Обработка и анализ медицинских изображений

Состав команды

Веселовский Леонид

@lvveselovskiy

lvveselovskiy@edu.hse.ru

EDA
Linear models
TG-Bot

Воробьёв Андрей

@Saprentum

amvorobev@edu.hse.ru

EDA
CNN model
Pipeline

Родионов Никита

@white_shpengler

narodionov_3@edu.hse.ru

EDA
CNN model
TG-Bot

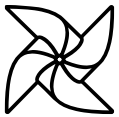
Куратор: Никифоров Михаил

Описание проекта

Основная идея проекта – создание сервиса, позволяющего в режиме онлайн по МРТ снимку определить вид опухоли головного мозга

Наша модель умеет различать 4 вида класса: 3 вида опухоли (глиома, менингиома, опухоль гипофиза), а также снимки здорового мозга

Для реализации этого проекта мы использовали пакетную модель tensorflow CNN EfficientNetB3. А так же используется контейнер для линейного стека слоёв sequential

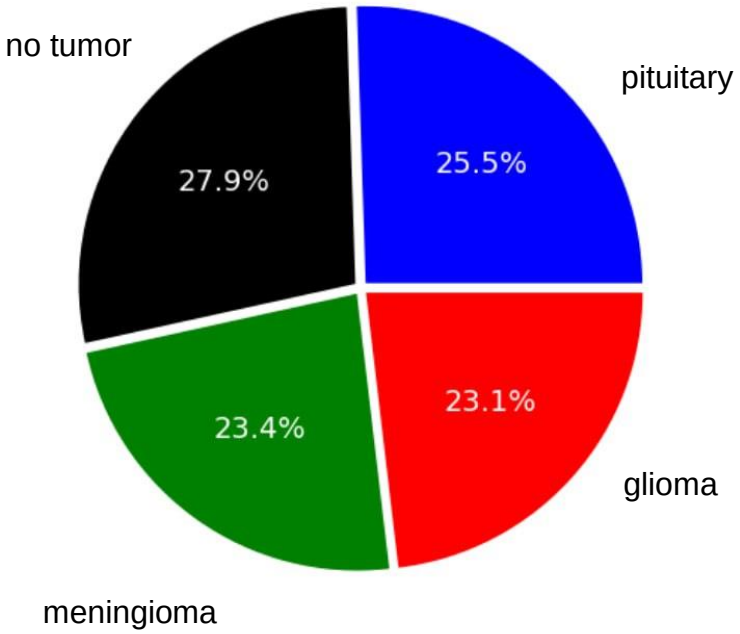


Эксперименты с параметрами были автоматизированы в Airflow

Полученная модель была обёрнута в Телеграм-бот, задеплоенный на виртуальный сервер рег.ру

