Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г. В. Плеханова»

Высшая школа кибертехнологий, математики и статистики

Кафедра математических методов в экономике

Выпускная квалификационная работа

по программе профессиональной переподготовки

«Основы Data Science на языке Python»

на тему «Создание MVP по оценке сходства с лицами знаменитостей»

Выполнил:

Коротков Фёдор Олегович

Преподаватель:

профессор кафедры математических методов в экономике,

д.э.н. Моисеев Никита Александрович

Москва

2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время одним из наиболее востребованных направлений работы с ИИ являются нейронные сети и в частности задачи практического применения CV (computer vision).

В качестве основной задачи данной выпускной квалификационной работы определена задача определения процента схожести черт тестируемого лица с лицами знаменитостей мирового уровня с использованием методов глубокого машинного обучения — библиотек CV.

Основная цель работы – показать полный процесс создания и деплоя в публичный доступ в сети интернет MVP (minimum viable product) по определению процента схожести тестируемого лица с лицами из базы данных, на которой обучена наша модель.

В качестве дополнительных целей, хотелось бы выделить постановку приоритета в реализации MVP на поддерживаемость и масштабируемость проекта, а так же достаточно легкую возможность перепрофилировать его в более узкоспециализированные решения, например создание базы данных лиц сотрудников компании и использовании в системе фиксации рабочего времени сотрудника или организации разных уровней допуска в соответсвующие помещения организации.

Исходный код программы приведён в приложении А.

1 Анализ предметной области и постановка задачи

1.1 Постановка задачи

К задачам создания MVP относятся:

- создание общей структуры проекта;

- организация ETL-процесса (extract, transform, load), то есть процесса извлечения (сбора) данных их преобразования и загрузки в базу данных проекта;

- построение модели по определению процента схожести тестируемого лица;

- оценка метрик полученной модели, локальное тестирование модели;

- деплой модели в публичный доступ в сеть интернет;

- тестирование и интерпретация полученной MVP;

- выгрузка как open-source project в публичный репозиторий GitHub.

1.2 Доступные ресурсы

Для выполнения задачи доступны следующие ресурсы:

- язык программирования Python3.8 со следующими библиотеками (так же указаны в файле requirements.txt для более быстрой и удобной установки): pandas==2.0.0, numpy, matplotlib, Google-Images-Search, bing-image-downloader, Pillow, tensorflow, face_recognition, scikit-learn, Flask;

- парсинг фотографий лиц знаменитостей из интернета;

- знания, полученные в ходе обучения, а также статьи и публикации, youtube.com уроки относящиеся к теме CV в части распознаванию лиц;

- среда разработки PyCharm;

- условно бесплатный хостинг ресурс для пайтон скриптов от проекта Anaconda – pythonanywhere.com

- облачный репозиторий для разработчиков Github.

1.3 Риски

В ходе выполнения данной работы существуют определенные риски, которые могут повлиять на точность результатов. В основном, большая часть рисков так или иначе обусловлена применением скриптов для автоматизации поиска фотографий для наполнения базы данных модели. Выделим основные «узкие места» применения скриптов и возможные последствия.

Один из главных рисков — это риск моделирования на некорректном наборе данных, то есть модель будет обучаться на фотографиях, которые принадлежат не тем лицам, которые подразумевались или на фотографиях отсутствуют лица вообще.

Существует риск снижения качества классификации конкретных знаменитостей из-за малого количества собранных изображений в виду технических причин: блокировка сайтом русского айпи, с которого ведется парсинг фото, ошибка загрузки фотографий из-за технических сбоев в соединении, антипарсинг системы типа cloudflare и др.

Также хотелось бы выделить в качестве «бутылочного горлышка» проблему квоты дискового пространства на условно-бесплатном хостинге pythonanywhere.com, которое наложило существенные ограничения на логику работы приложения — пришлось заменить часть кода, который изначально предполагал выбор случайной картинки из имеющейся базы данных знаменитостей и показа в конечном итоге для визуального сравнения результатов работы модели, на поисковый запрос в интернет и выдачу случайной фотографии лица знаменитости непосредственно в ходе рабочего процесса, что сказалось на скорости работы сервиса.

