РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА ПО ДИСЦИПЛИНЕ «МАШИННОЕ ОБУЧЕНИЕ И БОЛЬШИЕ ДАННЫЕ».
Тема: «Разработка Web-приложения (дашборда) для инференса (вывода) моделей ML и анализа данных»

Вывод машинного обучения (Machine learning inference)[1] включает в себя применение модели ML к набору данных и создание выходных данных или «прогноза». Обычно модель ML представляет собой программный код, реализующий математический алгоритм. Процесс вывода ML развертывает этот код в производственной среде, что позволяет генерировать прогнозы для входных данных, предоставленных реальными конечными пользователями.

Жизненный цикл ML состоит из двух основных частей:

1. **Фаза обучения** — включает в себя создание модели ML, ее обучение путем запуска модели на примерах помеченных данных, а затем тестирование и проверку модели путем ее запуска на невидимых примерах.
2. **Выводы ML** — включают в себя использование модели на реальных данных для получения действенных результатов. На этом этапе система вывода принимает входные данные от конечных пользователей, обрабатывает данные, передает их в модель ML и передает выходные данные обратно пользователям.

Аналитические информационные панели (дашборд) — это отличный способ для специалистов по данным передавать информацию компаниям, но их создание часто может быть дорогостоящим и трудоемким. Streamlit — это простая в использовании библиотека на основе Python, которая упрощает этот процесс.

Streamlit — это библиотека на основе Python, которая позволяет специалистам по данным легко создавать бесплатные приложения ML. Streamlit позволяет отображать описательный текст и выходные данные модели ML, визуализировать данные и производительность модели ML, а также изменять входные данные модели ML через пользовательский интерфейс с помощью боковых панелей.

ЗАДАНИЕ.

Используя возможности библиотеки Streamlit, создать Web-приложение (дашборд), основной целью которого является инференс (вывод) моделей ML и результатов анализа данных с различными вариантами их визуализации.

1. Выбрать **один из двух наборов данных**, соответствующих Вашему варианту (для регрессии или классификации). Среди всех построенных в ходе выполнения лабораторных работ моделей ML **выбрать шесть лучших модели** (на основе значений коэффициента детерминации для регрессии; F1 для классификации):

---

[1] Understanding Machine Learning Inference. – URL: https://www.run.ai/guides/machine-learning-inference/understanding-machine-learning-inference

Модель ML1: классическая модель обучения с учителем (например, линейная и полиномиальная регрессия с регуляризаторами и без них, деревья принятия решений, KNN, SVM, логистическая регрессия, байесовский классификатор).

Модель ML2: классическая модель обучения без учителя (например, любой алгоритм кластеризации, алгоритм PCA).

Модель ML3: ансамблевая модель (градиентный бустинг).

Модель ML4: ансамблевая модель (бэггинг).

Модель ML5: ансамблевая модель (стэкинг).

Модель ML6: глубокая полносвязная нейронная сеть.

***Допускается и приветствуется использование моделей ML с собственной реализацией (дополнительные баллы).**

2. **Сериализовать (сохранить) выбранные модели ML на жесткий диск**. При *сериализации моделей ML* TensorFlow, XGBoost и CatBoost использовать встроенные инструменты. При сериализации моделей ML Sklearn (или собственных моделей ML) использовать стандартный модуль **pickle**[2] в Python.

3. С помощью библиотек Streamlit (можно Dash) реализовать Web-приложение (дашборд).

Структура Web-приложения (дашборд):

- **Web-страница1 с информацией о разработчике моделей ML** (ФИО, номер учебной группы и цветная фотография, тема РГР);
- **Web-страница2 с информацией о наборе данных** (чему посвящен датасет (т.е. тематика или предметная область), описание признаки, особенности предобработки данных);
- **Web-страница3 с визуализациями зависимостей** в наборе данных (визуализации Matplotlib, Seaborn, минимум **4 различных вида визуализаций**);
- **Web-страница4,** с помощью которой можно получить **предсказание соответствующей модели ML** (см. п. 1): реализовать загрузку файла в формате *.csv, сделать ввод соответствующих данных с использованием интерактивных компонентов и валидации.

