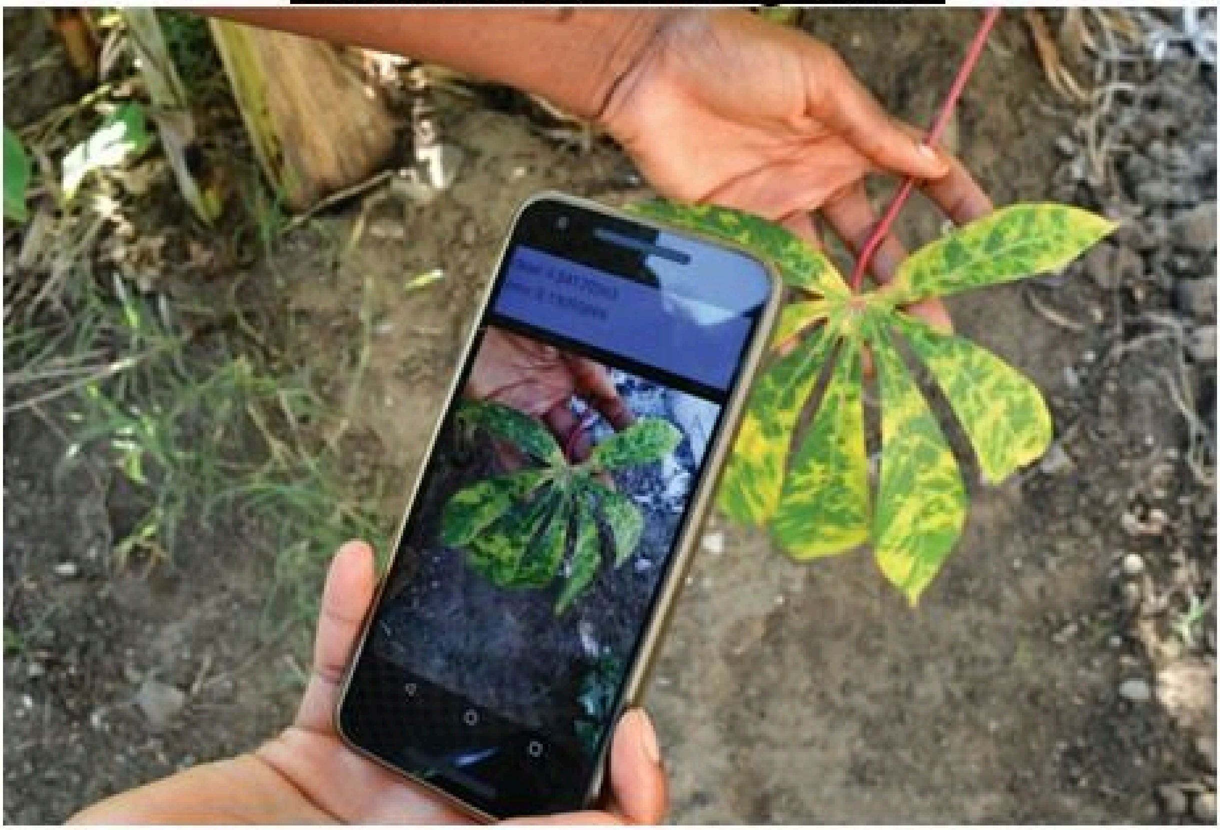
INNOMATICS
RESEARCH LABS

Internship Project Report -- 2021

Project Title:

Plant Disease Recognition



Domain: Agriculture

Duration Of Project: 31-12-2021 TO 25-01-2022

Under the Guidance of:

Kanav Bansal Sir (Chief Data Scientist) And

Nagamalleshwara Rao Sir (Mentor)

Presented By: INTERNS – BATCH 2021 (OCT-2021 to JAN-2022)

1. **Manuj Kumar Joshi**

Intern at Innomatics Research Lab

2. **Kashetti Prashanth**

Intern at Innomatics Research Lab

3. **Soujanya Vattikolla**

Intern at Innomatics Research Lab

4. **Priyanka Nandibhatla**

Intern at Innomatics Research Lab

5. **Pooja Roy Choudhary**

Intern at Innomatics Research Lab

Plant Disease classification:

COMPUTER VISION

"Computer Vision" - the ultimate goal is to use computers to emulate human vision, including learning and being able to make inferences and take actions based on visual inputs. It is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images.

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

Attribute Information:

This project is about collecting images of various infected, good and seemingly infected plant leaves. Then apply image processing on the images and predict the infected plant leaf's using Deep Learning + Image Processing.

