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

SAHITHI. PASUPULETI

221710310060

Under the Guidance of

Mr. M. Venkateswarlu

Assistant Professor

GITAM (DEEMED TO BE UNIVERSITY)

VISAKHAPATNAM HYDERABAD BENGALURU

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 Sahithi. Pasupuleti

Date: 221710310060

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 Sahithi.Pasupuleti (221710310060) 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 2019-20

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 CSE	Department of CSE

PROMIZE
IT SERVICES

Promize IT Services
Flat No. 303, Paradise Valley, Road NO.3,
Madhavapuri Hills, Chanda Nagar,
Ameenpur village,
Hyderabad, Telangana, India - 502032

Date: 16th June 2019

CERTIFICATE

This is to certify that the Internship titled **"Predicting amount of Purchase using Multiple Linear Regression"** is the bona fide work carried out by [redacted] student of the **GITAM University, Hyderabad,** in partial fulfillment for the award of **Bachelor of Technology** in Electronics and Communication Engineering during the period 29th April 2019 – 15th June 2019 at **PROMIZE IT SERVICES PRIVATE LIMITED – HYDERABAD.** During this period his conduct was found to be very good and he has shown good technical skills.

(Dinesh Kumar Jooshetti)
Project Head
Promize IT Services Pvt Ltd.

PROMIZE IT SERVICES PRIVATE LIMITED ★ HYDERABAD

(Suresh Shankar)
Director of Operations
Promize IT Services Pvt Ltd.

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 Computer Science and 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.

Sahithi . Pasupuleti
221710310060

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.

1.4 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 by 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.

1.4.2 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.

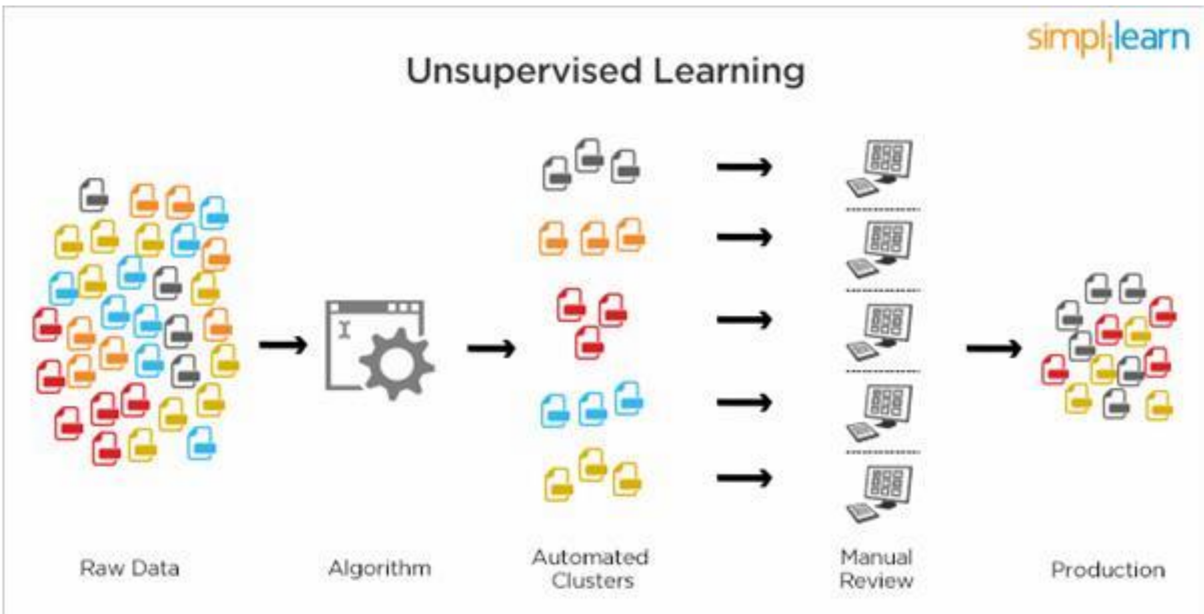


Figure 2 : Unsupervised Learning

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.

1.4.3 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.

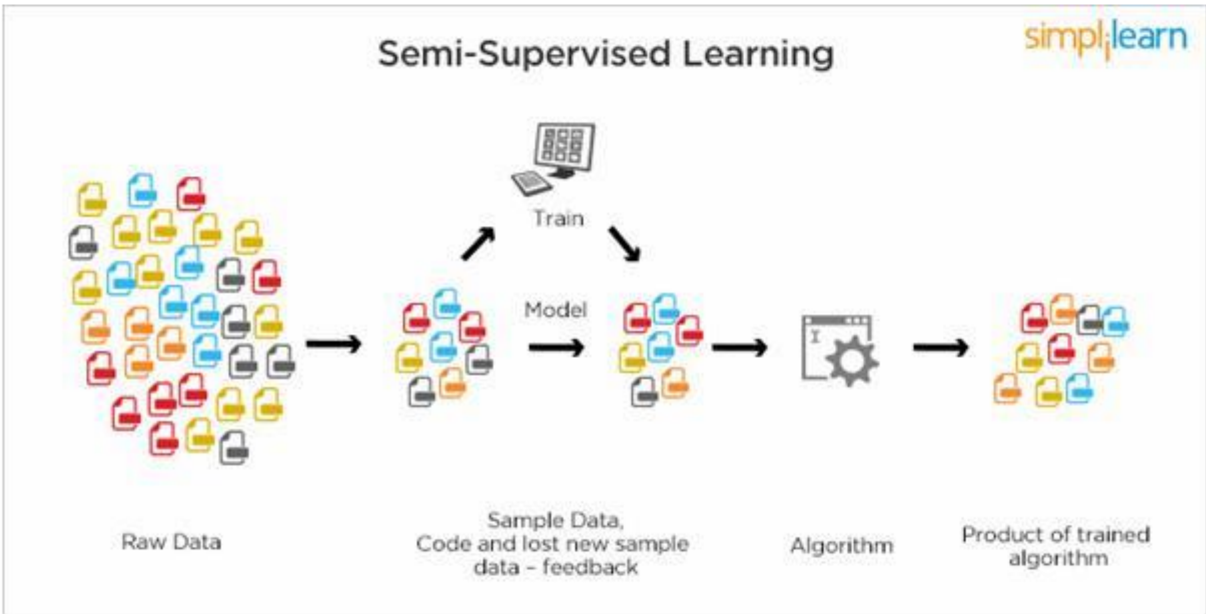


Figure 3 : Semi Supervised Learning

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.