Описание данных

k https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset



Class 1: glioma
Class 2: meningioma
Class 3: no tumor
Class 4: pituitary

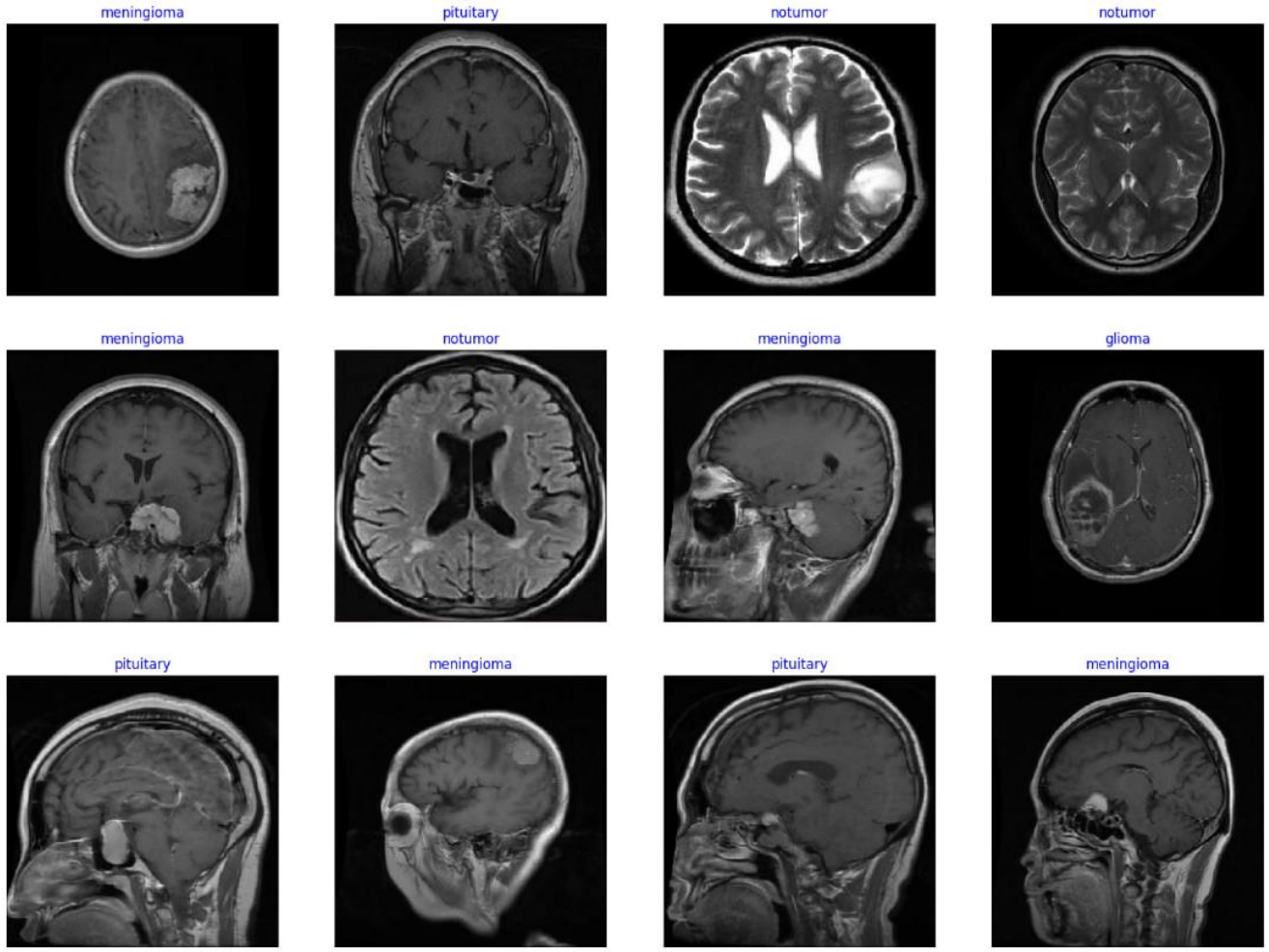
Overall number of Training subset files: 5712

Number of Class 1 files (Training set): 1321
Number of Class 2 files (Training set): 1339
Number of Class 3 files (Training set): 1595
Number of Class 4 files (Training set): 1457

