

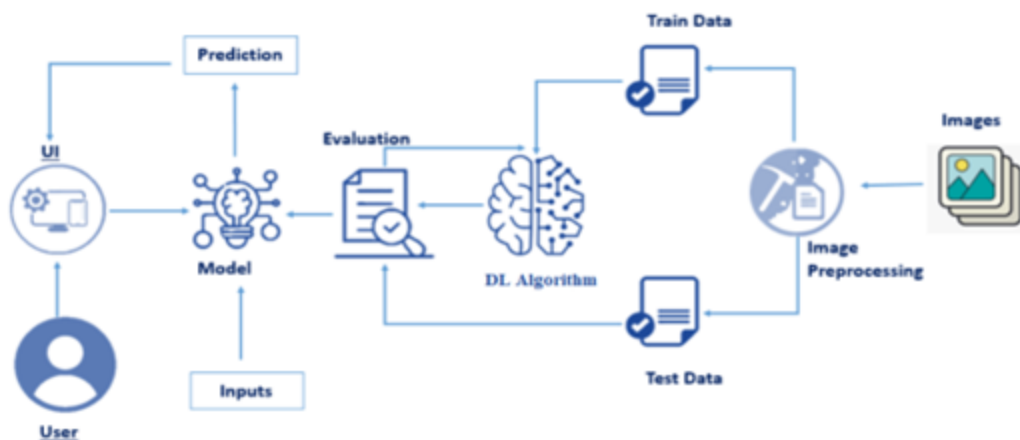
## Fertilizers Recommendation System for Disease Prediction

### Problem Statement

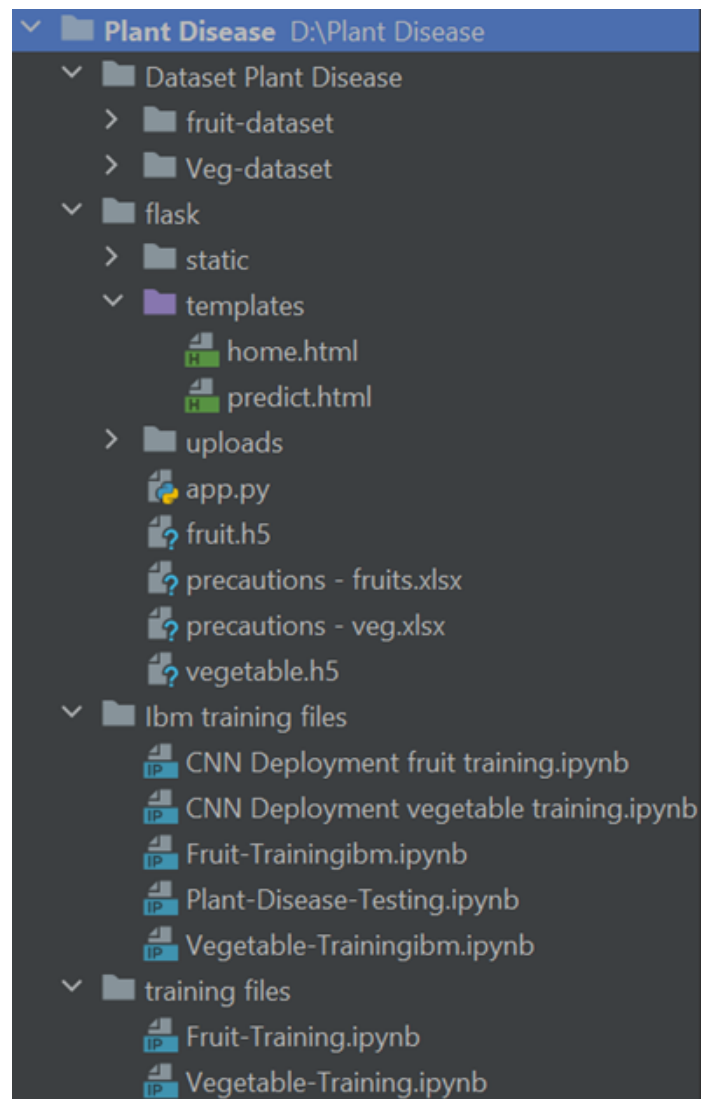
Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques.

An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

### Technical Architecture



## Project Structure



To accomplish the above tasks, the following activities must be completed:

- Download the dataset.
- Classify the dataset into train and test sets.
- Add the neural network layers.
- Load the trained images and fit the model.
- Test the model.
- Save the model and its dependencies.

