Vidyasagar University

A AI approach to faceswap of two images using Opencv python

By

NAME: **ADITYA DAS**

ROLL_NO.:

REGISTRATION NO.:

NAME: **SUPORNA SARKAR**

ROLL_NO.:

REGISTRATION NO.:

NAME: **ANANYA GHOSH**

ROLL_NO.:

REGISTRATION NO.:

NAME: **MOUMITA MAITY**

ROLL_NO.:

REGISTRATION NO.:

Under the supervised mentor

Dr. Partho Choudhury

ABSTRACT

Machine learning involves artificial intelligence and it is used in solving many problems in data science, also in the fields of image processing. One common application of artificial intelligence is the predict of an outcome based upon exiting data; The artificial intelligence learns patterns from the exiting dataset or output and then applies them to an unknown dataset in order to predict the outcome or output. In case of image processing the package called OpenCV or OpenCV3 is the efficient way to edit image or video. It takes input and process those input depends upon their pixel & density and applying algorithm, it generate outputs. Experiments with this tool were performed using two person's face dlib 68' point;

A comparative analytical approach was done to determine how the ensemble technique can be applied for improving output in case of swap two faces. The focus of this paper is not only on increasing the accuracy of the output of weak algorithms, but also on the implementation of the proper algorithm with a particular input. The results of the study indicate that ensemble technique, such as dlib 68' point method, facial land marking, triangle wrapping these are effectively improving the accuracy of the output and exhibit satisfactory performance in case of image processing. In a case study a maximum of 7% accuracy for the weak algorithm to use such another method like in our case dlib 68' point, facial land marking, triangle wrapping etc.

The performance of the process was future enhance with a feature selection implementation and the results showed significant improvement in prediction accuracy.

Chapter-1

1.1 PURPOSE OF AI IN FACE-Swapping:

When faceswapping was first developed and published, the technology was groundbreaking, it was a huge step in AI development. It was also completely ignored outside of academia because the code was confusing and fragmentary. It required a thorough understanding of complicated AI techniques and took a lot of effort to figure it out. Until one individual brought it together into a single, cohesive collection. It ran, it worked, and as is so often the way with new technology emerging on the internet, it was immediately used to create inappropriate content. Despite the inappropriate uses the software was given originally, it was the first AI code that anyone could download, run and learn by experimentation without having a Ph.D. in math, computer theory, psychology, and more. Before "deepfakes" these techniques were like black magic, only practiced by those who could understand all of the inner workings as described in esoteric and endlessly complicated books and papers. "Deepfakes" changed all that and anyone could participate in AI development. To us, developers, the release of this code opened up a fantastic learning opportunity. It allowed us to build on ideas developed by others, collaborate with a variety of skilled coders, experiment with AI whilst learning new skills and ultimately contribute towards an emerging technology which will only see more mainstream use as it progresses. Are there some out there doing horrible things with similar software? Yes. And because of this, the developers have been following strict ethical standards. Many of us don't even use it to create videos, we just tinker with the code to see what it does. Sadly, the media concentrates only on the unethical uses of this software. That is, unfortunately, the nature of how it was first exposed to the public, but it is not representative of why it was created, how we use it now, or what we see in its future. Like any technology, it can be used for good or it can be abused. It is our intention to develop FaceSwap in a way that its potential for abuse is minimized whilst maximizing its potential as a tool for learning, experimenting and, yes, for legitimate faceswapping. We are not trying to denigrate celebrities or to demean anyone. We are programmers, we are engineers, we are Hollywood VFX artists, we are activists, we are hobbyists, we are human beings. To this end, we feel that it's time to come out with a standard statement of what this software is and isn't as far as us developers are concerned. FaceSwap is not for creating inappropriate content. FaceSwap is not for changing faces without consent or with the intent of hiding its use. FaceSwap is not for any illicit, unethical, or questionable purposes.FaceSwap exists to experiment and discover AI techniques, for social or political commentary, for movies, and for any number of ethical and reasonable uses. We are very troubled by the fact that FaceSwap can be used for unethical and disreputable things. However, we support the development of tools and techniques that can be used ethically as well as provide education and experience in AI for anyone who wants to learn it hands-on. We will take a zero

tolerance approach to anyone using this software for any unethical purposes and will actively discourage any such uses.

1.2 What is AI?

AI is a technique that enables machines to mimic human behavior artificial intelligence is the theory and development of computer systems able to perform tasks normally requiring human intelligence such as visual perception speech recognition decision-making and translation between languages.

Now the term artificial intelligence was actually coined way back in 1956 by John McCarthy or professor at Dartmouth for years it was thought that computers would never match the power of the human brain but it has proven to not be the case well back then we did not have enough data and reputation per hour but now with big data coming into existence with the great advent of cheap use artificial intelligence is much possible

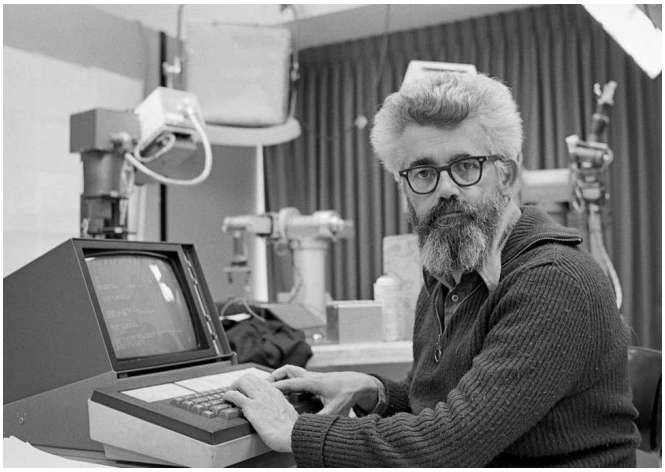


Figure 1: John McCarthy

Now this artificial intelligence machine learning and deep end they all come under the roof of data science well data science is something that has been there for ages and data science is the extraction of knowledge from beta by using different techniques and algorithms. Now artificial intelligence is the technique which enables machine to mimic human behavior and the idea behind here is fairly simple yet fascinating which is to make intelligent machines that can take decisions on its own now machine learning is a subset of artificial intelligence technique which uses statistical methods to enable machines to improve with experience.Now deep learning as we know is a subset of machine

learning which makes the same computation of multi-layer neural network feasible and uses neural networks to stimulate brain like decision making.

The importance of artificial intelligence it automates repetitive learning and discovery through data paper forms frequent high-volume computerized us reliably and without fatigue.Ai adapts through progressive learning algorithms to let the data do the programming the algorithm becomes a classifier or a predictor so just as the algorithm can teach itself how to play any game it can teach itself what product to recommend next online it analyzes more and deeper data using neural networks that have many hidden layers you need lots of data and to Train deep learning models because they learn directly from the data the more data you can feed them the more accurate they become now a achieves incredible accuracy its route to planning neural networks.There are two types of artificial intelligence and the first one is the narrow AI and the second one is the broad Ai.

Narrow a narrow AI is an artificial intelligence system that is designed and trained for one particular task now virtual assistance such as Amazon's Alexi the APUs CV used in Aero AI known area is sometimes also referred to as weak Ai Howard that does not mean that narrow area is inefficient or something of that sort on the contrary it is extremely good at routine jobs both physical and cognitive it is narrow AI that is threatening to replace many human jobs throughout the world. wide AI is that why AI is a system with cognitive abilities so that when the system is presented with an unfamiliar task it is intelligent enough to find a solution now here the system is capable of having intelligent behavior across a variety of tasks from driving a car to telling a joke and the techniques aim at replicating and surpassing many capabilities of human intelligence such as risk analysis and other cognitive processes the robots today are machine learning enabled tools that provide doctors with extended precision and control now these machines enables shortening the patient's hospital stay positive affecting the surgical experience and reducing the medical cost all at once different domains of artificial intelligence first of all we have neural networks so neural networks are a class of models within the general machine learning literature and they are specific set of algorithms that have revolutionized machine learning and artificial intelligence

1.3 What is Machine Learning?

Machine Learning is a concept which allows the machine to learn from examples and experience, and that too without being explicitly programmed. So instead of you writing the code, what you do is you feed data to the generic algorithm, and the algorithm/ machine builds the logic based on the given data.

1.4 Types of Machine Learning

Machine learning is sub-categorized to three

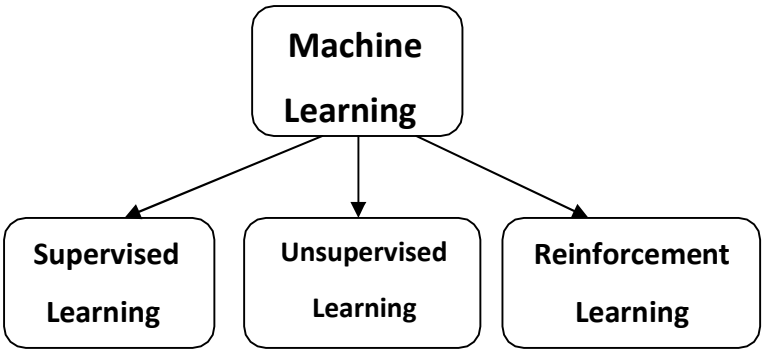


Fig:1.2 Types of Machine Learning

- **What is Supervised Learning?**

Supervised Learning is the one, where you can consider the learning is guided by a teacher. We have a dataset which acts as a teacher and its role is to train the model or the machine. Once the model gets trained it can start making a prediction or decision when new data is given to it.

- **What is Unsupervised Learning?**

The model learns through observation and finds structures in the data. Once the model is given a dataset, it automatically finds patterns and relationships in the dataset by creating clusters in it. What it cannot do is add labels to the cluster, like it cannot say this a group of apples or mangoes, but it will separate all the apples from mangoes.

- **What is Reinforcement Learning?**

It is the ability of an agent to interact with the environment and find out what is the best outcome. It follows the concept of hit and trial method. The agent is rewarded or penalized with a point for a correct or a wrong answer, and on the basis of the positive reward points gained the model trains itself. And again once trained it gets ready to predict the new data presented to it.

- **What is Deep Learning?**

Deep learning is one of the only methods by which we can overcome the challenges of feature extraction. This is because deep learning models are capable of learning to focus on the right features by themselves, requiring little guidance from the programmer. Basically, deep learning mimics the way our brain functions i.e. it learns from experience. As you know, our brain is made up of billions of neurons that allow us to do amazing things. Even the brain of a one year old kid can solve complex problems which are very difficult to solve even using super-computers. For example:

- Recognize the face of their parents and different objects as well.

- Discriminate different voices and can even recognize a particular person based on his/her voice.

- Draw inference from facial gestures of other persons and many more.

Actually, our brain has sub-consciously trained itself to do such things over the years. Now, the question comes, how deep learning mimics the functionality of a brain? Well, deep learning uses the concept of artificial neurons that functions in a similar manner as the biological neurons present in our brain. Therefore, we can say that Deep Learning is a

subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks

- **How Deep Learning Works?**

In an attempt to re-engineer a human brain, Deep Learning studies the basic unit of a brain called a brain cell or a neuron. Inspired from a neuron an artificial neuron or a perception was developed. Now, let us understand the functionality of biological neurons and how we mimic this functionality in the perception or an artificial neuron:

If we focus on the structure of a biological neuron, it has dendrites which are used to receive inputs. These inputs are summed in the cell body and using the Axon it is passed on to the next biological neuron as shown in the above image.

Similarly, a perception receives multiple inputs, applies various transformations and functions and provides an output.

As we know that our brain consists of multiple connected neurons called neural network, we can also have a network of artificial neurons called perceptrons to form a Deep neural network. So, let's move ahead in this Deep Learning Tutorial to understand how a Deep neural network looks like.

Any Deep neural network will consist of three types of layers:

- **The Input Layer**

the first layer is the input layer which receives all the inputs and the last layer is the output layer which provides the desired output.

- **The Hidden Layer**

All the layers in between these layers are called hidden layers. There can be n number of hidden layers thanks to the high end resources available these days.

- **The Output Layer**

The number of hidden layers and the number of perceptrons in each layer will entirely depend on the use-case you are trying to solve.

1.5 Relation between Ai, Machine learning and Deep learning

Artificial Intelligence makes it possible for the machines to learn from their experience. The machines adjust their response based on new inputs thereby performing human-like tasks by processing large amounts of data and recognizing patterns in them.

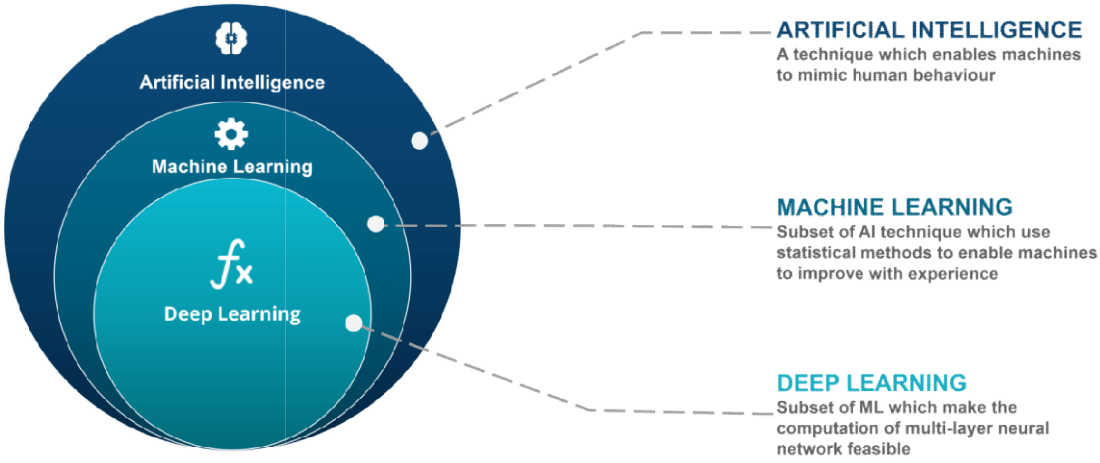


Figure -1.3 the relation between AI, Machine learning and Deep learning

The first church took generations to finish, so most of the workers working on it never saw the final outcome. Those working on it took pride in their craft, building bricks and chiselling stones that were to be placed into the Great Structure. So, as AI researchers, we should think ourselves as humble brick makers, whose job it is to study how to build components (e.g. parsers, planners, learning algorithms, etc) that someday someone, somewhere, will integrate into intelligent systems.

Some of the examples of Artificial Intelligence from our day to day life are Apple's Siri, the chess-playing computer, tesla's self-driving car and many more. These examples are based on deep learning and natural language processing.

Machine Learning is a subset of artificial intelligence. It allows the machines to learn and make predictions based on its experience (data).

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts or abstraction.

The concept of deep learning is not new. But recently its hype has increased, and deep learning is getting more attention. This field is a special kind of machine learning which is inspired by the functionality of our brain cells called artificial neural network. It simply takes data connections between all artificial neurons and adjusts them according to the data pattern. More neurons are needed if the size of the data is large. It automatically features learning at multiple levels of abstraction thereby allowing a system to learn complex functions mapping without depending on any specific algorithm.

1.6 Advantage of Deep learning over Machine learning

- Reduction in Human Error

  The phrase "human error" was born because humans make mistakes from time to time. Computers however, do not make these mistakes if they are programmed properly. With Artificial intelligence, the decisions are taken from the previously gathered information applying certain set of algorithms. So errors are reduced and the chance of reaching accuracy with a greater degree of precision is a possibility.

  Example: In Weather Forecasting using AI they have reduced majority of human error.

- Takes risks instead of Humans

This is one of the biggest advantage of Artificial intelligence. We can overcome many risky limitations of human by developing an AI Robot which in turn can do the risky things for us. Let it be going to mars, defuse a bomb, explore the deepest parts of oceans, minning for coal and oil, it can be used effectively in any kinds of natural or man made disasters.

Example: Have you heard about the Chernobyl nuclear power plant explosion in Ukraine? At that time there were no AI powered robots which can help us to minimise the affect of radiation by controlling the fire in early stages, as any human went close to the core was dead in matter of minutes. They eventually poured sand and boron from helicopters from a mere distance. AI Robots can be used in such situations where a human intervention can be hazardous.

- Available 24×7

An Average human will work for 4-6 hours a day excluding the breaks. Humans are built in such a way to get some time out for refreshing themselves and get ready for a new day of work and they even have weekly off's to stay intact with their work life and personal life. But using AI we can make machines work 24×7 without any breaks and they don't even get bored unlike humans.

Example: Educational Institutes and Helpline centers are getting many queries and issues which can be handled effectively using AI.

- Helping in Repetitive Jobs

In our day-to-day work, we will be performing many repetitive works like sending a thanking mail, verifying certain documents for errors and many more things. Using artificial intelligence we can productively automate these mundane tasks and can even remove "boring" tasks for humans and free them up to be increasingly creative.

Example: In banks, we often see many verifications of documents in order to get a loan which is a repetitive task for the owner of the bank. Using AI Cognitive Automation the owner can speed up the process of verifying the documents by which both the customers and the owner will be benefited.

- Digital Assistance

  Some of the highly advanced organizations use digital assistants to interact with users which saves the need of human resource. The digital assistant also used in many websites to provide things that user want. We can chat with them about what we are looking for. Some chatbots are designed in such a way that it becomes hard to determine that we're chatting with a chatbot or a human being.

  Example: We all know that organizations have a customer support team which needs to clarify the doubts and queries of the customers. Using AI the organizations can set up a Voicebot or Chatbot which can help customers with all their queries. We can see many organizations already started using them in their websites and mobile applications.

- Faster Decisions:

  Using AI alongside other technologies we can make machines take decisions faster than a human and carry out actions quicker. While taking a decision human will analyze many factors both emotionally and practically but AI-powered machine works on what it is programmed and delivers the results in a faster way.

  Example: We all have played a Chess game in Windows. It is nearly impossible to beat CPU in the hard mode because of the AI behind that game. It will take the best possible step in a very short time according to the algorithms used behind it.

- Daily Applications

  Daily applications such as Apple's Siri, Window's Cortana, Google's OK Google are frequently used in our daily routine whether it is for searching a location, taking a selfie, making a phone call, replying to a mail and many more.

  Example: Around 20 years ago, when we are planning to go somewhere we used to ask a person who already went there for the directions. But now all we have to do is

say “OK Google where is Visakhapatnam”. It will show you Visakhapatnam’s location on google map and best path between you and Visakhapatnam.

- New Inventions:

AI is powering many inventions in almost every domain which will help humans solve the majority of complex problems.

1.7 Limitation of Deep learning

In deep learning, everything is a vector, i.e. everything is a point in a geometric space. Model inputs (it could be text, images, etc) and targets are first "vectorized", i.e. turned into some initial input vector space and target vector space. Each layer in a deep learning model operates one simple geometric transformation on the data that goes through it. Together, the chain of layers of the model forms one very complex geometric transformation, broken down into a series of simple ones. This complex transformation attempts to maps the input space to the target space, one point at a time. This transformation is parameterized by the weights of the layers, which are iteratively updated based on how well the model is currently performing. A key characteristic of this geometric transformation is that it must be differentiable, which is required in order for us to be able to learn its parameters via gradient descent. Intuitively, this means that the geometric morphing from inputs to outputs must be smooth and continuous—a significant constraint.

The whole process of applying this complex geometric transformation to the input data can be visualized in 3D by imagining a person trying to uncouple a paper ball: the crumpled paper ball is the manifold of the input data that the model starts with. Each movement operated by the person on the paper ball is similar to a simple geometric transformation operated by one layer. The full uncoupling gesture sequence is the complex transformation of the entire model. Deep learning models are mathematical machines for uncoupling complicated manifolds of high-dimensional data.

That's the magic of deep learning: turning meaning into vectors, into geometric spaces, then incrementally learning complex geometric transformations that map one space to another. All you need are spaces of sufficiently high dimensionality in order to capture the full scope of the relationships found in the original data.

The space of applications that can be implemented with this simple strategy is nearly infinite. And yet, many more applications are completely out of reach for current deep learning techniques—even given vast amounts of human-annotated data. Say, for instance, that you could assemble a dataset of hundreds of thousands—even millions—of English language descriptions of the features of a software product, as written by a product manager, as well as the corresponding source code developed by a team of engineers to meet these requirements. Even with this data, you could not train a deep learning model to simply read a product description and generate the appropriate codebase. That's just one example among many. In general, anything that requires reasoning—like programming, or applying the scientific method—long-term planning, and algorithmic-like data manipulation, is out of reach for deep learning models, no matter how much data you throw at them. Even learning a sorting algorithm with a deep neural network is tremendously difficult.

This is because a deep learning model is "just" a chain of simple, continuous geometric transformations mapping one vector space into another. All it can do is map one data manifold X into another manifold Y, assuming the existence of a learnable continuous transform from X to Y, and the availability of a dense sampling of X:Y to use as training data. So even though a deep learning model can be interpreted as a kind of program, inversely most programs cannot be expressed as deep learning models—for most tasks, either there exists no corresponding practically-sized deep neural network that solves the task, or even if there exists one, it may not be learnable, i.e. the corresponding geometric transform may be far too complex, or there may not be appropriate data available to learn it.

Scaling up current deep learning techniques by stacking more layers and using more training data can only superficially palliate some of these issues. It will not solve the more fundamental problem that deep learning models are very limited in what they can represent, and that most of the programs that one may wish to learn cannot be expressed as a continuous geometric morphing of a data manifold.

- The risk of anthropomorphizing machine learning models

One very real risk with contemporary AI is that of misinterpreting what deep learning models do, and overestimating their abilities. A fundamental feature of the human mind is our "theory of mind", our tendency to project intentions, beliefs and knowledge on the things around us. Drawing a smiley face on a rock suddenly makes it "happy"—in our minds. Applied to deep learning, this means that when we are able to somewhat successfully train a model to generate captions to describe pictures, for instance, we are led to believe that the model "understands" the contents of the pictures, as well as the captions it generates. We then proceed to be very surprised when any slight departure from the sort of images present in the training data causes the model to start generating completely absurd captions.

1.8 Python as a programming language

- Python In Artificial Intelligence

One of the key features of python language is its simplicity in code writing. It uses 1/5th of the code when compared to other object oriented programs. This factor makes it the most sort after language used in trending domains like AI. AI has a wide horizon under which it deals with machine learning and deep learning.

Python has a variety of libraries that appeal to the needs of any programmer. It has some prebuilt libraries such as Numpy, SciPy, Pybrain etc., which are for advance and scientific computing. Python is platform independent, which makes it quite flexible in interfacing between other technologies. In addition, the current user base of the language is very diverse. Most python developers share queries and solutions on portals, which make it a comprehensive knowledge resource as well.

The language not only applies OOPs concepts but incorporates a scripting approach as well. There are numerous IDEs (Integrated Development Environment) like PyCharm, which allows users to carry out complex codes and algorithms of AI related projects. In an AI's SDLC (Software Development Life Cycle) phase like testing, debugging and

development, it becomes a cakewalk, when compared against other contemporary programming languages like Java, Javascript and Pearl.

These languages would definitely yield desired results but would make tasks cumbersome. Hence, looking at the numerous advantages of python, there is no doubt that it plays a crucial aspect in today's AI technologies.

- Deep Learning In Python

Deep learning is another trending domain in today's' world of Artificial Intelligence. Deep learning techniques are so powerful because they represent and learn of how to solve a problem in the best possible way. This is called as "Representation Learning". The deep learning programs are trained with numerous examples that make its predictions accurate. Deep learning models are extensively used in colorizing images and videos. It is used in identification of objects in photographs popularly termed as 'face-recognition'.

Python is the best platform to get started with deep learning models. Python is quick and easy to understand. It has a ton of features that make deep learning projects faster to operate and develop. The two most versatile libraries used by any deep learning expert is "Theano" and "Tensorflow". These are quiet technical and used exhaustively by research groups. The "Keras" library is written in pure python which provides an interface for the above two libraries.

1.9 Python Packages:

Now that we know how important Python is to the world and us. Let us deep dive into learning some of the technical aspects of the programming language. The below illustrated topics are rudimentary and would be easy to grasp.

- Python use in machine learning

Machine Learning, as the name suggests, is the science of programming a computer by which they are able to learn from different kinds of data. A more general definition given by Arthur Samuel is – "Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed." They are typically used to solve various types of life problems.

In the older days, people used to perform Machine Learning tasks by manually coding all the algorithms and mathematical and statistical formula. This made the process time consuming, tedious and inefficient. But in the modern days, it is become very much easy and efficient compared to the olden days by various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reason is its vast collection of libraries. Python libraries that used in Machine Learning are:

- Numpy
- Scipy
- Scikit-learn
- Pandas
- Matplotlib
- OpenCV

- NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

```
# Python program using NumPy # for some basic mathematical # operations
import numpy as np
# Creating two arrays of rank 2 x = np.array([[1, 2], [3, 4]])
 y = np.array([[5, 6], [7, 8]])
# Creating two arrays of rank 1 v = np.array([9, 10])
w = np.array([11, 12])
# Inner product of vectors print(np.dot(v, w), "\n")
# Matrix and Vector product print(np.dot(x, v), "\n")
# Matrix and matrix product print(np.dot(x, y))
```

```
Output:
219

[29 67]

[[19 22]
 [43 50]]
```

Figure 1.4 :The output of above program

- SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

```
# Python script using Scipy # for image manipulation
from scipy.misc import imread, imsave, imresize
 # Read a JPEG image into a numpy array
img = imread('D:/Programs / cat.jpg') # path of the image print(img.dtype, img.shape)

# Tinting the image
```

```
img_tint = img * [1, 0.45, 0.3]

# Saving the tinted image
imsave('D:/Programs / cat_tinted.jpg', img_tint)

# Resizing the tinted image to be 300 x 300 pixels img_tint_resize = imresize(img_tint, (300, 300))

# Saving the resized tinted image
imsave('D:/Programs / cat_tinted_resized.jpg', img_tint_resize)
```

output:

Original image:

Tinted image:

Resized tinted image:

Figure 1.5 output of above program

- Skikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with ML.

```
# Python script using Scikit-learn # for Decision Tree Clasifier

# Sample Decision Tree Classifier from sklearn import  datasets from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier

# load the iris datasets dataset = datasets.load_iris()

# fit a CART model to the data model = DecisionTreeClassifier()
model.fit(dataset.data, dataset.target) print(model)

# make predictions expected = dataset.target
predicted = model.predict(dataset.data)

# summarize the fit of the model print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
DecisionTreeClassifier(class_weight=None, criterion="gini",
max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False,
random_state=None,
            splitter="best")
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        50
           2       1.00      1.00      1.00        50

   micro avg       1.00      1.00      1.00       150
   macro avg       1.00      1.00      1.00       150
weighted avg       1.00      1.00      1.00       150

[[50  0  0]
 [ 0 50  0]
```

Figure 1.6 :output of the above program

- Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

```
# Python program using Pandas for # arranging a given set of data
# into a  table

# importing pandas as pd import pandas as pd

data = {"country": ["Brazil", "Russia", "India", "China", "South Africa"],
"capital": ["Brasilia", "Moscow", "New Dehli", "Beijing", "Pretoria"],
"area": [8.516, 17.10, 3.286, 9.597, 1.221],
```

"population": [200.4, 143.5, 1252, 1357, 52.98] }

data_table = pd.DataFrame(data) print(data_table)

Output:

	country	capital	area	population
0	Brazil	Brasilia	8.516	200.40
1	Russia	Moscow	17.100	143.50
2	India	New Dehli	3.286	1252.00
3	China	Beijing	9.597	1357.00
4	South Africa	Pretoria	1.221	52.98

Figure 1.7: output of above program

- Matpoltlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar charts, etc.