Overall number of Testing subset files: 1311

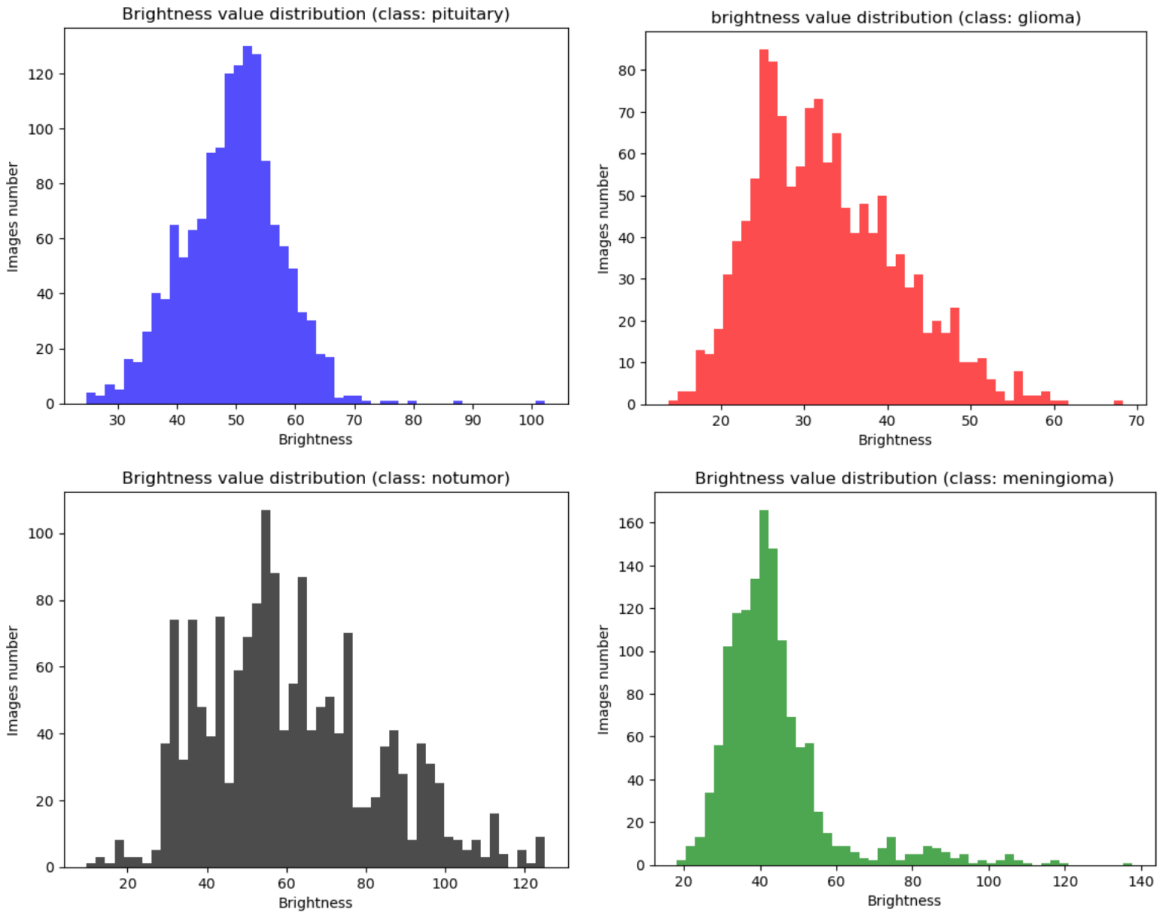
Number of Class 1 files (Testing set): 300
Number of Class 2 files (Testing set): 306
Number of Class 3 files (Testing set): 405
Number of Class 4 files (Testing set): 300