1.4 Ограничения

В ходе выполнения работы существует ряд ограничений, который может повлиять на выполнение работы:

- используются сторонние нейронные сети для решения задачи распознавания лиц на фотографиях и получения их эмбеддингов, что непосредственно влияет на качество конечной модели;

- ограниченность в вычислительных, временных и hardware ресурсах, что оказывается влияние на время осуществления ETL-процессов, обучение модели и, как следствие, точности прогнозов.

1.5 Критерии успешности

Для оценки качества модели, необходимо сравнить реальные и предсказанные значения и рассчитать следующие метрики:

- F1 score — f_1 мера

- MSE (Mean Squared Error) – среднеквадратическая ошибка;

- MAE (Mean Absolute Error) – средняя абсолютная ошибка;

- MAPE (Mean Absolute Percentage Error) — средний абсолютный процент ошибки.

Для оценки работоспособности MVP — тест модели развернутой в публичном доступе в сети интернет.

2 Создание структуры MVP, ETL-процесса

2.1 Создание структуры MVP

Общая структура проекта представлена на рисунке 1.

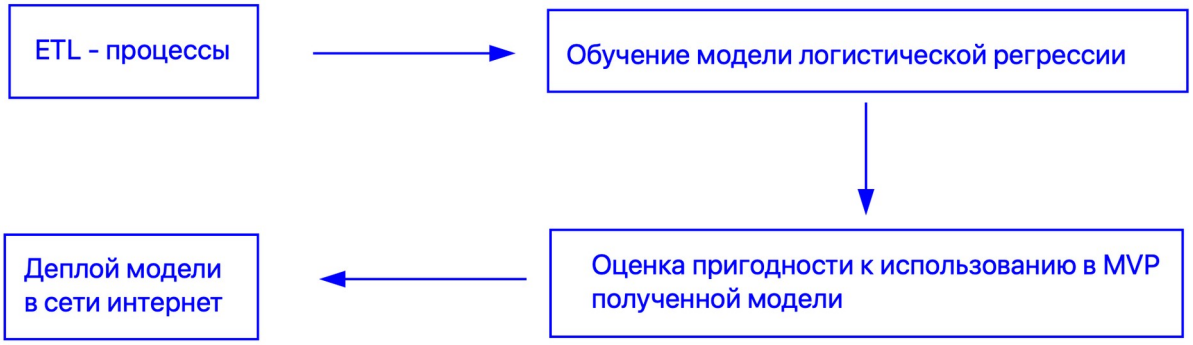

Рис. 1 – Общая структура проекта

Ключевыми блоками являются блоки ETL-процесса формирования базы данных изображений для последующего обучения модели, а так же блок создания модели и оценки ее качества. Остальные блоки не столь важны и могут легко варьироваться в зависимости от конкретных конечных целей профилирования MVP.

2.2 Организация ETL-процессов

Принципиальная схема сбора, обработки и выгрузки в локальную базу данных представлена на рисунке 2.

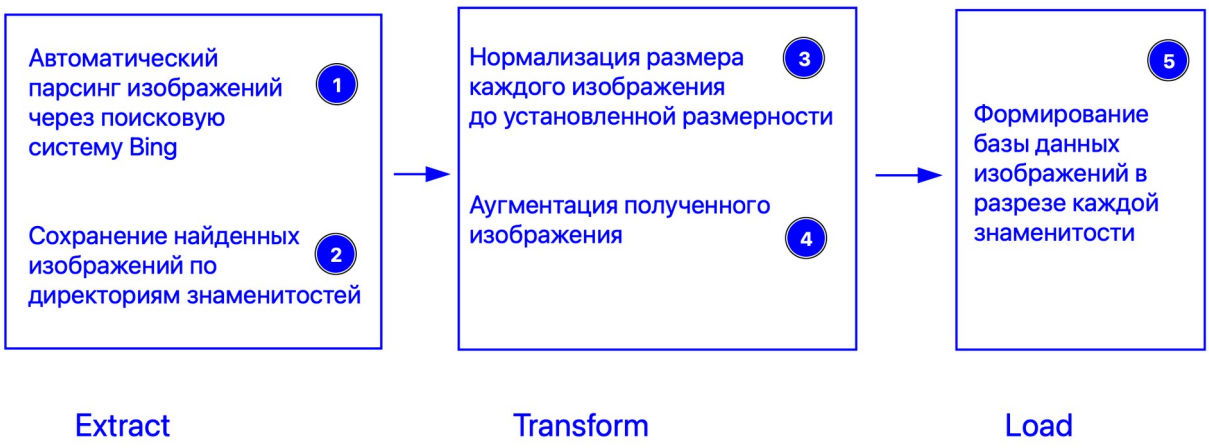

Рис. 2 – Схема организации ETL-процесса

Этап парсинга изображений был автоматизирован с помощью созданного скрипта и библиотеки bing_image_downloader.

