**AIM:**

**E**LECTRIC discharge is widespread in nature and is commonly used in the operation of industrial equipment. In an effort to understand the properties of electrical discharge, researchers have investigated several physical quantities of discharge, including the voltage, current, optical spectrum, ultrahigh-frequency electromagnetic waves, number of discharges, phase angles (for ac discharge), etc. However, such research works have been carried out without considering the relevant optical characteristics of discharge images even though the initial studies were based on the image, such as that of the original meaning of corona.

**OBJECTIVE:**

As a matter of fact, optical measurements are better suited than electromagnetic measurements for the purpose of determining discharge geometries. If the optical characteristics of the discharge image can be incorporated in the diagnosis, the reliability of traditional recognition methods may be increased.

**PROBLEM STATEMENT**

The traditional research of the discharge image was conducted from a qualitative point of view, such as morphology description, strong or weak light intensity, etc.. However, only a few studies focused on quantitative evaluation can be found. With the development of computer techniques, the digital image processing methods have been applied extensively to study discharge characteristics, such as breakdown paths, discharge area, etc., especially in ultraviolet (UV), which can help tackle complex problems by using statistical techniques

**ABSTRACT**

Can we detect electric discharge states in gases based on the information on visual images? This article proposes a new kind of method where we build several detection models for different states of corona discharge by applying four kinds of machine learning algorithms to extract color, brightness, and shape information characteristics of visible images taken by a digital camera. Every model is then tested on a new set of images to measure its performance. The four different machine learning algorithms are support vector machine (SVM), K-nearest neighbor regression (KNN), single layer perceptron (SLP), and decision tree (DT) algorithms. The prediction results show that the color features perform best among all three types of features and the KNN algorithm performs best among all four algorithms. This article also presents a discussion on how to choose the optimal detection areas of images for better detection performance. Our approach shows consistent results across different cameras and camera settings. The results demonstrate that even if only the visible light spectrum emitted from a plasma is captured, the color method can provide sufficient discharge information for economic and convenient use in discharge state detection because the species producing visible radiation are affected by radiation in all bands.

**SCOPE:**

Visible light communication (VLC) has developed rapidly in recent years. VLC has the advantages of high confidentiality, low cost, etc. It could be an effective way to connect online to offline (O2O). In this paper, an RGB-LED-ID detection and recognition method based on VLC using machine learning is proposed. Different from traditional encoding and decoding VLC, we develop a new VLC system with a form of modulation and recognition. We create different features for different LEDs to make it an Optical Barcode (OBC) based on a Complementary Metal-Oxide-Semiconductor (CMOS) senor and a pulse-width modulation (PWM) method.

**INTRODUCTION**

ELECTRIC discharge is widespread in nature and is commonly used in the operation of industrial equipment. In an effort to understand the properties of electrical discharge, researchers have investigated several physical quantities of discharge, including the voltage, current, optical spectrum, ultrahigh-frequency electromagnetic waves, number of discharges, phase angles (for ac discharge), etc. However, such research works have been carried out without considering the relevant optical characteristics of discharge images even though the initial studies were based on the image, such as that of the original meaning of corona. As a matter of fact, optical measurements are better suited than electromagnetic measurements for the purpose of determining discharge geometries. If the optical characteristics of the discharge image can be incorporated in the diagnosis, the reliability of traditional recognition methods may be increased.

In the past, the traditional research of the discharge image was conducted from a qualitative point of view, such as morphology description, strong or weak light intensity, etc. . However, only a few studies focused on quantitative evaluation can be found. With the development of computer techniques, the digital image processing methods have been applied extensively to study discharge characteristics, such as breakdown paths, discharge area, etc., especially in ultraviolet (UV), which can help tackle complex problems by using statistical techniques or the fractal theory. Although the images obtained by a high-speed camera (nanosecond time scale) can provide some details of a single discharge , the essence of gas discharge remains random under the same macroscopic physical conditions. On the other hand, the discharge used in some industrial applications is a collection of a large number of microdischarges. Therefore, a statistical evaluation of discharge images covering a large number of stochastic processes on a long time scale is still of great significance, compared with the research methods of high-speed cameras.

The color information produced by optical radiation has not been widely used in the study of discharge images. In 2000, Russell and Jones proposed the use of chromatic attributes to directly monitor the stability of plasma states. However, the studies were then limited only to the use of optical-electrical detection techniques, which can only be applied to a relatively large area for achieving a general understanding . In 2009, Koppisetty et al. attempted to establish a correlation of color information of the visual images with the progress of partial-vacuum breakdown. In 2016, Serrano et al. used color information to monitor arc welding. Developments in nonthermal plasmas have stagnated. We conducted research on the color difference in corona and surface discharge and filed for patents on using color information to detect the discharge state. In 2017, Prasad and Reddy introduced a method for extracting color information from discharge images, which was then converted to brightness metrics to study the relationship with discharge power, which is an important progress. To summarize, utilizing color information in the study of spatial distribution of nonthermal plasma discharge is an emerging area of research, due to the recent development of high-resolution digital cameras.

Research in AI/Machine Learning has made great progress in recent years. These techniques have demonstrated a great practical value in solving high-dimensional nonlinear problems and have been widely used in high-voltage fields. For example, in the past, there has been research on applying support vector machine (SVM) and neural networks to recognize patterns of insulator conditions, PRPD, ultrahigh frequency for pattern recognition, and fusion plasmas.

Our research is novel in terms of being the first in using machine-learning methods to analyze the color information on visual images. Besides SVM, we have also explored using K-nearest neighbor regression (KNN), single layer perceptron (SLP), and decision tree (DT) for analysis. These algorithms are widely used for regression tasks that would require supervised learning. Here, we will not go into details of the algorithms, and for a detailed description of the machine learning algorithm, one can refer to many literatures available. This article proposes a method where we build several detection models of different states of corona discharge by applying machine learning algorithms to extract the color, brightness, and shape characteristics of visual images. In the second part, the experimental set of corona discharge is introduced. In the third part, the idea of three primary color- [red, green, and blue (RGB)] gray level histogram (RGB-GLH) of visual images will be introduced and the specific process of applying machine-learning algorithms to analyze the characteristic information of visual images will be discussed. In the fourth part, the prediction results of our model are reported and compared. Finally, a summary will be presented.

The dc microgrid has been a research hotspot with the development of the distributed generation. The dc circuit breaker of the microgrid protection is urgently needed. The new dc hybrid circuit breaker (HCB) which is composed of the vacuum circuit breakers (VCBs) and insulated gate bipolar transistors (IGBTs) in parallel has thus attracted much attention. The dc HCB has become a research hotspot in the world. Genji et al. propose the HCB first, which is applied in the 400 V distribution networks. The HCB based on the parallel switch and gate turn-off thyristor is designed. Polman et al. proposed a bidirectional 600 V/6 kA hybrid dc switch by IGBTs. The design and realization of dc HCB are researched. The commutating circuits for hybrid breakers are simulated by MATLAB. The dc HCB is also used as the automatic transfer switch in ac power supply . ABB researches and develops a new high-voltage hybrid dc circuit breaker, which is composed of mechanical switches and semiconductor device IGBTs in parallel.

Digital Object Identifier interrupting time is less than 5 ms . The 10-kV dc HCB based on integrated gate-commutated thyristor and its breaking current is 3.6 kA . Alstom develops a 120-kV dc HCB based on the thyristors and the fast VCB and other ancillary equipment. The maximum interrupting current is 5.2 kA. Jingqiang et al. investigate the superconducting fault current limiter connecting to the traditional HCB in series, which is used to limit the main circuit current, so as to apply to the high-voltage dc transmission field. Khan et al.research on the application of the superconductor in the protection of the dc HCB by the simulation and test. However, due to the cost and reliability of high-voltage dc HCB, it will limit the application of HCB in the field of HVDC transmission. Therefore, with the advantages of the economic and reliability in the low voltage field, dc HCB is more worthy of study. There has been a lot of research on the vacuum arc commutating characteristic. The characteristic of parallel VCB and capacitors in dc interrupting assisted by the transverse magnetic field (TMF) or the nonuniform magnetic field is investigated, and the voltage and current are 400 V and 500 A, respectively, . The influence of the magnetic field, shunt capacitor on the vacuum arc stability, and the dc breaking capacity is obtained. The critical current which represents the minimum current of dc vacuum arc in stable state is studied by the simulation and experimental research on the characteristics of interaction between the vacuum arc and the magnetic field, and the influence of the circuit branch parameters on the vacuum arc extinguishing. The probability distribution of the current survival at different magnetic field and shunt capacitors is gained in Weibull distribution, which can be used to analyze the success or failure of the arc commutation. In the condition of the current limiter resistor (CLR) (0.5–2 ) and the dc current (below 1000 A), the vacuum arc commutating characteristic is researched, which may be useful to dc fault current limiter . The resistance of the CLR is always in ohms when it is in milliohms at the application for dc HCB Therefore, much research on the high-voltage dc HCB and the vacuum arc commutation in dc interruption, however, little work has been concentrated on the vacuum arc commutation between the vacuum arc and the IGBTs. In this paper, the principle of the dc HCB is discussed and the vacuum arc commutation is simplified.

