UNIVERSITY OF MUMBAI

PROJECT REPORT ON

PLANT DISEASE PREDICTION

PROJECT

SUBMITTED BY

ASHISH RAMJANAM MALLAH

UNDER THE GUIDANCE OF

PROF. AQUILA SHAIKH

LATE BHAUSAHEB HIRAY SMARNIKA SAMITI TRUST

HIRAY GROUP OF INSTITUTES

MUMBAI - 400051

MAHARASHTRA

MCA SEM III [2021-2022]

LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE OF COMPUTER APPLICATION

ISO 90012008 CERTIFIED

S.N. 341, Next to New English School, Govt. Colony,

Bandra (East), Mumbai – 400051,

Tel: 91-22-26570892/3181

Date:

CERTIFICATE

This is to certify that Mr. ASHISH RAMJANAM MALLAH

----------Roll No. 202114

is a student of MCA of 2th year Semester-III has completed successfully full-semester Mini-Project of subject "PLANT DISEASE PREDICTION" for the academic year 2020 – 21.

---------- **----------**

Subject In-Charge **Director**

External Examiner

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No.:- 2017016400250142 **SEAT No.:-**202114

Name of the Student:- ASHISH RAMJANAM MALLAH

Title of the Project:- PLANT DISEASE PREDICTION

Name of the Guide:- Prof. AQUILA SHAIKH

Teaching experience of the Guide:

Is this your first submission? Yes No

_____ _____

Signature of the Student **Signature of the Guide**

Date: _____ Date:_____

Signature of the Coordinator: _____

Date: _____

ACKNOWLEDGEMENT

I extend my deepest appreciation to my esteemed guide, **Prof. AQUILA SHAIKH & RASHMITA PRADHAN** for providing me with the possibility to complete this project with the right guidance and advice.

Special gratitude I give to my respected head of the division **PROF. VIKRAM PATALBANSI**, for allowing me to use the facilities available and also help me to coordinate my project Furthermore, I would also like to acknowledge with much appreciation the crucial role of faculty members on this occasion.

Last but not least, I would like to thank friends who help me to assemble the parts and gave a suggestion about the project.

Abstract

This is end to end deep learning project in agriculture domain. Farmers every year face economics loss and crop waste due to various diseases in plants. We will use image classification using CNN and built a application using which a farmer can take a picture and application will tell you if the plant has a disease or not.

TABLE OF CONTENTS

CONTENTS		PAGE NO.
INTRODUCTION 1)Introduction and Importance 1.1) Topic Intro 1.2) Need and Motivation		
DATA SET 2)Steps in Preparing Data for Model 2.1) Data Collection 2.2) Data Cleaning 2.3) Feature Engineering 2.4) Training a model		
LANGUAGE AND MODELS USED 3.1) Python 3.2) Numpy and Pandas for data cleaning 3.3) Matplotlib for data visualization 3.4) Sklearn for model building 3.5) Jupyter notebook, google colab & pycharm as IDE 3.6) create a app using Streamlit **3B) Models Used** 3B.1) CNN		
4) RESULTS AND DISCUSSIONS 4.1) BEST SUITED CNN MODEL SCORE 4.2) DEPLOYMENT APP		

5) ABBREVIATIONS		
6) CONCLUSION		
7) REFERENCE		

LIST OF TABLES

CONTENT		PAGE N0.
4.1)Plants Disease image Dataset		
4.2) Transformed Dataset images		

LIST OF FIGURES

Contents		Page No.
1. Real Vs Predicted Plots		
2. Comparison Bar Plot		
3. Website View		

Plants Disease Prediction Project

INTRODUCTION

INTRO & STEP:_

we are beginning end to end machine learning project or data science project for Plants Disease image classification. During this project, I will implement how projects are executed in big companies in a typical corporate environment. We will try to classify an image of my 3 plants Diseases. I am just giving an introduction and going over business requirements etc. In next one we will talk about data collection. While working on this project you will learn,

Steps:-

1) In this project we will create a Convolutional Neural Network which will be able to predict whether a plant is suffering from a disease. We will use different layers and other hyperparameters for building, training and testing this classifictaion model.We will be using tensorflow and keras for this project.

2) First we will mount our google drive on colab so that we can use the dataset directly from our drive. For this you first need to upload the data on your drive and then mount the drive on colab.

3) After mounting our drive we will locate the folder where our data is stored to use it in our colab notebook. Here you can see that I have 2 folders in my drive and 'Plant images' contains the images that we will work on.

4) Next we will import all the required libraries. As we are making a CNN model we will import all the required layers, activations, optimizers, etc.

5) Now we will observe some of the images that are their in our dataset. We will plot 12 images here using the matplotlib library.

6) After visualizing the images let us move forward and create a function which will convert the images into a numpy array. It is required because we will normalize our dataset after this.

7) Now we will convert all the images into numpy array.

8) We will also observe the number of images under different classes to see if the dataset is balanced or not

9) Next we will observe the shape of the image.

10) Checking the total number of the images which is the length of the labels list.

11) Next we will use sklearn train_test_split to split the dataset into testing and training data. Here I have taken test size as 0.2 so my data will be divided into 80% training and 20% testing data.

12) Now we will normalize the dataset of our images. As pixel values ranges from 0 to 255 so we will divide each image pixel with 255 to normalize the dataset.

13) Next we will create a network architecture for the model. We have used different types of layers according to their features namely Conv_2d (It is used to create a convolutional kernel that is convolved with the input layer to produce the output tensor), max_pooling2d (It is a downsampling technique which takes out the maximum value over the window defined by poolsize), flatten (It flattens the input and creates a 1D output), Dense (Dense layer produce the output as the dot product of input and kernel).

14) While compiling the model we need to set the type of loss which will be Binary Crossentropy for our model alongwith this we also need to set the optimizer and the metrics respectively.

15) Next we will split the dataset into validation and training data.

16) Fitting the model with the data and finding out the accuracy at each epoch to see how our model is learning. Now we will train our model on 10 epochs and a batch size of 128. You can try using more number of epochs to increase accuracy but here we can see that the model has already raeched a very high accuracy so we don't need to run it for more. During each epochs we can see how the model is performing by viewing the training and validation accuracy.

17) Saving the model using different techniques.

18) Next we will plot the accuracy of the model for the trainig history.

19) Evaluating the model to know the accuracy of the model.

20) Next we will use our model to predict predicting the testing dataset label.

21) Printing out the original and the predicted label.

REQUIREMENTS

❖ Hardware Requirement:

- Processor –Core i3
- Hard Disk – 160 GB
- Memory – 1GB RAM

❖ Software Requirement:

- Windows 7 or higher
- Python
- python Streamlit server

3.LANGUAGE & MODELS USED:-

Technology and tools wise this project covers:-

1) Python
2) Numpy and Pandas for data cleaning.
3) Matplotlib for data visualization.
4) Tensorflow and keras for model building using CNN.
5) Jupyter notebook, Google Colab and pycharm as IDE.
6) Using Python Streamlit server.

❖ Advantages

- Saves time
- Easy to access the system anywhere and anytime.

❖ Limitation

- Requires an active internet connection.

❖ Application

- This system can be used by the multiple peoples to get the counselling sessions online.

❖ Modules:

The system comprises of 3 major modules with their sub-modules as follows:

1. **Admin:**
    - **Add Image:** Admin can dray n drop image.
    - **View Image:** Admin can View the added image.
    - **Check Label with Class:** Admin can see Class and Label of the classify image predicted.

2. **User:**
    - **View Image:** User can view the image.
    - **View Classification Image:** User can view the predicted image with Disease classification.

DATA COLLECTION:-

There are 3 different ways of collecting data for our project,

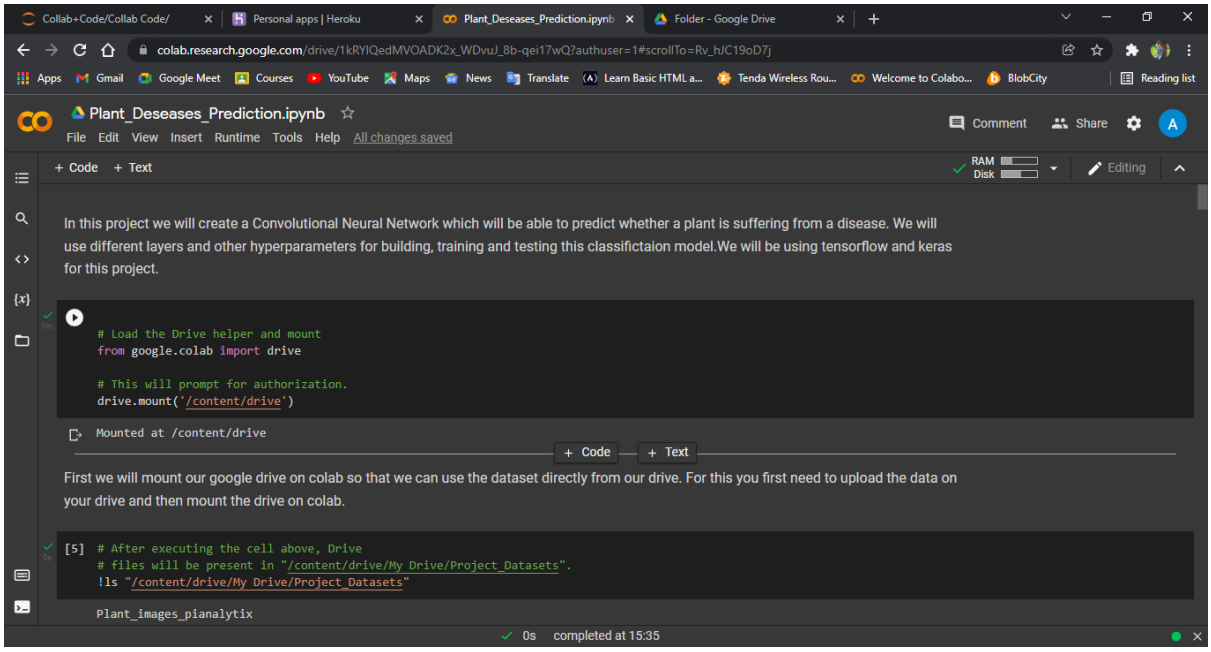
(1) Manually download images from google images

