Министерство образования и науки Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Факультет прикладной математики и компьютерных наук

ОТЧЕТ

По заданию №3 «Сверточные нейронные сети для классификации изображений. Передача обучения»

по дисциплине «Нейронные сети и глубинное обучение»

Подготовил студенты группы №932101:

Насутион Рафли Аулиа Ризки

Проверил:

Доцент кафедры теоретических основ информатики

С. В. Аксёнов

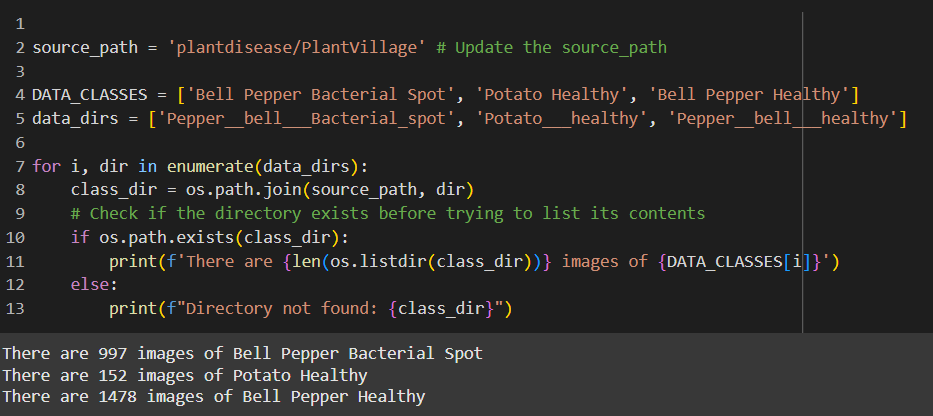
Томск – 2024

# Цель работы

Целью данной работы является получение навыки создания и обучения нейронных сетей для классификации изображений, а также переноса обучения.

# Задачи

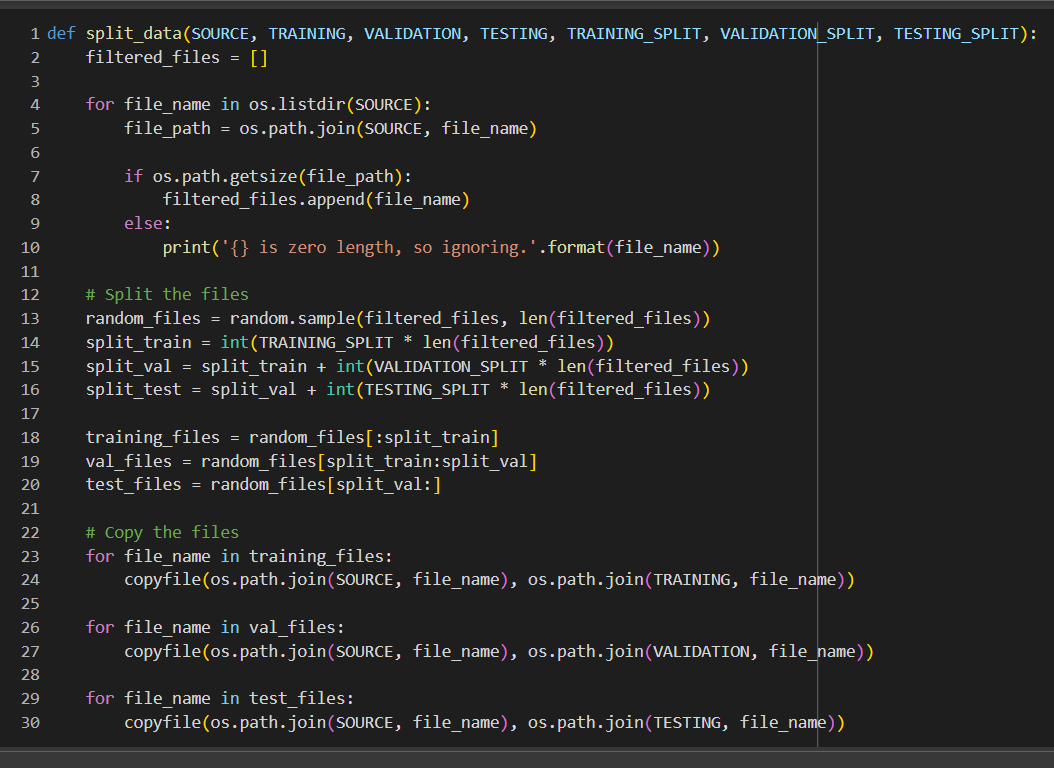
Данная работа составляет из 2-х задач для построения нейросетевых моделей: бинарный классификатор и многоклассовый классификатор. Также в качестве задач был получен 4й вариант: датасет болезнь растений (<https://www.kaggle.com/datasets/emmarex/plantdisease>) и для класса 0 – Pepper\_bell\_Bacterial\_spot, класса 1 – Potato\_healthy, и класса 2 – Pepper\_bell\_healthy. Датасет содержит из 997 изображений класса 0, 152 изображения класса 1 и 1478 изображений класса 2.