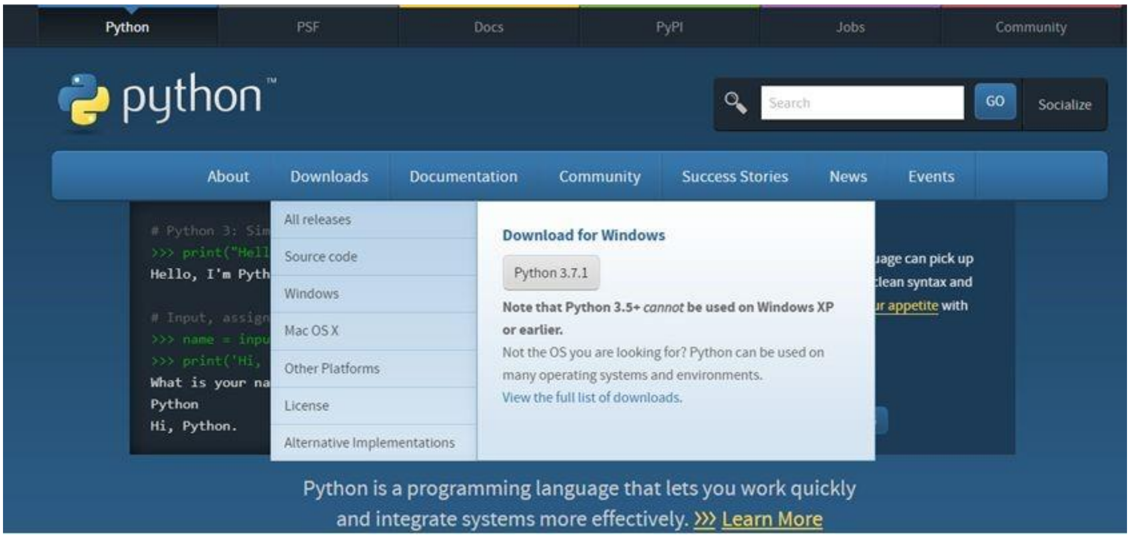


Figure 4 : Python download

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)

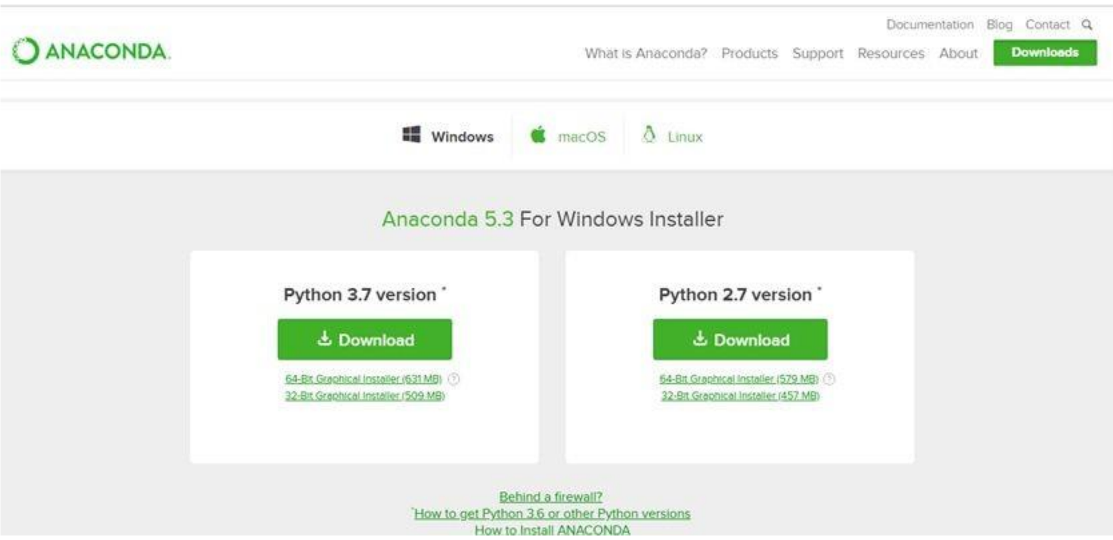


Figure 5 : Anaconda download

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:

2.6.1 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.

2.6.2 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.

2.7 PYTHON USING OOP's CONCEPTS:

2.7.1 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:

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

  - 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.

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

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

Figure 7 : Defining a Class

2.7.2 __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:

Now-a-days due to COVID-19(Corona Virus) every person must wear a mask.