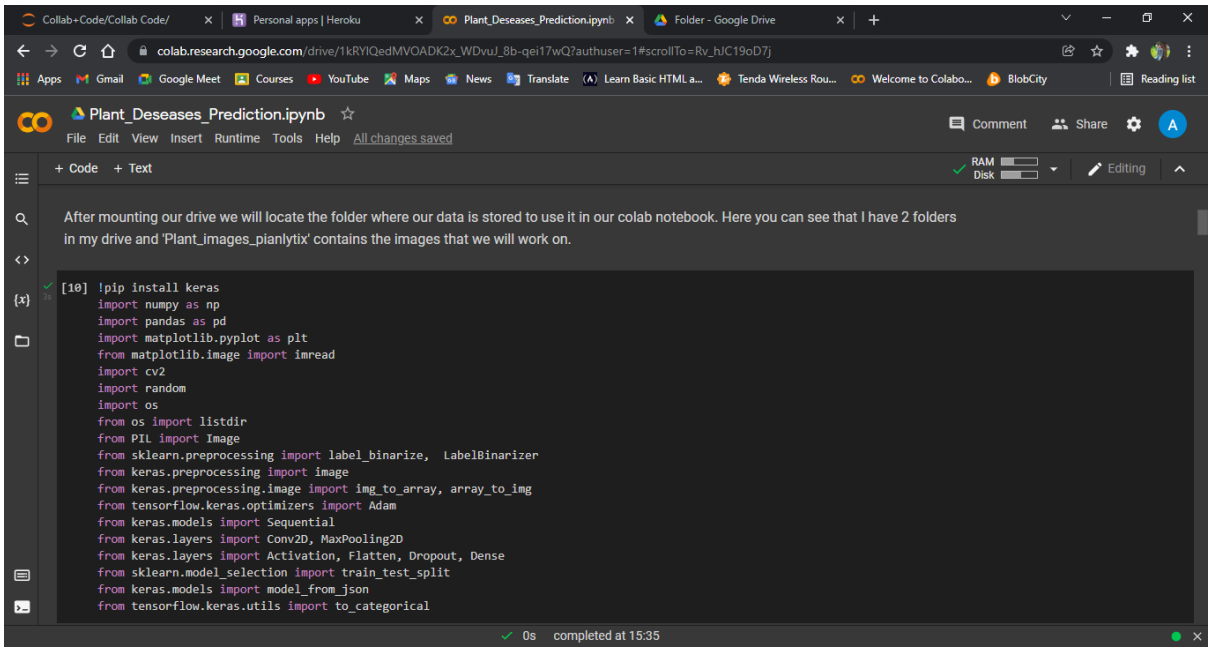
(2) Use python and web scrapping to automate downloading images from google

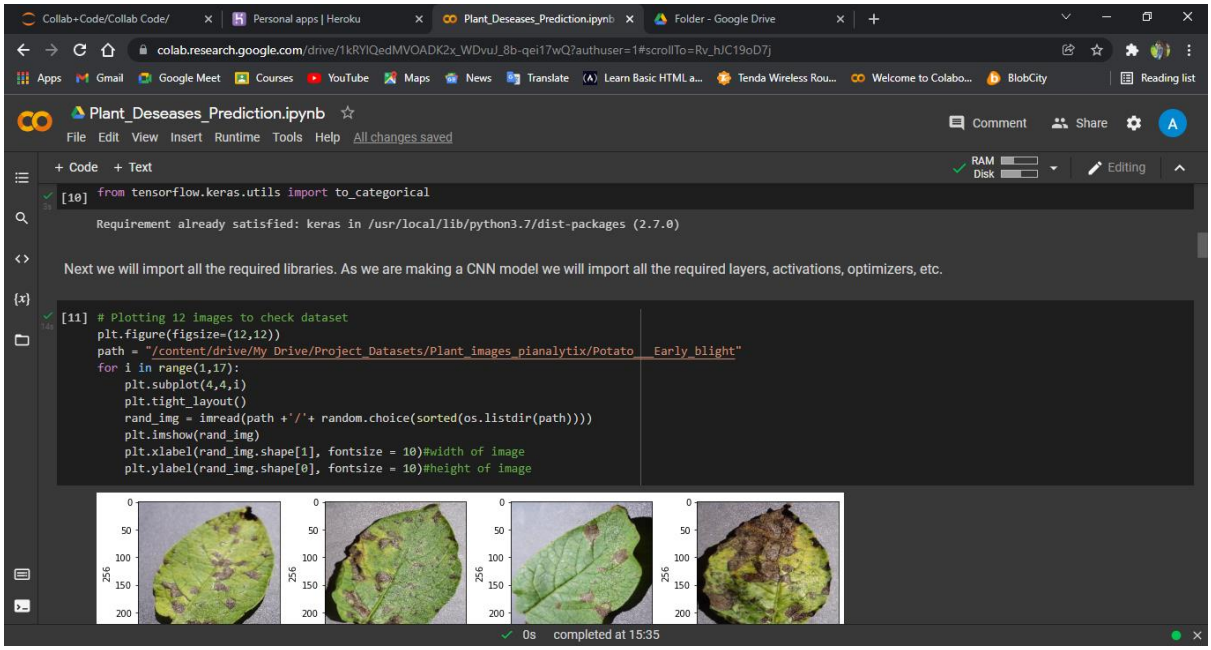
(3) Use a chrome extention called fatkun. You can download that from here: https://chrome.google.com/webstore/de...

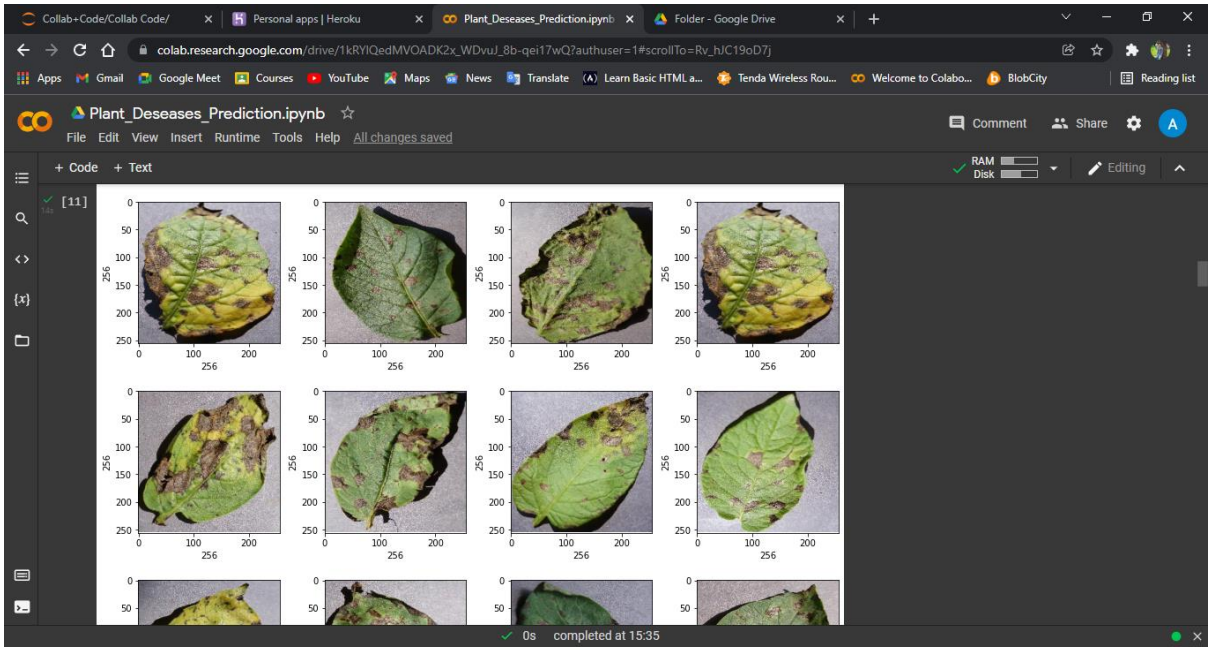
4) using kaggle datasets.

