MACHINE LEARNING

(FACE MASK CLASSIFIER)

Summer Internship Report Submitted in partial fulfillment of the

requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science and Engineering

By

P SAI SANTOSHI

RAMYA

221710309041

Under the

Guidance of

Mr. M.

Venkateswarlu

Assistant

Professor

Department Of Computer Science and
Engineering GITAM School of
Technology
GITAM (Deemed to be
University) Hyderabad-
502329
June 2020

DECLARATION

I submit this industrial training work entitled **"FACE MASK CLASSIFIER"** to

GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the

requirements for the award of the degree of "**Bachelor of Technology**" in

"**Computer Science and Engineering**". I declare that it was carried out

independently by me under the guidance of

Mr. M. Venkateswarlu, Asst. Professor, GITAM (Deemed To Be University),

Hyderabad, India.

The results embodied in this report have not been submitted to

any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD   P.SAI
                                                                SANTOSHI
                                                                 RAMYA

Date: 221710309041

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-

502329, India

Dated:

CERTIFICATE

      This is to certify that the Industrial Training Report entitled

"FACE MASK CLASSIFIER" is being submitted by P. Sai Santoshi

Ramya(221710309041) in partial fulfillment of the requirement for the award of

Bachelor of Technology in

Computer Science and Engineering at GITAM (Deemed To Be University),

Hyderabad during the academic year 2018-19

  It is faithful record work carried out by her at the **Computer Science and**

Engineering Department, GITAM University Hyderabad Campus under my

guidance and supervision.

 Mr. M. Venkateswarlu   **Dr.K.Manjunathachari**

Assistant Professor   Professor and HOD

Department of ECE   Department of ECE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr.   N. Siva Prasad,** Pro Vice Chancellor, GITAM Hyderabad and **Dr. CH. Sanjay,** Principal, GITAM Hyderabad

I would like to thank respected **Dr. K. Manjunathachari,** Head of the Department of Electronics and Communication Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me

guidelines to present a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mr. M. Venkateswarlu** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

P

Sai

Santo

shi

Ramy

a

221710309041

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on Cereals data set My perception of understanding the given data set has been in the view of undertaking a client's requirement of overcoming the stagnant point of sales of the products being manufactured by client.

To get a better understanding and work on a strategical approach for solution of the client, I have adapted the view point of looking at ratings of the products and for further deep understanding of the problem, I have taken the stance of a consumer and reasoned out the various factors of choice of the products and they purchase , and my primary objective of this case study was to look up the factors which were dampening the sale of products and corelate them to ratings of products and draft out an outcome report to client regarding the various accepts of a product manufacturing , marketing and sale point determination

Table of Contents:

LIST OF FIGURES:

CHAPTER 1 MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models

that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide

variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data

extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

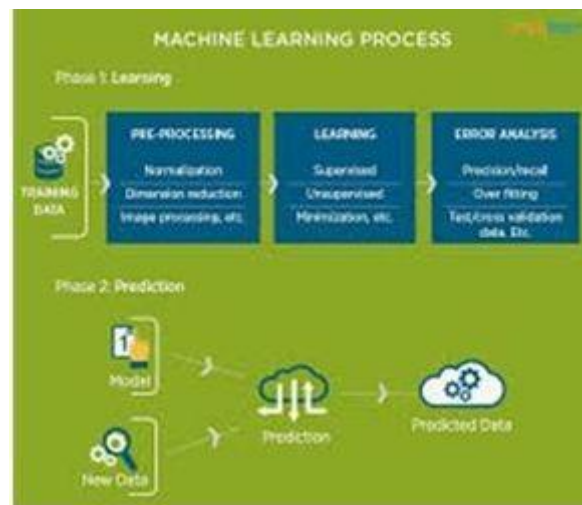The process flow depicted here represents how machine learning works

Figure 1 : The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and

heterogeneous. Machine learning comes as the solution to all this
chaos by proposing clever alternatives to analyzing huge volumes of
data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and
analysis.

TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of
data they input and output, and the type of task or problem that they are
intended to solve.

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses

that can consist of numeric values or string labels, such as classes

or tags, in order to later predict the correct response when posed

with new examples

comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights,

patterns, and relationships from a labelled training dataset – that is, a dataset that

already contains a known value for the target variable for each

record. Because you provide the machine learning algorithm

with the correct answers for a problem during training, it is able

to "learn" how the rest of the features relate to the target,

enabling you to uncover insights and make predictions about

future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression,
in

which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example

b
y

choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

Unsupervised Learning:

When an algorithm learns from plain examples without any associated response,

leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.
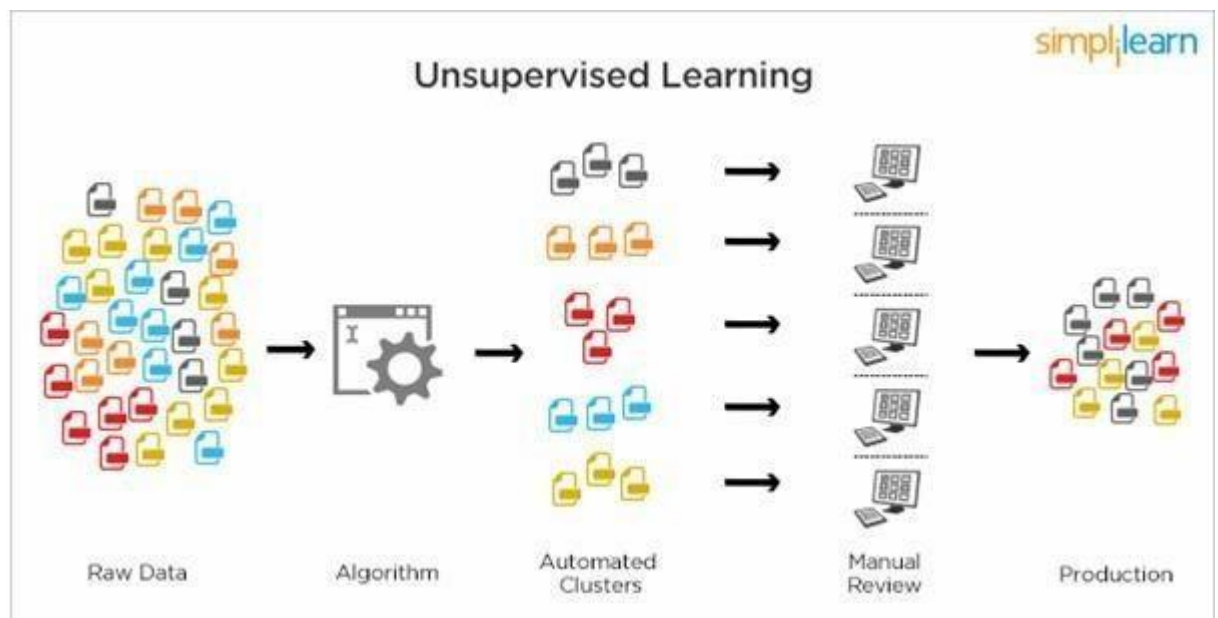
Figure 2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.
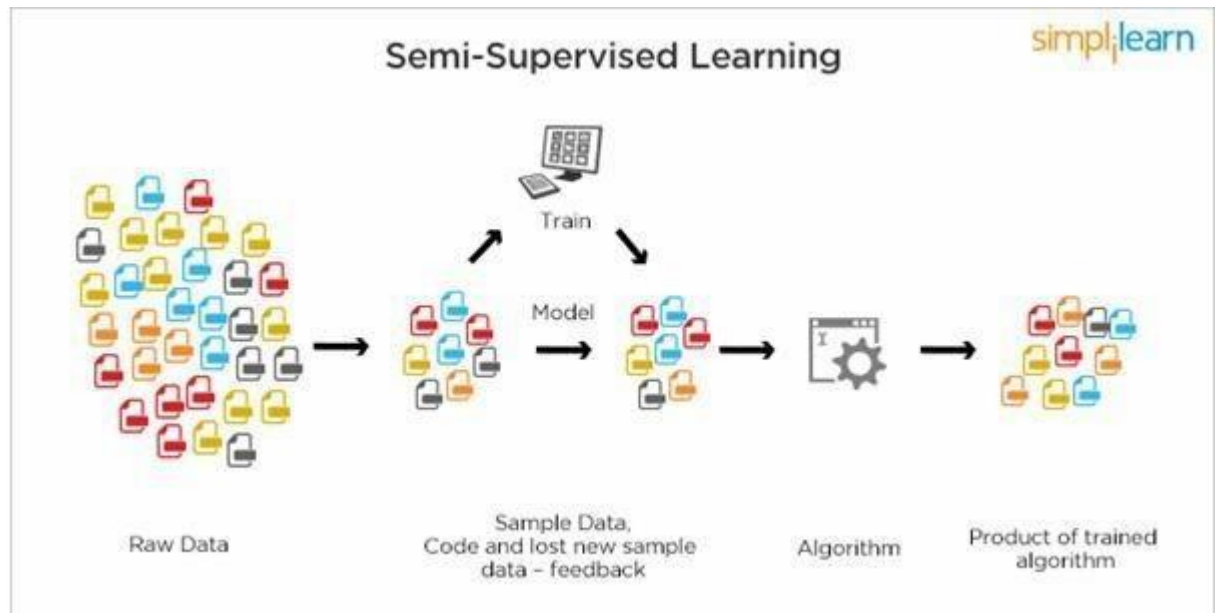
Figure 3 : Semi Supervised Learning

1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 2   PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 INTRODUCTION TO PYHTON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.

- Python is a general purpose programming language that is often applied in scripting roles

- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need

to compile your program before executing it. This is like PERL and PHP.

- Python is Interactive: You can sit at a Python prompt and interact with the interpreter

directly to write your programs.

- Python is Object-Oriented: Python supports the Object-Oriented style or technique of

programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's

- Its latest version is 3.7 , it is generally called as python3

2.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax,

This allows the student to pick up the language quickly.