The test circuit of the vacuum arc commutation is setup. The influence of the magnitude of the main current, the resistance of the IGBTs, and the externally applied TMF on the vacuum arc commutation is investigated. The interaction between the vacuum arc commutation and the externally applied TMF is discussed the first process is investigated. The main current flows through the HSVCB before the contacts separate because the current shunt is depended on the resistance ratio between the contact resistance (micro-ohms) and the resistance of IGBTs (milliohms). The current of the IGBTs increases as the vacuum arc is produced after the contacts separate. In this stage, the arc resistance and the equivalent resistance of the IGBTs are in the same order of magnitude. The arc resistance of the HSVCB is inversely proportional to the magnitude of the main current, which is due to that the arc voltage of vacuum interrupters with axial magnetic field (AMF) is almost constant when the current is below 10 kA. Therefore, the success or failure of the vacuum arc commutation is related to the current value, the ON-state resistance of IGBTs, the arc voltage of HSVCB, and the externally applied TMF. The duration of the vacuum arc commutation is about tens of microseconds when the current is below 200 A as shown in while it will increase to several milliseconds or even more as the main current is in kiloamperes. In addition, the series or parallel connection will also affect the vacuum arc commutation. So it is vital to investigate the vacuum arc commutation characteristic in order to improve and decrease the duration of vacuum arc commutation.

**SYSTEM ANALYSIS**

**REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

**REQUIREMENT SPECIFICATION**

**Functional Requirements**

* Graphical User interface with the User.

**Software Requirements**

For developing the application the following are the Software Requirements:

1. Python
2. Django

**Operating Systems supported**

1. Windows 10 64 bit OS

**Technologies and Languages used to Develop**

1. Python

**Debugger and Emulator**

* Any Browser (Particularly Chrome)

**Hardware Requirements**

For developing the application the following are the Hardware Requirements:

* Processor: Intel i3
* RAM: 4 GB
* Space on Hard Disk: minimum 1 TB

**TECHNOLOGY USED:**

**PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

**Interactive Mode Programming**

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>

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

>>> print "Hello, Python!"

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

Hello, Python!

**Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

print "Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

#!/usr/bin/python

print "Hello, Python!"

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

$ chmod +x test.py # This is to make file executable

$./test.py

This produces the following result −

Hello, Python!

**Python Identifiers**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

**Reserved Words**

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

and exec not

assert finally or

break for pass

class from print

continue global raise

def if return

del import try

elif in while

else is with

except lambda yield

**Lines and Indentation**

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

if True:

print "True"

else:

print "False"

However, the following block generates an error −

if True:

print "Answer"

print "True"

else:

print "Answer"

print "False"

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

#!/usr/bin/python

import sys

try:

# open file stream

file = open(file\_name, "w")

except IOError:

print "There was an error writing to", file\_name

sys.exit()

print "Enter '", file\_finish,

print "' When finished"

while file\_text != file\_finish:

file\_text = raw\_input("Enter text: ")

if file\_text == file\_finish:

# close the file

file.close

break

file.write(file\_text)

file.write("\n")

file.close()

file\_name = raw\_input("Enter filename: ")

if len(file\_name) == 0:

print "Next time please enter something"

sys.exit()

try:

file = open(file\_name, "r")

except IOError:

print "There was an error reading file"

sys.exit()

file\_text = file.read()

file.close()

print file\_text

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

total = item\_one + \

item\_two + \

item\_three

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

days = ['Monday', 'Tuesday', 'Wednesday',

'Thursday', 'Friday']

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

word = 'word'

sentence = "This is a sentence."

paragraph = """This is a paragraph. It is

made up of multiple lines and sentences."""

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

#!/usr/bin/python

# First comment

print "Hello, Python!" # second comment

This produces the following result −

Hello, Python!

You can type a comment on the same line after a statement or expression −

name = "Madisetti" # This is again comment

You can comment multiple lines as follows −

# This is a comment.

# This is a comment, too.

# This is a comment, too.

# I said that already.

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

'''

This is a multiline

comment.

'''

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

#!/usr/bin/python

raw\_input("\n\nPress the enter key to exit.")

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon ( ; ) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

import sys; x = 'foo'; sys.stdout.write(x + '\n')

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon ( : ) and are followed by one or more lines which make up the suite. For example −

if expression :

suite

elif expression :

suite

else :

suite

**Command Line Arguments**

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

$ python -h

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

Options and arguments (and corresponding environment variables):

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

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

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

**Python Lists**

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5 ];

list3 = ["a", "b", "c", "d"]

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";

The empty tuple is written as two parentheses containing nothing −

tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value −

tup1 = (50,);

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0];

print "tup2[1:5]: ", tup2[1:5];

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

tup1[0]: physics

tup2[1:5]: [2, 3, 4, 5]

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']

print "dict['Age']: ", dict['Age']

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

dict['Name']: Zara

dict['Age']: 7

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Alice']: ", dict['Alice']

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

dict['Alice']:

Traceback (most recent call last):

File "test.py", line 4, in <module>

print "dict['Alice']: ", dict['Alice'];

KeyError: 'Alice'

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

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

dict['Age']: 8

dict['School']: DPS School

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name']; # remove entry with key 'Name'

dict.clear(); # remove all entries in dict

del dict ; # delete entire dictionary

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

dict['Age']:

Traceback (most recent call last):

File "test.py", line 8, in <module>

print "dict['Age']: ", dict['Age'];

TypeError: 'type' object is unsubscriptable

Note − del() method is discussed in subsequent section.

**Properties of Dictionary Keys**

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}

print "dict['Name']: ", dict['Name']

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

dict['Name']: Manni

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

#!/usr/bin/python

dict = {['Name']: 'Zara', 'Age': 7}

print "dict['Name']: ", dict['Name']

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

Traceback (most recent call last):

File "test.py", line 3, in <module>

dict = {['Name']: 'Zara', 'Age': 7};

TypeError: unhashable type: 'list'

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

#!/usr/bin/python

tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

# Following action is not valid for tuples

# tup1[0] = 100;

# So let's create a new tuple as follows

tup3 = tup1 + tup2;

print tup3;

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

(12, 34.56, 'abc', 'xyz')

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

#!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000);

print tup;

del tup;

print "After deleting tup : ";

print tup;

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

('physics', 'chemistry', 1997, 2000)

After deleting tup :

Traceback (most recent call last):

File "test.py", line 9, in <module>

print tup;

NameError: name 'tup' is not defined

**DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability)and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models



**Create a Project**

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ django-admin startproject myproject

This will create a "myproject" folder with the following structure −

myproject/

manage.py

myproject/

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements −

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

$ python manage.py help

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

\_\_init\_\_.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': 'database.sql',

'USER': '',

'PASSWORD': '',

'HOST': '',

'PORT': '',

}

}

Database is set in the ‘Database’ dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)

PostGreSQL (django.db.backends.postgresql\_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django\_mongodb\_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME\_ZONE, LANGUAGE\_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

$ python manage.py runserver

You will get something like the following on running the above code −

Validating models...