РЕКОМЕНДАЦИИ К ОФОРМЛЕНИЮ ОТЧЕТА РГР:

1. Выполнить форматирование текста отчета в соответствии с требования к оформлению текста в отчете курсового проекта.
2. В **отчете** описать ход работы над заданием РГР. Ниже представлена структура отчета РГР.

Титульный лист.

Содержание. Создать автоматизированное оглавление отчета РГР.

Введение (1 стр.). Отразить актуальность темы РГР и применяемый для ее реализации стек технологий, конкретные изученный инструменты (библиотеки), (например, Sklearn, TensorFlow и др.) и отметить их роль в решении задач ML.

---

[2] Модуль **pickle** реализует алгоритм сериализации и десериализации объектов Python.

Основная часть. Включает в себя как минимум два пункта, в которых приветствуются подпункты (1.1, 1.2 и т. д.):

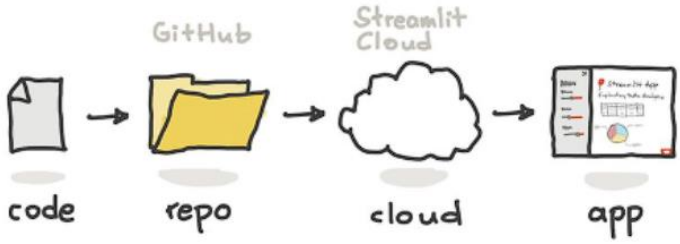
▪ Пункт 1. Описываются выбранные модели, процесс их сериализации (краткое описание самих моделей и используемых инструментов сериализации);

▪ Пункт 2. Описывается последовательно процесс разработки дашборда, приводятся небольшие фрагменты исходного кода. В пункте 2 сформировать ссылку на GitHub, куда предварительно размещены все исходные файлы РГР.

Таким образом, должен быть репозиторий на GitHub, в котором лежит ***readme*** с подробным описанием проекта РГР.

Мы ожидаем от вас в качестве результатов две ссылки:

- ссылка на ваш открытый GitHub-репозиторий,
- ссылка на веб-сервис (например, выложенный на Streamlit Cloud, https://streamlit.io/cloud)



Заключение (1 – 1,5 стр.). Описать, что было сделано в рамках РГР (выводы).

Список использованных источников. Список использованных ссылок на Web-ресурсы и литературы выполняется в виде нумерованного списка в соответствии с действующим **ГОСТом Р 7.0.100–2018** к библиографическому описанию На каждый источник в тексте отчета РГР должна быть ссылка, например, вида [1] в конце приложения, где 1 - номер источника. Количество ссылок в библиографическом списке должно быть не менее 3.

Приложения:

Приложение А содержит скрины Web-страниц приложения дашбрда в моменте исполнения.

Приложение Б содержит исходный код разработанного Web приложения.

3. Файл отчета сохранить под именем **РГР_Фамилия студента_Группа.docx** (например, **РГР_Иванов_МО-221.docx**).

4. После выполнения и оформления отчета РГР необходимо:

4.1. Распечатать титульный лист.

4.2. В строке **выполнил** студент должен поставить свою подпись.

4.3. В строке **принял** преподаватель должен поставить свою подпись.

4.4. Произвести сканирование или фотографирование титульного листа.

4.5. Заменить титульный лист без подписей на титульный лист с подписями в файле отчета РГР.

4. 6. Файл отчета РГР сохранить в формате PDF.

4.7. Выполненную РГР (файл отчета в формате PDF) разместить в своем личном кабинете на сайте ОмГТУ.

ПОДГОТОВКА РАБОЧЕЙ СРЕДЫ

Streamlit – это библиотека Python с открытым исходным кодом, которая очень популярна среди разработчиков в области ML. Ее преимущество перед другими инструментами в том, что для работы с ней не требуются дополнительные знания в области Web-технологий. Достаточно уверенного владения Python.

Подготовка среды для работы (следующие команды выполняем в терминале).