На этапе преобразования данных использовались библиотеки:

- PIL — преимущественно для открытия спарсенных изображений и перевода в RGB — цветовой канал.

- tensorflow — для нормализации размера изображений (по наибольшей стороне к заданному размеру. В нашем случае использовался размер в 256 пикселей). Так же данная библиотека использовалась для проведения процесса аугментации нашего набора данных — в нашем проекте использовался метод вертикального зеркалирования нормализованного изображения.

Благодаря использованию аугментации мы увеличили размер нашего набора данных вдвое — до 2770 изображений.

Для создания конечного набора данных (pandas data set), на котором предполагается последующее обучение модели был написан отдельный скрипт. Данное решение обусловлено желанием добиться независимости и гибкости в процессах редактирования базы данных изображений и будущими дообучениями моделей.

Для создания финального дата сета была использована библиотека для распознавания лиц — face_recognition.

При использовании данной библиотеки с помощью ее нейронных сетей на каждом изображении отыскивалось лицо или лица и получались координаты рамок вокруг ключевых фич, характеризующие найденные лица. В случае, если находилось более одного лица (истинно или ложно) такое изображение удалялось из последующей обработки и включения в финальный дата сет.

На рисунке 3 представлен пример работы данной библиотеки и локализации найденного лица.

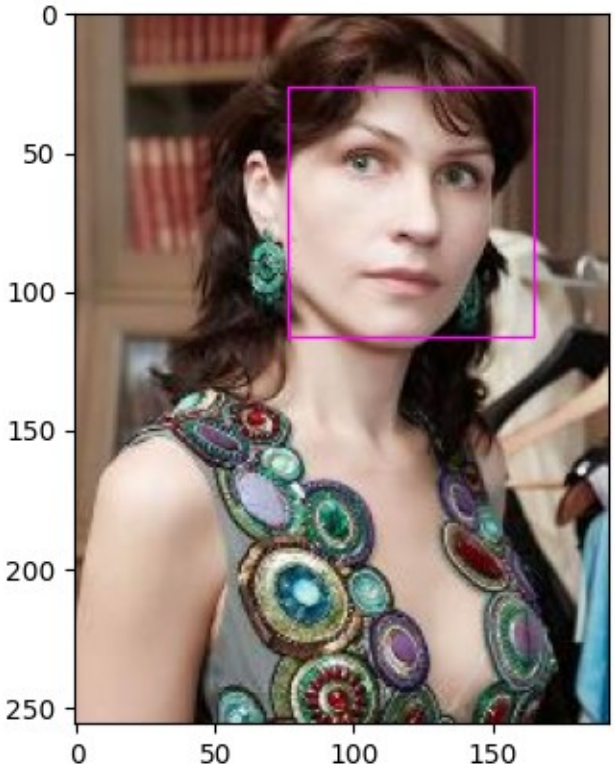

Рис. 3 – Пример локализации лица с помощью face_recognition

Далее, необходимо произвести кодирование полученной области в многомерный вектор, который будет описывать фичи полученной области. Это так же было сделано с помощью face_recognition и встроенного метода encoding. Полученный вектор называется эмбеддингом (embedding) данной локализации изображения и имеет 128-мерный размер. Таким образом, наш

дата сет будет содержать 128 фич для описания каждой целевой переменной — person_id. Кроме того, имеет смысл добавить в финальный дата сет категориальную переменную — имя и фамилию знаменитости.