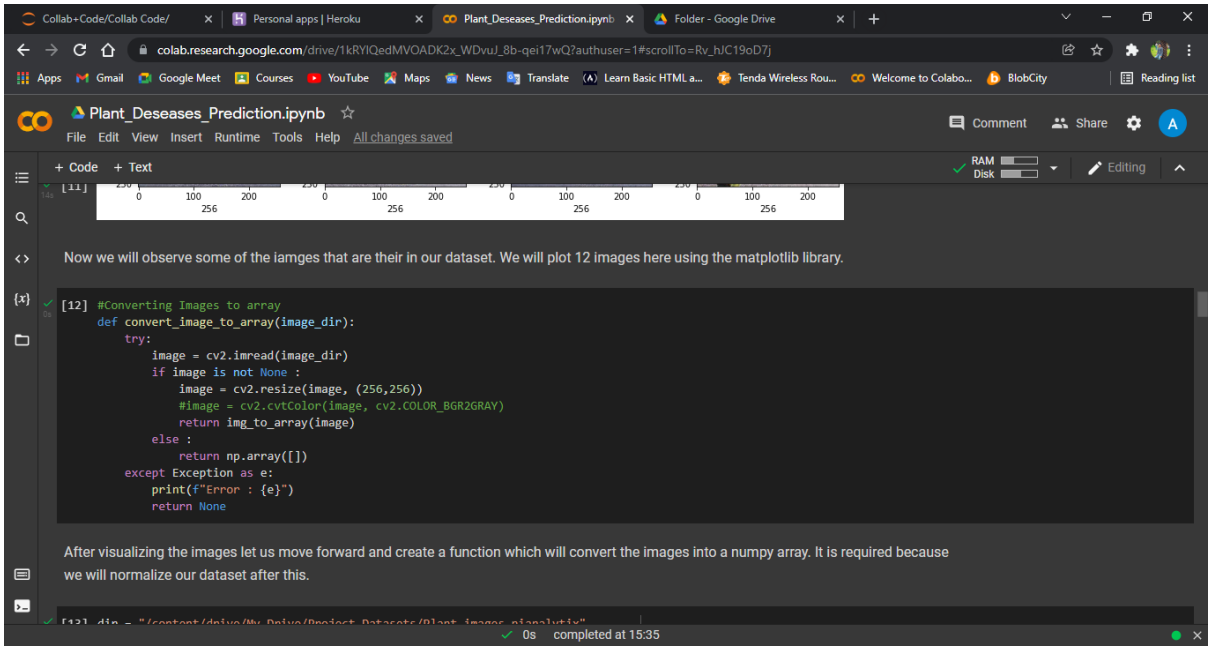
Google Colab ScreenShot:-

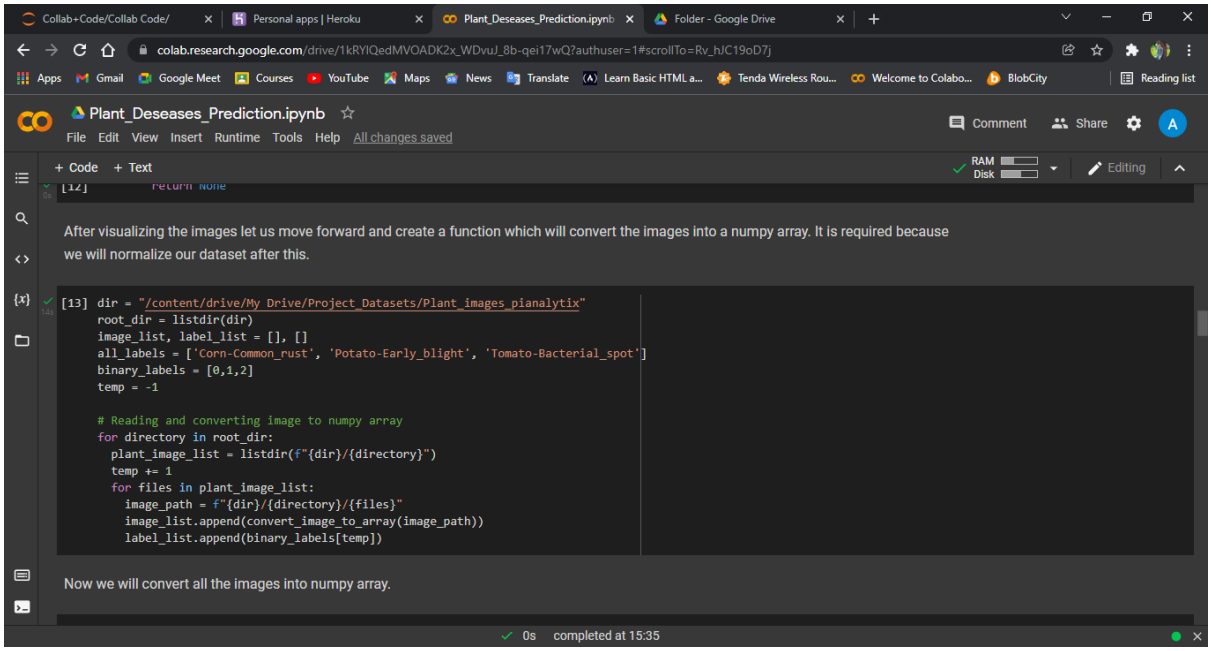


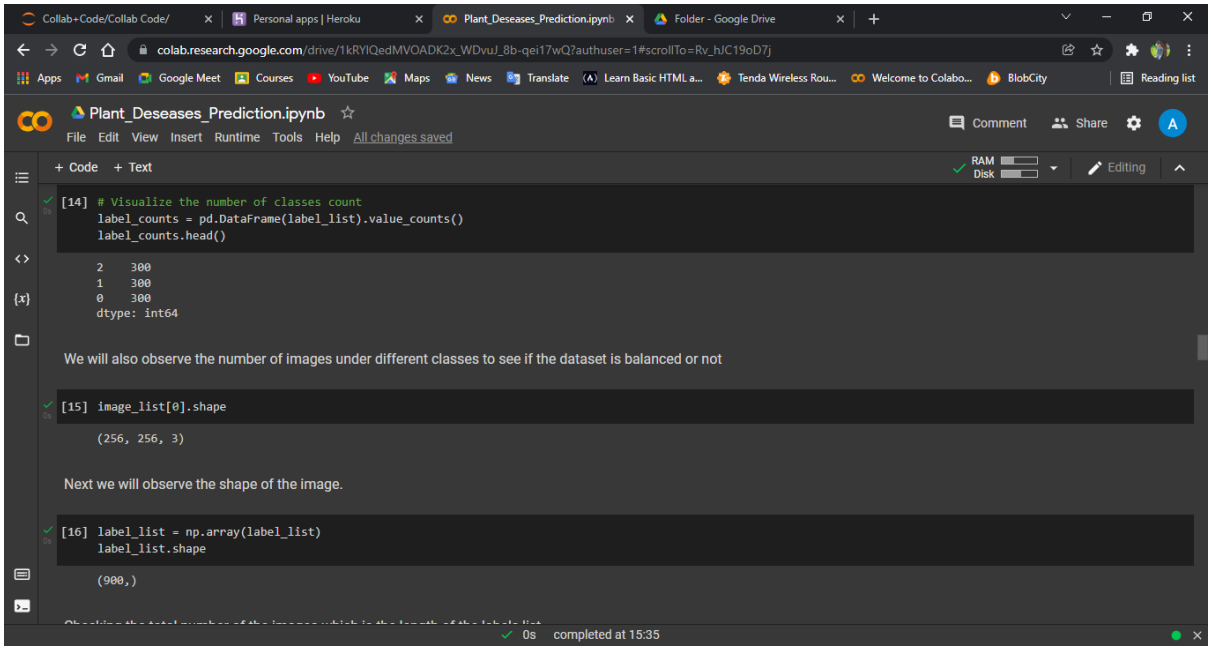


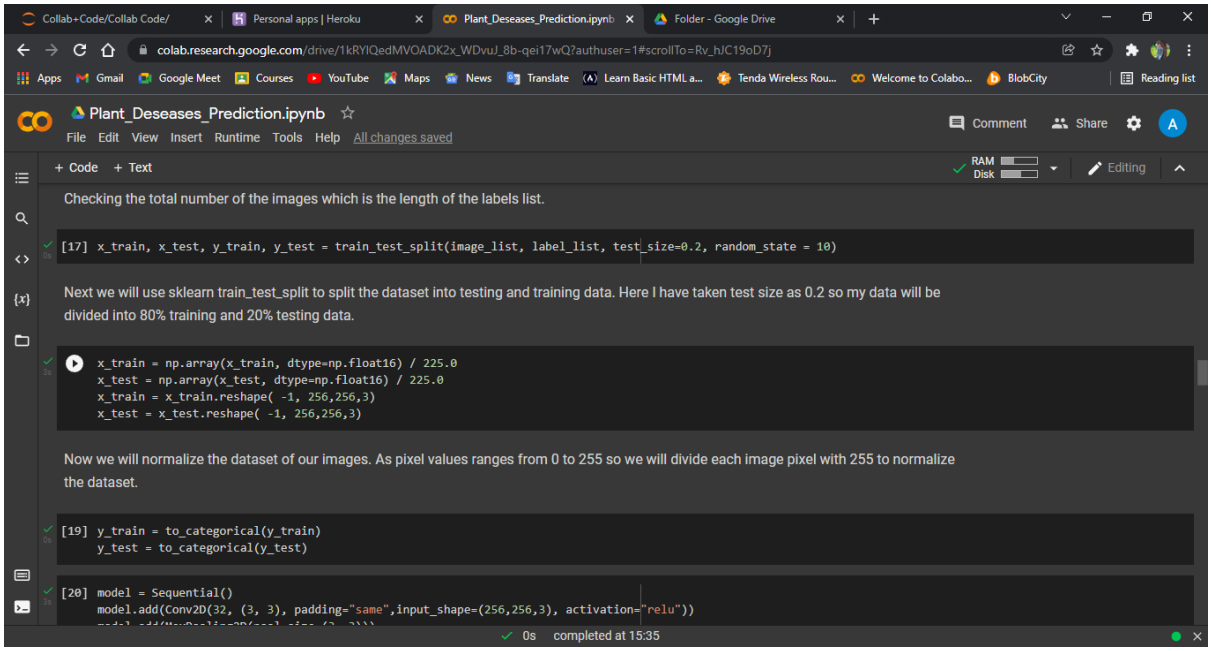

Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[10] from tensorflow.keras.utils import to_categorical
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages (2.7.0)
Next we will import all the required libraries. As we are making a CNN model we will import all the required layers, activations, optimizers, etc.
[11] # Plotting 12 images to check dataset
plt.figure(figsize=(12,12))
path = "/content/drive/My Drive/Project_Datasets/Plant_images_pianalytix/Potato___Early_blight"
for i in range(1,17):
plt.subplot(4,4,i)
plt.tight_layout()
rand_img = imread(path +'/'+ random.choice(sorted(os.listdir(path))))
plt.imshow(rand_img)
plt.xlabel(rand_img.shape[1], fontsize = 10)#width of image
plt.ylabel(rand_img.shape[0], fontsize = 10)#height of image
0s completed at 15:35

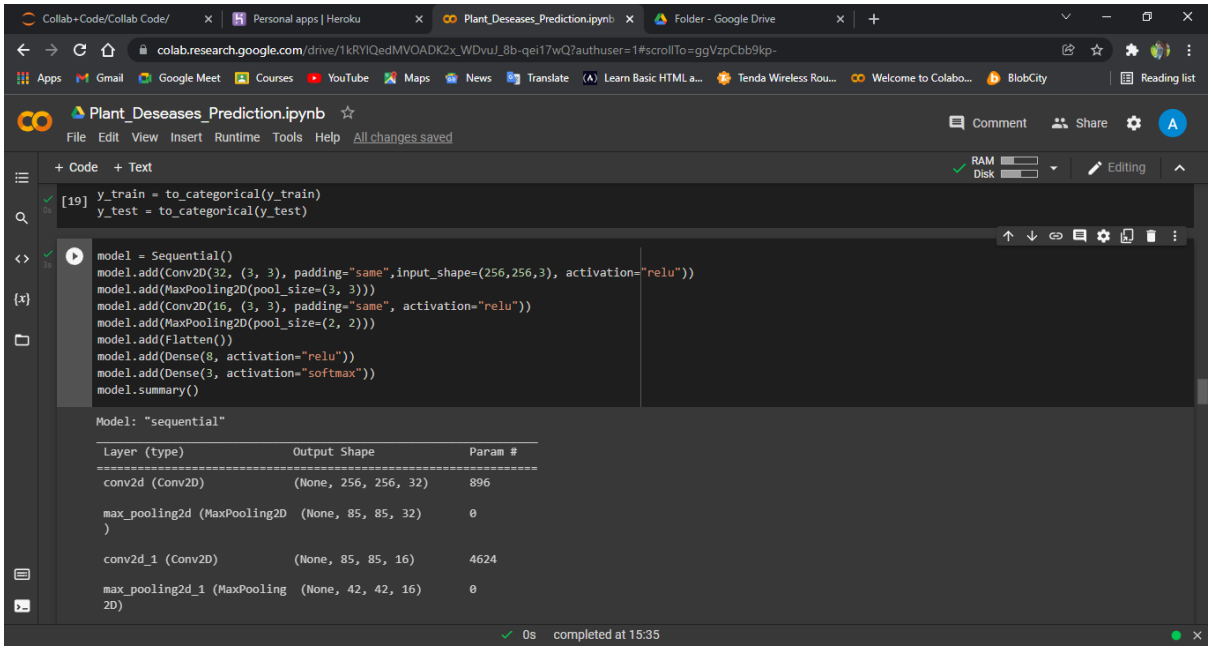

Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[11]
0s completed at 15:35

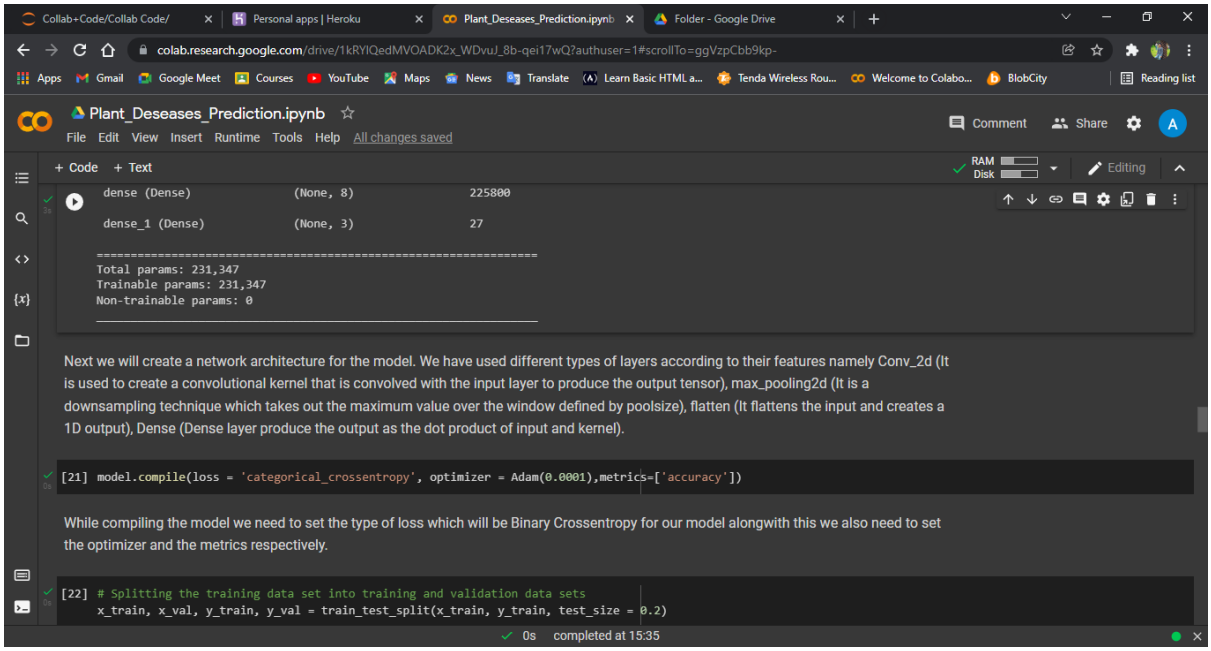

Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Now we will observe some of the iamges that are their in our dataset. We will plot 12 images here using the matplotlib library.
[12] #Converting Images to array
def convert_image_to_array(image_dir):
try:
image = cv2.imread(image_dir)
if image is not None :
image = cv2.resize(image, (256,256))
#image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
return img_to_array(image)
else :
return np.array([])
except Exception as e:
print(f"Error : {e}")
return None
After visualizing the images let us move forward and create a function which will convert the images into a numpy array. It is required because we will normalize our dataset after this.
0s completed at 15:35