OBJECTIVE:

The main purpose is to detect the diseased part of the plant leaf. Using python convolutional neural networks are implemented in order to classify the diseased part. Aim is to detect the diseased part by finding the optimum way with minimum cost.

Goal🎯

The goal of the project is we need to build a model, which can classify between healthy and diseased crop leaves and also if the crop has any disease, predict which disease is it.

Project Domain:

Open CV (Computer vision) and CNN and Transfer Learning.

Steps Involved in Image Processing: -

1. Image Acquisition
2. Image Enhancement
3. Image Restoration
4. Color Image Processing
5. Compression
6. Segmentation
7. Mobile Application

Libraries Involved:

- TensorFlow
- Keras
- Scikit Learn
- Pickle
- OpenCV

Abstract:

- Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product.

- Plant leaf diseases and destructive insects are a major challenge in the agriculture sector.
- It is very difficult to monitor the plant diseases manually. It requires tremendous amount of work, expertise in the plant diseases, and also require the excessive processing time.

- Faster and an accurate prediction of leaf diseases in crops could early treatment technique while considerably reducing economic losses.

- Hence, image processing is used for the detection of plant diseases.

Insights From Data:

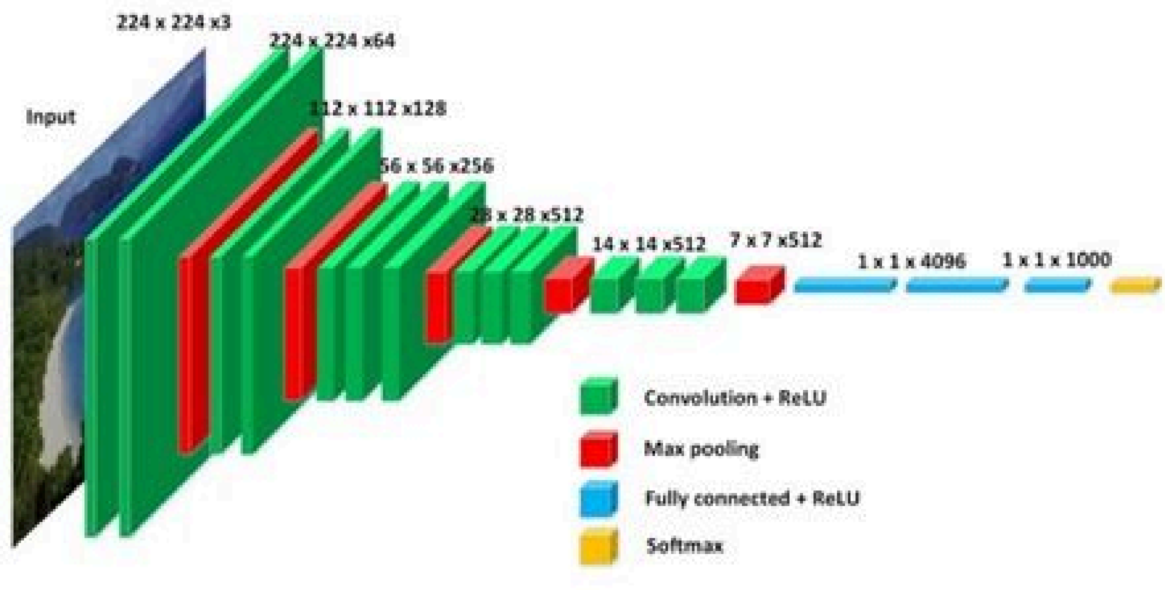
The plant leaf disease dataset consists of 87,900 images of leaves spanning 38 classes. Each class denotes a combination of the plant the leaf is from and the disease present in the leaf.

The dataset is divided into three parts as follows:

- **train** - 70,295 images divided into 38 classes.
- **valid** - 17,572 images divided into 38 classes.
- **test** - 33 images

MODEL USED – Vgg16

The architecture of Vgg16 uses 13 convolutional layers and 3 fully connected layers. The convolutional layers in Vgg16 are all 3*3 convolutional layers with a stride size of 1 and the same padding, and the pooling layers are all 2*2 pooling layers with a stride size of 2.



VGG-16 network architecture for feature extraction

MODEL ARCHITECTURE --- Vgg16

Input: Images

An image is nothing but a matrix of pixel values we flatten the image and feed it to a Multi-Level Perceptron for classification purposes.

Output:

Knowledge of the scene (recognize objects, people, activity happening there, distance of the object from camera and each other, …)

Introduction on VGGNet

The full name of VGG is the Visual Geometry Group, which belongs to the Department of Science and Engineering of Oxford University. It has released a series of convolutional network models beginning with VGG, which can be applied to face recognition and image classification, from VGG16 to VGG19. The original purpose of VGG's research on the depth of convolutional networks is to understand how the depth of convolutional networks affects the accuracy and accuracy of large-scale image classification and recognition. It is considered to be one of the excellent vision model architectures till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx.) parameters.

