|  |  |
| --- | --- |
| Федеральное агентство железнодорожного транспорта  Омский государственный университет путей сообщения  Кафедра «Автоматика и системы управления»  ЗНАКОМСТВО С PYTHON  Лабораторная работа № 8  по дисциплине «Информационные технологии» | |
|  | Студент гр. 23М                        Н.Л.Макаров                       Т.А.Абуталипов                        А.Е. Вижевитов  «  17  »        Мая         2024 г.  Руководитель –  старший преподаватель кафедры «АиСУ»                            Т.В. Васеева  «    »                  2024 г. |
| Омск 2024 | |

# Цель работы

Целью данной лабораторной работы является знакомство языком программирования Python, а также с сортировкой фото с помощью нейронной сети.

# Выполнение работы

## Для начала установим программу PyCharm и среду разработки Anaconda

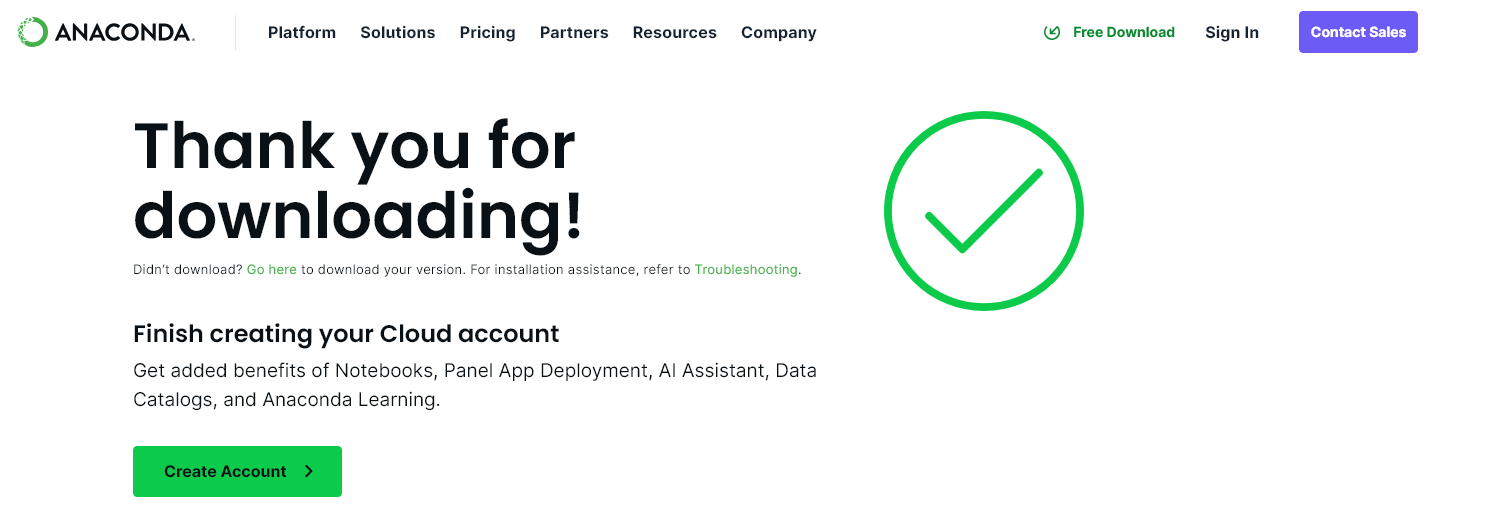


Рисунок 1 – Установка Anaconda

## Подгружаем нужные нам библиотеки для работы нейронной сети (рисунок 2)

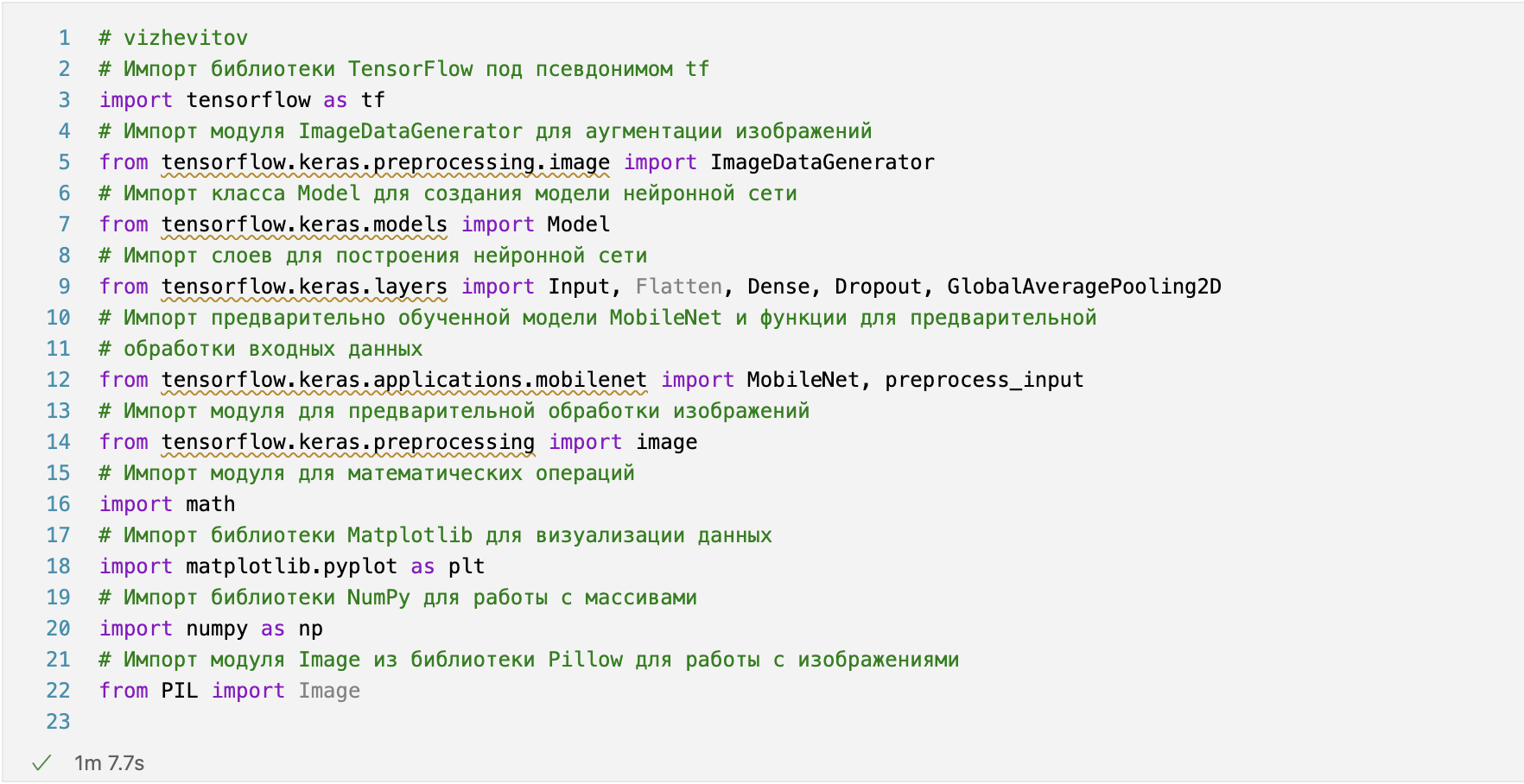


Рисунок 2 – Библиотеки

## Скачаем файлы для обучения нашей нейросети (рисунок 3)

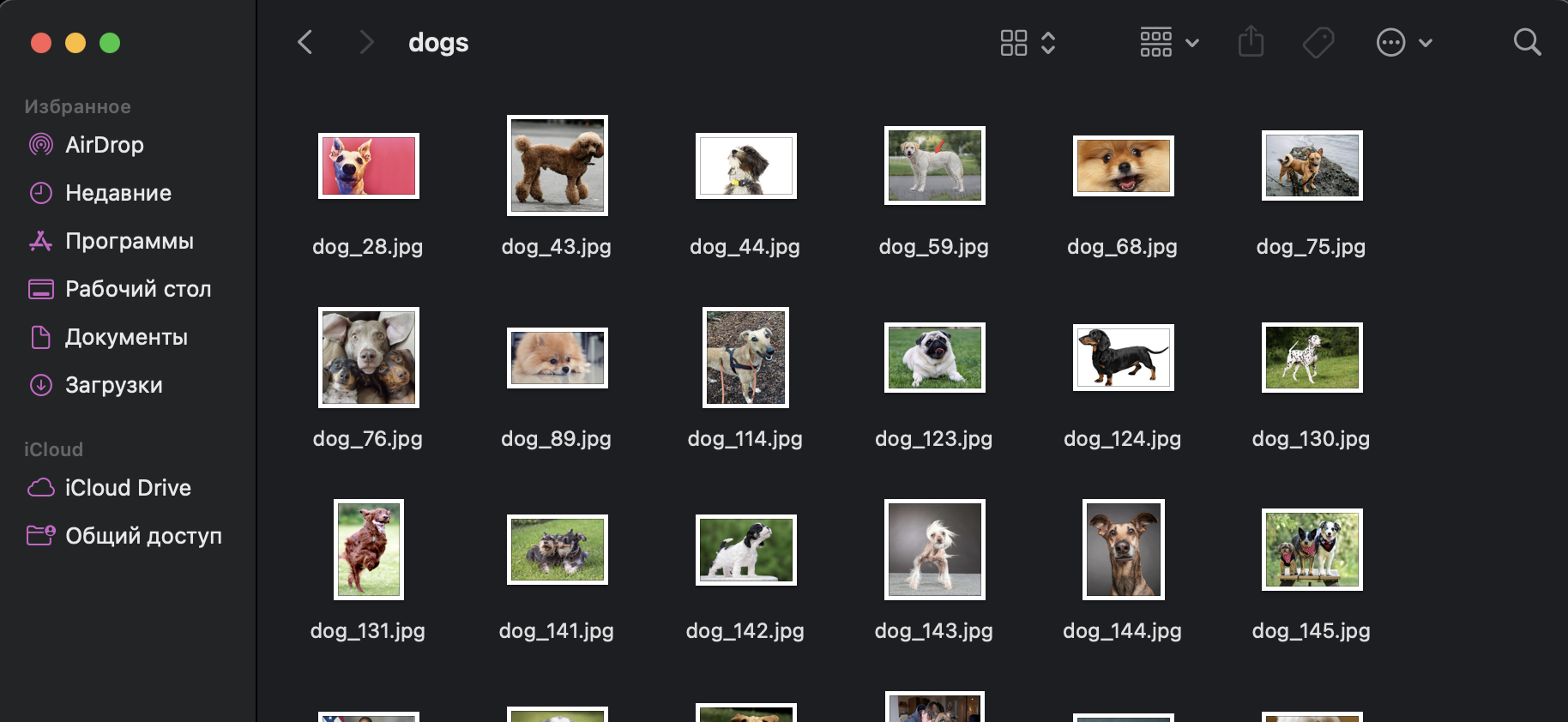


Рисунок 3 – Файлы для обучения

## Подключаемся к серверу Yupiter и подключаем python3

В приложении Anaconda-Navigator нажимаем Jupiter notebook после чего появляется терминал, из которого мы копируем адрес сервера. В VSCode нажимаем выбрать ядро и выбираем python3 и вставляем скопированный адрес сервера (рисунки 4-5).