Python program using Matplotlib # for forming a linear plot

# importing the necessary packages and modules import matplotlib.pyplot as plt
import numpy as np

# Prepare the data
x = np.linspace(0, 10, 100)

# Plot the data
plt.plot(x, x, label ='linear')

Add a legend plt.legend()

Show the plot plt.show()

Output:

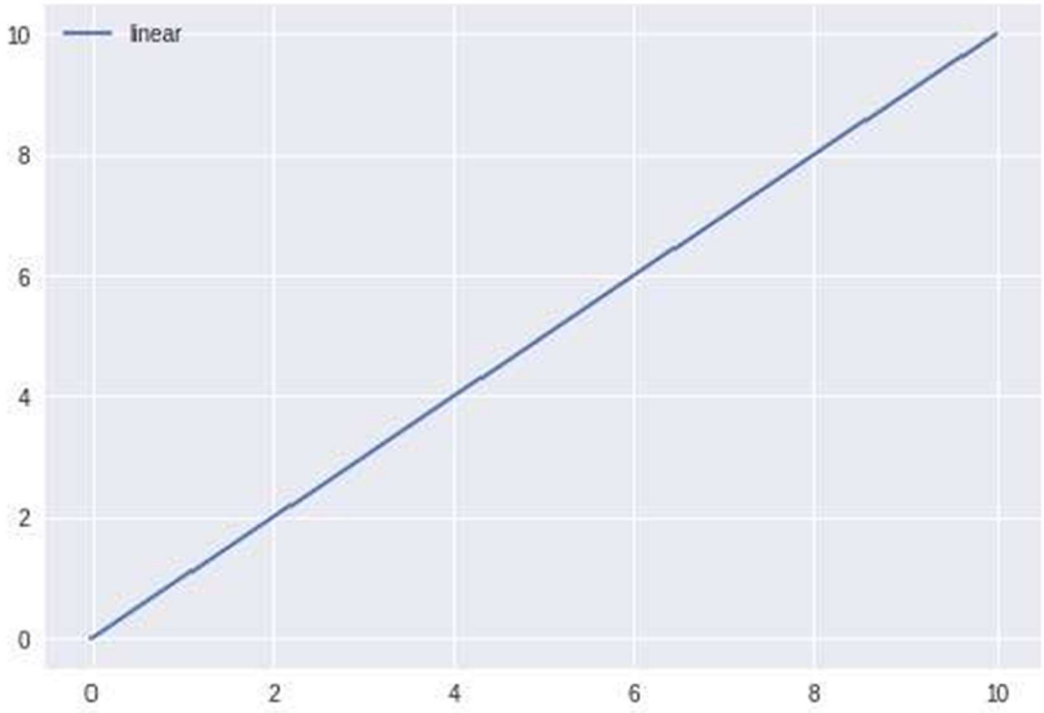


Figure 1.8: output of above program

- OpenCV is a Python library which is designed to solve computer vision problems. OpenCV was originally developed in 1999 by Intel but later it was supported by Willow Garage. OpenCV supports a wide variety of programming languages such as C++, Python, Java etc. Support for multiple platforms including Windows, Linux, and MacOS.OpenCV Python is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays.This makes it easier to integrate it with other libraries which use NumPy. For example, libraries such as SciPy and Matplotlib.

```
Import cv2

# colored Image
```

```
Img = cv2.imread (&ldquo;Penguins.jpg&rdquo;,1)


# Black and White (gray scale)

Img_1 = cv2.imread (&ldquo;Penguins.jpg&rdquo;,0)
```

1.10 Packages use in project:

1.10.1 NUMPY :

NumPy – which stands for numerical Python; It is a library consisting of multidimensional, array objects and collection of routines on processing those always. NumPy helps to perform mathematical and logical operation on array.

Operation using NumPy:

- Mathematical and logical on array.
- Furrier transforms and routines for shape manipulation.

- Operation related to linear algebra and in build functions for linear Algebra and random number generation..NumPy is often used along with packages like spicy (scientific Python) and MatPlotlib (Plotting library) .Import NumPy as me the most import object define in NumPy is an n dimensional array type called an array.

import NumPy a np

a = np.array ([1, 2, 3]) Print a

Output: -

[1, 2, 3]

import NumPy as np

a= np.array ([[12, 3],[4,5,6]])

Print a.Shape

Output:

(2, 3)

1.10.2 PANDAS :

Pandas is an open source library that allows to you performs data manipulation. Pandas library is built on top of NumPy. Pandas need NumPy to operands, it's provided and easy way to create manipulates the data. It uses series on 1D data structure and DataFrame for multidimensional data structure. It provided slides the data, merge, concatenate, reshape the data.

- What is DataFrame?

  A DataFrame is a 2D array with labels 2D access (row and Colum). It is standard way to data.

	Item	Price
0	A	2
1	B	3

- **Create a data frame**

We can convert a numpy array to a pandas DataFrame with pd.DataFrame( ) . Here opposite is also possible, using np.array( ) that we can perform in a operation.

- **Numpy to pandas**

Import numpy as np

h = [[1, 2], [3,4]]

table_h = pd.DataFrame(h)

print('DataFrame : ' table_h)

- **Pandas to Numpy**

array_h = np.array(table_h)

print('Numpy_array : ' array_h)

- **OUTPUT:**

	0	1
0	1	2
1	3	4

dataframe

Numpy array : [[1, 2] , [3, 4]]

- We have use directory to create a pandas DataFrame :

Dic = { ‘Name’ : [“JOHN”, “SMITH”], ‘AGE’ : [30, 40]}

Pd.DataFrame (Data = Dic)

- **OUTPUT:**

	AGE	NAME
0	30	JOHN
1	40	SMITH

- **Range data :**

- **Create data :**

Date_D = PD.Date_range (‘2030 01 01’, priodes = 6, freq = ‘D’ Print (‘Day : ‘. Dates_D)

- **OUTPUT**:

Day : DatetimeIndex ([‘2030-01-01’, ‘2030-01-02’, ‘2030-01-03’, ‘2030-01-04’, ‘2030-01-05’, ‘2030-01-06’], Dtye = ‘datatime64 [ns]’, freq = ‘D’)

- **Inspecting data :-**

We can check the head or tail of the dataset

Step 1 : Create random, sequence with numpy. The sequence has 4 columns and 6 rows.

Randon = np.random . rand n (6,4)

Step 2 : Then create a data frame using pandas ; Use data_m as an index for the DataFrame. It means each row will be given an name or index corresponding to a date.

- **CRATE WITH DATE:**

df = pd.DataFrame (random, index = date_m, coloms = list (‘ABCD’))

Step 3 : df .head (3)

- **OUTPUT:**

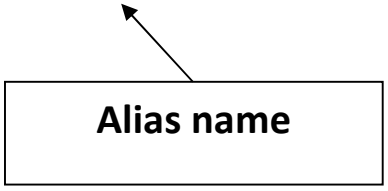
	A	B	C
2030-01-31	1.3	1.31	1.312

2030-02-28	2	4	6
2030-03-31	7	8	9

- **1.10.3 Matplotlib :-**

It is a plotting library for the python programming language and it's numerical mathematics extension. Numpy using these package we can generates plots, histograms, power spectra, bar charts, error charts, scatter diagram etc.

Import matplotlib . pyplot as plt



- **The code will be:**

```
X_values = [0, 1, 2, 3, 4, 5]

Squares = [0, 1, 4, 9, 16, 25]

 Plt.plot (x_values, squares)

 Plt.show ( )
```

- **Bar graph :-**

```
X_values = [50, 60, 10, 70, 30]

Y_values = [ “a”, “b”, “c”, “d”, “e” ]

Plt.bar (Y_values, X_values, color = “Red” )

Plt.show ( )
```

- **Different between Seaborn and matplotlib :-**

  Seaborn is complimentary to matplotlib and it specially tergates statistical data visualization. Seaborn extends matplotlib and that's why it cab address, limitation of matplotlib, seaborn work with different parameter. The matplotlib defaults that usually don’t speak to users are the colors, tick marks, styles.To load dataset in seaborn we need to use –

  load_dataset ( ) method.

- **HEATMAP IN PYTHON:**

A heatmap is a 2D graphical representation of data where the individual values that are containing in are representated as columns. It represents different colour coding to represents different values the heatmaps can be use to show where users have clicked on a page, how far they have scroll down a page.

A correlation heatmap uses colour cells, typically in a monochromatic scale, to show a 2D correlation matrix, the colour value of the cells is proportional to the number of measurement that match the dimension values.

- **Python Heatmap Code**

We will create a seaborn heatmap for a group of 30 Pharmaceutical Company stocks listed on the National Stock Exchange of India Ltd (NSE). The seaborn heatmap will display the stock symbols and its respective single-day percentage price change.

We collate the required market data on Pharma stocks and construct a comma-separated value (CSV) file comprising of the stock symbols and their respective percentage price change in the first two columns of the CSV file.

Since we have 30 Pharma companies in our list, we will create a heatmap matrix of 6 rows and 5 columns. Further, we want our seaborn heatmap to display the percentage price change for the stocks in a descending order. To that effect, we arrange the stocks in a descending order in the CSV file and add two more columns which indicate the position of each stock on X & Y axis of our heatmap.

- **Import the required Python packages**

We import the following Python packages:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

- **Load the dataset**

We read the dataset using the read_csv function from pandas and visualize the first ten rows using the print statement.

Create a Python Numpy array

Since we want to construct a 6 x 5 matrix, we create an n-dimensional array of the same shape for " ymbol" and the "Change" columns.

Create a Pivot in Python

The pivot function is used to create a new derived table from the given data frame object "df". The function takes three arguments; index, columns, and values. The cell values of the ne table are taken from column given as the values parameter, which in our case is the "Change" column.

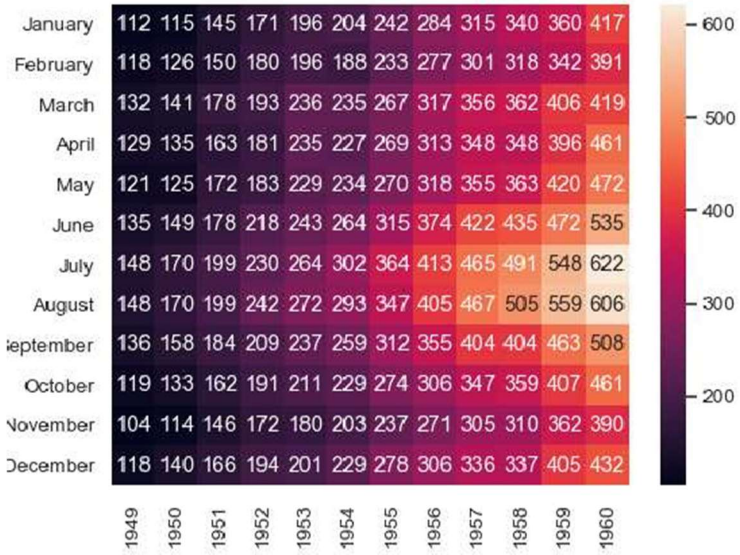
Create an Array to Annotate the Heatmap

In this step, we create an array which will be used to annotate te seaborn heatmap. We call the flatten method on the "symbol" and "percentage" arrays to flatten a Python list of lists in one line. The zip function which returns an iterator zips a list in Python. We run a Python For loop and by using the format function; we format the stock symbol and the percentage price change value as per our requirement.

Create the Matplotlib figure and define the plot.

We create an empty Matplotlib plot and define the figure size. We also add the title to the plot and s t the title's font size, and its distance from the plot using the set_position method.

We wish to display only the stock symbols and their respective single-day percentage price change. Hence, we hide the ticks for the X & axis, and also remove both the axes from the heatmap plot.



- **1.10.4 Sk_learn :-**

  scikit – learn is an open source python library that implements a range of machine learning algorithms

- Simple tool for datamining and analysis :- It's features various classification, regration, clustering algorithms, including support vector machine, Random forests, gradient boosting, K- means algorithms.
- It works with NUMPY, SYPY the easiest way to install scikit learn.

```
pip install _ - U_scikit_learn
```

- Load the dataset, a dataset is nothing but a collection of data, a dataset has 2 main components -
  - **Features**
  - **Response**
- **Features** : They can simply variables of our data, they can be more than one and represented by a feature matrix
- **Response** : This is the output variable depending on the feature variable.

// Load the iris dataset as an example.

```
from _sklearn . datasets_import load_iris
```

```
Iris = load_iris ( )
```

// store the feature matrix and respose vector

```
X = iris . daya # we are saving the feature to x
```

```
Y = iris . target # response
```

Store the feature and target names,

feature_names = iris .feature_names

target _names = iris. target _names

print(type (x))

print (X [: 5])

1.11 PROBLEM DOMAIN:-

Assuming to the problem domain we can see from the above discussion of the project(Face swapping using AI) that is the output of our code is not only depend upon how much accuracy tool/method/algorithm we should use in our code.

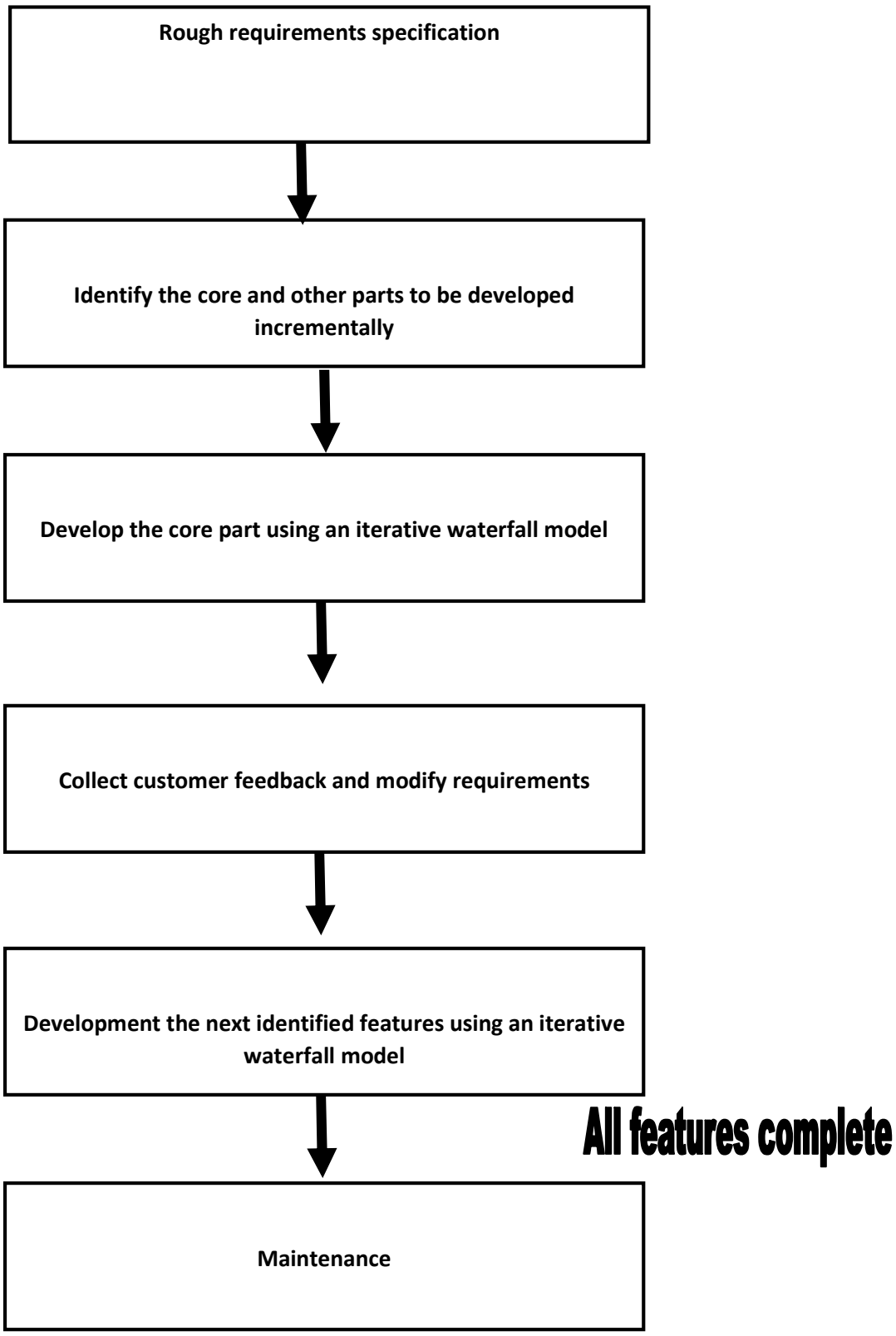
First of all we have to input those images which consist clear, noise less, undistorted image. And then we have to find the dlib 68’ points of each faces and use swap and triangle wrapping method to get perfect output image.

1.12 SOLUTION DOMAIN:-

- **Evolutionary Model**

This model has many of the features of the incremental model. As in case of the increment model, the software is developed over a number of increments. At each increment, a concept (feature) is implemented and is deployed at the client site. The software is successively refined and feature-enriched until the full software is realized. The principal idea behind the evolutionary life cycle model is conveyed by its name. In the incremental development model, complete requirements are first developed and the SRS document prepared. In contrast, in the evolutionary model, the requirements, plan, estimates, and solution evolve over the iterations, rather than fully defined and frozen in a major up-front specification effort before the development iterations begin. Such evolution is consistent with the pattern of unpredictable feature discovery and feature changes that take place in new product development.

Though the evolutionary model can also be viewed as an extension of the waterfall model, but it incorporates a major paradigm shift that has been widely adopted in many recent life cycle models. Due to obvious reasons, the evolutionary software development process is sometimes referred to as design a little, build a little, test a little deploy a little model. This means that after the requirements have been specified, the design, build, test, and deployment activities are iterated. A schematic representation of the evolutionary mode of development has been shown in Figure.


Rough requirements specification
Identify the core and other parts to be developed incrementally
Develop the core part using an iterative waterfall model
Collect customer feedback and modify requirements
Development the next identified features using an iterative waterfall model
All features complete
Maintenance

- ## Advantages

The evolutionary model of development has several advantages. Two important advantages using this model are the following:

- **Effective elicitation of actual customer requirements:**

In this model, the user gets a chance to experiment with partially developed software much before the complete requirements are developed. Therefore, the evolutionary model helps to accurately elicit user requirements with the help of feedback obtained on the delivery of different versions of the software. As a result, the change requests after delivery of the complete software gets substantially reduced.

- **Easy handling change requests:**

In this model, handling change requests is easier as no long-term plans are made. Consequently, reworks required due to change requests are normally much smaller compared to the sequential models.

- ## Disadvantages

The main disadvantages of the successive versions model are as follows:

- **Feature division into incremental parts can be non-trivial:**

For many development projects, especially for small-sized projects, it is difficult to divide the required features into several parts that can be incrementally implemented and delivered. Further, even for larger problems, often the features are so intertwined and dependent on each other that even an expert would need considerable effort to plan the incremental deliveries.

- **Ad hoc design:**

Since at a time design for only the current increment is done, the design can become ad hoc without specific attention being paid to maintainability and optimality.
Obviously, for moderate sized problems and for those for which the customer requirements are clear, the iterative waterfall model can yield a better solution.

Evolutionary versus incremental model of developments

The evolutionary and incremental have several things in common, such as incremental development and deployment at the client site. However, in a purely incremental model, the requirement specification is completed before any development activities start. Once the requirement specification is completed, the requirements are split into requirements. In a purely evolutionary development, the development of the first version starts off after obtaining a rough understanding of what is required. As the development proceeds, more and more requirements emerge. The modern development models, such as the agile mode e neither purely incremental, nor purely evolutionary, but is somewhat in between and is referred to as incremental and evolutionary model. In this are obtained and specified, but requirements that emerge later are accommodated.

Chapter-2

2.1. DATA ANALYSIS USING AI

As the word suggests Data Analytics refers to the techniques to analyze data to enhance productivity and business gain. Data is extracted from various sources and is cleaned and categorized to analyze different behavioral patterns. The techniques and the tools used vary according to the organization or individual.

2.1.1. Why is Data Analytics important?

As an enormous amount of data gets generated, the need to extract useful insights is a must for a business enterprise. Data Analytics has a key role in improving your business. Here are 4 main factors which signify the need for Data Analytics:

- Gather Hidden Insights – Hidden insights from data are gathered and then analyzed with respect to business requirements.
- Generate Reports – Reports are generated from the data and are passed on to the respective teams and individuals to deal with further actions for a high rise in business.
- Perform Market Analysis – Market Analysis can be performed to understand the strengths and the weaknesses of competitors.
- Improve Business Requirement – Analysis of Data allows improving Business to customer requirements and experience.

2.1.2. Top Tools in Data Analytics

With the increasing demand for Data Analytics in the market, many tools have emerged with various functionalities for this purpose. Either open-source or user- friendly, the top tools in the data analytics market are as follows.

- **R programming** – This tool is the leading analytics tool used for statistics and data modeling. R compiles and runs on various platforms such as UNIX, Windows, and

Mac OS. It also provides tools to automatically install all packages as per user-requirement.

- **Python** – Python is an open-source, object-oriented programming language which is easy to read, write and maintain. It provides various machine learning and visualization libraries such as Scikit-learn, TensorFlow, Matplotlib, Pandas, Keras etc. It also can be assembled on any platform like SQL server, a MongoDB database or JSON
- **Tableau Public** – This is a free software that connects to any data source such as Excel, corporate Data Warehouse etc. It then creates visualizations, maps, dashboards etc with real-time updates on the web.
- **QlikView –** This tool offers in-memory data processing with the results delivered to the end-users quickly. It also offers data association and data visualization with data being compressed to almost 10% of its original size.
- **SAS** – A programming language and environment for data manipulation and analytics, this tool is easily accessible and can analyze data from different sources.
- **Microsoft Excel** – This tool is one of the most widely used tools for data analytics. Mostly used for clients' internal data, this tool analyzes the tasks that summarize the data with a preview of pivot tables.
- **RapidMiner –** A powerful, integrated platform that can integrate with any data source types such as Access, Excel, Microsoft SQL, Tera data, Oracle, Sybase etc. This tool is mostly used for predictive analytics, such as data mining, text analytics, machine learning.
- **KNIME –** Konstanz Information Miner (KNIME) is an open-source data analytics platform, which allows you to analyze and model data. With the benefit of visual programming, KNIME provides a platform for reporting and integration through its modular data pipeline concept.
- **OpenRefine –** Also known as GoogleRefine, this data cleaning software will help you clean up data for analysis. It is used for cleaning messy data, the transformation of data and parsing data from websites.
- **Apache Spark** – One of the largest large-scale data processing engines, this tool executes applications in Hadoop clusters 100 times faster in memory and 10 times faster on disk. This tool is also popular for data pipelines and machine learning model development.

2.2 KNN Algorithm

K nearest neighbors or KNN Algorithm is a simple algorithm w ich uses the entire dataset in its training phase. Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction.

How does a KNN Algorithm work?

The k-nearest neighbors algorithm uses a very simple approach to perform classification. When tested with a new example, it looks through the training data and finds the k training examples that are closest to the new example. It then assigns the most common class label (among those k-training examples) to the test example.

What does 'k' in KNN Algorithm represent?

k in kNN algorithm represents the number of nearest neighbor points which are voting for the new test data's class.

If k=1, then test examples are given the same label as the closest example in the training set.

If k=3, the labels of the three closest classes are checked and the most common (i.e., occurring at least twice) label is assigned, and so on for larger k.

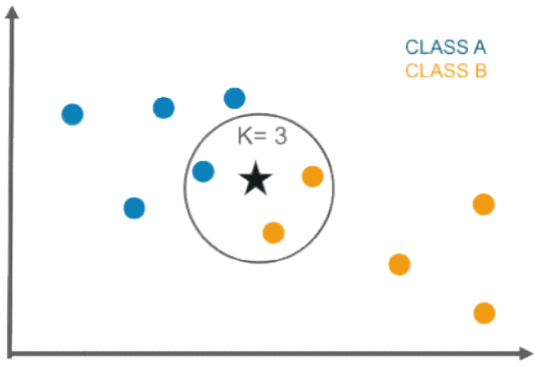


Figure 2.1 kNN Algorithm represent

- **KNN Algorithm Manual Implementation**

  Step1: Calculate the Euclidean distance between the new point and the existing points
  Step 2: Choose the value of K and select K neighbors closet to the new point.
  Step 3: Count the votes of all the K neighbors / Predicting Values

- **Implementation of kNN Algorithm using Python**

- **Handling the data**
  The very first step will be handling the iris dataset. Open the dataset using the open function and read the data lines with the reader function available under the csv module.
- **Calculate the distance**

In order to make any predictions, you have to calculate the distance between the new point and the existing points, as you will be needing k closest points.

- **Find k nearest point**

Now that you have calculated the distance from each point, we can use it collect the k most similar points/instances for the given test data/instance.

- **Predict the class**

Now that you have the k nearest points/neighbors for the given test instance, the next task is to predicted response based on those neighbors

- **Check the accuracy**

  Now that we have all of the pieces of the KNN algorithm in place. Let's check how accurate our prediction.

2.3 DECISION TREE CLASSIFIER

A Decision Tree has many analogies in real life and turns out, it has influenced a wide area of Machine Learning, covering both Classification and Regression. In decision analysis, a decision tree can be used to visually and decisions and decision making. explicitly represent.

- **What is a Decision Tree?**

A decision tree is a map of the possible outcomes of a series of related choices. It allows an individual or organization to weigh possible actions against one another based on their costs, probabilities, and benefits.

As the name goes, it uses a tree-like model of decisions. They can be used either to drive informal discussion or to map out an algorithm that predicts the best choice mathematically.

A decision tree typically starts with a single node, which branches into possible outcomes. Each of those outcomes leads to additional nodes, which branch off into other possibilities. This gives it a tree-like shape.

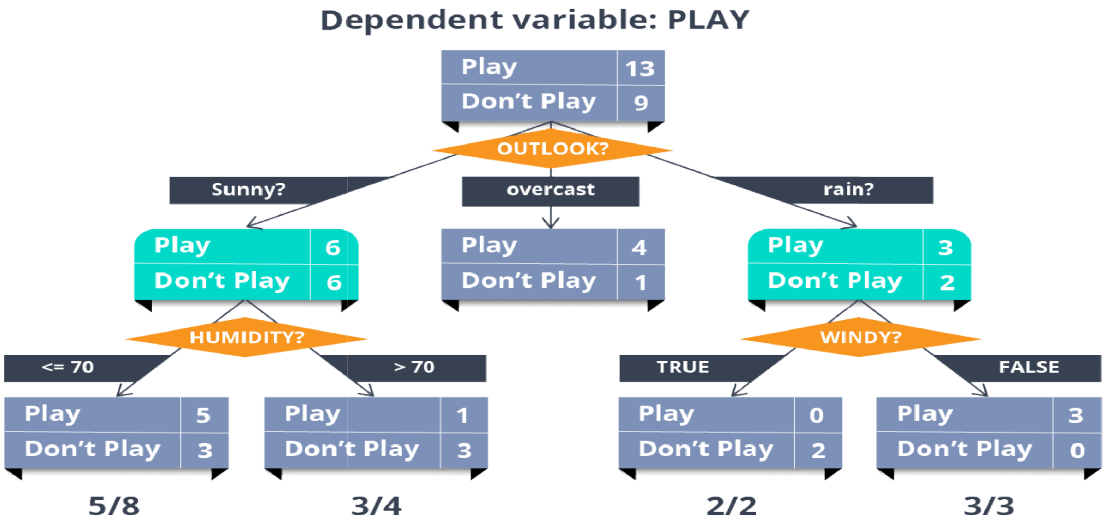


Figure 2.2 decetion tree

There are three different types of nodes: chance nodes, decision nodes, and end nodes. A chance node, represented by a circle, shows the probabilities of certain results. A decision node, represented by a square, shows a decision to be made, and an end node shows the final outcome of a decision path.

- ## Advantages & Disadvantages of Decision Trees

Advantages

- Decision trees generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are capable of handling both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

Disadvantages

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and a relatively small number of training examples.
- Decision trees can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a

- search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

- **Creating a Decision Tree**

Let us consider a scenario where a new planet is discovered by a group of astronomers. Now the question is whether it could be 'the next earth?' The answer to this question will revolutionize the way people live. Well, literally!

There is n number of deciding factors which need to be thoroughly researched to take an intelligent decision. These factors can be whether water is present on the planet, what is the temperature, whether the surface is prone to continuous storms, flora and fauna survives the climate or not, etc.