- Easy-to-read: Python code is more clearly defined and visible to the eyes.

- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.

- A broad standard library: Python's bulk of the library is very portable and cross-platform

compatible on UNIX, Windows, and Macintosh.

- Portable: Python can run on a wide variety of hardware platforms and has the same

interface on all platforms.

- Extendable: You can add low-level modules to the Python interpreter. These modules

enable programmers to add to or customize their tools to be more efficient.

- Databases: Python provides interfaces to all major commercial databases.

- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's

understand how to set up our Python environment.

- The most up-to-date and current source code, binaries, documentation, news, etc., is

available on the official website of Python.

2.4.1 Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS

distributions include a recent python.

- Download python from www.python.org

- When the download is completed, double click the file and follow the instructions

to install it.

- When python is installed, a program called IDLE is also installed along with it. It

provides a graphical user interface to work with python.

Figure 4 : Python download

2.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.

- Anaconda is a free open source distribution of python for large scale data

processing, predictive analytics and scientific computing.

- Conda is a package manager quickly installs and manages packages.

- In WINDOWS:

- In windows

- Step 1: Open Anaconda.com/downloads in web browser.

- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit

  graphic installer)

- Step 3: select installation type(all users)

- Step 4: Select path(i.e. add anaconda to path & register anaconda as default

  python 3.4) next click install and next click finish

- Step 5: Open jupyter notebook (it opens in default browser)

Figure 5 : Anaconda download

Figure 6 : Jupyter notebook

2.5 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

- Variables are nothing but reserved memory locations to store values.

- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

- Python has five standard data types –

  - numbers
  - Lists
  - Tuples
  - Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you

assign a value to them.

- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the

quotation marks.

- Python allows for either pairs of single or double quotes.

- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes

starting at 0 in the beginning of the string and working their way from -1 at the end.

- The plus (+) sign is the string concatenation operator and the asterisk (*) is the

repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.

- A list contains items separated by commas and enclosed within square brackets

([]).

- To some extent, lists are similar to arrays in C. One difference between them is that

all the items belonging to a list can be of different data type.

- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with

indexes starting at 0 in the beginning of the list and working their way to end -1.

- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the

repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.

- A tuple consists of a number of values separated by commas. Unlike lists, however,

tuples are enclosed within parentheses.

- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in

parentheses (( )) and cannot be updated.

- Tuples can be thought of as read-only lists.

- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a

tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays

or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings.

Values, on the other hand, can be any arbitrary Python object.

- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses.

You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or

directly from the Python prompt.

PYTHON USING OOP's CONCEPTS:

Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class.

  The attributes are data members (class

variables and instance variables) and methods, accessed via dot notation.

- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are

not used as frequently as instance variables are.

- Data member: A class variable or instance variable that holds data associated with

a class and its objects.

- Instance variable: A variable that is defined inside a method and belongs only to

the current instance of a class.

- Defining a Class:

    o We define a class in a very similar way how we define a function.

    o Just like a function ,we use parentheses and a colon after the class

name(i.e. ():) when we define a class. Similarly, the body of

our class is indented like a functions body is.

```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Figure 7 : Defining a Class

_____init____method in Class:

- The init method — also called a constructor — is a special method that runs when

an instance is created so we can perform any tasks to set up the instance.

- The init method has a special name that starts and ends with two

underscores:_init_().

CHAPTER 3:

CASE STUDY

3.1 PROBLEM STATEMENT:

Our goal is to train a custom deep learning model to detect whether a person is wearing a mask or is not wearing a mask.

3.2 DATASET

The given data set has the following Parameters

    1) Observations
    a) Experiments
       i)   Data
           (1) With mask
           (2) Without mask
       ii) Dest Folder
           (1) Test
               (a) With mask
               (b) Without mask
           (2) Train
               (a) With mask
               (b) Without mask
           (3) Validation
               (a) With mask
               (b) Without mask
               (c) Test.csv
               (d) Train.csv
    b) Mask classifier
    c) Readme

CHAPTER 4:MODEL BUILDING

4.1.1 IMPORTING THE LIBRARIES:

Problem statement

Our goal is to train a custom deep learning model to detect whether a person is wearing a mask or is not wearing a mask.

Importing libraries

```
[ ]  import os
     import matplotlib.pyplot as plt
```

loading dataset

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
```

```
↳  Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

First we need to download the data set in our drive and after that we need to import the libraries.

In the next step we should load the dataset from our google colab and mount the drive.

4.1.2 READING DIRECTORIES :

```
[ ]  pwd
```

```
↳  '/content'
```

```
[ ]  !ls '/content/drive/My Drive/2020/Face Mask Detection Dataset'
```

```
↳  with_mask  without_mask
```

```
[ ]  print(len(os.listdir('/content/drive/My Drive/2020/Face Mask Detection Dataset/with_mask')))
```

```
↳  690
```

```
[ ]  print(len(os.listdir('/content/drive/My Drive/2020/Face Mask Detection Dataset/without_mask')))
```

