредить основной операционной системе с помощью одного из способов виртуализации. Пользователь должен иметь возможность предоставить решение системе для проверки. В случае обычного онлайн судьи достаточно получить код программы в виде строки, однако для задач искусственного интеллекта также требуются наличие модели, натренированной пользователем на наборе данных X и код программы для запуска модели, например, в виде архива. Наличие программы подразумевает выбор некоего языка программирования. В мире искусственного интеллекта превалирует скриптовый язык Python из-за обилия байндингов для различных библиотек, а также легкости работы в самом языке, что позволяет сосредоточиться на прикладных задачах. В связи с необходимостью преподавания машинного обучения, а не методик программирования, эти свойства были основополагающими для выбора Python в качестве языка написания модельных скриптов. По окончании проверки модели, предоставленной пользователем следует ознакомить пользователя с результатом работы его модели в виде рейтинга.

1.2 Модель системы

Опишем абстрактную модель, которой должна следовать реализация:



Рис 1.1 Абстрактная модель системы.

Система должна иметь узел-клиент для обработки запросов пользователя о размещении архива на проверку, своем рейтинге на текущем задании, а также навигации по доступным пользователю трекам, узел-процессор для проверки моделей пользователей, а также узел-хранилище для сохранения результатов узла-процессора.

Узлы могут быть реализованы в одном монолитном приложении или же разделены для использования в микросервисной архитектуре, здесь и далее используется последний вариант для упрощения масштабируемости и снижения умственной нагрузки, связанной с управлением большой структурой проекта.

Узел-процессор должен иметь возможность безопасно запустить присланную в архиве модель. В отличие от обычных систем онлайн-тестирования для оценки заданий по программированию, для которых требуется только наличие кода программы, для оценки нейросетевой модели требуется её наличие в каком-либо сериализованном формате: pickle, joblib, hd5 и т.д.

Принимая во внимание возможность асинхронной обработки запросов многих современных веб-фреймворков один узел-процессор может обрабатывать несколько запросов одновременно, что означает необходимость наличия механизма атомарной десериализации архива внутри контейнера, для этой цели может быть использован временный каталог с уникальным идентификатором, созданный с помощью генератора случайных чисел или

universally unique identifier.

2. Инструментарий

Для реализации прототипа были выбраны CRUD-oriented веб-приложение на языке программирования Python с использованием микрофреймворка Flask, как узел-клиент, REST-oriented веб-приложение также с использованием Flask и модулем для кроссдоменных запросов CORS для реализации узла-процессора и базой данных MySQL в качестве узла-хранилища. Для изоляции запускаемых моделей было решено взять технологию контейнеризации Docker. Также рассмотрены недостатки данного прототипа и методы улучшения для будущих реализаций.

Рассмотрим используемые технологии и приведем их краткое описание.

1. Сериализация
2. Docker
3. СУБД

2.1 Сериализация

Pickle, средство, включенное в стандартную библиотеку Python, реализующее двоичные протоколы для «упаковывания»(pickling) и «распаковывания»(unpickling) структуры Python-объектов. Имеет преимущество над также встроенными в стандартную библиотеку модулем marshal в виду того, что marshal используется только для поддержки файлов, содержащих скомпилированный Python-байткод, и json т. к. в обоих модулях отсутствует поддержка сериализации пользовательских классов, к тому же json является форматом для сериализации текста и не поддерживает работу с двоичными файлами.

Рассмотрим примерное использование pickle для сохранения модели:

```
import pickle

model = ...

with open('mdl.pkl', 'wb') as f:
    pickle.dump(model, f)
```

Рисунок 2.1 pickling


```
import pickle

with open('mdl.pkl', 'rb') as f:
    model = pickle.load(f)

# model usage...
```

Рисунок 2.2 unpickling

Однако несмотря на простоту использования и доступность, pickle имеет несколько недостатков:

1. Не гарантирует безопасность, поэтому unpickling следует производить только тех файлов, которые были получены из надежного источника или подписывать сериализованные файлы с помощью модуля hmac;
2. Для использования десериализованной модели внутри узла-процессора требуется наличие информации о том какая библиотека была использована для её создания, а также её наличие внутри контейнера, чтобы воспользоваться функциями модели;
3. Модель, созданная с помощью одной версии библиотеки может отличаться от используемой внутри узла-процессора, что может привести к получению неправильных данных при оценке модели и соответственно к снижению полученного рейтинга

joblib, позволяет использовать парадигму «вычислений по требованию», которая снижает количество повторяющихся операций с помощью отслеживания входных данных, если данные не менялись, результаты вычислений будут возвращены из кэша. Отслеживание данных также позволяет воспроизводить эксперименты, что немаловажно для оценки точности моделей машинного обучения. К сожалению, все недостатки pickle также присутствует и при использовании joblib, а самая главная проблема использования десериализованной модели и её портативности все так же остается нерешенной.

hd5, используется многими фреймворками для сохранения моделей на определенных стадиях обучения и для единого формата хранения наборов данных, а также формат CoreML от Apple не подходит для универсальной обработки и оценки моделей так как первое просто формат сериализации, а второе подразумевает использование экосистемы Apple.

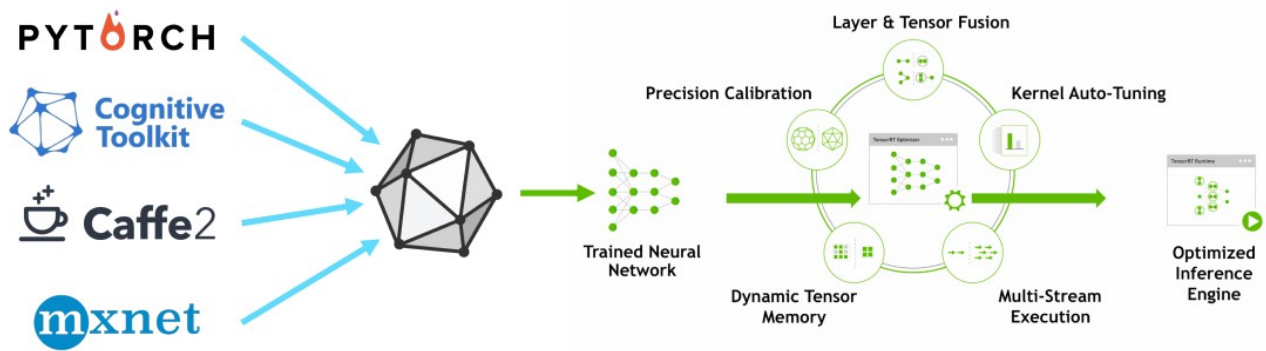


Рисунок 2.3 ONNX

Onnx[8] – является инициативой, организованной корпорациями Facebook и Google для создания открытого формата по описанию нейросетевых моделей, совместимый со множеством нейросетевых фреймворков. Этот формат хранит информацию о представлении нейронной сети, как множество направленных ациклических графов, где каждый узел графа является вызовом функции. В нейронных сетях часто используются функции тензорного умножения, свертки, функции активации, преобразования и т. д. Поток информации, проходящий по нейронной сети представляет собой соединение узлов из функций, хранящийся в ONNX графе вместе с параметрами. Этот формат поддерживает множество фреймворков, таких как: TensorFlow, Microsoft CNTK, PyTorch, SciKit-Learn, MatLab и Caffe2.

2.2 Docker

Для безопасного запуска пользовательских скриптов узел-процессор должен иметь возможность изолировать программу, которую требуется оценить. Одним из возможных решений является использование chroot jail – техники, изолирующий запускаемый процесс, а также его потомков от всей остальной операционной системы, давая доступ только к определенным каталогам и файлам. Однако применение этого способа является чересчур громоздким и в настоящее время небезопасным, так как на данный момент существует множество успешных методов обойти данный способ изоляции[11].

