IMAGE PROCESSING AND ANALYSIS WITH MACHINE LEARNING

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Under the guidance of

MAHENDRA DATTA

BY

AYAN ROY

HERITAGE INSTITUTE OF TECHNOLOGY

In association with

(ISO9001:2015)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

(Note: All entries of the pro forma of approval should be filled up with appropriate and complete information. Incomplete pro forma of approval in any respect will be summarily rejected.)

1.	Title of the Project:	**IMAGE PROCESSING AND ANALYSIS WITH MACHINE LEARNING**
2.	Project Members	**AYAN ROY**
3.	Name of the guide:	**Mr. MAHENDRA DUTTA**
4.	Address:	Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified) SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal, 700091

Project Version Control History

Version	**Primary Author**	**Description of Version**	**Date Completed**
Final	AYAN ROY	Project Report	23 OCTOBER, 2021

Signature of the Team member

Date: 23 October 2021

Ayan Roy

Ayan Roy

Signature of the Approver

Date: 23 October 2021

Mahendra Dutta.

Mr Mahendra Dutta

Project Proposal Evaluator

For Office Use Only

[✓ Approved] Not Approved

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **"IMAGE PROCESSING AND ANALYSIS WITH MACHINE LEARNING"** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, SALTLAKE, KOLKATA, WEST BENGAL,** is an authentic work carried out under the guidance of **MR. MAHENDRA DUTTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 23 October 2021

Name of the Student: AYAN ROY

Ayan Roy

Signature of the student

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of minor project entitled **"IMAGE PROCESSING AND ANALYSIS WITH MACHINE LEARNING"** is a record of bonafide work, carried out by **AYAN ROY** under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT®.** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

Mahendra Dutta.

MR. MAHENDRA DUTTA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. MAHENDRA DUTTA***, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

Brief Description of the Project

The name of the project is Image Processing and Analysis using Machine Learning. In this project I have used Image Processing techniques to analyse a sample image and identify the prominent colours in the image.

I have used the pixels of the image along with its RGB values as the dataset and visualized the top 5 colours in the image with the help of a pie plot.

CONTENTS

Image Processing

The field of digital image processing refers to processing digital images by means of a digital computer to get an enhanced image or to extract useful information pertinent to user from that image.

It is application of signal processing in which the input is an image and the output may also be a modified version of the input image or may have certain characteristic features associated with that input image.

What is a digital image?

An image may be defined as a two dimensional function f(x,y), where x and y are spatial coordinates, and the amplitude f of at any pair of coordinates (x,y) is called intensity or gray level of the image at that point.

The function $f(x,y)$ representing a digital image $(M \times N)$ can be represented in a matrix form as -

$$\begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ . & . & \dots & . \\ . & . & \dots & . \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix}$$

Sampling

Sampling is a technique of selecting individual points or a subset of points from the entire image to make statistical interferences from them and estimate characteristics of the whole image.

There are two types of sampling

1. **Down sampling** is the reduction in spatial resolution while keeping the same two-dimensional representation. It is typically used to reduce the storage and/or transmission requirements of images.

2. **Up sampling** is the increasing of the spatial resolution while keeping the 2D representation of an image. It is done by inserting zero-valued samples between original samples to increase the sampling rate.

Interpolation

Image interpolation occurs when you resize or distort your image from one pixel grid to another. Image resizing is necessary when you need to increase or decrease the total number of pixels, whereas remapping can occur when you are correcting for lens distortion or rotating an image. Zooming refers to increase the quantity of pixels, so that when you zoom an image, you will see more detail.

Interpolation works by using known data to estimate values at unknown points. Image interpolation works in two directions, and tries to achieve a best approximation of a pixel's intensity based on the values at surrounding pixels.

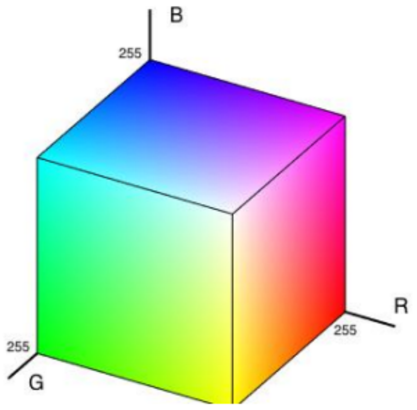
Common interpolation algorithms can be grouped into two categories: adaptive and non-adaptive. Adaptive methods change depending on what they are interpolating, whereas non-adaptive methods treat all pixels equally. Non-adaptive algorithms include: nearest neighbor, bilinear, bicubic, spline, sinc, lanczos and others. Adaptive algorithms include many proprietary algorithms in licensed software such as: Qimage, PhotoZoom Pro and Genuine Fractals.

RGB Model

The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

Color pigments are added to produce another effect different from mixing of lights based on the property of light absorption.

The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography.



Histogram

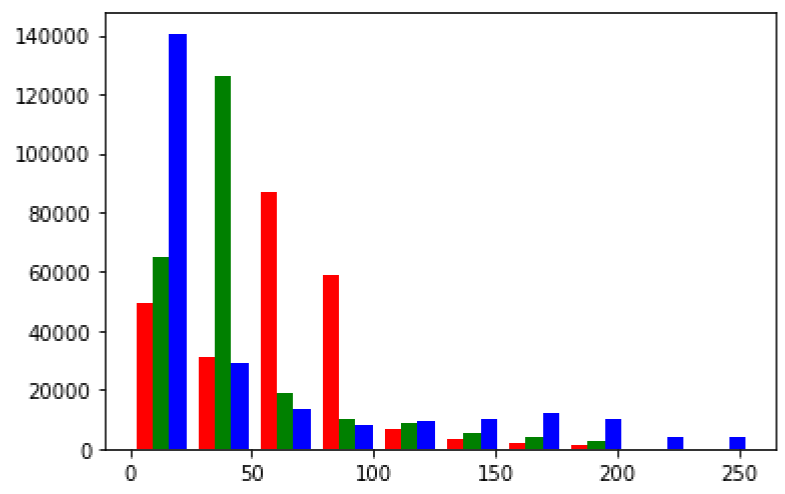
In an image processing context, the histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image.

For an 8-bit grayscale image there are 256 different possible intensities, and so the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale values.