Project flow:

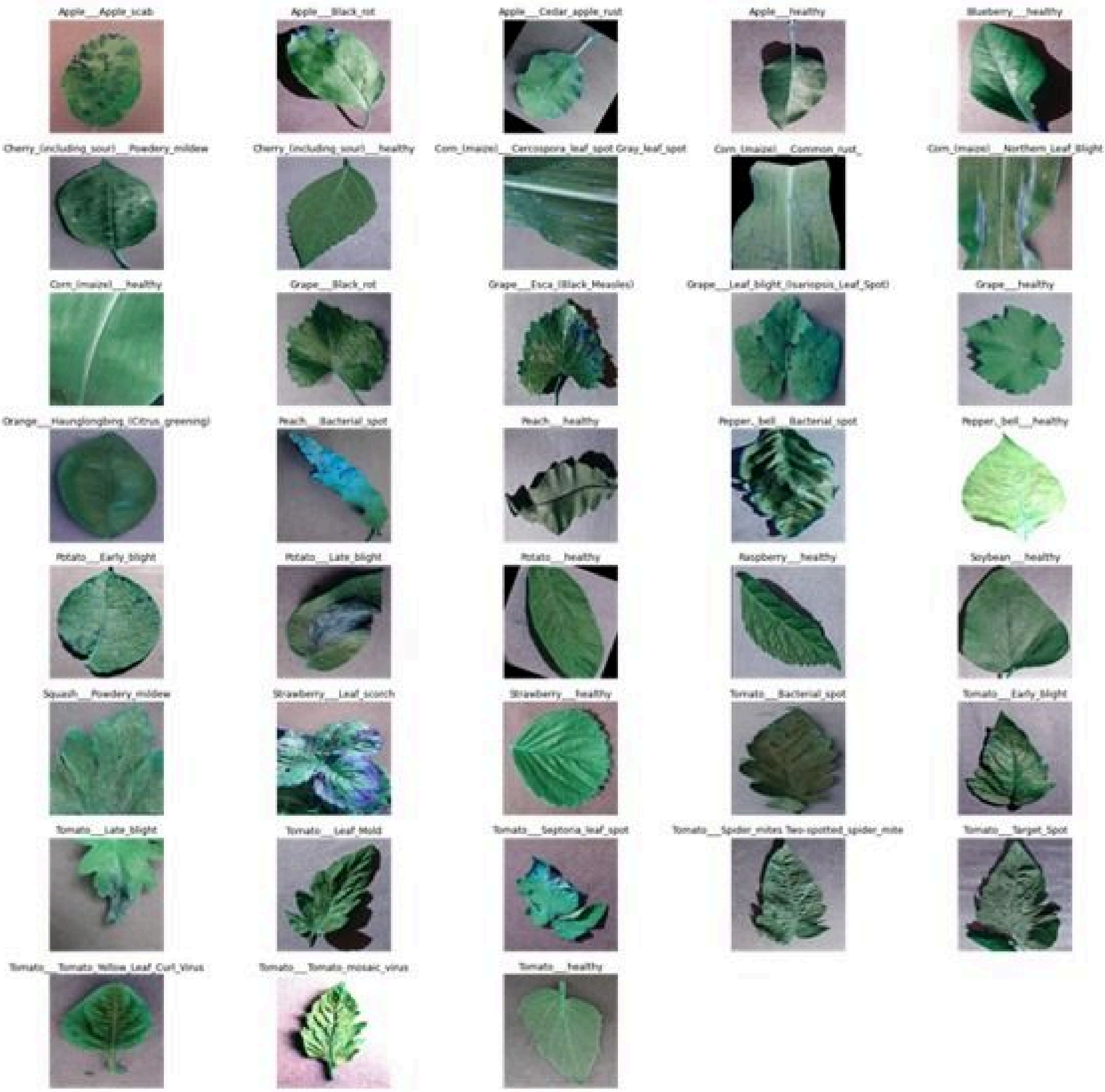
- Load and pre-process the image dataset.
  - Some leaves are taken as input and it is pre-processed for the improvement of image to remove the unwanted distortion for feature processing.
  - In the feature extraction the shape, color and texture are identified.
- Train the image classifier on the dataset.
  - Database consists of many images with healthy and unhealthy leaves and the images are trained to identify the particular image from the database.
- Use the trained classifier to predict image content.
  - In Neural Network classification the leaf is recognized whether the leaf is normal or abnormal.
  - If the leaf is abnormal the defect region is classified. The affected area and type of disease is identified.