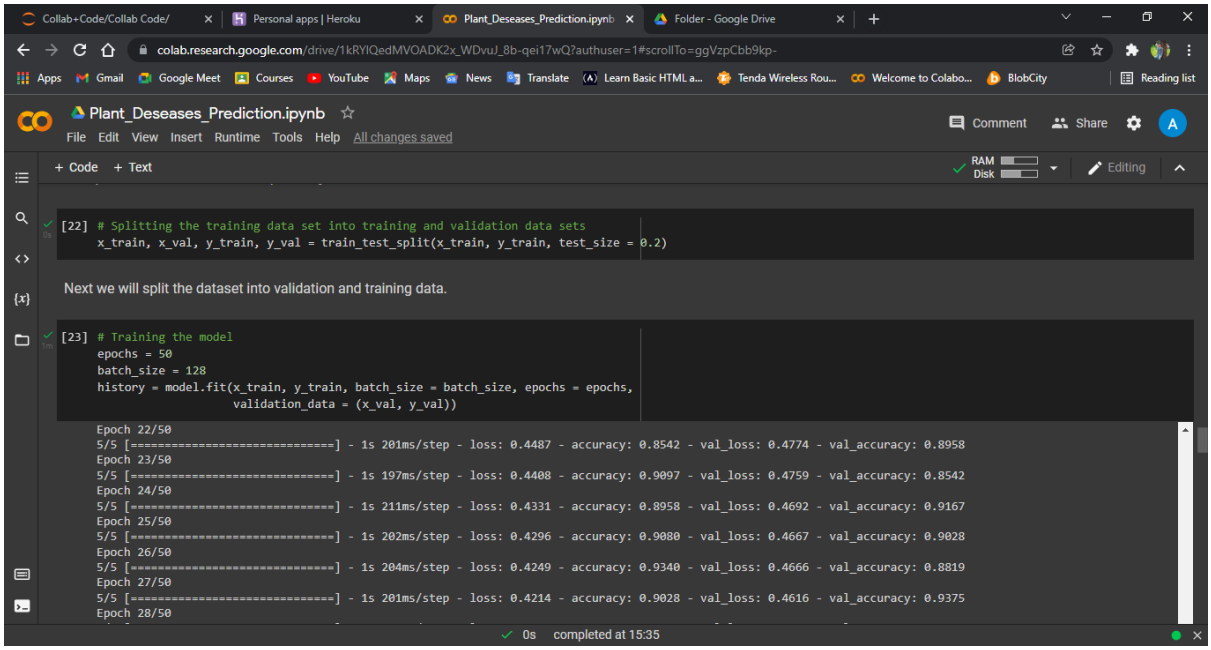

Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
After visualizing the images let us move forward and create a function which will convert the images into a numpy array. It is required because we will normalize our dataset after this.
[13] dir = "/content/drive/My Drive/Project_Datasets/Plant_images_pianalytix"
root_dir = listdir(dir)
image_list, label_list = [], []
all_labels = ['Corn-Common_rust', 'Potato-Early_blight', 'Tomato-Bacterial_spot']
binary_labels = [0,1,2]
temp = -1
# Reading and converting image to numpy array
for directory in root_dir:
plant_image_list = listdir(f"{dir}/{directory}")
temp += 1
for files in plant_image_list:
image_path = f"{dir}/{directory}/{files}"
image_list.append(convert_image_to_array(image_path))
label_list.append(binary_labels[temp])
Now we will convert all the images into numpy array.
0s completed at 15:35

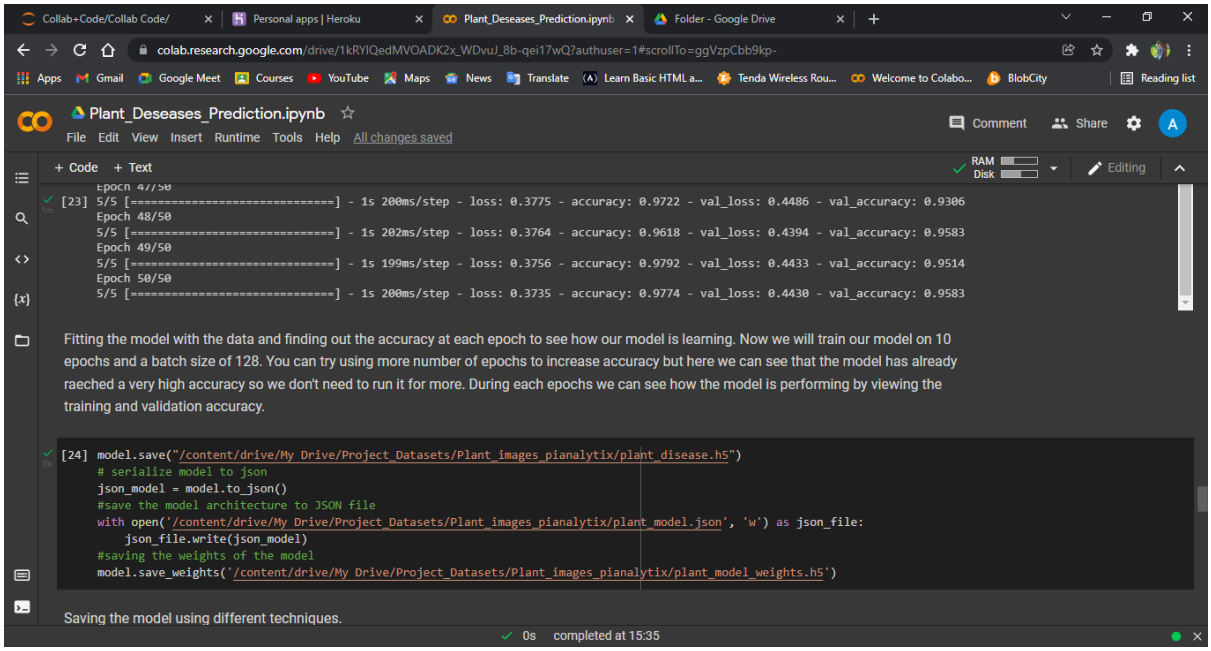

Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[14] # Visualize the number of classes count
label_counts = pd.DataFrame(label_list).value_counts()
label_counts.head()
2 300
1 300
0 300
dtype: int64
We will also observe the number of images under different classes to see if the dataset is balanced or not
[15] image_list[0].shape
(256, 256, 3)
Next we will observe the shape of the image.
[16] label_list = np.array(label_list)
label_list.shape
(900,)
0s completed at 15:35

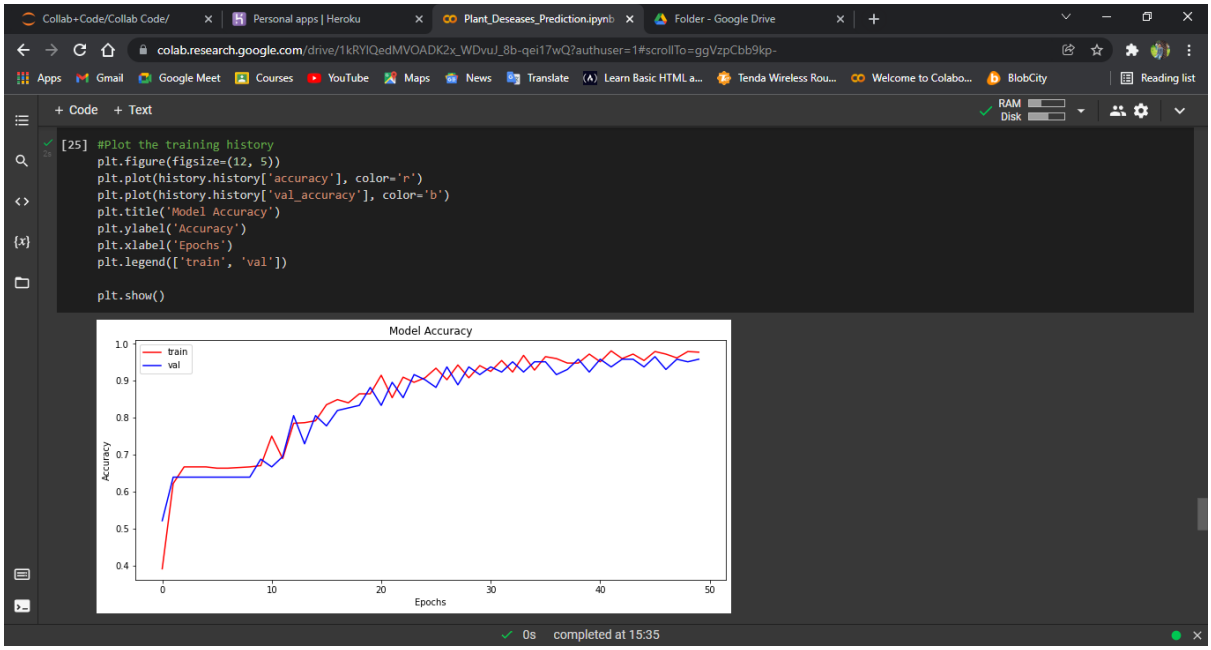

Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Checking the total number of the images which is the length of the labels list.
[17] x_train, x_test, y_train, y_test = train_test_split(image_list, label_list, test_size=0.2, random_state = 10)
Next we will use sklearn train_test_split to split the dataset into testing and training data. Here I have taken test size as 0.2 so my data will be divided into 80% training and 20% testing data.
x_train = np.array(x_train, dtype=np.float16) / 225.0
x_test = np.array(x_test, dtype=np.float16) / 225.0
x_train = x_train.reshape(-1, 256,256,3)
x_test = x_test.reshape(-1, 256,256,3)
Now we will normalize the dataset of our images. As pixel values ranges from 0 to 255 so we will divide each image pixel with 255 to normalize the dataset.
[19] y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
[20] model = Sequential()
model.add(Conv2D(32, (3, 3), padding="same",input_shape=(256,256,3), activation="relu"))
0s completed at 15:35