Histograms can also be taken of color images --- either individual histograms of red, green and blue channels can be taken, or a 3-D histogram can be produced, with the three axes representing the red, blue and green channels, and brightness at each point representing the pixel count. The exact output from the operation depends upon the implementation --- it may simply be a picture of the required histogram in a suitable image format, or it may be a data file of some sort representing the histogram statistics.

For Example

Here we have a histogram of the RGB values corresponding the number of pixels in the image.



OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

Installation

OpenCV can be directly downloaded and installed with the use of pip (package manager). To install OpenCV, just go to the command-line and type the following command:

```
pip install opencv-python
```

We can the import the module by typing

```
import cv2
```

Reading an image

For reading an image, use the `imread()` function in OpenCV. Here's the syntax:

```
imread(filename, flags)
```

It takes two arguments:

1. The first argument is the image name, which requires a fully qualified **pathname** to the file.
2. The second argument is an optional flag that lets you specify how the image should be represented.

Showing an Image

In OpenCV, you display an image using the `imshow()` function. Here's the syntax:

```
imshow(window_name, image)
```

This function also takes two arguments:

1. The first argument is the window name that will be displayed on the window.
2. The second argument is the image that you want to display.

Output

Color Space Convertion

`cv2.cvtColor()` method is used to convert an image from one color space to another. There are more than 150 color-space conversion methods available in OpenCV.

By default when we read an image the colour format is BGR and hence does not match with the original image. That's why we need to convert it to RGB format.

For example for converting an image from BGR format to RGB format we write –

```
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

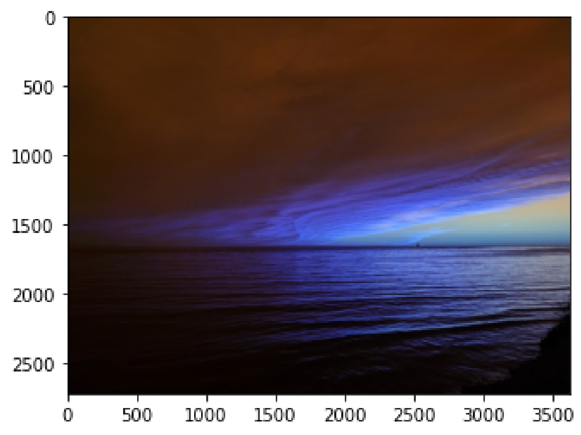


Image in BGR format

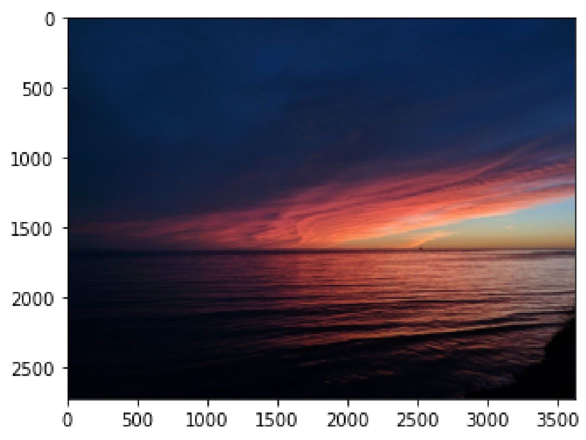


Image in RGB format

OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a 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-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the 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 enables 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.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high level dynamic datatypes and supports dynamic type checking.

- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**.
However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And, exec, not
Assert, finally, or
Break, for, pass
Class, from, print
continue, global, raise
def, if, return
del, import, try
elif, in, while
else, is, with
except, lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced. The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
print "True"
else:
print "False"
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python-h
usage: python [option]...[-c cmd|-m mod | file |-][arg]...

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d      : debug output from parser (also PYTHONDEBUG=x)

-E      : ignore environment variables (such as PYTHONPATH)

-h      : print this help message and exit [ etc.]
```

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.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10              # An integer assignment
weight=10.60            # A floating point
name="Ardent"            # A string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example −

```
a = b = c = 1
a,b,c = 1,2,"hello"
```

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types −

- String
- List
- Tuple

- 🎬 Dictionary
- 🎬 Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	**int(x [,base])** Converts x to an integer. base specifies the base if x is a string
2	**long(x [,base])** Converts x to a long integer. base specifies the base if x is a string.
3	**float(x)** Converts x to a floating-point number.
4	**complex(real [,imag])** Creates a complex number.
5	**str(x)** Converts object x to a string representation.
6	**repr(x)** Converts object x to an expression string.
7	**eval(str)** Evaluates a string and returns an object.
8	**tuple(s)** Converts s to a tuple.
9	**list(s)** Converts s to a list.

Counters

A Counter is a subclass of dict. Therefore it is an unordered collection where elements and their respective count are stored as a dictionary. This is equivalent to a bag or multiset of other languages.

Syntax

```
class collections.Counter([iterable-or-mapping])
```

It is obtained from the collection module.

FUNCTIONS

Defining a Function

- def function name(parameters):
    "function_docstring"
    function suite
    return [expression]

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

# Function definition is here

```
def change me(mylist):
        "This changes a passed list into this function"
        mylist.append([1,2,3,4]);
        print"Values inside the function: ",mylist
        return
```

# Now you can call changeme function

```
mylist=[10,20,30];
change me(mylist);
print" Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result −

```
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]
```

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;                    # This is global variable.

# Function definition is here

def sum( arg1, arg2 ):


# Add both the parameters and return them."

total= arg1 + arg2;      # Here total is local variable.
print"Inside the function local total: ", total
return total;


# Now you can call sum function

sum(10,20);
Print”Outside the function global total: ", total
```

When the above code is executed, it produces the following result −

```
Inside the function local total: 30
Outside the function global total: 0
```

MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
def print_func( par ):
    print"Hello : ", par
    return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
Import module1 [, module2 [… moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

```
def Pots ():
        print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function Isdn ()
- *Phone/G3.py* file having function G3 ()

Now, create one more file __init__.py in *Phone* directory −

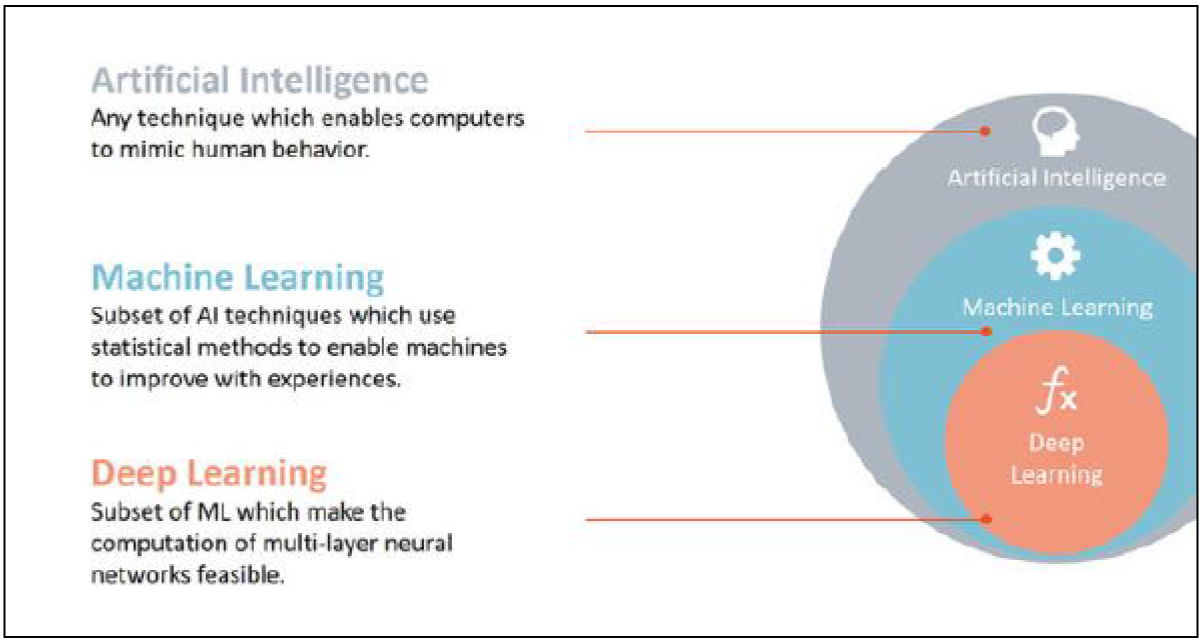
- Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import
```

ARTIFICIAL INTELLIGENCE

Introduction


Artificial Intelligence
Any technique which enables computers to mimic human behavior.
Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.
Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.
Artificial Intelligence
Machine Learning
Deep Learning

According to the father of Artificial Intelligence, John McCarthy, it is *"The science and engineering of making intelligent machines, especially intelligent computer programs".*

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems − The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.

To Implement Human Intelligence in Machines − Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as:-

Gaming − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing − It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems − These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information
Or map of the areas.

Doctors use clinical expert system to diagnose the patient.

Police use computer software that can recognize the face of criminal with the stored
portrait made by forensic artist.

Speech Recognition − Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition − The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

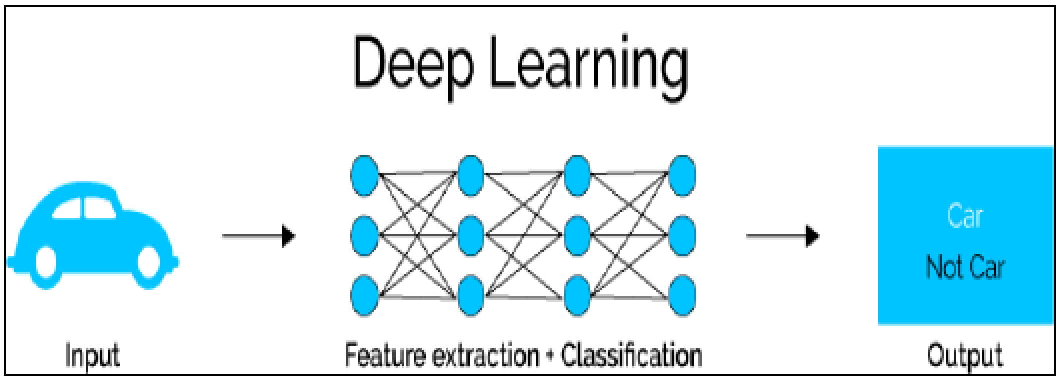
Application of AI

Deep Learning

Deep learning is a subset of machine learning. Usually, when people use the term deep learning, they are referring to deep artificial neural networks, and somewhat less frequently to deep reinforcement learning.

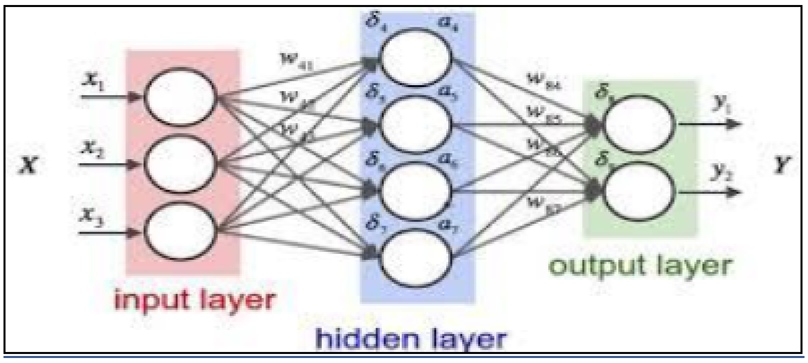
Deep learning is a class of machine learning algorithms that:

- Use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- Learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- Learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- Use some form of gradient descent for training via backpropagation.



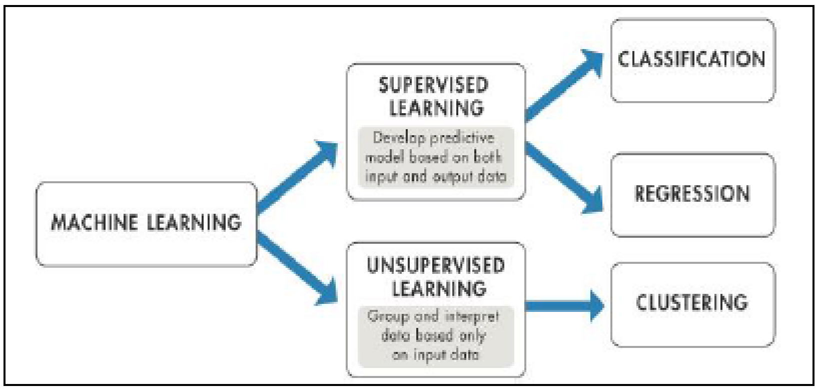
NEURAL NETWORKING

Artificial neural networks (**ANNs**) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance on) tasks by considering examples, generally without task-specific programming



An ANN is based on a collection of connected units or nodes called artificial neurons (analogous to biological neurons in an animal brain). Each connection between artificial neurons can transmit a signal from one to another.