Виртуальная машина также может быть использована для изоляции узла-процессора от всей остальной системы, существующие платформы для создания виртуальных машин, такие как QEMU, vmWare, VirtualBox, KVM и т. д., однако для запуска одной виртуальной машины требуется образ системы, а также ручная настройка всех компонентов системы для её полноценной работы. К тому же присутствует значительная разница в скорости между

программами, которые работают в виртуальной машине и контейнере[12]. Docker же позволяет создать полноценный образ системы с помощью декларативного конфигурационного файла с описанием процесса по её созданию.

Популярность облачной миграции и микросервисной архитектуры в настоящее время набирает обороты, дизайн такого подхода заключается в использовании независимых и малообъемных модулей, которые разрабатываются, запускаются и масштабируются с помощью систем оркестрации. Микросервисы представляют собой службо-ориентированный подход к архитектуре для разработки программных приложений, состоящих из служб, которые могут быть запущены и масштабированы с помощью полностью автоматической системы, минимизируя необходимость в централизованном управлении. Микросервисы реализуют отдельные бизнес-функции. Каждый микросервис работает в своем отдельном процессе и получает необходимую информацию от других микросервисов с помощью несложного механизма API. В отличие от монолитных приложений микросервисы отличаются малым размером и быстрой реакцией на возможные отказы системы, микросервисы мало зависят друг от друга, делая их идеальными для распределенных систем, где один поврежденный микросервис не затронет остальную систему. Такой подход позволяет масштабировать инфраструктуру во избежание простоя системы или недостаточной мощности при критической нагрузке. Для достижения этих преимуществ используется технология контейнеризации, позволяющая использовать виртуализацию на уровне операционной системы.

Kubernetes, платформа с открытым кодом, отдающая в распоряжение разработчика набор строительных материалов для создания гибких и регенерирующих кластерных систем контейнеров. Kubernetes абстрагирует ненужную сложность оркестрации микросервисов, позволяя создавать кластер из взаимозаменяемых контейнеров.

Рассмотрим конфигурацию Docker-контейнера, использованную прототипом для узла-процессора. Для его функционирования использовался микрофреймворк Flask с модулем для кроссдоменных запросов и веб-сервером gunicorn, также для запуска моделей и манипуляции наборами данных фреймворк scikit-learn и библиотека pandas, готовый контейнер

использовался как REST-служба узлом-клиентом[13].

```
FROM debian:stable

RUN apt-get update && apt-get install -y \
    curl \
    ca-certificates \
    sudo \
    git \
    bzip2 \
    libx11-6 \
    && rm -rf /var/lib/apt/lists/*

RUN mkdir /app
WORKDIR /app

/* configuring user */

RUN conda install -y -c anaconda flask \
    flask-cors \
    gunicorn \
    pandas \
    && conda clean -ya

RUN conda install -y scikit-learn && conda clean -ya

COPY ./app.py ./app.py

ENTRYPOINT gunicorn --workers=4 --threads 10 -b 0.0.0.0:64544 app:app
```

Рисунок 2.4 Docker-образ

kubernetes дает возможности самой гибкой настройки решений для распределенных систем, используя концепции самовосстановления и автоматической масштабируемости. Для того, чтобы мы могли использовать кластер нужно создать конфигурацию для Service и Deployment. Опишем подробнее способ использования узла-процессора внутри кластера. Для этого в kubernetes доступны такие объекты, как Pod, Service и Deployment. Объект Pod — это минимально допустимая единица вычислений в kubernetes, содержащая один или несколько контейнеров. Компонент Service позволяет открыть доступ к контейнерам из локальной или внешней сети без настройки поиска служб внутри самих контейнеров, а Deployment декларативно описать состояние некоего набора Pods и запустить контролируемое обновление с помощью Deployment Controller. В данной конфигурации мы указываем порт на котором Service будет принимать запросы, изображение, используемое репликами и их количество. В итоге наш кластер будет содержать три реплики, обрабатывающие модели, которые могут быть взаимозаменяемо

использованы службой типа LoadBalancer.

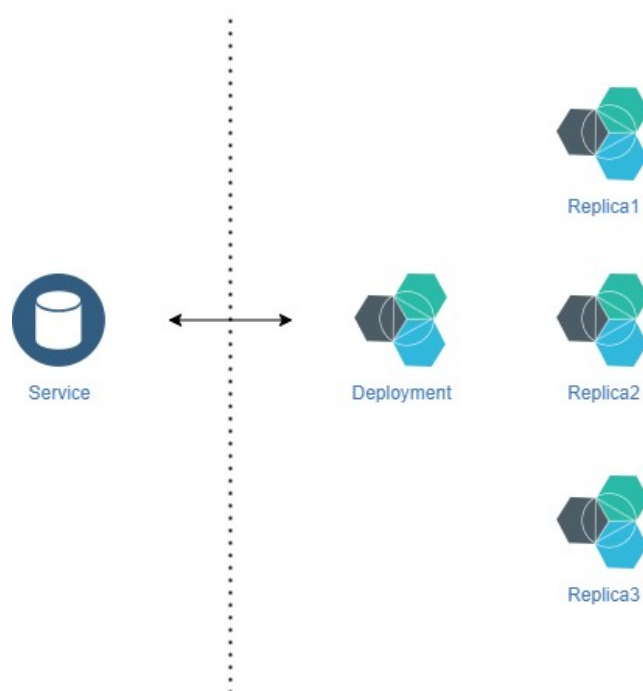


Рисунок 2.5 Диаграмма связи Service и Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pyml-deployment
  labels:
    app: pyml
spec:
  replicas: 3
  selector:
    matchLabels:
      app: pyml
  template:
    metadata:
      labels:
        app: pyml
    spec:
      containers:
        - name: pyml
          image: pyml:latest
          imagePullPolicy: Never
          ports:
            - containerPort: 64544
```

Рисунок 2.6 Конфигурация Deployment