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

Data Set Link: https://github.com/prajnasb/observations

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

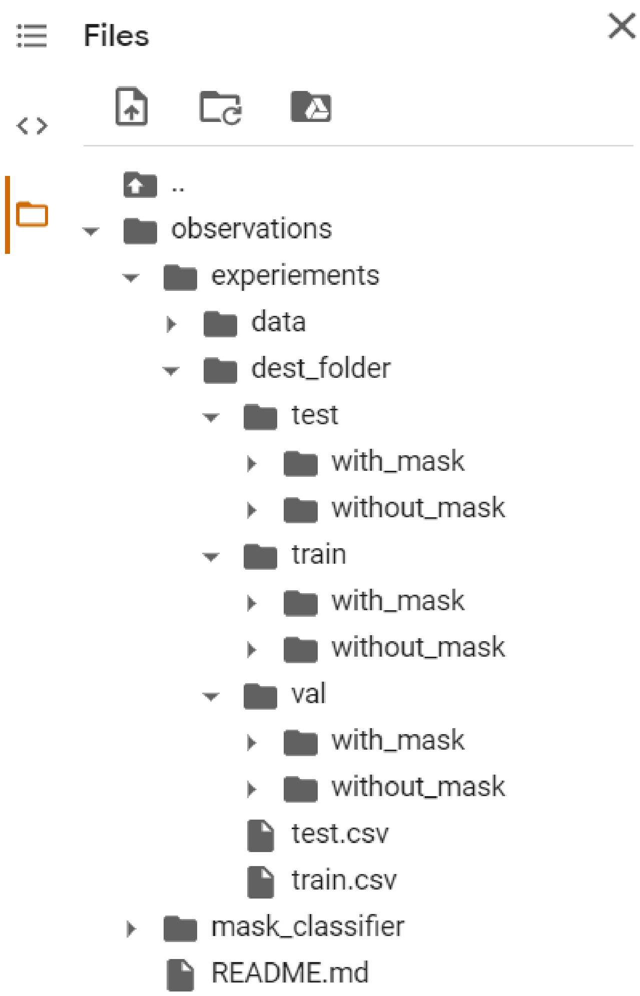


Figure 8: Dataset

3.3 OBJECTIVE OF CASE STUDY

To get a better understanding and chalking out a plan of solution of the client, we have adapted the view point of looking at product categories and for further deep understanding of the problem, we have considered all the factors of training data, testing data and validation data, which has images of with mask and without mask. The main objective of this case study was to look up the factors of the data set and its classification and draft outcome report of mask classification

CHAPTER 4

MODEL BUILDING

4.1 VISUALIZATION OF DATA:

4.1.1 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the algorithm.

We are importing os for reading the directories and matplotlib for plotting the graphs and images .

\+ Code + Text

▾ Importing the libraries

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

Figure 9: Importing libraries

4.1.2 LOADING THE DATA SET:

The data is loaded by git clone command which is inbuilt in google colaboratory , so the data is imported from github directly .

It reads all the data and stores in the content directory , it loads all the images and files.

▾ Loading the DataSet

```
[ ]  !git clone https://github.com/prajnasb/observations.git
```

```
Cloning into 'observations'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 1638 (delta 9), reused 0 (delta 0), pack-reused 1604
Receiving objects: 100% (1638/1638), 75.94 MiB | 39.13 MiB/s, done.
Resolving deltas: 100% (20/20), done.
```

Fig 10: Loading Data set

4.1.3 READING DIRECTORIES :

The data is stored in content directory so need to read the directory.

Reading directories

```
[44] pwd
'/content'

[45] os.listdir("/content")
['.config', 'observations', 'sample_data']

[46] os.listdir("/content/observations")
['README.md', 'mask_classifier', 'experiements', '.git']

[47] os.listdir("/content/observations/experiements")
['data', 'dest_folder']

[48] os.listdir("/content/observations/experiements/dest_folder")
['train.csv', 'test.csv', 'val', 'test', 'train']

[49] os.listdir("/content/observations/experiements/dest_folder/train")
['without_mask', 'with_mask']
```

Fig 11: reading directories

```
[50] os.listdir("/content/observations/experiements/dest_folder/test")
['without_mask', 'with_mask']

[51] os.listdir("/content/observations/experiements/dest_folder/val")
['without_mask', 'with_mask']
```

Fig 12: Reading Directories

Using the command os.listdir we are reading the directories, at first it in content directory , and there are 3 parts config, observations , sample data.

Our data set is named as Observations , so now we are reading the observations directory to know the folders and directories present in it.

On reading observations we got mask_classifier ,experiments, read me and git folders , then we are reading experiments directory , in that we found data and dest_folder.

In dest folder we have train, test, val folders/directories which has the images of with mask and without mask for training, testing and validating the data which are split from the original image data which is in data directory.

4.1.4 LENGTH OF DATA:

No.of images used in the model , no.of images with mask, no.of images with out mask.

Length of dataset(No.of images)

```
[52] #With Mask
     print(len(os.listdir("/content/observations/experiements/data/with_mask")))

     690

[53] #Without Mask
     print(len(os.listdir("/content/observations/experiements/data/without_mask")))

     686
```

Fig 13: Length of Data

In data directory we have 2 folders with mask and without mask , using len and print function we are printing the no.of images of with mask and without mask folders which are 690 images of with mask and 686 images of without mask.

4.1.5 : LOADING THE DATA FROM DIRECTORIES:

The dest folder has all the data of training , testing and validation in separate directories , so first we assign base_dir to dest folder using os.path.join which joins/assigns the dest folder path to base_dir .

Now creating train_dir and using base_dir as main path we are assigning the train folder to train_dir using os.path.join.

In the same way we are assigning the test folder to test_dir and val folder to validation_dir.

Next we are assigning the withmask folder of train directory to train_with_mask_dir using os.path.join and train_dir as main directory, similarly we also assigned without mask folder to train_without_mask_dir.

Next we are assigning the withmask folder of test directory to test_with_mask_dir using os.path.join and test_dir as main directory, similarly we also assigned without mask folder to test_without_mask_dir.