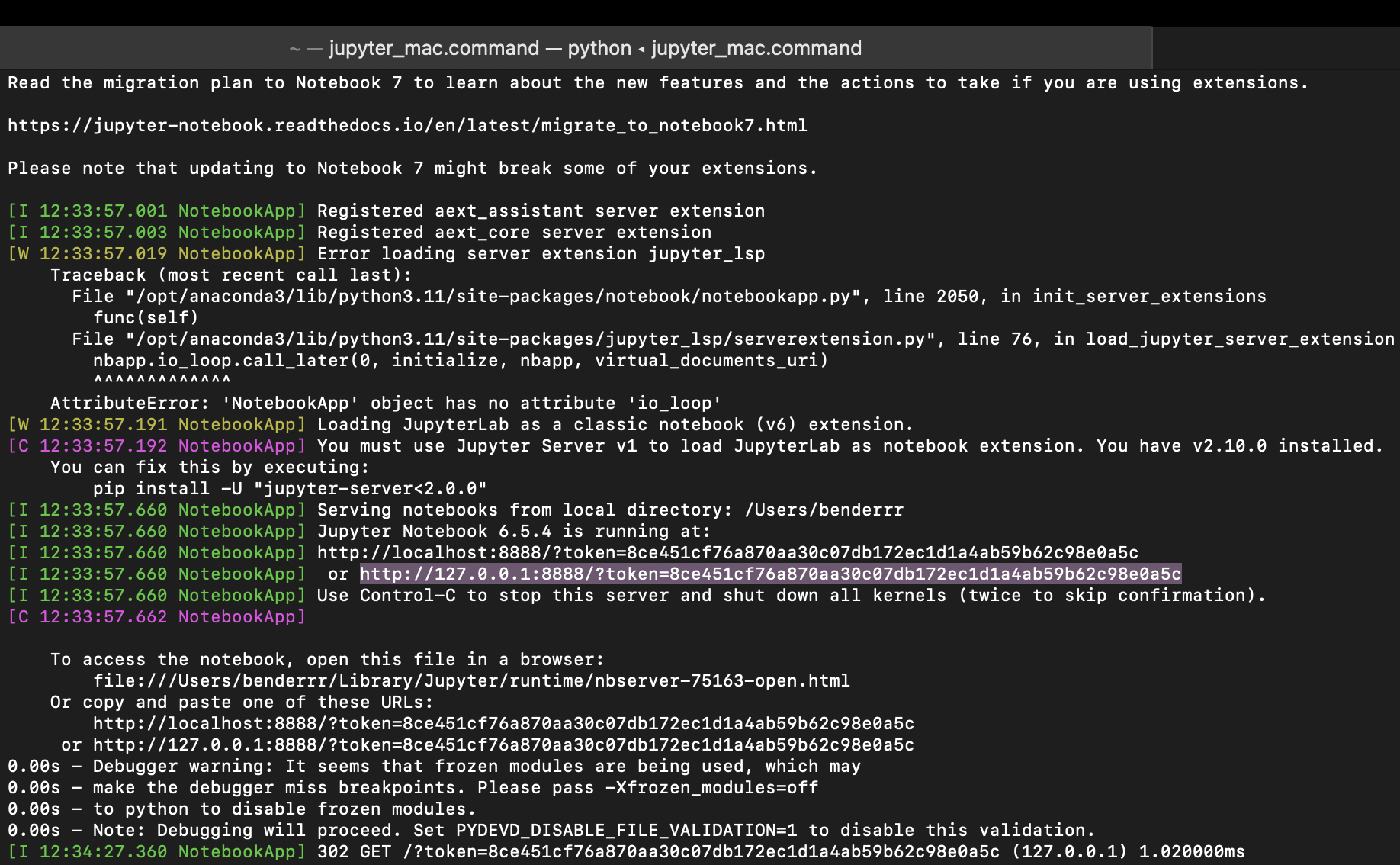


Рисунок 4 – Адрес сервера

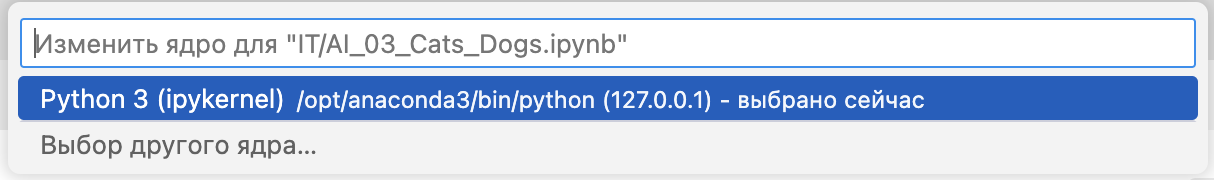


Рисунок 5 – Pychon3

## Данные константы служат для установления параметров и структуры обучения нейронной сети в процессе машинного обучения, особенно при работе с изображениями (рисунок 6)

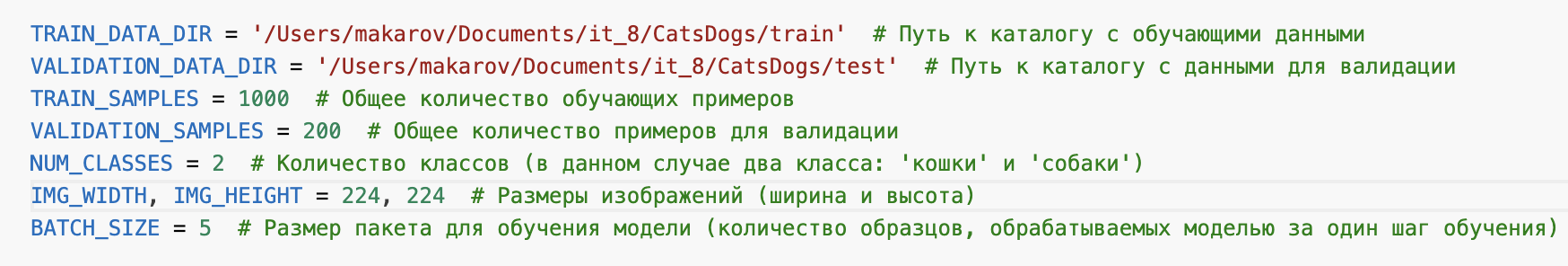


Рисунок 6 – Константы

## Создаются два генератора данных. Для обучения нейросети недостаточно пары сотен фотографий.

Поэтому будет использоваться такой прием как аргументация – это техника, используемая в машинном обучении, особенно при работе с изображениями, для искусственного увеличения объема обучающих данных путем создания модифицированных версий уже существующих данных. Это особенно важно в ситуациях, когда количество доступных данных ограничено, чтобы обеспечить достаточное разнообразие и предотвратить переобучение модели (рисунок 7).