```

apiVersion: v1
kind: Service
metadata:
  name: pyml-service
  labels:
    name: pyml-service
spec:
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 64544
  type: LoadBalancer
  selector:
    app: pyml

```

Рисунок 2.7 Конфигурация Service

2.3 СУБД

На данный момент для использования есть множество систем управления базами данных, наиболее популярным из них являются PostgreSQL[15] и MySQL[14], который дефакто был и является стандартным выбором при возникновении потребности в работе с SQL, хранением и анализом данных. Одно из возможных внутренних устройств MySQL MyISAM является одним из быстрых средств при чтении данных.

PostgreSQL является объектной СУБД в то время, как MySQL только реляционной. Это означает наличие в PostgreSQL средств моделирования, доступные объекто-ориентированных языках, такие как наследование таблиц и перегрузка функций. PostgreSQL одна из СУБД, которые наиболее отвечают стандарту SQL, а набор её возможностей превышает все доступные решения среди бесплатных СУБД. К тому же при использовании InnoDB в MySQL, который поддерживает транзакции и ключи-ограничения, отличия в скорости работы с PostgreSQL становятся минимальными.

Приведем схему, используемую СУБД MySQL для узла-хранилища:

Для демонстрации состоятельности абстрактной модели был реализован прототип с помощью стека технологий Flask/MySQL/Docker.

Для того, чтобы воспользоваться системой пользователю достаточно снабдить её архивом, который будет отправлен в кластер для обработки, система запустит присланную программу и сравнит её вывод с набором тестов. Далее результаты работы программы и оценка её успешности будут записаны в базу данных для отображения и сравнения результатов позже.

Администратору системы же нужно будет добавить необходимые задания и проверочные тесты для оформления и проверки решения пользователя, что налагает некие ограничения на валидность термина «тестирования» в данном контексте. Как и в обычном, ручном режиме проверки решения задач искусственного интеллекта все зависит от объема данных для данной задачи и её трудоемкости. Например такие относительно простые задачи, как распознавание цифр по изображениям в низком разрешении, написанных разным почерком или «Титаник», где по тренировочному набору данных надо определить судьбу пассажиров из тестовой выборки, являются таковыми из-за их возраста и вариативности: существует ограниченное количество написания цифр на маленькой площади, а свой рейс Титаник совершил только один раз. Это значит, что для их проверки потребуется относительно малый набор данных и тренировка моделей для таких задач не займет много времени. В то время, как распознавание лиц или решение задач с малым объемом данных накладывает некие ограничения как на студентов, так и на систему, оставляя решение таких задач неподходящим для обучающих систем.

3.1 Получение рейтинга

Пользователь отправляет в систему архив, содержащий скрипт, запускающий модель и саму модель. Система запускает пользовательскую модель, натренированную на данных, которыми пользователь был изначально снабжен в треке, на валидационном наборе данных. Система сравнивает пользовательский результат, который он записывает в csv файл с результатами, которые должны быть на самом деле, рейтинг это процент совпадения этих наборов.



Имя	Размер
 mdl.pkl	398 211
 script.py	535

Рисунок 3.2 Пример сабмита

3.2 Пример задачи

Рассмотрим реализацию одной из нейросетевых задач "Титаник". Для решения получения оценки за решение этой задачи необходимо снабдить систему архивом, содержащим скрипт, который будет оперировать данными, а также все другие необходимые файлы, например натренированную модель.

Титаник — известная задача в мире искусственного интеллекта, ориентированная в большей мере на начинающих. Датасет Титаник содержит данные пассажиров корабля. Цель задачи — построить модель, которая лучшим образом сможет предсказать, остался ли произвольный пассажир в живых или нет.

Задача подразумевает не только обучение модели, но и обучение студента основным навыкам Data Science, такие, как feature engineering - он должен определить какие свойства пассажиров дают лучшие показатели при анализе на тестовой выборке, также в данных присутствуют пропуски — обычное явление в работе с реальными данными.

Система состоит из треков или направлений, каждый из которых содержит в себе задания. Задача пользователя заключается в отправлении архива, содержащим, написанный им скрипт и модель, натренированную на исходных данных X , которую этот скрипт запустит внутри кластера, чтобы получить набор данных y' .

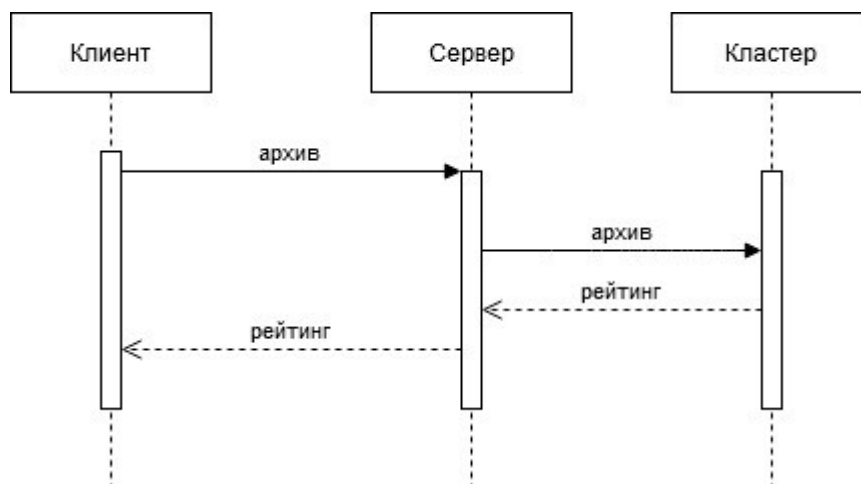


Рисунок 3.3 Диаграмма взаимодействия системы и пользователя

Каждое из заданий должно содержать один или несколько тестов и два скрипта: для подготовки окружения `prepare_script.py` и получения рейтинга `get_score.py`.

```
1 import os
2 from sys import stdin
3 from io import StringIO
4 from zipfile import ZipFile
5
6 with ZipFile((os.environ['ZIP_NAME'])) as z:
7     f_size = sum(e.file_size for e in z.infolist()) // 1000
8     if f_size > 4096:
9         print('zip file is too big')
10        sys.exit(1)
11    z.extractall(os.environ['ZIP_EXTRACT'])

1 import os
2 import pandas as pd
3
4 tst = os.listdir(os.environ['TESTS_DIR'])
5 usr = os.listdir(os.environ['OUTPUT'])
6
7 for t, u in zip(tst, usr):
8     t_df = pd.read_csv(os.path.join(os.environ['TESTS_DIR'], t))
9     u_df = pd.read_csv(os.path.join(os.environ['OUTPUT'], u))
10    diff = t_df.Survived.eq(u_df.Survived).mean()
11
12 print('SCORE=%s' % diff)
```

Рисунок 3.4 Скрипты «Титаник»

Их содержание может отличаться в зависимости от задания, в «Титанике» первый распаковывает архив, если он не превысил допустимый размер, а второй проверяет наборы данных u и u'

Рассмотрим как система проверяет решение, которым снабжает её пользователь:

В начале пользователь отправляет архив сабмита, это происходит путем передачи архива в виде потока данных с помощью директивы `multipart/form-data` в узел-клиент, далее архив кодируется в `base64` строку для отправки её в узел-процессор.

```
bio = io.BytesIO()
archive.save(bio)
data = {'bin' : codecs.encode(bio.getvalue(), "base64").decode(),
        'tests' : tests,
        'ps' : ps,
        'gs' : gs,
        'time' : [25, 15]}

rs = r.post('http://localhost:64544/', json=data)

state, score, report = handle_results(rs, course_uid, euid)
```

Рисунок 3.5 Обработка сабмита

После получения наборов данных y и записи их, в заранее указанный в системной переменной OUTPUT путь, узел-процессор считывает все y и с помощью библиотеки pandas вычисляет R .

```
for t in tests:
    t_name, t_ext, t_content = t
    exe_env.update({'OUTPUT' : 'results/%s_usr_out%s' % (t_name, t_ext)})
    runtime = subprocess.run(exe_script, input=t_content[0], env=exe_env, \
        shell=True, capture_output=True, text=True)
    runtime_logs.update({'t_name' : {'rcode' : runtime.returncode,
        'stderr' : runtime.stderr,
        'stdout' : runtime.stdout}})
    if runtime.returncode:
        failed = True
        break
```

Рисунок 3.6 Тестирование сабмита

Сценарий обработки может завершиться с ошибкой, как например ошибка в коде или превышение времени работы.

```
if failed:
    info = {'state' : 'FAILED',
        'logs' : runtime_logs}
    return jsonify(info)

info = {'state' : 'PASSED',
    'code' : testbed.returncode,
    'testbed_err' : testbed.stderr,
    'testbed_out' : testbed.stdout,
    'script' : script}
```

Рисунок 3.7 Получение результатов сабмита

В данном случае в результатах будет запись о неудаче, если же всё прошло успешно в таблицу оценок запишется рейтинг.

Оценки

User	Titanic
Сергеев Сергей Сергеевич	0.9712918660287081

Рисунок 3.8 Таблица оценок