Next we are assigning the withmask folder of validation directory to val_with_mask_dir using os.path.join and validation_dir as main directory, similarly we also assigned without mask folder to val_without_mask_dir.

```
▾ Filename

▾ Loading Data

[54] base_dir = "/content/observations/experiements/dest_folder"
     train_dir = os.path.join(base_dir,'train')
     test_dir = os.path.join(base_dir,'test')
     validation_dir = os.path.join(base_dir,'val')
     train_with_mask_dir = os.path.join(train_dir,'with_mask')
     train_without_mask_dir = os.path.join(train_dir,'without_mask')
     test_with_mask_dir = os.path.join(test_dir,'with_mask')
     test_without_mask_dir = os.path.join(test_dir,'without_mask')
     val_with_mask_dir = os.path.join(validation_dir,'with_mask')
     val_without_mask_dir = os.path.join(validation_dir,'without_mask')
```

Fig 14: Loading Data

4.1.6 GIVING FILE NAMES :

```
5] #Filenames
   with_mask_filename = os.listdir(train_with_mask_dir)
   with_mask_filename[:4]

   ['416-with-mask.jpg',
    'augmented_image_90.jpg',
    '76-with-mask.jpg',
    '125-with-mask.jpg']

6] without_mask_filename = os.listdir(train_without_mask_dir)
   without_mask_filename[:10]

   ['370.jpg',
    'depositphotos-142113624-stock-photo-smiling-indian-man-face.jpg',
    'augmented_image_90.jpg',
    '398.jpg',
    'augmented_image_202.jpg',
    '52.jpg',
    '160.jpg',
    '87.jpg',
    '328.jpg',
    'augmented_image_156.jpg']
```

Fig 15: Giving File Names

In the with_mask_filename , we are reading the data from the directory train_with_mask_dir using the os.listdir() , this command reads all the data in the directory and assigns it and then printing the names of with mask images by calling with mask filename , [:4] is given for count of names needed to print.

In the without_mask_filename , we are reading the data from the directory train_without_mask_dir using the os.listdir() , this command reads all the data in the directory and assigns it and then printing the names of with mask images by calling without mask filename , [:10] is given for count of names needed to print.

4.1.7 : 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 . From the train data which is train_with_mask_dir , images is read using imread key word and displayed.

▾ Display image with mask

```
[57] import matplotlib.pyplot as plt
     plt.imshow(plt.imread(train_with_mask_dir+'/augmented_image_108.jpg'))
```

```
<matplotlib.image.AxesImage at 0x7fa236ae8ba8>
```

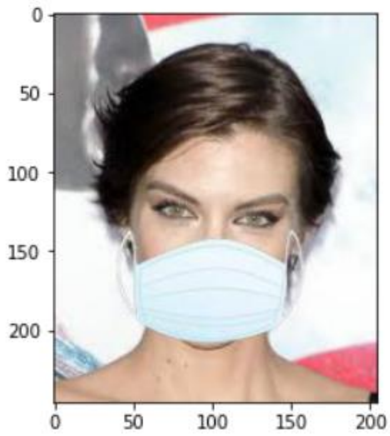


Fig 16: Display image with mask

Display image without mask

```
[58] plt.imshow(plt.imread(train_without_mask_dir+'/435.jpg'))
```

```
<matplotlib.image.AxesImage at 0x7fa236df0828>
```

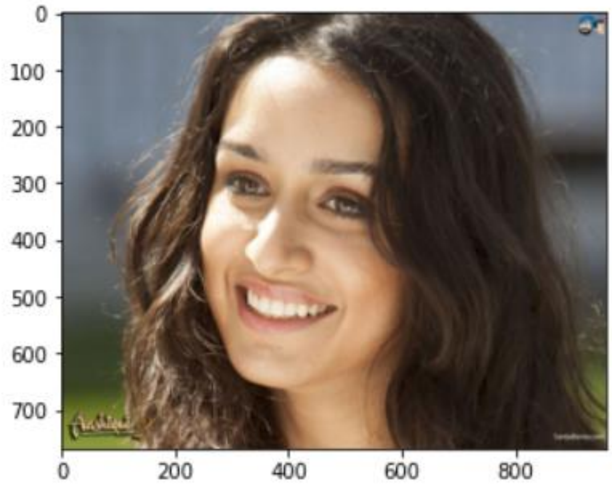


Fig 17: Display image without mask

Display of set of images with mask

```
[ ] plt.figure(figsize=(16,16))
    j = 1
    for i in range(10):
      img = plt.imread(os.path.join(train_with_mask_dir,with_mask_filename[i]))
      plt.subplot(4,5,j)
      plt.imshow(img)
      plt.title(img.shape)
      plt.axis('off')
      j += 1
```

Fig 18: Set of images with mask

Display of set of images without mask

```
[ ] plt.figure(figsize=(16,16))
    j = 1
    for i in range(8):
      img = plt.imread(os.path.join(train_without_mask_dir,without_mask_filename[i]))
      plt.subplot(4,4,j)
      plt.imshow(img)
      plt.title(img.shape)
      plt.axis('off')
      j += 1
```

Fig 19: Set of images without mask

The figure size is given 16 . range is for the no .of images need to be printed .

Using subplot we can print multiple images and no. of rows and columns given in brackets . Title is the shape of image.

4.2 DATA PREPROCESSING :

4.2.1 Creating Train and validation data from Folder:

Creating Train and valiadtion data from Folder

```
[ ]  from tensorflow.keras.preprocessing.image import ImageDataGenerator

     # All images will be rescaled by 1./255
     train_datagen = ImageDataGenerator(rescale=1./255)
     val_datagen = ImageDataGenerator(rescale=1./255)

     # Flow training images in batches of 20 using train_datagen generator
     train_generator = train_datagen.flow_from_directory(
             train_dir,  # This is the source directory for training images
             target_size=(150, 150),  # All images will be resized to 150x150
             batch_size=20,
             # Since we use binary_crossentropy loss, we need binary labels
             class_mode='binary')

     # Flow validation images in batches of 20 using val_datagen generator
     validation_generator = val_datagen.flow_from_directory(
             validation_dir,
             target_size=(150, 150),
             batch_size=20,
             class_mode='binary')

     Found 1315 images belonging to 2 classes.
     Found 142 images belonging to 2 classes.
```

Fig 20: Train and Validation data

We are importing the libraries required which are tenserflow, keras, and from them we are importing image data generator.

Using Imagedatagenerator we are scaling the image using the rescale key word.