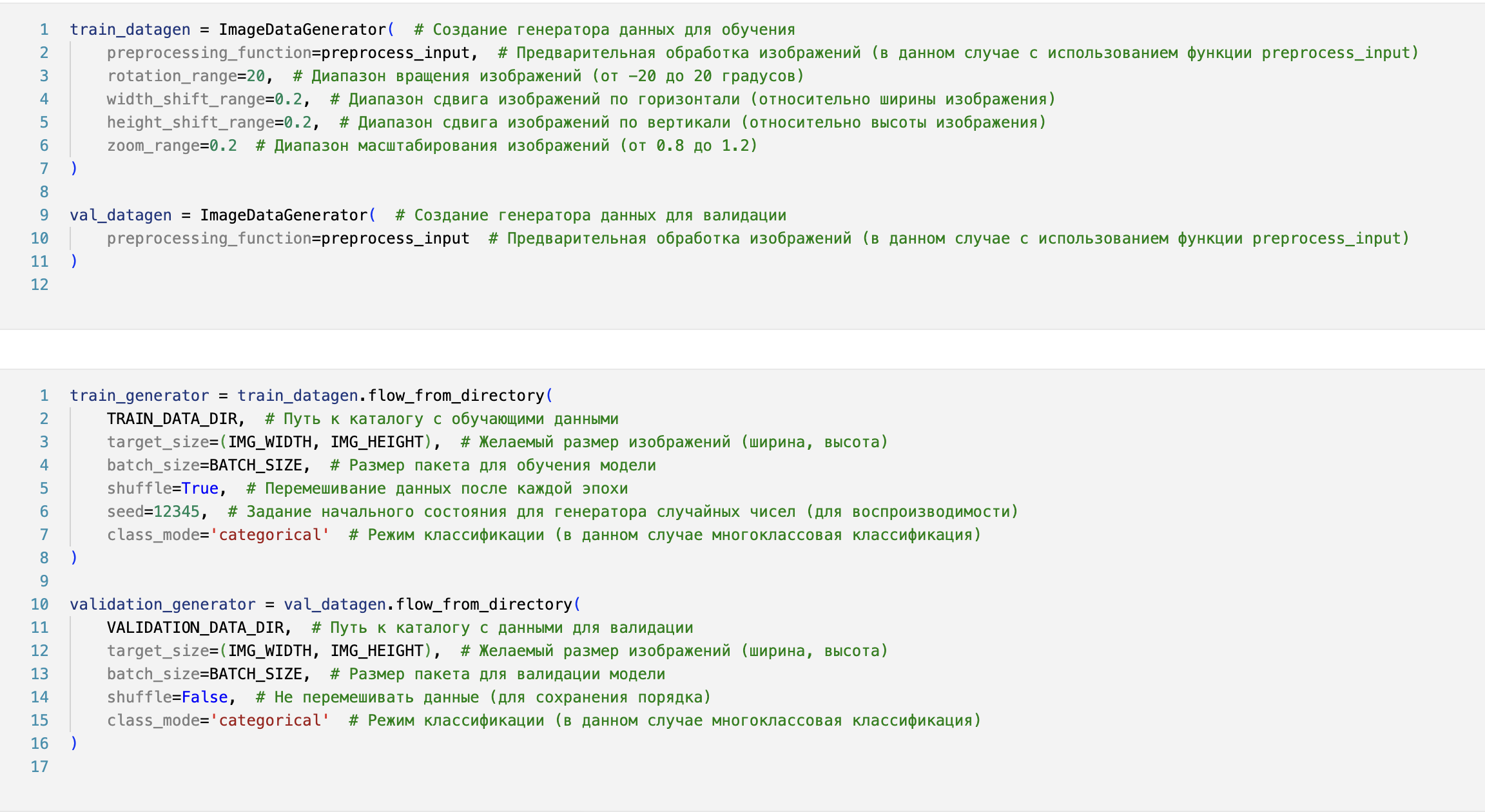


Рисунок 7 – Аргументация для нейронной сети

## Создание модели на основе нейронной сети

В базовой модели нужно отключить обучение (для всех слоев устанавливается trainable = False). Входное изображение поступает на вход базовой модели. После базовой модели добавляется несколько своих слоев:слой пулинга GlobalAveragePooling2D; полносвязный слой из 16 нейронов Dense; используется механизм Дропаут (случайное отключение нейронных связей во время обучения); последний слой должен выдавать два значения (вероятности того, что на фото кошка или собака – два класса). Таким образом мы решаем задачу двухклассовой классификации. (рисунок 7).



Рисунок 7 – Создание модели

## Компилируем и обучаем модель (рисунок 8)

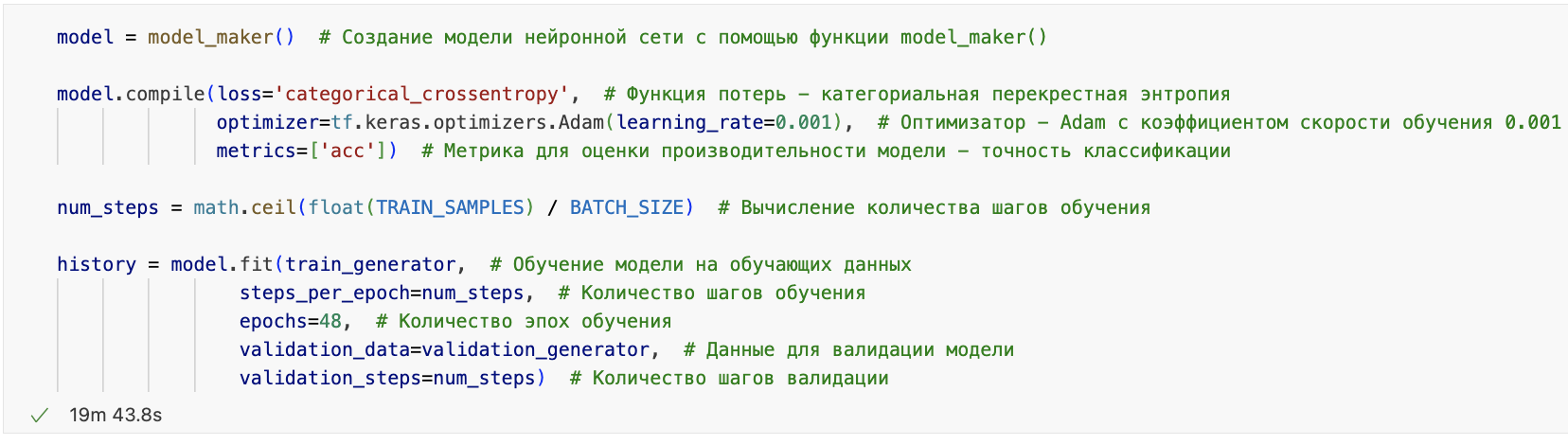
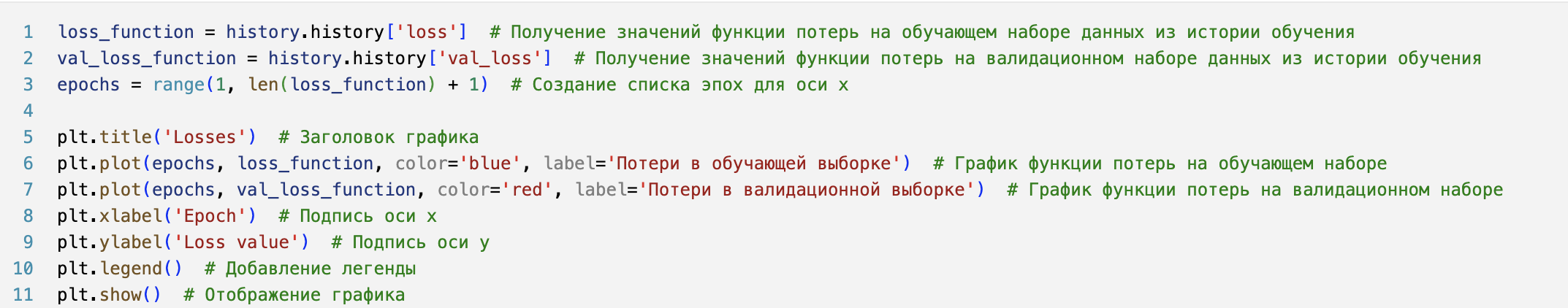