```
test_data = pd.read_csv(data_test).fillna(1)

features = ["Pclass", "Sex", "SibSp", "Parch"]
X_test = pd.get_dummies(test_data[features])

predictions = model.predict(X_test)

output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': predictions})
output.to_csv(os.environ['OUTPUT'], index=False)
```

Рисунок 3.9 Скрипт для решения задачи «Титаник»

Титаник — известная задача, ориентированная в большей мере на начинающих в машинном обучении. Датасет Титаник содержит данные пассажиров корабля. Цель задачи — построить модель, которая лучшим образом сможет предсказать, остался ли произвольный пассажир в живых или нет.

Last score: 0.9712918660287081

Supplement files:

[train.csv](#)

Решение

 Файл не выбран

Рисунок 3.10 Отображение результата проверки программы

Приведем пример сабмитов для решения задачи «Титаник» с помощью RandomForest и многослойного пецептрона, проведем оценку точности с помощью confusion matrix[7] и встроенных в библиотеки keras и sklearn метрик. Также покажем как данные модели могут быть переведены в формат ONNX для запуска в системе оценки без использования пользовательских скриптов.

Опишем использованный метод для оценки тестовых моделей, confusion matrix:

В задачах статистической классификации в области машинного обучения матрица ошибки или confusion matrix позволяет визуализировать степень натренированности алгоритма, где в строке хранится информация о настоящем количестве класса, а в столбце количество предсказанных алгоритмом классов. В целом confusion matrix позволяет получить более целостное представление об обучении алгоритма, по количеству true positive, true negative предсказаний можно по-настоящему оценить качество классификатора, в то время как обычные метрики могут быть ввести в заблуждение, даже при наличие неплохих результатов, например при несбалансированном наборе данных, где один из классов будет полностью проигнорирован при достаточно высокой метрике точности классификатора.

		True value					Sum
		Y_1	...	Y_i	...	Y_k	
Predicted value	\hat{Y}_1	n_{11}	...	n_{1i}	...	n_{1k}	$n_{1\bullet}$

	\hat{Y}_i	n_{i1}	...	n_{ii}	...	n_{ik}	$n_{i\bullet}$

	\hat{Y}_k	n_{k1}	...	n_{ki}	...	n_{kk}	$n_{k\bullet}$
Sum		$n_{\bullet 1}$...	$n_{\bullet i}$...	$n_{\bullet k}$	n

Рисунок 3.11 Confusion matrix общего вида

		True value	
		P	N
Predicted value	\hat{P}	True Positive	False Positive
	\hat{N}	False Negative	True Negative

Рисунок 3.12 Двуклассовая матрица ошибок

Чем сильнее главная диагональ confusion matrix, т. е. количество правильно предсказанных классов, тем лучше качество модели-классификатора. В данной работе используется нормализованная confusion matrix, отображающая процент правильно классифицированных классов.

Рассмотрим код для графического отображения confusion matrix.

```

def plotconfmatrix(y_pred, y_test, plotname):
    matrix = confusion_matrix(y_pred, y_test)
    matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

    plt.figure(figsize=(16,7))

    sns.set(font_scale=1.4)
    sns.heatmap(matrix, annot=True, annot_kws={'size':30},
                cmap=plt.cm.Greens, linewidths=0.2)

    class_names = ['0', '1']
    tick_marks = np.arange(len(class_names))
    tick_marks2 = tick_marks + 0.5
    plt.xticks(tick_marks, class_names, rotation=25)
    plt.yticks(tick_marks2, class_names, rotation=0)
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.title('Confusion Matrix for %s Model' % (plotname))

    plt.show()

```

Рисунок 3.13 Код для отображения confusion matrix

Деревья принятия решений наиболее изученный класс нейросетевых моделей и является членом так называемого стандартного машинного обучения, наряду с SupportVectorModels и kMeans классификатором. DecisionTrees отображают входные характеристики в выходные классы по определенным правилам. Модель обучается путем рекурсивного выбора особенности и границы, увеличивающие общий объем информации. Другими словами, DecisionTree ищут границы, минимизирующие энтропию рабочего множества. Само по себе дерево решений имеет свои недостатки - подверженность чрезмерной подгонке и чувствительность к небольшим изменениям во входных данных. Однако эти эффекты можно смягчить, используя их объединение. В ансамбле, где окончательная классификация определяется большинством голосов многих деревьев, содержащиеся в модели. Ключевое преимущество алгоритмов ансамблевого дерева решений, таких как случайных лесов, заключается в том, что достоверность любой классификации можно напрямую интерпретировать из пропорции решений отдельных деревьев. Для обучения модели был использована модель RandomForest[9] из фреймворка scikit-learn.

```

from skl2onnx import convert_sklearn
from skl2onnx.common.data_types import Int64TensorType, FloatTensorType
import onnxruntime as rt
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import plot_confusion_matrix

train_data = pd.read_csv('train.csv').fillna(1).replace("male", 0).replace("female", 1)

test_datax = pd.read_csv('test_in.csv').fillna(1).replace("male", 0).replace("female", 1)
test_datay = pd.read_csv('test_out.csv').fillna(1).replace("male", 0).replace("female", 1)

y = train_data["Survived"]

features = ["Pclass", "Sex", "SibSp", "Parch"]
X = pd.get_dummies(train_data[features])
X_test = pd.get_dummies(test_datax[features])
y_test = test_datay["Survived"]