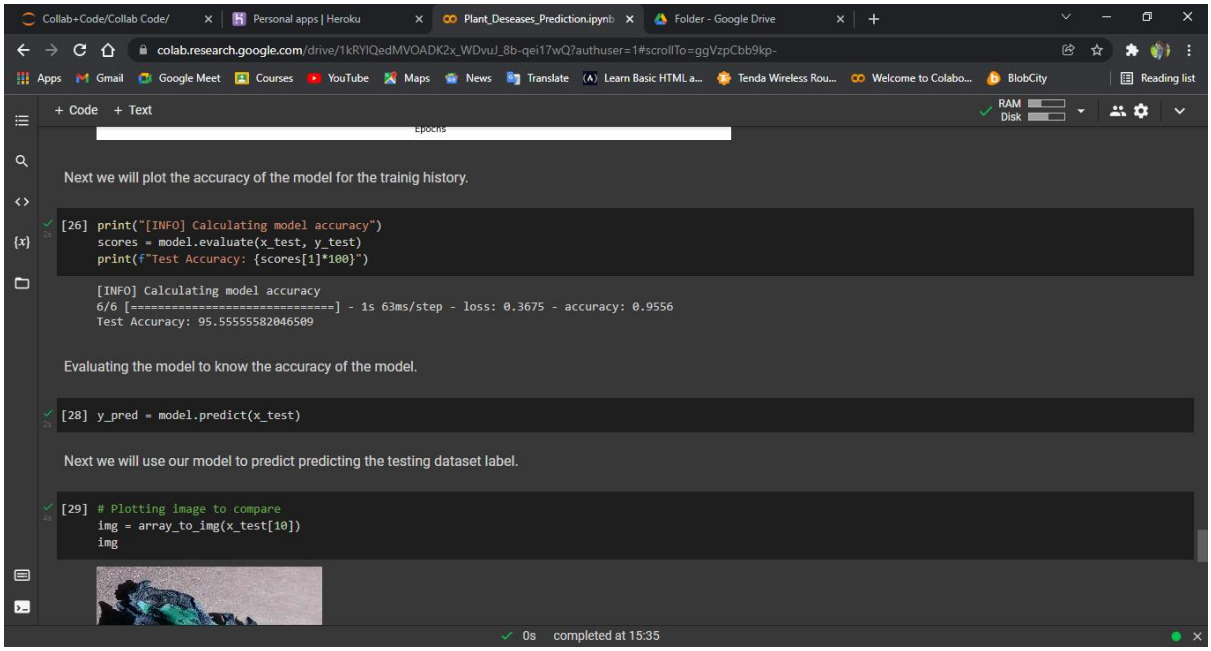

Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[19] y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
model = Sequential()
model.add(Conv2D(32, (3, 3), padding="same",input_shape=(256,256,3), activation="relu"))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(8, activation="relu"))
model.add(Dense(3, activation="softmax"))
model.summary()
Model: "sequential"
Layer (type) Output Shape Param #
conv2d (Conv2D) (None, 256, 256, 32) 896
max_pooling2d (MaxPooling2D (None, 85, 85, 32) 0
conv2d_1 (Conv2D) (None, 85, 85, 16) 4624
max_pooling2d_1 (MaxPooling (None, 42, 42, 16) 0
2D)
0s completed at 15:35

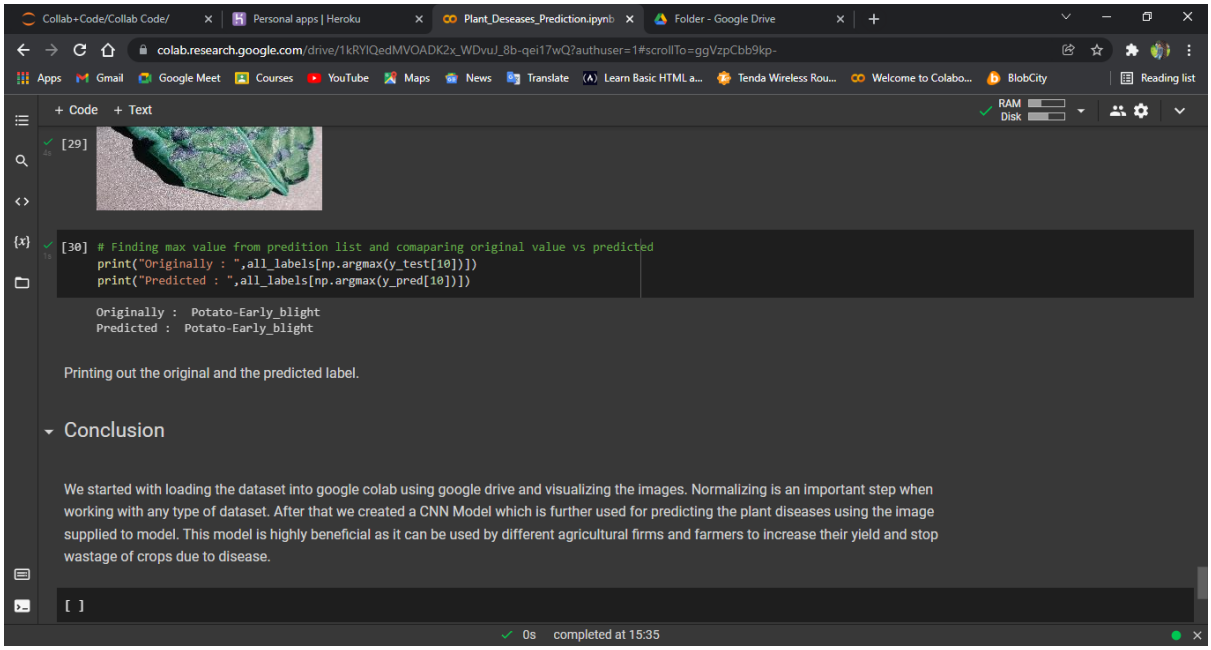

Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
dense (Dense) (None, 8) 225800
dense_1 (Dense) (None, 3) 27
Total params: 231,347
Trainable params: 231,347
Non-trainable params: 0
Next we will create a network architecture for the model. We have used different types of layers according to their features namely Conv_2d (It is used to create a convolutional kernel that is convolved with the input layer to produce the output tensor), max_pooling2d (It is a downsampling technique which takes out the maximum value over the window defined by poolsize), flatten (It flattens the input and creates a 1D output), Dense (Dense layer produce the output as the dot product of input and kernel).
[21] model.compile(loss = 'categorical_crossentropy', optimizer = Adam(0.0001),metrics=['accuracy'])
While compiling the model we need to set the type of loss which will be Binary Crossentropy for our model alongwith this we also need to set the optimizer and the metrics respectively.
[22] # Splitting the training data set into training and validation data sets
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.2)
0s completed at 15:35


Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[22] # Splitting the training data set into training and validation data sets
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.2)
Next we will split the dataset into validation and training data.
[23] # Training the model
epochs = 50
batch_size = 128
history = model.fit(x_train, y_train, batch_size = batch_size, epochs = epochs,
validation_data = (x_val, y_val))
Epoch 22/50
5/5 [==========] - 1s 201ms/step - loss: 0.4487 - accuracy: 0.8542 - val_loss: 0.4774 - val_accuracy: 0.8958
Epoch 23/50
5/5 [==========] - 1s 197ms/step - loss: 0.4408 - accuracy: 0.9097 - val_loss: 0.4759 - val_accuracy: 0.8542
Epoch 24/50
5/5 [==========] - 1s 211ms/step - loss: 0.4331 - accuracy: 0.8958 - val_loss: 0.4692 - val_accuracy: 0.9167
Epoch 25/50
5/5 [==========] - 1s 202ms/step - loss: 0.4296 - accuracy: 0.9080 - val_loss: 0.4667 - val_accuracy: 0.9028
Epoch 26/50
5/5 [==========] - 1s 204ms/step - loss: 0.4249 - accuracy: 0.9340 - val_loss: 0.4666 - val_accuracy: 0.8819
Epoch 27/50
5/5 [==========] - 1s 201ms/step - loss: 0.4214 - accuracy: 0.9028 - val_loss: 0.4616 - val_accuracy: 0.9375
Epoch 28/50
0s completed at 15:35


Plant_Deseases_Prediction.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[23] 5/5 [==========] - 1s 200ms/step - loss: 0.3775 - accuracy: 0.9722 - val_loss: 0.4486 - val_accuracy: 0.9306
Epoch 48/50
5/5 [==========] - 1s 202ms/step - loss: 0.3764 - accuracy: 0.9618 - val_loss: 0.4394 - val_accuracy: 0.9583
Epoch 49/50
5/5 [==========] - 1s 199ms/step - loss: 0.3756 - accuracy: 0.9792 - val_loss: 0.4433 - val_accuracy: 0.9514
Epoch 50/50
5/5 [==========] - 1s 200ms/step - loss: 0.3735 - accuracy: 0.9774 - val_loss: 0.4430 - val_accuracy: 0.9583
Fitting the model with the data and finding out the accuracy at each epoch to see how our model is learning. Now we will train our model on 10 epochs and a batch size of 128. You can try using more number of epochs to increase accuracy but here we can see that the model has already raeched a very high accuracy so we don't need to run it for more. During each epochs we can see how the model is performing by viewing the training and validation accuracy.
[24] model.save("/content/drive/My Drive/Project_Datasets/Plant_images_pianalytix/plant_disease.h5")
# serialize model to json
json_model = model.to_json()
#save the model architecture to JSON file
with open('/content/drive/My Drive/Project_Datasets/Plant_images_pianalytix/plant_model.json', 'w') as json_file:
json_file.write(json_model)
#saving the weights of the model
model.save_weights('/content/drive/My Drive/Project_Datasets/Plant_images_pianalytix/plant_model_weights.h5')
Saving the model using different techniques.
0s completed at 15:35