- Build a Web application using a flask that integrates with the model built.

### *Download the dataset*

The dataset has been downloaded from the link provided.

### *Classify the dataset into train and test sets*

The dataset has been classified and the testing and training sets are provided separately.

### *Image Preprocessing*

With all the data collected, train the model. But, before training the model preprocess the images and then feed them on to the model for training. Keras ImageDataGenerator class is used for image preprocessing. Image Pre-processing includes the following main tasks

- Import ImageDataGenerator Library.
- Configure ImageDataGenerator Class.
- Applying ImageDataGenerator functionality to the train set and test set.

### *Build the model*

A model has been built and trained with multiple input images.

### *Test the model*

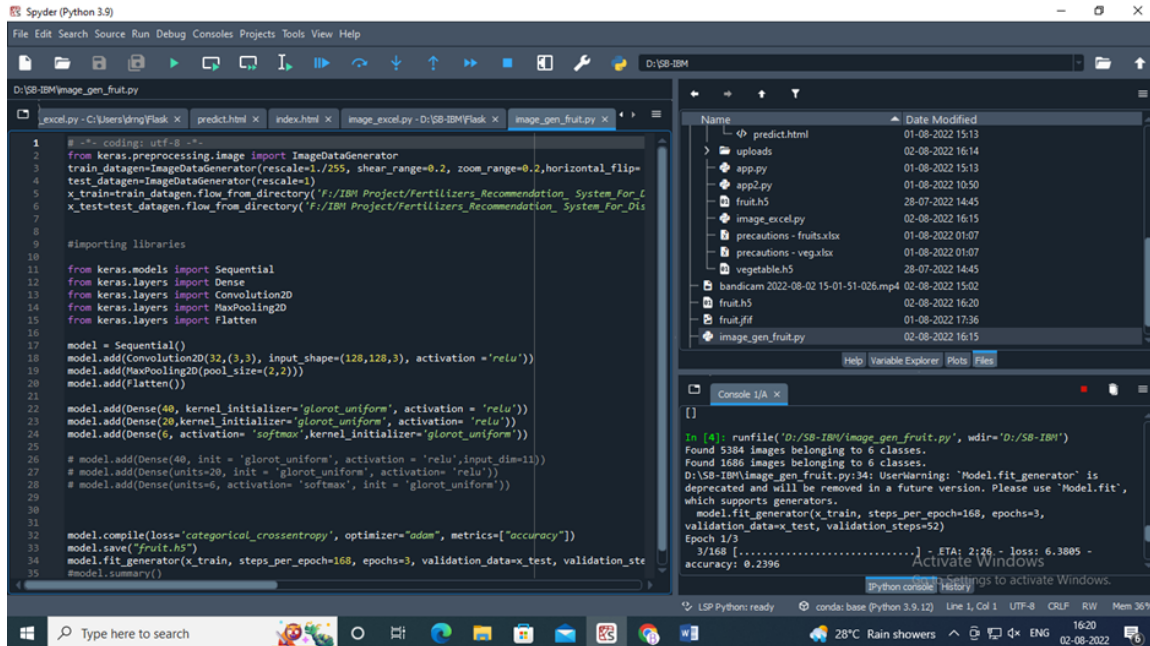
Here, the model built previously has been tested with random input images to provide precautions for the diseases that might affect.

### *Build web application*

A front end web application has been designed with Flask and integrated with the model built. The interfaces receives an image of the affected fruit or vegetable leaf as input and predicts the disease and provides precautions to overcome those diseases.