0 errors found

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

**Create an Application**

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py −

$ python manage.py startapp myapp

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

myapp/

\_\_init\_\_.py

admin.py

models.py

tests.py

views.py

\_\_init\_\_.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED\_APPS tuple in the settings.py file of your project (add your app name) −

INSTALLED\_APPS = (

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'myapp',

)

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

#-\*- coding: utf-8 -\*-

from django import forms

class LoginForm(forms.Form):

user = forms.CharField(max\_length = 100)

password = forms.CharField(widget = forms.PasswordInput())

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

#-\*- coding: utf-8 -\*-

from myapp.forms import LoginForm

def login(request):

username = "not logged in"

if request.method == "POST":

#Get the posted form

MyLoginForm = LoginForm(request.POST)

if MyLoginForm.is\_valid():

username = MyLoginForm.cleaned\_data['username']

else:

MyLoginForm = Loginform()

return render(request, 'loggedin.html', {"username" : username})

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

<html>

<body>

<form name = "form" action = "{% url "myapp.views.login" %}"

method = "POST" >{% csrf\_token %}

<div style = "max-width:470px;">

<center>

<input type = "text" style = "margin-left:20%;"

placeholder = "Identifiant" name = "username" />

</center>

</div>

<br>

<div style = "max-width:470px;">

<center>

<input type = "password" style = "margin-left:20%;"

placeholder = "password" name = "password" />

</center>

</div>

<br>

<div style = "max-width:470px;">

<center>

<button style = "border:0px; background-color:#4285F4; margin-top:8%;

height:35px; width:80%;margin-left:19%;" type = "submit"

value = "Login" >

<strong>Login</strong>

</button>

</center>

</div>

</form>

</body>

</html>

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

{% csrf\_token %}

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

<html>

<body>

You are : <strong>{{username}}</strong>

</body>

</html>

Now, we just need our pair of URLs to get started: myapp/urls.py

from django.conf.urls import patterns, url

from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

url(r'^connection/',TemplateView.as\_view(template\_name = 'login.html')),

url(r'^login/', 'login', name = 'login'))

When accessing "/myapp/connection", we will get the following login.html template rendered −

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE\_CLASSES and the INSTALLED\_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE\_CLASSES should have −

'django.contrib.sessions.middleware.SessionMiddleware'

And INSTALLED\_APPS should have −

'django.contrib.sessions'

By default, Django saves session information in database (django\_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won’t see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side −

def login(request):

username = 'not logged in'

if request.method == 'POST':

MyLoginForm = LoginForm(request.POST)

if MyLoginForm.is\_valid():

username = MyLoginForm.cleaned\_data['username']

request.session['username'] = username

else:

MyLoginForm = LoginForm()

return render(request, 'loggedin.html', {"username" : username}

Then let us create formView view for the login form, where we won’t display the form if cookie is set −

def formView(request):

if request.session.has\_key('username'):

username = request.session['username']

return render(request, 'loggedin.html', {"username" : username})

else:

return render(request, 'login.html', {})

Now let us change the url.py file to change the url so it pairs with our new view −

from django.conf.urls import patterns, url

from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

url(r'^connection/','formView', name = 'loginform'),

url(r'^login/', 'login', name = 'login'))

When accessing /myapp/connection, you will get to see the following page

**LITERATURE SURVEY**

## 1) An Incremental Analysis Of Spark Paths In Air Using 3-Dimensional Image Processing

**AUTHORS: D. H. Qiu, J. M. K. MacAlpine, and Z. Y. Li**

# A video camera has been used to record the path of spark breakdowns across a positive point/plane air-gap. By using a prism as a mirror, two images were obtained from which the path could be determined in three dimensions by subsequent computer processing and analysis. Images were analyzed for each of five angular positions of the point electrode's axis, the gap remaining constant. The fractal dimension of each path was determined and found to increase slightly with the inclination of the pointed electrode. Each spark appeared to have propagated semi-randomly in a direction which included a 'memory' factor, that is, it tended to follow the previous direction while moving monotonically towards the opposite electrode. This was confirmed by an analysis of 400 spark paths which showed that each section has a specific relationship to the previous section, but that there is a wide variation around this mean which leads to the characteristic shape of a spark path. Simulations based on these results yield spark tracks which look similar to those measured and have similar fractal dimensions. These analyses strongly suggest that the development of spark paths in air is by successive steps and in a direction suggested by an angular probability distribution which is related to the field at the tip of the propagating leader.

# 2) Fast Imaging Of Intermittent Electrospraying Of Water With Positive Corona Discharge

**AUTHORS:**  **B. Pongrác, H. H. Kim, M. Janda, V. Martišovitš, and Z. Machal**

The effect of the electrospraying of water in combination with a positive direct current (dc) streamer corona discharge generated in air was investigated in this paper. We employed high-speed camera visualizations and oscilloscopic discharge current measurements in combination with an intensified charge-coupled device camera for fast time-resolved imaging. The repetitive process of Taylor cone formation and droplet formation from the mass fragments of water during the electrospray was visualized. Depending on the applied voltage, the following intermittent modes of electrospraying typical for water were observed: dripping mode, spindle mode, and oscillating-spindle mode. The observed electrospraying modes were repetitive with a frequency of a few hundreds of Hz, as measured from the fast image sequences. This frequency agreed well with the frequency of the measured streamer current pulses. The presence of filamentary streamer discharges at relatively low voltages probably prevented the establishment of a continuous electrospray in the cone–jet mode. After each streamer, a positive glow corona discharge was established on the water filament tip, and it propagated from the stressed electrode along with the water filament elongation. The results show a reciprocal character of intermittent electrospraying of water, and the presence of corona discharge, where both the electrospray and the discharge affect each other. The generation of a corona discharge from the water cone depended on the repetitive process of the cone formation. Also, the propagation and curvature of the water filament were influenced by the discharge and its resultant space charge. Furthermore, these phenomena were partially influenced by the water conductivity.

# 3) Statistical evaluation of AC corona images in long-time scale and characterization of short-gap leader

**AUTHORS** **: X. Li et al**

Although the image of discharge in the nanosecond time scale can provide some details of a single discharge, the essence of gas discharge remains random under the same macroscopic physical conditions. Therefore, the statistical evaluation of discharge images including a large number of stochastic processes in a long-time scale is still of great significance. In this paper, a digital image processing method presented in our previously paper is used to research the statistic indicators of AC corona discharge image in the time scale of seconds, and the axial distribution of the average gray level and the gray level standard deviation about corona discharge image are determined. Then, these statistical indicators are utilized to study the long brush-like corona, and a clear "stem" caused by the point electrode and not by the ball head electrode was found, even if they all belong to the highly non uniform electric field. Considering its corresponding current pulse rise time, we believe that the leader discharge also exists in the cm-level short gap. These results indicate that the statistical analysis on the longtime scales can be used in discharge research, and further image information mining will likely be used to provide some new characteristic parameters.

# 4) Colorimetric analysis for on-line arc-welding diagnostics by means of plasma optical spectroscopy

**AUTHORS** : **Jesus Mirapeix Serrano, Ruben Ruiz Lombera , Jose J. Valdiande , Jose Miguel Lopez-Higuera**

In this paper an analysis on the suitability of employing a colorimetric analysis of the acquired plasma spectra to perform online arc-welding quality monitoring will be discussed. Different colorimetric parameters like the color temperature will be evaluated in comparison to the standard approach based on the estimation of the plasma electronic temperature. In particular, experimental tests performed with a TIG arc-welding process will show that some colorimetric parameters exhibit a better response in terms of detection and correlation to some defects than the traditional spectroscopic approach.

**5)** **Digital Image Processing Techniques For Estimating Power Released From The Corona Discharges**

**AUTHORS**: **D. S. Prasad and B. S. Reddy**

Digital image processing is being widely used in all fields of science and engineering, like the detection of cancer cells, identification of the soil properties etc. Application of image processing to high voltage phenomenon has not been well explored. The present work attempts to identify the possible application of image processing techniques to obtain the image parameters which can be used as indicators to estimate the corona power. A color thresholding based image segmentation method is employed on the digital images of corona and thereby effective spread of corona plasma has been extracted. The variations in the parameters: average intensity, area of the spread, energy, entropy and luminosity content in the processed images are studied in relation with the measured power. It is observed that the parameters extracted from the corona images share a unique relation with the measured corona power, and can be used for the prediction analysis.

**EXISTING SYSTEM:**

In the existing system the traditional research of the discharge image was conducted from a qualitative point of view, such as morphology description, strong or weak light intensity, etc.. However, only a few studies focused on quantitative evaluation can be found. With the development of computer techniques, the digital image processing methods have been applied extensively to study discharge characteristics, such as breakdown paths, discharge area, etc., especially in ultraviolet (UV), which can help tackle complex problems by using statistical techniques or the fractal theory. Although the images obtained by a high-speed camera (nanosecond time scale) can provide some details of a single discharge, the essence of gas discharge remains random under the same macroscopic physical conditions.

**DISADVANTAGES OF EXISTING SYSTEM:**

* Limited only for optical-electrical detection techniques.
* Color information of the visual images with the progress of partial-vacuum breakdown.
* Research works have been carried out without considering the relevant optical characteristics of discharge images.

**Algorithm**: Electrode Geometry, Dielectric Barrier Discharges

**PROPOSED SYSTEM:**

The proposed system proposes a method where we build several detection models of different states of corona discharge by applying machine learning algorithms to extract the color, brightness, and shape characteristics of visual images. In the second part, the experimental set of corona discharge is introduced. In the third part, the idea of three primary color-[red, green, and blue (RGB)] gray level histogram (RGB-GLH) of visual images will be introduced and the specific process of applying machine-learning algorithms to analyze the characteristic information of visual images will be discussed. In the fourth part, the prediction results of our model are reported and compared.