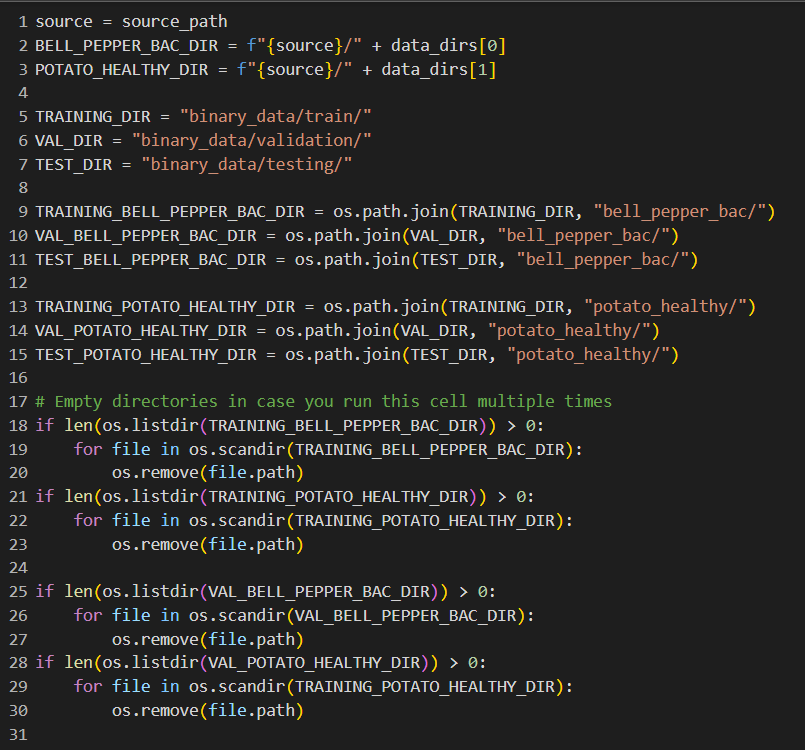


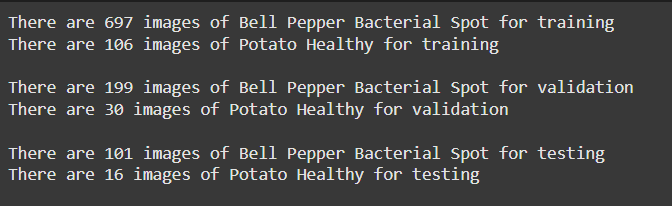
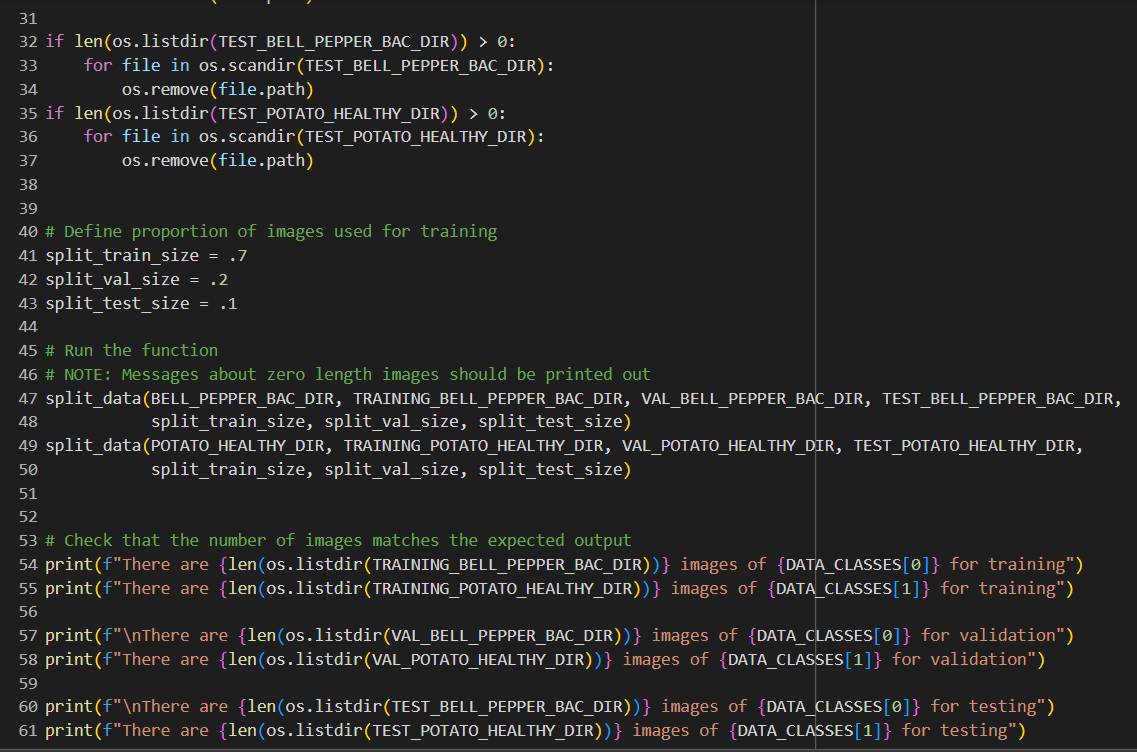
Целью данного классификатора является правильное предсказание изображения принадлежит в каком классе.

## Разделение данных на обучающий, проверочный и тестовый набор данных

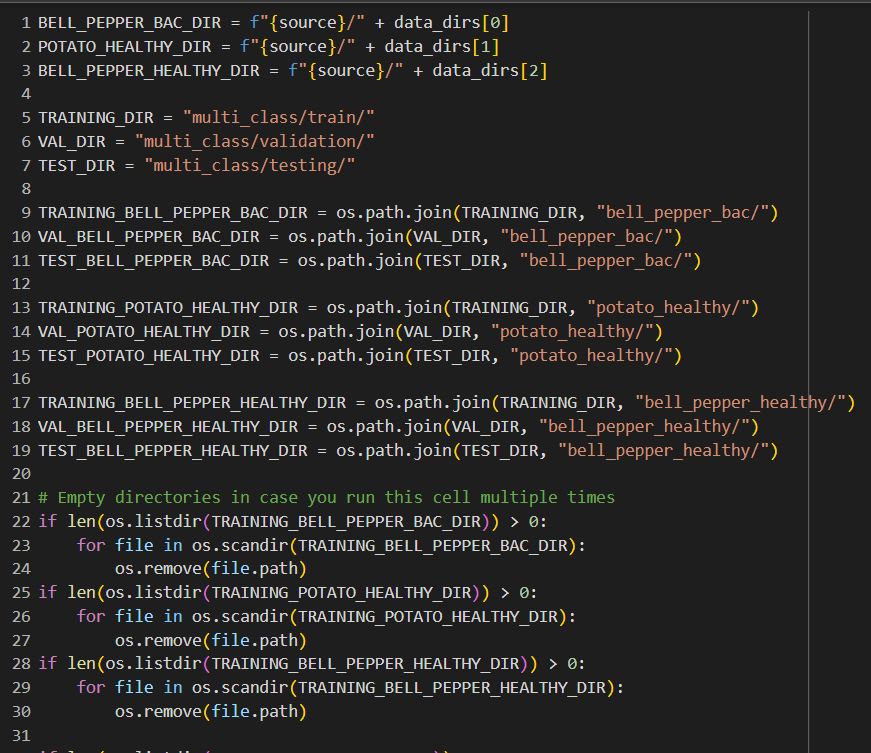
Перед тем как начать работать под построением модель, нам нужно разделить данные на 3 наборы (обучающий, проверочный и тестовый набор) для дальнейшей работы. Для этого мы создадим функцию чтобы упросить этот процесс, далее мы вызываем эту функцию и получим папку с разными папками которые разделить данные на обучающий, валидационный и тестовый набор данных.

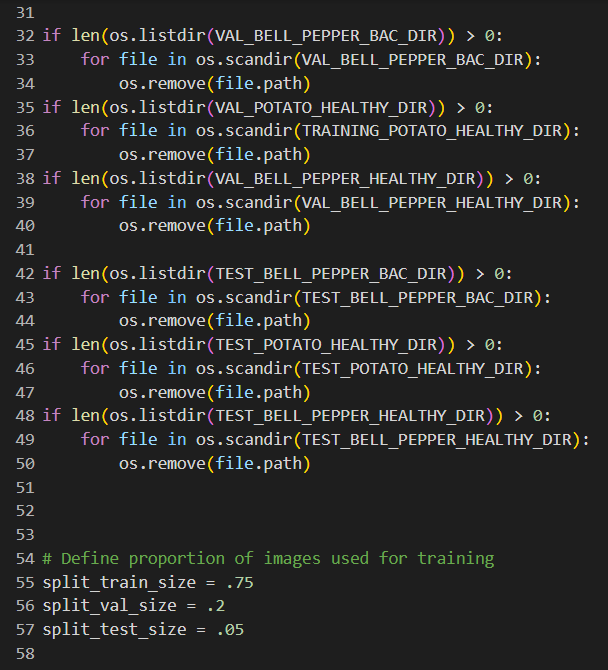






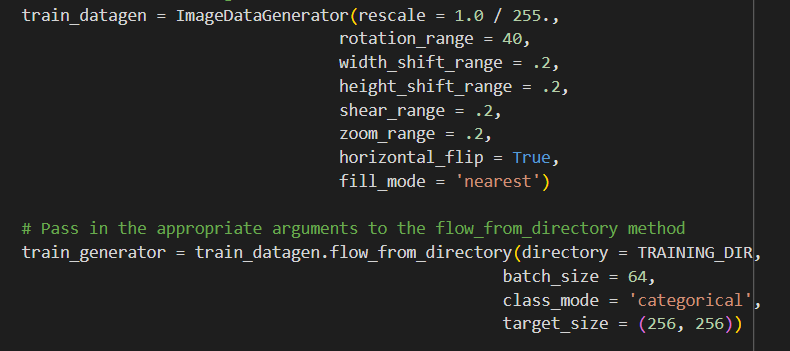
Этот код выше используется для распределения бинарных данных (данные принадлежащие в классе 0 и 1).