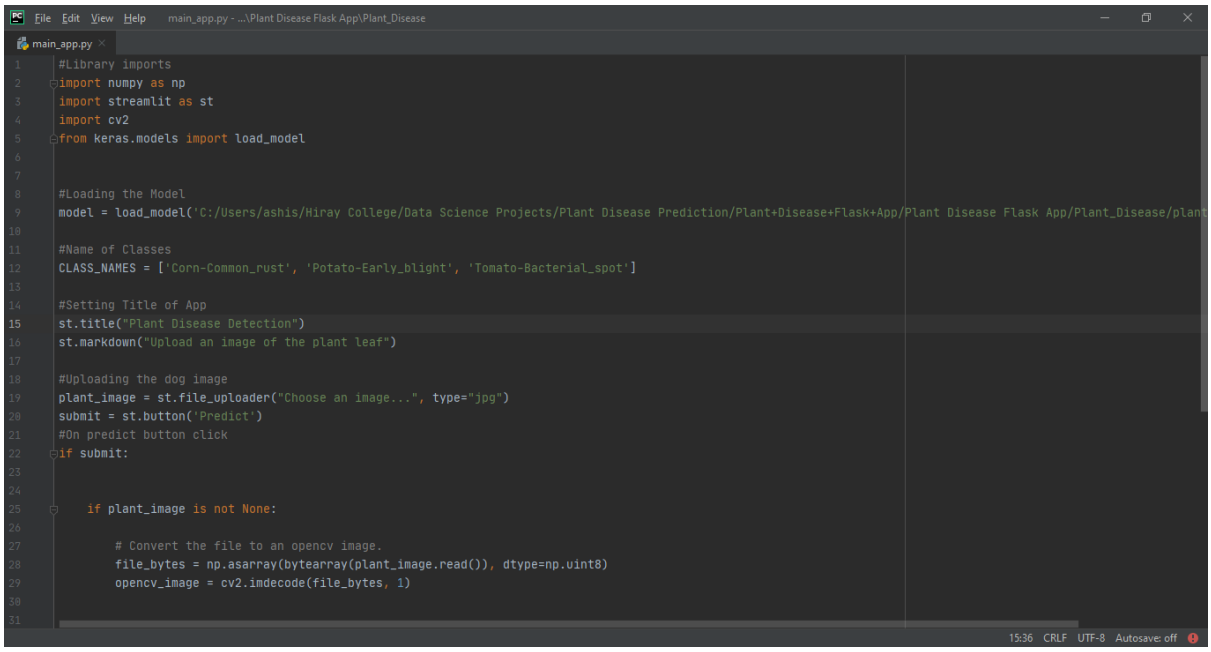

Collab+Code/Collab Code/
Personal apps | Heroku
Plant_Deseases_Prediction.ipynb
Folder - Google Drive
colab.research.google.com/drive/1kRYIQedMVOADK2x_WDvuJ_8b-qei17wQ?authuser=1#scrollTo=ggVzpCbb9kp-
+ Code + Text
RAM
Disk
[25] #Plot the training history
plt.figure(figsize=(12, 5))
plt.plot(history.history['accuracy'], color='r')
plt.plot(history.history['val_accuracy'], color='b')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'val'])
plt.show()
Model Accuracy
train
val
Accuracy
Epochs
0s completed at 15:35


Collab+Code/Collab Code/
Personal apps | Heroku
Plant_Deseases_Prediction.ipynb
Folder - Google Drive
colab.research.google.com/drive/1kRYIQedMVOADK2x_WDvuJ_8b-qei17wQ?authuser=1#scrollTo=ggVzpCbb9kp-
+ Code + Text
RAM
Disk
Epochs
Next we will plot the accuracy of the model for the trainig history.
[26] print("[INFO] Calculating model accuracy")
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
[INFO] Calculating model accuracy
6/6 [==============================] - 1s 63ms/step - loss: 0.3675 - accuracy: 0.9556
Test Accuracy: 95.55555582046509
Evaluating the model to know the accuracy of the model.
[28] y_pred = model.predict(x_test)
Next we will use our model to predict predicting the testing dataset label.
[29] # Plotting image to compare
img = array_to_img(x_test[10])
img
0s completed at 15:35

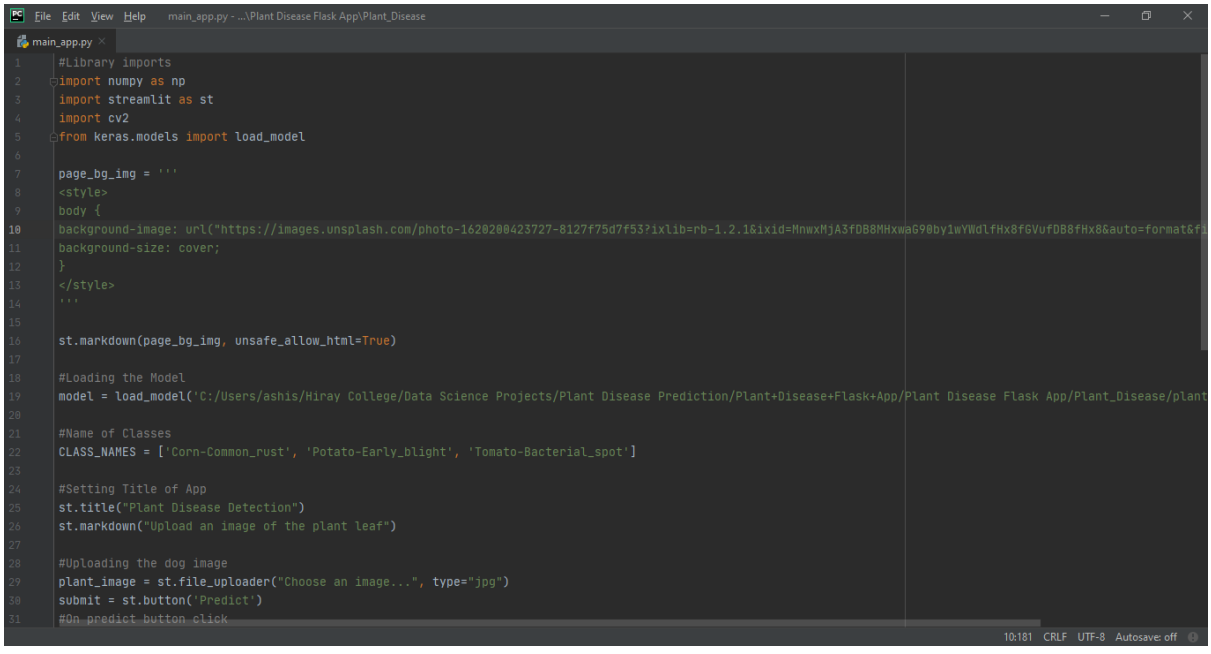

Collab+Code/Collab Code/
Personal apps | Heroku
Plant_Deseases_Prediction.ipynb
Folder - Google Drive
colab.research.google.com/drive/1kRYIQedMVOADK2x_WDvuJ_8b-qei17wQ?authuser=1#scrollTo=ggVzpCbb9kp-
Apps
Gmail
Google Meet
Courses
YouTube
Maps
News
Translate
Learn Basic HTML a...
Tenda Wireless Rou...
Welcome to Colabo...
BlobCity
Reading list
+ Code
+ Text
RAM
Disk
[29]
[30] # Finding max value from predition list and comaparing original value vs predicted
print("Originally : ",all_labels[np.argmax(y_test[10])])
print("Predicted : ",all_labels[np.argmax(y_pred[10])])
Originally : Potato-Early_blight
Predicted : Potato-Early_blight
Printing out the original and the predicted label.
Conclusion
We started with loading the dataset into google colab using google drive and visualizing the images. Normalizing is an important step when working with any type of dataset. After that we created a CNN Model which is further used for predicting the plant diseases using the image supplied to model. This model is highly beneficial as it can be used by different agricultural firms and farmers to increase their yield and stop wastage of crops due to disease.
[ ]
0s completed at 15:35