Now here we start some review and visualizations from the CODEFILE:

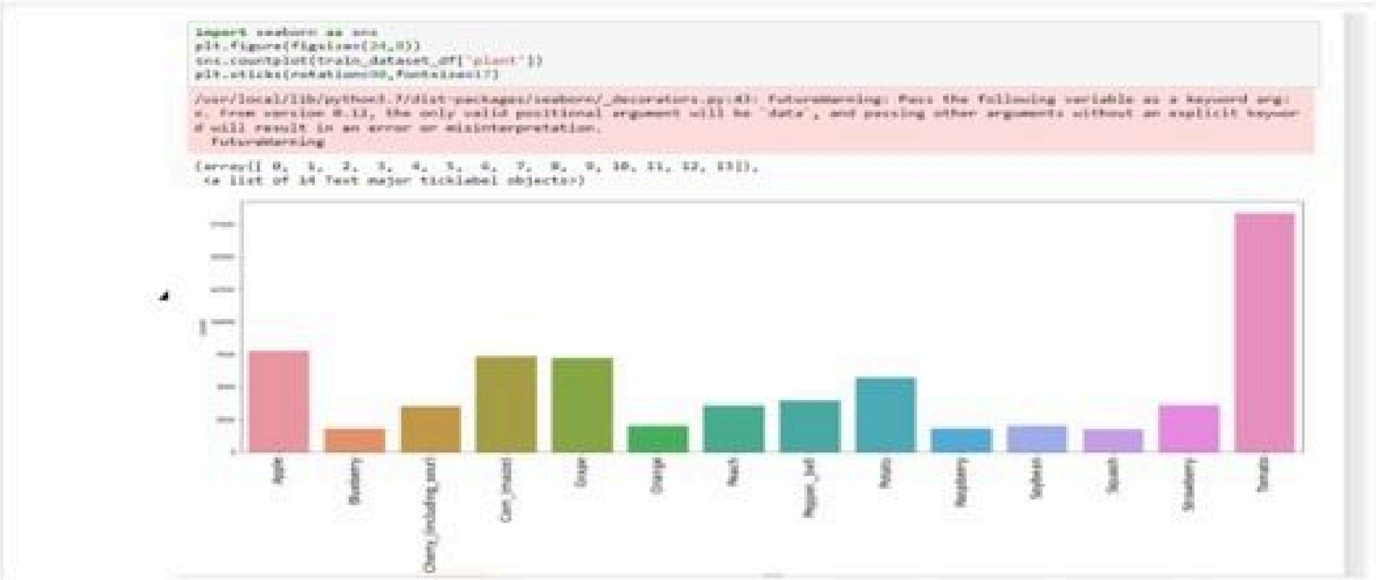
1. How does our data look like?
2. We note that the data is already augmented. This is relevant to have a predictive model that generalizes well: the predictions will not be dependent on the quality of the image, or the rotation. Data augmentation in image processing is mainly the following operation on the original image: rotating/flipping, blurring.
3. Visualization done for the number of images for the particular plant,
4. Number of diseases present in that plant including healthy one.
5. Distribution of images per class.

HOW OUR DATA LOOKS LIKE?

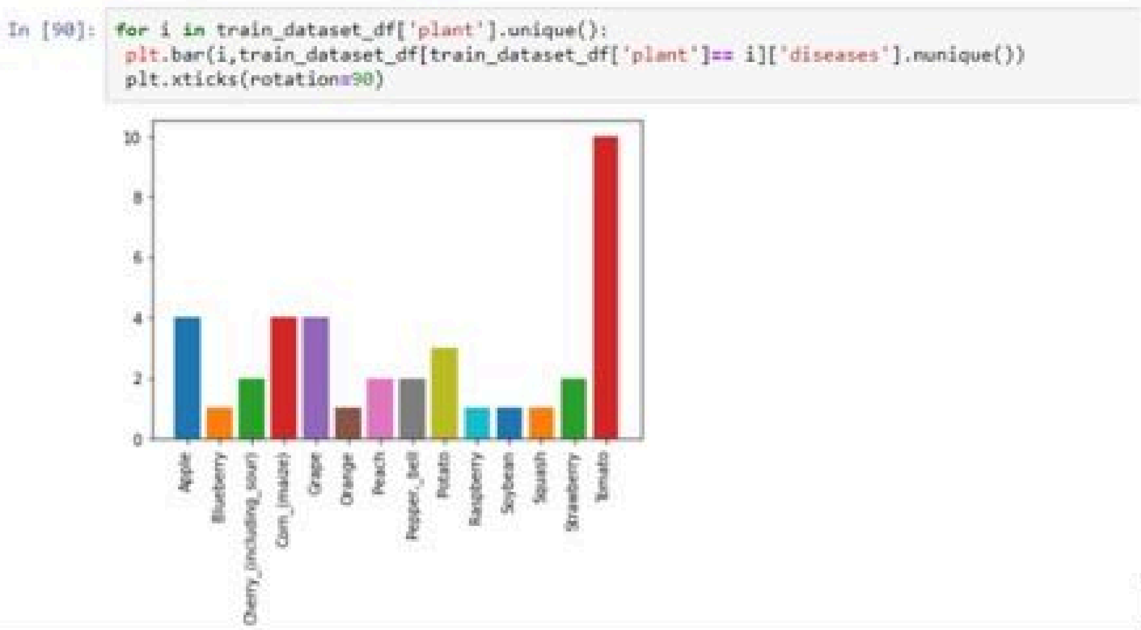
In [ ]:
```
plt.figure(figsize=(25,25))
for i,classes in enumerate(train_dataset_df['Labels'].unique()):
  plt.subplot(8,5,i+1).set_title(classes)
  plt.imshow(cv2.imread(train_dataset_df.loc[train_dataset_df['Labels']== classes].reset_index().Image_Paths[0]))
  plt.axis('off')
plt.show()
```