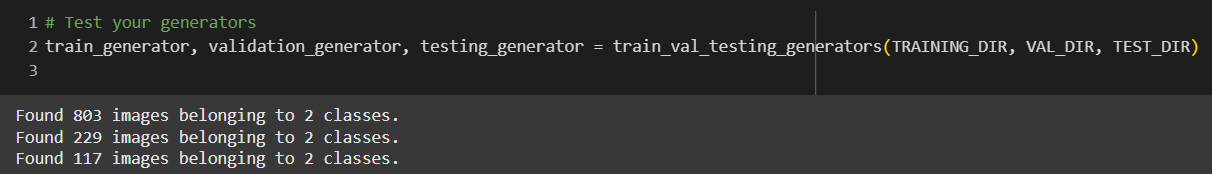


## Обработка и распределение данных

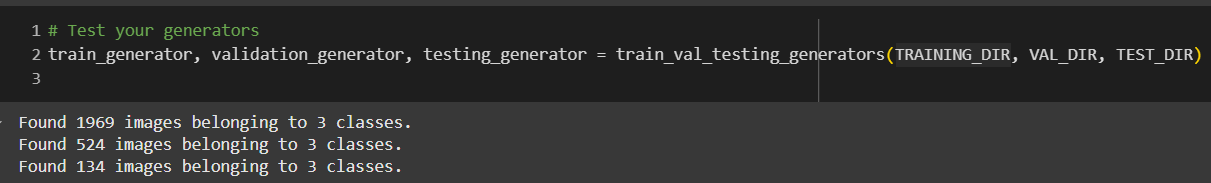
Для того чтобы модель хорошо работалась, нам нужно преобразовать наше изображение так чтобы в каждом пикселе значение было в диапазоне 0 – 1, и поэтому мы делим все значение на 255. Также мы сделаем функцию для генерирования партий данных поскольку данных у нас очень много. Функция ImageDataGenerator от библиотеки tensorflow.keras.preprocessing.image была использована.



В итоге мы получим партии для каждых наборов данных. Код ниже используется для генерирования бинарных данных.

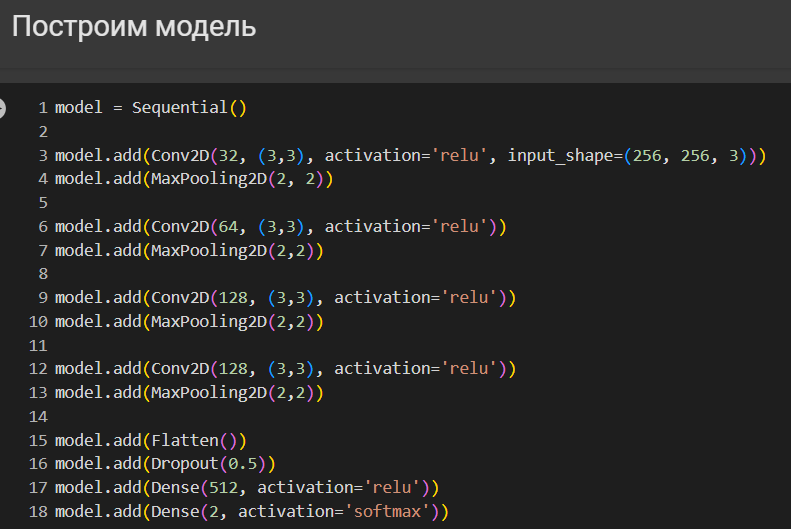


А код ниже используется для генерирования многоклассовых данных.

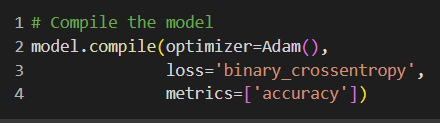


## Бинарный классификатор

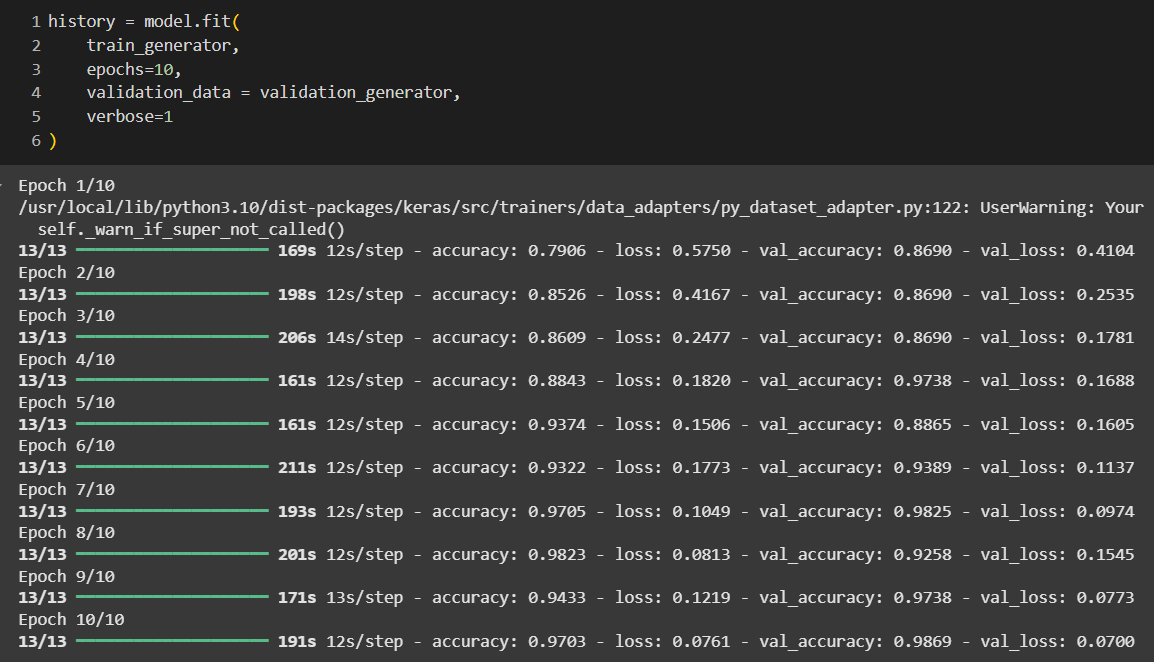
Мы создадим модель бинарного классификатора с помощью Sequential() от библиотеки tensorflow.keras.models. Мы построили модель с 9 скрытыми слоями с функцией активации ReLu. Выходной слой с 2-мя перцептронами и функцией активацией softmax. Тут input\_shape для первой слой (256,256,3) потому что наше изображение имеет 256 пикселей на высоте и ширине также 3 цветовые канали.



