Структура

Основой тестового задания является решение некоторой целевой задачи (например задачи классификации/регрессии/сегментации).
Разработка проекта может состоять из двух частей:
1) подготовка инструментов решения целевой задачи. Как правило, это обучение искусственной нейронной сети, включающее следующие этапы: подготовка/валидация данных для обучения, выбор и обоснование подходящего инструмента (архитектура модели искусственной нейронной сети), обучение модели (baseline), корректировка данных и модели для увеличения точности (аугментация данных для обучения, подбор параметров модели, использование пред обученных весов и т.п.). В зависимости от целевой задачи могут использоваться альтернативные подходы (алгоритмы классического computer vision, boosting алгоритмы и т.п.);
2) реализация веб-приложения (или десктопного приложения) для демонстрации работы полученных инструментов решения целевой задачи.

Целевые задачи

Предсказание доходов ресторана

Цель: подготовить модели для предсказания доходов, оценить их результаты, предложить варианты улучшения.
Данные:
https://drive.google.com/file/d/1H1vlZItCcFjxwKEBDer8gwlwvbhHN0sc/view

Задачи:
1. Исследовать текущие актуальные модели по данной задаче.
2. Распарсить данные в удобный для обучения вид. Данные разбить на трэйн, валид и тест. Для данных провести feature selection.
3. Обучить регрессор с архитектурами типа XGBoost, RandomForest, etc. (на выбор, выбор обосновать)
4. Снять метрики точности на тестовых данных. Построить график для сравнения предсказанных и реальных данных.
5. Сравнить результаты с текущими актуальными моделями.
6. Предложить свои варианты улучшения метрик точности результатов.
Примечания:
1. При отсутствии необходимых мощностей gpu. Использовать средства Google Colab, либо AWS cloud
2. Рекомендуемый фреймворк для использования Keras+Tensorflow2.0.

Приложение для демонстрации

Общие требования:
1) язык интерфейса - English;
2) логирование в файл и консоль (с помощью модуля logging);
3) настроить авторизацию и регистрацию пользователей;
4) наличие формы для заполнения исходных данных (например загрузка изображения и ввод некоторой дополнительной информации);
5) возможность сохранить полученные результаты на диск (например в форматах pdf и json);
6) если целевая задача связана с компьютерным зрением, в приложении также необходимо визуализировать результат на исходном изображении. Полученное изображение отобразить пользователю;
7) данные о зарегистрированных пользователях и результаты проведенных вычислений необходимо сохранять в базу данных;
8) для каждого пользователя хранить количество проведенных вычислений.

Основной вариант использования:
1) пользователь открывает приложение;
2) пользователь проходит авторизацию/регистрацию;
3) после авторизации происходит переход на главную страницу (главное меню). Здесь пользователь может перейти либо в свой аккаунт (там отображается информация о пользователе с возможностью редактирования), либо на страницу с формой для проведения вычислений.
4) пользователь переходит на страницу с формой, загружает исходные данные и запускает вычисление (щелчком по соответствующей кнопке);
5) после завершения вычислений пользователю отображается результат;
6) пользователь сохраняет полученные результаты в виде файла (щелчком по соответствующей кнопке).

Django web-application

1) рекомендуется использовать встроенную систему авторизации и аутентификации;
2) необходимо продемонстрировать работу с DTL;
3) можно выбрать любую БД, которую поддерживает Django ORM.

Flask web-application

1) рекомендуется использовать Flask-Login;
2) взаимодействовать с базой данных можно либо с помощью Flask-SQLAlchemy, либо используя другие ORM (например peewee).

PyQT desktop application

1) взаимодействовать с базой данных можно с помощью ORM (например peewee).