**ADVANTAGES OF PROPOSED SYSTEM:**

* We prepare our image library taken by a digital camera at first, then select the feature quantities, such as color, brightness, and shape information characteristics of visible images.
* Each image is composed of a large number of pixels, and in an effort to extract sufficient amount of meaningful color information.
* We also explored using the GLH of black-and-white image as features. Using gray level features means that we only retain and consider the information on brightness.

**Algorithm**: Support Vector Machine (SVM), K-nearest Neighbor Regression (KNN), Single Layer Perceptron (SLP), and Decision Tree (DT).

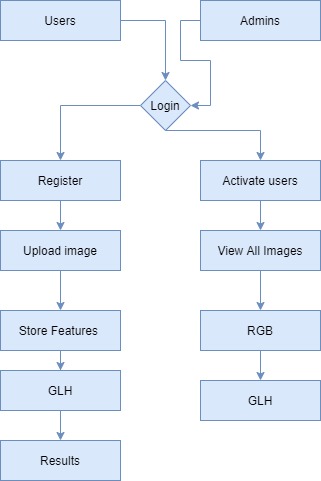
**SYSTEM DESIGN**

**SYSTEM ARCHITECTURE:**

****

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. 

**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**METHODOLOGY:**

**MODULES:**

* **User**
* **Admin**
* **Data Preprocess**
* **Machine Learning**

**MODULES DESCRIPTION:**

**User:**

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the customer. Once admin activated the customer then user can login into our system. User can upload corona discharge images by browser. in the server side we need to execute two times. First we can enable Matplotlib.use(TkAgg) then the tkinter graph will be displayed. The graph are grayscale, rgb, glh values are curved in the graph. The user can test the color features as input for all images and can find the MeanRmse, same we can test Shape and brightness features. Second onwards we can disble the Matplotlib.use(Agg) the next the graph will not displayed and smooth execution possible.

**Admin:**

Admin can login with his credentials. Once he login he can activate the users. The activated user only login in our applications. The admin can view the all user uploaded images. By clicking any image he can get the Graph of RGB values, Histograms graph. The image resining and coverted into gray scale also displayed to the admin side.

**Data Preprocess:**

We need to prepare our image library taken by a digital camera at first are upload the images by code folder which is taken from github to test the code, then select the feature quantities, such as color, brightness, and shape information characteristics of visible images, and finally establish several detection models for different states of corona discharge. We looked at shape information of the black-and-white image and experimented with related features by using histogram-oriented gradients (HOG). HOG compute histograms for gradients within each pixel block to determine the local orientation/shape. In our case, we applied HOG algorithms with (orientation bin size, pixels per cell, cells per block) = (8, 8, 4) on our image of 150 × 950 pixels. We ended up with a matrix of (number of blocks per row, number of blocks per columns, number of cells per row, number of cells per column, number of orientations) = (115, 15, 4, 4, 8) for each image. This is then flattened and used as a feature vector.

**Machine Learning:**

We build several detection models for different states of corona discharge by applying four types of machine learning algorithms to extract the information characteristics of visible images. The four types of machine-learning algorithms are SVM, KNN, SLP, and DT algorithms. SVM is a generalized linear classifier for binary classification of data using supervised learning and kernel methods. It can be used to classify data nonlinearly, and is one of the common kernel-learning methods. The KNN classification algorithm is one of the simplest methods in data mining classification technology. The core idea of KNN is that if the majority of the k-most adjacent samples of a certain sample X in the feature space belong to a certain category, then sample X also belongs to this category and is assigned to the characteristics of the samples in this category. SLP is a type of simple, one-layer, feed-forward artificial neural network. DT is a method to approximate the value of the discrete function. After data are processed, readable rules in the form of a DT are generated by an induction algorithm. Then, when a new data point comes in, it is classified by following the tree structure from top to bottom. All of our algorithms would output one single class per image for prediction.

**ALGORITHMS USED**

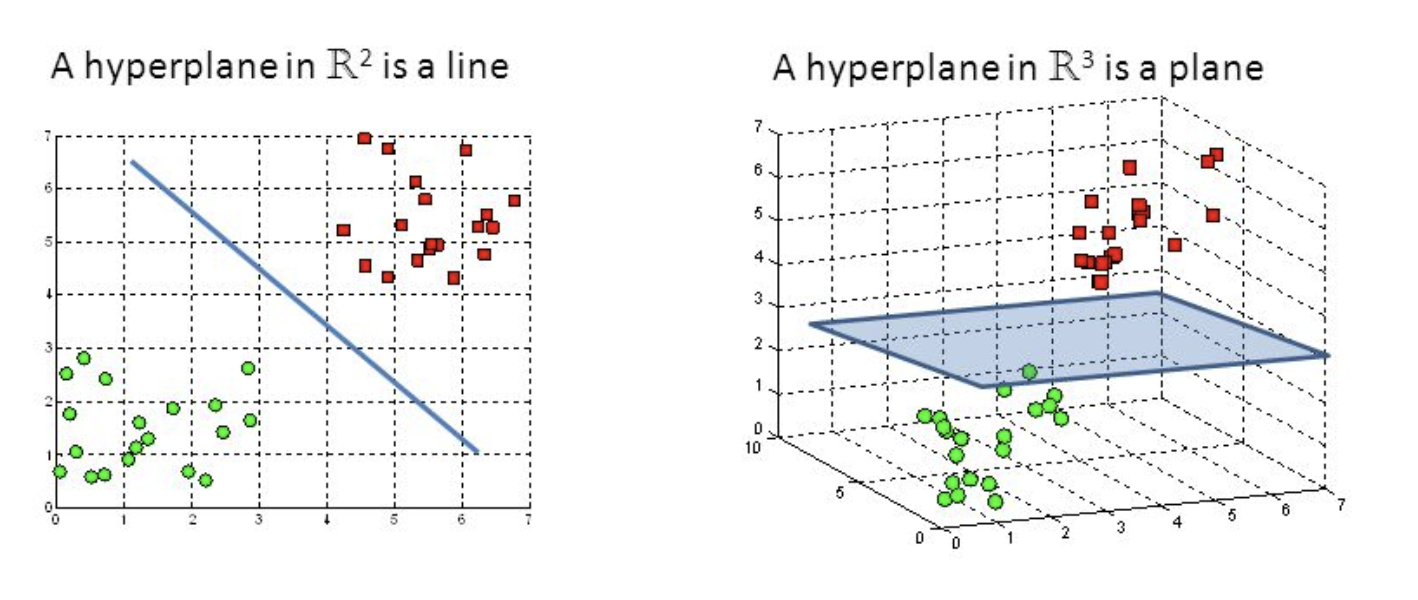
**SUPPORT VECTOR MACHINE?**

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

Possible hyperplanes

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes and Support Vectors



Hyperplanes in 2D and 3D feature space

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



Support Vectors

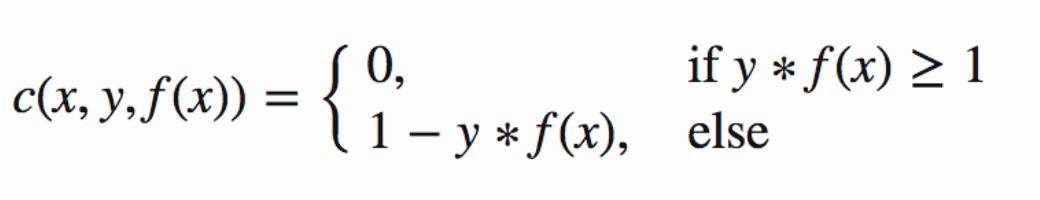
Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

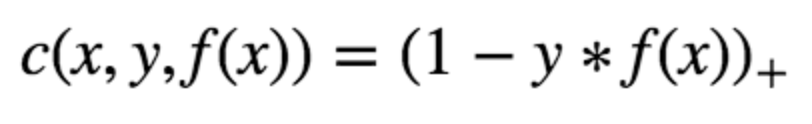
Large Margin Intuition

In logistic regression, we take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function. If the squashed value is greater than a threshold value(0.5) we assign it a label 1, else we assign it a label 0. In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin.

Cost Function and Gradient Updates

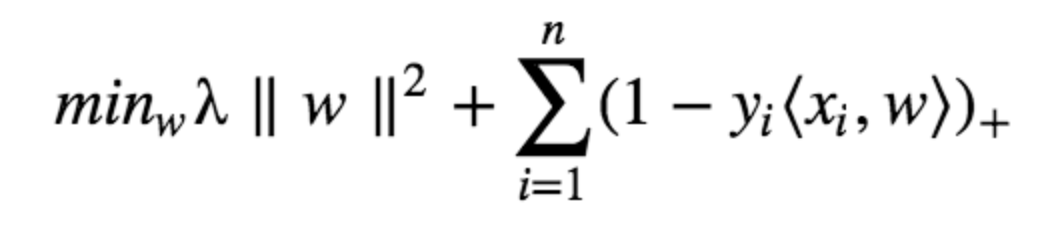
In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.