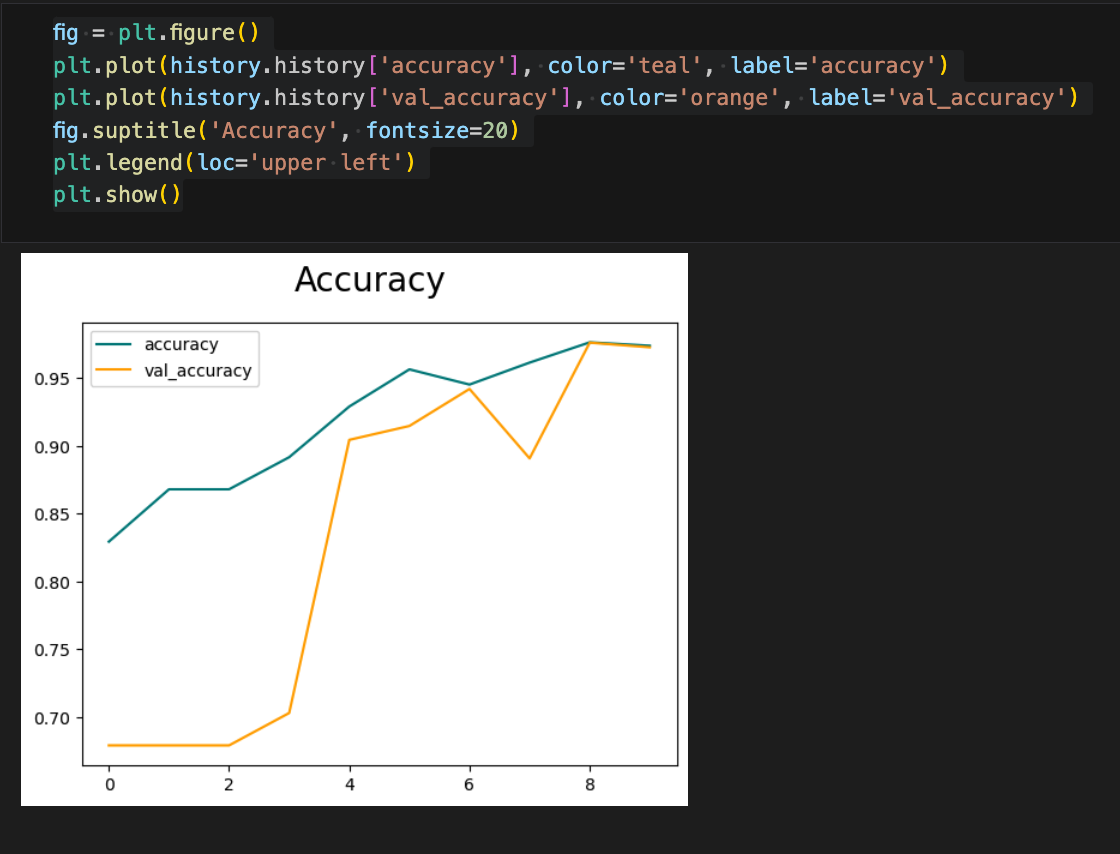
Далее мы компилируем эту модель с binary crossentropy в качестве loss function, adam в качестве optimizer, и accuracy в качестве metrics.



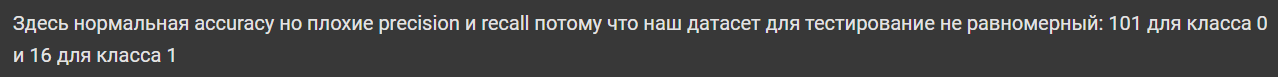
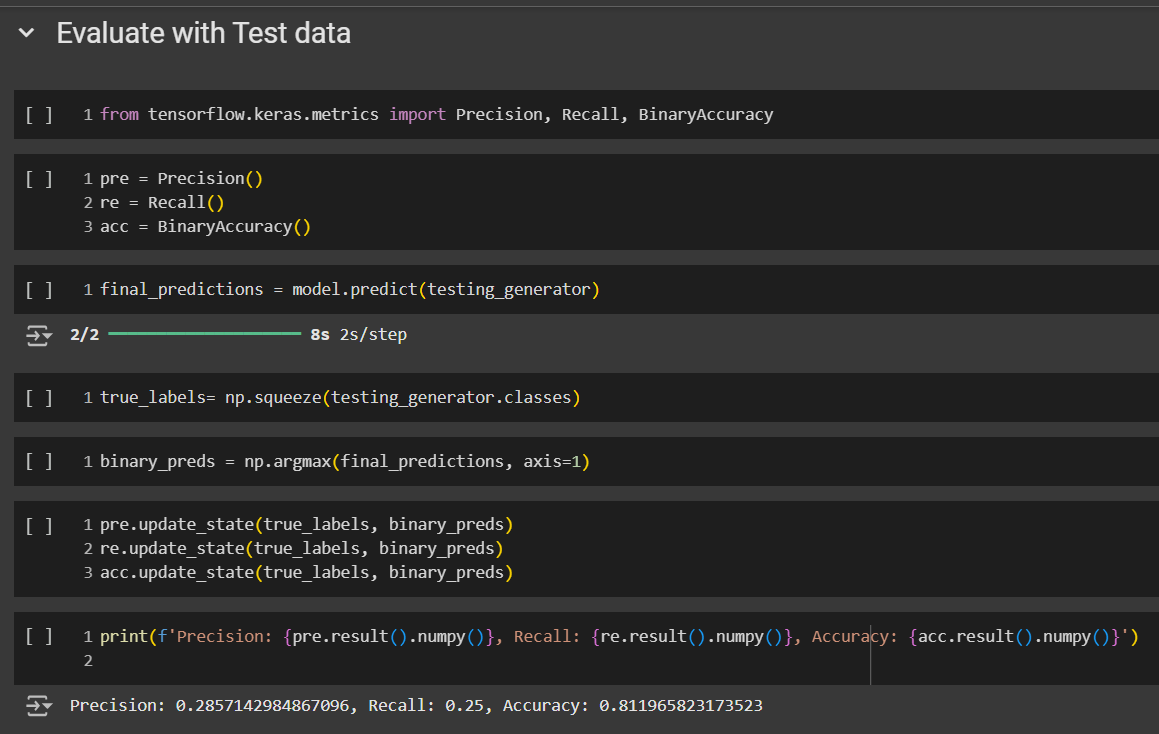
Потренируем модель в 10 эпохах которая изучаем изображения от 13 партии. Далее выводим результаты тренировки.