Разведочный анализ

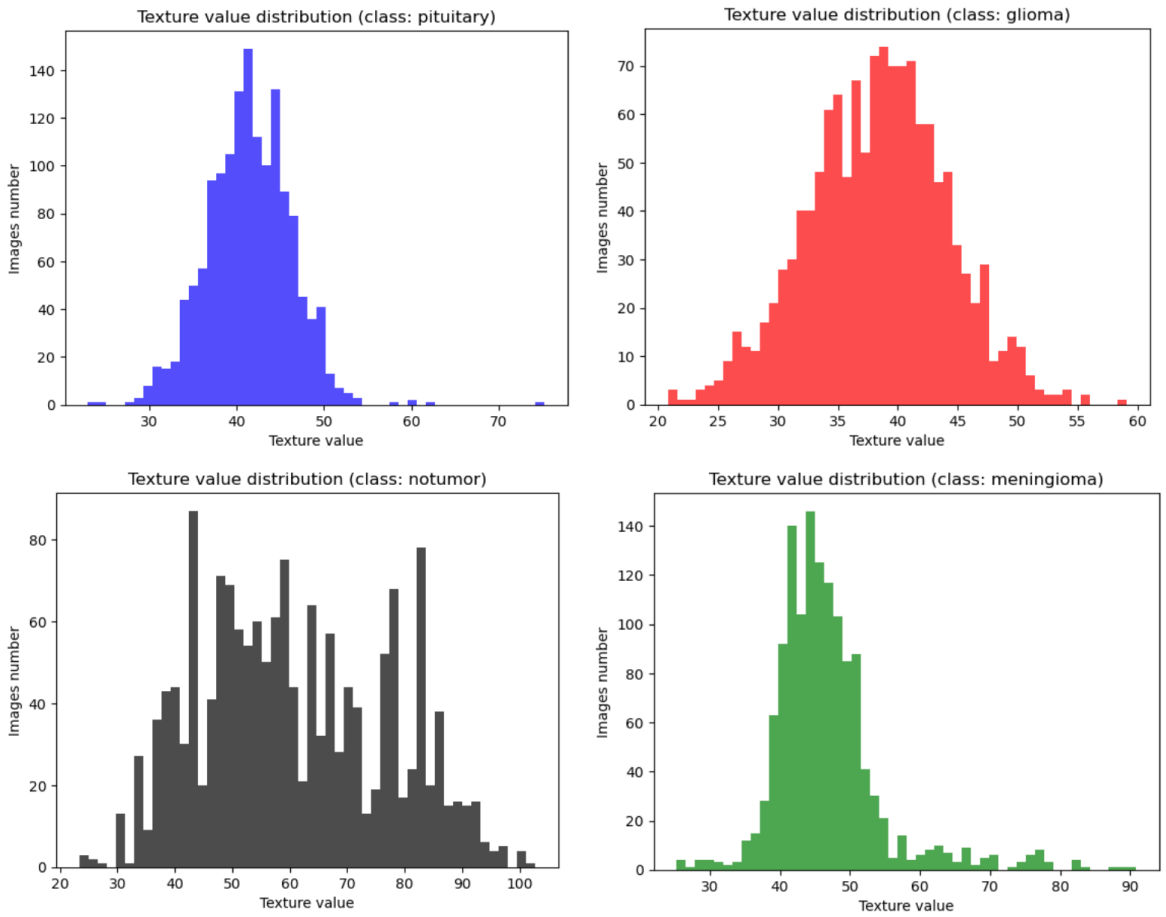


Разведочный анализ

Brightness



Texture



Линейные модели (SVM & OVR)

Первым этапом все изображения приводились к одному формату (**Image Size** = (244,244), **color_mode** = 'grayscale')

После чего делали Flatten Images, для того чтобы перевести изображения в представление 1D вектора. На выходе получили матрицу с **50176** признаками

Для уменьшения размерности провели PCA преобразования, отобрав 2250 признаков, после чего обучали линейные модели

LinearSVC

Test Accuracy: 0.8765
Classification Report:

	precision	recall	f1-score	support
0	0.80	0.78	0.79	151
1	0.80	0.79	0.79	164
2	0.96	0.96	0.96	199
3	0.92	0.96	0.94	142
accuracy			0.88	656
macro avg	0.87	0.87	0.87	656
weighted avg	0.88	0.88	0.88	656

LogisticRegression(multi_class='ovr')

Test Accuracy: 0.9085
Classification Report:

	precision	recall	f1-score	support
0	0.86	0.83	0.84	151
1	0.84	0.83	0.83	164
2	0.98	0.98	0.98	199
3	0.94	0.98	0.96	142
accuracy			0.91	656
macro avg	0.90	0.91	0.90	656
weighted avg	0.91	0.91	0.91	656