Let us create a decision tree to find out whether we have discovered a new habitat. The habitable temperature falls into the range 0 to 100 Celsius

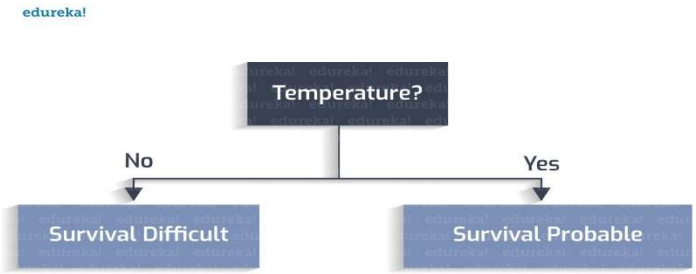


Figure 2.3 : Creating a Decision Tree

- **Whether water is present or not?**

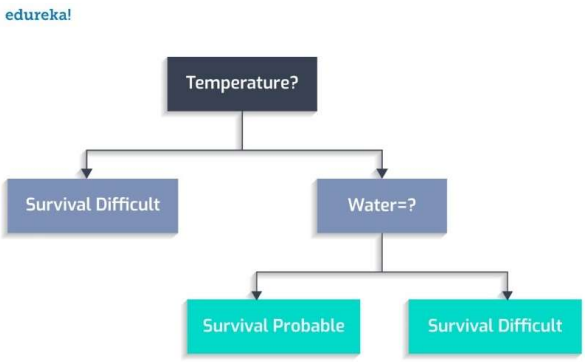


Figure 2.4 : Creating a Decision Tree

- **Whether flora and fauna flourishes?**

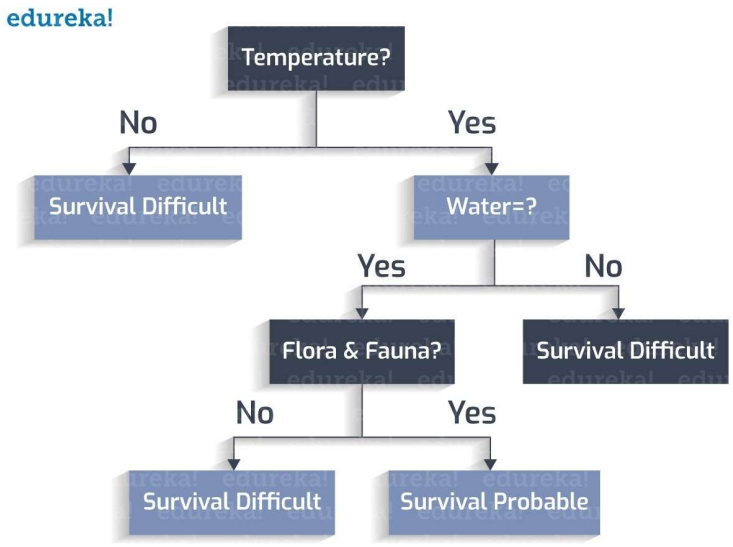


Figure 2.5: Creating a Decision Tree

- **The planet has a stormy surface?**

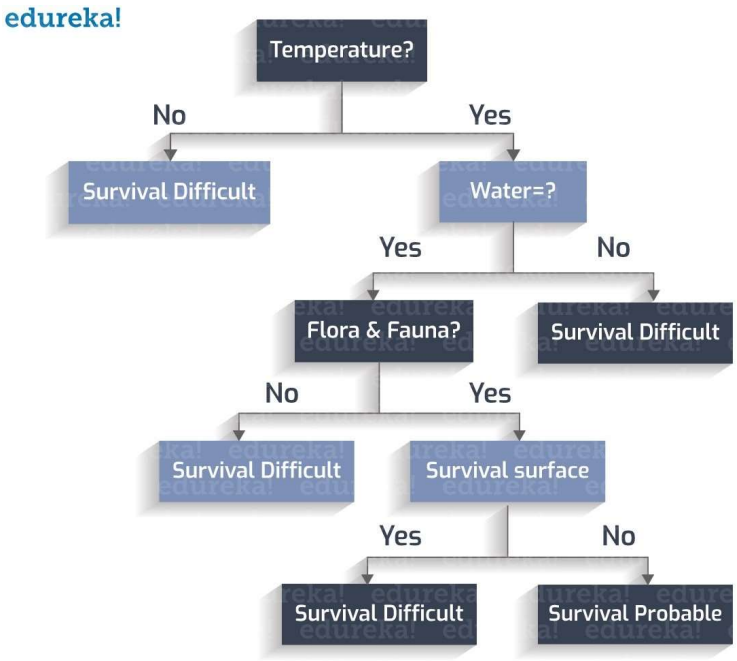


Figure 2.6 : Creating a Decision Tree

- **Classification Rules:**

Classification rules are the cases in which all the scenarios are taken into consideration and a class variable is assigned to each.

Class Variable:

Each leaf node is assigned a class-variable. A class-variable is the final output which leads to our decision.

Let us derive the classification rules from the Decision Tree created:

1. If Temperature is not between 273 to 373K, -> Survival Difficult

2. If Temperature is between 273 to 373K, and water is not present, -> Survival Difficult

3. If Temperature is between 273 to 373K, water is present, and flora and fauna is not present -> Survival Difficult

4. If Temperature is between 273 to 373K, water is present, flora and fauna is present, and a stormy surface is not present -> Survival Probable

5. If Temperature is between 273 to 373K, water is present, flora and fauna is present, and a stormy surface is present -> Survival Difficult

- **Decision Tree**

A decision tree has the following constituents :

- Root Node: The factor of 'temperature' is considered as the root in this case.
- Internal Node: The nodes with one incoming edge and 2 or more outgoing edges.
- Leaf Node: This is the terminal node with no out-going edge.

As the decision tree is now constructed, starting from the root-node we check the test condition and assign the control to one of the outgoing edges, and so the condition is again tested and a node is assigned. The decision tree is said to be complete when all the test conditions lead to a leaf node. The leaf node contains the class-labels, which vote in favor or against the decision.

Now, you might think why did we start with the 'temperature' attribute at the root? If you choose any other attribute, the decision tree constructed will be different.Correct. For a particular set of attributes, there can be numerous different trees created. We need to choose the optimal tree which is done by following an al orithmic approach. We will now see 'the greedy approach' to create a perfect decision tree.

- **Random Forest Classification**

Classification is the method of predicting the class of a given input data point.

Classification problems are common in machine learning and they fall under the Supervised learning method.

Let's say you want to classify your emails into 2 groups, spam and non-spam emails. For this kind of problems, where you have to assign an input data point into different classes, you can make use of classification algorithms.

Under classification we have 2 types:

- Binary Classification
- Multi-Class Classification

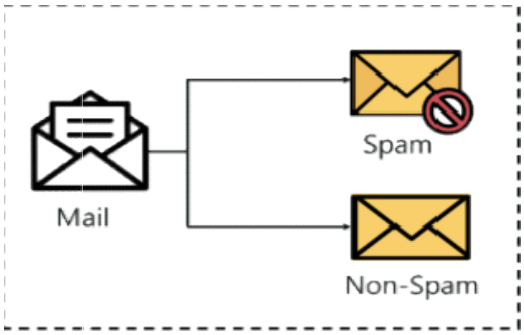


Figure 2.7 : Example

The example that I gave earlier about classifying emails as spam and non-spam is of here we're classifying emails into 2 classes (spam and non-binary type because spam).

But let's say that we want to classify our emails into 3 classes:

- Spam messages
- Non-Spam messages
- Drafts

So here we're classifying emails into more than 2 classes, this is exactly what multi- class classification means.

One more thing to note here is that it is common for classification models to predict a continuous value. But this continuous value represents the probability of a given data point belonging to each output class.

2.4 What Is Random Forest?

Random forest algorithm is a supervised classification and regression algorithm. As the name suggests, this algorithm randomly creates a forest with several trees.

Generally, the more trees in the forest the more robust the forest looks like. Similarly, in the random forest classifier, the higher the number of trees in the forest, greater is the accuracy of the results.

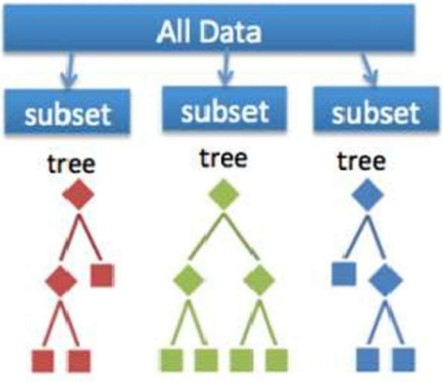


Figure 2.8 : Random Forest

In simple words, Random forest builds multiple decision trees (called the forest) and glues them together to get a more accurate and stable prediction. The forest it builds is a collection of Decision Trees, trained with the bagging method

Before we discuss Random Forest in depth, we need to understand how Decision Trees work.

- **What Is The Difference Between Random Forest And Decision Trees?**

Let's say that you're looking to buy a house, but you're unable to decide which one to buy. So, you consult a few agents and they give you a list of parameters that you should consider before buying a house. The list includes:

- Price of the house
- Locality
- Number of bedrooms
- Parking space
- Available facilities

These parameters are known as predictor variables, which are used to find the response variable. Here's a diagrammatic illustration of how you can represent the above problem statement using a decision tree.

- **Why Use Random Forest?**

You might be wondering why we use Random Forest when we can solve the same problems using Decision trees. Let me explain.

- Even though Decision trees are convenient and easily implemented, they lack accuracy. Decision trees work very effectively with the training data that was used to build them, but they're not flexible when it comes to classifying the new sample. Which means that the accuracy during testing phase is very low.
- This happens due to a process called Over-fitting.
- This means that the disturbance in the training data is recorded and learned as concepts by the model. But the problem here is that these concepts do not

apply to the testing data and negatively impact the model's ability to classify the new data, hence reducing the accuracy on the testing data.

This is where Random Forest comes in. It is based on the idea of bagging, which is used to reduce the variation in the predictions by combining the result of multiple Decision trees on different samples of the data set.

- **How Does Random Forest Work?**

To understand Random forest, consider the below sample data set. In this data set we have four predictor variables, namely:

- Weight
- Blood flow
- Blocked Arteries
- Chest Pain

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Abnormal	No	No	130	No
Normal	Yes	Yes	195	Yes
Normal	No	Yes	218	No
Abnormal	Yes	Yes	180	Yes

Figure 2.9 : Random Forest

These variables are used to predict whether or not a person has heart disease. We're going to use this data set to create a Random Forest that predicts if a person has heart disease or not.

- **Creating A Random Forest**
- **Step 1: Create a Bootstrapped Data Set**

Bootstrapping is an estimation method used to make predictions on a data set by re- sampling it. To create a bootstrapped data set, we must randomly select samples from the original data set. A point to note here is that we can select the same sample more than once.

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	Yes	Yes	195	Yes
Abnormal	No	No	130	No
Abnormal	Yes	Yes	180	Yes
Abnormal	Yes	Yes	180	Yes

Figure 2.10 : Random Forest

In the above figure, I have randomly selected samples from the original data set and created a bootstrapped data set. Simple, isn't it? Well, in real-world problems you'll never get such a small data set, thus creating a bootstrapped data set is a little more complex.

- **Step 2: Creating Decision Trees**

- Our next task is to build a Decision Tree by using the bootstrapped data set created in the previous step. Since we're making a Random Forest we will not consider the entire data set that we created, instead we'll only use a random subset of variables at each step.
- In this example, we're only going to consider two variables at each step. So, we begin at the root node, here we randomly select two variables as candidates for the root node.

- Let's say we selected Blood Flow and Blocked arteries. Out of these 2 variables, we must now select the variable that best separates the samples. For the sake of this example, let's say that Blocked Arteries is a more significant predictor and thus assign it as the root node.
- Our next step is to repeat the same process for each of the upcoming branch nodes. Here, we again select two variables at random as candidates for the branch node and then choose a variable that best separates the samples.

Just like this, we build the tree by only considering random subsets of variables at each step. By following the above process, our tree would look something like this:

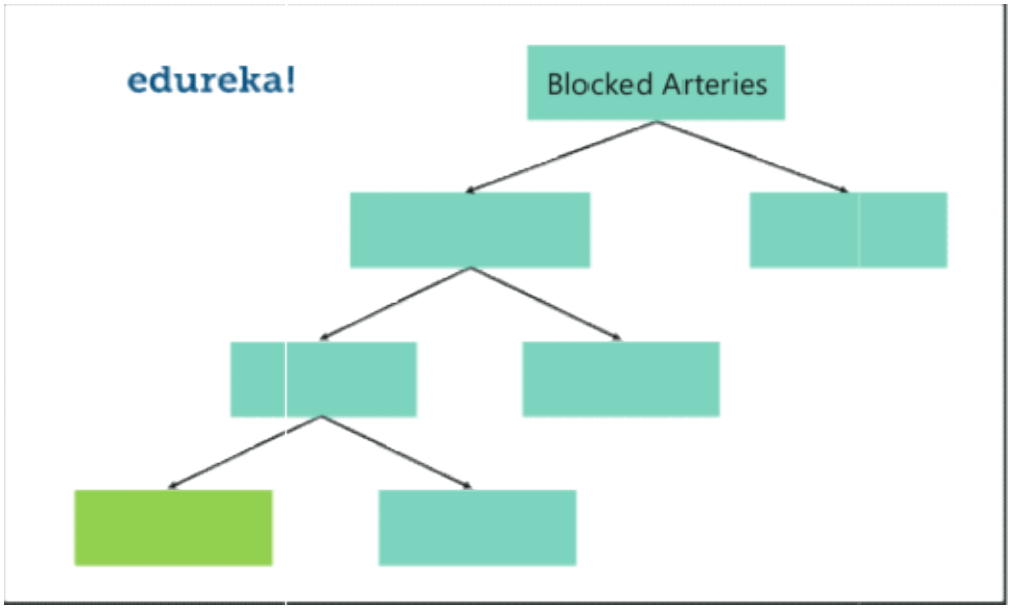


Figure 2.11: Decetion tree

- **Step 3: Go back to Step 1 and Repeat**

Like I mentioned earlier, Random Forest is a collection of Decision Trees. Each Decision Tree predicts the output class based on the respective predictor variables used in that tree. Finally, the outcome of all the Decision Trees in a Random Forest is recorded and the class with the majority votes is computed as the output class.

Thus, we must now create more decision trees by considering a subset of random predictor variables at each step. To do this, go back to step 1, create a new bootstrapped data set and then build a Decision Tree by considering only a subset of variables at each step. So, by following the above steps, our Random Forest would look something like this:

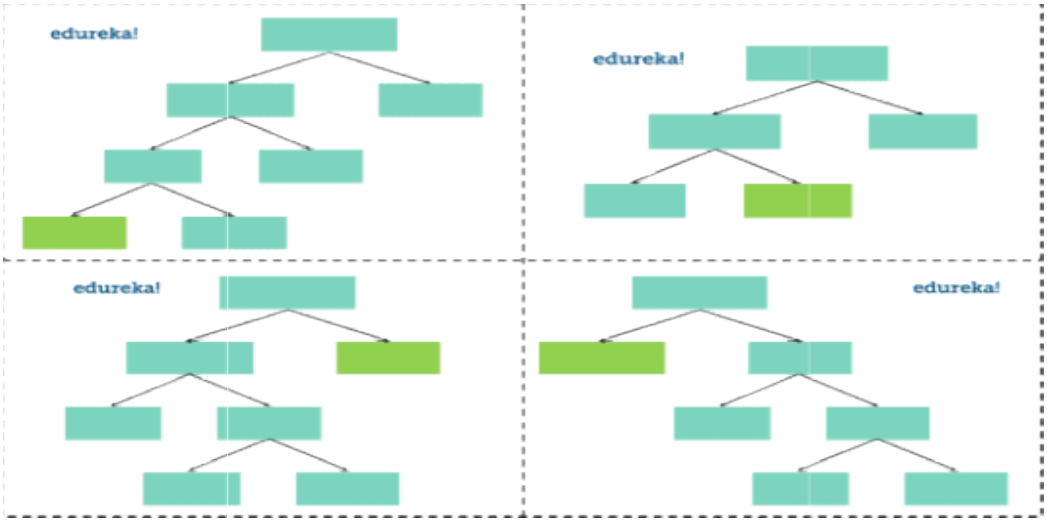


Figure 2.12: Decetion Tree

This iteration is performed 100's of times, therefore creating multiple decision trees with each tree computing the output, by using a subset of randomly selected variables at each step.

Having such a variety of Decision Trees in a Random Forest is what makes it more effective than an individual Decision Tree created using all the features and the whole data set.

- **Step 4: Predicting the outcome of a new data point**

Now that we've created a random forest, let's see how it can be used to predict whether a new patient has heart disease or not. The below diagram has the data about the new patient. All we have to do is run this data down the decision trees that we made.

The first tree shows that the patient has heart disease, so we keep a track of that in a table as shown in the figure.

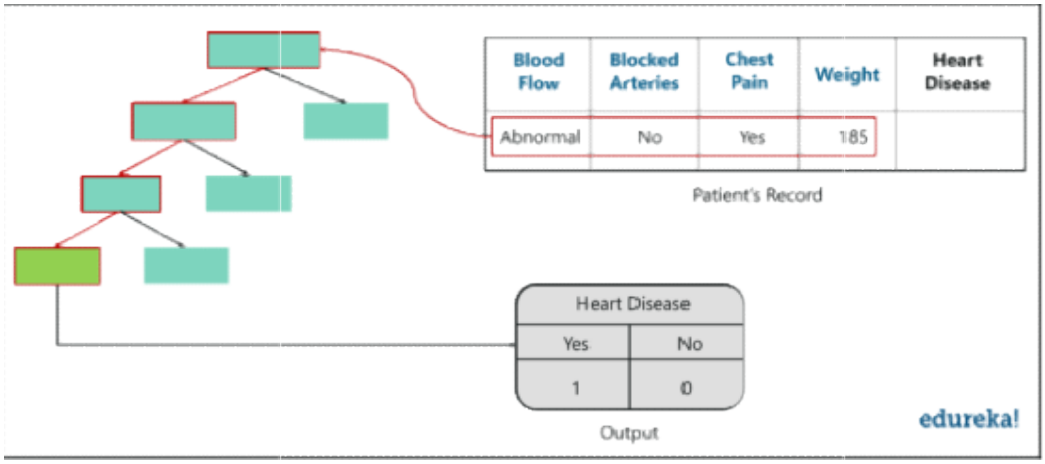


Figure 2.13 :Output – Random Forest

Similarly, we run this data down the other decision trees and keep a track of the class predicted by each tree. After running the data down all the trees in the Random Forest, we check which class got the majority votes. In our case, the class 'Yes' received the most number of votes, hence it's clear that the new patient has heart disease.

To conclude, we bootstrapped the data and used the aggregate from all the trees to make a decision, this process is known as Bagging.

- **Step 5: Evaluate the Model**

Our final step is to evaluate the Random Forest model. Earlier while we created the bootstrapped data set, we left out one entry/sample since we duplicated another sample. In a real-world problem, about 1/3rd of the original data set is not included in the bootstrapped data set.

The below figure shows the entry that didn't end up in the bootstrapped data set.

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	No	Yes	218	No

Figure 2.14:This sample data set that does not include in the bootstrapped data set is known as the Out-Of-Bag (OOB) data set.

2.5 Project Code:

```
1. #! /usr/bin/env python
2. import sys
3. import numpy as np
4. import cv2
5. # Read points from text file
6. def readPoints(path) :
7. # Create an array of points.
8. points = [];
9. # Read points
10.     with open(path) as file :
11.         for line in file :
12.             x, y = line.split()
13.             points.append((int(x), int(y)))
14. return points
15. # Apply affine transform calculated using srcTri and dstTri to src and
16. # output an image of size.
17. def applyAffineTransform(src, srcTri, dstTri, size) :
18. # Given a pair of triangles, find the affine transform.
19.     warpMat = cv2.getAffineTransform( np.float32(srcTri), np.float32(dstTri) )
20. # Apply the Affine Transform just found to the src image
21.     dst = cv2.warpAffine( src, warpMat, (size[0], size[1]), None,
    flags=cv2.INTER_LINEAR, borderMode=cv2.BORDER_REFLECT_101 )
22. return dst
23. # Check if a point is inside a rectangle
24. def rectContains(rect, point) :
25.     if point[0] < rect[0] :
26.         return False
27.     elif point[1] < rect[1] :
28.         return False
29.     elif point[0] > rect[0] + rect[2] :
30.         return False
```

```
31.     elif point[1] > rect[1] + rect[3] :
32.         return False
33.     return True
34. #calculate delanauy triangle
35. def calculateDelaunayTriangles(rect, points):
36.     #create subdiv
37.     subdiv = cv2.Subdiv2D(rect);
38.     # Insert points into subdiv
39.     for p in points:
40.         subdiv.insert(p)
41. triangleList = subdiv.getTriangleList();
42. delaunayTri = []
43. pt = []
44. for t in triangleList:
45.         pt.append((t[0], t[1]))
46.         pt.append((t[2], t[3]))
47.         pt.append((t[4], t[5]))
48. pt1 = (t[0], t[1])
49.         pt2 = (t[2], t[3])
50.         pt3 = (t[4], t[5])
51. if rectContains(rect, pt1) and rectContains(rect, pt2) and rectContains(rect, pt3):
52.             ind = []
53.             #Get face-points (from 68 face detector) by coordinates
54.             for j in range(0, 3):
55.                 for k in range(0, len(points)):
56.                     if(abs(pt[j][0] - points[k][0]) < 1.0 and abs(pt[j][1] - points[k][1]) < 1.0):
57.                         ind.append(k)
58.             # Three points form a triangle. Triangle array corresponds to the file tri.txt in
    FaceMorph
59.             if len(ind) == 3:
60.                 delaunayTri.append((ind[0], ind[1], ind[2]))
61. pt = []
62. return delaunayTri
```

```
63. # Warps and alpha blends triangular regions from img1 and img2 to img
64. def warpTriangle(img1, img2, t1, t2) :
65. # Find bounding rectangle for each triangle
66.     r1 = cv2.boundingRect(np.float32([t1]))
67.     r2 = cv2.boundingRect(np.float32([t2]))
68. # Offset points by left top corner of the respective rectangles
69.     t1Rect = []
70.     t2Rect = []
71.     t2RectInt = []
72. for i in range(0, 3):
73.         t1Rect.append(((t1[i][0] - r1[0]),(t1[i][1] - r1[1])))
74.         t2Rect.append(((t2[i][0] - r2[0]),(t2[i][1] - r2[1])))
75.         t2RectInt.append(((t2[i][0] - r2[0]),(t2[i][1] - r2[1])))
76.
77.     # Get mask by filling triangle
78.     mask = np.zeros((r2[3], r2[2], 3), dtype = np.float32)
79.     cv2.fillConvexPoly(mask, np.int32(t2RectInt), (1.0, 1.0, 1.0), 16, 0);
80. # Apply warpImage to small rectangular patches
81.     img1Rect = img1[r1[1]:r1[1] + r1[3], r1[0]:r1[0] + r1[2]]
82.     #img2Rect = np.zeros((r2[3], r2[2]), dtype = img1Rect.dtype)
83. size = (r2[2], r2[3])
84.
85.     img2Rect = applyAffineTransform(img1Rect, t1Rect, t2Rect, size)
86.
87.     img2Rect = img2Rect * mask
88.
89.     # Copy triangular region of the rectangular patch to the output image
90.     img2[r2[1]:r2[1]+r2[3],        r2[0]:r2[0]+r2[2]]        =        img2[r2[1]:r2[1]+r2[3],
    r2[0]:r2[0]+r2[2]] * ( (1.0, 1.0, 1.0) - mask )
91.
92.     img2[r2[1]:r2[1]+r2[3],        r2[0]:r2[0]+r2[2]]        =        img2[r2[1]:r2[1]+r2[3],
    r2[0]:r2[0]+r2[2]] + img2Rect
93.
```

```
94.
95. if __name__ == '__main__' :
96.
97.     # Make sure OpenCV is version 3.0 or above
98.     (major_ver, minor_ver, subminor_ver) = (cv2.__version__).split('.')
99.
100.            if int(major_ver) < 3 :
101.                print >>sys.stderr, 'ERROR: Script needs OpenCV 3.0 or higher'
102.                sys.exit(1)
103.
104.            # Read images
105.            filename1 = 'donald_trump.jpg'
106.            filename2 = 'ted_cruz.jpg'
107.
108.            img1 = cv2.imread(filename1);
109.            img2 = cv2.imread(filename2);
110.            img1Warped = np.copy(img2);
111.
112.            # Read array of corresponding points
113.            points1 = readPoints(filename1 + '.txt')
114.            points2 = readPoints(filename2 + '.txt')
115.
116.            # Find convex hull
117.            hull1 = []
118.            hull2 = []
119.
120.            hullIndex = cv2.convexHull(np.array(points2), returnPoints = False)
121.
122.            for i in range(0, len(hullIndex)):
123.                hull1.append(points1[int(hullIndex[i])])
124.                hull2.append(points2[int(hullIndex[i])])
125.
126.
```

```
127.        # Find delanauy traingulation for convex hull points
128.        sizeImg2 = img2.shape
129.        rect = (0, 0, sizeImg2[1], sizeImg2[0])
130.
131.        dt = calculateDelaunayTriangles(rect, hull2)
132.
133.        if len(dt) == 0:
134.            quit()
135.
136.        # Apply affine transformation to Delaunay triangles
137.        for i in range(0, len(dt)):
138.            t1 = []
139.            t2 = []
140.
141.            #get points for img1, img2 corresponding to the triangles
142.            for j in range(0, 3):
143.                t1.append(hull1[dt[i][j]])
144.                t2.append(hull2[dt[i][j]])
145.
146.            warpTriangle(img1, img1Warped, t1, t2)
147.
148.
149.        # Calculate Mask
150.        hull8U = []
151.        for i in range(0, len(hull2)):
152.            hull8U.append((hull2[i][0], hull2[i][1]))
153.
154.        mask = np.zeros(img2.shape, dtype = img2.dtype)
155.
156.        cv2.fillConvexPoly(mask, np.int32(hull8U), (255, 255, 255))
157.
158.        r = cv2.boundingRect(np.float32([hull2]))
159.
```

```
160.        center = ((r[0]+int(r[2]/2), r[1]+int(r[3]/2)))
161.
162.
163.        # Clone seamlessly.
164.        output = cv2.seamlessClone(np.uint8(img1Warped), img2, mask, center,
    cv2.NORMAL_CLONE)
165.
166.        cv2.imshow("Face Swapped", output)
167.        cv2.waitKey(0)
168.
169.        cv2.destroyAllWindows()
```

Chapter-3

3.1 Tools & packages using in project

3.1.1 OpenCV

- **What Is Computer Vision?**

We all use Facebook, correct? Let us say you and your friends went on a vacation and you clicked a lot of pictures and you want to upload them on Facebook and you did. But now, wouldn't it take so much time just to find your friends faces and tag them in each and every picture? Well, Facebook is intelligent enough to actually tag people for you.

So, how do you think the auto tag feature works? In simple terms, it works on computer vision.

Computer Vision is an interdisciplinary field that deals with how computers can be made to gain a high-level understanding from digital images or videos.

The idea here is to automate tasks that the human visual systems can do. So, a computer should be able to recognize objects such as that of a face of a human being or a lamppost or even a statue.

- **How Does A Computer Read An Image?**

Consider the below image:

Fig:3.1

We can figure out that it is an image of the New York Skyline. But, can a computer find this out all on its own? The answer is no! The computer reads any image as a range of values between 0 and 255. For any color image, there are 3 primary channels – Red, green and blue. How it works is pretty simple A matrix is formed for every primary color and later these matrices combine to provide a Pixel value for the individual R, G, B colors. Each element of the matrices provide data pertaining to the intensity of brightness of the pixel.