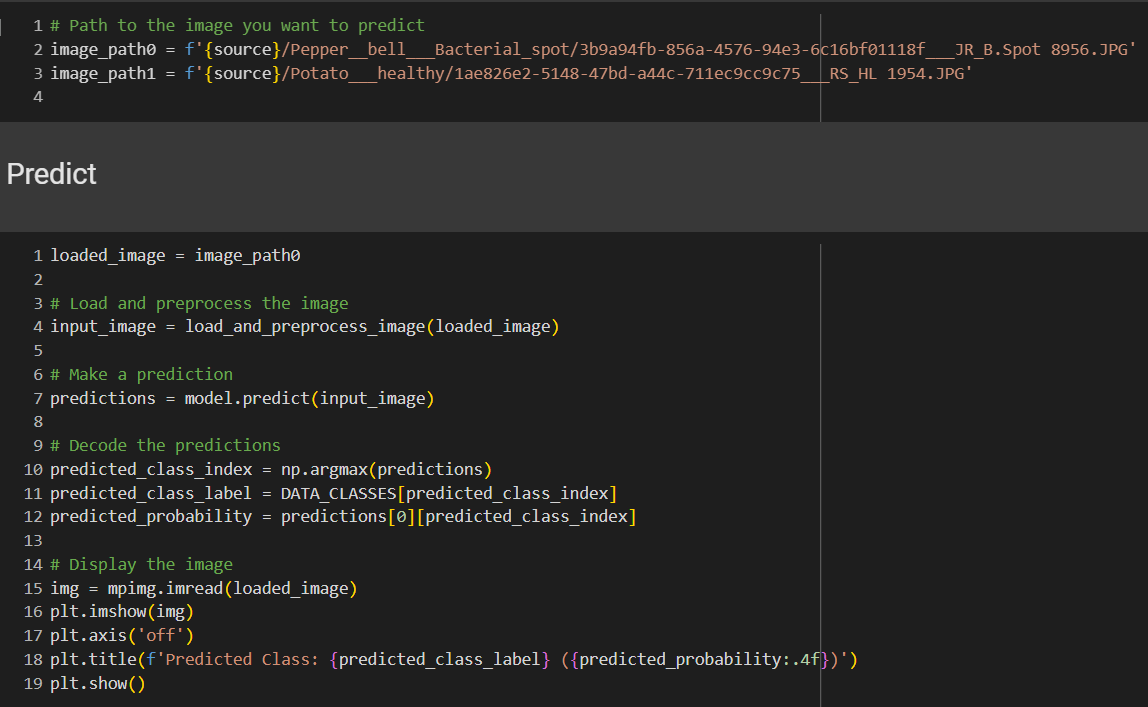


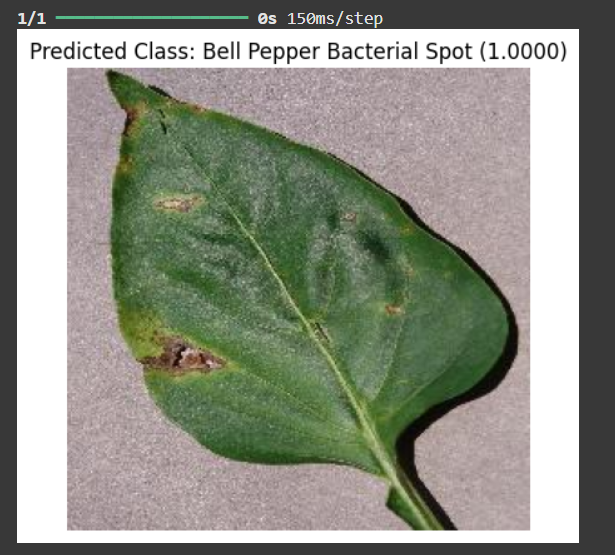


Поскольку результат тренировки мы считали уже хорошо, мы можем начать прогноз с тестовыми данными. Мы увидим, что наша модель нормально работает 81.2% accuracy (точность) но для значений precission и recall слишком маленькие потому что наш датасет для тестирование не равномерный. У класса 0 – 101 изображение, а у класса 1 – только 16 изображений.



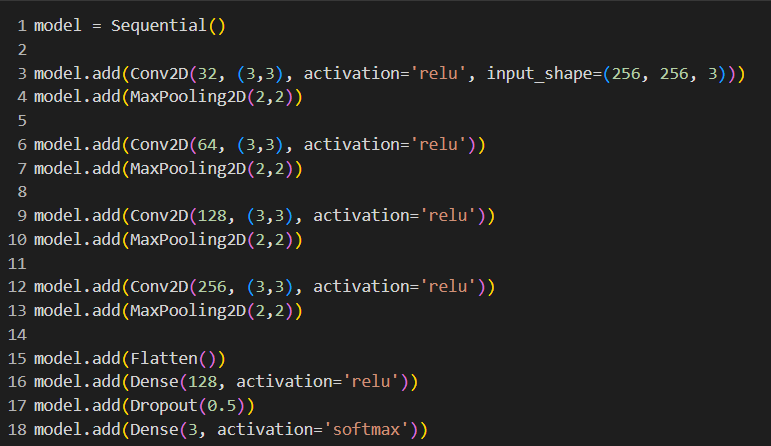
## Тестирование на реальные данные



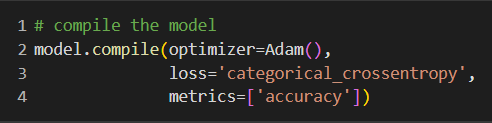


## Многоклассовый классификатор без предварительно обученной модели

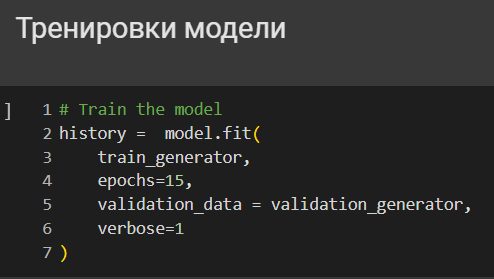
Построили модель с тем же методом как у бинарного классификатора, но разница только в выходном слое который содержит 3 перцептрона которые соответствуют с классам (1, 2 и 3).

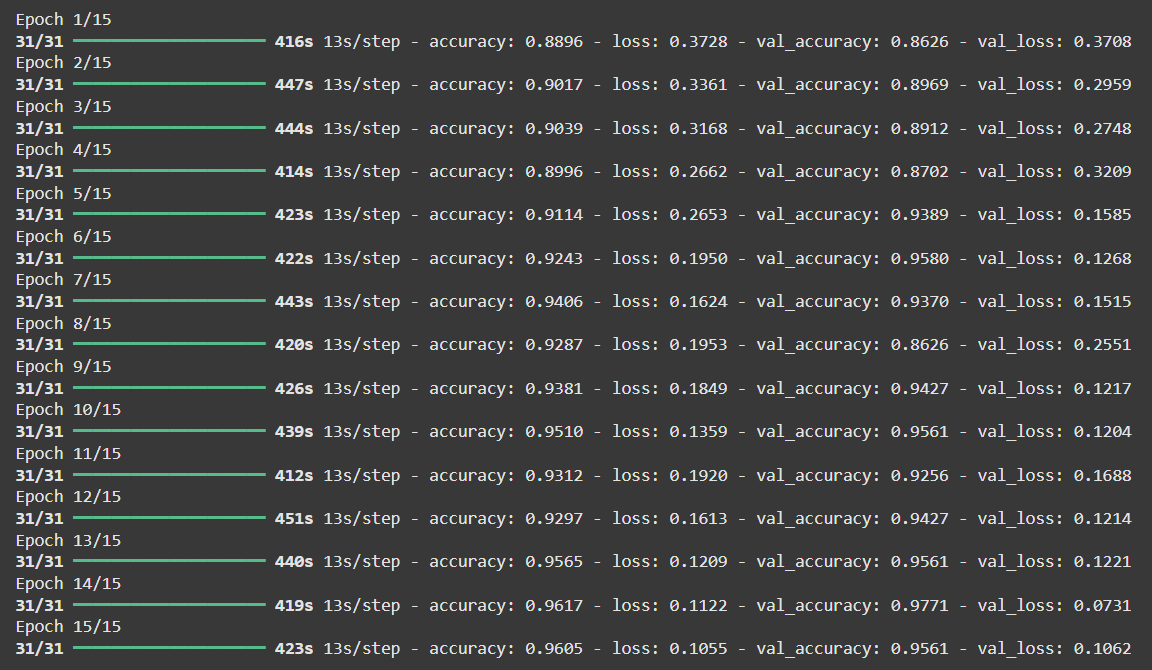


Скомпилируем модель с функцией loss categorical crossentropy.