Таким образом, наш финальный дата сет имеет 130 столбцов: имя и фамилию знаменитости, уникальный идентификатор знаменитости и 128 фичей, взятых из разложенного по элементам эмбеддинга. Так же в процессе создания финального дата сета был создан матчинговый дата сет, в котором содержатся только имя и фамилия знаменитости и его уникальный номер. Это нужно для того, чтобы более легко выводить во фронтенд результат матчинга. Примеры дата сетов представлены на рисунке 4.

```
           person name  target       x_0  ...     x_125     x_126     x_127
0    Svetlana Kamynina       0 -0.138202  ... -0.062513  0.061855  0.137531
1    Svetlana Kamynina       0 -0.212163  ... -0.029929  0.066204  0.132119
2    Svetlana Kamynina       0 -0.182329  ... -0.031636  0.061558  0.105531
3    Svetlana Kamynina       0 -0.149917  ... -0.091569  0.089505  0.095115
4    Svetlana Kamynina       0 -0.139294  ... -0.046278  0.036704  0.065077
..                 ...     ...       ...  ...       ...       ...       ...
113    Keira Knightley       2 -0.165561  ... -0.010705 -0.038994  0.052694
114    Keira Knightley       2 -0.135885  ... -0.031284  0.024060  0.050944
115    Keira Knightley       2 -0.151451  ... -0.005299  0.033146  0.100182
116    Keira Knightley       2 -0.067948  ... -0.050780 -0.005500  0.068263
117    Keira Knightley       2 -0.156079  ... -0.054903  0.053592  0.114619

[118 rows x 130 columns]
              person  person_id
0   Svetlana Kamynina          0
1  Michelle Rodriguez          1
2     Keira Knightley          2
```

Рис. 4 – Пример структуры финального дата сета (вверху) и матчингового (внизу)

Завершающим этапом было сохранение двух дата сетов с помощью встроенного python-модуля pickle.

3 Моделирование

В качестве основной модели была выбрана модель логистической регрессии, поскольку при таком количестве фич и размере финального дата сета ее функционала более чем достаточно. В дальнейшем эта модель может быть использована в качестве бейзлайна для сравнения с другими моделями либо тюнингом текущей.

Создание модели было вынесено в отдельный скрипт.

Этапы моделирования следующие (рисунок 5):

- загрузка с помощью pickle финального и матчингового дата сетов;

- формирование дата сета фичей из финального путем выброса двух столбцов — имя и фамилия знаменитости и его id;

- формирования дата сета с таргетами — это колонка с id;

- разбиение всех дата сетов на тренировочные и тестовые выборки в соотношении 80% на 20% и стратификации по таргету — чтобы мы были уверены, что выборки будут более-менее одинаково наполненные;

- обучение модели в многоклассовом режиме;

- расчет (с усреднением по методу micro) и оценка метрик получившейся модели.

```
df_name = 'data_set.pkl'
df = pd.read_pickle(df_name)
X = df.drop(['person name', 'target'], axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=0)
model = LogisticRegression(multi_class='multinomial')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
f_1 = f1_score(y_test, y_pred, average='micro')
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
print(f'f1 = {f_1}')
print(f'mse = {mse}')
print(f'mae = {mae}')
print(f'mape = {mape}')
```

Рис. 5 — Основные этапы построения модели логистической регрессии

Финальный дата сет и метрики модели представлены на рисунке 6.

```
            person name  target       x_0  ...     x_125     x_126     x_127
0     Svetlana Kamynina       0 -0.138202  ... -0.062513  0.061855  0.137531
1     Svetlana Kamynina       0 -0.212163  ... -0.029929  0.066204  0.132119
2     Svetlana Kamynina       0 -0.182329  ... -0.031636  0.061558  0.105531
3     Svetlana Kamynina       0 -0.149917  ... -0.091569  0.089505  0.095115
4     Svetlana Kamynina       0 -0.139294  ... -0.046278  0.036704  0.065077
...                 ...     ...       ...  ...       ...       ...       ...
2768        Zoe Saldana      73 -0.081204  ... -0.060320  0.151300 -0.012495
2769        Zoe Saldana      73 -0.148280  ... -0.094321  0.076726 -0.016689
2770        Zoe Saldana      73 -0.116344  ... -0.005060  0.065910  0.023910
2771        Zoe Saldana      73 -0.084897  ... -0.065017  0.121616  0.028736
2772        Zoe Saldana      73 -0.099750  ... -0.097373  0.082724 -0.005441

[2773 rows x 130 columns]
f1 = 0.97
mse = 20.1
mae = 0.66
mape = 0.04
```

Рис. 6 — Фрагмент финального дата сета и посчитанные метрики модели

F1 — метрика модели достаточно высокая, что обусловлено большим количеством фичей, при небольшом количестве таргетов (128 на 73 соответственно). Таким образом, модель достаточно хорошо оптимизирована.