```
↳  686
```

here, we need to check using pwd.After that ls to know the path of the drive in which our dataset in there

in the drive.

There are two dataset 1. with mask 2. without mask

we need to print the no.of images that are present in the with mask dataset

then no.of images in the without mask dataset

4.1.3 GIVING FILE NAMES :

```
[ ]  base_dir = "/content/drive/My Drive/2020"
     train_dir=os.path.join(base_dir,'Face Mask Detection Dataset')
     mask_dir = os.path.join(train_dir,'with_mask')
     withoutmask_dir = os.path.join(train_dir,'without_mask')
```

```
[ ]  data_mask=os.listdir(mask_dir)
     data_mask[:5]
```

```
☐→  ['317-with-mask.jpg',
      'augmented_image_76.jpg',
      '173-with-mask.jpg',
      '163-with-mask.jpg',
      '467-with-mask.jpg']
```

```
[ ]  data_nomask=os.listdir(withoutmask_dir)
     data_nomask[:5]
```

```
☐→  ['34.jpg', '264.jpg', 'augmented_image_314.jpg', '59.jpg', '128.jpg']
```

Here, we need to do the filenames using train_dir, train_dir,mask_dir and without_mask_dir to know the path which is there in our drive.

after that we need to know the images which are there in our dataset by using data_mask and data_nomask.

I want to print 5 images so i gave 5 and i got 5 images as .jpg for both the datasets

4.1.4 Creating Train and validation data from Folder:

▾ Train and validation data

```
[ ]  ocessing.image import ImageDataGenerator


     Generator(rescale=1./255, validation_split=0.2).flow_from_directory(train_dir, class_mode="binary",subset="training", batch_s

     nerator(rescale=1./255, validation_split=0.2).flow_from_directory(train_dir, class_mode="binary",subset="validation", batch_s
```

```
☐→  Found 1101 images belonging to 2 classes.
     Found 275 images belonging to 2 classes.
```

```
[ ]  train_generator
```

```
☐→  <keras_preprocessing.image.directory_iterator.DirectoryIterator at 0x7f161ccd4cf8>
```

Here we need to split the train and validation dataset to find the images from the classes.

we need to import the Imagedatagenerator and split the data
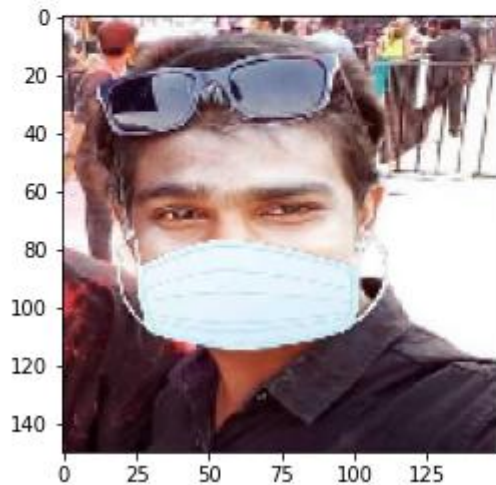
4.1.5 : DISPLAY THE IMAGES:

Displaying the single image of person wearing mask and a person not wearing mask

Here we imported matplotlib as plt . Displayed the image using plot and the imshow key word .