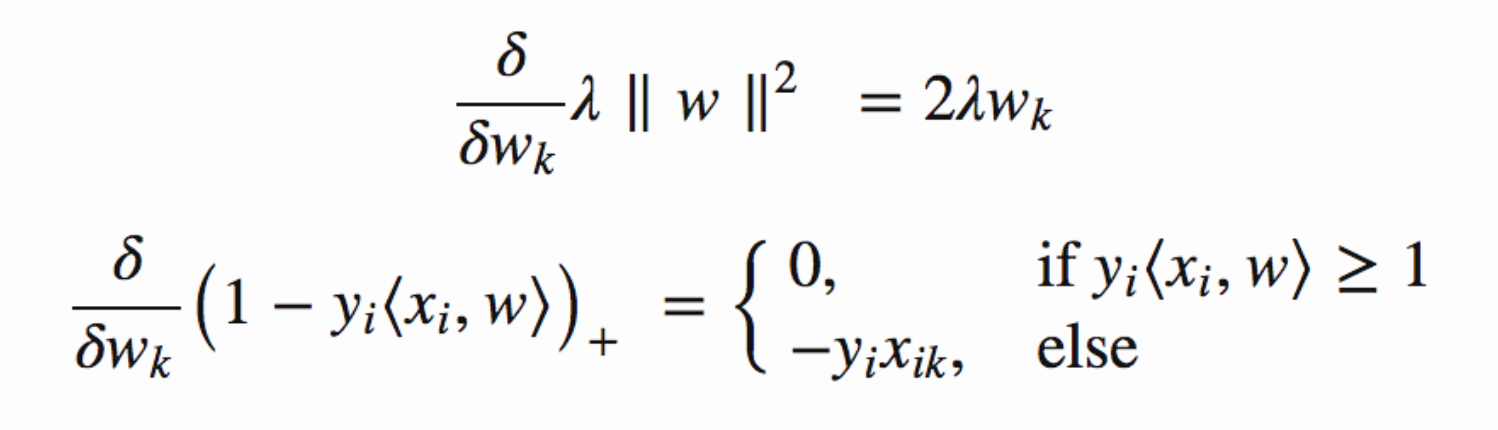
Hinge loss function (function on left can be represented as a function on the right)

The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions looks as below.



Loss function for SVM

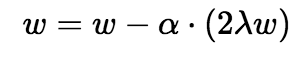
Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.



Gradients

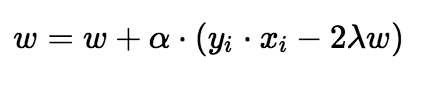
When there is no misclassification, i.e our model correctly predicts the class of our data point, we only have to update the gradient from the regularization parameter.





Gradient Update — No misclassification

When there is a misclassification, i.e our model make a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform gradient update.



Gradient Update — Misclassification

# **K-Nearest Neighbor(KNN) Algorithm for Machine Learning**

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



## Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



## How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
* **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



* Firstly, we will choose the number of neighbors, so we will choose the k=5.
* Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



* By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

## How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* Large values for K are good, but it may find some difficulties.

## Advantages of KNN Algorithm:

* It is simple to implement.
* It is robust to the noisy training data
* It can be more effective if the training data is large.

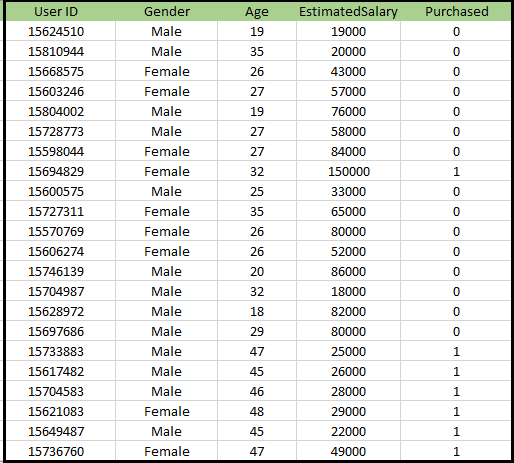
## Disadvantages of KNN Algorithm:

* Always needs to determine the value of K which may be complex some time.
* The computation cost is high because of calculating the distance between the data points for all the training samples.

## Python implementation of the KNN algorithm

To do the Python implementation of the K-NN algorithm, we will use the same problem and dataset which we have used in Logistic Regression. But here we will improve the performance of the model. Below is the problem description:

**Problem for K-NN Algorithm:** There is a Car manufacturer company that has manufactured a new SUV car. The company wants to give the ads to the users who are interested in buying that SUV. So for this problem, we have a dataset that contains multiple user's information through the social network. The dataset contains lots of information but the **Estimated Salary** and **Age** we will consider for the independent variable and the **Purchased variable** is for the dependent variable. Below is the dataset:



**Steps to implement the K-NN algorithm:**

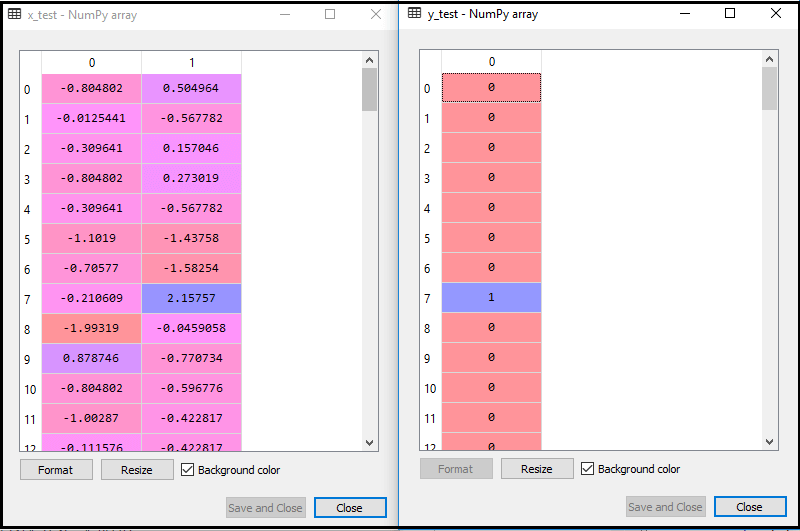
* Data Pre-processing step
* Fitting the K-NN algorithm to the Training set
* Predicting the test result
* Test accuracy of the result(Creation of Confusion matrix)
* Visualizing the test set result.

**Data Pre-Processing Step:**

The Data Pre-processing step will remain exactly the same as Logistic Regression. Below is the code for it:

1. # importing libraries
2. **import** numpy as nm
3. **import** matplotlib.pyplot as mtp
4. **import** pandas as pd
6. #importing datasets
7. data\_set= pd.read\_csv('user\_data.csv')
9. #Extracting Independent and dependent Variable
10. x= data\_set.iloc[:, [2,3]].values
11. y= data\_set.iloc[:, 4].values
13. # Splitting the dataset into training and test set.
14. from sklearn.model\_selection **import** train\_test\_split
15. x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.25, random\_state=0)
17. #feature Scaling
18. from sklearn.preprocessing **import** StandardScaler
19. st\_x= StandardScaler()
20. x\_train= st\_x.fit\_transform(x\_train)
21. x\_test= st\_x.transform(x\_test)

By executing the above code, our dataset is imported to our program and well pre-processed. After feature scaling our test dataset will look like:



From the above output image, we can see that our data is successfully scaled.

* **Fitting K-NN classifier to the Training data:**  
  Now we will fit the K-NN classifier to the training data. To do this we will import the **KNeighborsClassifier** class of **Sklearn Neighbors** library. After importing the class, we will create the **Classifier** object of the class. The Parameter of this class will be
  + **n\_neighbors:** To define the required neighbors of the algorithm. Usually, it takes 5.
  + **metric='minkowski':** This is the default parameter and it decides the distance between the points.
  + **p=2:** It is equivalent to the standard Euclidean metric.