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(X, y)
predictions = model.predict(X_test)

```

Рисунок 3.14 Код для модели RandomForest

Давайте рассмотрим результаты обучения, исследуя одно из деревьев решений случайного леса. Алгоритм дерева решений состоит из таких критериев разделения, как:

- Примесь Джини

Определяет частоту появления неправильно классифицированного элемента

- Энтропия

Метрика случайности в дереве решений

- Дисперсия
- Полученная информация

Статистическое свойство, описывающее, насколько хорошо данный атрибут разделяет обучающие примеры в соответствии с их классификацией в выходном наборе данных.

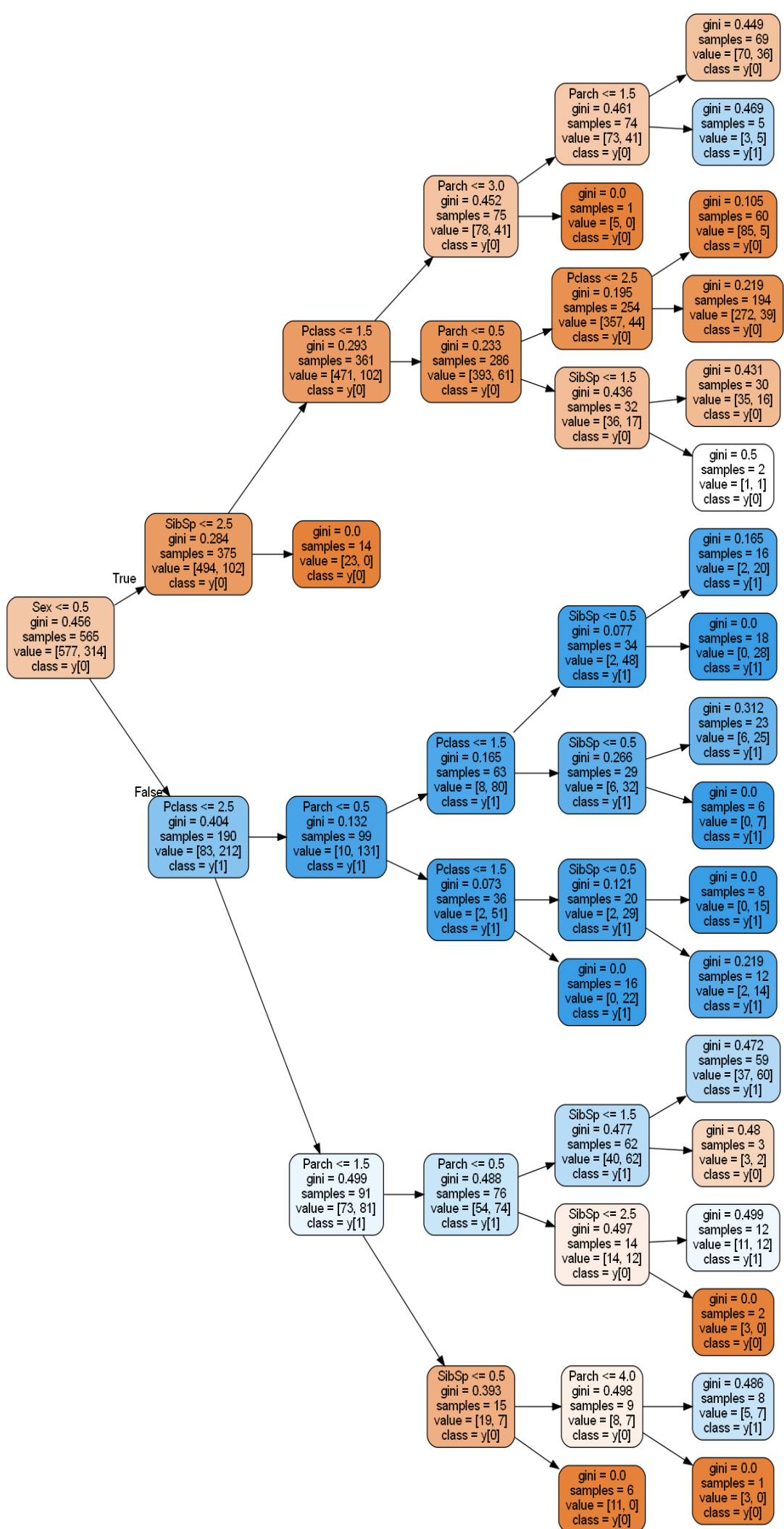


Рисунок 3.15 Отображение одного из деревьев решений внутри RandomForest

Оценим результаты обучения модели на тестовом наборе данных, как видно, присутствует сильная главная диагональ

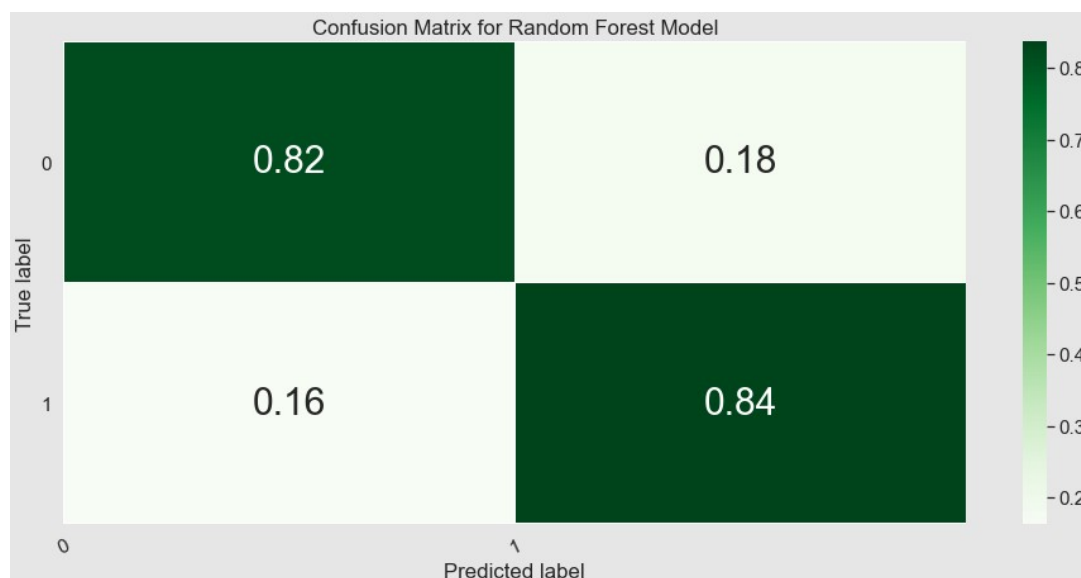


Рисунок 3.16 ConfusionMatrix для RandomForest

Многослойный перцептрон[10] — самая простая модель нейронных сетей и наиболее часто встречается на курсах обучения искусственному интеллекту, в отличие от двуслойного перцептрона, имеющего проблемы с обучением сложным функциям, продемонстрированные Minsky и Papert(1969), многослойные сети прямого распространения позволяют аппроксимировать любую измеримую функцию[16]. Это свойство означает, что при работе с этой моделью нейронных сетей любая неудача при экспериментах заключается в неправильном обучении, мало или наоборот слишком большом количестве скрытых слоев или присутствие стохастической нежели детерминистской связи между входными и выходными данными. Эта идея необходима для дальнейшего понимания и изучения других архитектур нейронных сетей, таких, как глубокие и сверточные нейронные сети, а также исследований в искусственном интеллекте в целом.

Реализация этой модели с помощью фреймворка Keras и оценка точности с помощью confusion matrix приведена далее.

Keras является одним из наиболее продвинутых фреймворков в сфере нейронных сетей[6] в плане быстродействия, правильной утилизации выделенных ресурсов, а также удобства использования его API, благодаря

высокому уровню абстракции в отличие от других фреймворков. Для его использования требуется значительно меньшее число строк кода. Keras удобен в освоении, благодаря исчерпывающей и простой в использовании документации. Keras имеет мощные встроенные функции для мониторинга прогресса обучения и реализации таких показателей, как показатель точности. Слои, предоставляемые Keras, покрывают практически все требования для построения специализированной нейронной сети. Кроме того, Keras предоставляет множество слоев для настройки модели с особенной архитектурой. Существует множество руководств и ресурсов, которые могут помочь в разработке моделей глубокого обучения. Для решения представленной использовалась линейная модель, которая представляет собой последовательный набор слоев.

```
def get_model():  
  
    ssize = 4  
  
    model = Sequential()  
  
    model.add(Dense(units=16, activation="relu", input_dim=ssize))  
    model.add(Dense(units=8, activation="relu"))  
    model.add(Dense(units=4, activation="relu"))  
    model.add(Dense(units=1, activation="relu"))  
  
    model.compile(  
  
        loss = tf.keras.losses.BinaryCrossentropy(),  
        optimizer = 'Adam',  
        metrics=['accuracy']  
    )  
  
    return model
```

Рисунок 3.17 Код для модели MLP

Рассмотрим архитектуру получившейся модели.

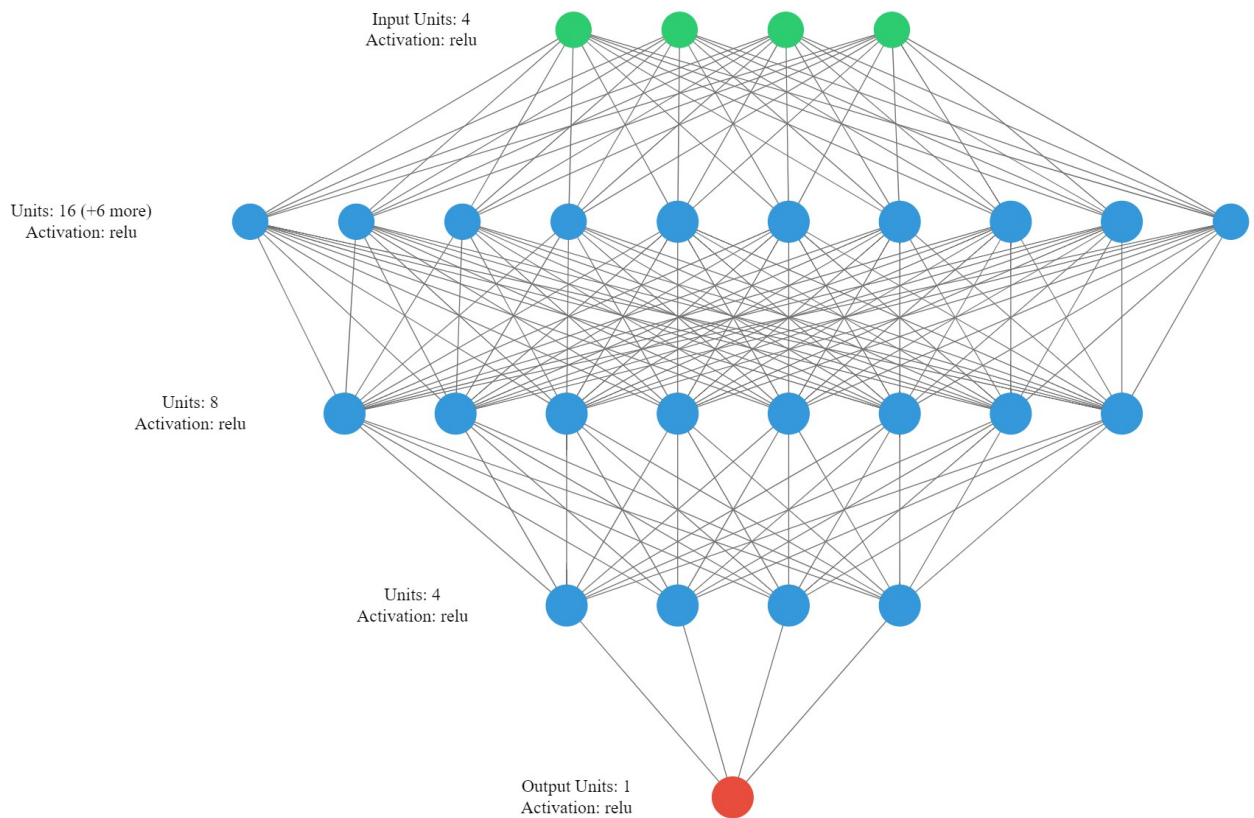


Рисунок 3.18 Отображение архитектуры MLP

В отличие от `scikit-learn` моделей, которые подстраивались в данном случае под данные на котором они обучались, Keras использует числа с плавающей точкой для результатов предсказаний, что потребовало дополнительной обработки с помощью библиотеки `numpy`.