Корреляционная матрица и укрупненный фрагмент представлены на рисунке 7 и 8. Видно, что иногда некоторые фичи имеют высокую мультиколлинеарность, что обусловлено спецификой фич — кодированием цвета.

Завершающим действием является сохранение обученной модели в файл с помощью pickle, для последующего деплоя проекта.

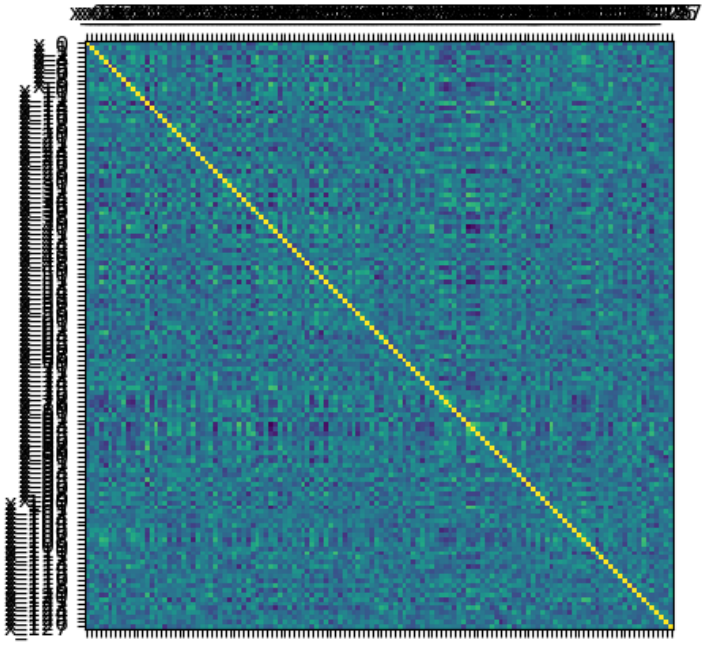

Рис. 7 — корреляционная матрица (полная)

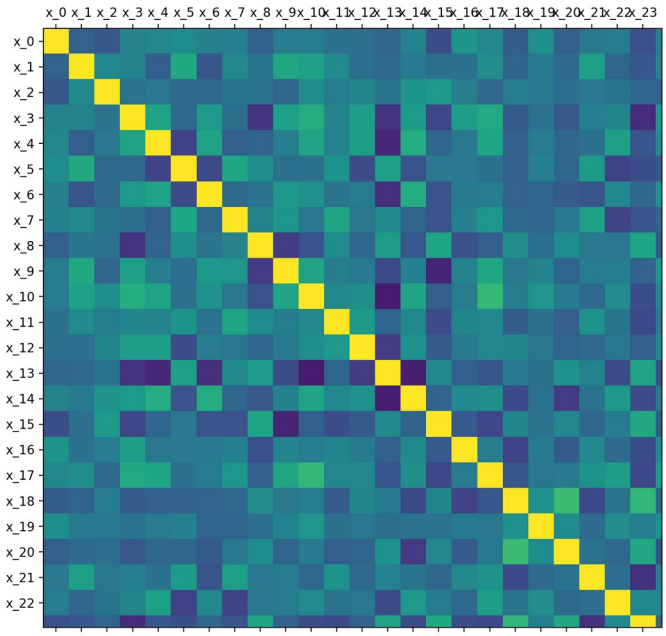

Рис. 8 - укрупненный фрагмент корреляционной матрицы

Таким образом, в качестве бейзлайн-модели мы можем использовать данную модель в целях реализации MVP.

4 Деплой проекта

Для хостинга проекта была выбрана условно-бесплатная платформа для размещения python-based скриптов — pythonanywhere.com

Web-адрес проекта: http://mysuperstarcity.pythonanywhere.com/

Для реализации фронтенда и серверной части был выбран фреймворк Flask, поскольку его функционал полностью покрывает потребности данного проекта и, кроме того, на данной платформе имеется возможность запускать flask-based приложения практически «из коробки».