And then we will fit the classifier to the training data. Below is the code for it:

1. #Fitting K-NN classifier to the training set
2. from sklearn.neighbors **import** KNeighborsClassifier
3. classifier= KNeighborsClassifier(n\_neighbors=5, metric='minkowski', p=2 )
4. classifier.fit(x\_train, y\_train)

**Output: By executing the above code, we will get the output as:**

Out[10]:

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2,

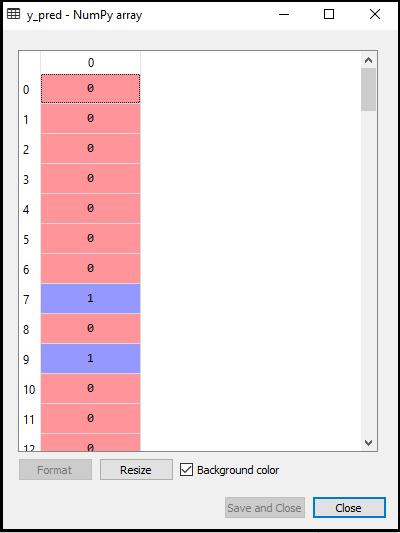
weights='uniform')

* **Predicting the Test Result:** To predict the test set result, we will create a **y\_pred** vector as we did in Logistic Regression. Below is the code for it:

1. #Predicting the test set result
2. y\_pred= classifier.predict(x\_test)

**Output:**

The output for the above code will be:

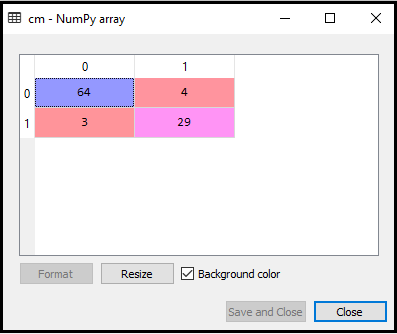


* **Creating the Confusion Matrix:**  
  Now we will create the Confusion Matrix for our K-NN model to see the accuracy of the classifier. Below is the code for it:

1. #Creating the Confusion matrix
2. from sklearn.metrics **import** confusion\_matrix
3. cm= confusion\_matrix(y\_test, y\_pred)

In above code, we have imported the confusion\_matrix function and called it using the variable cm.

**Output:** By executing the above code, we will get the matrix as below:



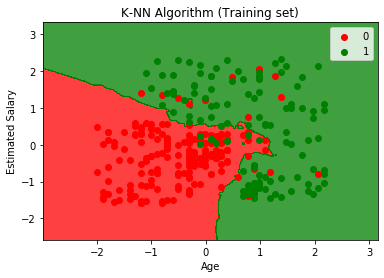
In the above image, we can see there are 64+29= 93 correct predictions and 3+4= 7 incorrect predictions, whereas, in Logistic Regression, there were 11 incorrect predictions. So we can say that the performance of the model is improved by using the K-NN algorithm.

* **Visualizing the Training set result:**  
  Now, we will visualize the training set result for K-NN model. The code will remain same as we did in Logistic Regression, except the name of the graph. Below is the code for it:

1. #Visulaizing the trianing set result
2. from matplotlib.colors **import** ListedColormap
3. x\_set, y\_set = x\_train, y\_train
4. x1, x2 = nm.meshgrid(nm.arange(start = x\_set[:, 0].min() - 1, stop = x\_set[:, 0].max() + 1, step  =0.01),
5. nm.arange(start = x\_set[:, 1].min() - 1, stop = x\_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green' )))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. **for** i, j in enumerate(nm.unique(y\_set)):
11. mtp.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1],
12. c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('K-NN Algorithm (Training set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

**Output:**

**By executing the above code, we will get the below graph:**

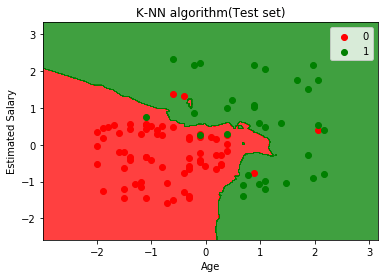


The output graph is different from the graph which we have occurred in Logistic Regression. It can be understood in the below points:

* + As we can see the graph is showing the red point and green points. The green points are for Purchased(1) and Red Points for not Purchased(0) variable.
  + The graph is showing an irregular boundary instead of showing any straight line or any curve because it is a K-NN algorithm, i.e., finding the nearest neighbor.
  + The graph has classified users in the correct categories as most of the users who didn't buy the SUV are in the red region and users who bought the SUV are in the green region.
  + The graph is showing good result but still, there are some green points in the red region and red points in the green region. But this is no big issue as by doing this model is prevented from overfitting issues.
  + Hence our model is well trained.
* **Visualizing the Test set result:**  
  After the training of the model, we will now test the result by putting a new dataset, i.e., Test dataset. Code remains the same except some minor changes: such as **x\_train and y\_train** will be replaced by **x\_test and y\_test**.  
  Below is the code for it:

1. #Visualizing the test set result
2. from matplotlib.colors **import** ListedColormap
3. x\_set, y\_set = x\_test, y\_test
4. x1, x2 = nm.meshgrid(nm.arange(start = x\_set[:, 0].min() - 1, stop = x\_set[:, 0].max() + 1, step  =0.01),
5. nm.arange(start = x\_set[:, 1].min() - 1, stop = x\_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green' )))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. **for** i, j in enumerate(nm.unique(y\_set)):
11. mtp.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1],
12. c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('K-NN algorithm(Test set)')
14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

**Output:**



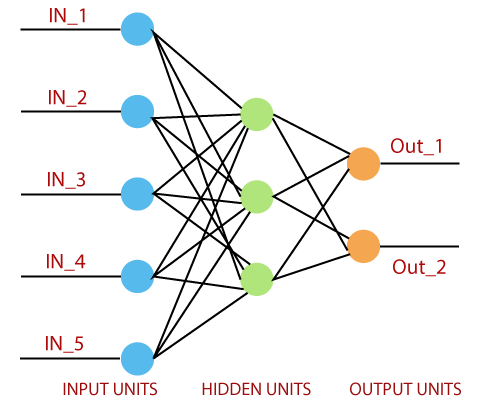
The above graph is showing the output for the test data set. As we can see in the graph, the predicted output is well good as most of the red points are in the red region and most of the green points are in the green region.

However, there are few green points in the red region and a few red points in the green region. So these are the incorrect observations that we have observed in the confusion matrix(7 Incorrect output).

# **Single Layer Perceptron in TensorFlow**

The perceptron is a single processing unit of any neural network. **Frank Rosenblatt** first proposed in **1958** is a simple neuron which is used to classify its input into one or two categories. Perceptron is a linear classifier, and is used in supervised learning. It helps to organize the given input data.

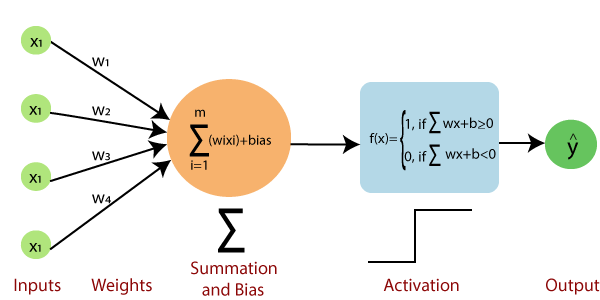
A perceptron is a neural network unit that does a precise computation to detect features in the input data. Perceptron is mainly used to classify the data into two parts. Therefore, it is also known as **Linear Binary Classifier**.



Perceptron uses the step function that returns +1 if the weighted sum of its input 0 and -1.

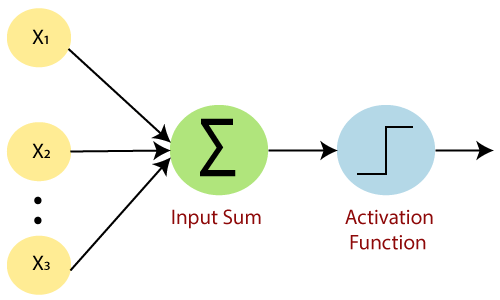
The activation function is used to map the input between the required value like (0, 1) or (-1, 1).

A regular neural network looks like this:

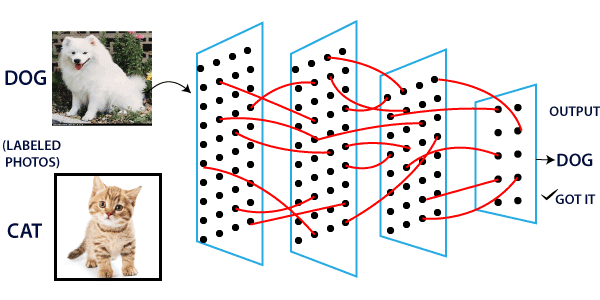


### **The perceptron consists of 4 parts.**

* **Input value or One input layer:** The input layer of the perceptron is made of artificial input neurons and takes the initial data into the system for further processing.
* **Weights and Bias:**  
  **Weight:** It represents the dimension or strength of the connection between units. If the weight to node 1 to node 2 has a higher quantity, then neuron 1 has a more considerable influence on the neuron.  
  **Bias:** It is the same as the intercept added in a linear equation. It is an additional parameter which task is to modify the output along with the weighted sum of the input to the other neuron.
* **Net sum:** It calculates the total sum.
* **Activation Function:** A neuron can be activated or not, is determined by an activation function. The activation function calculates a weighted sum and further adding bias with it to give the result.