## Model Generation

### image\_gen\_fruit.py



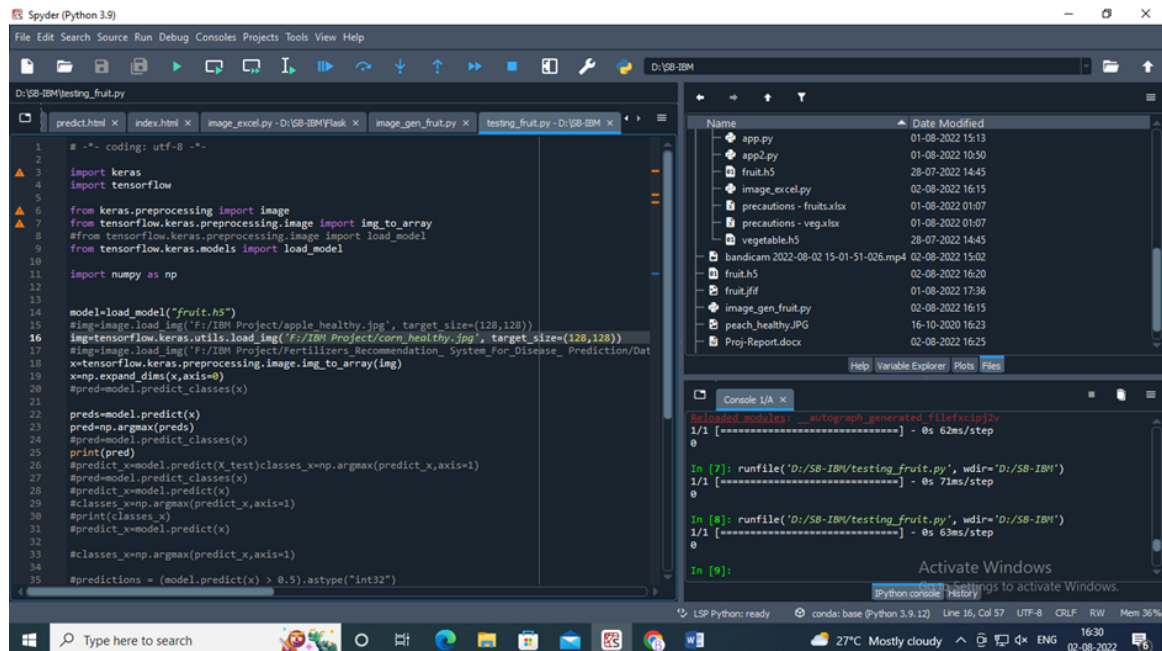
The screenshot shows the Spyder Python IDE with the file `image_gen_fruit.py` open. The script defines a Keras model for image generation. The console output shows the execution of `runfile('D:/SB-IBM/image_gen_fruit.py', wdir='D:/SB-IBM')`, which found 5384 images belonging to 6 classes and 1686 images belonging to 6 classes. It then runs `model.fit_generator(x_train, steps_per_epoch=168, epochs=3, validation_data=x_test, validation_steps=52)`, showing progress for Epoch 1/3 with a loss of 6.3085 and an accuracy of 0.2396.

```
1 #-*- coding: utf-8 -*-
2 from keras.preprocessing.image import ImageDataGenerator
3 train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
4 test_datagen=ImageDataGenerator(rescale=1)
5 x_train=train_datagen.flow_from_directory('F:/IBM Project/Fertilizers_Recommendation_System_For_Disease_Prediction/Data')
6 x_test=test_datagen.flow_from_directory('F:/IBM Project/Fertilizers_Recommendation_System_For_Disease_Prediction/Data')
7
8 #importing libraries
9
10 from keras.models import Sequential
11 from keras.layers import Dense
12 from keras.layers import Convolution2D
13 from keras.layers import MaxPooling2D
14 from keras.layers import Flatten
15
16 model = Sequential()
17 model.add(Convolution2D(32, (3, 3), input_shape=(128,128,3), activation = 'relu'))
18 model.add(MaxPooling2D(pool_size=(2,2)))
19 model.add(Flatten())
20
21 model.add(Dense(40, kernel_initializer='glorot_uniform', activation = 'relu'))
22 model.add(Dense(20, kernel_initializer='glorot_uniform', activation= 'relu'))
23 model.add(Dense(6, activation= 'softmax', kernel_initializer='glorot_uniform'))
24
25 # model.add(Dense(40, init = 'glorot_uniform', activation = 'relu', input_dim=11))
26 # model.add(Dense(units=20, init = 'glorot_uniform', activation= 'relu'))
27 # model.add(Dense(units=6, activation= 'softmax', init = 'glorot_uniform'))
28
29 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
30
31 model.save('fruit.h5')
32
33 model.fit_generator(x_train, steps_per_epoch=168, epochs=3, validation_data=x_test, validation_steps=52)
34 model.summary()
```

Console Output:

```
In [4]: runfile('D:/SB-IBM/image_gen_fruit.py', wdir='D:/SB-IBM')
Found 5384 images belonging to 6 classes.
Found 1686 images belonging to 6 classes.
D:/SB-IBM/image_gen_fruit.py:34: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.
model.fit_generator(x_train, steps_per_epoch=168, epochs=3, validation_data=x_test, validation_steps=52)
Epoch 1/3
3/168 [.....] - ETA: 2.36s - loss: 6.3085 - accuracy: 0.2396
```