Consider the following image:

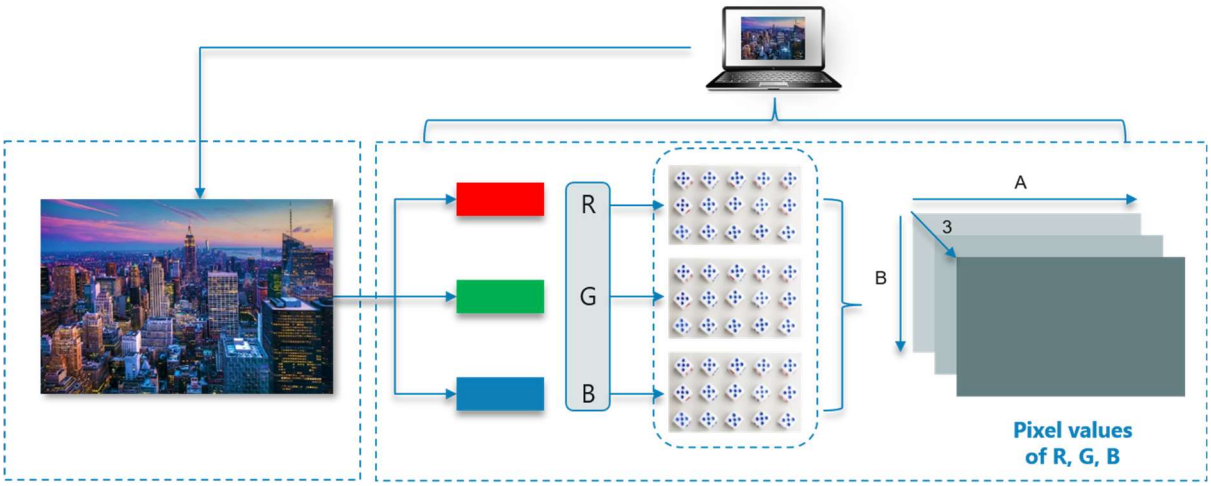


Fig.: 3.2

As shown, the size of the image here can be calculated as B x A x 3.

Note: For a black-white image, there is only one single channel.

Next up on this OpenCV Python Tutorial blog, let us look at what OpenCV actually is.

- **What Is OpenCV?**

OpenCV is a Python library which is designed to solve computer vision problems. OpenCV was originally developed in 1999 by Intel but later it was supported by Willow Garage.

```
Import cv2
1
2
3
4# colored Image
5
6Img = cv2.imread (&ldquo;Penguins.jpg&rdquo;,1)
7
8
9
10# Black and White (gray scale)
11
12Img_1 = cv2.imread (&ldquo;Penguins.jpg&rdquo;,0)
13
```

OpenCV supports a wide variety of programming languages such as C++, Python, Java etc. Support for multiple platforms including Windows, Linux, and MacOS.

OpenCV Python is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays.

This makes it easier to integrate it with other libraries which use NumPy. For example, libraries such as SciPy and Matplotlib.

Next up on this OpenCV Python Tutorial blog, let us look at some of the basic operations that we can perform with OpenCV.

- ## Basic Operations With OpenCV?

Let us look at various concepts ranging from loading images to resizing them and so on.

- ## Loading an image using OpenCV:

As seen in the above piece of code, the first requirement is to import the OpenCV module.

Later we can read the image using **imread** module. The 1 in the parameters denotes that it is a color image. If the parameter was 0 instead of 1, it would mean that the image being imported is a black and white image. The name of the image here is 'Penguins'. Pretty straightforward, right?

- ## Image Shape/Resolution:

We can make use of the shape sub-function to print out the shape of the image. Check out the below image:

```
Import cv2

# Black and White (gray scale)

Img = cv2.imread (&ldquo;Penguins.jpg&rdquo;,0)

Print(img.shape)
```

By shape of the image, we mean the shape of the NumPy array. As you see from executing the code, the matrix consists of 768 rows and 1024 columns.

- ## Displaying the image:

Displaying an image using OpenCV is pretty simple and straightforward. Consider the below image:

```
import cv2

# Black and White (gray scale)

Img = cv2.imread (&ldquo;Penguins.jpg&rdquo;,0)

cv2.imshow(&ldquo;Penguins&rdquo;, img)

cv2.waitKey(0)

# cv2.waitKey(2000)

cv2.destroyAllWindows()
```

As you can see, we first import the image using **imread.** We require a window output to display the images, right?

We use the **imshow** function to display the image by opening a window. There are 2 parameters to the **imshow** function which is the name of the window and the image object to be displayed.

Later, we wait for a user event. **waitKey** makes the window static until the user presses a key. The parameter passed to it is the time in milliseconds.

And lastly, we use **destroyAllWindows** to close the window based on the waitForKey parameter.

- ## Resizing the image:

Similarly, resizing an image is very easy. Here's another code snippet:

```
import cv2

# Black and White (gray scale)

img = cv2.imread (&ldquo;Penguins.jpg&rdquo;,0)

resized_image = cv2.resize(img, (650,500))

cv2.imshow(&ldquo;Penguins&rdquo;, resized_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Here, **resize** function is used to resize an image to the desired shape. The parameter here is the shape of the new resized image.

Later, do note that the image object changes from **img** to **resized_image,** because of the image object changes now.

Rest of the code is pretty simple to the previous one, correct?

I am sure you guys are curious to look at the penguins, right? This is the image we were looking to output all this while!

Fig:3.3

There is another way to pass the parameters to the resize function. Check out the following representation:

```
Resized_image = cv2.resize(img, int(img.shape[1]/2), int(img.shape[0]/2)))
```

Here, we get the new image shape to be half of that of the original image.

Next up on this OpenCV Python Tutorial blog, let us look at how we perform face detection using OpenCV.

- # Face Detection Using OpenCV

This seems complex at first but it is very easy. Let me walk you through the entire process and you will feel the same.

- **Step 1:** Considering our prerequisites, we will require an image, to begin with. Later we need to create a cascade classifier which will eventually give us the features of the face.
- **Step 2:** This step involves making use of OpenCV which will read the image and the features file. So at this point, there are NumPy arrays at the primary data points.

All we need to do is to search for the row and column values of the face NumPy ndarray. This is the array with the face rectangle coordinates.

- **Step 3:** This final step involves displaying the image with the rectangular face box.

Check out the following image, here I have summarized the 3 steps in the form of an image for easier readability:

Pretty straightforward, correct?

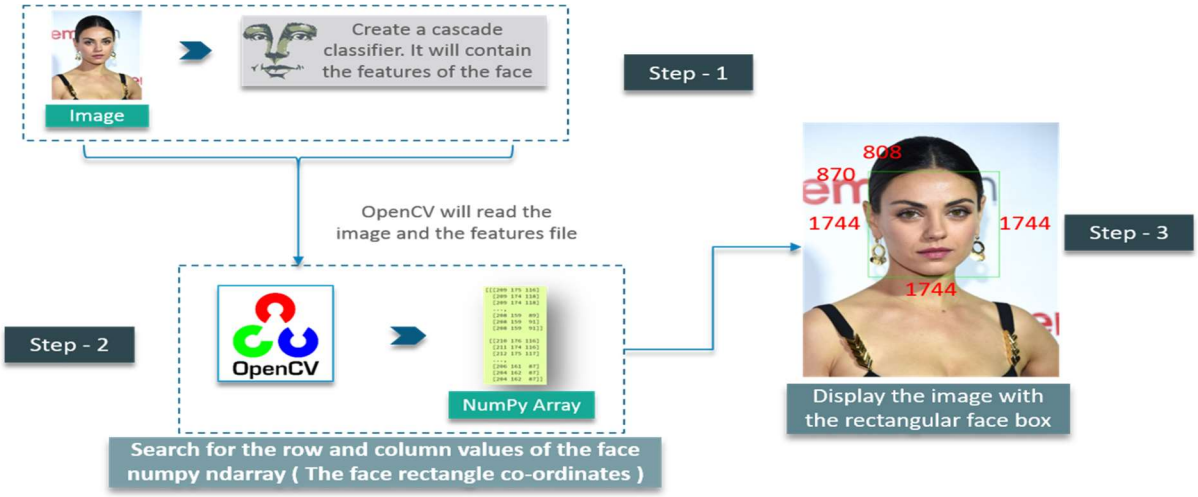


Fig:3.4

First, we create a **CascadeClassifier** object to extract the features of the face as explained earlier. The path to the XML file which contains the face features is the parameter here.

The next step would be to read an image with a face on it and convert it into a black and white image using **COLOR_BGR2GREY**. Followed by this, we search for the coordinates for the image. This is done using **detectMultiScale**.

What coordinates, you ask? It's the coordinates for the face rectangle. The **scaleFactor** is used to decrease the shape value by 5% until the face is found. So, on the whole – Smaller the value, greater is the accuracy.

Finally, the face is printed on the window.

- **Adding the rectangular face box:**

This logic is very simple – As simple as making use of a for loop statement. Check out the following image

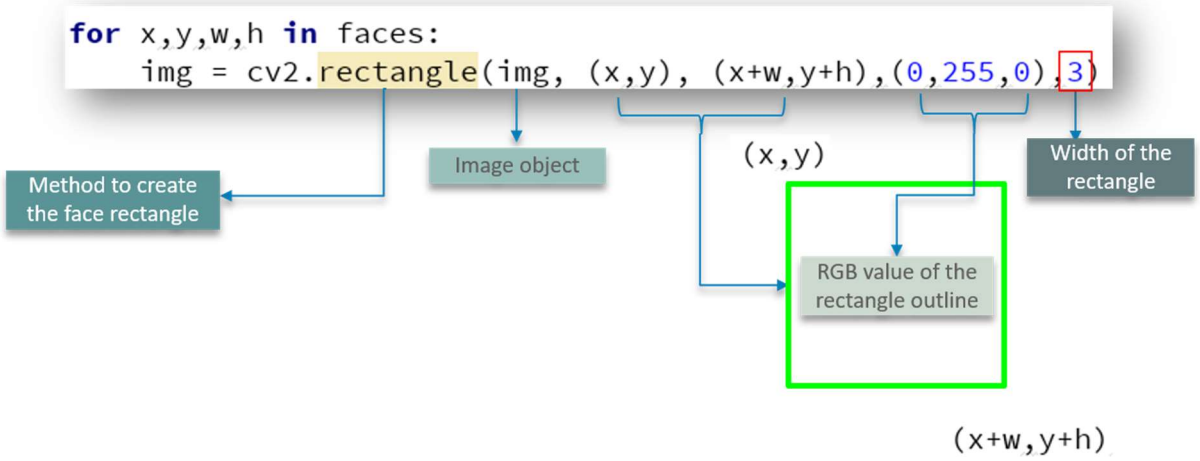


Fig.:3.5

We define the method to create a rectangle using **cv2.rectangle** by passing parameters such as the image object, RGB values of the box outline and the width of the rectangle.Let us check out the entire code for face detection:

```python
import cv2

# Create a CascadeClassifier Object
face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

# Reading the image as it is
img = cv2.imread("photo.jpg")

# Reading the image as gray scale image
gray_img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

# Search the co-ordintes of the image
faces = face_cascade.detectMultiScale(gray_img, scaleFactor = 1.05,
        
             
for x,y,w,h in faces:
    img = cv2.rectangle(img, (x,y), (x+w,y+h),(0,255,0),3)

resized = cv2.resize(img, (int(img.shape[1]/7),int(img.shape[0]/7)))

cv2.imshow("Gray", resized)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Next up on this OpenCV Python Tutorial blog, let us look at how to use OpenCV to capture video with the computer webcam.

• Capturing Video Using OpenCV

Capturing videos using OpenCV is pretty simple as well. the following loop will give you a better idea. Check it out:

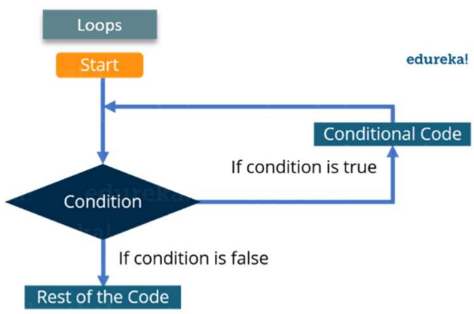
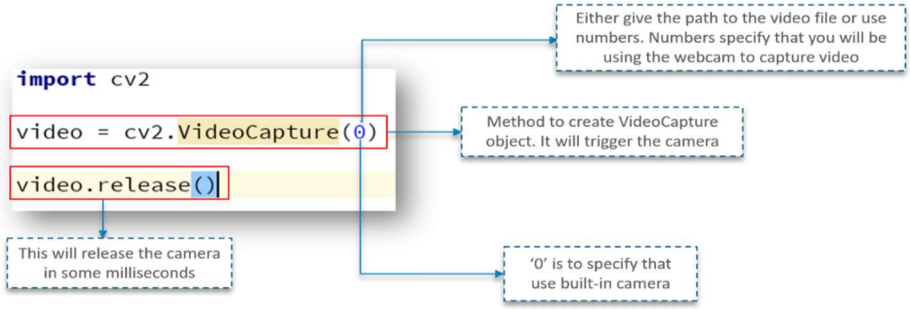


Fig.:3.6

The images are read one-by-one and hence videos are produced due to fast processing of frames which makes the individual images move.

- # Capturing Video:

Check out the following image:



Fig,:3.7

First, we import the OpenCV library as usual. Next, we have a method called **VideoCapture** which is used to create the VideoCapture object. This method is used to trigger the camera on the user's machine. The parameter to this function denotes if the program should make use of the built-in camera or an add-on camera. '0' denotes the built-in camera in this case.

And lastly, the **release** method is used to release the camera in a few milliseconds.When you go ahead and type in and try to execute the above code, you will notice that the camera light switches on for a split second and turns off later. Why does this happen? This happens because there is no time delay to keep the camera functional.

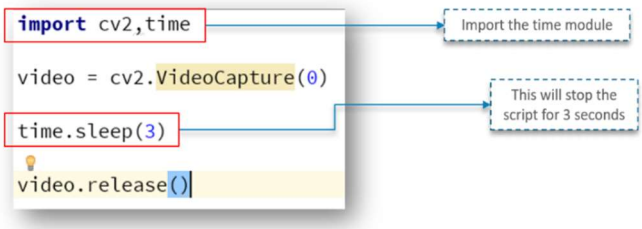


Fig.:3.8

Looking at the above code, we have a new line called **time.sleep(3)** – This makes the script to stop for 3 seconds. Do note that the parameter passed is the time in seconds. So, when the code is executed, the webcam will be turned on for 3 seconds.

- **Adding the window:**

Adding a window to show the video output is pretty simple and can be compared to the same methods used for images. However, there is a slight change. Check out the following code:

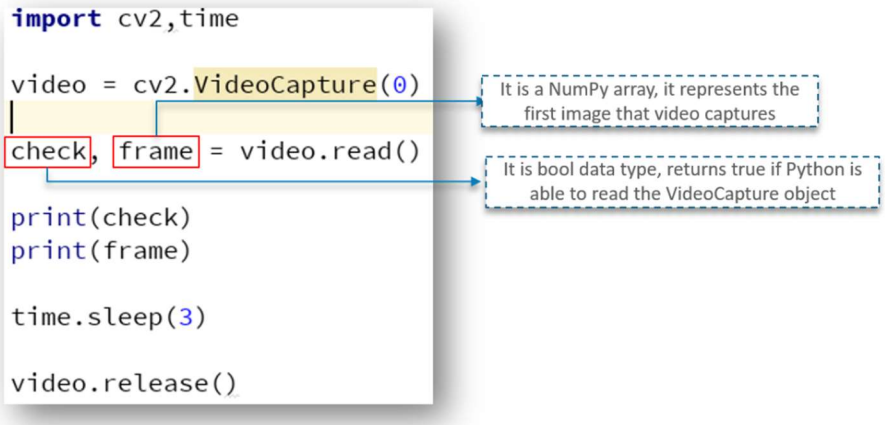


Fig.:3.9

Here, we have defined a NumPy array which we use to represent the first image that the video captures – This is stored in the **frame** array.

We also have **check** – This is a boolean datatype which returns **True** if Python is able to access and read the **VideoCapture** object.

Check out the output below:

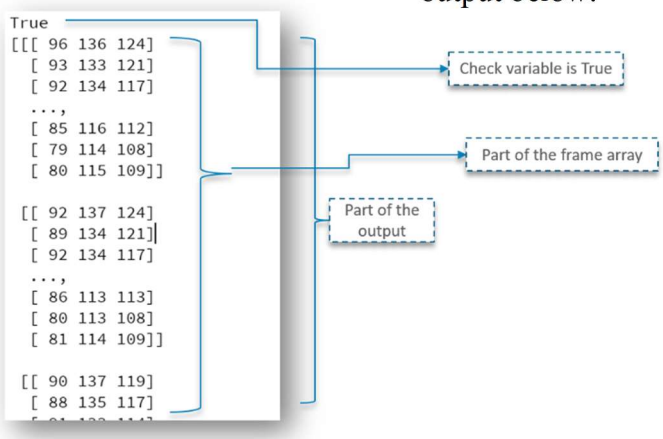


Fig.:3.10

As you can check out, we got the output as **True** and the part of the frame array is printed.

But we need to read the first frame/image of the video to begin, correct?

To do exactly that, we need to first create a frame object which will read the images of the **VideoCapture** object.

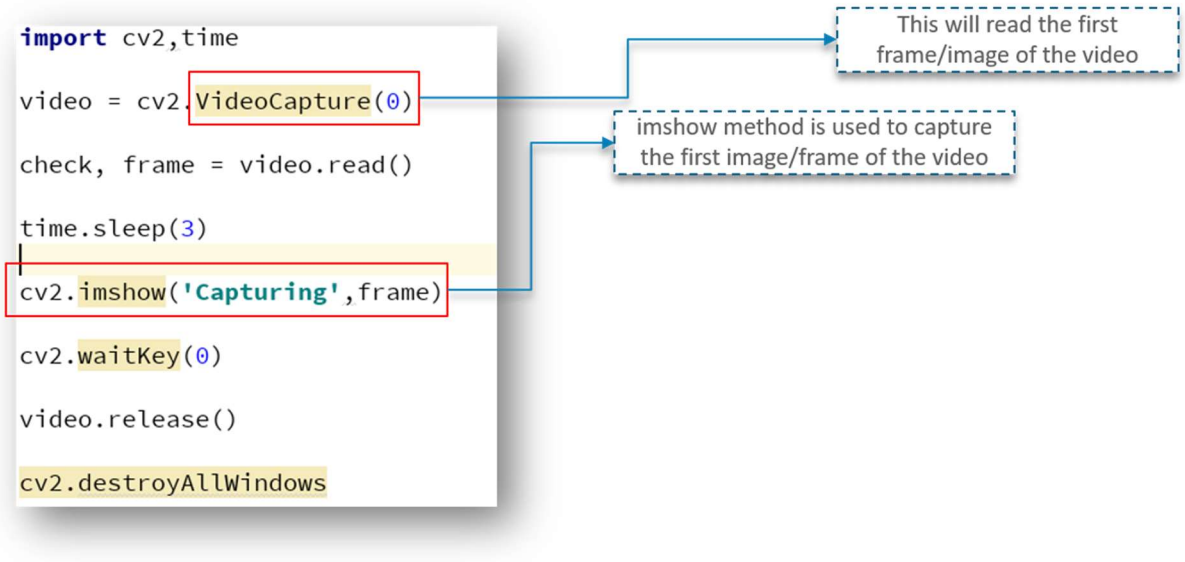


Fig.:3.11

As seen above, the **imshow** method is used to capture the first frame of the video.

All this while, we have tried to capture the first image/frame of the video but directly capturing the video.

- **Capturing Video Directly:**

In order to capture the video, we will be using the **while** loop. While condition will be such that, until unless **'check'** is **True.** If it is, then Python will display the frames.

Here's the code snippet image:

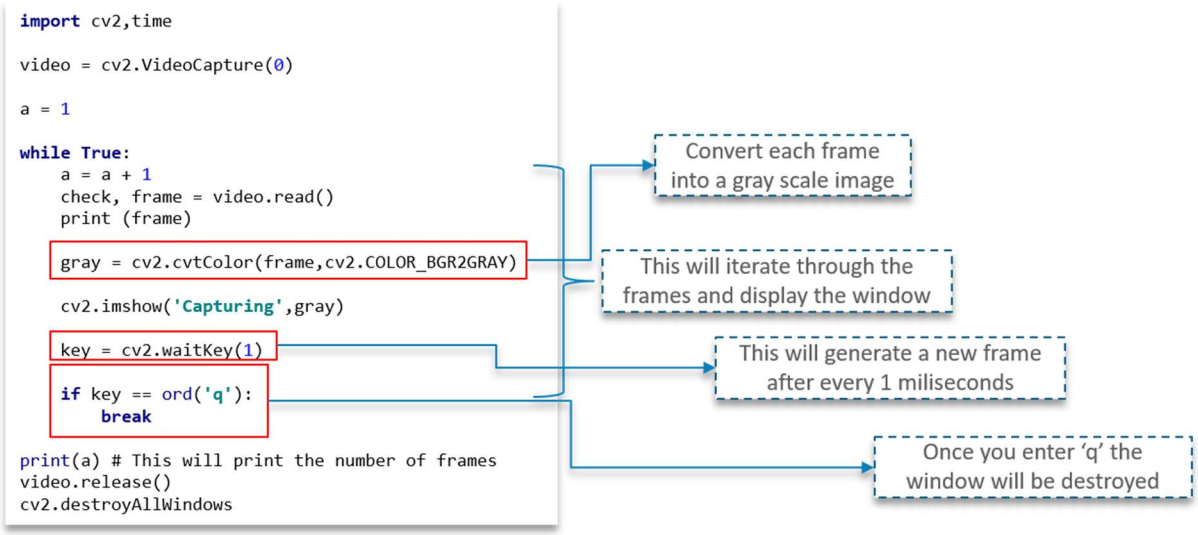


Fig.:3.12

We make use of the **cvtColor** function to convert each frame into a grey-scale image as explained earlier.

waitKey(1) will make sure to generate a new frame after every millisecond of a gap.

It is important here that you note that the **while** loop is completely in play to help iterate through the frames and eventually display the video.

There is a user event trigger here as well. Once the 'q' key is pressed by the user, the program window closes.

OpenCV is pretty easy to grasp, right? I personally love how good the readability is and how quickly a beginner can get started working with OpenCV.

Next up on this OpenCV Python Tutorial blog, let us look at how to use a very interesting motion detector use case using OpenCV.

- **Use Case: Motion Detector Using OpenCV**
- **Problem Statement:**

You have been approached by a company that is studying human behavior. Your task is to give them a webcam, that can detect the motion or any movement in front of it. This should return a graph, this graph should contain how long the human/object was in front of the camera.

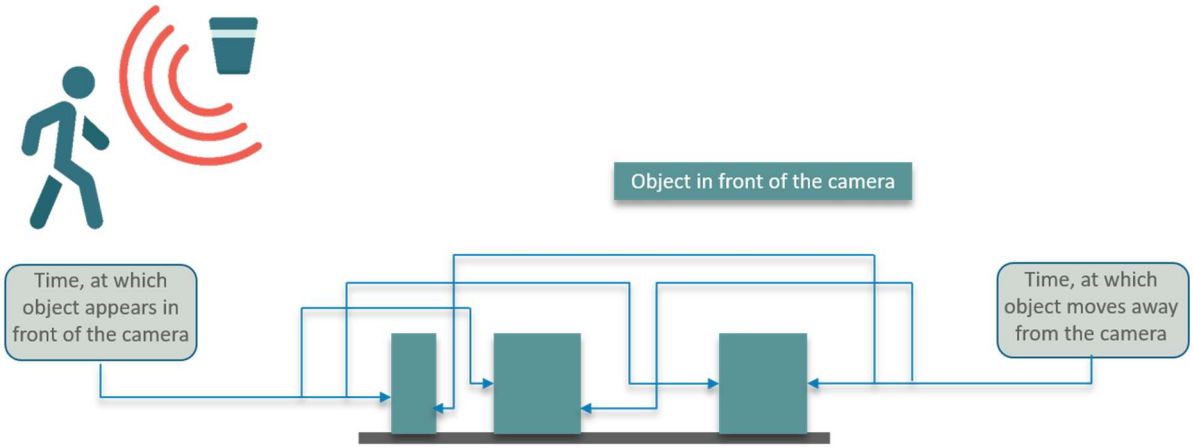


Fig.:3.13

So, now that we have defined our problem statement, we need to build a solution logic to approach the problem in a structured way.

Consider the below diagram:

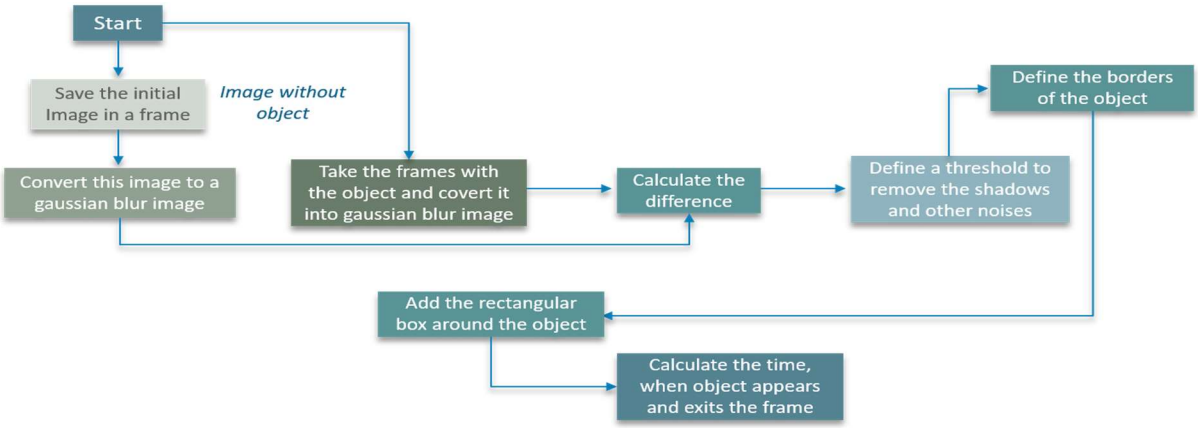


Fig.:3.14

Initially, we save the image in a particular frame.

The next step involves converting the image to a Gaussian blur image. This is done so as to ensure we calculate a palpable difference between the blurred image and the actual image.

At this point, the image is still not an object. We define a threshold to remove blemishes such as shadows and other noises in the image.

Borders for the object are defined later and we add a rectangular box around the object as we discussed earlier on the blog.

Lastly, we calculate the time at which the object appears and exits the frame.

Pretty easy, right?

Here's the code snippet:

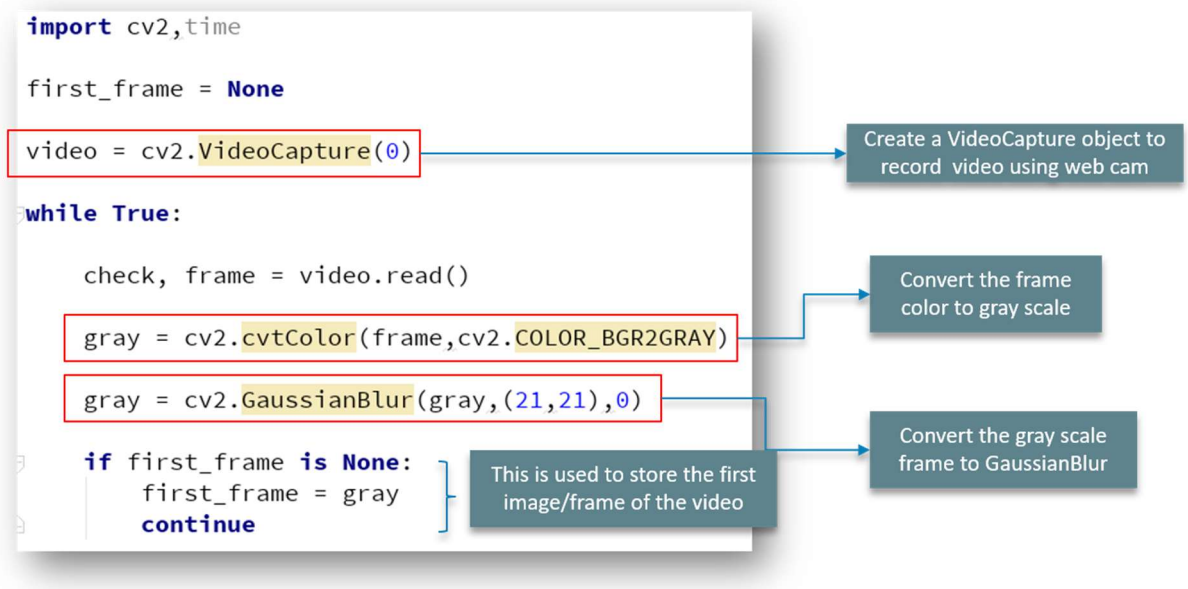


Fig.:3.15

The same principle follows through here as well. We first import the package and create the **VideoCapture** object to ensure we capture video using the webcam.