PyCharm Screenshot:-

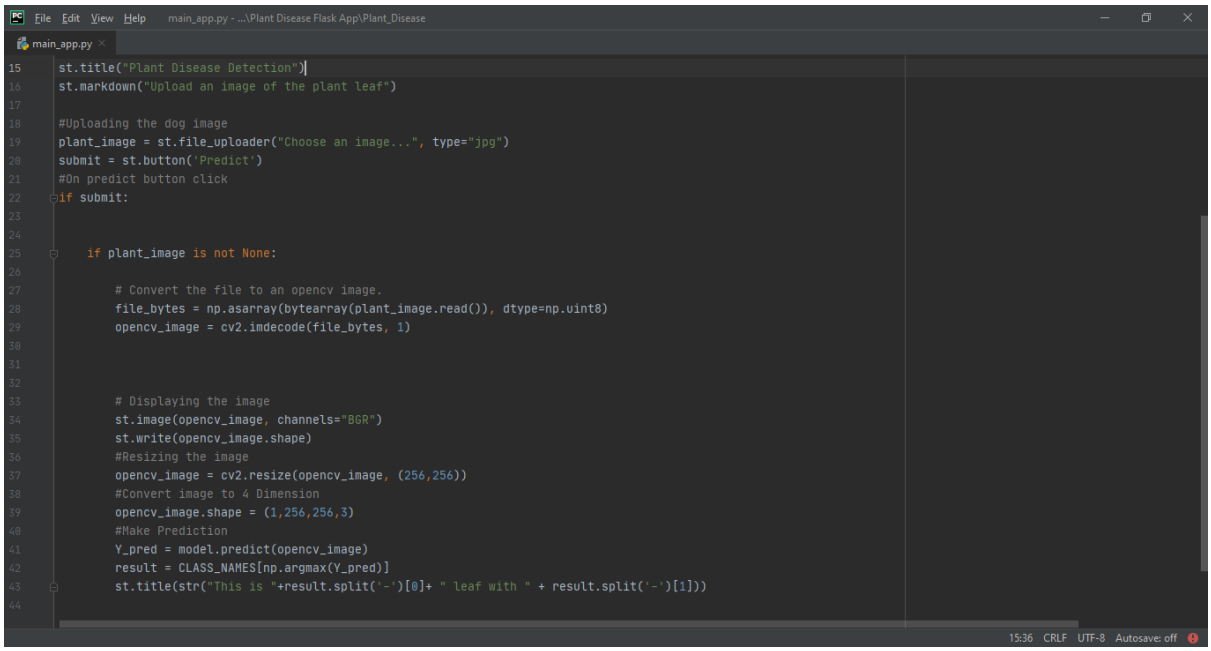
StreamlitServer:-



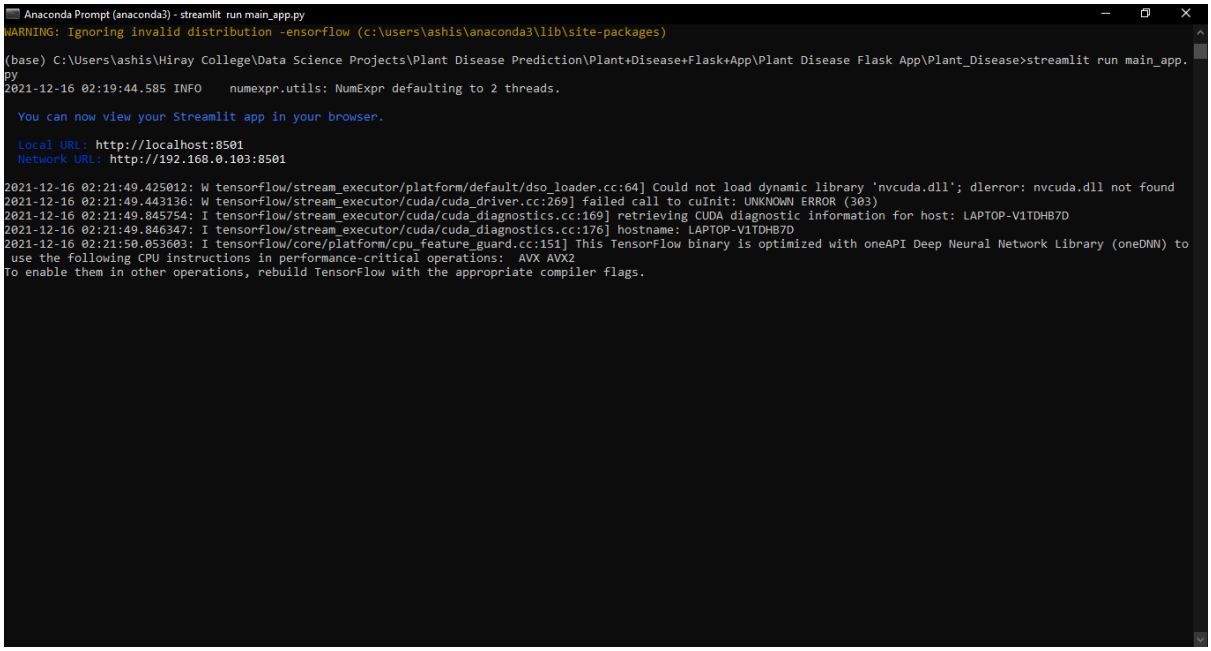
```
#Library imports
import numpy as np
import streamlit as st
import cv2
from keras.models import load_model


#Loading the Model
model = load_model('C:/Users/ashis/Hiray College/Data Science Projects/Plant Disease Prediction/Plant+Disease+Flask+App/Plant Disease Flask App/Plant_Disease/plant

#Name of Classes
CLASS_NAMES = ['Corn-Common_rust', 'Potato-Early_blight', 'Tomato-Bacterial_spot']

#Setting Title of App
st.title("Plant Disease Detection")
st.markdown("Upload an image of the plant leaf")

#Uploading the dog image
plant_image = st.file_uploader("Choose an image...", type="jpg")
submit = st.button('Predict')
#On predict button click
if submit:


    if plant_image is not None:

        # Convert the file to an opencv image.
        file_bytes = np.asarray(bytearray(plant_image.read()), dtype=np.uint8)
        opencv_image = cv2.imdecode(file_bytes, 1)
```



```
#Library imports
import numpy as np
import streamlit as st
import cv2
from keras.models import load_model

page_bg_img = '''
<style>
body {
background-image: url("https://images.unsplash.com/photo-1620200423727-8127f75d7f53?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fi
background-size: cover;
}
</style>
'''

st.markdown(page_bg_img, unsafe_allow_html=True)

#Loading the Model
model = load_model('C:/Users/ashis/Hiray College/Data Science Projects/Plant Disease Prediction/Plant+Disease+Flask+App/Plant Disease Flask App/Plant_Disease/plant

#Name of Classes
CLASS_NAMES = ['Corn-Common_rust', 'Potato-Early_blight', 'Tomato-Bacterial_spot']

#Setting Title of App
st.title("Plant Disease Detection")
st.markdown("Upload an image of the plant leaf")

#Uploading the dog image
plant_image = st.file_uploader("Choose an image...", type="jpg")
submit = st.button('Predict')
#On predict button click
```


File Edit View Help
main_app.py - ...\Plant Disease Flask App\Plant_Disease
main_app.py
st.title("Plant Disease Detection")
st.markdown("Upload an image of the plant leaf")
#Uploading the dog image
plant_image = st.file_uploader("Choose an image...", type="jpg")
submit = st.button('Predict')
#On predict button click
if submit:
if plant_image is not None:
# Convert the file to an opencv image.
file_bytes = np.asarray(bytearray(plant_image.read()), dtype=np.uint8)
opencv_image = cv2.imdecode(file_bytes, 1)
# Displaying the image
st.image(opencv_image, channels="BGR")
st.write(opencv_image.shape)
#Resizing the image
opencv_image = cv2.resize(opencv_image, (256,256))
#Convert image to 4 Dimension
opencv_image.shape = (1,256,256,3)
#Make Prediction
Y_pred = model.predict(opencv_image)
result = CLASS_NAMES[np.argmax(Y_pred)]
st.title(str("This is "+result.split('-')[0]+ " leaf with " + result.split('-')[1]))
15:36 CRLF UTF-8 Autosave: off

Anaconda Command Prompt to run streamlit:-

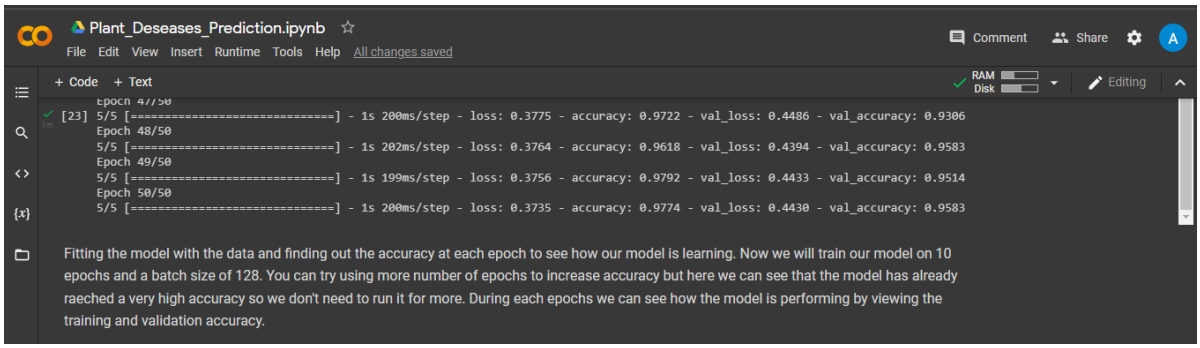


RESULTS AND DISCUSSIONS

Best Suited Model

So, our study showed that……..

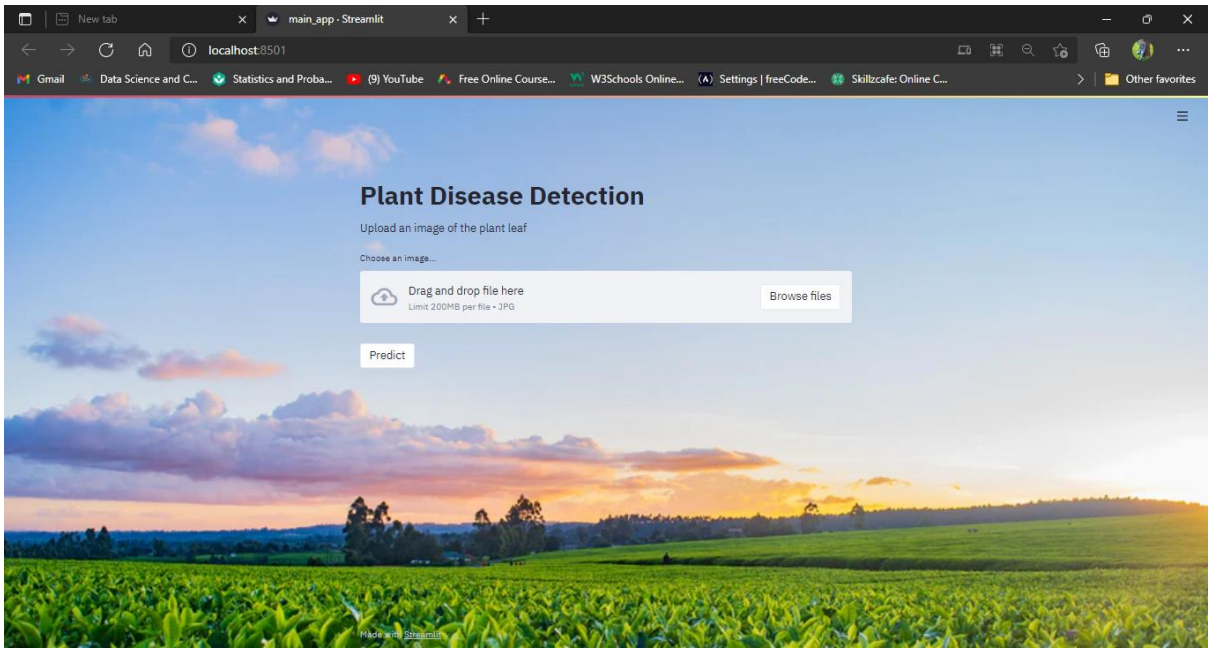
Convolutional Neural Networks displayed the best performance for this Dataset and can be used for deploying purposes.