### testing\_fruit.py



The screenshot shows the Spyder Python IDE with the file `testing_fruit.py` open. The script loads the trained model and tests it on a new image. The console output shows the execution of `runfile('D:/SB-IBM/testing_fruit.py', wdir='D:/SB-IBM')`, which loaded the model and predicted the class for the input image.

```
1 #-*- coding: utf-8 -*-
2 import keras
3 import tensorflow
4
5 from keras.preprocessing import image
6 from tensorflow.keras.preprocessing.image import img_to_array
7 from tensorflow.keras.preprocessing.image import load_model
8 from tensorflow.keras.models import load_model
9
10 import numpy as np
11
12 model=load_model('fruit.h5')
13
14 #img=image.load_img('F:/IBM Project/apple healthy.jpg', target_size=(128,128))
15 #img=load_img('F:/IBM Project/apple healthy.jpg', target_size=(128,128))
16 #img=load_img('F:/IBM Project/Fertilizers_Recommendation_System_For_Disease_Prediction/Data')
17 #img=load_img('F:/IBM Project/Fertilizers_Recommendation_System_For_Disease_Prediction/Data')
18 x=load_img('F:/IBM Project/Fertilizers_Recommendation_System_For_Disease_Prediction/Data')
19 x=np.expand_dims(x,axis=0)
20 #pred=model.predict_classes(x)
21
22 preds=model.predict(x)
23 pred=np.argmax(preds)
24 #pred=model.predict_classes(x)
25 print(pred)
26 #pred_x=model.predict(X_test)classes_x=np.argmax(pred_x,axis=1)
27 #pred=model.predict_classes(x)
28 #pred_x=model.predict(x)
29 #classes_x=np.argmax(pred_x,axis=1)
30 #print(classes_x)
31 #pred_x=model.predict(x)
32
33 #classes_x=np.argmax(pred_x,axis=1)
34
35 #predictions = (model.predict(x) > 0.5).astype("int32")
```

Console Output:

```
In [2]: runfile('D:/SB-IBM/testing_fruit.py', wdir='D:/SB-IBM')
1/1 [.....] - 0s 71ms/step
0
In [8]: runfile('D:/SB-IBM/testing_fruit.py', wdir='D:/SB-IBM')
1/1 [.....] - 0s 63ms/step
0
In [9]:
```

## Output of Testing

The screenshot shows the Spyder Python IDE with the file `testing_fruit.py` open. The code defines a Keras model for fruit disease classification. The output of the model is displayed in the console window.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Jul 26 12:53:23 2022
4
5 @author: drng
6 """
7 import keras
8 import tensorflow
9
10 from keras.preprocessing import image
11 from tensorflow.keras.preprocessing.image import img_to_array
12 from tensorflow.keras.preprocessing.image import load_model
13 from tensorflow.keras.models import load_model
14
15 import numpy as np
16
17 model=load_model("fruit.h5")
18 #img=image.load_img('F:/IBM Project/apple_healthy.jpg', target_size=(128,128))
19 img=tensorflow.keras.utils.load_img('F:/IBM Project/apple_healthy.jpg',
20 #img=image.load_img('F:/IBM Project/Fertilizers_Recommendation_System
21 x=tensorflow.keras.preprocessing.image.img_to_array(img)
22 x=np.expand_dims(x,axis=0)
23 #pred=model.predict_classes(x)
24
25 preds=model.predict(x)
26 pred=np.argmax(preds)
27 #pred=model.predict_classes(x)
28 print(pred)
29 #predict_x=model.predict(X_test)classes_x=np.argmax(predict_x,axis=1)
30 #pred=model.predict_classes(x)
31 #predict_x=model.predict(x)
32 #classes_x=np.argmax(predict_x,axis=1)
33 #print(classes_x)
34 #pred=model.predict(x)
35
```

The console window shows the following output:

```
Layer (type) Output Shape Param #
-----
conv2d (Conv2D) (None, 126, 126, 32) 896
max_pooling2d (MaxPooling2D) (None, 63, 63, 32) 0
flatten (Flatten) (None, 127008) 0
dense (Dense) (None, 40) 5080360
dense_1 (Dense) (None, 20) 820
dense_2 (Dense) (None, 6) 126
-----
Total params: 5,082,202
Trainable params: 5,082,202
Non-trainable params: 0
```

The console also shows the prediction result:

```
In [4]:
```

## app.py

The screenshot shows the Spyder Python IDE with the file `app.py` open. The code defines a Flask web application for fruit disease classification. The output of the application is displayed in the console window.