Since 255 is the maximin pixel value. Rescale 1./255 is to transform every pixel value from range [0,255] -> [0,1]. We are rescaling all the images of train and validation and storing them in train_datagen and val_datagen.

Now we are dividing the images into 20 batches and each batch size is 150x150 using the class mode as binary , the class mode is binary because we have 2 classes for our image data which are with mask and without mask

These divided batches of train data images will be stored in train_generator and validation data images in validation_generator.

Then we got output as 1315 images of 2 classes in train generator and 142 images of 2 classes in validation generator.

```
[ ] train_generator
```

```
<keras_preprocessing.image.directory_iterator.DirectoryIterator at 0x7f6bb99eef98>
```

```
[ ] imgs,labels = train_generator.next()
    print(imgs.shape)
    print(labels.shape)
    plt.imshow(imgs[0,:,:,:])
```

```
(20, 150, 150, 3)
(20,)
<matplotlib.image.AxesImage at 0x7f6b772ad198>
```

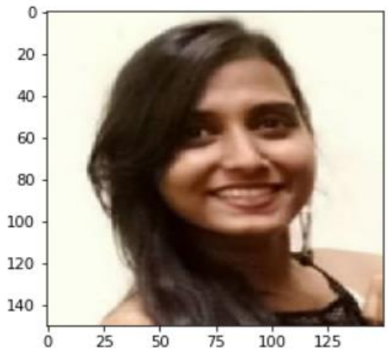


Fig 21: display image using train generator

We are checking whether the train generator has got all the rescaled values of training data and using that we are displaying a image

We have assigned train generator to imgs and labels .

4.2.2 DISPLAY OF RANDOM IMAGES:

▾ Displaying random images of with mask and without mask

```
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
```

Fig 22: Random images

Printing random images of people with mask and without mask using the train generator which is assigned to imgs and plotted the images using subplot and imshow().

4.2.3 HISTOGRAM OF DATA:

▾ Histogram of dataset

```
[ ]  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
```

Fig 23: code of histogram

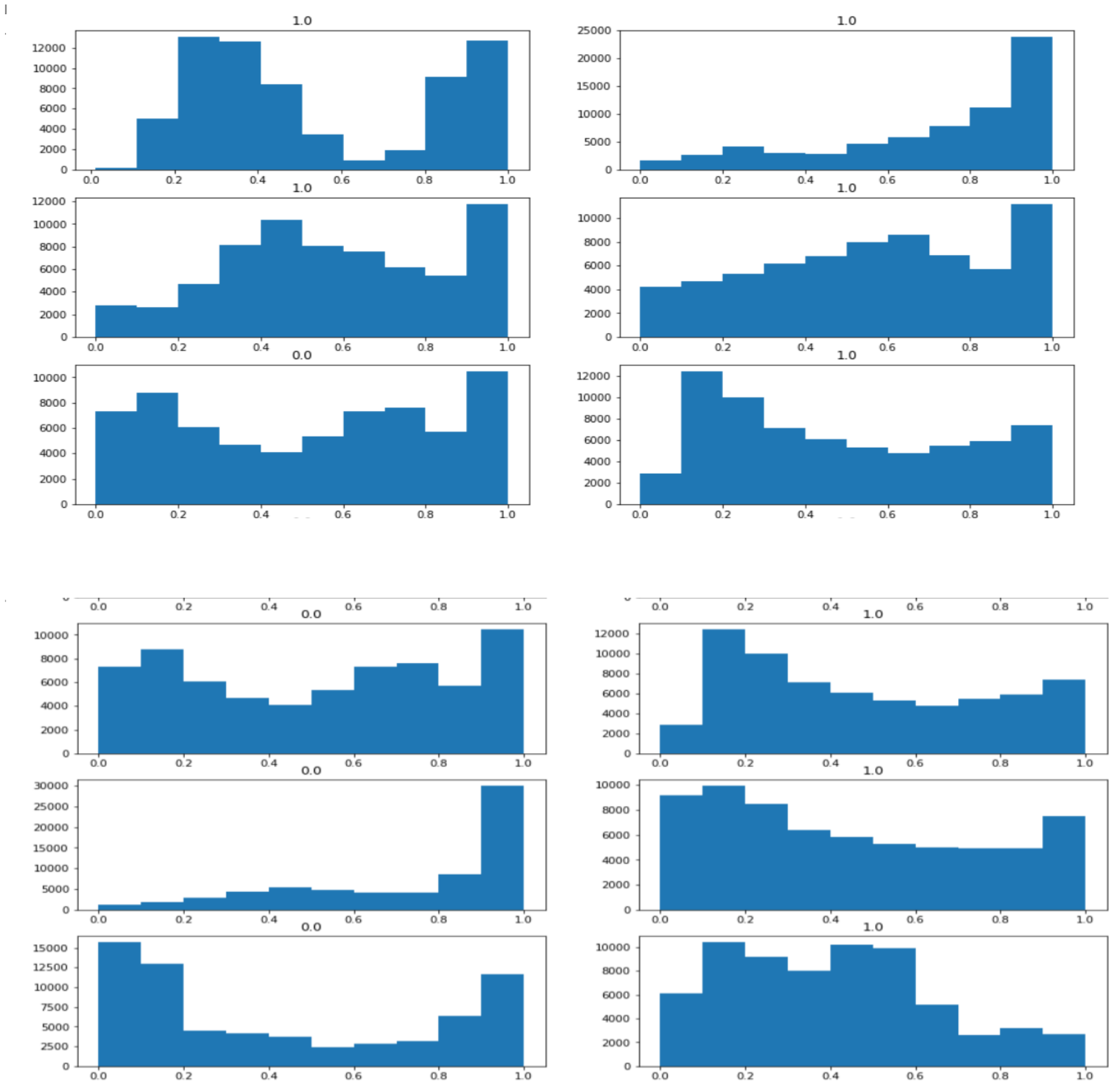


Fig 24: output of histogram

Using plt.hist command we have printed the histogram of different images of train data by using train generator.