```
[ ]  imgs,labels = train_generator.next()
     print(imgs.shape)
     print(labels.shape)
     plt.imshow(imgs[0,:,:,:])
```
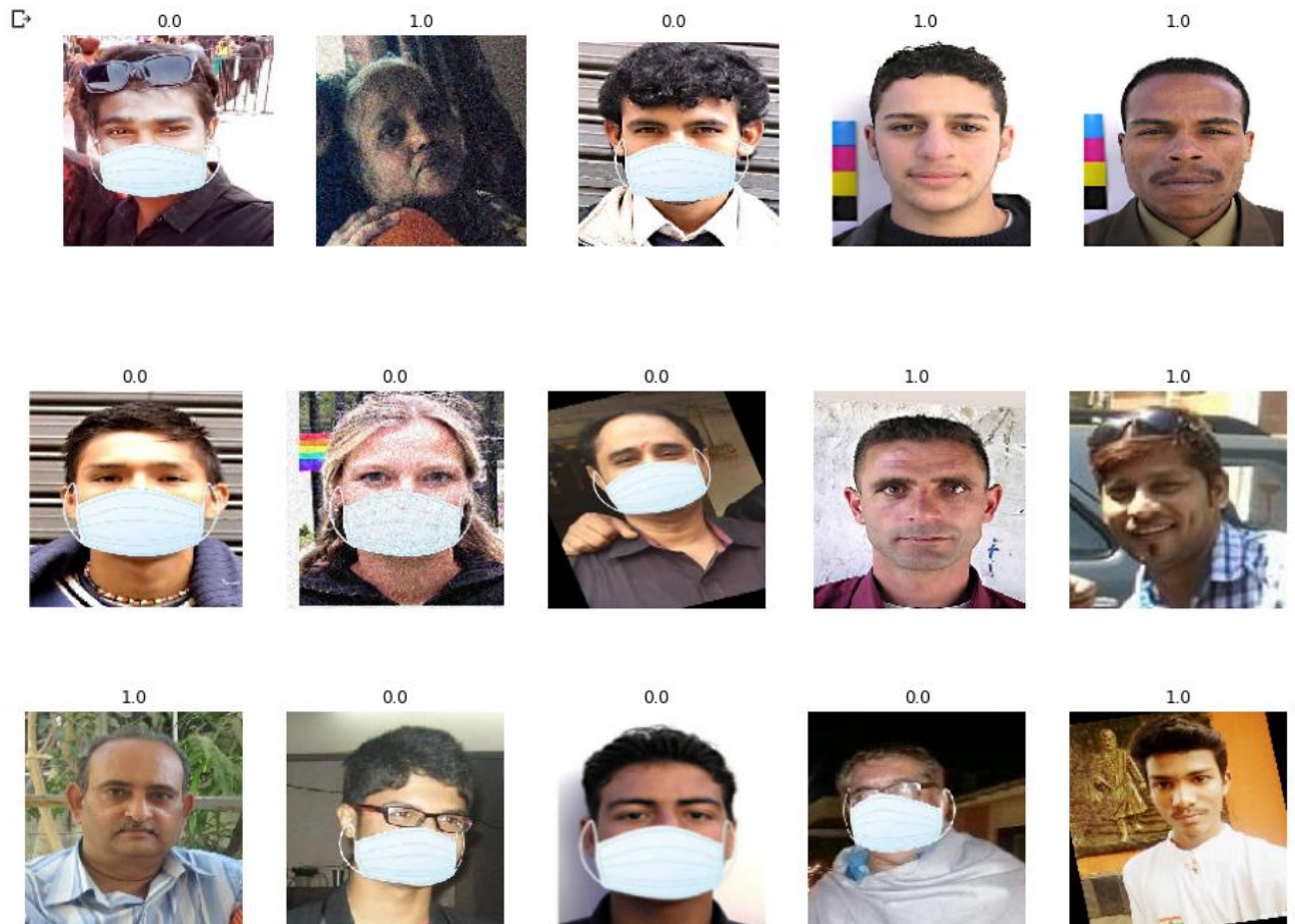
```
(128, 150, 150, 3)
(128,)
<matplotlib.image.AxesImage at 0x7f15dc8af780>
```


In this step we need to print the image from the train generator

images and labels. To show the image we need to write plt

Display set of images

```python
plt.figure(figsize=(16,16))
pos=1 ##plot position
for i in range(20):
    plt.subplot(4,5,pos)
    plt.imshow(imgs[i,:,:,:])# To display the image
    plt.title(labels[i])
    plt.axis('off')
    pos+=1
```