Рисунок 8 – Компиляция и обучение модели

## Построим график для визуализации обучения нейросети (рисунок 9)

  
Рисунок 9 – График

## Визуализированная проверка изображений (рисунок 10)

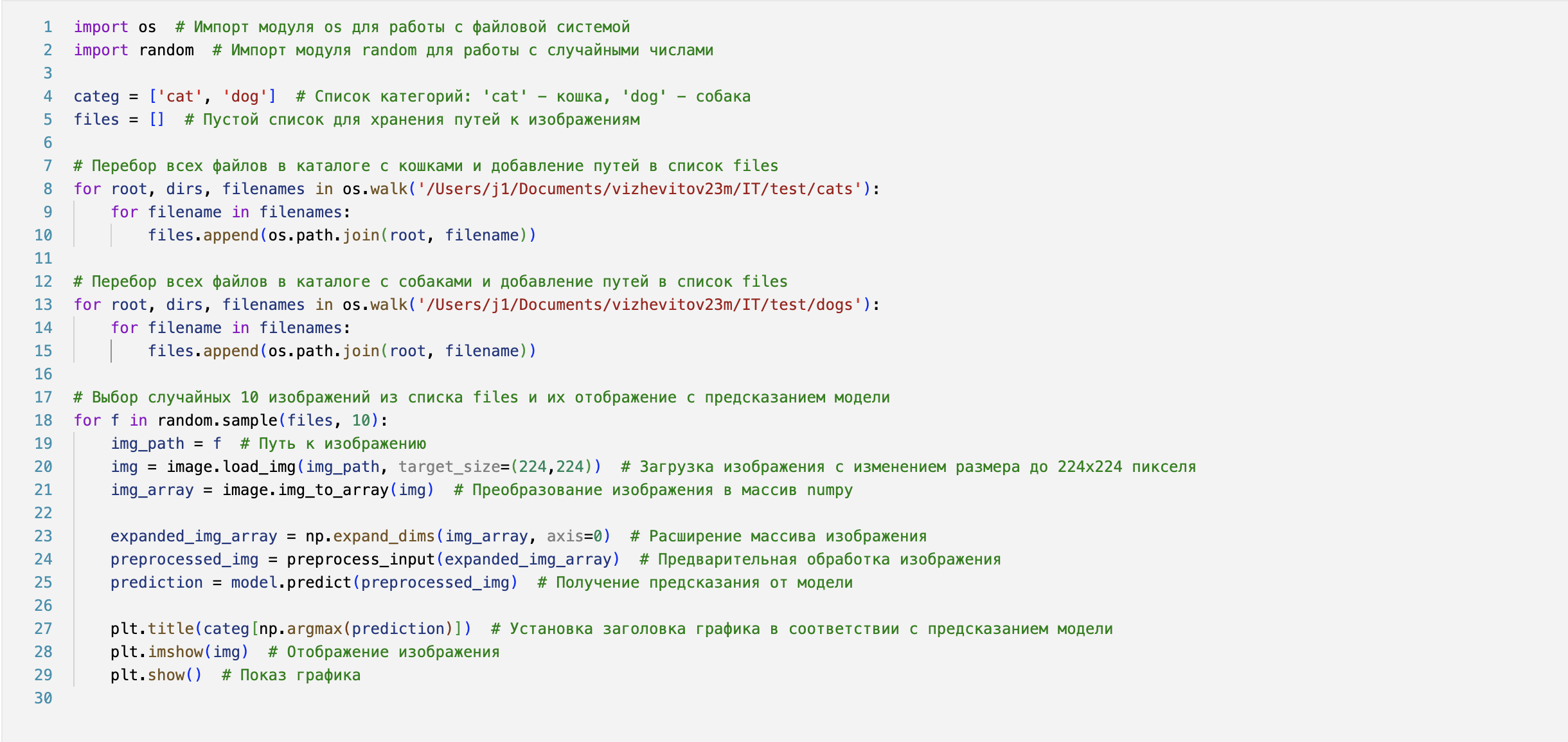


Рисунок 10 – Проверка изображений

## Теперь модель можно использовать и проверить, как она работает с помощью графика и анализа соответствия полученных изображений (рисунки 11-12)