Были созданы системные директории для flask:

- static (рисунок 9) — для хранения статичного наполнения сайта, а так же там создаются временные директории и файлы для тестируемых изображений, которые после окончания использования сервиса удаляются из соображения сохранения конфиденциальности личных данных, а так же с целью экономии ограниченного в 500 мб пространства платформы.

- templates (рисунок 10) — директория для хранения шаблонов страниц сайтов. В настоящее время реализовано 4 шаблон: доменный, страница с результатами анализа пользовательской фотографии и две страницы для обработки ошибок, возникающие в случае неуспешной попытки распознавания лица или наличия более, чем одного лица на фото.

- src (рисунок 11) — системная директория проекта, которая хранит предобученную локально модель, матчинговый дата сет и два списка, из которых flask выводит списки людей, на которых на текущий момент обучена модель, чтобы пользователь видел перечень знаменитостей, из которых будет произведен матчинг эмбеддингов.

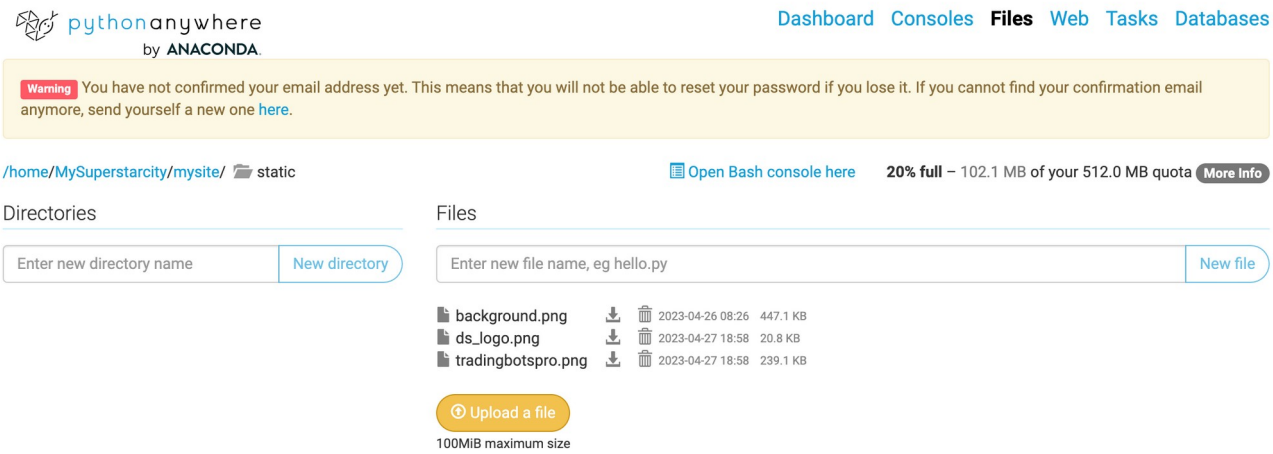

Рис. 9 — Содержание static директории MVP

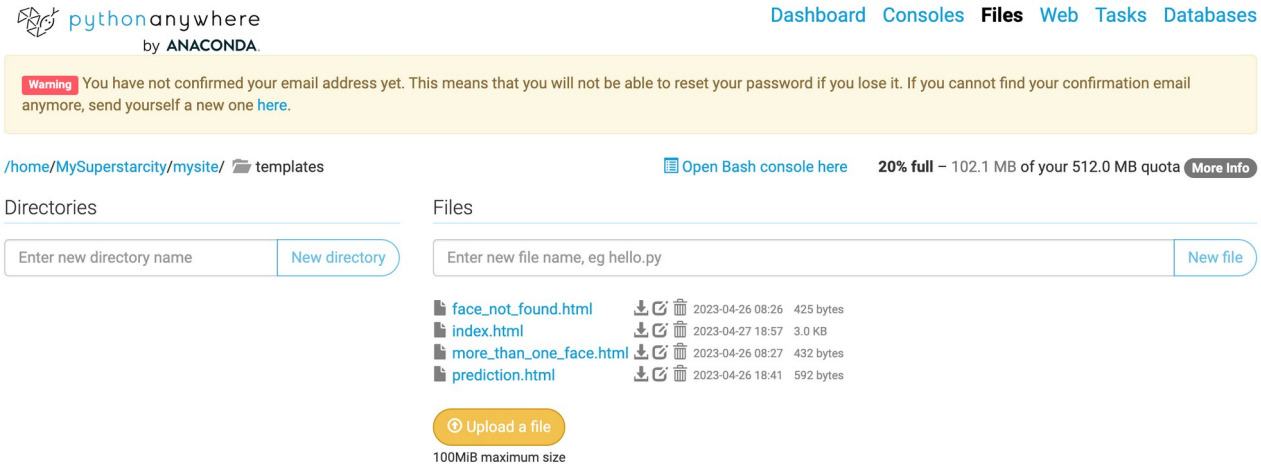