```
13 model=load_model("fruit.h5")
14 @app.route("/")
15 def home():
16     return render_template("index.html")
17
18 @app.route("/prediction")
19 def prediction():
20     return render_template("predict.html")
21
22 @app.route("/predict",methods=['POST'])
23
24
25 def predict():
26     if request.method=="POST":
27         f=request.files['image']
28         basepath=os.path.dirname(__file__)
29         file_path=os.path.join(
30             basepath,'uploads',secure_filename(f.filename))
31         f.save(file_path)
32         img=image.load_img(file_path,target_size=(128,128))
33         x=image.img_to_array(img)
34         x=np.expand_dims(x,axis=0)
35         plant=request.form['plant']
36         preds=model.predict(x)
37         print(preds)
38         test = " ".join([str(item) for item in preds])
39         #return test
40         if(plant=="fruit"):
41             y = np.argmax(model.predict(x),axis=1)
42             index = ['Apple__Black_rot','Apple__healthy','Corn_(maize)__Northern_Leaf_Blight',
43             dfpred.read_excel('precautions - fruits.xlsx')
44             text = df.iloc[y[0]]['caution']
45             print(df.iloc[y[0]]['caution'])
46
```

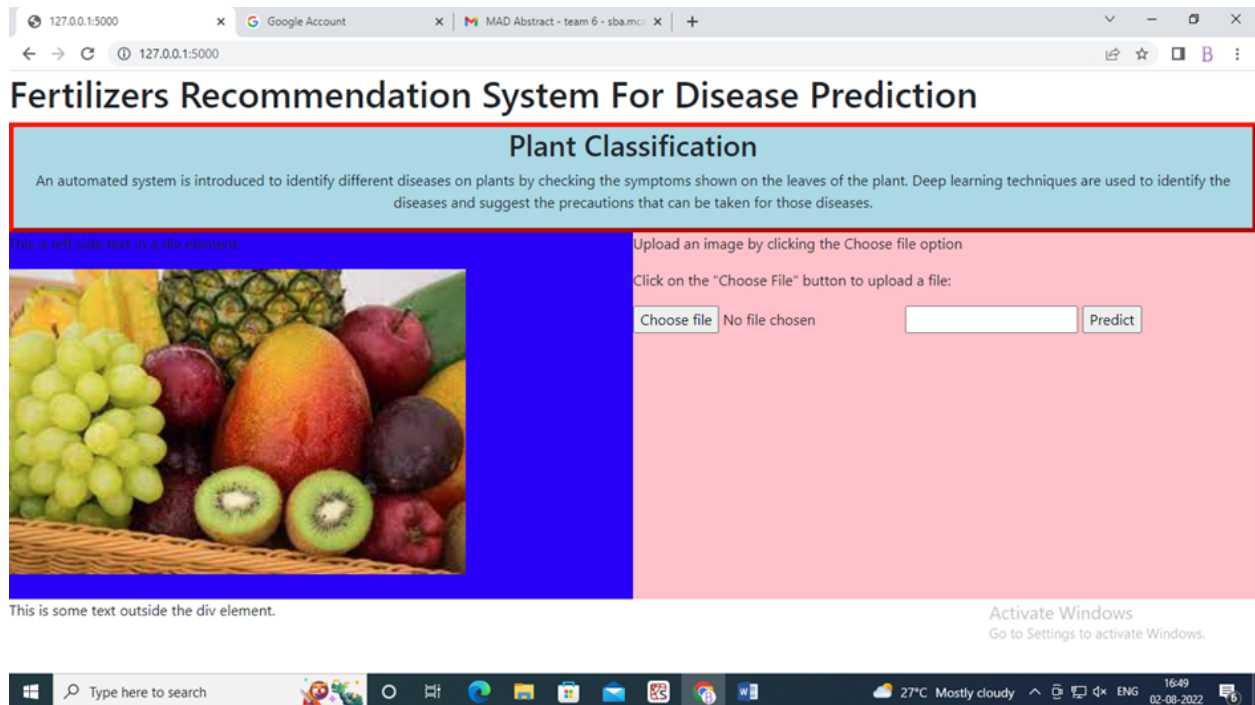
The console window shows the following output:

```
1/1 [=====] - 0s 63ms/step
0
```

The console also shows the prediction result:

```
In [9]: runfile('D:/SB-IBM/Flask/app.py', wdir='D:/SB-IBM/Flask')
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production
deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Restarting with watchdog (windowsapi)
```

## Prediction Output



### Precaution message