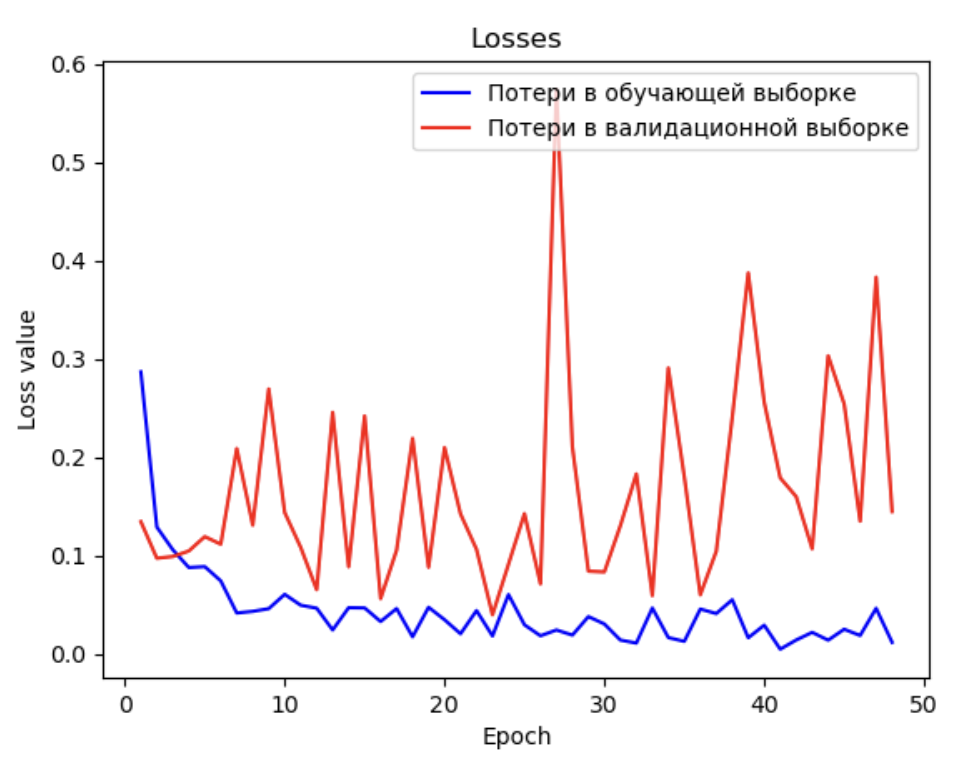


Рисунок 11 – График после проверки

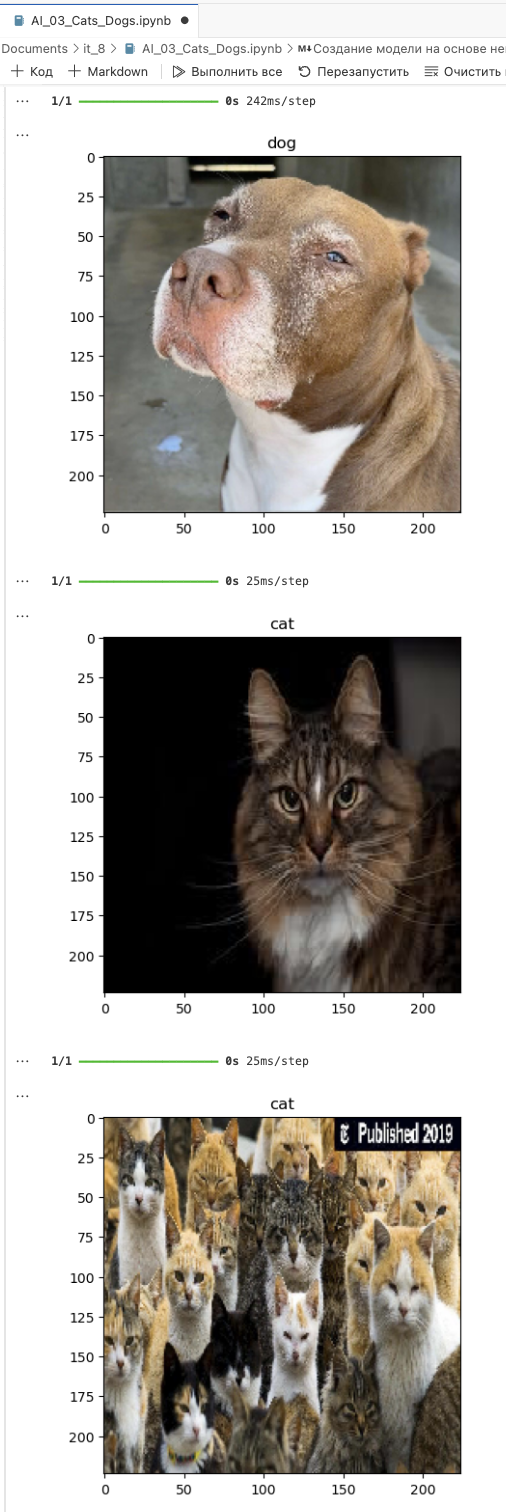


Рисунок 12 – Соответствие изображений

# Задания для защиты

## Другие готовые нейронные сети для извлечения информации:

- VGG (например, VGG16 или VGG19) - очень глубокие сети, которые показали отличные результаты в задачах распознавания изображений.

- ResNet (например, ResNet50, ResNet101) - сети с остаточными блоками, позволяющими обучать очень глубокие архитектуры без потери производительности.

- Inception (например, InceptionV3) - сети с модулями, которые параллельно обрабатывают входные данные разными сверточными блоками.

- Xception - архитектура, основанная на глубоких сверточных сетях с раздельными свертками, улучшающими эффективность и производительность сети.

## Влияние количества эпох обучения на результаты сети MobileNet:

- Количество эпох обучения влияет на способность нейронной сети обучаться и адаптироваться к данным. С увеличением числа эпох сеть может лучше настроить свои веса на специфические особенности набора данных.

- Недообучение: если эпох слишком мало, сеть может не успеть «научиться» и показать низкую производительность.

- Переобучение: если проводить обучение слишком много эпох, особенно без механизмов предотвращения переобучения (например, ранняя остановка, регуляризация), сеть может "запомнить" обучающие данные и плохо работать на новых, неизвестных данных.

- Оптимальное число эпох подбирается экспериментально, исходя из динамики изменения функции потерь и точности на обучающем и валидационном наборах данных.