The while loop iterates through the individual frames of the video. We convert the color frame to a grey-scale image and later we convert this grey-scale image to Gaussian blur.

We need to store the first image/frame of the video, correct? We make use of the **if** statement for this purpose alone.

Now, let us dive into a little more code:

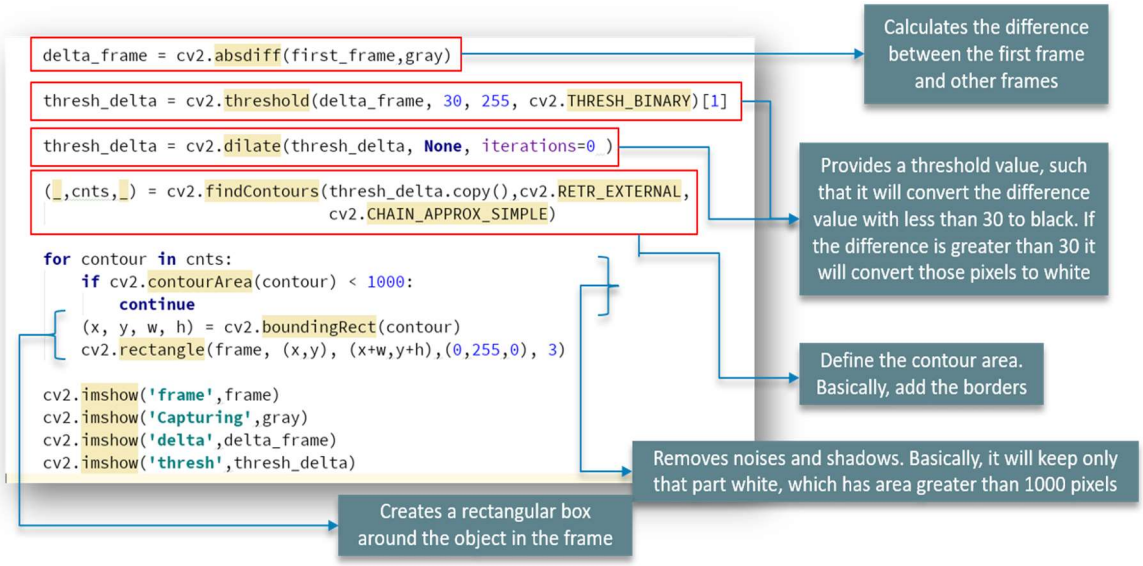


Fig.:3.16

We make use of the **absdiff** function to calculate the difference between the first occurring frame and all the other frames.

The **threshold** function provides a threshold value, such that it will convert the difference value with less than 30 to black. If the difference is greater than 30 it will convert those pixels to white color. **THRESH_BINARY** is used for this purpose.

Later, we make use of the **findContours** function to define the contour area for our image. And we add in the borders at this stage as well.

The **contourArea** function, as previously explained, removes the noises and the shadows. To make it simple, it will keep only that part white, which has an area greater than 1000 pixels as we've defined for that.

Later, we create a rectangular box around our object in the working frame.

And followed by this is this simple code:

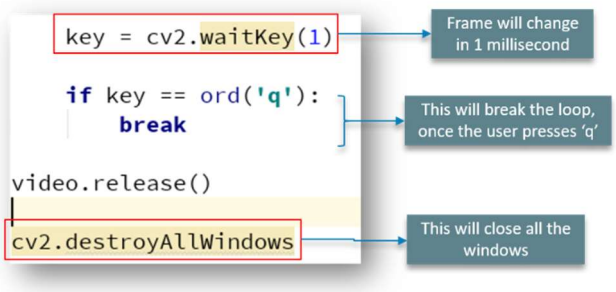


Fig.3.17

As discussed earlier, the frame changes every 1 millisecond and when the user enters **'q',** the loop breaks and the window closes.

We've covered all of the major details on this OpenCV Python Tutorial blog. One thing that remains with our use-case is that we need to calculate the time for which the object was in front of the camera.

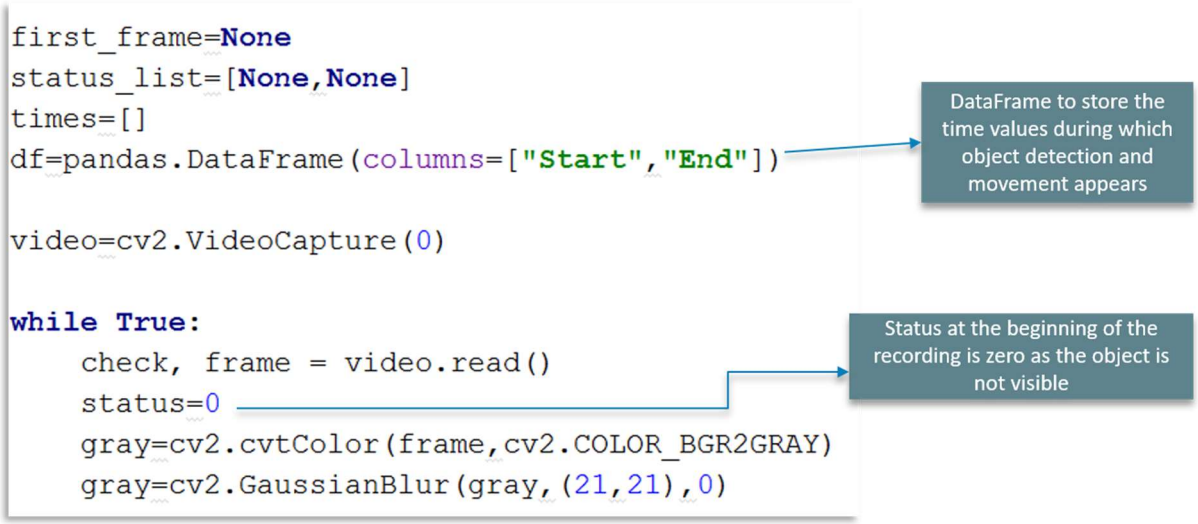


Fig.3.18

We make use of **DataFrame** to store the time values during which object detection and movement appear in the frame.

Followed by that is **VideoCapture** function as explained earlier. But here, we have a flag bit we call **status.** We use this status at the beginning of the recording to be **zero** as the object is not **visible** initially.

```
(_,cnts,_)=cv2.findContours(thresh_frame.copy(),cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

for contour in cnts:
    if cv2.contourArea(contour) < 10000:
        continue
    status=1
```

Change in status when the object is being detected

Fig.3.19

We will change the status flag to 1 when the object is being detected as shown in the above figure. Pretty simple, right?

```
    (x, y, w, h)=cv2.boundingRect(contour)
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 3)
status_list.append(status)

status_list=status_list[-2:]


if status_list[-1]==1 and status_list[-2]==0:
    times.append(datetime.now())
if status_list[-1]==0 and status_list[-2]==1:
    times.append(datetime.now())
```

List of status for every frame

Record datetime in a list when change occurs

Fig.3.20

We are going to make a list of the status for every scanned frame and later record the date and time using **datetime** in a list if and where a change occurs.

```
print(status_list)
print(times)

for i in range(0,len(times),2):
    df=df.append({"Start":times[i],"End":times[i+1]},ignore_index=True)

df.to_csv("Times.csv")

video.release()
cv2.destroyAllWindows
```

Store time values in a DataFrame

Write the DataFrame to a CSV file

Fig.3.21

And we store the time values in a **DataFrame** as shown in the above explanatory diagram. We'll conclude by writing the **DataFrame** to a CSV file as shown.

Plotting the Motion Detection Graph:

The final step in our use-case to display the results. We are displaying the graph which denotes the motion on 2-axes. Consider the below code:

```
from motion_detector import df
from bokeh.plotting import figure, show, output_file
from bokeh.models import HoverTool, ColumnDataSource

df["Start_string"]=df["Start"].dt.strftime("%Y-%m-%d %H:%M:%S")
df["End_string"]=df["End"].dt.strftime("%Y-%m-%d %H:%M:%S")


cds=ColumnDataSource(df)

p=figure(x_axis_type='datetime',height=100, width=500, responsive=True,title="Motion Graph")
p.yaxis.minor_tick_line_color=None
p.ygrid[0].ticker.desired_num_ticks=1

hover=HoverTool(tooltips=[("Start","@Start_string"),("End","@End_string")])
p.add_tools(hover)

q=p.quad(left="Start",right="End",bottom=0,top=1,color="red",source=cds)

output_file("Graph1.html")
show(p)
```

Import the DataFrame from the motion_detector.py

Convert time to a string format

The DataFrame of time values is plotted on the browser using Bokeh plots

Fig.3.22

To begin with, we import the **DataFrame** from the **motion_detector.py** file.

The next step involves converting time to a readable string format which can be parsed.

Lastly, the **DataFrame** of time values is plotted on the browser using **Bokeh plots.**

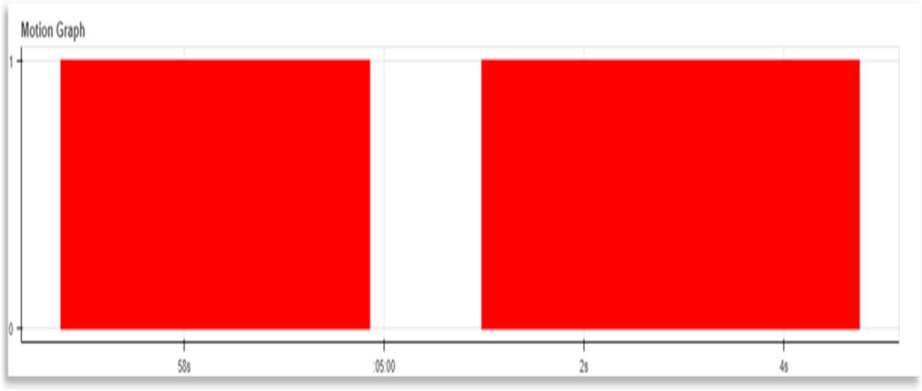
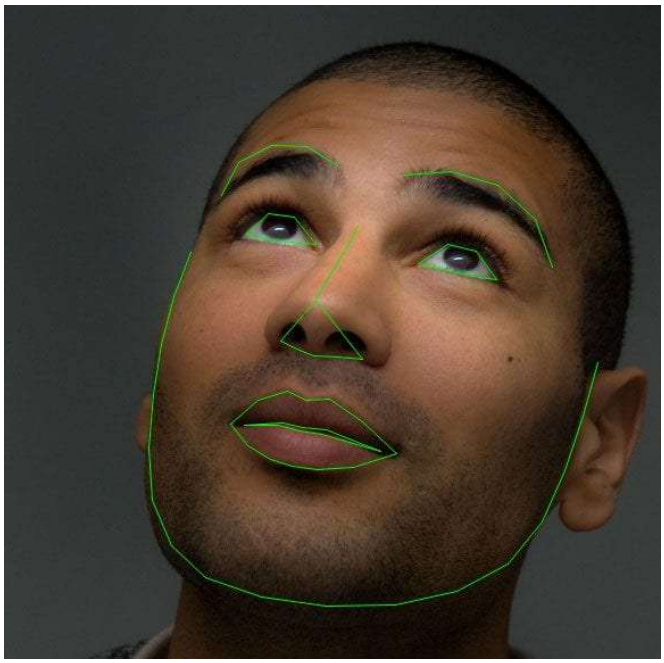


Fig.3.23

This will be very handy when you are trying to develop applications that require image recognition and similar principles. Now, you should also be able to use these concepts to develop applications easily with the help of OpenCV in Python.

3.1.2 Dlib 68 point & Facial recognition:

- **what are facial landmarks?**



Detecting facial landmarks is a subset of the shape prediction problem. Given an input image (and normally an ROI that specifies the object of interest), a shape predictor attempts to localize key points of interest along the shape.

In the context of facial landmarks, our goal is detect important facial structures on the face using shape prediction methods.

Detecting facial landmarks is therefore a two step process:

Step #1: Localize the face in the image.
Step #2: Detect the key facial structures on the face ROI.
Face detection (Step #1) can be achieved in a number of ways.

We could use OpenCV's built-in Haar cascades.

We might apply a pre-trained HOG + Linear SVM object detector specifically for the task of face detection.

Or we might even use deep learning-based algorithms for face localization.

In either case, the actual algorithm used to detect the face in the image doesn't matter. Instead, what's important is that through some method we obtain the face bounding box (i.e., the (x, y)-coordinates of the face in the image).

Given the face region we can then apply Step #2: detecting key facial structures in the face region.

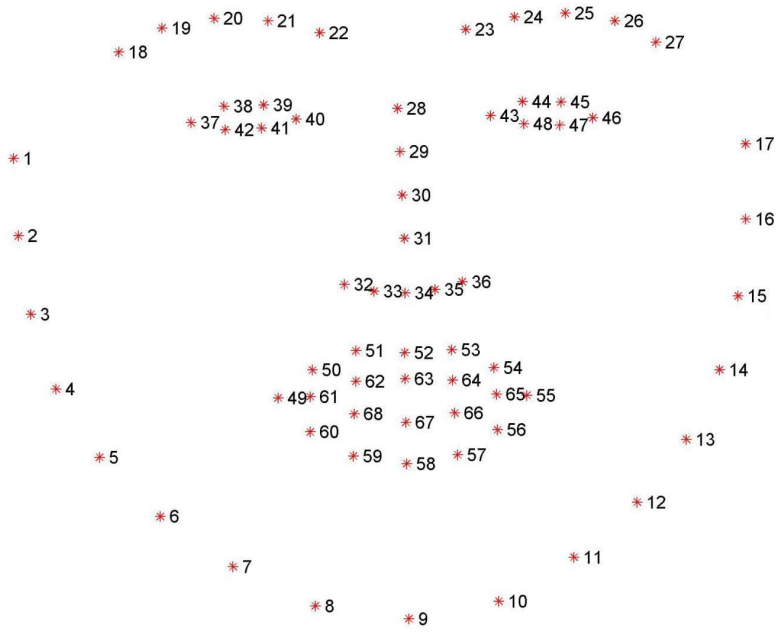
There are a variety of facial landmark detectors, but all methods essentially try to localize and label the following facial regions:

Mouth
Right eyebrow
Left eyebrow
Right eye
Left eye
Nose
Jaw
The facial landmark detector included in the dlib library is an implementation of the One Millisecond Face Alignment with an Ensemble of Regression Trees paper by Kazemi and Sullivan (2014).

This method starts by using:

- A training set of labeled facial landmarks on an image. These images are manually labeled, specifying specific (x, y)-coordinates of regions surrounding each facial structure.
- Priors, of more specifically, the probability on distance between pairs of input pixels. Given this training data, an ensemble of regression trees are trained to estimate the facial landmark positions directly from the pixel intensities themselves (i.e., no "feature extraction" is taking place).

The end result is a facial landmark detector that can be used to detect facial landmarks in real-time with high quality predictions.



These annotations are part of the 68 point iBUG 300-W dataset which the dlib facial landmark predictor was trained on.

It's important to note that other flavors of facial landmark detectors exist, including the 194 point model that can be trained on the HELEN dataset.

Regardless of which dataset is used, the same dlib framework can be leveraged to train a shape predictor on the input training data — this is useful if you would like to train facial landmark detectors or custom shape predictors of your own.

In the remaining of this blog post I'll demonstrate how to detect these facial landmarks in images.

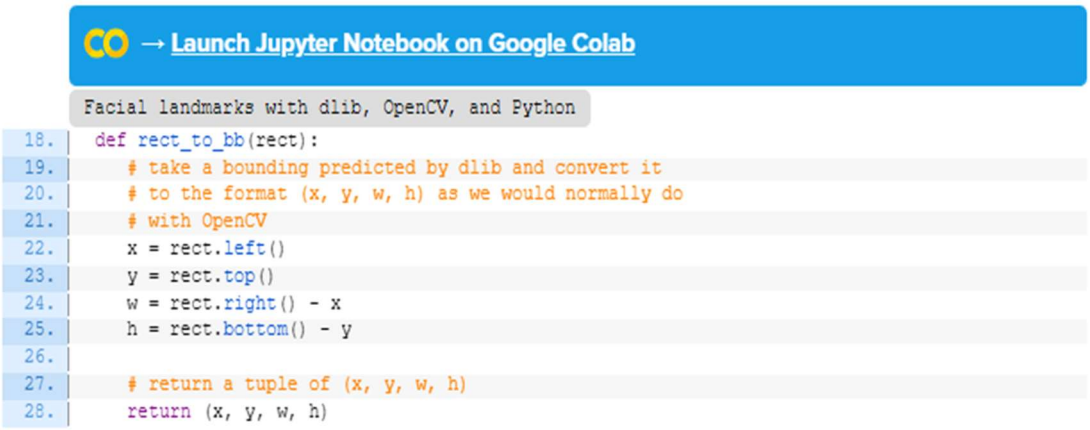
Future blog posts in this series will use these facial landmarks to extract specific regions of the face, apply face alignment, and even build a blink detection system.

- **Detecting facial landmarks with dlib, OpenCV, and Python**

In order to prepare for this series of blog posts on facial landmarks, I've added a few convenience functions to my imutils library, specifically inside **face_utils.py.**

We'll be reviewing two of these functions inside face_utils.py now and the remaining ones next week.

The first utility function is rect_to_bb , short for "rectangle to bounding box":



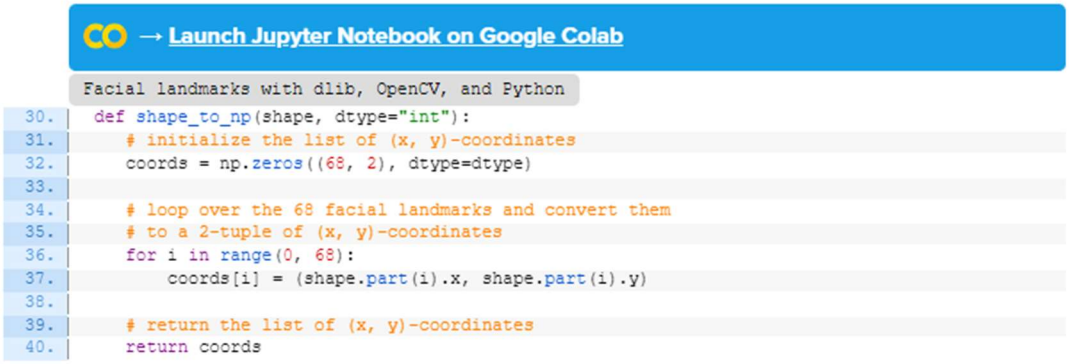
This function accepts a single argument, rect, which is assumed to be a bounding box rectangle produced by a dlib detector (i.e., the face detector).

The rect object includes the (x, y)-coordinates of the detection.

However, in OpenCV, we normally think of a bounding box in terms of "(x, y, width, height)" so as a matter of convenience, the rect_to_bb function takes this rect object and transforms it into a 4-tuple of coordinates.

Again, this is simply a matter of conveinence and taste.

Secondly, we have the shape_to_np function:



The dlib face landmark detector will return a shape object containing the 68 (x, y)-coordinates of the facial landmark regions.

Using the shape_to_np function, we cam convert this object to a NumPy array, allowing it to "play nicer" with our Python code.

Given these two helper functions, we are now ready to detect facial landmarks in images.

Open up a new file, name it facial_landmarks.py , and insert the following code:

→ Launch Jupyter Notebook on Google Colab

Facial landmarks with dlib, OpenCV, and Python

```
1.  # import the necessary packages
2.  from imutils import face_utils
3.  import numpy as np
4.  import argparse
5.  import imutils
6.  import dlib
7.  import cv2
8.
9.  # construct the argument parser and parse the arguments
10. ap = argparse.ArgumentParser()
11. ap.add_argument("-p", "--shape-predictor", required=True,
12.    help="path to facial landmark predictor")
13. ap.add_argument("-i", "--image", required=True,
14.    help="path to input image")
15. args = vars(ap.parse_args())
```

Lines 2-7 import our required Python packages.

We'll be using the face_utils submodule of imutils to access our helper functions detailed above.

We'll then import dlib . If you don't already have dlib installed on your system, please follow the instructions in my previous blog post to get your system properly configured.

Lines 10-15 parse our command line arguments:

--shape-predictor : This is the path to dlib's pre-trained facial landmark detector. You can download the detector model here or you can use the "Downloads" section of this post to grab the code + example images + pre-trained detector as well.

--image : The path to the input image that we want to detect facial landmarks on.

Now that our imports and command line arguments are taken care of, let's initialize dlib's face detector and facial landmark predictor:

→ Launch Jupyter Notebook on Google Colab

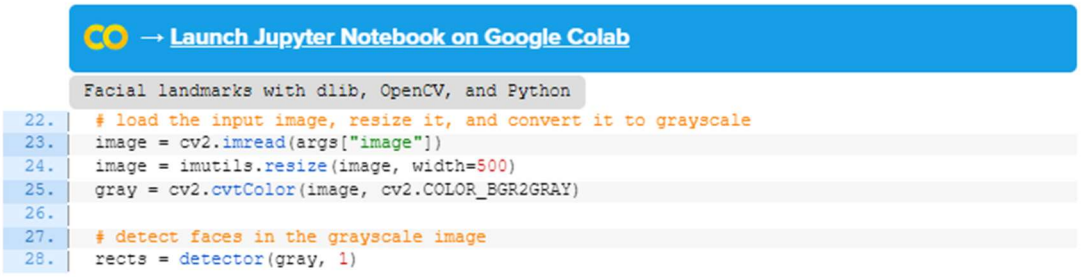
Facial landmarks with dlib, OpenCV, and Python

```
17. # initialize dlib's face detector (HOG-based) and then create
18. # the facial landmark predictor
19. detector = dlib.get_frontal_face_detector()
20. predictor = dlib.shape_predictor(args["shape_predictor"])
```

Line 19 initializes dlib's pre-trained face detector based on a modification to the standard Histogram of Oriented Gradients + Linear SVM method for object detection.

Line 20 then loads the facial landmark predictor using the path to the supplied --shape-predictor.

But before we can actually detect facial landmarks, we first need to detect the face in our input image:



Line 23 loads our input image from disk via OpenCV, then pre-processes the image by resizing to have a width of 500 pixels and converting it to grayscale (Lines 24 and 25).

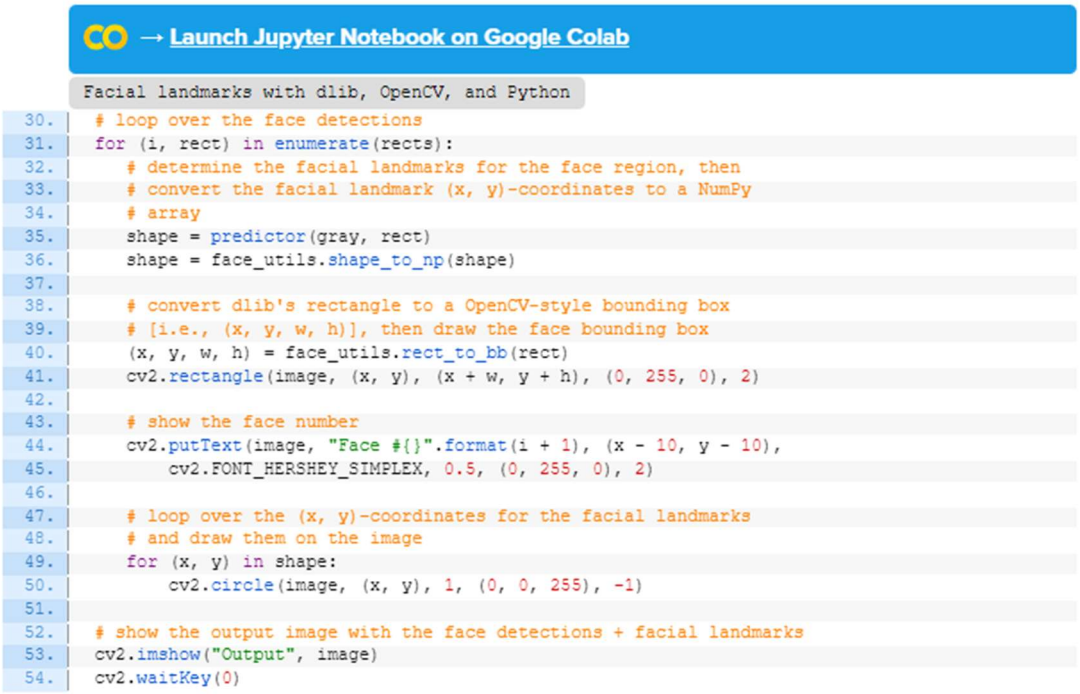
Line 28 handles detecting the bounding box of faces in our image.

The first parameter to the detector is our grayscale image (although this method can work with color images as well).

The second parameter is the number of image pyramid layers to apply when upscaling the image prior to applying the detector (this it the equivalent of computing cv2.pyrUp N number of times on the image).

The benefit of increasing the resolution of the input image prior to face detection is that it may allow us to detect more faces in the image — the downside is that the larger the input image, the more computaitonally expensive the detection process is.

Given the (x, y)-coordinates of the faces in the image, we can now apply facial landmark detection to each of the face regions:



We start looping over each of the face detections on Line 31.

For each of the face detections, we apply facial landmark detection on Line 35, giving us the 68 (x, y)-coordinates that map to the specific facial features in the image.

Line 36 then converts the dlib shape object to a NumPy array with shape (68, 2).

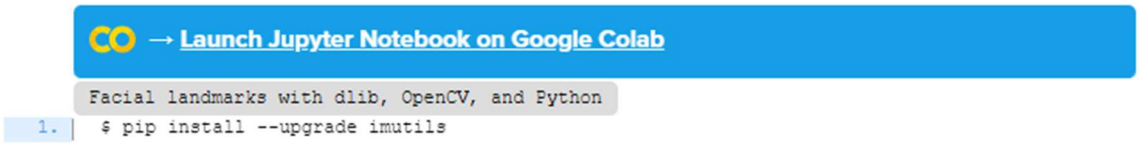
Lines 40 and 41 draw the bounding box surrounding the detected face on the image while Lines 44 and 45 draw the index of the face.

Finally, Lines 49 and 50 loop over the detected facial landmarks and draw each of them individually.

Lines 53 and 54 simply display the output image to our screen.

Facial landmark visualizations

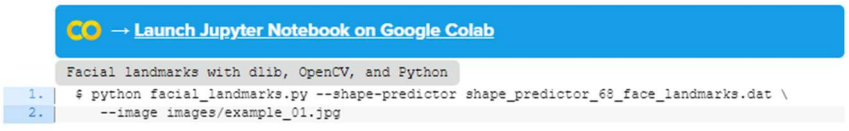
Before we test our facial landmark detector, make sure you have upgraded to the latest version of imutils which includes the face_utils.py file:

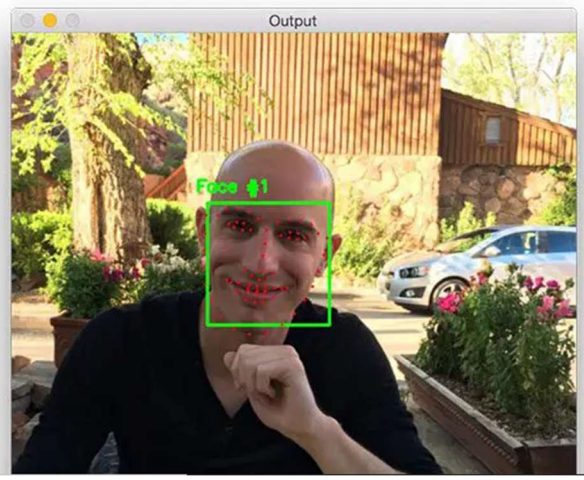


Note: If you are using Python virtual environments, make sure you upgrade the imutils inside the virtual environment.

From there, use the “Downloads” section of this guide to download the source code, example images, and pre-trained dlib facial landmark detector.