4.3 BUILDING THE MODEL:

4.3.1 IMPORTING REQUIRED LIBRARIES:

▾ Build a model

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

Fig 25: Importing libraries

The methods Sequential, Convo2D, Dense,Flatten,MaxPooling2D are imported from tensorflow and keras models.

4.3.2 MODEL:

```
] 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()
```

Fig 26: Code for Model.

We are invoking the sequential function into model and using the function we are adding the convo layers by maxpooling .

We are giving different sizes for different layers of convo using convo2D and activation as relu

The final output is given with dense 1 as we have only 2 classes and activation as sigmoid .

Finally calling summary for knowing the total parameters of model.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 148, 148, 16)      448
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 74, 74, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 72, 72, 32)        4640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               9470464
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 9,494,561
Trainable params: 9,494,561
Non-trainable params: 0
_________________________________________________________________
```

Fig 27: output for model.

The output has 9,494,561 parameters totally which are trainable.

4.3.3 COMPILING THE MODEL:

Compiling the model

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

Fig 28: Compiling the model.

The model is compiled using compile function and loss which uses binary cross entropy function and the metrics as accuracy.

4.3.4 TRAINING THE MODEL:

Train the Model

```
[ ] history = model.fit(train_generator,epochs=15,validation_data=validation_generator,batch_size=32)
```

Fig 29: Training the model.

Using model . fit function we are testing the accuracy of the model which will be best fit or under fit or over fit model .

We are using train generator and validation generator to take all the images of train and validation data

We are giving the epochs as 15 and the batch size as 32 which mean with size of 32 it divides into 15 batches .

```
Epoch 1/15
66/66 [==============================] - 35s 526ms/step - loss: 0.5196 - accuracy: 0.7840 - val_loss: 0.1156 - val_accuracy: 0.9648
Epoch 2/15
66/66 [==============================] - 34s 522ms/step - loss: 0.1858 - accuracy: 0.9308 - val_loss: 0.0605 - val_accuracy: 0.9718
Epoch 3/15
66/66 [==============================] - 34s 519ms/step - loss: 0.1075 - accuracy: 0.9605 - val_loss: 0.0439 - val_accuracy: 0.9859
Epoch 4/15
66/66 [==============================] - 34s 518ms/step - loss: 0.0649 - accuracy: 0.9741 - val_loss: 0.0468 - val_accuracy: 0.9789
Epoch 5/15
66/66 [==============================] - 34s 519ms/step - loss: 0.0467 - accuracy: 0.9810 - val_loss: 0.0586 - val_accuracy: 0.9718
Epoch 6/15
66/66 [==============================] - 34s 517ms/step - loss: 0.0561 - accuracy: 0.9833 - val_loss: 0.0320 - val_accuracy: 0.9859
Epoch 7/15
66/66 [==============================] - 34s 514ms/step - loss: 0.0344 - accuracy: 0.9871 - val_loss: 0.0754 - val_accuracy: 0.9789
Epoch 8/15
66/66 [==============================] - 34s 520ms/step - loss: 0.0710 - accuracy: 0.9878 - val_loss: 0.0575 - val_accuracy: 0.9789
Epoch 9/15
66/66 [==============================] - 34s 515ms/step - loss: 0.0245 - accuracy: 0.9939 - val_loss: 0.0269 - val_accuracy: 0.9930
Epoch 10/15
66/66 [==============================] - 34s 519ms/step - loss: 0.0150 - accuracy: 0.9947 - val_loss: 0.0469 - val_accuracy: 0.9859
Epoch 11/15
66/66 [==============================] - 34s 517ms/step - loss: 0.0491 - accuracy: 0.9886 - val_loss: 0.0383 - val_accuracy: 0.9789
Epoch 12/15
66/66 [==============================] - 34s 515ms/step - loss: 0.0157 - accuracy: 0.9954 - val_loss: 0.2116 - val_accuracy: 0.9648
Epoch 13/15
66/66 [==============================] - 35s 523ms/step - loss: 0.0304 - accuracy: 0.9947 - val_loss: 0.1762 - val_accuracy: 0.9718
Epoch 14/15
66/66 [==============================] - 34s 517ms/step - loss: 0.0244 - accuracy: 0.9962 - val_loss: 0.4869 - val_accuracy: 0.9155
Epoch 15/15
66/66 [==============================] - 34s 518ms/step - loss: 0.0154 - accuracy: 0.9939 - val_loss: 0.0155 - val_accuracy: 0.9859
```

Fig 30: output for training model

In the first epoch the loss value is 0.5 , accuracy value is 0.7, val loss is 0.1 and val accuracy is 0.9 .

By the end of 15^{th} epoch the loss value has decreased to 0.01 and val loss to 0.015 and the accuracy increased to 0.99 and val accuracy to 0.98.

The overall accuracy of the model is 98% which is a best fit model.