## Сортируем фото разных моделей машин: Ferrari и Vaz (рисунки 13-16)

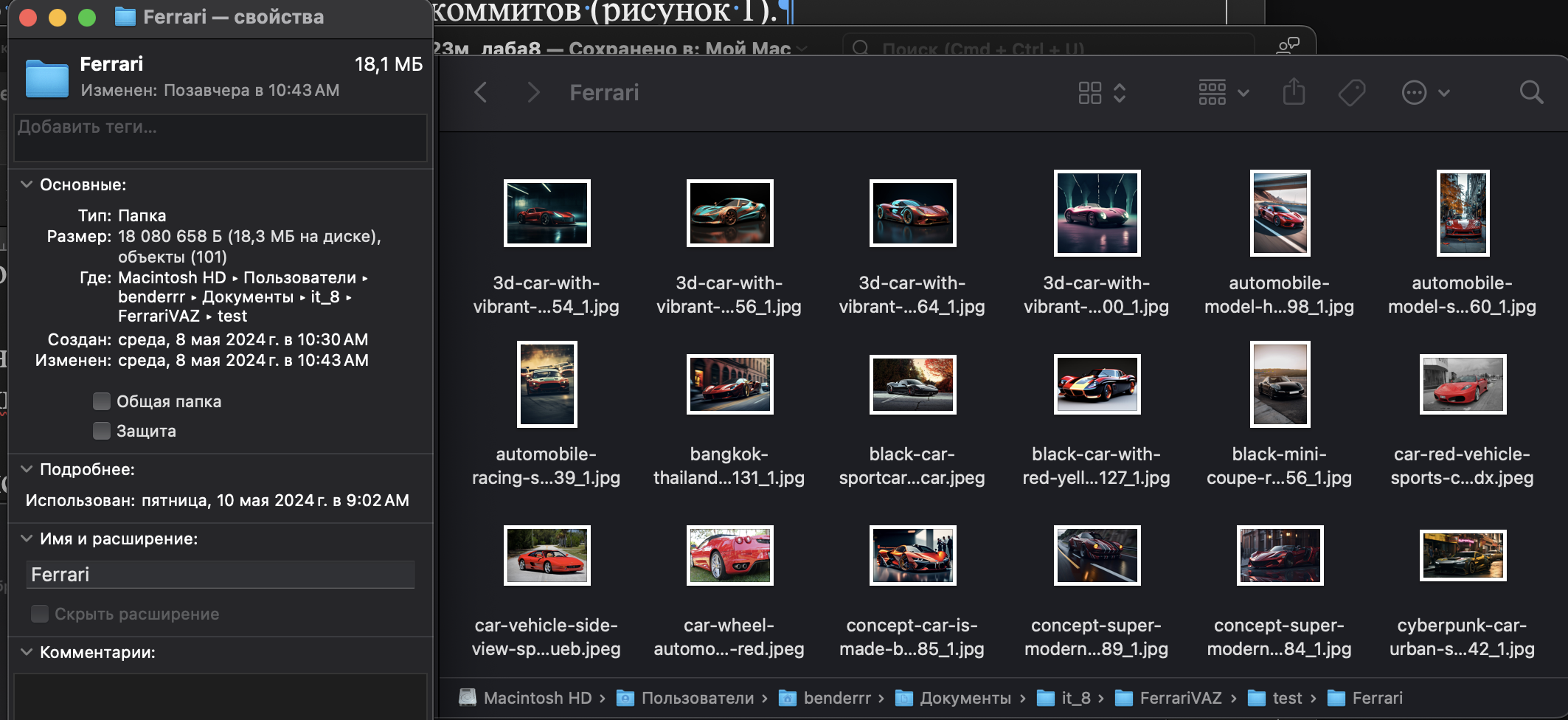


Рисунок 13 – Фотографии Ferrari

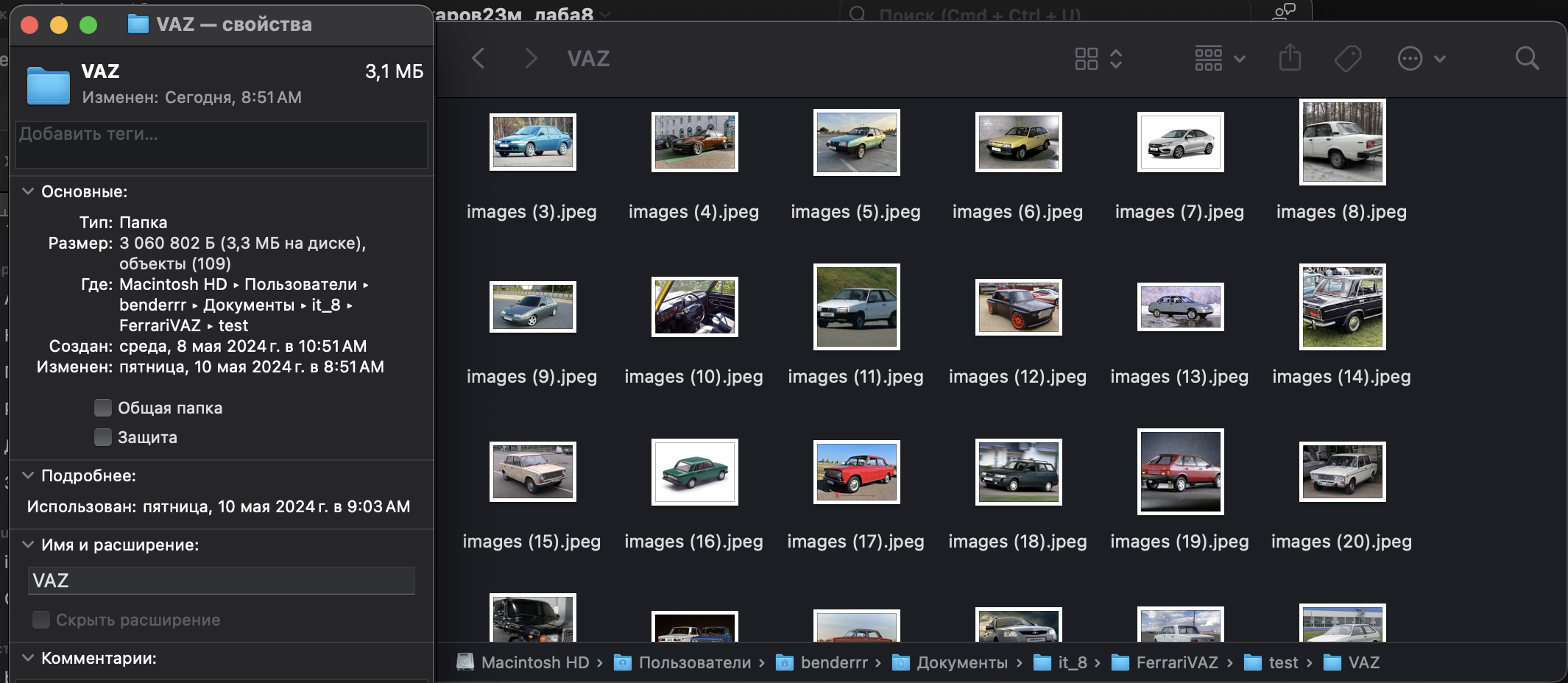


Рисунок 14 – Фотографии VAZ

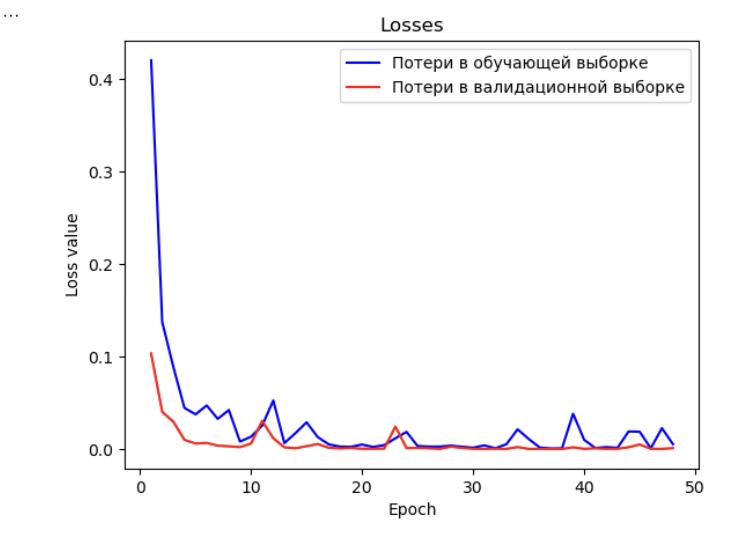


Рисунок 15 – График после выполнения сортировки