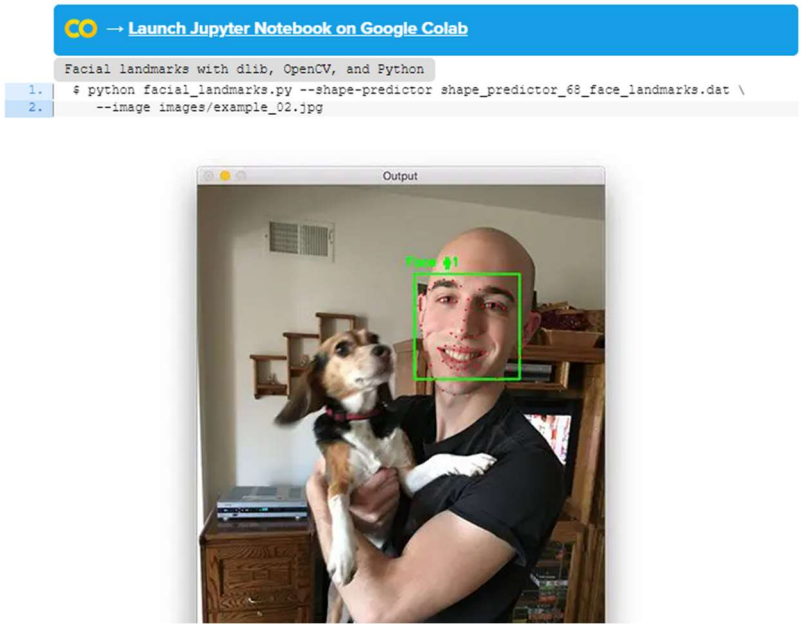
Once you’ve downloaded the .zip archive, unzip it, change directory to facial-landmarks , and execute the following command:





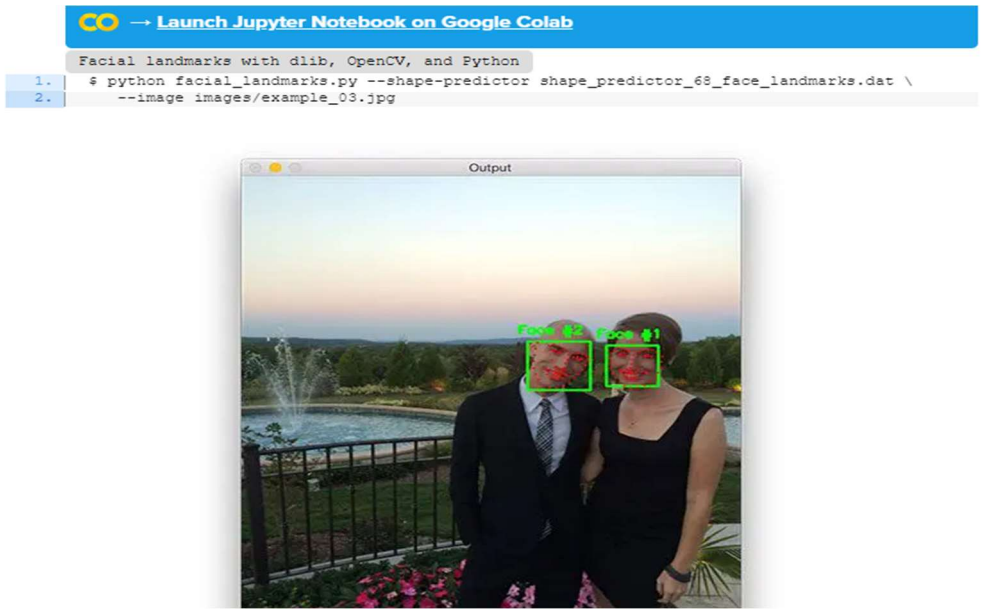
Notice how the bounding box of my face is drawn in green while each of the individual facial landmarks are drawn in red.

The same is true for this second example image:



Here we can clearly see that the red circles map to specific facial features, including my jawline, mouth, nose, eyes, and eyebrows.

Let's take a look at one final example, this time with multiple people in the image:



For both people in the image (myself and Trisha, my fiancée), our faces are not only detected but also annotated via facial landmarks as well.

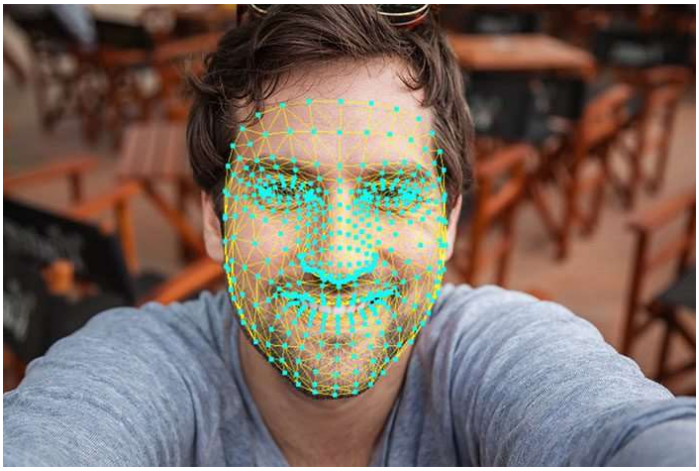
Alternative facial landmark detectors

Dlib's 68-point facial landmark detector tends to be the most popular facial landmark detector in the computer vision field due to the speed and reliability of the dlib library.

However, other facial landmark detection models exist.

To start, dlib provides an alternative 5-point facial landmark detector that is faster than the 68-point variant. This model works great if all you need are the locations of the eyes and the nose.

One of the most popular new facial landmark detectors comes from the MediaPipe library which is capable of computing a 3D face mesh



3.1.3 Extract and warp triangles

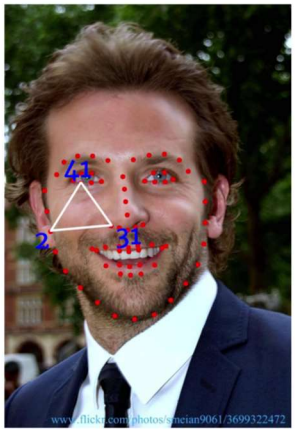
Extract and warp triangles

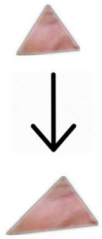
Once we have the triangulation of both faces we take the triangles of the source face and we extract them.

We also need to take the coordinates of the triangles of the destination face, so that we can warp the triangles of the source face to have same size and perspective of the matchin triangle on the destination face.

The code below shows how to warp the triangles of the source image.

```
# Warp triangles
points = np.float32(points)
points2 = np.float32(points2)
M = cv2.getAffineTransform(points, points2)
warped_triangle = cv2.warpAffine(cropped_triangle, M, (w, h))
warped_triangle = cv2.bitwise_and(warped_triangle, warped_triangle, mask=cropped_tr2_mask)
```



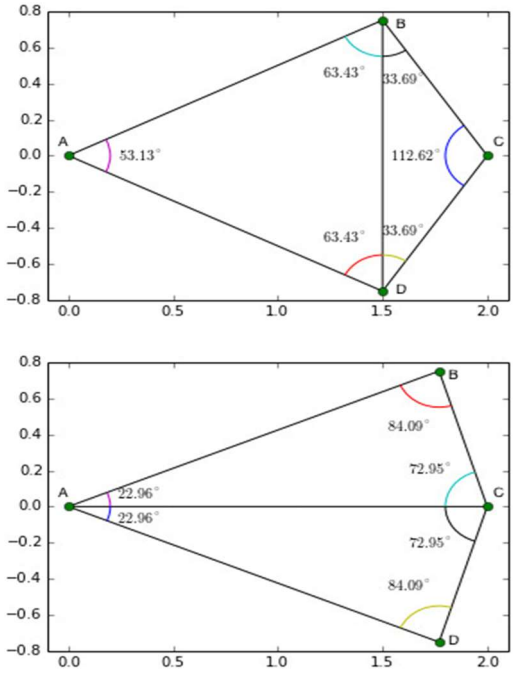


- **What is Delaunay Triangulation ?**

Given a set of points in a plane, a triangulation refers to the subdivision of the plane into triangles, with the points as vertices. In Figure 1, we see a set of landmarks on the left image, and the triangulation in the middle image. A set of points can have many possible triangulations, but Delaunay triangulation stands out because it has some nice properties. In a Delaunay triangulation, triangles are chosen such that no point is inside the circumcircle of any triangle. Figure 2. shows Delaunay triangulation of 4 points A, B, C and D. In the top image, for the triangulation to be a valid Delaunay triangulation, point C should be outside the circumcircle of triangle ABD, and point A should be outside the circumcircle of triangle BCD.

An interesting property of Delaunay triangulation is that it does not favor “skinny” triangles (i.e. triangles with one large angle).

Figure 2 shows how the triangulation changes to pick “fat” triangles when the points are moved. In the top image, the points B and D have their x-coordinates at x = 1.5, and in the bottom image they are moved to the right to x = 1.75. In the top image angles ABC and ABD are large, and Delaunay triangulation creates an edge between B and D splitting the two large angles into smaller angles ABD, ADB, CDB, and CBD. On the other hand in the bottom image, the angle BCD is too large, and Delaunay triangulation creates an edge AC to divide the large angle.



There are many algorithms to find the Delaunay triangulation of a set of points. The most obvious (but not the most efficient) one is to start with any triangulation, and check if the circumcircle of any triangle contains another point. If it does, flip the edges (as show in Figure 2.) and continue until there are no triangles whose circumcircle contains a point.

Any discussion on Delaunay triangulation has to include Voronoi diagrams because the Voronoi diagram of a set of points is mathematical dual to its Delaunay triangulation.

- **What is a Voronoi Diagram ?**

Given a set of points in a plane, a Voronoi diagram partitions the space such that the boundary lines are equidistant from neighboring points. Figure 3. shows an example of a Voronoi diagram calculated from the points shown as black dots. You will notice that every boundary line passes through the center of two points. If you connect the points in neighboring Voronoi regions, you get a Delaunay triangulation!

Delaunay triangulation and Voronoi diagram are related in more ways than one. Georgy Voronoy, the mathematician after which Voronoi diagram is named, was Boris Delaunay's Ph.D. advisor

- **Delaunay Triangulation and Voronoi Diagram using OpenCV**

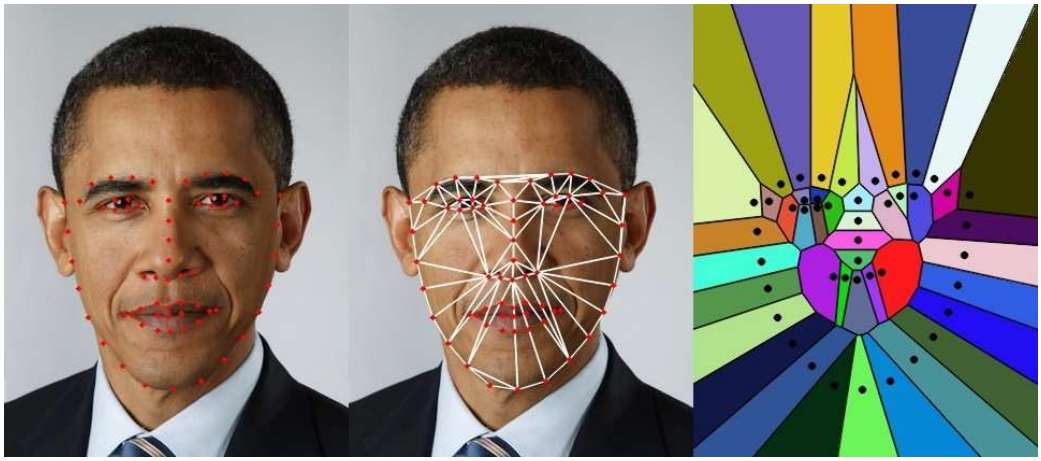


Figure 1. Left : Image of President Obama with landmarks detected using dlib.

Center : Delaunay triangulation of the landmarks. Right : Corresponding Voronoi Diagram.

In a previous post I had discussed two libraries for facial landmark detection, and had pointed to several interesting applications like Face Morphing, Face Replacement etc. that use facial landmarks. In many such applications a triangulation of facial landmarks is first found (See Figure 1), and these triangles are warped to do something interesting. This post will help us understand Delaunay triangulation and Voronoi diagrams (a.k.a Voronoi tesselation, Voronoi decomposition, Voronoi partition, and Dirichlet tessellation), and help us uncover a barely documented function in OpenCV.

- **Delaunay Triangulation & Voronoi Diagram in OpenCV**

- 1. Collect all the points in a vector.

C++

```
1    vector<Point2f>
```

```
points;
//
This
is how
you
can
add
one
point.
points.push_back(Point2f(x,y));
```

- Python

```
points = []
# This
is how
you
can
add
one
point.
points.append((x,
y))
```

- 2. Define the space you want to partition using a rectangle (**rect**). If the points you have defined in the previous step are defined on an image, this rectangle can be (0, 0, width, height). Otherwise, you can choose a rectangle that bounds all the points.

- C++

```
Mat img =
imread("image.jpg");
Size size =
img.size();
Rect rect(0,
0,
size.width,
size.height);
```

- Python

```
img = cv2.imread("image.jpg");
size = img.shape
rect = (0, 0,
size[1],
size[0])
```

- Create an instance of Subdiv2D with the rectangle obtained in the previous step.

  C++

```
Subdiv2D
subdiv(rect);
```

- Python

```
subdiv  =
cv2.Subdiv2D(rect);
```

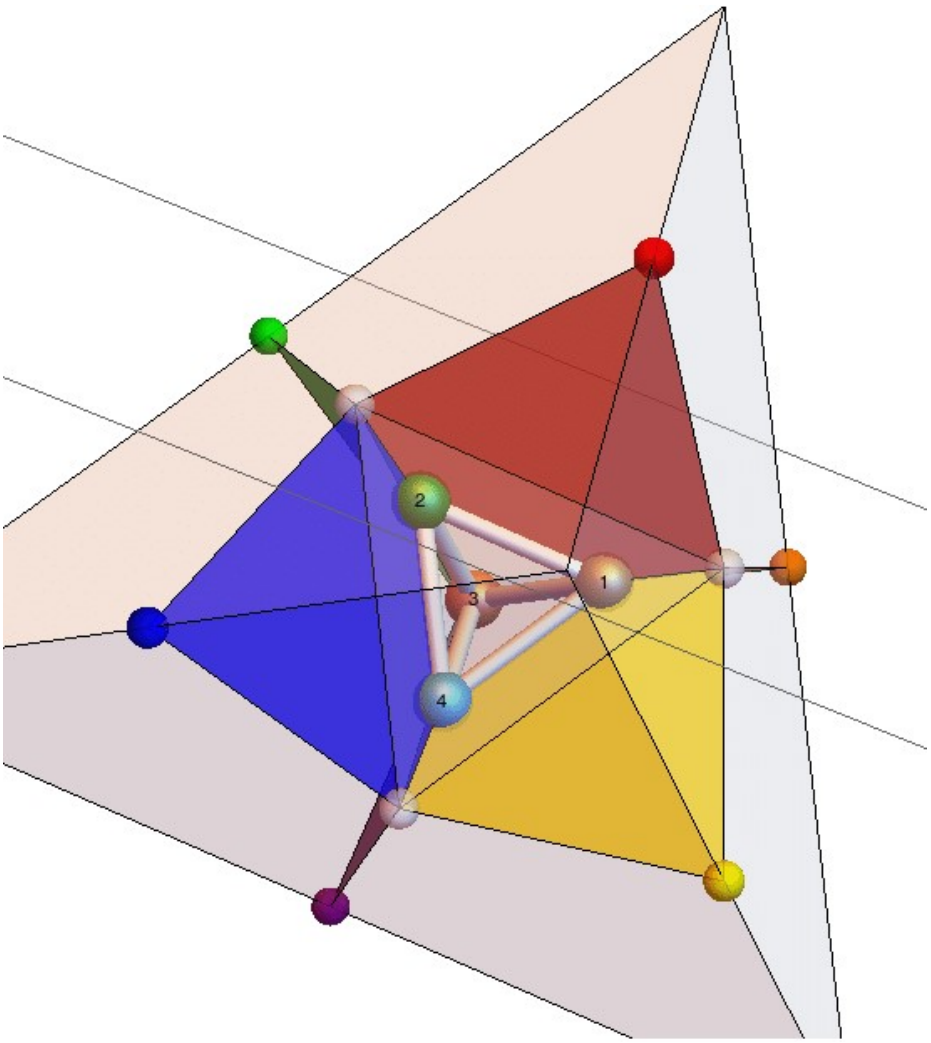
Insert the points into **subdiv** using **subdiv.insert(point)**. The video above shows an animation of triangulation as we add points to subdiv.Use **subdiv.getTriangleList** to get a list of Delaunay triangles.Use **subdiv.getVoronoiFacetList** to get the list of Voronoi facets.

- # OpenCV Example for Delaunay Triangulation & Voronoi Diagram

Here is a complete working example. I have copied some of this code from examples that come with OpenCV and simplified and modified it to suit our purpose. The python example that comes with OpenCV uses the old (and ugly) interface, and so I wrote it from scratch. This code assumes an image is stored in **image.jpg** and the points are stored in **points.txt**. Each row of **points.txt** contains the x and y coordinates of a point separated by a space. E.g.

207 242
210 269
214 297
220 322
229 349

- **Triangulation of images for Face Mesh Application**



first, we need to pass the image to the detector then that object will be used to extract landmarks using predictor. Afterward, storing extracted landmarks (x and y) into the landmarks (list). We're going to mesh up the face into triangles. This step is the core of our face-swapping. Here we will interchange each triangle with the correspondent triangle of the destination image. The triangulation of the destination image needs to have the same pattern just like that of the triangulation of the source image. This means that the connection of the connecting symbols has to be the same. So after we do the triangulation of the source image, from that triangulation we take the indexes of the (x and y) so that we can replicate the same triangulation on the destination image. Once we have the triangles indexes we loop through them and we triangulate the destination face.

3.1.4 convex_hull method:

- **What is a Convex Hull?**

  Let us break the term down into its two parts — Convex and Hull.

  A Convex object is one with no interior angles greater than 180 degrees. A shape that is not convex is called Non-Convex or Concave.

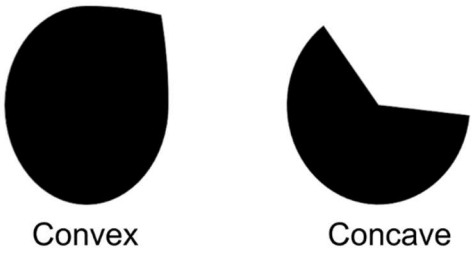


Figure 1: Example of a Convex Object and a Concave Object

Hull means the exterior or the shape of the object.

Therefore, the Convex Hull of a shape or a group of points is a tight fitting convex boundary around the points or the shape.

The Convex Hull of the two shapes in Figure 1 is shown in Figure 2. The Convex Hull of a convex object is simply its boundary. The Convex Hull of a concave shape is a convex boundary that most tightly encloses it

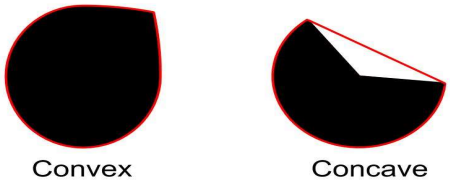


Figure 2: The Convex hull of the two black shapes is shown in red.

- **Gift Wrapping Algorithms**

Given a set of points that define a shape, how do we find its convex hull? The algorithms for finding the Convext Hull are often called **Gift Wrapping** algorithms. The video below explains a few algorithms with excellent animations:

Easy, huh? A lot of things look easy on the surface, but as soon as we impose certain constraints on them, things become pretty hard.

For example, the Jarvis March algorithm described in the video has complexity O(nh) where n is the number of input points and h is the number of points in the convex hull. Chan's algorithm has complexity O(n log h).

Is an O(n) algorithm possible? The answer is YES, but boy the history of finding a linear algorithm for convex hull is a tad embrassing.

The first O(n) algorithm was published by Sklansky in 1972. It was later proven to be incorrect. Between 1972 and 1989, 16 different linear algorithms were published and 7 of them were found to be incorrect later on!

This reminds me of a joke I heard in college. Every difficult problem in mathematics has a simple, easy to understand wrong solution!

Now, here is an embarrassing icing on the embarrassing cake. The algorithm implemented in OpenCV is one by Sklansky (1982). It happens to be INCORRECT!

It is still a popular algorithm and in a vast majority of cases, it produces the right result. This algorithm is implemented in the convexHull class in OpenCV. Let's now see how to use it.

- **convexHull in OpenCV**

  By now we know the Gift Wrapping algorithms for finding the Convex Hull work on a collection of points.

  How do we use it on images?

  We first need to binarize the image we are working with, find contours and finally find the convex hull. Let's go step by step.

  Step 1: Read the Input Image

Python

```
src = cv2.imread("sample.jpg", 1) # read input image
```

C++

```
Mat src;
src = imread("sample.jpg", 1); // read input image
```

Step 2: Binarize the input image

We perform binarization in three steps —

1. Convert to grayscale
2. Blur to remove noise
3. Threshold to binarize image

The results of these steps are shown in Figure 3. And here is the code.

Python

```
gray = cv2.cvtColor(src,
cv2.COLOR_BGR2GRAY) #
convert to grayscale
blur = cv2.blur(gray,
(3, 3)) # blur the
image
ret,
thresh = cv2.threshold(blur, 50, 255,
cv2.THRESH_BINARY)
```

C++

```
Mat gray,
blur_image,
threshold_output;
cvtColor(src, gray,
COLOR_BGR2GRAY); //
convert to
grayscale
blur(gray,
blur_image,
Size(3,
3)); //
apply blur
to
grayscaled
image
threshold(blur_image,
threshold_output, 50,
255,
THRESH_BINARY); //
apply binary
thresholding
```

As you can see, we have converted the image into binary blobs. We next need to find the boundaries of these blobs.

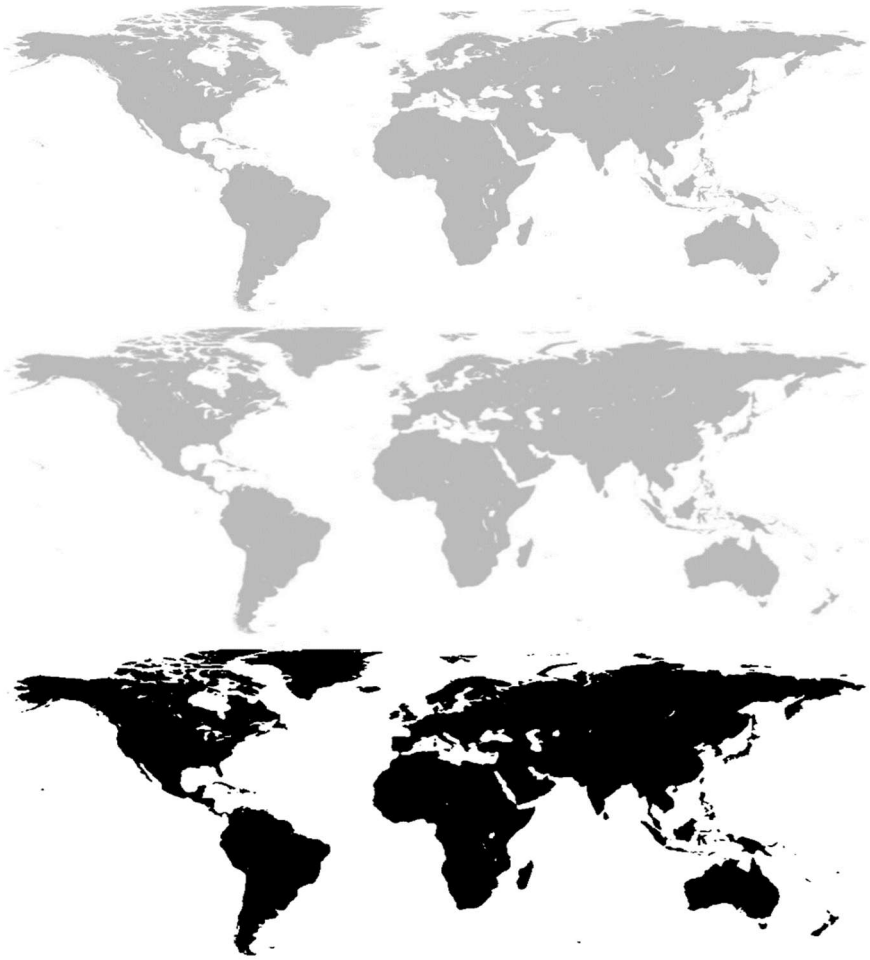


Figure 3: Topmost: Grayscaled Image. Middle: Blurred Image. Bottom: Thresholded Image

- **Step 3: Use findContour to find contours**

Next, we find the contour around every continent using the **findContour** function in OpenCV. Finding the contours gives us a list of boundary points around each blob.

If you are a beginner, you may be tempted to think why we did not simply use edge detection? Edge detection would have simply given us locations of the edges. But we are interested in finding how edges are connected to each other. **findContour** allows us to find those connections and returns the points that form a contour as a list.