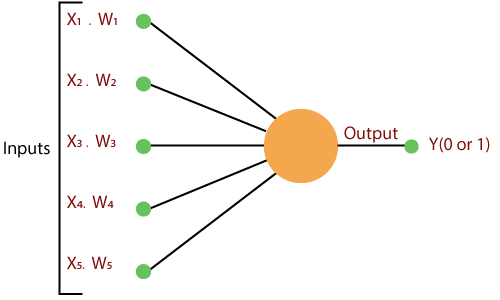
A standard neural network looks like the below diagram.



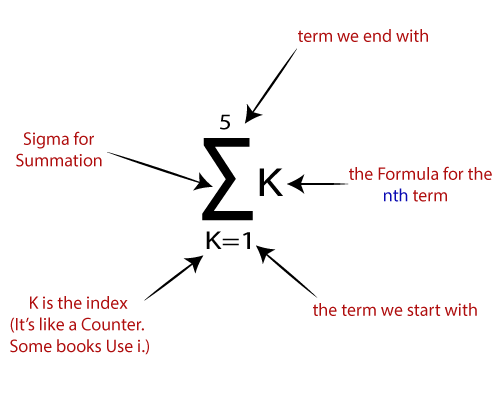
### **How does it work?**

The perceptron works on these simple steps which are given below:

**a.** In the first step, all the inputs x are multiplied with their weights **w**.



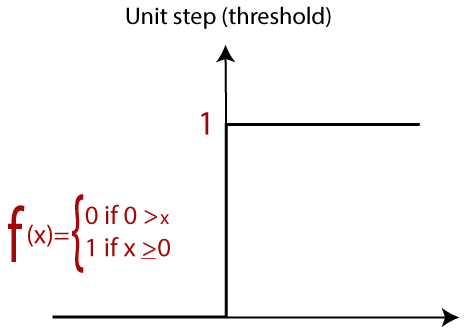
**b.** In this step, add all the increased values and call them the **Weighted sum**.



**c.** In our last step, apply the weighted sum to a correct **Activation Function**.

**For Example:**

A Unit Step Activation Function



There are two types of architecture. These types focus on the functionality of artificial neural networks as follows-

* Single Layer Perceptron
* Multi-Layer Perceptron

## Single Layer Perceptron

The single-layer perceptron was the first neural network model, proposed in 1958 by Frank Rosenbluth. It is one of the earliest models for learning. Our goal is to find a linear decision function measured by the weight vector w and the bias parameter b.

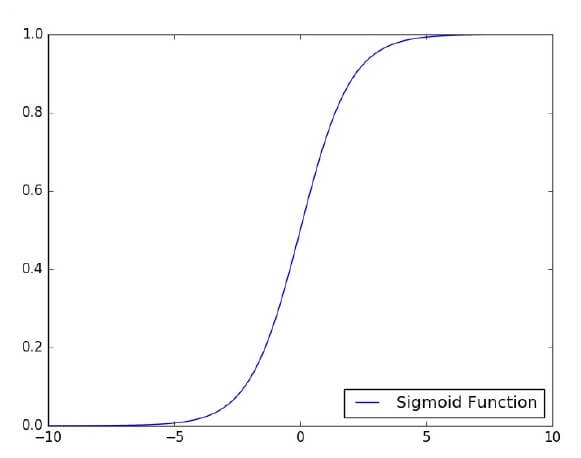
To understand the perceptron layer, it is necessary to comprehend artificial neural networks (ANNs).

The artificial neural network (ANN) is an information processing system, whose mechanism is inspired by the functionality of biological neural circuits. An artificial neural network consists of several processing units that are interconnected.

This is the first proposal when the neural model is built. The content of the neuron's local memory contains a vector of weight.

The single vector perceptron is calculated by calculating the sum of the input vector multiplied by the corresponding element of the vector, with each increasing the amount of the corresponding component of the vector by weight. The value that is displayed in the output is the input of an activation function.

Let us focus on the implementation of a single-layer perceptron for an image classification problem using TensorFlow. The best example of drawing a single-layer perceptron is through the representation of "**logistic regression**."



Now, We have to do the following necessary steps of training logistic regression-

* The weights are initialized with the random values at the origination of each training.
* For each element of the training set, the error is calculated with the difference between the desired output and the actual output. The calculated error is used to adjust the weight.
* The process is repeated until the fault made on the entire training set is less than the specified limit until the maximum number of iterations has been reached

# **Decision Tree Classification Algorithm**

* Decision Tree is a **Supervised learning technique**that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where**internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
* In a Decision tree, there are two nodes, which are the **Decision Node** and**Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
* The decisions or the test are performed on the basis of features of the given dataset.
* **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.**
* It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
* In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
* A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
* Below diagram explains the general structure of a decision tree:

#### **Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.**



## Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

* Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
* The logic behind the decision tree can be easily understood because it shows a tree-like structure.

## Decision Tree Terminologies

 **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

 **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

 **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

 **Branch/Sub Tree:** A tree formed by splitting the tree.

 **Pruning:** Pruning is the process of removing the unwanted branches from the tree.

 **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

**How does the Decision Tree algorithm Work?**

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

* **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
* **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
* **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
* **Step-4:** Generate the decision tree node, which contains the best attribute.
* **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

**Example:** Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



## Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

* **Information Gain**
* **Gini Index**

### **1. Information Gain:**

* Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
* It calculates how much information a feature provides us about a class.
* According to the value of information gain, we split the node and build the decision tree.
* A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) \*Entropy(each feature)

**Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

**Where,**

* **S= Total number of samples**
* **P(yes)= probability of yes**
* **P(no)= probability of no**

### **2. Gini Index:**

* Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
* An attribute with the low Gini index should be preferred as compared to the high Gini index.
* It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
* Gini index can be calculated using the below formula:

Gini Index= 1- ∑jPj2

## Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning**technology used:

* **Cost Complexity Pruning**
* **Reduced Error Pruning.**

## Advantages of the Decision Tree

* It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
* It can be very useful for solving decision-related problems.
* It helps to think about all the possible outcomes for a problem.
* There is less requirement of data cleaning compared to other algorithms.

## Disadvantages of the Decision Tree

* The decision tree contains lots of layers, which makes it complex.
* It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
* For more class labels, the computational complexity of the decision tree may increase.

## Python Implementation of Decision Tree

Now we will implement the Decision tree using Python. For this, we will use the dataset "**user\_data.csv**," which we have used in previous classification models. By using the same dataset, we can compare the Decision tree classifier with other classification models such as [KNN](https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning) [SVM,](https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm) [LogisticRegression,](https://www.javatpoint.com/logistic-regression-in-machine-learning) etc.

Steps will also remain the same, which are given below:

* **Data Pre-processing step**
* **Fitting a Decision-Tree algorithm to the Training set**
* **Predicting the test result**
* **Test accuracy of the result(Creation of Confusion matrix)**
* **Visualizing the test set result.**

**SOURCECODE**

**SOURCE CODE**

User Side views.py

**from** django.shortcuts **import** render, HttpResponse  
**from** .forms **import** UserRegistrationForm  
**from** .models **import** UserRegistrationModel, CoronaDischargeModel  
**from** django.contrib **import** messages  
**from** django.core.files.storage **import** FileSystemStorage  
**from** .rgbglhcodes.StartImagePreprocessing **import** StartProcess  
**from** .rgbglhcodes.SVMCode **import** UserSVMCode  
**from** .rgbglhcodes.ShapeUtility **import** UserImageShape  
**from** .rgbglhcodes.UserBrightness **import** UserImageBrightness  
**from** django\_pandas.io **import** read\_frame  
*# Create your views here.***import** matplotlib  
*#matplotlib.use("Agg")***from** matplotlib **import** style  
*#style.use("ggplot")  
  