```
model_basic = get_model()
rs_basic = model_basic.fit( X, y, validation_split=0.2, epochs=250, batch_size=32, verbose=0, callbacks=[])
predictions1 = model_basic.predict(X_test)
predictions11 = np.round(predictions1.flatten()).astype(np.int64)
```

Рисунок 3.19 Код для получение предсказаний из модели MLP

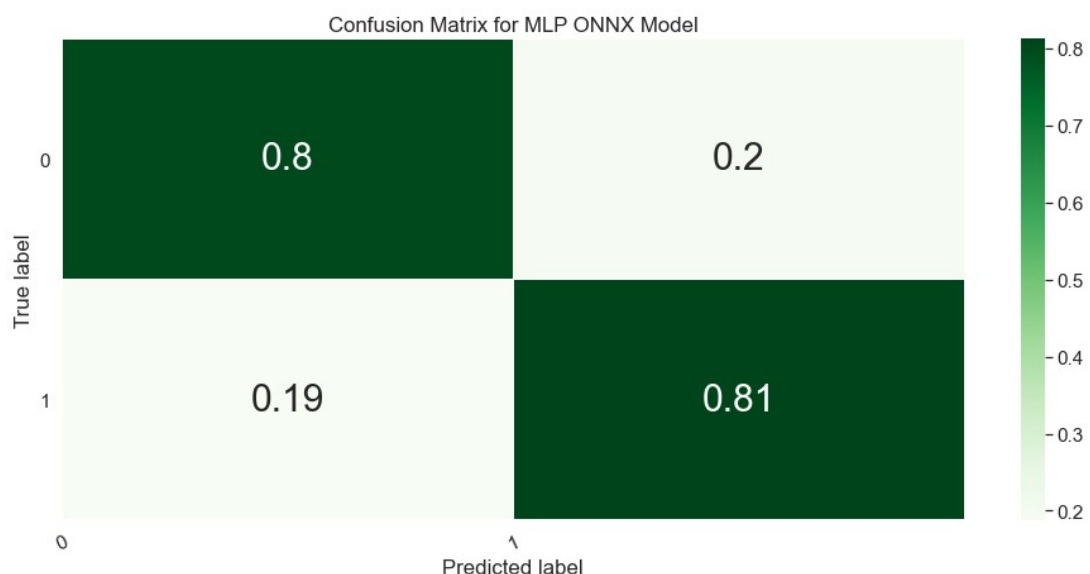


Рисунок 3.20 ConfusionMatrix для модели MLP

Как было рассмотрено ранее для использования системой модели

пользователю требовалось снабдить её скриптом для десериализации pickle объекта в модель, однако с помощью формата onnx достаточно отправить сконвертированную модель в систему, а далее узел-процессор сможет обработать её, вне зависимости какой фреймворк использовался для создания модели.

```
def onnxeval.mdlstr, X_eval):
    sess = rt.InferenceSession.mdlstr)
    input_name = sess.get_inputs()[0].name
    label_name = sess.get_outputs()[0].name
    return sess.run([label_name], {input_name: X_eval})
```

Рисунок 3.21 Код для получения предсказаний из сериализованной модели

Преобразуем модели RandomForest и MLP в формат ONNX и оценим их точность на том же тестовом наборе данных. Для того, чтобы сконвертировать модель нужно указать тип и размерность входного слоя, в данном случае используется 4 параметра из набора данных «Титаник»: Pclass — класс, к которому принадлежал пассажир, Sex — пол пассажира, SibSp — число братьев, сестер или супругов, присутствовавших вместе с пассажиром на рейсе и Parch — количество родителей или детей с которыми путешествовал пассажир.

```
initial_type = [('X', FloatTensorType([None, 4]))]
mdl_onx = convert_sklearn(model, initial_types=initial_type).SerializeToString()
spr = onnxeval(mdl_onx, X_test.to_numpy().astype(np.float32))[0]
plotconfmatrix(spr, y_test, "Random Forest ONNX")
```

Рисунок 3.22 Получение результатов сериализованного RandomForest

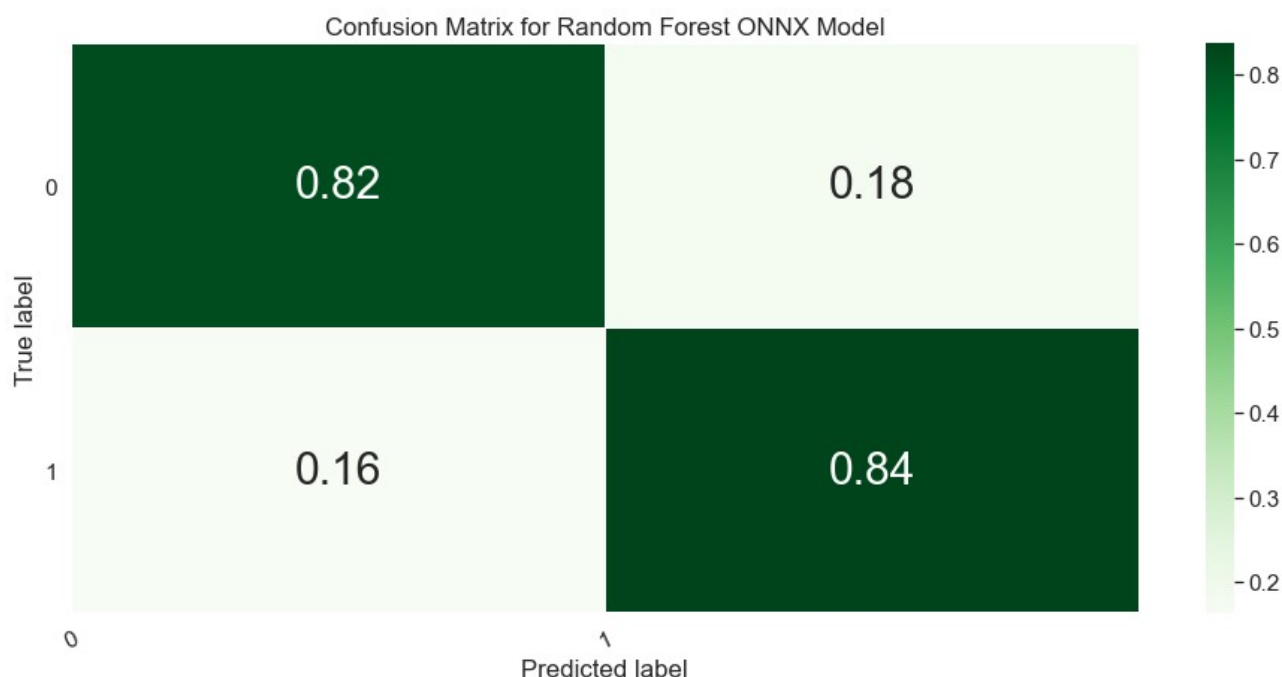


Рисунок 3.23 ConfusionMatrix для сериализованной модели RandomForest

Заметим, что для модели Keras потребовалось преобразовать результаты

предсказания с помощью библиотеки `numru`, те же преобразования можно применять и к результатам работы сериализованной модели `RandomForest`, что в итоге не нарушает универсальность запуска и обработки моделей в формате `ONNX`.