Python

```
# Finding
contours
for the
thresholded
image
im2, contours,
hierarchy = cv2.findContours(thresh,
cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

C++

```
vector<
vector<Point>
>
contours; //
list of
contour
points
vector<Vec4i>
hierarchy;
// find
contours
findContours(threshold_output,
contours, hierarchy,
RETR_TREE,
CHAIN_APPROX_SIMPLE, Point(0,
0));
```

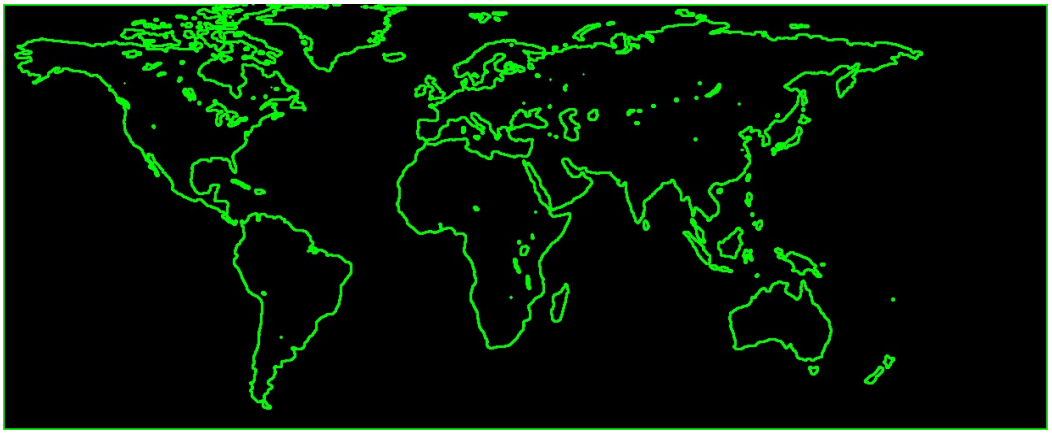


Fig. 4 Output of Finding Contours of the Thresholded Image

- **Step 4: Find the Convex Hull using convexHull**

Since we have found the contours, we can now find the Convex Hull for each of the contours. This can be done using **convexHull** function.

Python

```
#
create
hull
array
for
convex
hull
points
hull = []


#
calculate
points
for each
contour
for i in range(len(contours)):
    #
creating
convex
hull
object
for each
contour
    hull.append(cv2.convexHull(contours[i], False))
```

C++

```
1    //
     create
     hull
     array
     for
     convex
     hull
     points
2    vector< vector<Point>
     >
     hull(contours.size());
3    for(int i = 0; i
     <
     contours.size();
     i++)
4        convexHull(Mat(contours[i]),
     hull[i], False);
```

- **Step 5: Draw the Convex Hull**

The final step is to visualize the convex hulls we have just found. Of course a convex hull itself is just a contour and therefore we can use OpenCV's **drawContours**.

Python

```
1    #
     create
     an
     empty
     black
     image
2    drawing = np.zeros((thresh.shape[0],
     thresh.shape[1], 3), np.uint8)
3
4    # draw
     contours
     and hull
     points
```

```
for i in range(len(contours)):
    color_contours = (0, 255, 0) # green - color for contours
    color = (255, 0, 0) # blue - color for convex hull
    # draw ith contour
    cv2.drawContours(drawing, contours, i, color_contours, 1, 8, hierarchy)
    # draw ith convex hull object
    cv2.drawContours(drawing, hull, i, color, 1, 8)
```

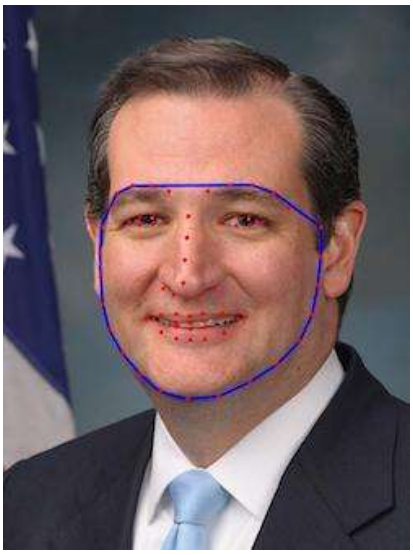
C++

```
// create a blank image (black image)
Mat drawing = Mat::zeros(threshold_output.size(), CV_8UC3);

for(int i = 0; i < contours.size(); i++) [
```

```
5       Scalar color_contours = Scalar(0, 255, 0); // green - color for contours
6       Scalar color = Scalar(255, 0, 0); // blue - color for convex hull
7       // draw ith contour
8       drawContours(drawing, contours, i, color_contours, 1, 8, vector<Vec4i>(), 0, Point());
9       // draw ith convex hull
10      drawContours(drawing, hull, i, color, 1, 8, vector<Vec4i>(), 0, Point());
11  }
```

- **Applications of Convex Hull**
- **Boundary from a set of points**

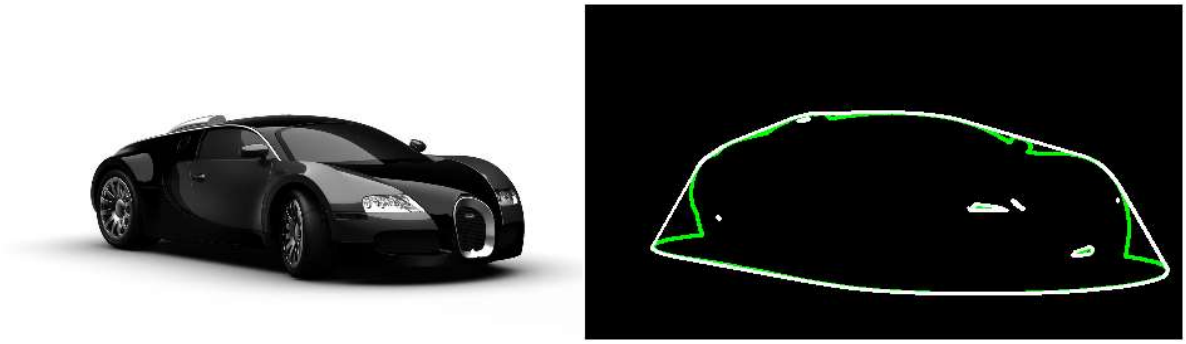
Regular readers of this blog may be aware we have used convexHull before in our face swap application. Given the facial landmarks detected using Dlib, we found the boundary of the face using the convex hull



There are several other applications, where we recover feature points information instead of contour information. For example, in many active lighting systems like Kinect, we recover a grayscale depth map that is a collection of points. We can use the convex hull of these points to locate the boundary of an object in the scene

- **Collision Avoidance**

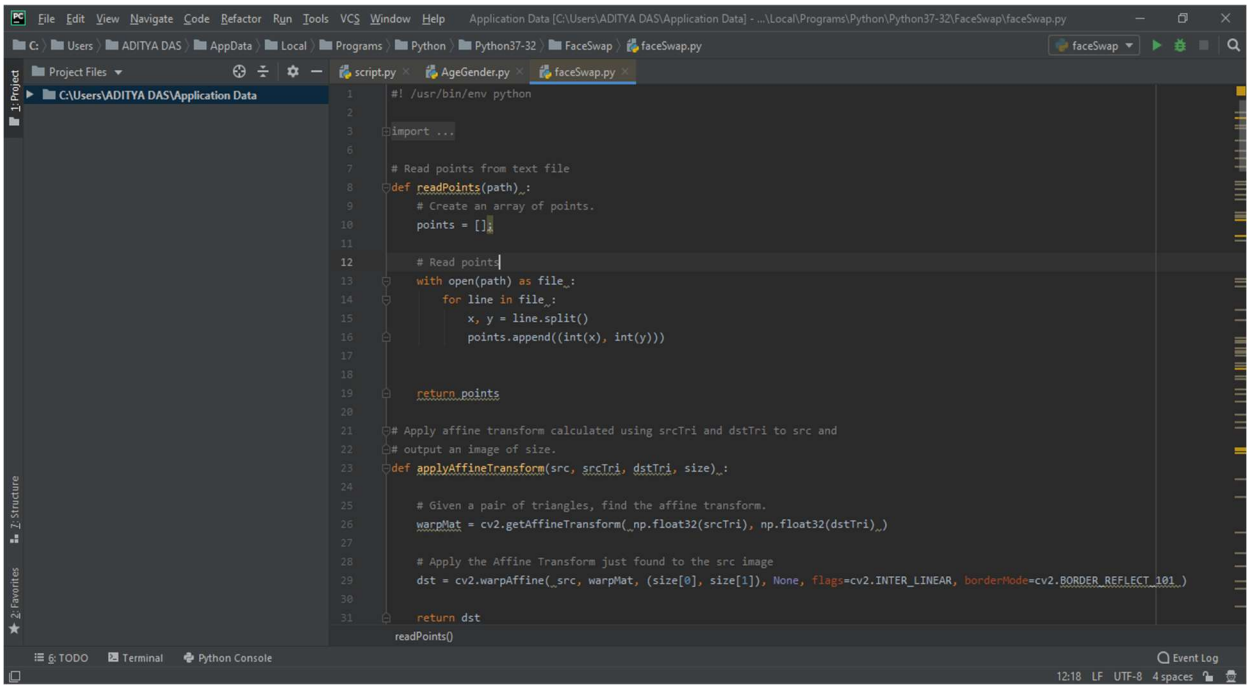
Imagine a car as a set of points and the polygon (minimal set) containing all the points. Now if the convex hull is able to avoid the obstacles, so should the car. Finding the intersection between arbitrary contours is computationally much more expensive than finding the collision between two convex polygons. So you are better off using a convex hull for collision detection or avoidance.



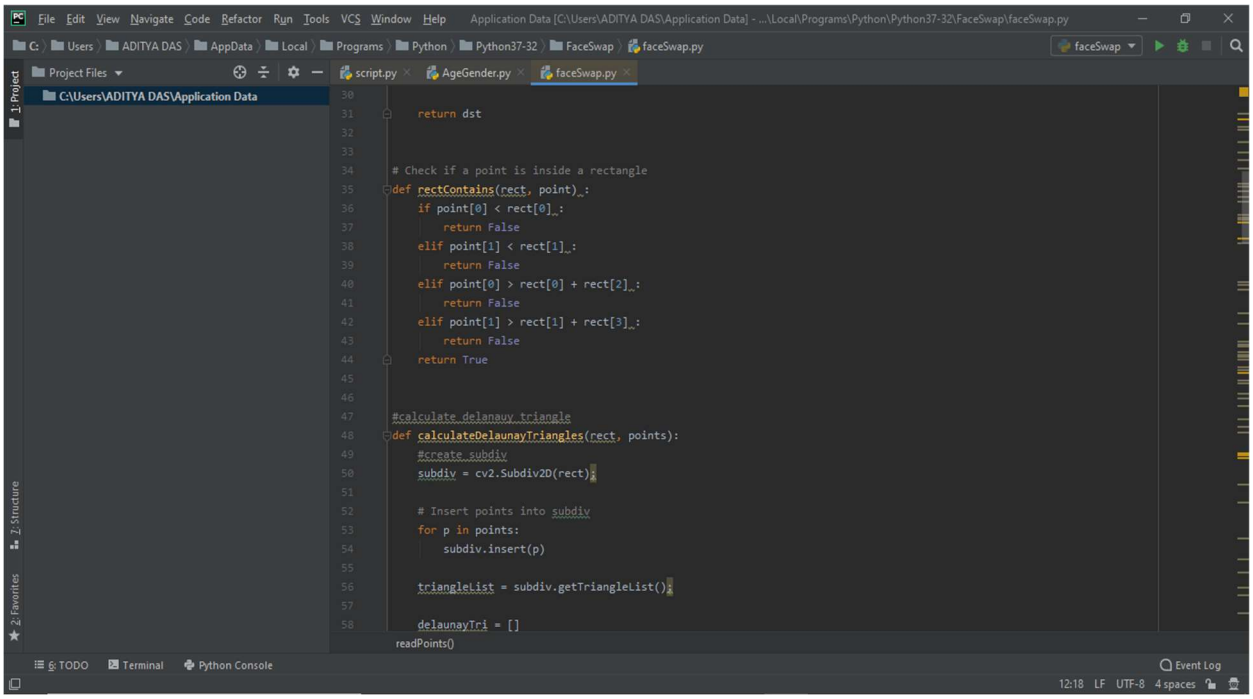
Chapter-4

4.1 Result section:

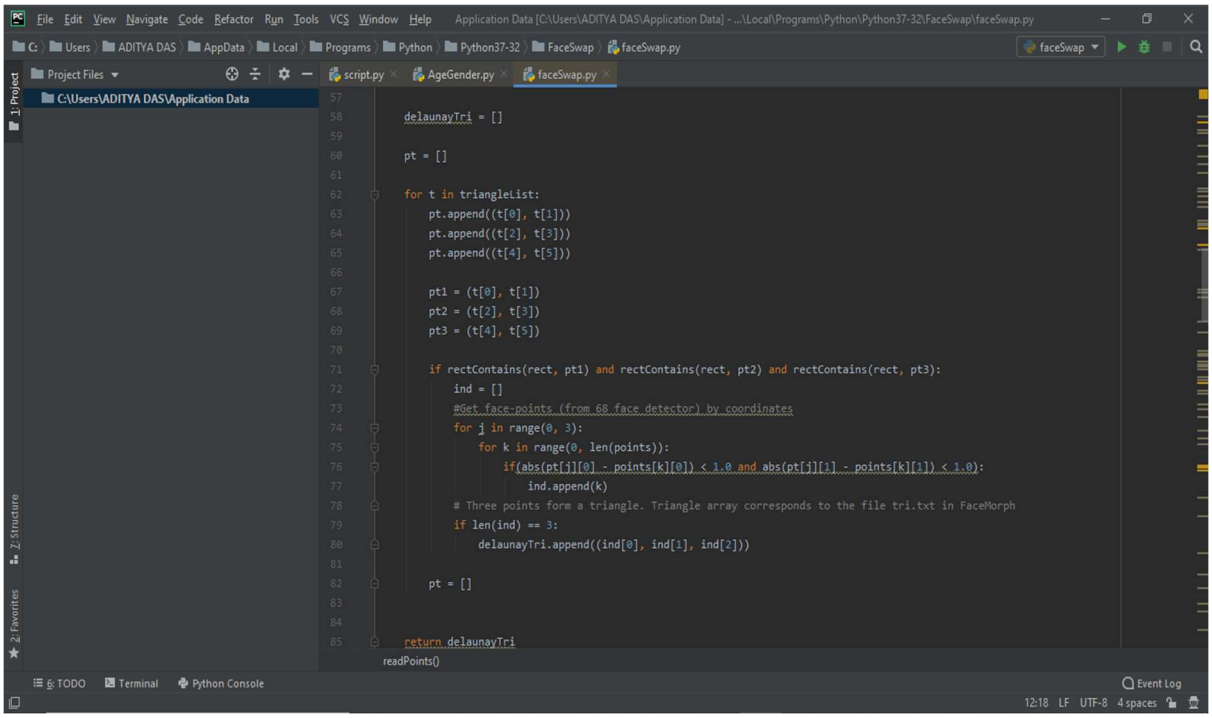
4.1.1 Snapshot of overall program:



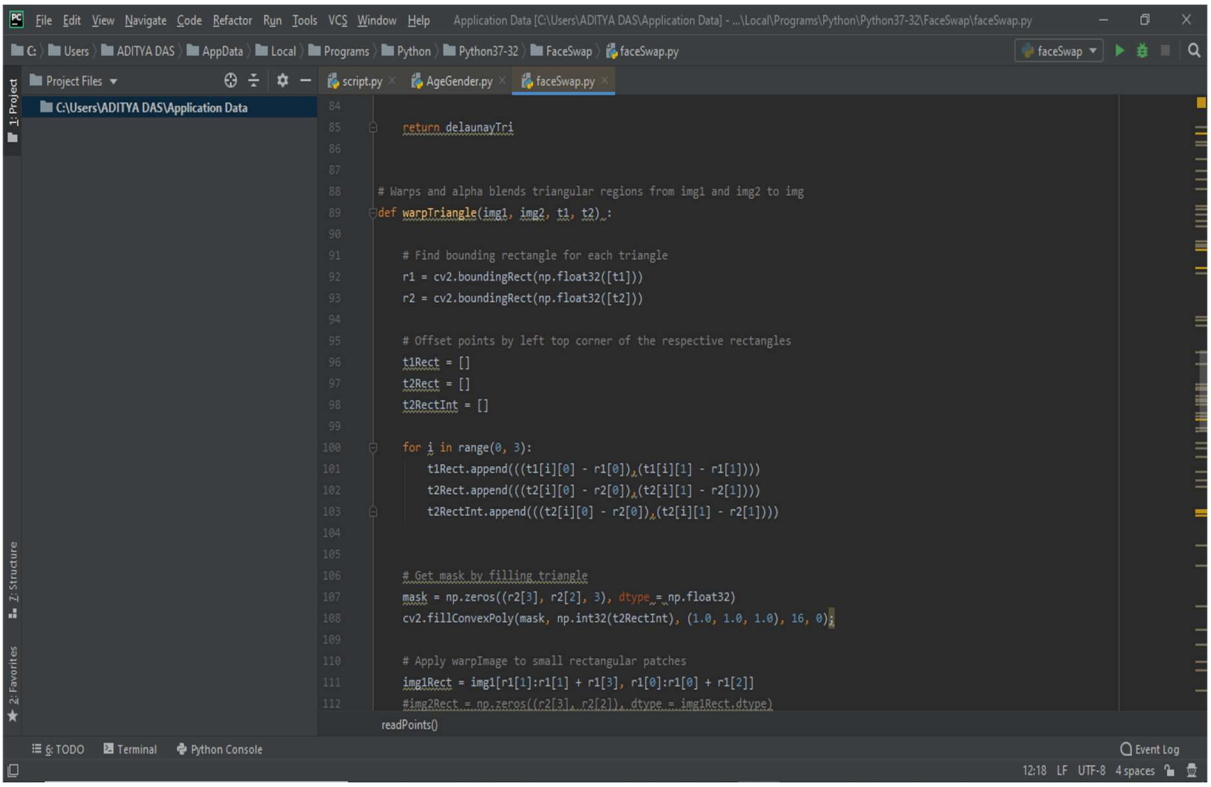
```
#! /usr/bin/env python

import ...

# Read points from text file
def readPoints(path) :
    # Create an array of points.
    points = [];

    # Read points
    with open(path) as file_ :
        for line in file_ :
            x, y = line.split()
            points.append((int(x), int(y)))


    return points

# Apply affine transform calculated using srcTri and dstTri to src and
# output an image of size.
def applyAffineTransform(src, srcTri, dstTri, size) :

    # Given a pair of triangles, find the affine transform.
    warpMat = cv2.getAffineTransform( np.float32(srcTri), np.float32(dstTri) )

    # Apply the Affine Transform just found to the src image
    dst = cv2.warpAffine( src, warpMat, (size[0], size[1]), None, flags=cv2.INTER_LINEAR, borderMode=cv2.BORDER_REFLECT_101 )

    return dst
```



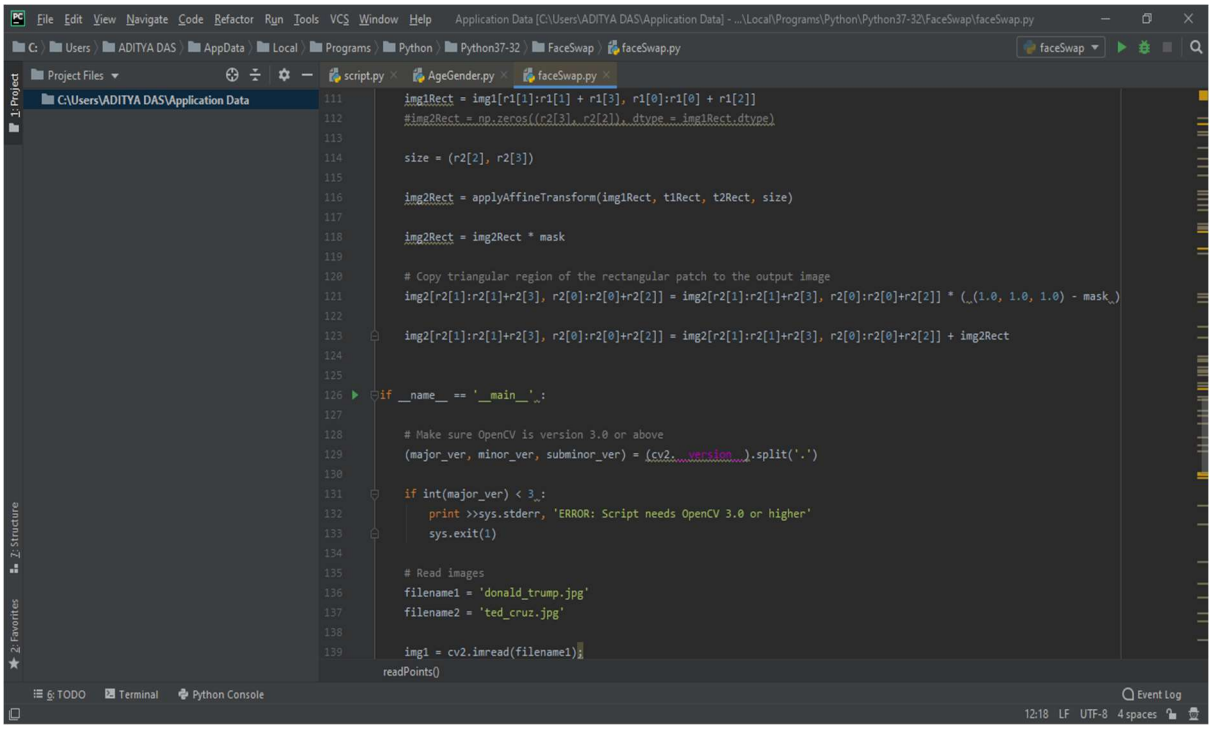
```
    return dst


# Check if a point is inside a rectangle
def rectContains(rect, point) :
    if point[0] < rect[0] :
        return False
    elif point[1] < rect[1] :
        return False
    elif point[0] > rect[0] + rect[2] :
        return False
    elif point[1] > rect[1] + rect[3] :
        return False
    return True


#calculate delanauy triangle
def calculateDelaunayTriangles(rect, points):
    #create subdiv
    subdiv = cv2.Subdiv2D(rect);

    # Insert points into subdiv
    for p in points:
        subdiv.insert(p)

    triangleList = subdiv.getTriangleList();

    delaunayTri = []
```



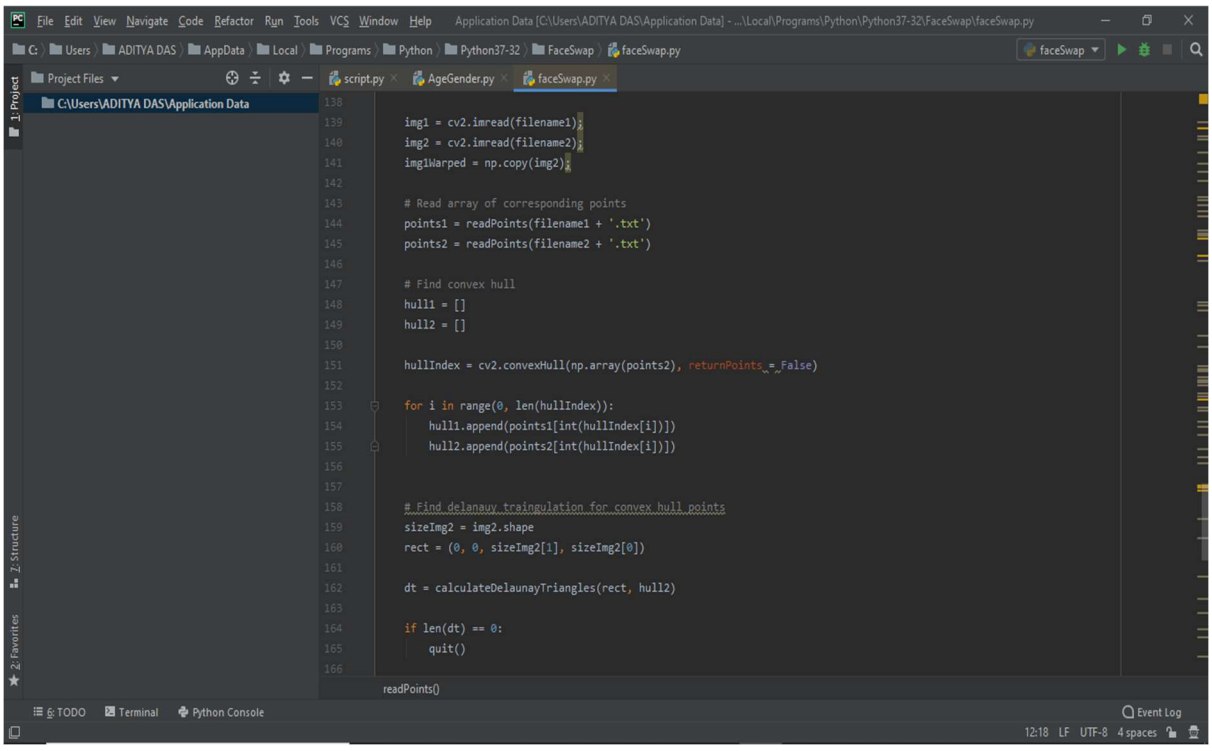
```
    delaunayTri = []

    pt = []

    for t in triangleList:
        pt.append((t[0], t[1]))
        pt.append((t[2], t[3]))
        pt.append((t[4], t[5]))

        pt1 = (t[0], t[1])
        pt2 = (t[2], t[3])
        pt3 = (t[4], t[5])

        if rectContains(rect, pt1) and rectContains(rect, pt2) and rectContains(rect, pt3):
            ind = []
            #Get face-points (from 68 face detector) by coordinates
            for j in range(0, 3):
                for k in range(0, len(points)):
                    if(abs(pt[j][0] - points[k][0]) < 1.0 and abs(pt[j][1] - points[k][1]) < 1.0):
                        ind.append(k)
            # Three points form a triangle. Triangle array corresponds to the file tri.txt in FaceMorph
            if len(ind) == 3:
                delaunayTri.append((ind[0], ind[1], ind[2]))

            pt = []


    return delaunayTri
```



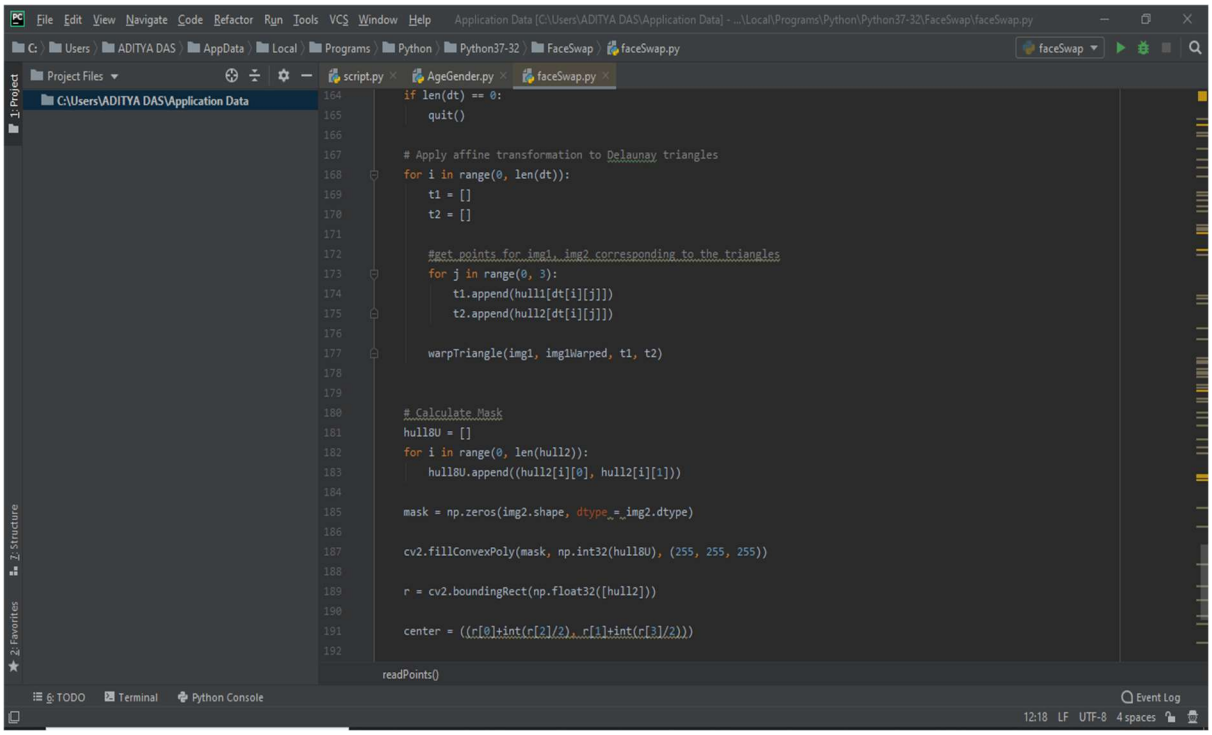
```
    return delaunayTri


# Warps and alpha blends triangular regions from img1 and img2 to img
def warpTriangle(img1, img2, t1, t2) :

    # Find bounding rectangle for each triangle
    r1 = cv2.boundingRect(np.float32([t1]))
    r2 = cv2.boundingRect(np.float32([t2]))

    # Offset points by left top corner of the respective rectangles
    t1Rect = []
    t2Rect = []
    t2RectInt = []

    for i in range(0, 3):
        t1Rect.append(((t1[i][0] - r1[0]),(t1[i][1] - r1[1])))
        t2Rect.append(((t2[i][0] - r2[0]),(t2[i][1] - r2[1])))
        t2RectInt.append(((t2[i][0] - r2[0]),(t2[i][1] - r2[1])))


    # Get mask by filling triangle
    mask = np.zeros((r2[3], r2[2], 3), dtype = np.float32)
    cv2.fillConvexPoly(mask, np.int32(t2RectInt), (1.0, 1.0, 1.0), 16, 0);