MACHINE LEARNING


MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.
Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labelled training data*.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[1., 0., 0.],
 [ 0., 1., 2.]]
```

NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

```
Import numpy as np

a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])

print 'Our array is:'
Print a
print '\n'
print 'The items in the second column are:'
print a[...,1]
print '\n'
print 'The items in the second row are:'
print a[1...]
print '\n'
print 'The items columns 1 onwards are:'
print a [...,1:]
```

OUTPUT

```
Our array is:
[[1 2 3]
[3 4 5]
[4 5 6]]

The items in the second column are:
[2 4 5]

The items in the second row are:
[3 4 5]

The items column 1 onwards are:
[[2 3]
[4 5]
[5 6]]
```

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

Installation

OpenCV can be directly downloaded and installed with the use of pip (package manager). To install OpenCV, just go to the command-line and type the following command:

```
pip install opencv-python
```

We can the import the module by typing

```
import cv2
```

Reading an image

For reading an image, use the `imread()` function in OpenCV. Here's the syntax:

```
imread(filename, flags)
```

It takes two arguments:

3. The first argument is the image name, which requires a fully qualified **pathname** to the file.
4. The second argument is an optional flag that lets you specify how the image should be represented.

Showing an Image

In OpenCV, you display an image using the `imshow()` function. Here's the syntax:

```
imshow(window_name, image)
```

This function also takes two arguments:

3. The first argument is the window name that will be displayed on the window.
4. The second argument is the image that you want to display.

Output

0
500
1000
1500
2000
2500
0
500
1000
1500
2000
2500
3000
3500

<u>**Color Space Convertion**</u>

`cv2.cvtColor()` method is used to convert an image from one color space to another. There are more than 150 color-space conversion methods available in OpenCV.

By default when we read an image the colour format is BGR and hence does not match with the original image. That's why we need to convert it to RGB format.

For example for converting an image from BGR format to RGB format we write –

```
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

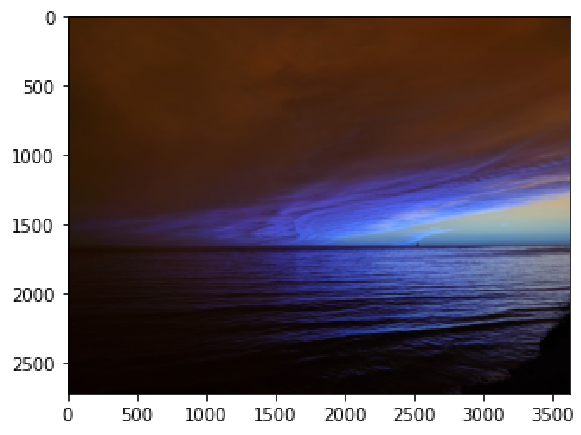


Image in BGR format

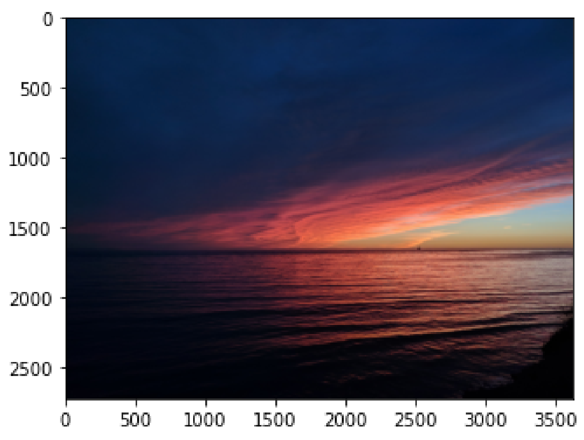


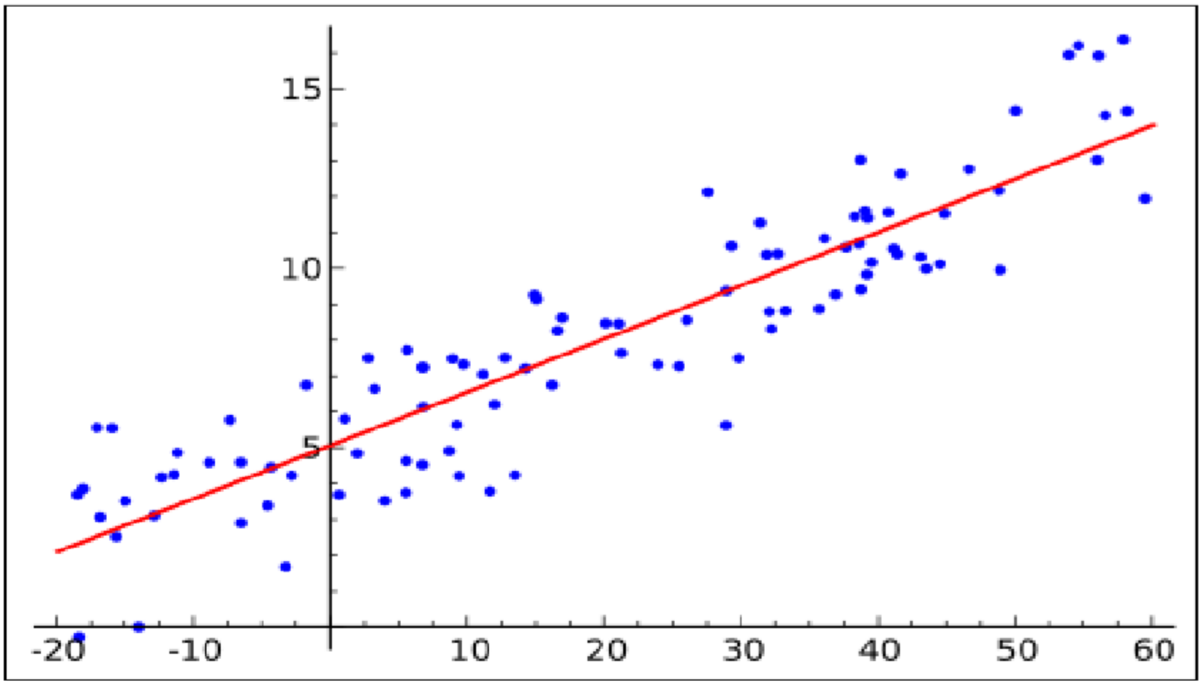
Image in RGB format

SCIKIT-LEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.

LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model [1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x.

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted E(y | x), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data.

CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

Types of Clustering Algorithms

Since the task of clustering is subjective, the means that can be used for achieving this goal are plenty. Every methodology follows a different set of rules for defining the '*similarity'* among data points. In fact, there are more than 100 clustering algorithms known. But few of the algorithms are used popularly, let's look at them in detail:

- **Connectivity models:** As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.

- **Centroid models:** These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category. In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset. These models run iteratively to find the local optima.

- **Distribution models:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.

- **Density Models:_**These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.

K-means Clustering

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*. The algorithm works iteratively to assign each data point to one of *K* groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the *K*-means clustering algorithm are:

1. The centroids of the *K* clusters, which can be used to label new data
2. Labels for the training data (each data point is assigned to a single cluster)

Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have formed organically. The "Choosing K" section below describes how the number of groups can be determined.

Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

Gaussian Mixture Model

A Gaussian mixture model is a **probabilistic model** that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.It is also called Expectation-Maximization Clustering or EM Clustering and is based on the optimization strategy. Gaussian Mixture models are used for representing Normally Distributed subpopulations within an overall population. The advantage of Mixture models is that they do not require which subpopulation a data point belongs to. It allows the model to learn the subpopulations automatically. This constitutes a form of unsupervised learning.

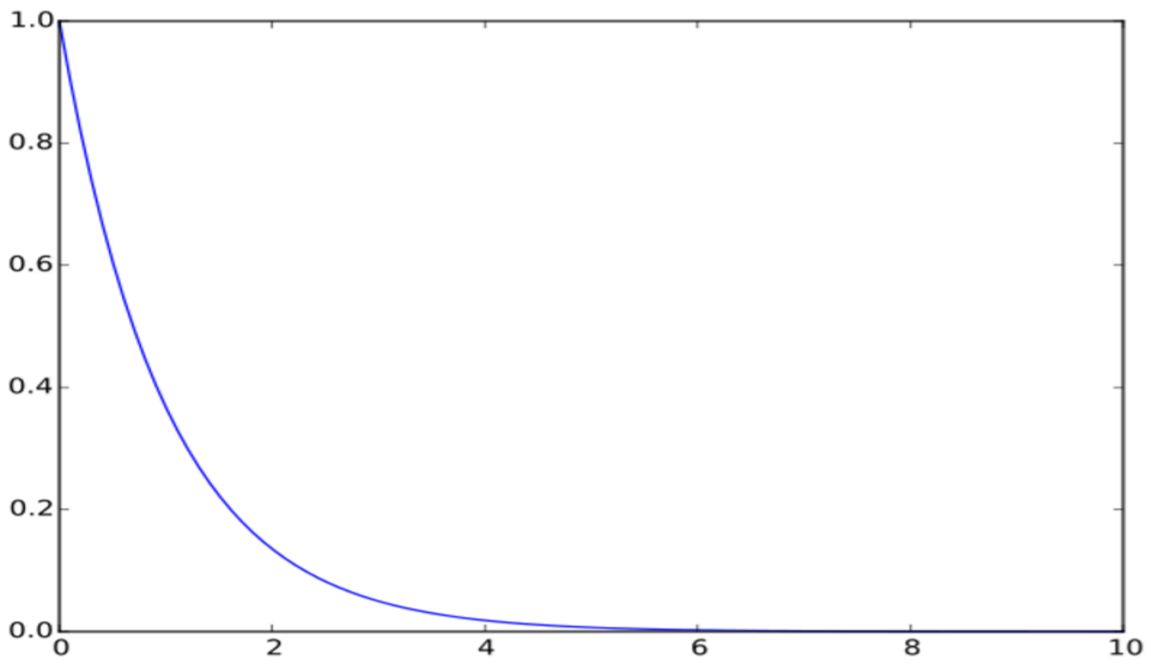
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged .SciPy makes use of matplotlib.

