A Micro Project report on

### DIAGNOSE CROP DISEASE WITH MACHINE LEARNING

Submitted to the CMR Institute of Technology in partial fulfilment of the requirement for the award of the Laboratory of

### MACHINE LEARNING AND DATA SCIENCE LAB

**of**

**III B.Tech. II Semester**

#### In

**Computer Science and Engineering**

Submitted by

**A.DILEEP (20R01A0561)**

**A. YOGESHWAR (20R01A0562)**

**B.SWATHI (20R01A0563)**

**B. NITHIN GOUD (20R01A0564)**

**B. VIGNESH (20R01A0565)**

**B. PARAMESHWARA (20R01A0566)**

**B .KARTHIK REDDY (20R01A0567)**

Under the Guidance

#### of Dr.S.Alagumuthukrishnan

(Associate Professor)

#### CMR INSTITUTE OF TECHNOLOGY (UGC AUTONOMUS)

**(Approved by AICTE,Affiliated to JNTU,Kukatpally,Hyderabad) Kandlakoya,Medchal Road,Hyderabad**

### 2022-2023

**CMR INSTITUTE OF TECHNOLOGY (UGC AUTONOMOUS)**

**(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad) Kandlakoya, Medchal Road, Hyderabad.**



**CERTIFICATE**

### This is to certify that a Micro Project entitled with: “DIAGNOSE CROP DISEASE WITH MACHINE LEARNING” is being

Submitted By

**A.DILEEP (20R01A0561)**

**A. YOGESHWAR (20R01A0562)**

**B.SWATHI (20R01A0563)**

**B. NITHIN GOUD (20R01A0564)**

**B. VIGNESH (20R01A0565)**

**B. PARAMESHWARA (20R01A0566)**

**B .KARTHIK REDDY (20R01A0567)**

In partial fulfillment of the requirement for award of the Machine Learning and Data Science Laboratory of III B. Tech II Semester in CSE to the CMRIT, Hyderabad is a record of a bonafide work carried out under our guidance and supervision.

**Signature of Faculty Signature of HOD**

**Dr.S. Alagumuthukrishnan Prof. A. Prakash (Associate Professor)**

### ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy**, **Director**, CMR Institute of Technology, **Dr. B. Satyanarayana**, **Principal** and **Prof.A. Prakash**, **Head of Department**, Dept. of Computer Science and Engineering), CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to our Machine Learning and Data Science Lab faculty in-charge **Dr.S.Alagumuthukrishnan**, Dept. of Computer Science and Engineering, CMR Institute of Technology for his constant guidance, encouragement and moral support throughout the project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for successful completion of the project.

**A.DILEEP (20R01A0561)**

**A. YOGESHWAR (20R01A0562)**

**B.SWATHI (20R01A0563)**

**B. NITHIN GOUD (20R01A0564)**

**B. VIGNESH (20R01A0565)**

**B. PARAMESHWARA (20R01A0566)**

**B .KARTHIK REDDY (20R01A0567)**

### CONTENTS

#### Particulars Page No.

1. Introduction 5
2. Algorithm 6
3. Requirements (S/W and H/W) 8
4. Implementation 9
5. Results 11
6. Conclusion 15
7. References 15

**INTRODUCTION**

Rainfall pattern prediction revolves around the forecasting of the trend in rainfall (like intensity and time) based on its previous results. Rainfall prediction that can predict heavy or no rainfall can, in turn, prevent any huge risk such as property damage, flood, or drought. Accuracy in such predictions is very crucial and for that purpose traditional methods prove inefficient. Therefore, we require a reliable technique like machine learning models to take up such tasks. Machine Learning models when perfectly tuned can predict with the finest accuracies. Accurate forecasting can help the agricultural sector. Like farmers for instance can choose which crops to grow, when and how much they should sow and harvest the yields etc. with minimal loss, maximum profit, and optimum methodologies and standards.

## ALGORITHM

Four machine learning algorithms have been used to implement this project, namely

* Linear Regression
* Lasso Regression
* Random Forest

### Linear Regression

The most elementary regression model is the simple linear regression which explains the linear relationship between the dependent variable (Target variable) and one independent variable using a straight line

Y = β0 + β1X

The best fit line is identified by minimizing the error term RSS (Residual sum of error) which is the sum of squares of residual for each point. Residual for a point is identified by subtracting the actual value with the predicted value. Linear regression makes the following four assumptions:

1. There should be a linear and additive relationship be- tween the dependent and independent variable (Linearity of residual ). If a linear model is fit to a non-linear, non-additive set, then the regression model will fail to capture the trend, thus resulting in an inefficient model. This will lead to erroneous prediction on unseen data.
2. The error terms should not be dependent on one another, and there should be no correlation between the residual (error) terms.
3. The mean of residual should follow a normal distribution with mean equal to zero or close to zero. This is done

to check whether the selected line is actually the line of best fit.