    # Apply warpImage to small rectangular patches
    img1Rect = img1[r1[1]:r1[1] + r1[3], r1[0]:r1[0] + r1[2]]
    #img2Rect = np.zeros((r2[3], r2[2]), dtype = img1Rect.dtype)
```



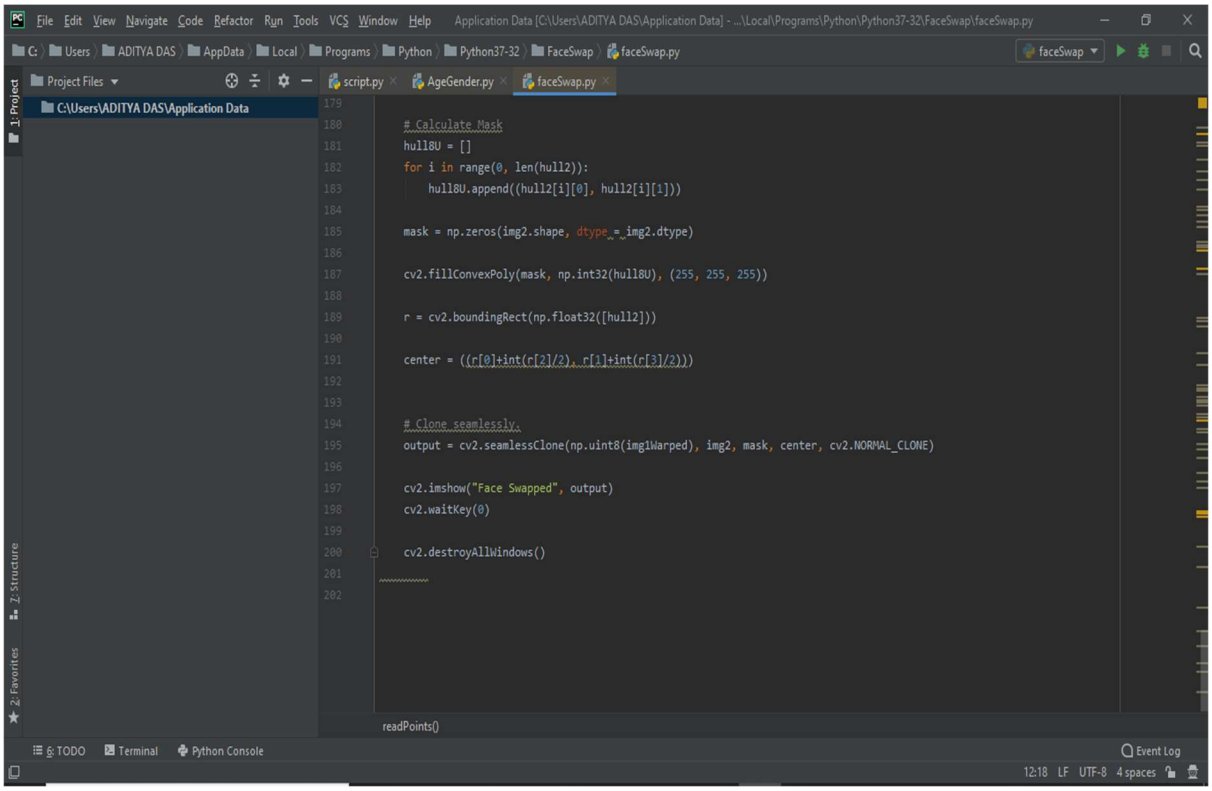
```
    img1Rect = img1[r1[1]:r1[1] + r1[3], r1[0]:r1[0] + r1[2]]
    #img2Rect = np.zeros((r2[3], r2[2]), dtype = img1Rect.dtype)

    size = (r2[2], r2[3])

    img2Rect = applyAffineTransform(img1Rect, t1Rect, t2Rect, size)

    img2Rect = img2Rect * mask

    # Copy triangular region of the rectangular patch to the output image
    img2[r2[1]:r2[1]+r2[3], r2[0]:r2[0]+r2[2]] = img2[r2[1]:r2[1]+r2[3], r2[0]:r2[0]+r2[2]] * ( (1.0, 1.0, 1.0) - mask )

    img2[r2[1]:r2[1]+r2[3], r2[0]:r2[0]+r2[2]] = img2[r2[1]:r2[1]+r2[3], r2[0]:r2[0]+r2[2]] + img2Rect


if __name__ == '__main__':

    # Make sure OpenCV is version 3.0 or above
    (major_ver, minor_ver, subminor_ver) = (cv2.__version__).split('.')

    if int(major_ver) < 3 :
        print >>sys.stderr, 'ERROR: Script needs OpenCV 3.0 or higher'
        sys.exit(1)

    # Read images
    filename1 = 'donald_trump.jpg'
    filename2 = 'ted_cruz.jpg'

    img1 = cv2.imread(filename1);
```



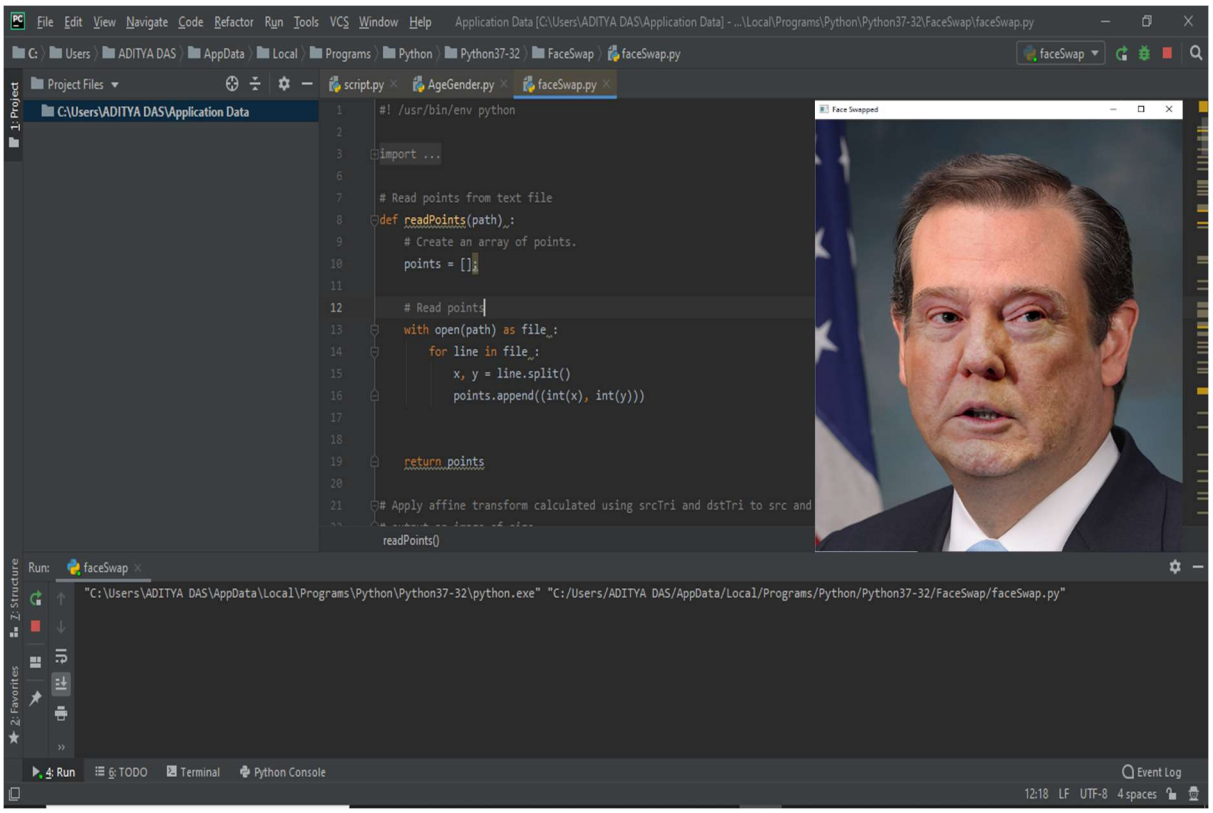
```
    img1 = cv2.imread(filename1);
    img2 = cv2.imread(filename2);
    img1Warped = np.copy(img2);

    # Read array of corresponding points
    points1 = readPoints(filename1 + '.txt')
    points2 = readPoints(filename2 + '.txt')

    # Find convex hull
    hull1 = []
    hull2 = []

    hullIndex = cv2.convexHull(np.array(points2), returnPoints = False)

    for i in range(0, len(hullIndex)):
        hull1.append(points1[int(hullIndex[i])])
        hull2.append(points2[int(hullIndex[i])])


    # Find delanauy traingulation for convex hull points
    sizeImg2 = img2.shape
    rect = (0, 0, sizeImg2[1], sizeImg2[0])

    dt = calculateDelaunayTriangles(rect, hull2)

    if len(dt) == 0:
        quit()
```



```
    if len(dt) == 0:
        quit()

    # Apply affine transformation to Delaunay triangles
    for i in range(0, len(dt)):
        t1 = []
        t2 = []

        #get points for img1, img2 corresponding to the triangles
        for j in range(0, 3):
            t1.append(hull1[dt[i][j]])
            t2.append(hull2[dt[i][j]])

        warpTriangle(img1, img1Warped, t1, t2)


    # Calculate Mask
    hull8U = []
    for i in range(0, len(hull2)):
        hull8U.append((hull2[i][0], hull2[i][1]))

    mask = np.zeros(img2.shape, dtype = img2.dtype)

    cv2.fillConvexPoly(mask, np.int32(hull8U), (255, 255, 255))

    r = cv2.boundingRect(np.float32([hull2]))

    center = ((r[0]+int(r[2]/2), r[1]+int(r[3]/2)))
```



```
    # Calculate Mask
    hull8U = []
    for i in range(0, len(hull2)):
        hull8U.append((hull2[i][0], hull2[i][1]))

    mask = np.zeros(img2.shape, dtype = img2.dtype)

    cv2.fillConvexPoly(mask, np.int32(hull8U), (255, 255, 255))

    r = cv2.boundingRect(np.float32([hull2]))

    center = ((r[0]+int(r[2]/2), r[1]+int(r[3]/2)))


    # Clone seamlessly.
    output = cv2.seamlessClone(np.uint8(img1Warped), img2, mask, center, cv2.NORMAL_CLONE)

    cv2.imshow("Face Swapped", output)
    cv2.waitKey(0)

    cv2.destroyAllWindows()
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Application Data [C:\Users\ADITYA DAS\Application Data] - ...\Local\Programs\Python\Python37-32\FaceSwap\faceSwap.py
C: Users ADITYA DAS AppData Local Programs Python Python37-32 FaceSwap faceSwap.py
faceSwap
Project Files
C:\Users\ADITYA DAS\Application Data
script.py
AgeGender.py
faceSwap.py
#! /usr/bin/env python
import ...
# Read points from text file
def readPoints(path):
# Create an array of points.
points = []
# Read points
with open(path) as file:
for line in file:
x, y = line.split()
points.append((int(x), int(y)))
return points
# Apply affine transform calculated using srcTri and dstTri to src and
readPoints()
Face Swapped
Run: faceSwap
"C:\Users\ADITYA DAS\AppData\Local\Programs\Python\Python37-32\python.exe" "C:/Users/ADITYA DAS/AppData/Local/Programs/Python/Python37-32/FaceSwap/faceSwap.py"
4: Run
6: TODO
Terminal
Python Console
Event Log
12:18 LF UTF-8 4 spaces
1: Project
7: Structure
2: Favorites

4.1.2 TESTING:

- **UNIT TESTING**

  Unit is undertaken after coding of a module is complete, all syntax errors have been reomoved, and the code has been reviewed. This activity is typically undertaken by the coder of the module himself in the coding phase. Before carrying out unit testing, the unit test cases have to be designed and the test environment for the unit under test has to be developed. In this section, we first discuss the environment needed to perform unit testing.

  - **Driver and stub modules**

  In order to test a single module, we need a complete environment to provide all relevant code that is necessary for execution of the module. That is, besides the module under ties the following are needed to test the module:

  - The procedures belonging to other modules that the module under test calls.
  - Non-local data structures that the module accesses.
  - A procedure to call the functions of the module under test with appropriate parameters.

  Modules required to provide the necessary environment (which either call, provide the required global data, or are called by the module under test) are usually not available unit

  They too have been unit tested. In this context, stubs and drivers are designed to provide the complete environment for a module so that testing can be carried out. The role of stub and driver modules is pictorially shown in Figure 10.3. We briefly discuss the stub and driver modules that are required to provide the necessary environment for carrying out unit testing are briefly discussed in the following.

  Stub: A sub module consists of several stub procedures that are called by the module under test. A stub procedure is a dummy procedure that takes the same parameters as the function called by the unit under test but has a highly simplified behavior. For example, a stub procedure may produce the expected behavior using a simple table look up mechanism, rather than performing actual computations.

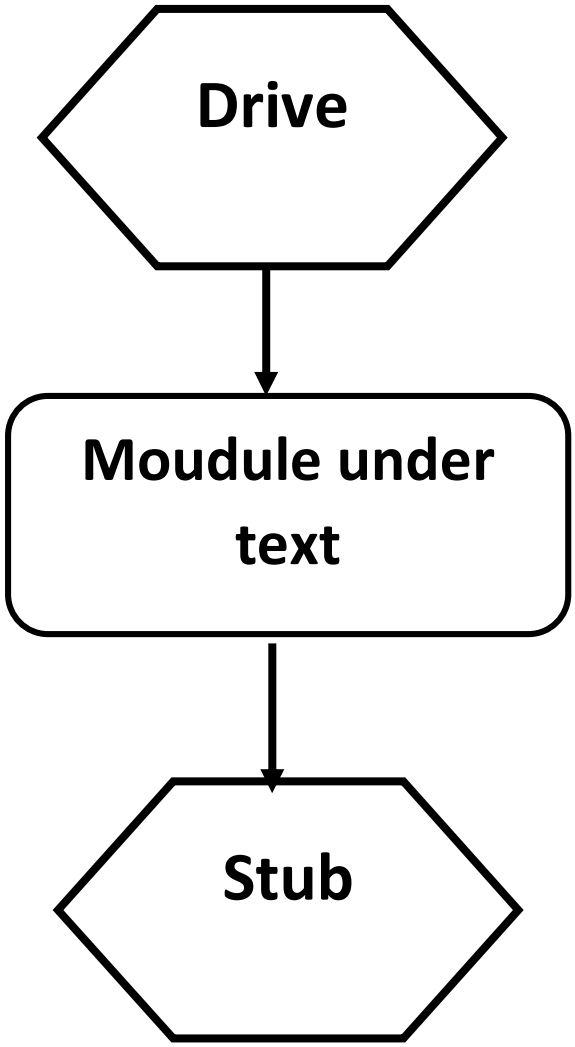


FIGURE Unit testing with the help of driver and stub modules.

Driver: A driver module contains the non-local data structures that are accessed by the module under test. Additionally, it should also have the code to call the different functions of the unit under test with appropriate parameter values for testing.

- **BLACK-BOX TESTING**

In black-box testing, test cases are designed from an examination of the input/output values only and no knowledge of design or code is required. The following are the two main approaches available to design black box test cases:

- Equivalence class partitioning
- Boundary value analysis

In the following subsections, we will elaborate these two test case design techniques.

- **Equivalence Class Partitioning**

In the equivalence class partitioning approach, the domain of input values to the unit under test is partitioned into a set of equivalence classes. The partitioning is done such that for every input data belonging to the same equivalence class, the program behaves similarly. Equivalence classes for a unit under test can be designed by examining the input data and output data. The following are two general guidelines for designing the equivalence classes:

- If the input data values to a system can be specified by a range of values, then one valid and two invalid equivalence classes can be defined. For example, if the equivalence class is the set of integers in the range 1 to 10 (i.e., [1, 10]), then the two invalid equivalence classes are [- ,0], [11,+] and the valid equivalence class is [1,10].
- If the input data assumes values from a set of discrete members of some domain, then one equivalence class for the valid input values and another equivalence class for the invalid input values should be defined. For example, if the valid equivalence classes are {A, B, C), then the invalid equivalence class is U-{A,B,C}, where U is the universe of all possible input values.
- Boundary Value Analysis

A type of programming error that is frequency committed by programmers is missing out on the special consideration that should be to the values at the boundaries of different equivalence classes of inputs. The reason behind programmers committing such errors might purely be due to psychological factors. Programmers often fail to properly address the special processing required by the input values that lie at the boundary of the different equivalence classes. For example, a programmer may improperly use < instead of <=, or conversely <= for <, etc...)

To design boundary value test cases, it is required to examine the equivalence classes to check if any of the equivalence classes contains a range of values. For those equivalence classes that are not a range of values (i.e., consist of a discrete collection of values) no boundary value test cases can be defined. For an equivalence class that is a range of values, the boundary values need to be included in the test suite. For

example, if equivalence contains the integers in the range 1 to 10, then the boundary value test suite is {0, 1, 10, 11}.

- Summary of the Black-box Test Suite Design Approach

We now summarise the important steps in the black-box test suite design approach:

- Examine the input and output values of the program.
- Identify the equivalence classes.
- Design equivalence class test cases by picking one representative value from each equivalence class.
- Design the boundary value test cases as follows. Examine if any equivalence class is a range of values. Include the values at the boundaries of such equivalence classes in the test suite.

The strategy for black-box testing is intuitive and simple. For black-box testing, the most important step is the identification of the equivalence classes. Often, the identification of the equivalence classes is not straightforward. However, with little practice one would be able to identify all equivalence classes in the input data domain. Without practice, one may overlook many equivalence classes in the input data set. Once the equivalence classes are identified, the equivalence class and boundary value test cases can be selected almost mechanically.

- **WHITE-BOX TESTING**

White-box testing is an important type of unit testing. A large number of white-box testing strategies exist. Each testing strategy essentially designs test cases based on analysis of some aspect of source code and is based on some heuristic. We first discuss some basic concepts associated with white-box testing, and follow it up with a discussion on specific testing strategies.

- Basic Concepts

A white-box testing strategy can either be coverage-based or fault-based. Fault-based testing

A fault-based testing strategy targets to detect certain types of faults. An example of a fault-based strategy is mutation testing, which is discussed later in this section.
Coverage-based testing

A coverage-based testing strategy attempts to execute (or cover) certain elements of a program. Popular examples of coverage-based testing strategies are statement branch coverage, multiple condition coverage, and path coverage-based testing.

- **INTEGRATION TESTING**

Integration testing is carried out after all (or at least some of) the modules have been unit tested. Successful completion of unit testing, to a large extent, ensures that the unit (or module) as a whole works satisfactorily. In this context, the objective of integration testing is to detect the errors at the module interfaces (call parameters). For example, it is checked that no parameter mismatch occurs when one module invokes the functionality of another module. Thus, the primary objective of integration testing is to test the module interfaces, i.e., there are no

errors in parameter passing, when one module invokes the functionality of another module.
During integration testing, different modules of a system are integrated in a planned manner using an integration plan. The integration plan specifies the steps and the order in which modules are combined to realise the full system. After each integration step, the partially integrated system is tested.
An important factor that guides the integration plan is the module dependency graph. Thus, by examining the structure chart, the integration plan can be developed. Any one (or a mixture) of the following approaches can be used to develop the test plan:

- Big-bang approach to integration testing
- Top-down approach to integration testing
- Bottom-up approach to integration testing
- Mixed (also called sandwiched) approach to integration testing

In the following subsections, we provide an overview of these approaches to integration testing.

- **Big-bang approach to integration testing**

Big-bang testing is the most obvious approach to integration testing. In this approach, all the modules making up a system are integrated in a single step. In simple words, all the unit tested modules of the system are simply linked together and tested. However, this technique can meaningfully be used only for very small systems. The main problem with this approach is that once a failure has been detected during integration testing, it is very difficult to localise the error as the error may potentially exist in any of the modules. Therefore, debugging errors reported during big-bang integration testing are very expensive to fix. As a result, bing-bang integration testing is almost never used for large programs.

- **Bottom-up approach to integration testing**

Large software products are often made up of several subsystems. A subsystem might consist of many modules which communicate among each other through well defined interfaces. In bottom-up integration testing, first the modules for the each subsystem are integrated. Thus, the subsystems can be integrated separately and independency. The primary purpose of carrying out the integration testing a subsystem is to test whether the interfaces among various modules making up the subsystem work satisfactorily. The test cases must be carefully chosen to exercise the interfaces in all possible manners.

In a pure bottom-up testing no stubs are required, and only test-drivers are required. Large software systems normally require several levels of subsystem testing, lower-level subsystems are successively combined to form higher-level subsystems. The principal advantage of bottom-up integration testing is that several disjoint subsystems can be tested simultaneously. Another advantage of bottom-up testing is that the low-level modules get tested thoroughly, since they are exercised in each integration step. Since the low-level modules do I/O and other critical functions, testing the low-level modules thoroughly increases the reliability of the system. A disadvantage of bottom-up testing is the complexity that occurs when the system is made up of a large number of small subsystems that are at the same level. This extreme case corresponds to the big-bang approach.

- **Top-down approach to integration testing**

Top-down integration testing starts with the root module in the structure chart and one or two subordinate modules of the root module. After the top-level 'skeleton' has been tested, the modules that are at the immediately lower layer of the 'skeleton' are combined with it and tested. Top-down integration testing approach requires the use of program stubs to simulate the effect of lower-level routines that are called by the routines under test. Pure top-down integration does not require any driver routines. An advantage top-down integration testing is that it requires writing only stubs, and stubs are simpler to write compared to drivers. A disadvantage of the top-down integration testing approach is that in the absence of lower-level routines, it becomes difficult to exercise the top-level routines in the desired manner since the lower level routines usually perform input/output (I/0) operations.

- **SYSTEM TESTING**

After all the units of a program have been integrated together and tested, system testing is taken up.
The system testing procedures are the same for both

object-oriented and procedural programs, since system test cases are designed solely based on the SRS document and the actual implementation (procedural or object-oriented) is immaterial.

There are three main kinds of system testing. These are essentially similar tests, but differ in who carries out the testing:

- **Alpha Testing:** Alpha testing refers to the system testing carried out by the test team within the developing organisation.
- **Beta Testing:** Beta testing is the system testing performed by a select group of friendly customers.

- **Acceptance Testing:** Acceptance testing is the system testing performed by the customer to determine whether to accept the delivery of the system.

As can be observed from the above discussions, in the different types of system tests, the test cases can be the same,but the difference is with respect to who designs test cases and carries out testing.

Before a fully integrated system is accepted for system testing, smoke testing is performed. Smoke testing is done to check whether at least the main functionalities of the software are working properly. Unless the software is stable and at least the main functionalities are working satisfactorily, system testing is not undertaken.The functionality tests are designed to check whether the software satisfies the functional requirements as documented in the SRS document. The performance tests, on the other hand, test the conformance of the system with the non-functional requirements of the system. We have already discussed how to design the functionality test cases by using a black-box approach. So, in the following subsection we discuss only smoke and performance testing.

4.3 Conclusion & Future scope:

AI is making lots of progress in the scientific sector. Artificial Intelligence can handle large quantities of data and processes it quicker than human minds. This makes it perfect for research where the sources contain high data volumes. AI is already making breakthroughs in this field. A great example is 'Eve,' which is an AI-based robot. It discovered an ingredient of toothpaste that can cure a dangerous disease like Malaria. Imagine a common substance present in an everyday item that is capable of treating Malaria; it's a significant breakthrough, no doubt. Drug discovery is a fast-growing sector, and AI is aiding the researchers considerably in this regard.
Biotechnology is another field where researchers are using AI to design microorganisms for industrial applications. Science is witnessing significant changes thanks to AI and ML. Learn more about the benefits of AI.

2. AI in Cyber Security

Cybersecurity is another field that's benefitting from AI. As organizations are transferring their data to IT networks and cloud, the threat of hackers is becoming more significant.
One triumphant attack can wreak havoc on an organization. To keep their data and resources secure, organizations are making massive investments in cybersecurity. The future scope of AI in cybersecurity is bright.
Cognitive AI is an excellent example of this field. It detects and analyses threats, while also providing insights to the analysts for making better-informed decisions. By using Machine Learning algorithms and Deep Learning networks, the AI gets better and more durable over time. This makes it capable of fighting more advanced threats that might develop with them.
Many institutions are using AI-based solutions to automate the repetitive processes present in cybersecurity. For example, IBM has IBM Resilient, which is an agnostic and open platform that gives infrastructure and hub for managing security responses.
Another field is fraud detection. AI can help in detecting frauds and help organizations and people in avoiding scams. For example, Recurrent Neural

Networks are capable of detecting fraud in their early stages. They can scan extensive quantities of transactions quickly and classify them according to their trustworthiness.

By identifying fraudulent transactions and tendencies, organizations can save a lot of time and resources. It surely lessens the risk of losing money.

3. AI in Data Analysis

Data analysis can benefit largely from AI and ML. AI algorithms are capable of improving with iterations, and this way, their accuracy, and precision increase accordingly. AI can help data analysts with handling and processing large datasets. AI can identify patterns and insights that human eyes can't notice without putting in a lot of effort. Moreover, it is faster and more scalable at doing so. For example, Google Analytics has Analytics Intelligence, which uses machine learning to help webmasters get insights on their websites faster.

You can ask Analytics Intelligence a question in simple English, and it would give you a prompt reply. It also provides webmasters with Smart Lists, Smart Goals, Conversion Probability, and other features that help the webmaster in improving the results of their site.

The scope of AI in data analytics is rising rapidly. Another example of AI applications in this sector is predicting outcomes from data. Such systems use the analytics data to predict results and the appropriate course of action to achieve those results. Learn more about AI applications.

As mentioned earlier, AI systems can handle tons of data and process it much faster than humans. So, they can take customer data and make more accurate predictions of customer behavior, preferences, and other required factors. Helixa.ai is a great example of such an AI application. They use AI to provide insights into customer (or audience) behavior for higher accuracy and better results. Agencies and marketers can use their services to build precise buyer personas and create better-targeted ad campaigns.

4. AI in Transport

The transport sector has been using AI for decades. Airplanes have been using autopilot to steer them in the air since 1912. An autopilot system controls the trajectory of a plane, but it isn't restricted to aircraft alone. Ships and spacecraft also use autopilot to help them maintain the correct course.
Autopilot helps the human operator and assists them in heading in the right direction. A pilot of a modern aircraft usually works for 7 minutes; the autopilot handles most of the steering of the plane. This allows the pilots to focus on other more important areas of the flight, such as the weather and the trajectory of the plane.

Another area where the future scope of AI is quite broad is driverless cars. Many companies are developing autonomous vehicles, which will rely heavily on AI and ML to operate optimally. Experts believe self-driving cars will bring many long-term and short-term benefits, including lower emissions and enhanced road safety. For example, self-driving cars will be free from human errors, which account for 90% of traffic accidents. Many companies, including Tesla and Uber, are developing these vehicles.

5. AI in Home

AI has found a special place in people's homes in the form of Smart Home Assistants. Amazon Echo and Google Home are popular smart home devices that let you perform various tasks with just voice commands.
You can order groceries, play music, or even switch on/off the lights in your living room with just a few voice commands. Both of them rely on Voice Recognition technologies, which are a result of Artificial Intelligence and Machine Learning. They constantly learn from the commands of their users to understand them better and become more efficient.
Smart assistants are also present in mobile phones. Apple's Siri and Google Assistant are great examples of this sort. They also learn to recognize their users' voices to interpret them better all the time. And they can perform a plethora of tasks. Microsoft also has a smart assistant, which is called Cortana.
You can use these smart assistants for various tasks such as:

- Playing a song
- Asking a question
- Buying something online
- Opening an app

There's a lot of room left for improvement, but surely, the scope of AI in the smart home sector is booming.

6. AI in Healthcare

The medical sector is also using this technology for its advantages. AI is helping medical researchers and professionals in numerous ways.

For example, the Knight Career Institute and Intel have made a collaborative cancer cloud. This cloud takes data from the medical history of cancer (and similar) patients to help doctors in making a better diagnosis. Preventing cancer from moving to higher stages is its most effective treatment at this time.

We've already mentioned how AI is helping researchers in their field too. Apart from finding a cure for cancer, some organizations are using AI to help patients get telemedicine. The UK's National Health Service uses Google's DeepMind platform to detect health risks in people through apps.

Wrong diagnoses are a significant problem in the medical sector. AI can help doctors in avoiding these errors by providing them with relevant databases and recommendations. It can analyze the database of patients with similar symptoms and suggest the treatment that was the most successful in those cases.

Many major organizations, including IBM and Microsoft, are collaborating with medical institutions to solve the various problems present in the healthcare sector.

AI can also help in reducing medical costs by preventing diseases beforehand and helping doctors in making better diagnoses. BCIs (Brain-computer Interfaces) is another area where the medical sector is utilizing AI. These interfaces help in predicting problems related to speaking or moving that might develop due to problems in the brain. They use AI to help these patients overcome these issues, too, by decoding neural activates.

7. Python in image processing:

Images define the world, each image has its own story, it contains a lot of crucial information that can be useful in many ways. This information can be obtained with the help of the technique known as Image Processing.

It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at a time and extract useful insights from them. It has a wide range of applications in almost every field. Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

Through this article, you will learn about classical algorithms, techniques, and tools to process the image and get the desired output.

REFERANCE & BOOKS

- https://www.edureka.co/blog/python-opencv-tutorial/
- https://learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/
- https://learnopencv.com/facial-landmark-detection/
- https://learnopencv.com/tag/face-recognition/
- https://learnopencv.com/warp-one-triangle-to-another-using-opencv-c-python/
- https://learnopencv.com/face-swap-using-opencv-c-python/
- https://www.analyticsvidhya.com/blog/2021/07/give-hermione-granger-a-cool-pair-of-glasses-by-building-snapchat-filter-using-opencv/
- https://1lib.in/book/3647627/667c74
- https://1lib.in/book/3306705/d9c0e3?dsource=recommend

❖ **Project Contributors**