EXAMPLE

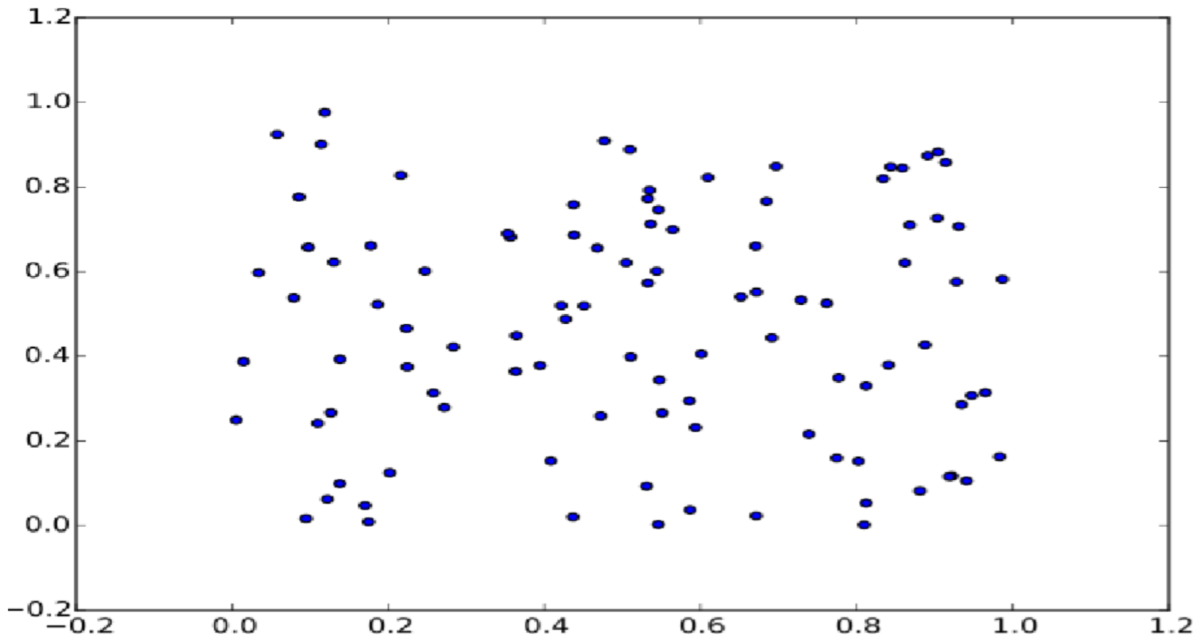
LINE PLOT

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>> a =np.linspace(0,10,100)
>>> b =np.exp(-a)
>>>plt.plot (a,b)
>>>plt.show ()
```



SCATTER PLOT

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimport rand
>>> a =rand(100)
>>> b =rand(100)
>>>plt.scatter(a, b)
>>>plt.show ()
```



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

Data Frame object for data manipulation with integrated indexing.
Tools for reading and writing data between in-memory data structures and different file formats.
Data alignment and integrated handling of missing data.
Reshaping and pivoting of data sets.
Label-based slicing, fancy indexing, and sub setting of large data sets.
Data structure column insertion and deletion.
Group by engine allowing split-apply-combine operations on data sets.
Data set merging and joining.
Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
Time series-functionality: Date range generation.

ALGORITHM

Image Processing
Data Collection
Clustering
Identifying Colours

Visualization

Image Processing: I have downloaded the sample image from the internet and processed the image to the required conditions, which included down sampling, interpolation and converting to the right color space.

Data Collection: I have used the number of pixels in the sample image along with its RGB value as the data set.

Clustering: I have used certain clustering algorithms like K-means clustering and Gaussian Mixture Model to separate the data set into different clusters. With each cluster representing the main colours present in the image.

Identifying the Colours: I then identify the colours of each of the cluster.

Visualization: The top 5 colours present in the image are then visualized with the help of a pie plot.

Codes for the Project

```
In [2]: from sklearn.cluster import KMeans
        import matplotlib.pyplot as plt
        import pandas as pd
        import numpy as np
        import cv2
        from collections import Counter
        from skimage.color import rgb2lab, deltaE_cie76
```

Importing libraries

```
In [3]: def RGB2HEX(color):
            return "#{:02x}{:02x}{:02x}".format(int(color[0]), int(color[1]), int(color[2]))
```

Function to convert the colours from RGB format to HEX colour codes

```
In [4]: def get_image(image_path):
            image = cv2.imread(image_path)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            return image
```

Function to read an image and convert it from BGR to RGB format. By default when we read an image the colour format is BGR and hence does not match with the original image. That's why we need to convert it to RGB format.

```
In [5]: def get_colors(image, number_of_colors, show_chart):

            modified_image = cv2.resize(image, (600, 400), interpolation = cv2.INTER_AREA)
            modified_image = modified_image.reshape(modified_image.shape[0]*modified_image.shape[1], 3)
            df = pd.DataFrame(modified_image, columns = ['Red','Green','Blue'])
            print(df)

            clf = KMeans(n_clusters = number_of_colors)
            labels = clf.fit_predict(modified_image)

            counts = Counter(labels)
            # sort to ensure correct color percentage
            counts = dict(sorted(counts.items()))

            center_colors = clf.cluster_centers_

            ordered_colors = [center_colors[i] for i in counts.keys()]
            hex_colors = [RGB2HEX(ordered_colors[i]) for i in counts.keys()]
            rgb_colors = [ordered_colors[i] for i in counts.keys()]

            # We get ordered colors by iterating through the keys

            if (show_chart):
                plt.figure(figsize = (8, 6))
                plt.pie(counts.values(), labels = hex_colors, colors = hex_colors)

            return rgb_colors
```

In the above function, we first resize the image from (3627, 2720) to (600, 400) Resizing an image needs a way to calculate pixel values for the new image from the original one. For this we use an interpolation process called - INTER_AREA INTER_AREA is resampling using pixel area relation. It may be a preferred method for image decimation (downsampling) as it gives more free results.

We then reshape the image in terms of the number of pixels and the RGB combination of each pixel. Hence if we display the image as a dataframe, we get 240000 rows and 3 columns.

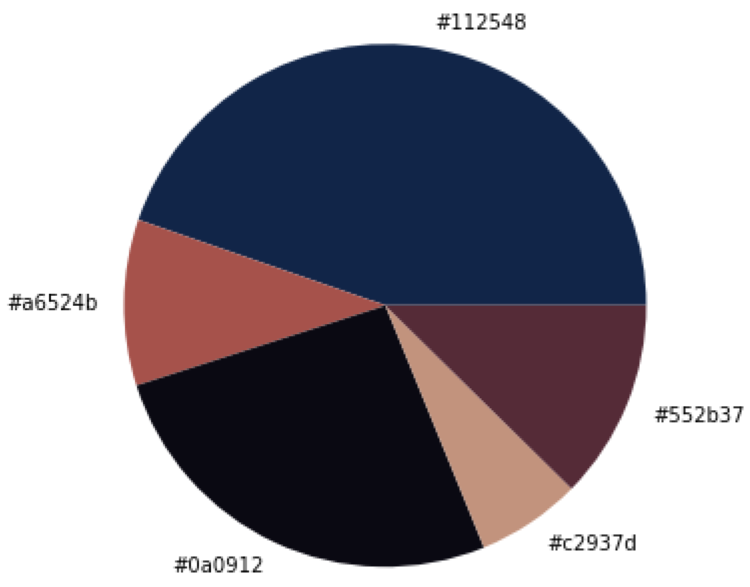
We then use KMeans algorithm to find different clusters. Each cluster has its own data points with almost similar colours. The clusters are predicted and stored in the variable labels, which are then sorted and stored in the variable counts.