1. The error term must have a constant variance. This phenomenon is known as homoscedasticity. The presence of non-constant variance is referred as heteroscedasticity.

### Lasso Regression

Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

### Random Forest

Random Forest is a classifier that contains a number of individual decision trees on numerous subsets of the given dataset and takes the average to enhance the predictive accu- racy of that dataset. Rather than depending on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. One big advantage of random forest is that it can be used for both classification and regression problems. In ML language, random forests are also called an ensemble or bagging method. Random Forest does the tasks in two- phases. First being, to create the random forest by combining N decision tree, while the second task is to make predictions for each tree created in the first phase.

The Working process can be explained in the following steps: Step-1: Selecting random K data points from the training set.

Step-2: Building the decision trees associated with the selected data points. Step-3: Choosing the number N for decision trees that you want to build.

Step-4: Repeating Step 1 and 2.

Step-5: Finding the predictions of each decision tree, for new data points, and assigning the new data points to the category that wins the majority votes.

## REQUIRMENTS

**Hardware:**

* + - Computer
    - Hard Disk

## Software:

* + - Python 3.8
    - Windows 7 and Above
    - Numpy
    - Pandas
    - MatplotLib

## IMPLEMENTATION

**import** os

os**.**listdir("../input/plant-diseases-classification-using-alexnet")

*# Importing Keras libraries and packages*

**from** keras.models **import** Sequential

**from** keras.layers **import** Convolution2D

**from** keras.layers **import** MaxPooling2D

**from** keras.layers **import** Flatten

**from** keras.layers **import** Dense

**from** keras.layers **import** Dropout

**from** keras.layers.normalization **import** BatchNormalization

*# Initializing the CNN*

classifier **=** Sequential()

*# Convolution Step 1*

classifier**.**add(Convolution2D(96, 11, strides **=** (4, 4), padding **=** 'valid', input\_shape**=**(224, 224, 3), activation **=** 'relu'))

classifier**.**add(MaxPooling2D(pool\_size **=** (2, 2), strides **=** (2, 2), padding **=** 'valid'))

classifier**.**add(BatchNormalization())

classifier**.**add(Convolution2D(256, 11, strides **=** (1, 1), padding**=**'valid', activation **=** 'relu'))

classifier**.**add(MaxPooling2D(pool\_size **=** (2, 2), strides **=** (2, 2), padding**=**'valid'))

classifier**.**add(BatchNormalization())

classifier**.**add(Convolution2D(384, 3, strides **=** (1, 1), padding**=**'valid', activation **=** 'relu'))

classifier**.**add(BatchNormalization())

classifier**.**add(Convolution2D(384, 3, strides **=** (1, 1), padding**=**'valid', activation **=** 'relu'))

classifier**.**add(BatchNormalization())

classifier**.**add(Convolution2D(256, 3, strides**=**(1,1), padding**=**'valid', activation **=** 'relu'))

classifier**.**add(MaxPooling2D(pool\_size **=** (2, 2), strides **=** (2, 2), padding **=** 'valid'))

classifier**.**add(BatchNormalization())

classifier**.**add(Flatten())

*# Full Connection Step*

classifier**.**add(Dense(units **=** 4096, activation **=** 'relu'))

classifier**.**add(Dropout(0.4))

classifier**.**add(BatchNormalization())

classifier**.**add(Dense(units **=** 4096, activation **=** 'relu'))

classifier**.**add(Dropout(0.4))

classifier**.**add(BatchNormalization())

classifier**.**add(Dense(units **=** 1000, activation **=** 'relu'))

classifier**.**add(Dropout(0.2))

classifier**.**add(BatchNormalization())

classifier**.**add(Dense(units **=** 38, activation **=** 'softmax'))

classifier**.**summary()

*# let's visualize layer names and layer indices to see how many layers*

*# we should freeze:*

**from** keras **import** layers

**for** i, layer **in** enumerate(classifier**.**layers):

print(i, layer**.**name)

*# the first 8 layers and unfreeze the rest:*

print("Freezed layers:")

**for** i, layer **in** enumerate(classifier**.**layers[:20]):

print(i, layer**.**name)

layer**.**trainable **=** **False**

*#trainable parameters decrease after freezing some bottom layers*

classifier**.**summary()

*# Compiling the Model*

**from** keras **import** optimizers

classifier**.**compile(optimizer**=**optimizers**.**SGD(lr**=**0.001, momentum**=**0.9, decay**=**0.005),

loss**=**'categorical\_crossentropy',

metrics**=**['accuracy'])