```
[ ]  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()
```

Fig 31 : Code for Graph of training model

Reading the history of accuracy, val accuracy, loss and val loss using history . history and plotting the graph for history of accuracies and losses .

Assigning the history of accuracy to train_acc , val accuracy to val_acc , loss to train_loss, val loss to val_loss for labelling the graph.

First we wrote the code for accuracy graph using subplot and plot position as 1,2,1 and using plot we are reading all the epochs of train accuracy and val accuracy and giving the labels of train_ acc and val_acc which reads all the history of both accuracies. The plot title as accuracy.

Next we wrote code for loss graph using subplot and plot position as 1,2,2 and using plot we read the epochs of train loss and val loss , giving the labels of train_loss and val_loss which reads all the history of both losses . The plot title is loss.

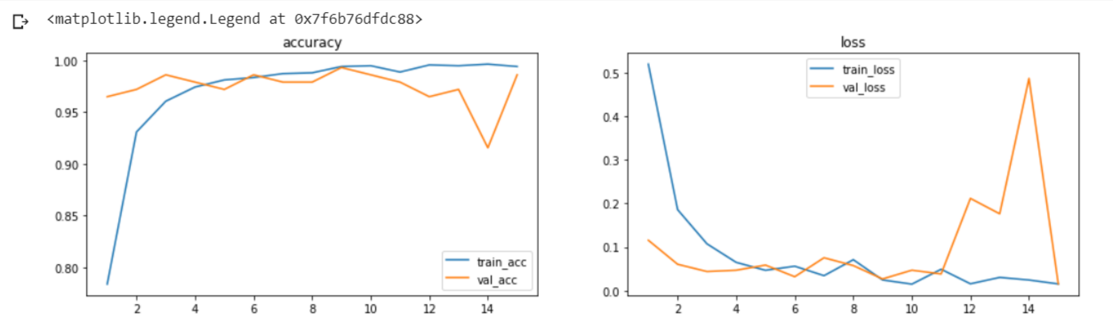


Fig 32: Graph of training model

The graph has accuracy increasing and loss decreasing for training data
Accuracy and loss are fluctuating for validation data.
Overall its increasing so it's a best fit model.

4.4 PREDICTING THE IMAGE:

Predicting image from Test data

1. Read the image
2. check the shape
3. Resize into required shape(1,1501503)
4. Apply scaling

Making a new directory as Random

Fig 33: Terms of prediction.

For testing the model we need to predict the images of with mask and without mask of test data.

But the images of test data are separated in different folders as with mask and without mask , we need to test the images randomly . So for the predicton of images randomly we are making a new directory named "random" and moving all the images of both the with mask and without mask folders to the random directory and then random directory is used to predict the images .

4.4.1 MAKING A NEW DIRECTORY:

Making a new directory as Random

```
[ ]: !mkdir random
```

```
[ ]: !cd random
```

```
[ ]: pwd
[ ]: '/content'
```

Fig 34: New directory

Using mkdir command we are making a new directory “random” and cd command is used for running the further codes in random directory.

Moving all the with mask and without mask images of test data to random directory

```
]: !mv /content/observations/experiements/dest_folder/test/with_mask/* random
```

```
]: !mv /content/observations/experiements/dest_folder/test/without_mask/* random
```

```
]: len(os.listdir('/content/random'))
]: 194
```

Fig 35: Moving images to random.

Using mv command we move all the images of with mask and without mask folders of test directory to random directory.

We used os.listdir to read the data of random directory and len function to print no of images in random directory which are 194 images .

4.4.2: TESTING AND PREDICTING FOR RANDOM IMAGES:

Testing the model for random images

```
import random, os
path = '/content/random'
random_filename = random.choice([
    x for x in os.listdir(path)
    if os.path.isfile(os.path.join(path, x))
])

print(random_filename)
```

```
179.jpg
```

Fig 36: printing random file name-1

We are importing random and os libraries .

We are assigning the path of random directory to “path” variable

Now we assigning “random_filename” to read any one random image from the directory random using the key work random . choice

In random choice we are reading the path and joining it to “x” variable which reads any one image from 194 images

Finally we print the name of image using print function.

```
from tensorflow.keras.preprocessing import image
import numpy as np
img = plt.imread(os.path.join('/content/random',random_filename))
print(type(img))
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,:,:,:])
```

Fig 37: Code for showing the random image-1.

We are importing image from tensorflow and keras .

We are reading the the image which is randomly selected from 194 test images using random_filename and storing it in “img”. Then plotting the image after rescaling and resizing it .We print the actual size and the scaled size of the image along with the image.

```
<class 'numpy.ndarray'>
(267, 189, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
```

```
: <matplotlib.image.AxesImage at 0x7f6b76ba4908>
```

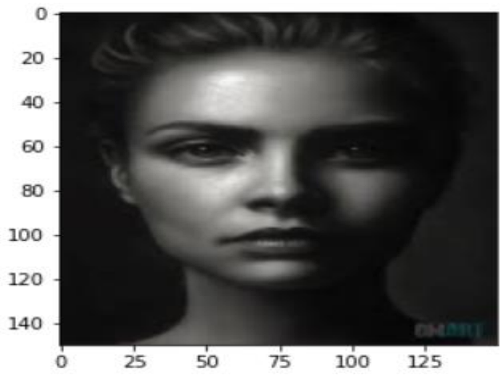


Fig 38: Output of random image-1.

```
]: model.predict(img)
]: array([[0.9984786]], dtype=float32)
```

Fig 39: predicting image 1

We are predicting the value of random image using model . predict and the range of without mask images are 0-1 if the image is predicted correct it shows around 1 and if not it goes to less value .

As this image is a person without mask it is giving 0.99 value so the model is predicting correctly.

```
]: classes = model.predict_classes(img)
   print(classes)
   if (classes==1):
     print("with out mask")
   else :
     print("with mask")

   [[1]]
   with out mask
```

Fig 40: Reading the class of random image -1

To say whether the image is with mask or with out mask image we use model . predict _ classes to know the class of image

Class of without mask is 1 and with mask is 0 ,so if class is 1 it prints without mask and else which is 0 it prints with mask.

Here the image is without mask so its class is 1 .

NOW TESTING FOR SOME OTHER RANDOM IMAGES:

```
[46] import random, os
     path = '/content/random'
     random_filename = random.choice([
         x for x in os.listdir(path)
         if os.path.isfile(os.path.join(path, x))
     ])

     print(random_filename)

     114-with-mask.jpg
```

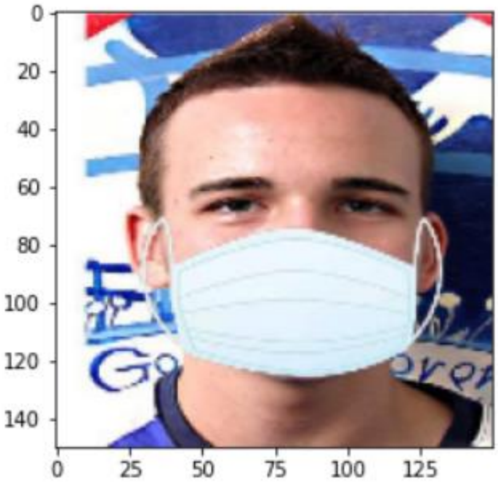
Fig 41: random file name 2

It is image of person with mask.