NUMBER OF IMAGES FOR A PARTICULAR PLANT



NUMBER OF DISEASES PRESENT IN THAT PLANT INCLUDING HEALTHY ONE

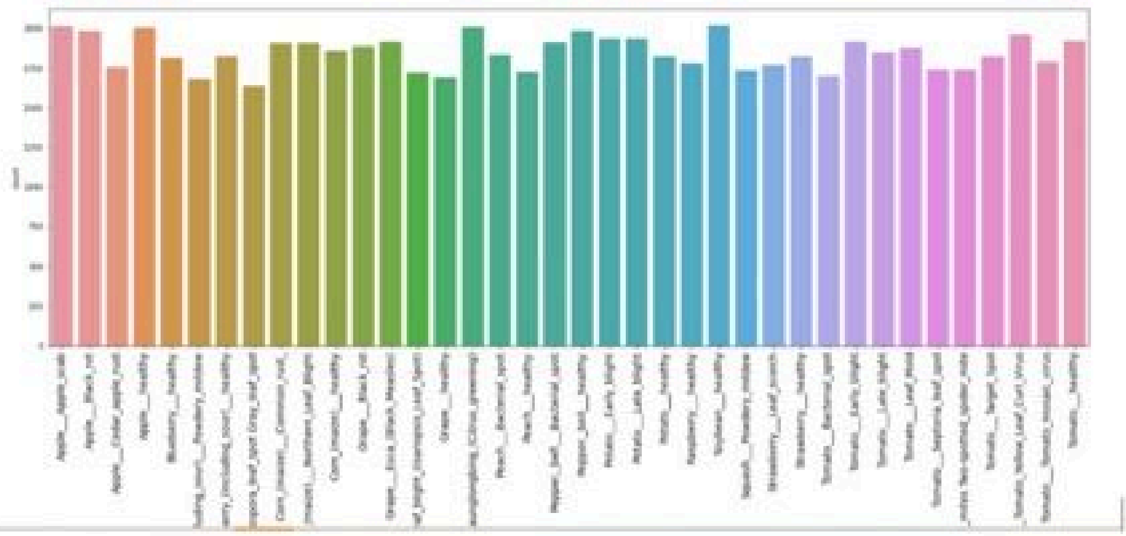


DISTRIBUTION OF NUMBER OF IMAGES PER CLASS

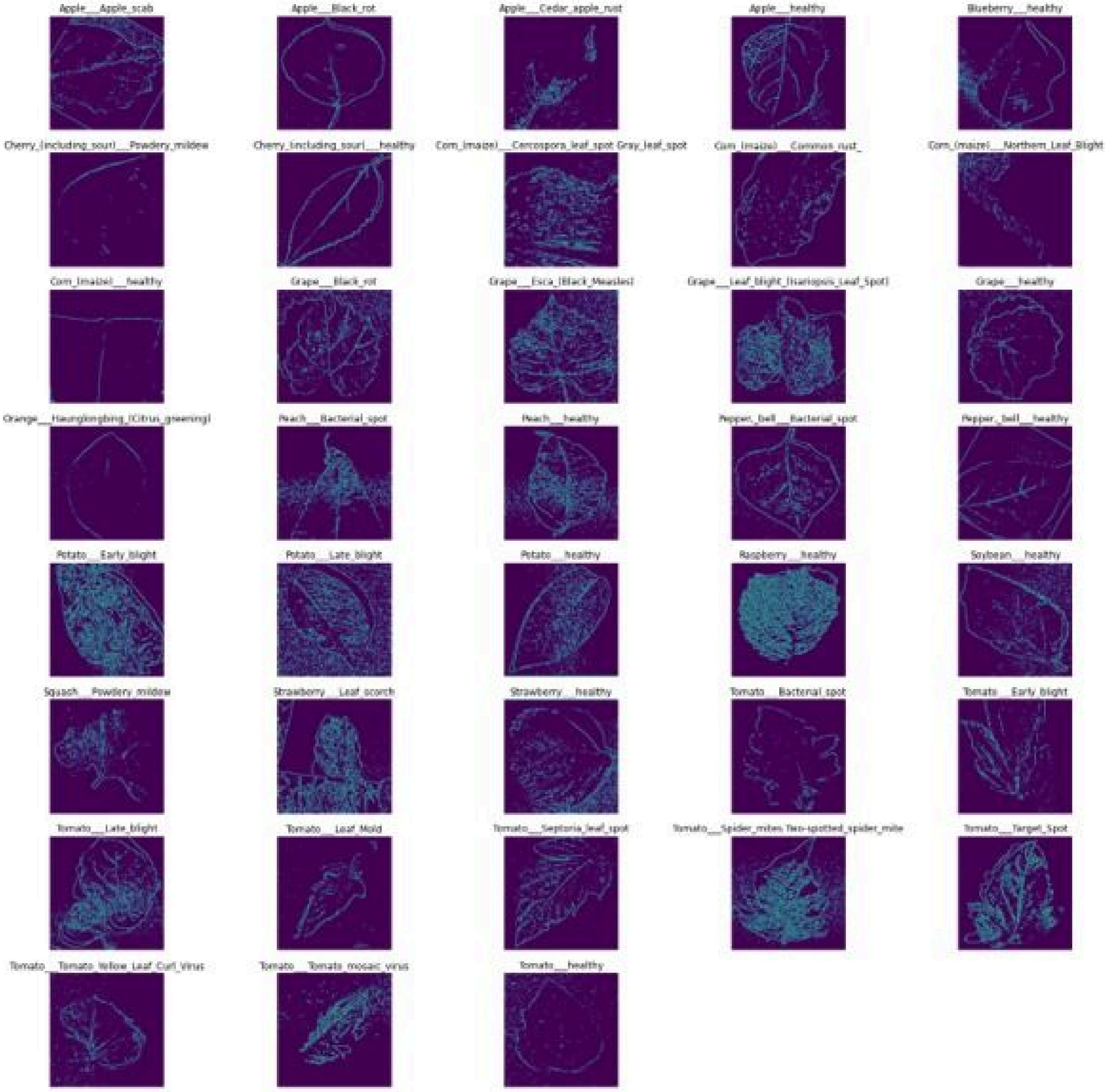
```
import seaborn as sns
plt.figure(figsize=(24,8))
sns.countplot(train_dataset_df['Labels'])
plt.xticks(rotation=90,fontsize=14)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37]), <a list of 38 Text major ticklabel objects>)
```



Using Canny Edge detector:



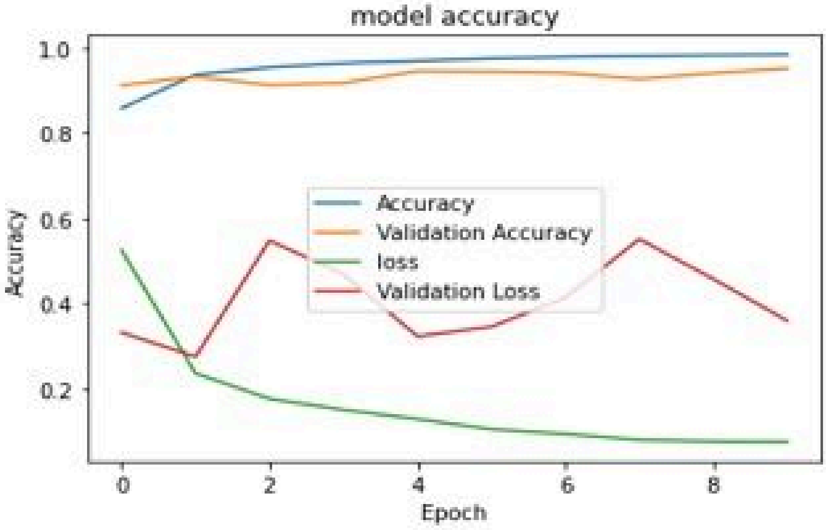
Model Building:

```
In [ ]: base_model = tf.keras.applications.VGG16(weights='imagenet', input_shape=(224,224,3), include_top =False)

        Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kerne
        ls_notop.h5
        58892288/58889256 [==============================] - 0s 0us/step
        58900480/58889256 [==============================] - 0s 0us/step

In [ ]: for layer in base_model.layers:
          layer.trainable = False

In [ ]: network =Sequential()
        network.add(base_model)

        network.add(Flatten())
        network.add(Dense(38, activation='softmax'))

In [ ]: base_model.summary()

        Model: "vgg16"
        _________________________________________________________________
         Layer (type)                Output Shape              Param #
        =================================================================
         input_1 (InputLayer)        [(None, 224, 224, 3)]     0

In [ ]: network.summary()

        Model: "sequential"
        _________________________________________________________________
         Layer (type)                Output Shape              Param #
        =================================================================
         vgg16 (Functional)          (None, 7, 7, 512)         14714688

         flatten (Flatten)           (None, 25088)             0

         dense (Dense)               (None, 38)                953382

        =================================================================
        Total params: 15,668,070
        Trainable params: 953,382
        Non-trainable params: 14,714,688
        _________________________________________________________________

In [ ]: network.compile(
          loss='categorical_crossentropy',
          optimizer='adam',
          metrics=['accuracy']
        )

In [ ]:  hist=network.fit_generator(
          train_gen,
          validation_data=valid_gen,
          epochs=10,verbose=1)

        2197/2197 [==============================] - 385s 168ms/step - loss: 0.3294 - accuracy: 0.8586 - val_loss: 0.3324 - val_accurac
        y: 0.9122
        Epoch 2/10
        2197/2197 [==============================] - 384s 175ms/step - loss: 0.2367 - accuracy: 0.9372 - val_loss: 0.2756 - val_accurac
        y: 0.9325
        Epoch 3/10
        2197/2197 [==============================] - 384s 175ms/step - loss: 0.1768 - accuracy: 0.9547 - val_loss: 0.3479 - val_accurac
        y: 0.9132
        Epoch 4/10
        2197/2197 [==============================] - 383s 174ms/step - loss: 0.1510 - accuracy: 0.9643 - val_loss: 0.4097 - val_accurac
        y: 0.9177
        Epoch 5/10
        2197/2197 [==============================] - 383s 174ms/step - loss: 0.1293 - accuracy: 0.9698 - val_loss: 0.3234 - val_accurac
        y: 0.9460
        Epoch 6/10
        2197/2197 [==============================] - 384s 175ms/step - loss: 0.1052 - accuracy: 0.9754 - val_loss: 0.3460 - val_accurac
        y: 0.9449
        Epoch 7/10
        2197/2197 [==============================] - 384s 175ms/step - loss: 0.0942 - accuracy: 0.9790 - val_loss: 0.4138 - val_accurac
        y: 0.9414
        Epoch 8/10
        2197/2197 [==============================] - 384s 175ms/step - loss: 0.0802 - accuracy: 0.9815 - val_loss: 0.5516 - val_accurac
        y: 0.9273
        Epoch 9/10
        2197/2197 [==============================] - 383s 174ms/step - loss: 0.0770 - accuracy: 0.9834 - val_loss: 0.4581 - val_accurac
        y: 0.9410
        Epoch 10/10
        2197/2197 [==============================] - 383s 174ms/step - loss: 0.0761 - accuracy: 0.9846 - val_loss: 0.3680 - val_accurac
        y: 0.9522
```