- Создать в выбранной для проекта папке, виртуальное окружение:

```
$ python3 -m venv venv
```

- Активировать окружение:

```
$ source venv/bin/activate
```

- Обновить пакетный менеджер pip:

```
$ pip install --upgrade pip
```

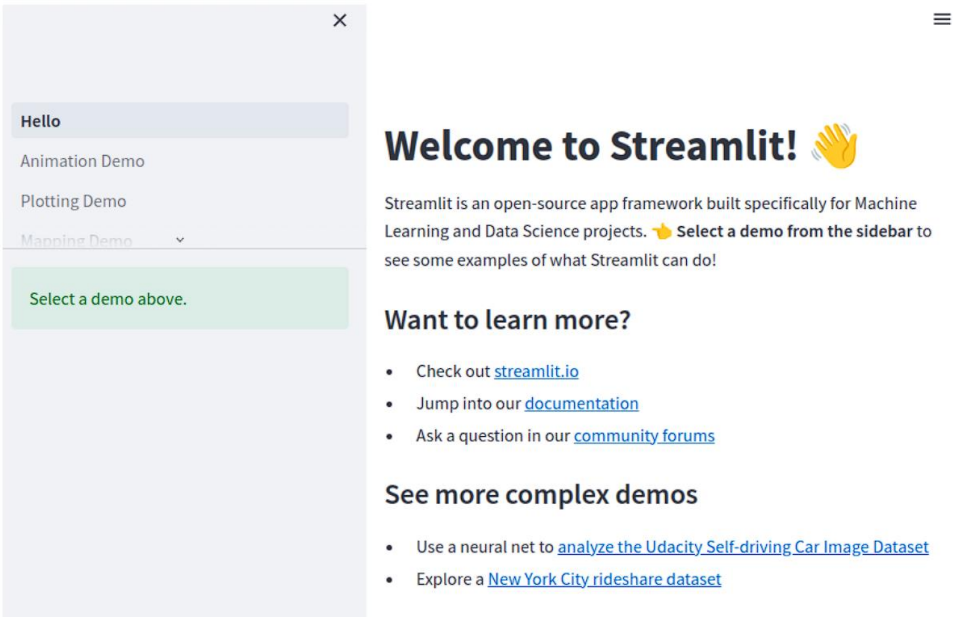
- Установить фреймворк Streamlit:

```
$ pip install streamlit
```

- Чтобы убедиться в корректности установки, запустим демо библиотеки:

```
$ streamlit hello
```

Если все отработало верно, то откроется страница демонстрационного приложения, в котором можно познакомиться с возможностями, предоставляемыми Streamlit:



ПОЛЕЗНЫЕ ИСТОЧНИКИ ПО РГР

Материалы по Streamlit

1. Документация Streamlit. – URL: https://streamlit.io
2. Как создать свое первое приложение на Streamlit. – URL: https://docs.streamlit.io/library/get-started/create-an-app
3. Начало работы с библиотекой Streamlit. Get started. – URL: https://docs.streamlit.io/en/stable/getting_started.html#create-your-firststreamlit-app
4. Описание виджетов библиотеки streamlit. Display interactive widgets. – URL: https://docs.streamlit.io/en/stable/api.html#display-interactive-widgets
5. Описание графических возможностей библиотеки Streamlit. Display charts. – URL: https://docs.streamlit.io/en/stable/api.html#displaycharts
6. Основные функции фреймворка с примерами кода. – URL: https://docs.streamlit.io/library/api-reference
7. Развертывание моделей машинного обучения. Часть первая. Размещаем Web-приложение в облачной платформе Heroku. – URL: https://habr.com/ru/articles/664076/
8. Python Serialization. – URL: https://pythongeeks.org/python-serialization/

Примеры по Streamlit

1. Streamlit. Поиск кратчайшего пути. – URL: https://habr.com/ru/articles/568836/
2. Streamlit Tutorial: A Beginner's Guide to Building Machine Learning-Based Web Applications in Python. – URL: https://builtin.com/machine-learning/streamlit-tutorial
3. Web-приложение для предсказания цены на недвижимость на Python и Streamlit. – URL: https://www.youtube.com/watch?v=ZjxPMwL552s&ab_channel=miracl6
4. Галерея приложений реализованных на Streamlit (со ссылками на репозитории проектов, очень полезно!) – URL: https://streamlit.io/gallery
5. Как написать веб-приложение для демонстрации data science-проекта на Python. – URL: https://blog.skillfactory.ru/kak-napisat-veb-prilozhenie-dlya-demonstratsii-data-science-proekta-na-python/
6. Учебный проект на Python: интерфейс в 40 строк кода. Ч. 2. – URL: https://habr.com/ru/company/skillfactory/blog/509340/