Рис. 10 — Содержание templates директории MVP

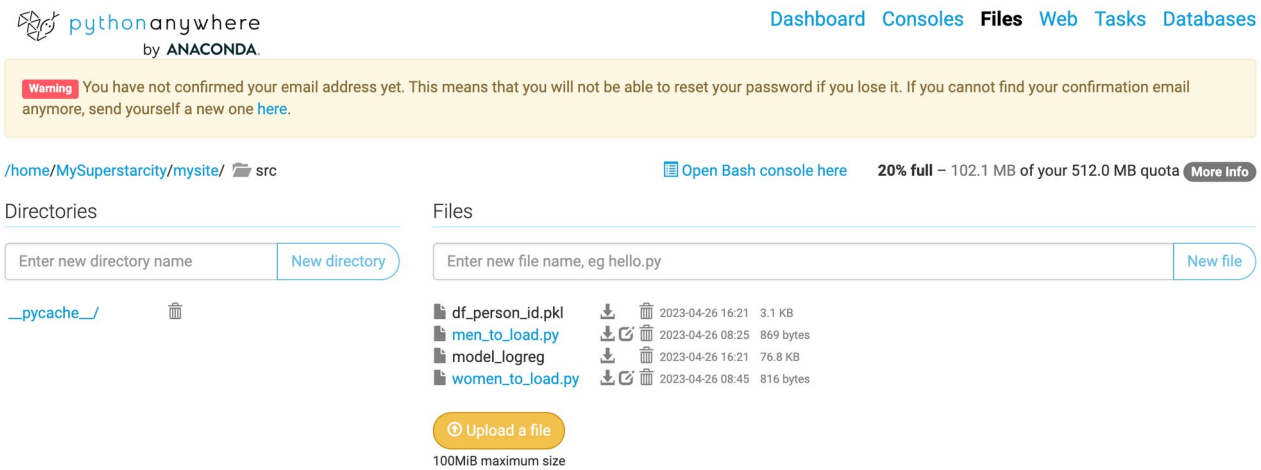

Рис. 11 — Содержание src директории MVP

Проведем несколько тестов нашего проекта. Результат матчинга фотографии без лица, с двумя лицами и с одним лицом представлен на рисунке 12, 13 и 14 соответственно.

Face was not found, please reload the photo with only the single face.

Try another photo!

Рис. 12 — Результат матчинга фотографии без лиц

Was found more than one face, please, reload the photo with only single face.

Try another photo!

Рис. 13 — Результат матчинга фотографии с двумя лицами

Yours highest SUPERSTAR matching is:

DANILA KOZLOVSKY with probability 7.39 %

Try another photo!

Рис. 14 — Результат матчинга фотографии с одним лицом

В завершении, проекта весь код выгружен в публичный доступ на мой github-профиль (рисунок 15): https://github.com/teslatrader/PlekhanovDiploma

Для удобства клонирования использованных в проекте библиотек и исключения конфликта pandas с pickle при операциях экспорта/импорта данных создан файл requirements.txt (рисунок 16), который позволит одной командой установить нужные библиотеки: pip3 install -r requirements.txt