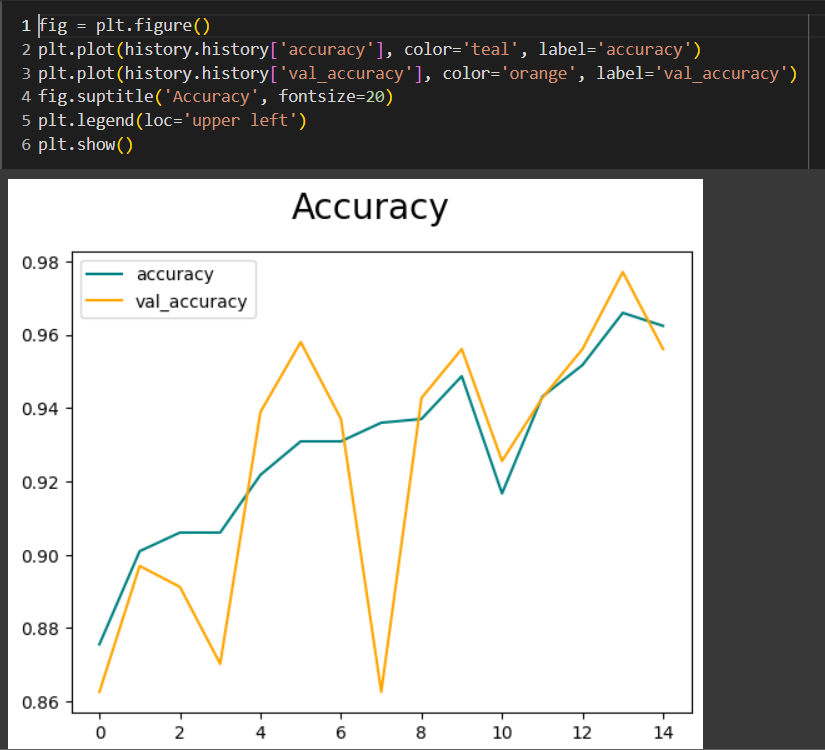


Потренируем модель в 15 эпохах и выводим результаты.



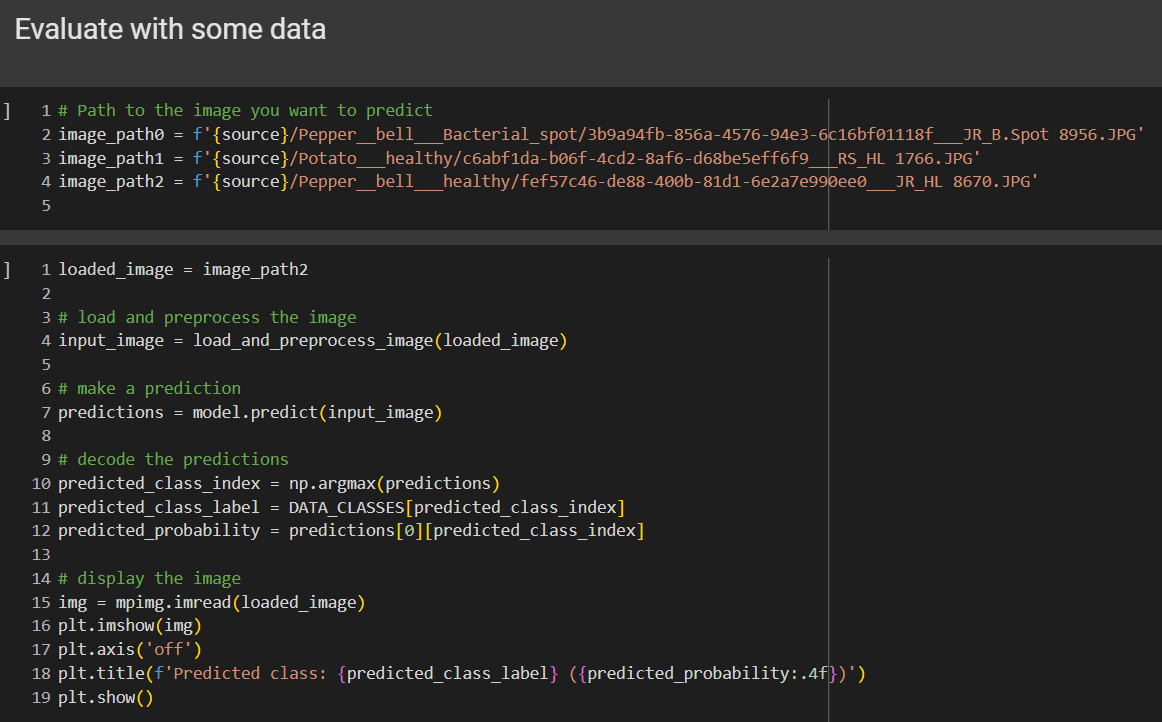


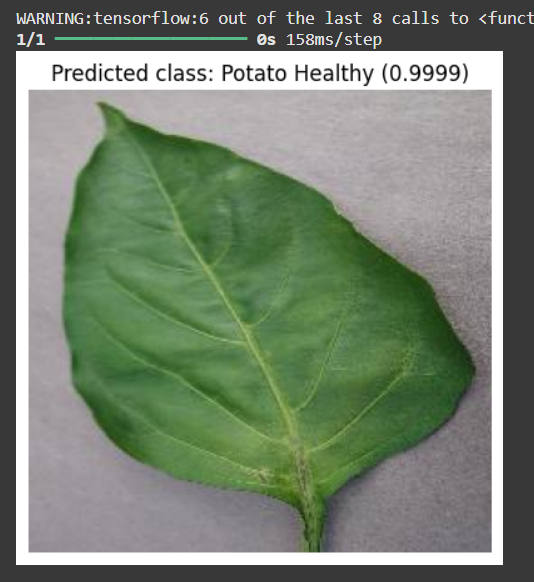




Поскольку результат тренировки мы считали уже хорошо, мы можем начать прогноз с тестовыми данными. Мы увидим, что наша модель сейчас не очень хоршо работает с 55.9% accuracy но сейчас precission иrecall уже стали хорошими потому что данные для тестирование уже более сбалансированы.