```
onnx_model = keras2onnx.convert_keras(model_basic, model_basic.name).SerializeToString()
spr = onnxeval(onnx_model, X_test.to_numpy().astype(np.float32))[0]
spr1 = np.round(spr.flatten()).astype(np.int64)
```

Рисунок 3.24 Получение результатов сериализованного MLP

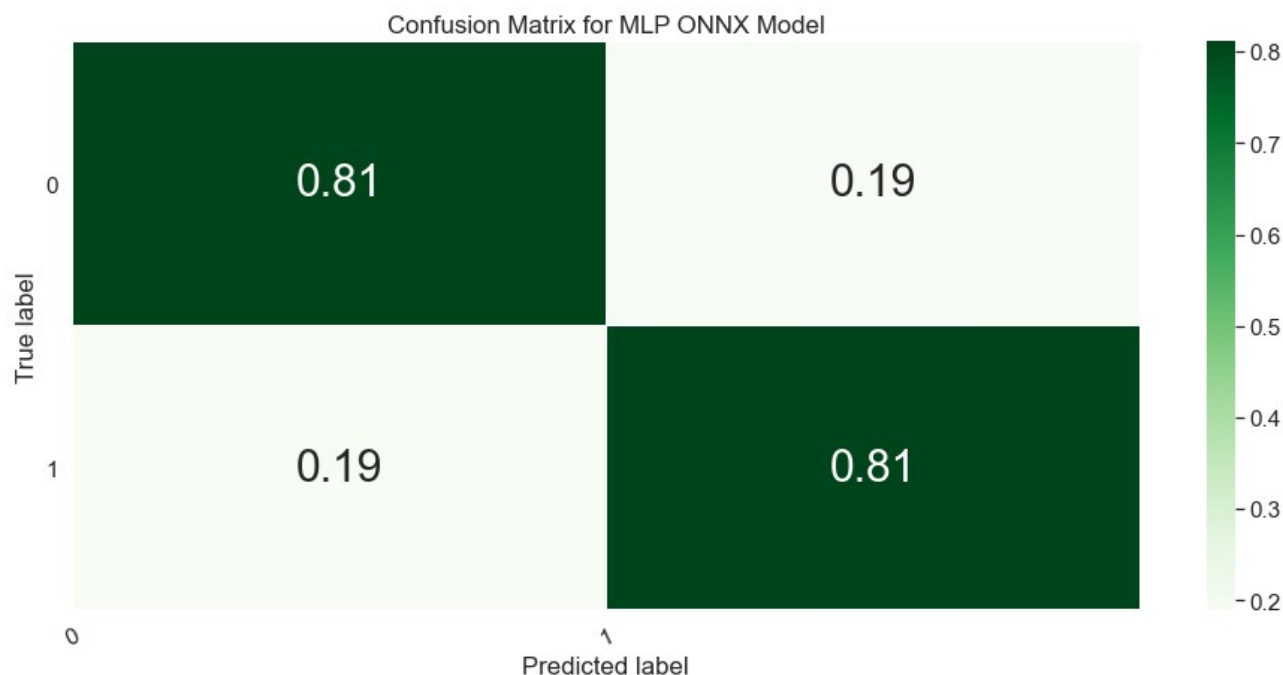


Рисунок 3.25 ConfusionMatrix для сериализованной модели MLP

Заключение

Обучение дисциплинам, связанных с компьютерными науками не может обходиться без внедрения ЭВМ в учебный процесс, будь то в качестве инструмента для выполнения задачи или же одного из участников организации учебного процесса. Однако отсутствие решения с открытым исходным кодом для автоматического тестирования моделей искусственного интеллекта ставит необходимость реализации такой системы. Пандемия COVID-2019 также сделала немаловажным использование автоматизированных систем, как для предотвращения риска дальнейшего распространения вируса в связи с очным видом обучения, так и снижение иммунитета из-за стресса, связанного с дистанционным обучением у преподавательского состава и студентов.

Была проведена работа по проектированию и созданию системы автоматического тестирования для проверки решений, использующих

нейронные сети и искусственного интеллекта в целом.

Была продемонстрирована применимость такого типа систем для решения проблем, связанной с трудоемкой проверкой нейросетевых задач.

Список литературы

1. Kurnia, A., Lim, A., & Cheang, B. (2001). *Online Judge. Computers & Education*, 36(4), 299–315.
2. <http://www.kremlin.ru/events/president/news/64545>
3. Moutsatsos, I. K., Hossain, I., Agarinis, C., Harbinski, F., Abraham, Y., Dobler, L., ... Parker, C. N. (2016). *Jenkins-CI, an Open-Source Continuous Integration System, as a Scientific Data and Image-Processing Platform. SLAS DISCOVERY: Advancing Life Sciences R&D*, 22(3), 238–249. doi:10.1177/1087057116679993
4. Tadlaoui, M. A. and Mohamed Chekou. “A blended learning approach for teaching python programming language: towards a post pandemic pedagogy.” (2021).
5. Leite Leonardo, Rocha Carla, Kon Fabio, Milojicic Dejan, Meirelles Paulo, "A Survey of DevOps Concepts and Challenges," ACM Computing Surveys (CSUR), 10 December 2019, Vol.52(6), pp.1-35.
6. Elshaw, R., Wahab, A., Barnawi, A. et al. DLBench: a comprehensive experimental evaluation of deep learning frameworks. Cluster Comput (2021). <https://doi.org/10.1007/s10586-021-03240-4>
7. I. Düntsch and G. Gediga, “Confusion Matrices and Rough Set Data Analysis,” J. Phys. Conf. Ser., vol. 1229, pp. 1–6, 2019, doi: 10.1088/1742-6596/1229/1/012055.
8. Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. <https://github.com/onnx/onnx>, 2019
9. Louppe G (2014) Understanding random forests: from theory to practice. Doctoral dissertation, University of Liège
10. Rosenblatt, F.. “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review* 65 6 (1958): 386-408 .
11. Xiaolin Geng, Xuewen Zeng, Linlin Hu and Zhichuan Guo, An novel Architecture and Inter-process Communication Scheme to Adapt Chromium Based on Docker Container, Inter.CongressInfo. Comm.Tech. (2017) 691-696
12. Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., & Zhou, W. (2018). A Comparative Study of Containers and Virtual Machines in Big Data Environment. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 178-185.

13. Edward Verenich, Alvaro Velasquez, M. G. Sarwar Murshed, & Faraz Hussain. (2020). FlexServe: Deployment of PyTorch Models as Flexible REST Endpoints.
14. Dataanyze. 2021. MySQL Market Share and Competitor Report. <https://www.datanyze.com/market-share/databases--272/mysql-market-share>.
15. Joseph M. Hellerstein. (2019). Looking Back at Postgres.
16. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366. doi:10.1016/0893-6080(89)90020-8
17. Thomas Rausch, Waldemar Hummer, Vinod Muthusamy, Alexander Rashed, & Schahram Dustdar (2019). Towards a Serverless Platform for Edge AI. In *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*. USENIX Association.

Приложение А

db.sql