Accuracy: 0.9846

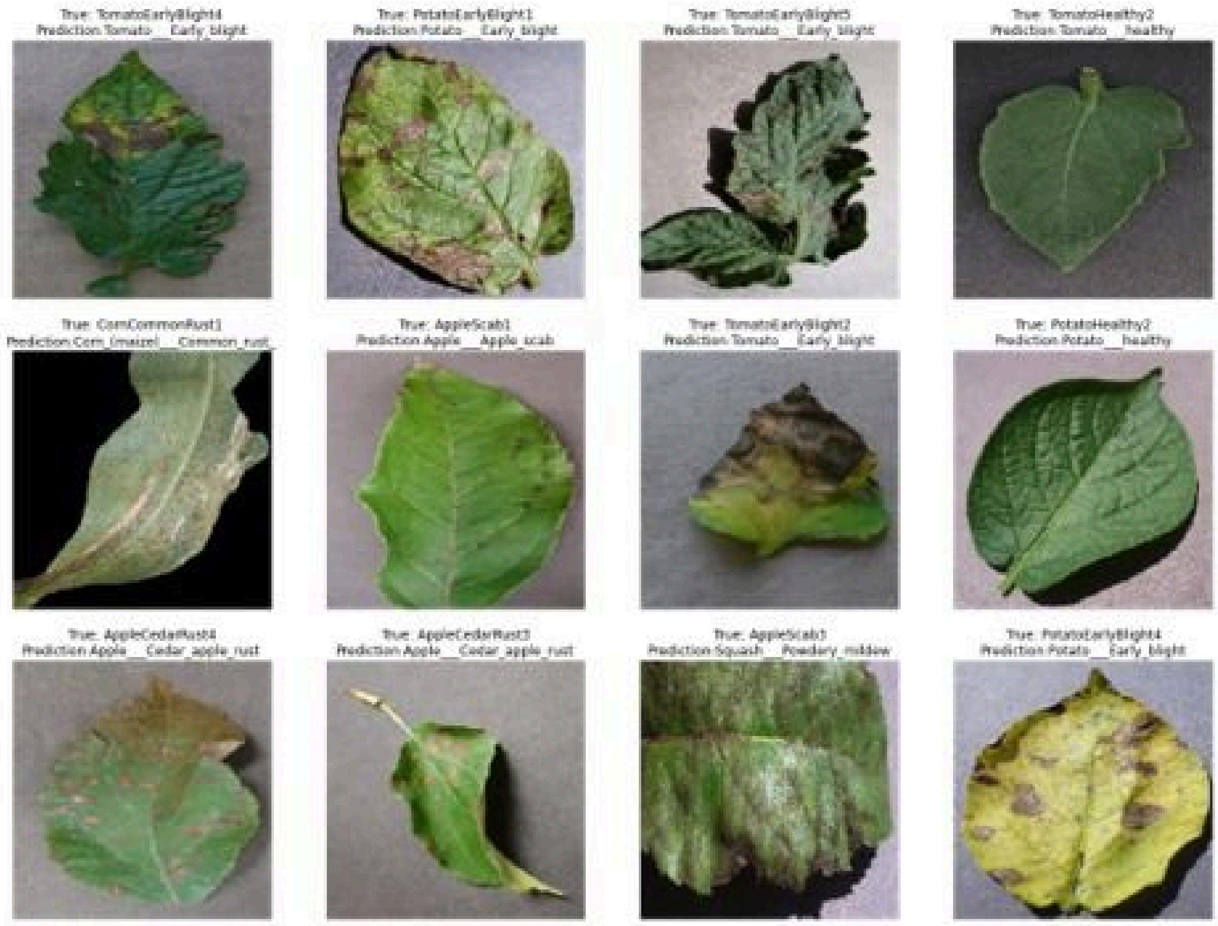
Val_accuracy: 0.9522

Plotting Accuracy and Plot



Prediction

```
In [ ]: plt.subplots(nrows = 3, ncols = 4, figsize = (20, 15))

        for i in range(12):
            plt.subplot(3, 4, i + 1)
            plt.axis(False)
            plt.grid(False)
            plt.imshow(test_image_data[i])
            plt.title(f"True: {test_image_filenames[i][:-4]}\nPrediction:{test_pred_classes[i]}")

        plt.show()
```



DEADLINES FOR THE PROJECT SUBMISSION

Submissions for a project:

1. **Project Outline Submission: (6th Jan 2022) --** We have given the gist of the project (description, datasets), techniques, plan for the project execution, etc. Date will be communicated along with the project allocation and explained to the mentors.

2. **Mid Project Submission: (18th Jan 2022) –** We have presented the EDA like extraction of features, segmentation done, visualizations, filters used, code implementation of the project, proper interpretation of the results, future scope and improvements.

3. **Final Project Submission: (25th Jan 2022) --** We have presented the final working code of the project along with challenges faced, improvements performed throughout the project, future scope, etc.

FOR MOBILE APPLICATION:

For making mob app we have tried using flutter but because of some issues and challenges while working on it, at last again we have tried to build our mob app with the use of kivy in visual studio code.

FUTURE SCOPE:

For the future scope our team 3PMS will try to use different models like INCEPTION, MOBILENET with some techniques of feature extraction and use of different filters.

PROJECT EXECUTION:

Under the guidance of Nag Sir. (Data Scientist Innomatics Research Labs, Hyderabad)

Thank You Sir From Team 3PMS

THANK YOU NOTE:

It's a wonderful experience while working on this project for all of us. We really had a best experience with Innomatics Institute. We all would like to give a big thanks to our mentor Chief Data Scientist Kanav Bansal sir and Nagamalleshwara Rao Sir for their immense support to all us throughout this internship journey.