We get ordered_colours by iterating through the keys.

```
In [7]: image = cv2.imread('picture1.jpg')


        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        plt.imshow(image)

        get_colors(get_image('picture1.jpg'), 5, True)
```

Output

```
        Red  Green  Blue
0         5     34    64
1         5     34    64
2         4     33    64
3         4     32    64
4         6     33    63
...     ...    ...   ...
239995    2      2     4
239996    2      2     4
239997    1      1     3
239998    2      2     4
239999    2      2     4

[240000 rows x 3 columns]



[array([17.4259639 , 37.03662652, 72.62811358]),
 array([166.32534386,  82.60828518,  75.57373647]),
 array([10.78977562,  9.24394546, 18.15522502]),
 array([194.72554801, 147.28707858, 125.3877067 ]),
 array([85.70169457, 43.26681069, 55.95544153])]
```



The above model was done by using K-means clustering

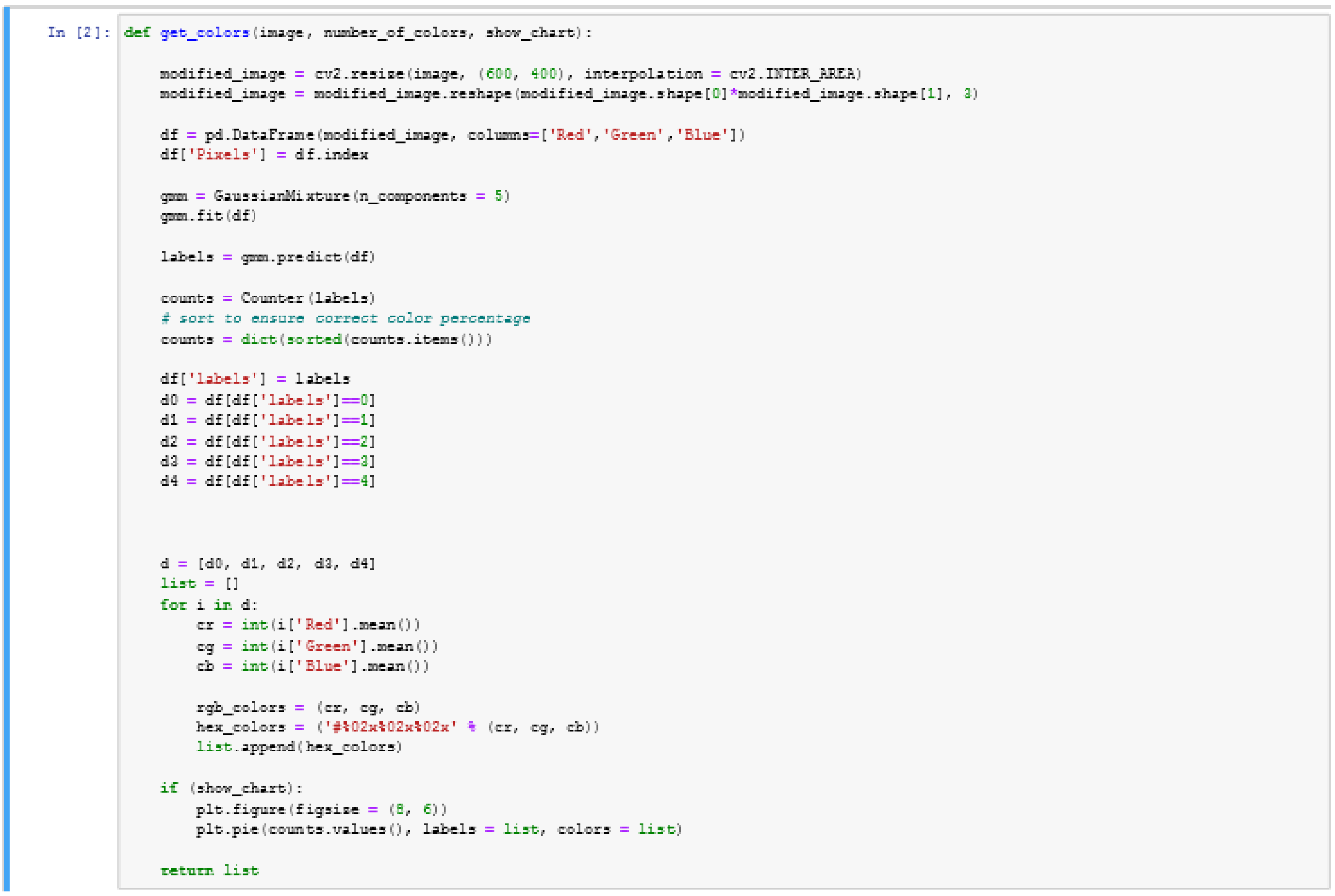
The second model shown below is done by using Gaussian Mixture Model

```
In [1]:  from sklearn.mixture import GaussianMixture
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import cv2
        from collections import Counter
        from skimage.color import rgb2lab, deltaE_cie76
```

Importing libraries

```
In [ ]: def get_image(image_path):
            image = cv2.imread(image_path)
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            return image
```

Function to read image and convert from BGR format to RGB



```
In [2]: def get_colors(image, number_of_colors, show_chart):

            modified_image = cv2.resize(image, (600, 400), interpolation = cv2.INTER_AREA)
            modified_image = modified_image.reshape(modified_image.shape[0]*modified_image.shape[1], 3)

            df = pd.DataFrame(modified_image, columns=['Red','Green','Blue'])
            df['Pixels'] = df.index

            gmm = GaussianMixture(n_components = 5)
            gmm.fit(df)

            labels = gmm.predict(df)

            counts = Counter(labels)
            # sort to ensure correct color percentage
            counts = dict(sorted(counts.items()))

            df['labels'] = labels
            d0 = df[df['labels']==0]
            d1 = df[df['labels']==1]
            d2 = df[df['labels']==2]
            d3 = df[df['labels']==3]
            d4 = df[df['labels']==4]



            d = [d0, d1, d2, d3, d4]
            list = []
            for i in d:
                cr = int(i['Red'].mean())
                cg = int(i['Green'].mean())
                cb = int(i['Blue'].mean())

                rgb_colors = (cr, cg, cb)
                hex_colors = ('#%02x%02x%02x' % (cr, cg, cb))
                list.append(hex_colors)

            if (show_chart):
                plt.figure(figsize = (8, 6))
                plt.pie(counts.values(), labels = list, colors = list)

            return list
```

In the above function, we first resize the image from (3627, 2720) to (600, 400) We then reshape the image in terms of the number of pixels and the RGB combination of each pixel. Hence if we display the image as a dataframe (df), we get 240000 rows and 3 columns.