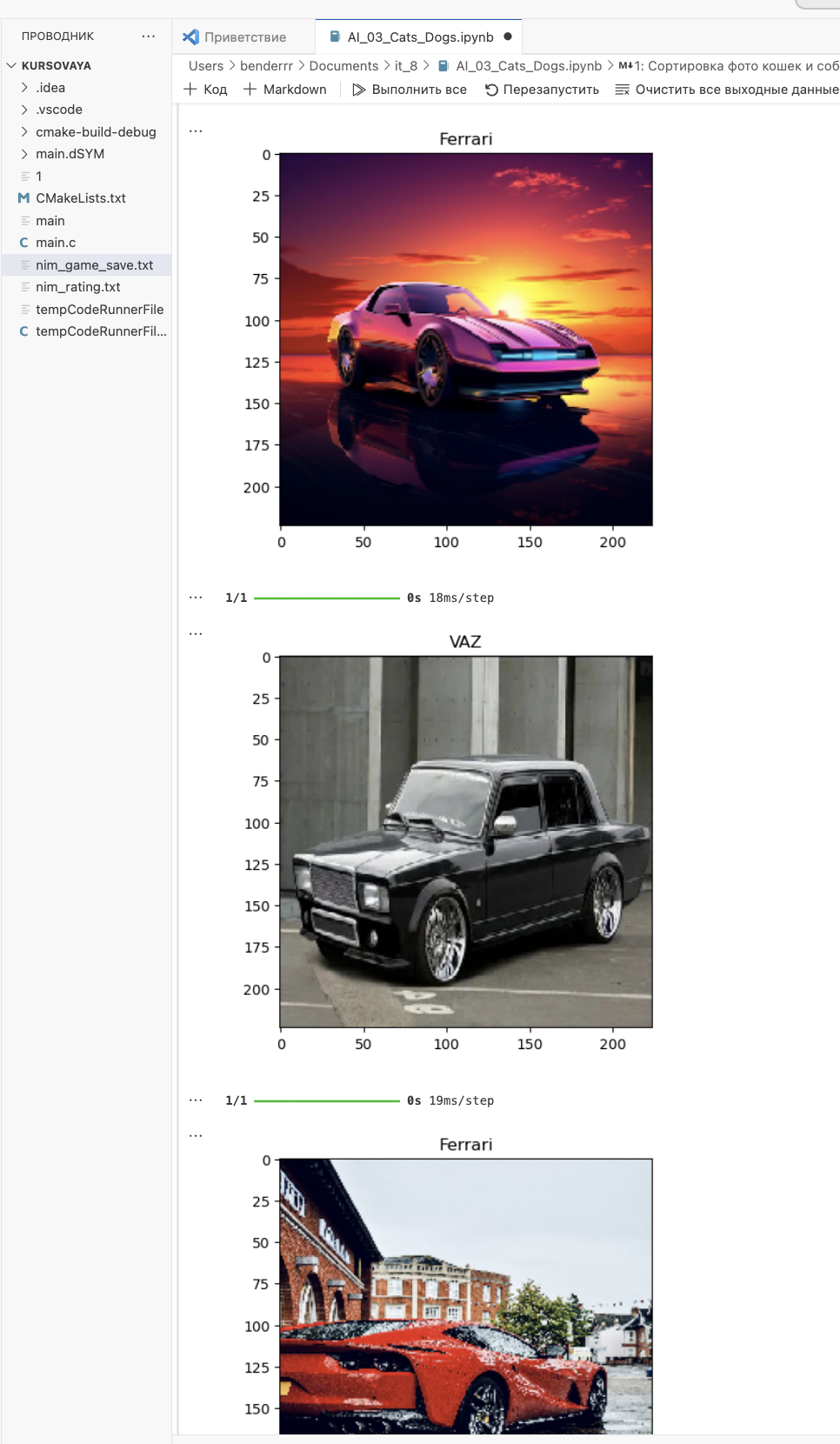


Рисунок 16 – Результат сортировки изображений

Вывод

В ходе данной лабораторной работы мы познакомились с языком программирования Python. Мы научились использовать этот язык для обучения нейронной сети. Испытали эту сеть сортируя фото кошек и собак.