CNN EfficientNetB3

Было решено использовать CNN модель, ниже представлен код модели

```
base_model = tf.keras.applications.efficientnet.EfficientNetB3(include_top = False , weights = 'imagenet' ,
                                                               input_shape = img_shape, pooling= 'max')
model = Sequential([
    base_model,
    BatchNormalization(axis= -1 , momentum= 0.99 , epsilon= 0.001),
    Dense(256, kernel_regularizer = regularizers.l2(l= 0.01) , activity_regularizer = regularizers.l1(0.005),
         bias_regularizer= regularizers.l1(0.005) , activation = 'relu'),
    Dropout(rate= 0.4 , seed = 75),
    Dense(num_class , activation = 'softmax')
])

model.compile(Adamax(learning_rate = 0.001) , loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

CNN EfficientNetB3

После обучения мы уже имеем хорошее accuracy, превосходящее простые линейные модели.

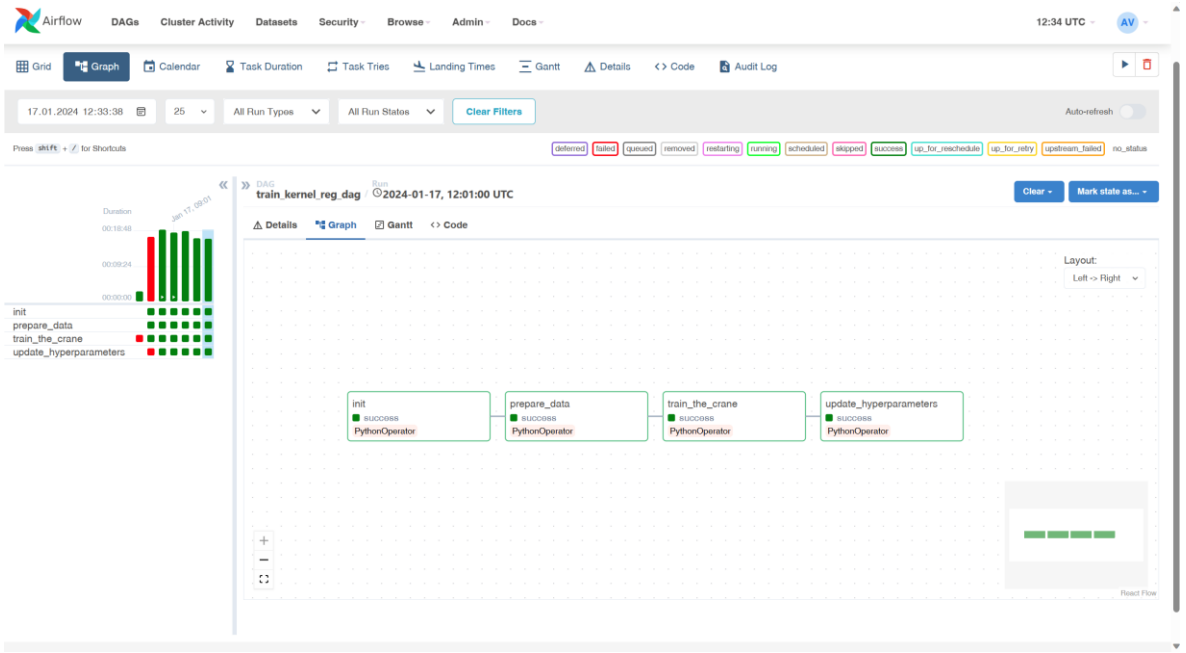
Проводился ряд экспериментов с перебором гиперпараметров для подбора наиболее точных значений:

Bias & activity regularizer value = 0.05; dropout = 0.4		
Kernel regularizer value	Test accuracy	Test Loss
0.014	0.9375	3.719017744064331
0.015	0.9375	3.950064182281494
0.016	0.96875	3.703428030014038
0.017	0.953125	3.8614089488983154
0.018	0.953125	3.69682240486145
0.019	0.859375	4.495482444763184

Bias & activity regularizer value = 0.007 (modified in accordance with the best test results); kernel regularizer value = 0.017		
Dropout rate	Test accuracy	Test Loss
0.25	0.953125	3.2740769386291504
0.30	0.9375	3.5424821376800537
0.35	0.953125	3.719808578491211
0.40	0.96875	3.842622756958008
0.45	0.984375	4.059911251068115
0.50	0.921875	4.190309524536133

Kernel regularizer value = 0.017; dropout =0.4		
Bias & activity regularizer values	Test accuracy	Test Loss
0.003	0.953125	3.5733211040496826
0.004	0.984375	3.984596014022827
0.005	0.9375	3.8716304302215576
0.006	0.921875	3.865628957748413
0.007	0.984375	3.7739477157592773
0.008	0.921875	3.8522238731384277

Pipeline & Airflow



На этапе раннего прототипа локально реализована часть пайплайна на базе **Airflow**.