```
import matplotlib.pyplot as plt
imgs,labels = train_generator.next()
plt.figure(figsize=(16,16))
pos = 1  ## plot position
for i in range(10):
  plt.subplot(5,2,pos)
  plt.hist(imgs[i,:,:,:].flat) # To display the histogram
  plt.title(labels[i])
  pos += 1
```


we need to display the images both with and without mask images. by using all the labels,shapes and sizes.

After that the histogram of an image normally refers to a histogram of the pixel intensity values. The histogram is a graph showing the number of pixels in an image at each different intensity value found in that image.

It is basically a graphical representation of the data which is the histogram.

4.2. HISTOGRAM OF DATA:
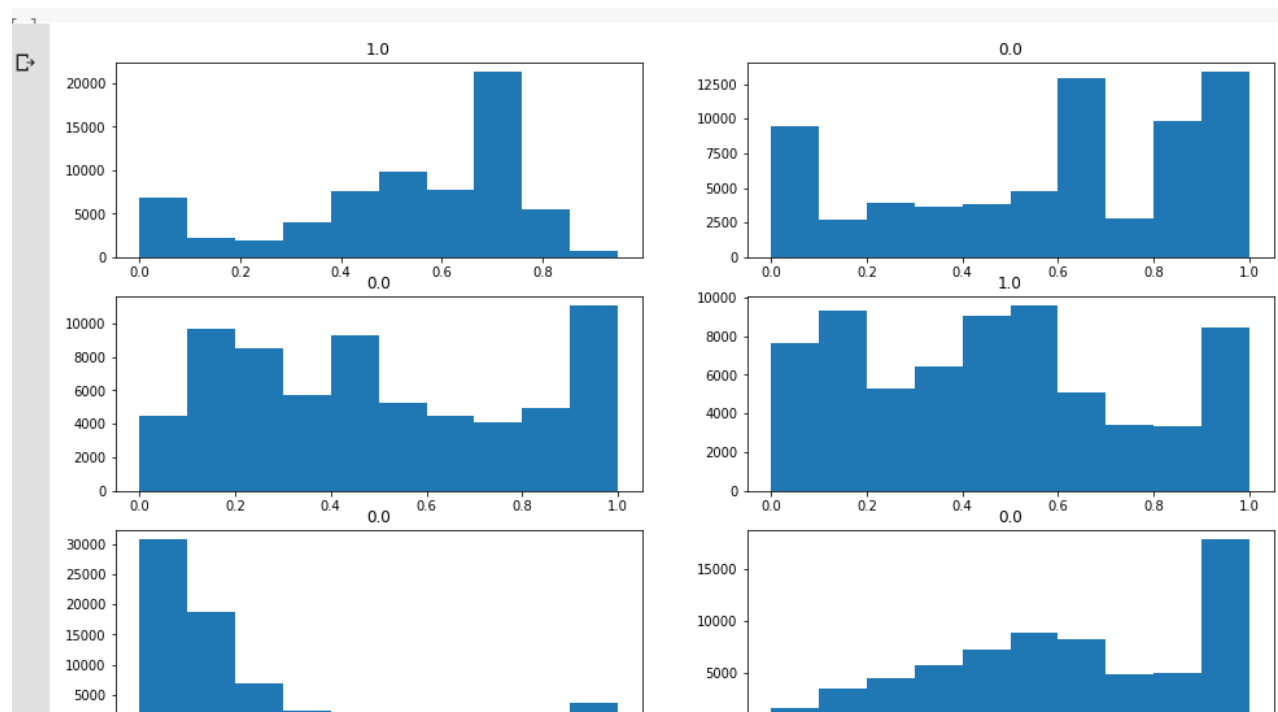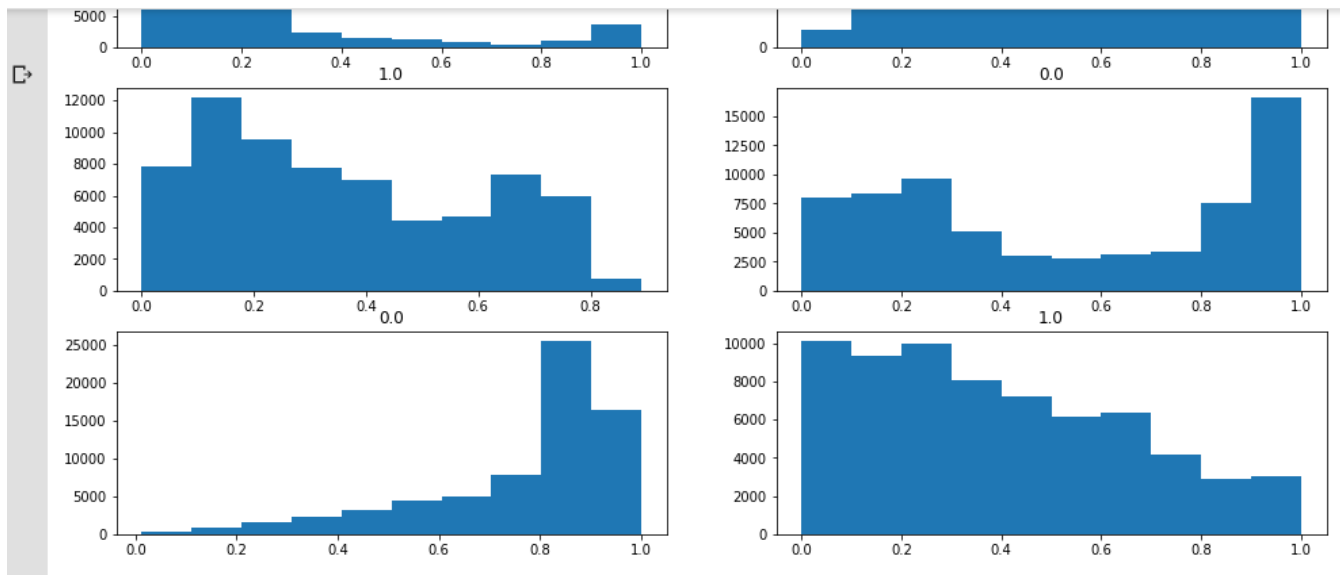
4.3 BUILDING THE MODEL:

4.3.1 IMPORTING REQUIRED LIBRARIES:

▾ building a model

```
[ ]  ## import required methods
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Conv2D,Dense,Flatten,MaxPooling2D
```

```
[ ]  model = Sequential()
     ## add a conv layer folloed by maxpooling
     model.add(Conv2D(16,3,activation='relu',input_shape=(150,150,3)))
     model.add(MaxPooling2D(2))
     ## add a conv layer folloed by maxpooling
     model.add(Conv2D(32,3,activation='relu'))
     model.add(MaxPooling2D(2))
     ## add a conv layer folloed by maxpooling
     model.add(Conv2D(64,3,activation='relu'))
     model.add(MaxPooling2D(2))
     # Convert the faeturemap into 1D array
     model.add(Flatten())
     # Fully connected layer with 512 neurons
     model.add(Dense(512,activation='relu'))
     ## Final output layer
     model.add(Dense(1,activation='sigmoid'))