```
] from tensorflow.keras.preprocessing import image
  import numpy as np
  img = plt.imread(os.path.join('/content/random',random_filename))
  print(type(img))
  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,:,:,:])
```

Fig 42: code for showing random image 2

```
<class 'numpy.ndarray'>
(433, 327, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7fe5d65a0e80>
```




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

```
array([[8.592877e-20]], dtype=float32)
```

Fig 43: prediction of random image 2

Here the image is a person with mask. The array value is accuracy of predicting the person with mask , if the person wear a mask properly it varies if not it decreases.

```
[49] classes = model.predict_classes(img)
     print(classes)
     if (classes==1):
       print("with out mask")
     else :
       print("with mask")
```

```
[[0]]
with mask
```

Fig 44: class of random image 2

Class is 0 and with mask image .

```
[50] import random, os
     path = '/content/random'
     random_filename = random.choice([
         x for x in os.listdir(path)
         if os.path.isfile(os.path.join(path, x))
     ])

     print(random_filename)
```

```
284.jpg
```

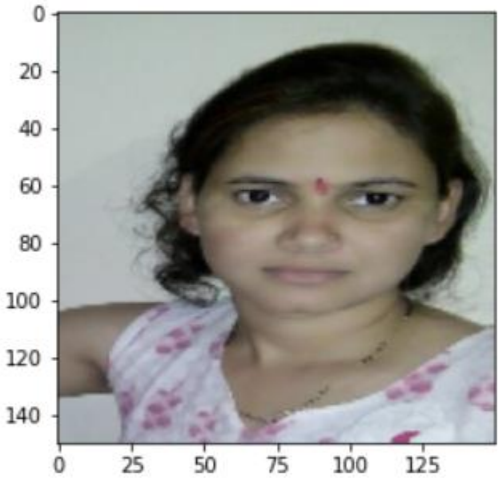
Fig 45: random filename 3

```
1] from tensorflow.keras.preprocessing import image
   import numpy as np
   img = plt.imread(os.path.join('/content/random',random_filename))
   print(type(img))
   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,:,:,:])
```

Fig 46: code for testing random image 3

It is a image of person without mask.

```
<class 'numpy.ndarray'>
(576, 398, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7fe5d62bc8d0>
```




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

```
array([[1.]], dtype=float32)
```

Fig 47: Prediction of random image 3

The image is without mask and the array value is 1 ,so the model predicted the person is not wearing a mask.

```
[53] classes = model.predict_classes(img)
     print(classes)
     if (classes==1):
       print("with out mask")
     else :
       print("with mask")
```

```
[[1]]
with out mask
```

Fig 48: class of random image 3

Class is also 1 and its with out mask image.

Testing the model for random images

```
[57] import random, os
     path = '/content/random'
     random_filename = random.choice([
         x for x in os.listdir(path)
         if os.path.isfile(os.path.join(path, x))
     ])

     print(random_filename)
```

```
464-with-mask.jpg
```

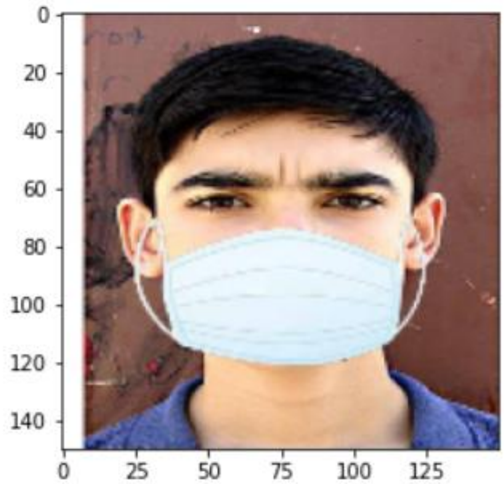
Fig 49: random filename 4

```
[58] from tensorflow.keras.preprocessing import image
     import numpy as np
     img = plt.imread(os.path.join('/content/random',random_filename))
     print(type(img))
     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,:,:,:])
```

Fig 50: code of showing random image 4

The image is a person with mask.

```
<class 'numpy.ndarray'>
(428, 320, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7fe5d6463550>
```




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

```
array([[5.7457684e-13]], dtype=float32)
```

Fig 51: Predicting random image 4

The array value is less than the previous image because the person is not wearing the mask properly, so the array value is lesser.

```
[60] classes = model.predict_classes(img)
     print(classes)
     if (classes==1):
       print("with out mask")
     else :
       print("with mask")
```

```
[[0]]
with mask
```

Fig 52: class of random image 4

Class is 0 and the image is with mask

Hence the model has predicted all the images randomly and with correct accuracy , the model is a best fit model.

CONCLUSION:

The proposed work is designed to develop a model to detect, recognize and classify human face. The softwares used to test the functionality are Anaconda and python 3 and google colaboratory. For Face detection dataset we used convolution neural network model for face recognition and classification. The os , matplotlib, tenserflow and other libraries works in support with python programming. The performance measures are validated with the CNN model designed with an accuracy of 98%. However the results prove that the network architecture designed has better advancements and this application is widely used to achieve face classification and recognition.

REFERENCES:

DATA SET LINK:

https://github.com/prajnasb/observations

TENSOR FLOW LINK:

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

MACHINE LEARNING LINK:

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

PYTHON LINKS:

https://www.python.org/

https://en.wikipedia.org/wiki/Python_(programming_language)