```
{Kernel_reg_train_dag.py:135} INFO - Training process is performed with the following kernel regularizer hyperparameter:0.025
```

Главный (на данный момент) **DAG** осуществляет переборку заданных гиперпараметров модели в автоматическом режиме. Значение гиперпараметра, используемого в каждом запуске, фиксируется логе.

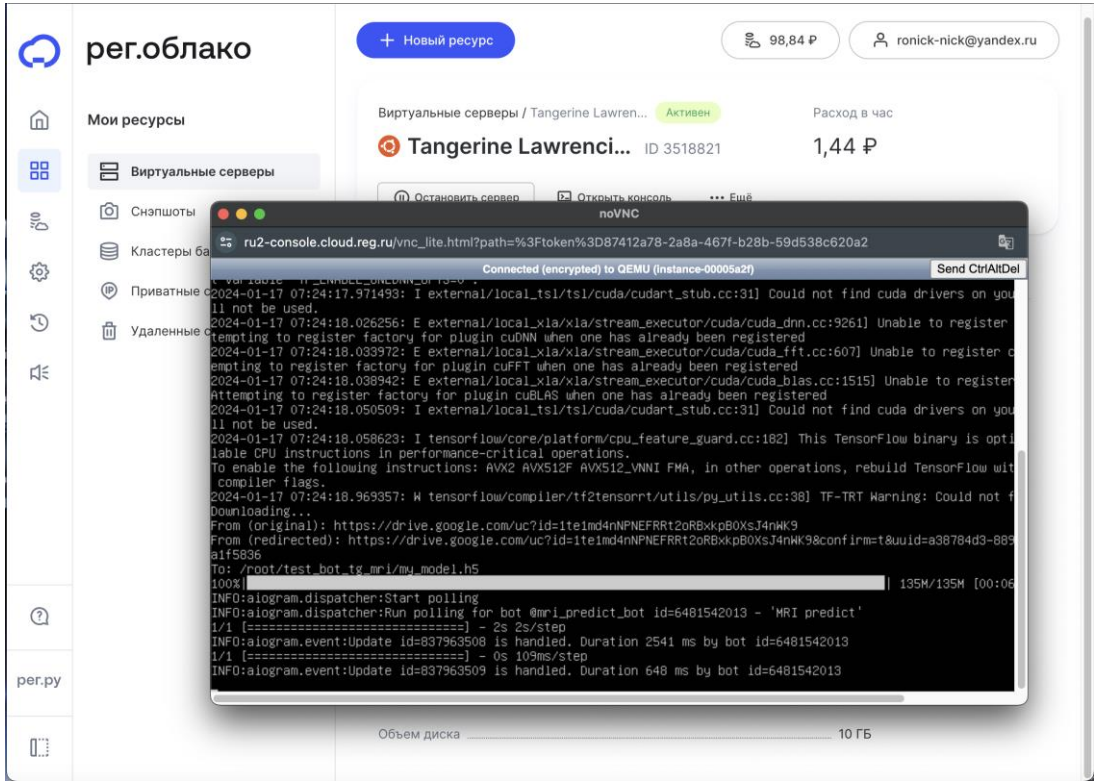
```
{Kernel_reg_train_dag.py:169} INFO - Training process executed successfully with following results
{Kernel_reg_train_dag.py:170} INFO - Train Loss: , 2.6332006454467773
{Kernel_reg_train_dag.py:171} INFO - Train Accuracy: , 1.0
{Kernel_reg_train_dag.py:172} INFO - Validation Loss: , 2.5505058765411377
{Kernel_reg_train_dag.py:173} INFO - Validation Accuracy: , 1.0
{Kernel_reg_train_dag.py:174} INFO - Test Loss: , 2.6259500980377197
{Kernel_reg_train_dag.py:175} INFO - Test Accuracy: , 0.9375
```

Логгируются также и метрики качества. На основании данных из логов удается отслеживать влияние изменения определенного гиперпараметра модели на точность предсказаний, **без необходимости вносить изменения каждый раз вручную**.

Telegram Bot

Был реализован телеграмм-бот, который принимает на вход снимок МРТ и возвращает предсказанный класс.

Бот развёрнут на сервисе рег.ру



Ссылка на Телеграм:
https://t.me/mri_predict_bot

Ссылка на гифку с работой бота:
https://github.com/Nicky-Georgia/mri_tumor_classification/blob/main/gif_bot.gif

Дальнейшее развитие проекта

Развёртывание полноценного пайплайна с использованием сервиса s3

Контейнеризация приложения

Развитие модели, уточнение гипперпарметров с целью повышения её качества

Дальнейшая разработка сервиса и добавление новых фич для него