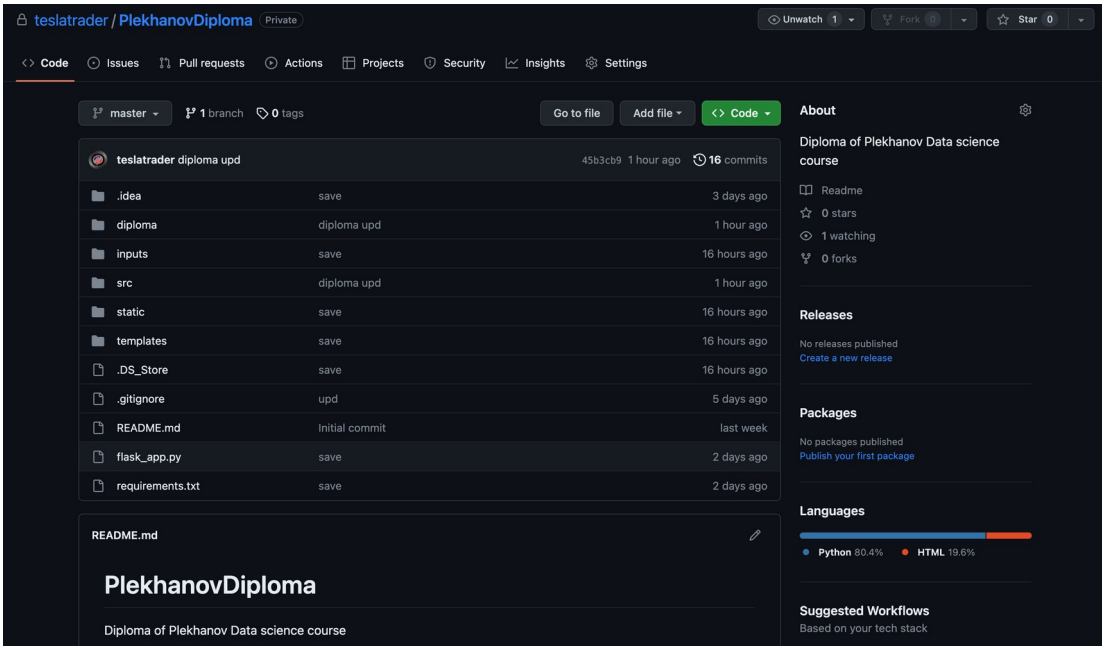

Рис. 15 — Репозиторий проекта на github.com

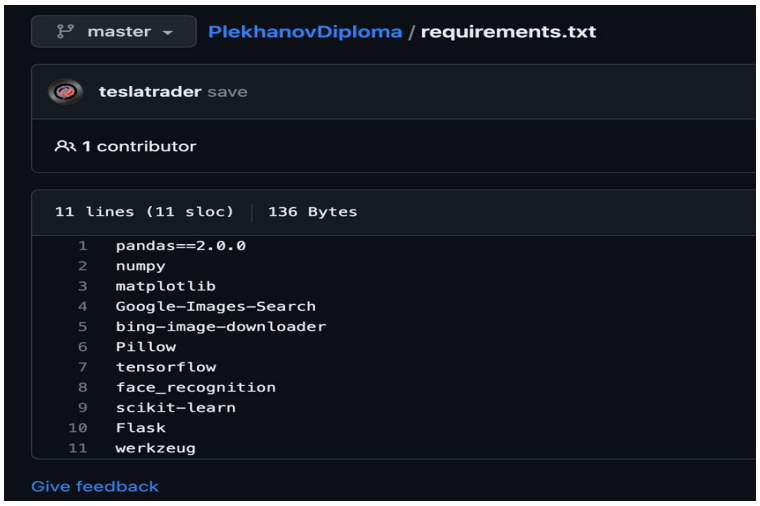

Рис. 16 — Перечень использованных в проекте библиотек

ЗАКЛЮЧЕНИЕ

В рамках данной работы был показан полный процесс создания и развертки модели машинного обучения в публичное пользование в сети интернет в качестве MVP.

В результате выполненной работы было показан процесс автоматизации сбора и обработки графических изображений, их нормализации и генерирования признаков (feature engineering) для последующего создания дата сета и применения методов машинного обучения. Было установлено, что полученная модель имеет достаточное качество прогнозирования и может быть использована в качестве бейзлайн-модели для реализации MVP.

В завершении показан процесс развертки модели в публичный доступ с имплементацией простейшего фронтенда и обработки ошибок распознавания тестируемых изображений.

В заключении необходимо отметить, что данный проект имеет множество точек для последующего роста, узкого профилирования и практического применения в реальной жизни.

Приложение А

(обязательное)

Исходный код программы

Структура кодовой базы проекта имеет следующий вид и является не целесообразным включать код в настоящее приложение.

Любой желающий может ознакомиться с кодом, так как он выложен в публичный доступ в мой репозиторий на github-профиль по адресу:

https://github.com/teslatrader/PlekhanovDiploma