     ## let us see the the summary
     model.summary()
```

Model: "sequential"

4.3.2 MODEL:

```
Model: "sequential"
_____
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 16)      448
_____
max_pooling2d (MaxPooling2D) (None, 74, 74, 16)        0
_____
conv2d_1 (Conv2D)            (None, 72, 72, 32)        4640
_____
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_____
conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496
_____
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 64)        0
_____
flatten (Flatten)            (None, 18496)             0
_____
dense (Dense)                (None, 512)               9470464
_____
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 9,494,561
Trainable params: 9,494,561
Non-trainable params: 0
_____
```

Next we need to build the model, we will build the model from the dataset

4.3.3 COMPILING THE MODEL:

```python
### Compiling the modle
import tensorflow as tf
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),metrics=['accuracy'])
```

Then compiling the model from the building the model

· Training model

```python
history = model.fit(train_generator,epochs=15,validation_data=val_generator,batch_size=32)
```

```
Epoch 1/15
9/9 [==============================] - 23s 3s/step - loss: 2.1258 - accuracy: 0.5531 - val_loss: 0.5575 - val_accuracy: 0.7455
Epoch 2/15
9/9 [==============================] - 22s 2s/step - loss: 0.6699 - accuracy: 0.7103 - val_loss: 1.3511 - val_accuracy: 0.5018
Epoch 3/15
9/9 [==============================] - 23s 3s/step - loss: 0.7483 - accuracy: 0.7275 - val_loss: 0.5155 - val_accuracy: 0.6800
Epoch 4/15
9/9 [==============================] - 24s 3s/step - loss: 0.5446 - accuracy: 0.7965 - val_loss: 0.3876 - val_accuracy: 0.9200
Epoch 5/15
9/9 [==============================] - 23s 3s/step - loss: 0.2866 - accuracy: 0.9237 - val_loss: 0.1889 - val_accuracy: 0.9418
Epoch 6/15
9/9 [==============================] - 23s 3s/step - loss: 0.1222 - accuracy: 0.9609 - val_loss: 0.1586 - val_accuracy: 0.9418
Epoch 7/15
9/9 [==============================] - 26s 3s/step - loss: 0.3934 - accuracy: 0.8728 - val_loss: 0.1462 - val_accuracy: 0.9455
Epoch 8/15
9/9 [==============================] - 22s 2s/step - loss: 0.0905 - accuracy: 0.9691 - val_loss: 0.1045 - val_accuracy: 0.9527
Epoch 9/15
9/9 [==============================] - 22s 2s/step - loss: 0.2978 - accuracy: 0.9146 - val_loss: 0.1577 - val_accuracy: 0.9527
Epoch 10/15
9/9 [==============================] - 23s 3s/step - loss: 0.0773 - accuracy: 0.9791 - val_loss: 0.0959 - val_accuracy: 0.9709
Epoch 11/15
9/9 [==============================] - 23s 3s/step - loss: 0.0388 - accuracy: 0.9927 - val_loss: 0.0799 - val_accuracy: 0.9745
Epoch 12/15
9/9 [==============================] - 23s 3s/step - loss: 0.0263 - accuracy: 0.9955 - val_loss: 0.0968 - val_accuracy: 0.9709
```

```
Epoch 12/15
9/9 [==============================] - 23s 3s/step - loss: 0.0263 - accuracy: 0.9955 - val_loss: 0.0968 - val_accuracy: 0.9709
Epoch 13/15
9/9 [==============================] - 23s 3s/step - loss: 0.2472 - accuracy: 0.9192 - val_loss: 0.0789 - val_accuracy: 0.9745
Epoch 14/15
9/9 [==============================] - 22s 2s/step - loss: 0.0261 - accuracy: 0.9964 - val_loss: 0.0716 - val_accuracy: 0.9782
Epoch 15/15
9/9 [==============================] - 23s 3s/step - loss: 0.0160 - accuracy: 0.9955 - val_loss: 0.0774 - val_accuracy: 0.9782
```

here we will train the model that is by epoch..It is basically to measure the accuracy of the model. It is called train or test because we split the dataset into two sets. we train the model using training set
 15th epoch the loss value has decreased to 0.01 and val loss to 0.016 and the accuracy is 0.99 and val accuracy 0.97.

The overall accuracy of the model is 97% which is a best fit model.

```
[19] train_acc = history.history['accuracy']
     val_acc = history.history['val_accuracy']
     train_loss = history.history['loss']
     val_loss = history.history['val_loss']