*# image preprocessing*

**from** keras.preprocessing.image **import** ImageDataGenerator

train\_datagen **=** ImageDataGenerator(rescale**=**1.**/**255,

shear\_range**=**0.2,

zoom\_range**=**0.2,

width\_shift\_range**=**0.2,

height\_shift\_range**=**0.2,

fill\_mode**=**'nearest')

valid\_datagen **=** ImageDataGenerator(rescale**=**1.**/**255)

batch\_size **=** 128

base\_dir **=** "../input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)"

training\_set **=** train\_datagen**.**flow\_from\_directory(base\_dir**+**'/train',

target\_size**=**(224, 224),

batch\_size**=**batch\_size,

class\_mode**=**'categorical')

valid\_set **=** valid\_datagen**.**flow\_from\_directory(base\_dir**+**'/valid',

target\_size**=**(224, 224),

batch\_size**=**batch\_size,

class\_mode**=**'categorical')

class\_dict **=** training\_set**.**class\_indices

print(class\_dict)

li **=** list(class\_dict**.**keys())

print(li)

train\_num **=** training\_set**.**samples

valid\_num **=** valid\_set**.**samples

*# checkpoint*

**from** keras.callbacks **import** ModelCheckpoint

weightpath **=** "best\_weights\_9.hdf5"

checkpoint **=** ModelCheckpoint(weightpath, monitor**=**'val\_acc', verbose**=**1, save\_best\_only**=True**, save\_weights\_only**=True**, mode**=**'max')

callbacks\_list **=** [checkpoint]

*#fitting images to CNN*

history **=** classifier**.**fit\_generator(training\_set,

steps\_per\_epoch**=**train\_num**//**batch\_size,

validation\_data**=**valid\_set,

epochs**=**25,

validation\_steps**=**valid\_num**//**batch\_size,

callbacks**=**callbacks\_list)

*#saving model*

filepath**=**"AlexNetModel.hdf5"

classifier**.**save(filepath)

*#plotting training values*

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

sns**.**set()

acc **=** history**.**history['acc']

val\_acc **=** history**.**history['val\_acc']

loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(1, len(loss) **+** 1)

*#accuracy plot*

plt**.**plot(epochs, acc, color**=**'green', label**=**'Training Accuracy')

plt**.**plot(epochs, val\_acc, color**=**'blue', label**=**'Validation Accuracy')

plt**.**title('Training and Validation Accuracy')

plt**.**ylabel('Accuracy')

plt**.**xlabel('Epoch')

plt**.**legend()

plt**.**figure()

*#loss plot*

plt**.**plot(epochs, loss, color**=**'pink', label**=**'Training Loss')

plt**.**plot(epochs, val\_loss, color**=**'red', label**=**'Validation Loss')

plt**.**title('Training and Validation Loss')

plt**.**xlabel('Epoch')

plt**.**ylabel('Loss')

plt**.**legend()

plt**.**show()

*# predicting an image*

**from** keras.preprocessing **import** image

**import** numpy **as** np

image\_path **=** "../input/new-plant-diseases-dataset/test/test/TomatoEarlyBlight1.JPG"

new\_img **=** image**.**load\_img(image\_path, target\_size**=**(224, 224))

img **=** image**.**img\_to\_array(new\_img)

img **=** np**.**expand\_dims(img, axis**=**0)

img **=** img**/**255

print("Following is our prediction:")

prediction **=** classifier**.**predict(img)

*# decode the results into a list of tuples (class, description, probability)*

*# (one such list for each sample in the batch)*

d **=** prediction**.**flatten()

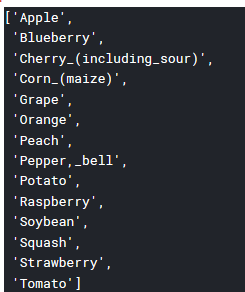
j **=** d**.**max()

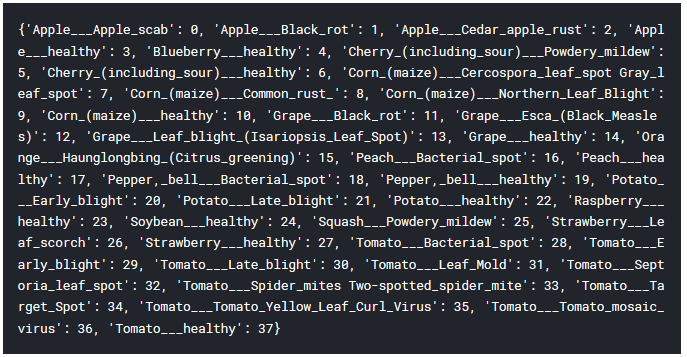
**for** index,item **in** enumerate(d):

**if** item **==** j:

class\_name **=** li[index]

# RESULTS





**Conclusion**

This project can be used for simplifying the process of crop disease detection. This project follows one touch process for detection of disease in crop if present. By detecting disease one can take steps for treatment of the same. This project will be beneficial on the specific crop category on which it is trained. This project can save a lot of time which is wasted on detection of diseases found in crop that too with good accuracy.

# 

# References