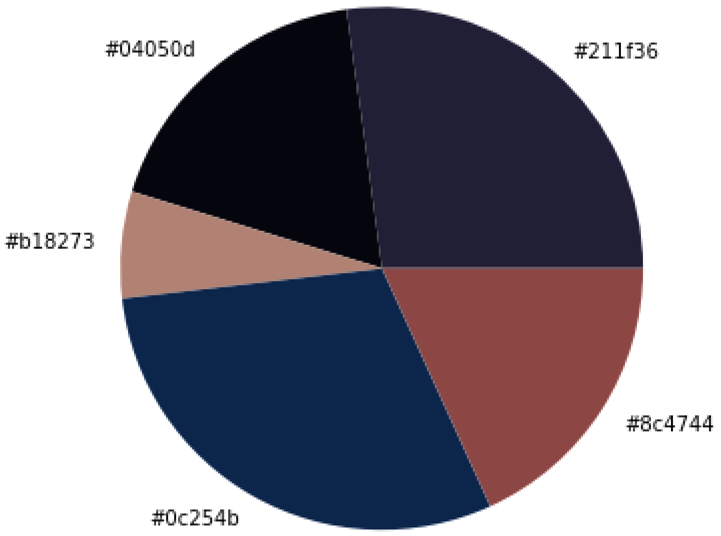
We then fit the dataframe (df) into the Gaussian Mixture Model. The predicted values are stored in the variable labels.

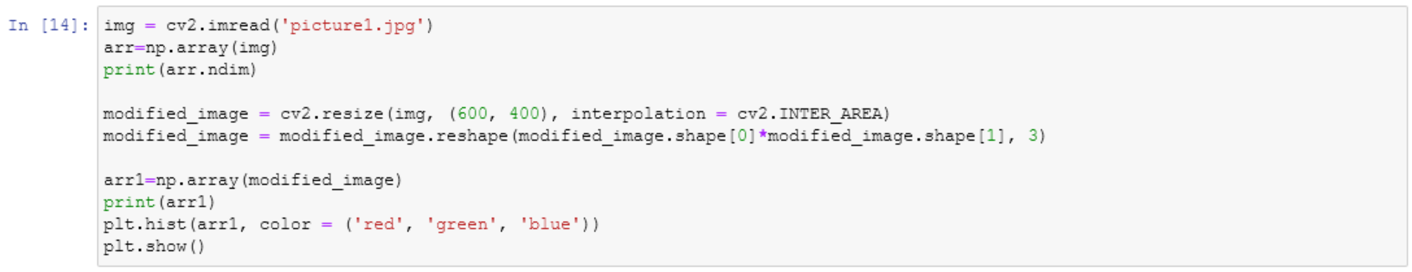
I then divide the original dataframe (df)into subgroups based on the different labels, each of the clusters have. I then find the mean values of the RGB colors in each of the clusters, and iterate through the different dataframes (d0, d1, d2, d3, d4) to represent each cluster by a single mean color.

```
image = cv2.imread('picture1.jpg')

image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image)

get_colors(get_image('picture1.jpg'), 5, True)
```

Output

```
['#211f36', '#04050d', '#b18273', '#0c254b', '#8c4744']
```



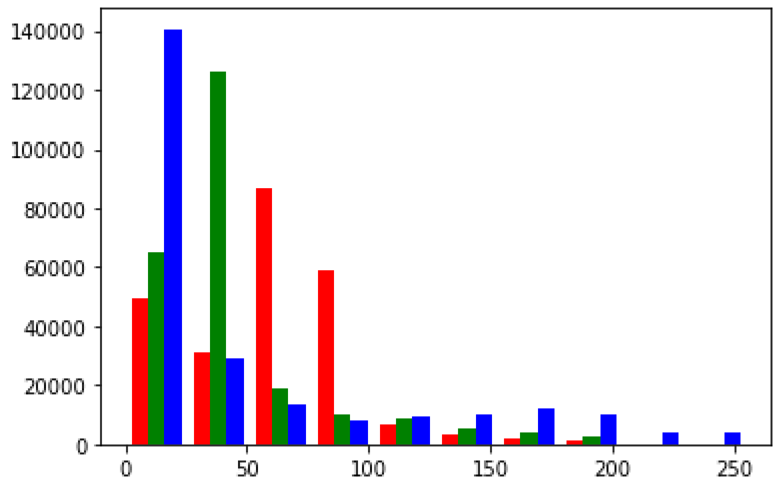


```
In [14]: img = cv2.imread('picture1.jpg')
         arr=np.array(img)
         print(arr.ndim)

         modified_image = cv2.resize(img, (600, 400), interpolation = cv2.INTER_AREA)
         modified_image = modified_image.reshape(modified_image.shape[0]*modified_image.shape[1], 3)

         arr1=np.array(modified_image)
         print(arr1)
         plt.hist(arr1, color = ('red', 'green', 'blue'))
         plt.show()
```

In the above code I have used a histogram to visualize the RGB values of the sample image corresponding to the number of pixels.

Here is the output

```
3
[[64 34  5]
 [64 34  5]
 [64 33  4]
 ...
 [ 3  1  1]
 [ 4  2  2]
 [ 4  2  2]]
```



Here is a histogram showing the RGB values corresponding to the number of pixels.

<u>Conclusion –</u>

Hence by making this project I have been able to identify the colours in an image and find out which colours represent the image the most. I have used two clustering algorithms for this, K-means algorithm and Gaussian Mixture Model, where K-means is a centroid based algorithm or distance based algorithm while Gaussian Mixture Model is more of a probabilistic model.

Future Scope –

This project can be used in applications which can help to sort and even search images by specific colours.

This will also help in analysing images better and extract better information, using the appropriate image processing techniques.

Bibliography –

https://nanonets.com/blog/machine-learning-image-processing/?utm_source=pocket_mylist

https://scikit-learn.org/stable/modules/mixture.html

https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

https://docs.opencv.org/4.5.4/

THANK YOU