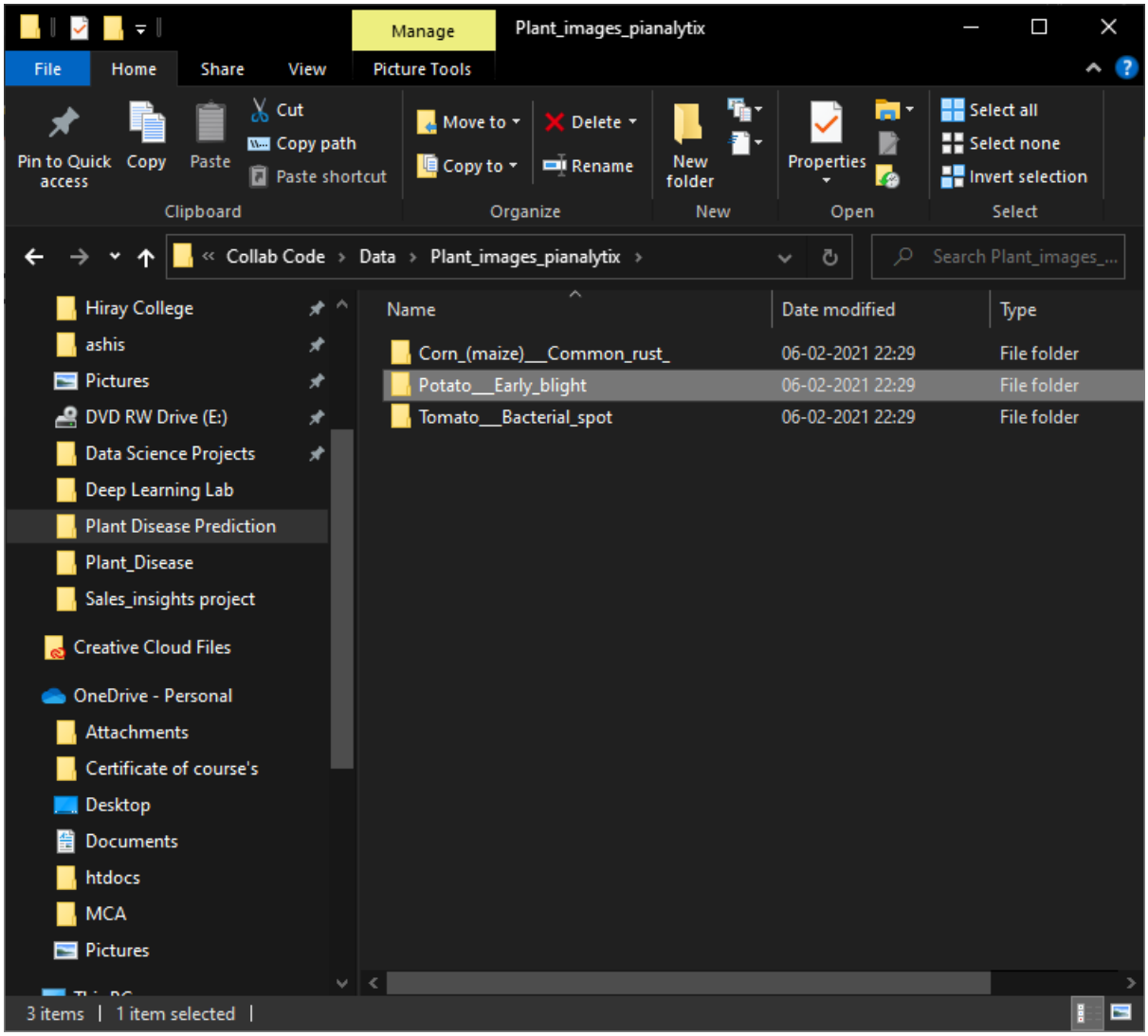


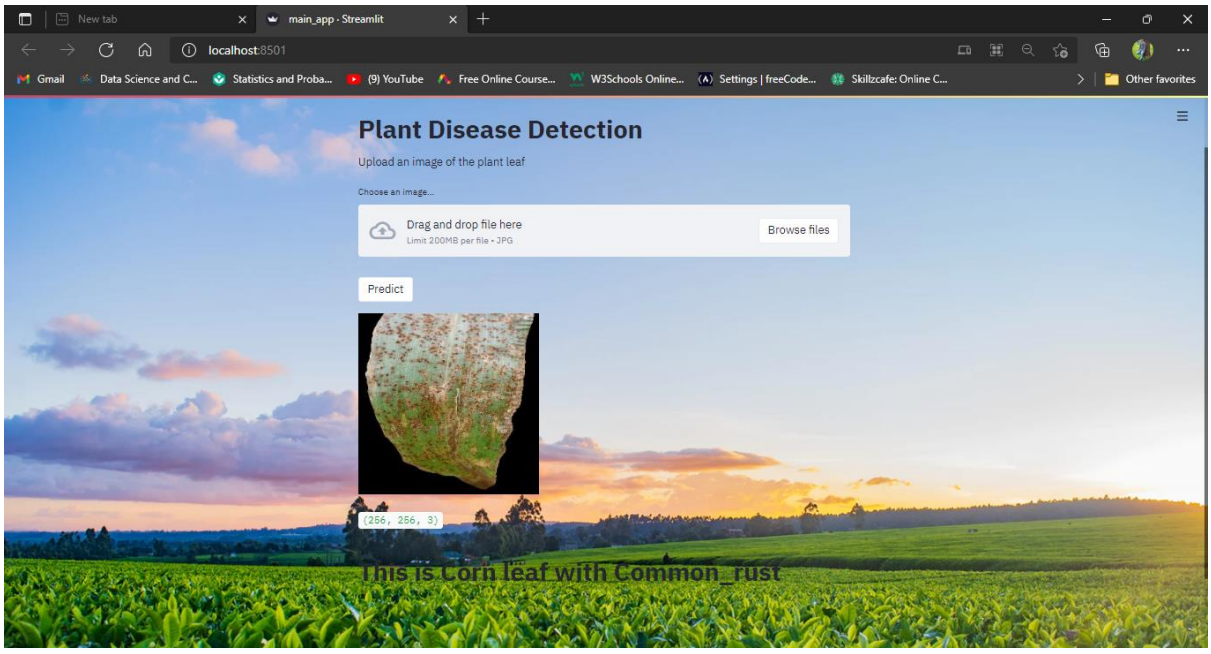
Deployment App

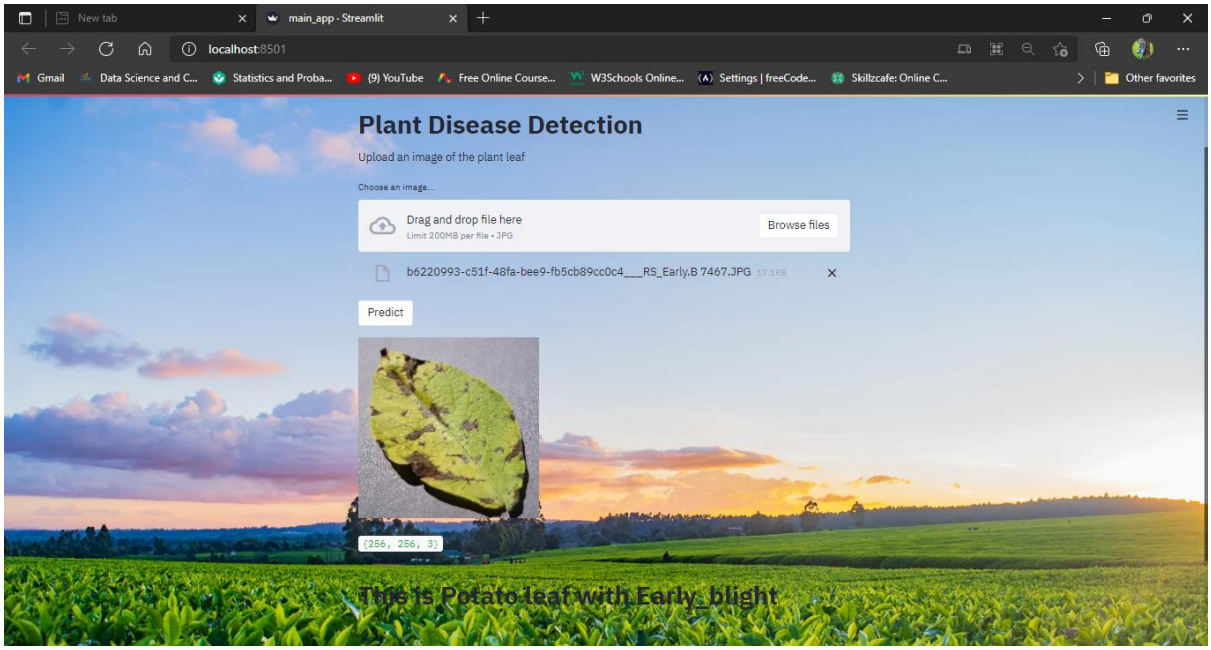
The Model is deployed through Python Web App Streamlit

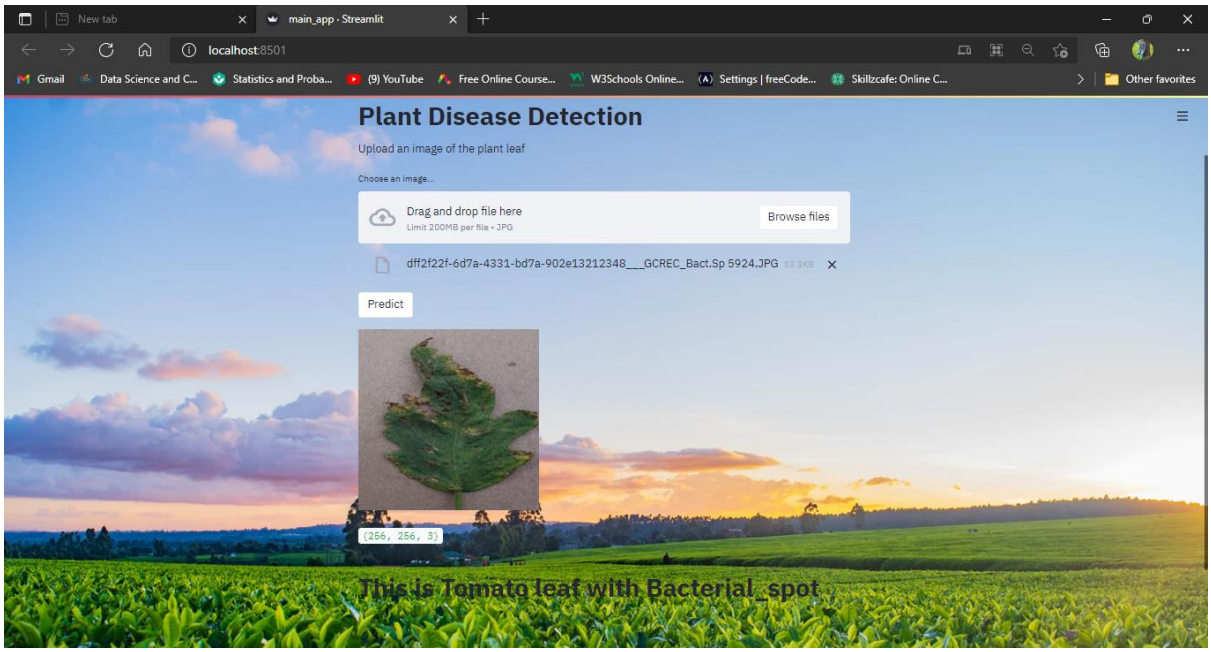
Project Output ScreenShot:-




Manage
Plant_images_pianalytix
File
Home
Share
View
Picture Tools
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Delete
Copy to
Rename
Organize
New folder
New
Properties
Open
Select all
Select none
Invert selection
Select
Collab Code › Data › Plant_images_pianalytix ›
Search Plant_images_...
Hiray College
ashis
Pictures
DVD RW Drive (E:)
Data Science Projects
Deep Learning Lab
Plant Disease Prediction
Plant_Disease
Sales_insights project
Creative Cloud Files
OneDrive - Personal
Attachments
Certificate of course's
Desktop
Documents
htdocs
MCA
Pictures
Name
Date modified
Type
Corn_(maize)___Common_rust_ 06-02-2021 22:29 File folder
Potato___Early_blight 06-02-2021 22:29 File folder
Tomato___Bacterial_spot 06-02-2021 22:29 File folder
3 items | 1 item selected |


New tab
main_app · Streamlit
localhost:8501
Gmail
Data Science and C...
Statistics and Proba...
(9) YouTube
Free Online Course...
W3Schools Online...
Settings | freeCode...
Skillzcafe: Online C...
Other favorites
Plant Disease Detection
Upload an image of the plant leaf
Choose an image...
Drag and drop file here
Limit 200MB per file • JPG
Browse files
Predict
(256, 256, 3)
This is Corn leaf with Common_rust


main_app · Streamlit
localhost:8501
Plant Disease Detection
Upload an image of the plant leaf
Drag and drop file here
Browse files
b6220993-c51f-48fa-bee9-fb5cb89cc0c4___RS_Early.B 7467.JPG
Predict
(256, 256, 3)
This is Potato leaf with Early_blight


main_app · Streamlit
localhost:8501
Plant Disease Detection
Upload an image of the plant leaf
Drag and drop file here
Browse files
dff2f22f-6d7a-4331-bd7a-902e13212348___GCREC_Bact.Sp 5924.JPG
Predict
(256, 256, 3)
This is Tomato leaf with Bacterial_spot

==========

CONCLUSION

We started with loading the dataset into google colab using google drive and visualizing the images. Normalizing is an important step when working with any type of dataset. After that we created a CNN Model which is further used for predicting the plant diseases using the image supplied to model. This model is highly beneficial as it can be used by different agricultural firms and farmers to increase their yield and stop wastage of crops due to disease.

- Model for our Dataset with BEST ACCURACY of 0.95.

❖ Bibliography

- ✔ Find Open Datasets and Machine Learning Projects | Kaggle
- ✔ Towards Data Science
- ✔ RxJS, ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.js, Weka, Solidity (tutorialspoint.com)
- ✔ Learn R, Python & Data Science Online | DataCamp
- ✔ YouTube
- ✔ https://www.khanacademy.org
- ✔ Analyticsvidhya.com
- ✔ Machinelearningplus.com
- ✔ Simplilearn.com
- ✔ www.tutorialpoint.com

==========