     epochs = list(range(1,16))
     plt.figure(figsize=(16,4))
     plt.subplot(1,2,1)
     plt.plot(epochs,train_acc,label='train_acc')
     plt.plot(epochs,val_acc,label='val_acc')
     plt.title('accuracy')
     plt.legend()
     plt.subplot(1,2,2)
     plt.plot(epochs,train_loss,label='train_loss')
     plt.plot(epochs,val_loss,label='val_loss')
     plt.title('loss')
     plt.legend()
```
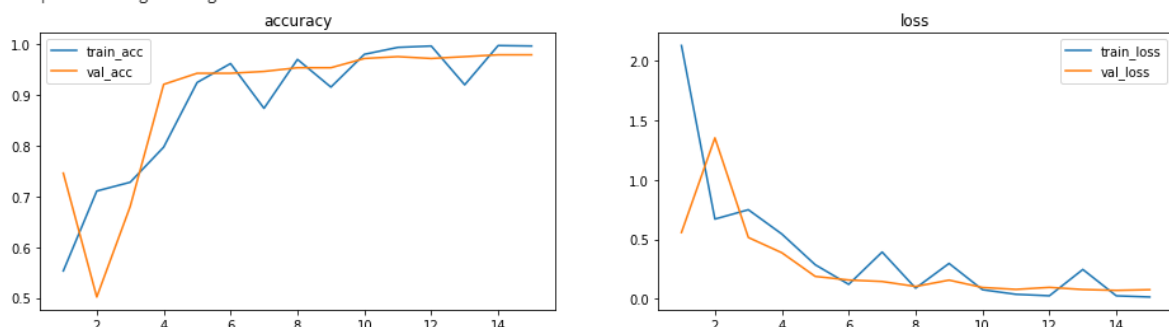
<matplotlib.legend.Legend at 0x7f15d876d9e8>

<matplotlib.legend.Legend at 0x7f15d876d9e8>
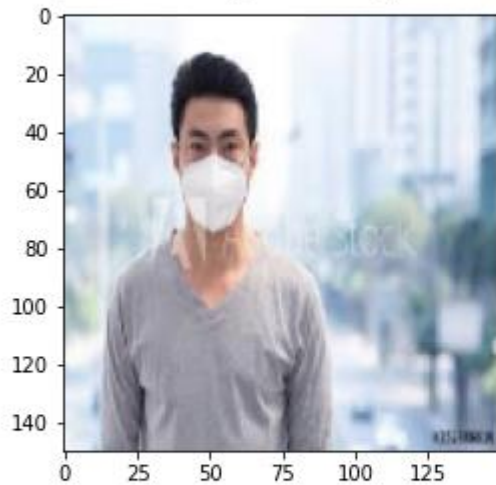

training accuracy is usually the accuracy you get if you apply the model on training data,

4.4 PREDICTING THE IMAGE

Predicted image with mask

```
[20] from tensorflow.keras.preprocessing import image
     import numpy as np
     img = image.load_img('/tmp/mask.jpg')
     print(type(img))
     #print(img.shape)
     img = tf.keras.preprocessing.image.img_to_array(img)
     print(img.shape)
     print(type(img))
     img = tf.image.resize(img,(150,150))
     ## Scaling
     img = img/255
     print(img.shape)
     img = np.expand_dims(img,axis=0)
     print(img.shape)
     plt.imshow(img[0,:,:,:])
```

```
<class 'PIL.JpegImagePlugin.JpegImageFile'>
(183, 275, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7f15ce522e48>
```

```
<class 'PIL.JpegImagePlugin.JpegImageFile'>
(183, 275, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7f15ce522e48>
```



```
21] model.predict(img)
```

```
array([[0.91225547]], dtype=float32)
```

Here we need to predict the images by both with and without mask

we need to download two images one is with mask and other is without mask.

then we need to print the image after that we need to predict the image by using model
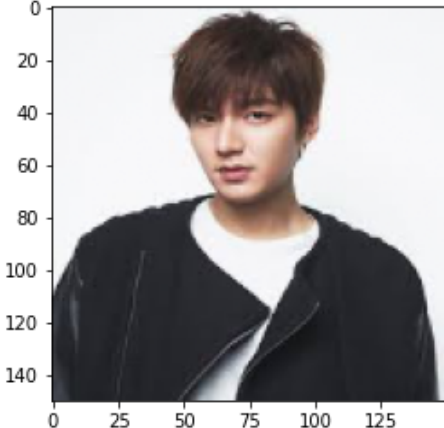
4.5 PREDICTING WITHOUT MASK

Predicting image without mask

```
[25] from tensorflow.keras.preprocessing import image
     import numpy as np
     img = image.load_img('/tmp/without_mask.jpg')
     print(type(img))
     #print(img.shape)
     img = tf.keras.preprocessing.image.img_to_array(img)
     print(img.shape)
     print(type(img))
     img = tf.image.resize(img,(150,150))
     ## Scaling
     img = img/255
     print(img.shape)
     img = np.expand_dims(img,axis=0)
     print(img.shape)
     plt.imshow(img[0,:,:,:])
```

```
<class 'PIL.JpegImagePlugin.JpegImageFile'>
(236, 214, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7f15ce3cdb38>
  0
```

[25]
```
<class 'PIL.JpegImagePlugin.JpegImageFile'>
(236, 214, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7f15ce3cdb38>
```



```
[26] model.predict(img)
```

```
array([[0.00057225]], dtype=float32)
```

here also we need to do same as the previous cell an image without mask and predict the image

REFERENCES:

TENSOR FLOW LINK:

https://www.tensorflow.org/install/errors

MACHINE LEARNING LINK:

https://en.wikipedia.org/wiki/Machine_learning

DATA SET LINK:

https://drive.google.com/drive/folders/1tkIxVB6z6-TBlVEkp_TQaavLVU-QdnlM?usp=sharing