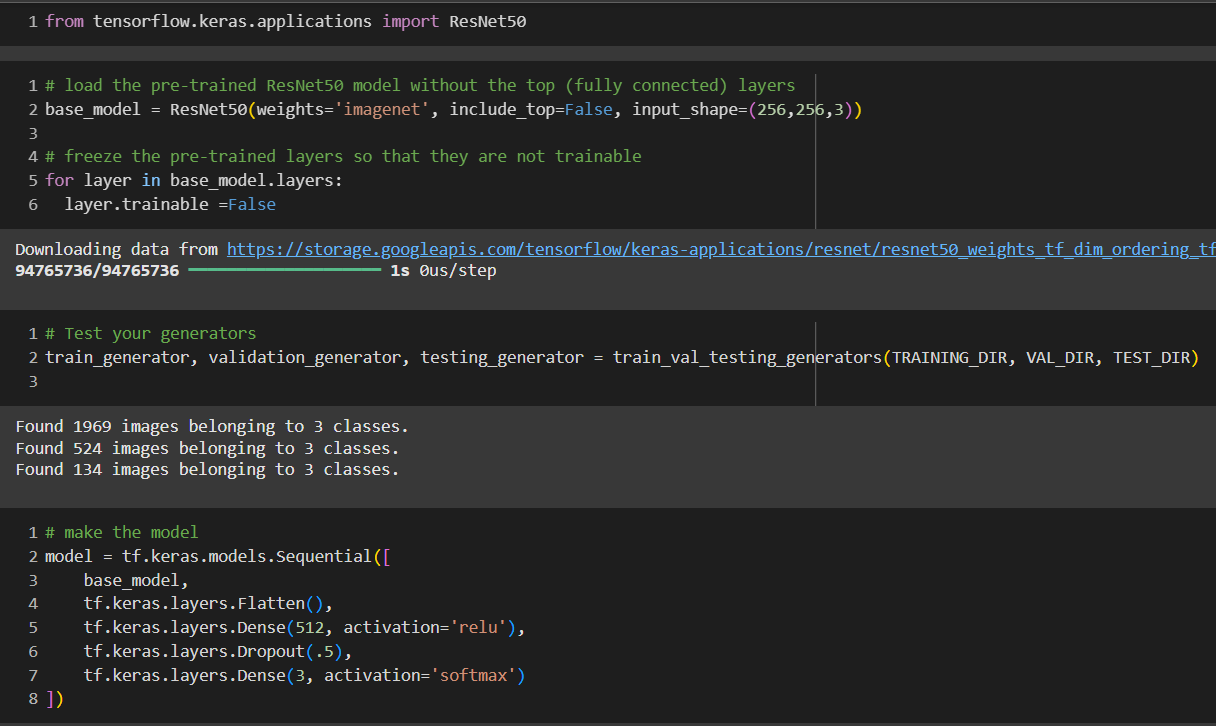
## Тестирование на реальные данные



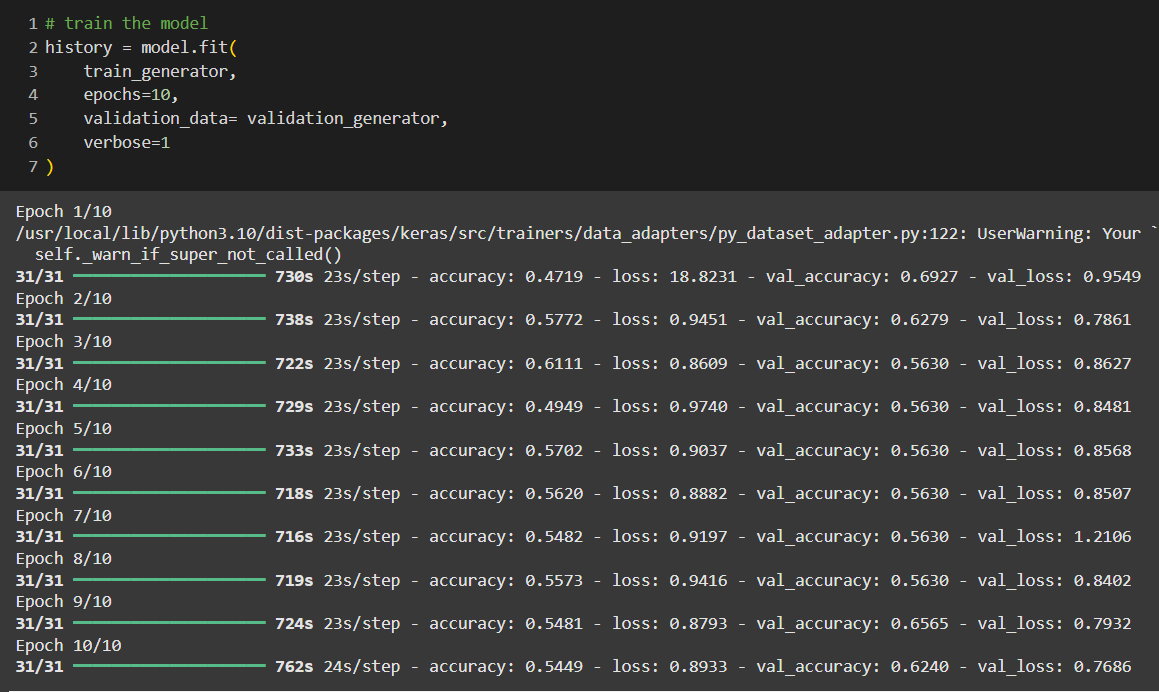


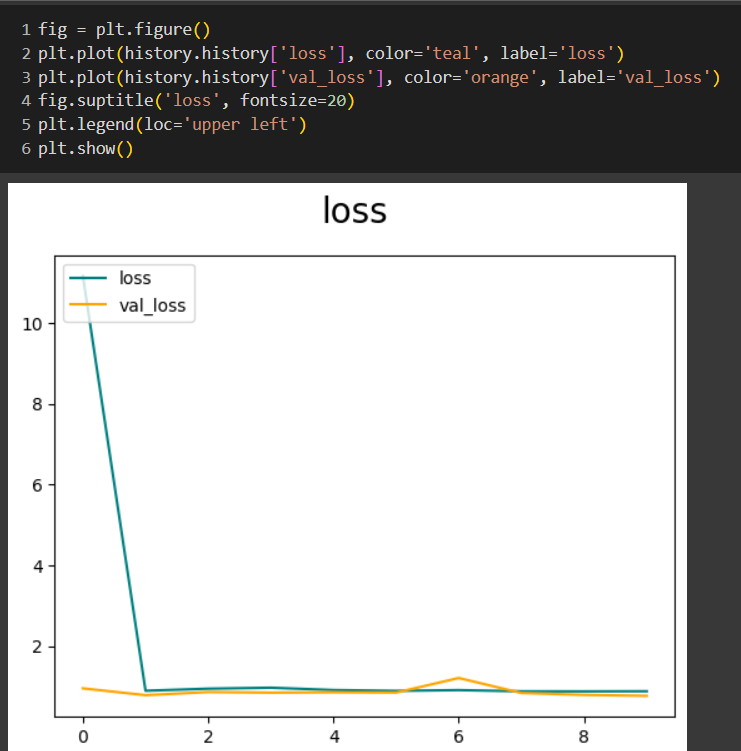
## Многоклассовый классификатор с предварительно обученной моделью