```
CREATE TABLE IF NOT EXISTS usr (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `login` varchar(255),  
    `password` varchar(255) NOT NULL,  
    `public_name` varchar(255),  
    `uuid` varchar(255),  
    primary key(id)  
) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

```
CREATE TABLE IF NOT EXISTS exercise (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `name` varchar(255),  
    `lang` varchar(255),  
    `desc` text,  
    `uuid` varchar(255),  
    `being_edited` boolean default true,  
    `is_file` boolean default false,  
    primary key(id)  
) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

```
CREATE TABLE IF NOT EXISTS course (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `name` varchar(255),  
    `desc` varchar(255),  
    `uuid` varchar(255),  
    primary key(id)  
) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

```
CREATE TABLE IF NOT EXISTS ex_usr_rel (  
    `ex_id` int(11) NOT NULL,  
    `usr_id` int(11) NOT NULL,  
    `last_submitted` TIMESTAMP(6),  
    `state` varchar(255),  
    `score` varchar(255),  
    `report` text,  
    primary key(ex_id, usr_id),  
    foreign key(ex_id) references exercise (id) on delete cascade,  
    foreign key(usr_id) references usr (id) on delete cascade  
);
```

```
CREATE TABLE IF NOT EXISTS admin_usr_rel (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `usr_id` int(11) NOT NULL,  
    primary key(id),  
    foreign key(usr_id) references usr (id) on delete cascade  
);
```

```
CREATE TABLE IF NOT EXISTS course_admin_rel (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `course_id` int(11) NOT NULL,
    `admin_id` int(11) NOT NULL,
    primary key(id),
    foreign key(course_id) references course (id) on delete cascade,
    foreign key(admin_id) references admin_usr_rel (usr_id) on delete cascade
);
```

```
CREATE TABLE IF NOT EXISTS course_usr_rel (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `course_id` int(11) NOT NULL,
    `usr_id` int(11) NOT NULL,
    primary key(id),
    foreign key(course_id) references course (id) on delete cascade,
    foreign key(usr_id) references usr (id) on delete cascade
);
```

```
CREATE TABLE IF NOT EXISTS course_ex_rel (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `course_id` int(11) NOT NULL,
    `ex_id` int(11) NOT NULL,
    primary key(id),
    foreign key(course_id) references course (id) on delete cascade,
    foreign key(ex_id) references exercise (id) on delete cascade
);
```

```
CREATE TABLE IF NOT EXISTS link_course_rel (
    `link_id` varchar(8),
    `course_id` int(11) NOT NULL,
    foreign key(course_id) references course (id) on delete cascade,
    primary key (`link_id`)
);
```

app.py

```
from flask import Flask, request, jsonify, Blueprint
from flask_cors import CORS
import json, subprocess, os
import shutil, datetime
import io, codecs

app = Flask(__name__, static_url_path='/static')
CORS(app)

@app.route('/isup', methods=['GET'])
def isup():
    rs = subprocess.run('python3 -V', capture_output=True, shell=True)
    if rs.returncode:
        return 'ok', 403
    return 'ok'

@app.route('/', methods=['POST'])
def ml_route():
    if not request.json:
        return 403, 'post json'

    ps = request.json['ps']
    gs = request.json['gs']
    tests = request.json['tests']
    pld = request.json.get('bin')

    name = 'dir%s' % datetime.datetime.now().microsecond

    os.mkdir('%s' % name)
    os.chdir('%s' % name)

    max_time, min_time = request.json['time']
    os.mkdir('tdir')
    os.mkdir('out')

    if pld:
        bio = io.BytesIO(codecs.decode(pld.encode(), "base64"))
        with open('pld.zip', 'wb') as f:
            f.write(bio.getbuffer())
    else:
        script = request.json.get('script')
        with open('out/script.py', 'w+') as f:
            f.write(script)

    with open('ps.py', 'w+') as f:
        f.write(ps)
    with open('gs.py', 'w+') as f:
```

```

f.write(gs)

for t in tests:
    t_name, t_ext, t_content = t
    c = t_content[1]
    with open(os.path.join('tdir', t_name + '_out_' + t_ext), 'w+') as f:
        f.write(c)

ps_env = os.environ.copy()

ps_env.update({'ZIP_NAME' : 'pld.zip', 'ZIP_EXTRACT' : 'out'})

ps_script = 'python3 ps.py'

ps_runtime = subprocess.run(ps_script, capture_output=True, shell=True,
env=ps_env)

exe_env = {'PATH' : '/home/user/miniconda/bin:/usr/bin'}

os.chdir('out')
os.mkdir('results')
exe_script = 'timeout -k %s %s python3 script.py' % (max_time,
min_time)

global runtime_logs
runtime_logs = {}
failed = False

for t in tests:
    t_name, t_ext, t_content = t
    exe_env.update({'OUTPUT' : 'results/%s_usr_out%s' % (t_name,
t_ext)})
    runtime = subprocess.run(exe_script, input=t_content[0], env=exe_env,
shell=True, capture_output=True, text=True)
    runtime_logs.update({t_name : {'rcode' : runtime.returncode,
'stderr' : runtime.stderr,
'stdout' : runtime.stdout}})
    if runtime.returncode:
        failed = True
        break

os.chdir('../')

gs_env = os.environ.copy()
gs_env.update({'TESTS_DIR' : 'tdir', 'OUTPUT' : 'out/results'})

test_script = 'timeout -k 120 110 python3 gs.py'

testbed = subprocess.run(test_script, env=gs_env, shell=True,
capture_output=True, text=True)

```

```
#subprocess.run('rm %s.py' % name, shell=True)
#os.rmdir('%s' % name)
with open('out/script.py') as f:
    script = f.read()
```

```
os.chdir('../')
shutil.rmtree('%s' % name, ignore_errors=True)
```

```
if failed:
    info = {'state' : 'FAILED',
           'logs' : runtime_logs}
    return jsonify(info)
```

```
info = {'state' : 'PASSED',
        'code' : testbed.returncode,
        'testbed_err' : testbed.stderr,
        'testbed_out' : testbed.stdout,
        'script' : script}
```

```
return jsonify(info)
```

Приложение Б

Dockerfile

```
FROM debian:stable

RUN apt-get update && apt-get install -y \
    curl \
    ca-certificates \
    sudo \
    git \
    bzip2 \
    libx11-6 \
    && rm -rf /var/lib/apt/lists/*

RUN mkdir /app
WORKDIR /app

# Create a non-root user and switch to it
RUN adduser --disabled-password --gecos " --shell /bin/bash user \
    && chown -R user:user /app
RUN echo "user ALL=(ALL) NOPASSWD:ALL" > /etc/sudoers.d/90-user
USER user

# All users can use /home/user as their home directory
ENV HOME=/home/user
RUN chmod 777 /home/user

ENV CONDA_AUTO_UPDATE_CONDA=false
ENV PATH=/home/user/miniconda/bin:$PATH
RUN curl -sLo ~/miniconda.sh
https://repo.continuum.io/miniconda/Miniconda3-py38_4.8.2-Linux-x86_64.sh \
    && chmod +x ~/miniconda.sh \
    && ~/miniconda.sh -b -p ~/miniconda \
    && rm ~/miniconda.sh \
    && conda install -y python==3.8.1 \
    && conda clean -ya

# CUDA 10.2-specific steps
RUN conda install -y -c pytorch \
    cudatoolkit=10.2 \
    "pytorch=1.5.0=py38_cuda10.2.89_cudnn7.6.5_0" \
    "torchvision=0.6.0=py38_cu102" \
    && conda clean -ya

ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8
```

```
RUN conda install -y -c anaconda flask \  
    flask-cors \  
    gunicorn \  
    pandas \  
&& conda clean -ya
```

```
RUN conda install -y scikit-learn && conda clean -ya
```

```
COPY ./app.py ./app.py
```

```
ENTRYPOINT gunicorn --workers=4 --threads 10 -b 0.0.0.0:64544 app:app
```