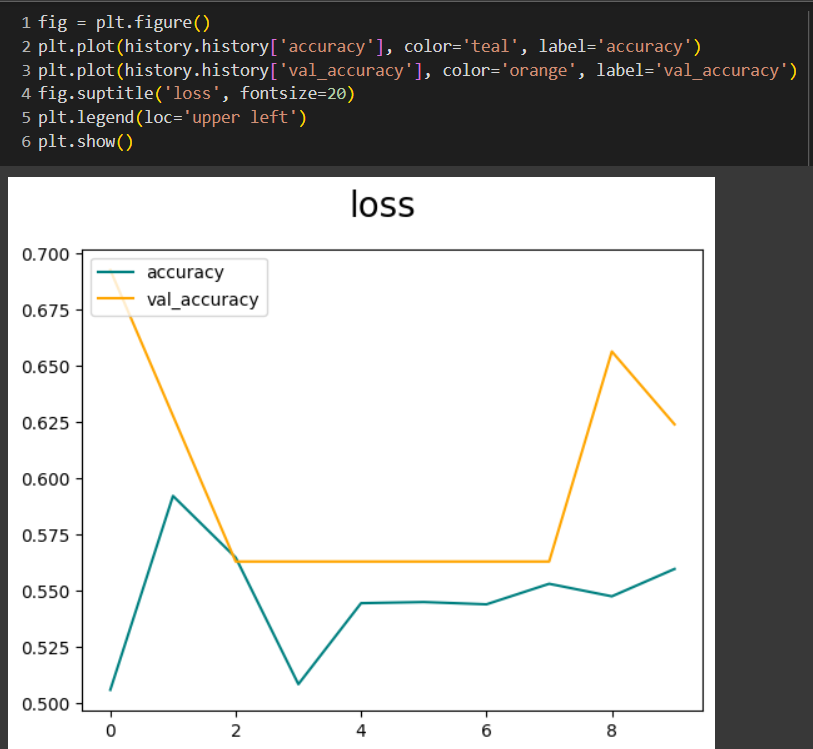
Сейчас мы будем использовать pre-trained модель ResNet50 от библиотеки tensorflow.keras.applications для построения скрытых слоёв.



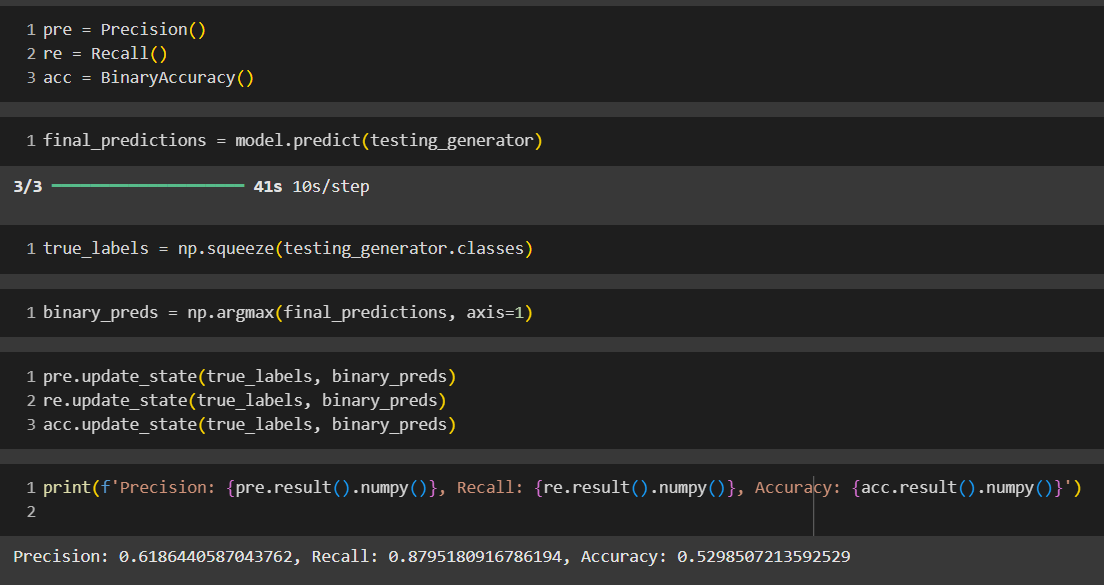
А сейчас мы потренируем нашу модель в 10 эпохах и выводим результаты.



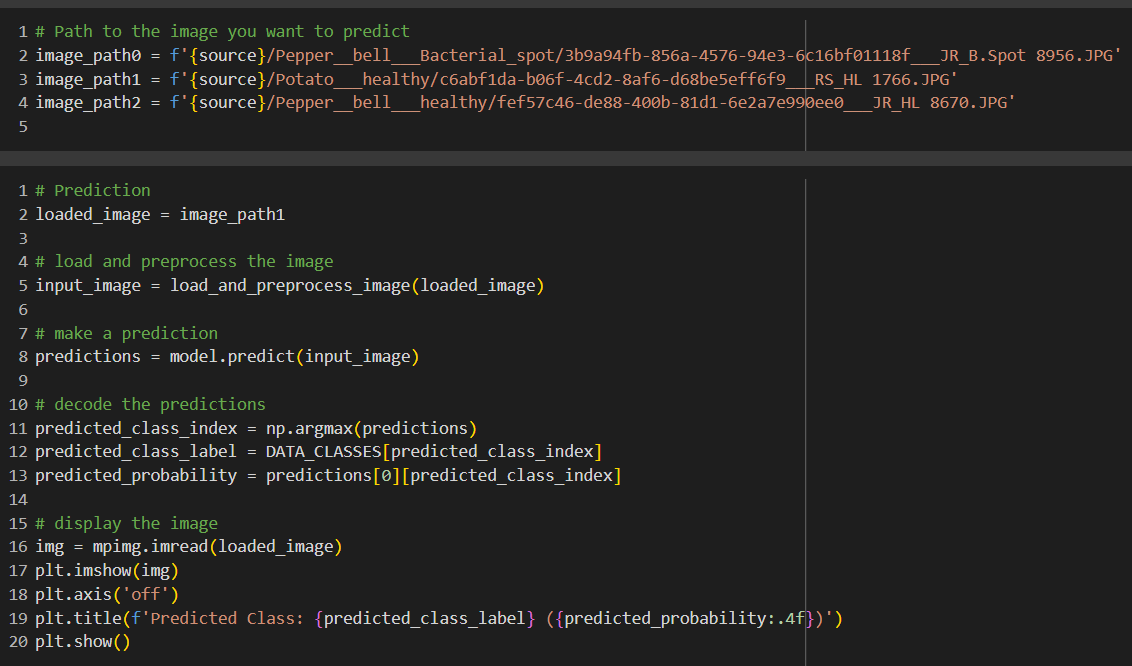


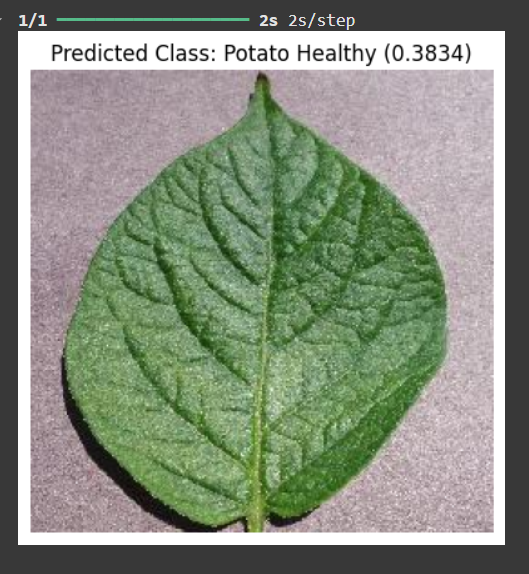


Поскольку результат тренировки мы считали уже хорошо, мы можем начать прогноз с тестовыми данными. С этой моделью значение loss стабильно хотя accuracy не очень хорошо. Мы увидим, что наша модель нормально работает (53%).



## Тестирование на реальные данные





# Заключение

В ходе работы были построены 2 нейронные сетевые модели классификаторы (бинарный, многоклассовый) с помощью разных библиотек в Python. Цель данной работы достигнута. Код и Python Notebook выложен в Google Colab (https://colab.research.google.com/drive/1PDv8pIfHBAxP-IeWjdCu2DaGbSAbrPDL?usp=sharing)