  
# Create your views here.***def** UserRegisterActions(request):  
 **if** request.method == **'POST'**:  
 form = UserRegistrationForm(request.POST)  
 **if** form.is\_valid():  
 print(**'Data is Valid'**)  
 form.save()  
 messages.success(request, **'You have been successfully registered'**)  
 form = UserRegistrationForm()  
 **return** render(request, **'UserRegistrations.html'**, {**'form'**: form})  
 **else**:  
 messages.success(request, **'Email or Mobile Already Existed'**)  
 print(**"Invalid form"**)  
 **else**:  
 form = UserRegistrationForm()  
 **return** render(request, **'UserRegistrations.html'**, {**'form'**: form})  
**def** UserLoginCheck(request):  
 **if** request.method == **"POST"**:  
 loginid = request.POST.get(**'loginname'**)  
 pswd = request.POST.get(**'pswd'**)  
 print(**"Login ID = "**, loginid, **' Password = '**, pswd)  
 **try**:  
 check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)  
 status = check.status  
 print(**'Status is = '**, status)  
 **if** status == **"activated"**:  
 request.session[**'id'**] = check.id  
 request.session[**'loggeduser'**] = check.name  
 request.session[**'loginid'**] = loginid  
 request.session[**'email'**] = check.email  
 print(**"User id At"**, check.id, status)  
 **return** render(request, **'users/UserHome.html'**, {})  
 **else**:  
 messages.success(request, **'Your Account Not at activated'**)  
 **return** render(request, **'UserLogin.html'**)  
 **except** Exception **as** e:  
 print(**'Exception is '**, str(e))  
 **pass** messages.success(request, **'Invalid Login id and password'**)  
 **return** render(request, **'UserLogin.html'**, {})  
**def** UserHome(request):  
 **return** render(request, **'users/UserHome.html'**, {})  
  
**def** UploadImageForm(request):  
 loginid = request.session[**'loginid'**]  
 data = CoronaDischargeModel.objects.filter(loginid=loginid)  
 **return** render(request, **'users/UserImageUploadForm.html'**, {**'data'**: data})  
  
**def** UploadImageAction(request):  
 image\_file = request.FILES[**'file'**]  
 *# let's check if it is a csv file* **if not** image\_file.name.endswith(**'.jpg'**):  
 messages.error(request, **'THIS IS NOT A JPG FILE'**)  
 fs = FileSystemStorage(location=**"media/datasets/"**)  
 filename = fs.save(image\_file.name, image\_file)  
 *# detect\_filename = fs.save(image\_file.name, image\_file)* uploaded\_file\_url = **"/media/datasets/"**+filename *#fs.url(filename)* print(**"Image path "**,uploaded\_file\_url)  
 username = request.session[**'loggeduser'**]  
 loginid = request.session[**'loginid'**]  
 email = request.session[**'email'**]  
 obj = StartProcess()  
 colorinfo,picbrightness,picshape = obj.process(filename)  
 colorinfo = colorinfo.tolist()  
 redColor = colorinfo[0]  
 greenColor = colorinfo[1]  
 blueColor = colorinfo[2]  
 picHeight = picshape[0]  
 picWidht = picshape[1]  
 blockofPixel = picshape[2]  
 picbrightness = picbrightness  
  
 CoronaDischargeModel.objects.create(username=username,email=email,loginid=loginid,filename=filename,file=uploaded\_file\_url,redColor=redColor, greenColor=greenColor, blueColor=blueColor, picHeight=picHeight,picWidht=picWidht, blockofPixel=blockofPixel, picbrightness=picbrightness)  
 data = CoronaDischargeModel.objects.filter(loginid=loginid)  
 **return** render(request, **'users/UserImageUploadForm.html'**, {**'data'**:data})  
  
**def** UserSVMTest(request):  
 obj = UserSVMCode()  
 svmMrmse = obj.startSvm()  
 knnmrmse = obj.startKnn()  
 dtmrmse = obj.startDecisionTree()  
 slpmrmse = obj.startSLP()  
 **return** render(request,**'users/UserColors.html'**,{**'svmMrmse'**:svmMrmse,**'knnmrmse'**:knnmrmse,**"dtmrmse"**:dtmrmse,**"slpmrmse"**:slpmrmse})  
  
**def** UserShapeTest(request):  
 data = CoronaDischargeModel.objects.all()  
 df = read\_frame(data)  
 df = df[[**'picHeight'**,**'picWidht'**,**'blockofPixel'**,**'picbrightness'**]]  
 *#print(df.head())* obj = UserImageShape()  
 svmMrmse = obj.startSvm(df)  
 knnmrmse = obj.startKnn(df)  
 dtmrmse = obj.startDecisionTree(df)  
 slpmrmse = obj.startSLP(df)  
 **return** render(request, **'users/UserShapeAsFeatures.html'**,  
 {**'svmMrmse'**: svmMrmse, **'knnmrmse'**: knnmrmse, **"dtmrmse"**: dtmrmse, **"slpmrmse"**: slpmrmse})  
  
  
**def** UserBrightness(request):  
 data = CoronaDischargeModel.objects.all()  
 df = read\_frame(data)  
 df = df[[**'redColor'**, **'greenColor'**, **'blueColor'**, **'picbrightness'**]]  
 obj = UserImageBrightness()  
 svmMrmse = obj.startSvm(df)  
 knnmrmse = obj.startKnn(df)  
 dtmrmse = obj.startDecisionTree(df)  
 slpmrmse = obj.startSLP(df)  
 **return** render(request, **'users/UserBrightnessFeatures.html'**,  
 {**'svmMrmse'**: svmMrmse, **'knnmrmse'**: knnmrmse, **"dtmrmse"**: dtmrmse, **"slpmrmse"**: slpmrmse})  
  
 **return** HttpResponse(**"This is Shit"**)  
  
**def** GetImageHOGRGBGLH(request):  
 **if** request.method==**'GET'**:  
 imgname = request.GET.get(**'imagename'**)  
 print(**"Values Must Be "**,imgname)  
 obj = StartProcess()  
 colorinfo, picbrightness, picshape = obj.process(imgname)  
  
 loginid = request.session[**'loginid'**]  
 data = CoronaDischargeModel.objects.filter(loginid=loginid)  
 **return** render(request, **'users/UserImageUploadForm.html'**, {**'data'**: data})

User Side Models:

**from** django.db **import** models  
  
*# Create your models here.***class** UserRegistrationModel(models.Model):  
 name = models.CharField(max\_length=100)  
 loginid = models.CharField(unique=**True**, max\_length=100)  
 password = models.CharField(max\_length=100)  
 mobile = models.CharField(unique=**True**, max\_length=100)  
 email = models.CharField(unique=**True**, max\_length=100)  
 locality = models.CharField(max\_length=100)  
 address = models.CharField(max\_length=1000)  
 city = models.CharField(max\_length=100)  
 state = models.CharField(max\_length=100)  
 status = models.CharField(max\_length=100)  
  
 **def** \_\_str\_\_(self):  
 **return** self.loginid  
  
 **class** Meta:  
 db\_table = **'UserRegistrations'  
  
  
class** CoronaDischargeModel(models.Model):  
 username = models.CharField(max\_length=100)  
 email = models.CharField(max\_length=100)  
 loginid = models.CharField(max\_length=100)  
 filename = models.CharField(max\_length=100)  
 file = models.FileField(upload\_to=**'files/'**)  
 redColor = models.FloatField()  
 greenColor = models.FloatField()  
 blueColor = models.FloatField()  
 picHeight = models.FloatField()  
 picWidht = models.FloatField()  
 blockofPixel = models.FloatField()  
 picbrightness = models.FloatField()  
 cdate = models.DateTimeField(auto\_now\_add=**True**)  
  
 **def** \_\_str\_\_(self):  
 **return** self.loginid  
  
 **class** Meta:  
 db\_table = **"CoronaDischargeImages"**

user form:

**from** django **import** forms  
**from** .models **import** UserRegistrationModel  
  
  
**class** UserRegistrationForm(forms.ModelForm):  
 name = forms.CharField(widget=forms.TextInput(attrs={**'pattern'**: **'[a-zA-Z]+'**}), required=**True**, max\_length=100)  
 loginid = forms.CharField(widget=forms.TextInput(attrs={**'pattern'**: **'[a-zA-Z]+'**}), required=**True**, max\_length=100)  
 password = forms.CharField(widget=forms.PasswordInput(attrs={**'pattern'**: **'(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}'**,  
 **'title'**: **'Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters'**}),  
 required=**True**, max\_length=100)  
 mobile = forms.CharField(widget=forms.TextInput(attrs={**'pattern'**: **'[56789][0-9]{9}'**}), required=**True**,  
 max\_length=100)  
 email = forms.CharField(widget=forms.TextInput(attrs={**'pattern'**: **'[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,}$'**}),  
 required=**True**, max\_length=100)  
 locality = forms.CharField(widget=forms.TextInput(), required=**True**, max\_length=100)  
 address = forms.CharField(widget=forms.Textarea(attrs={**'rows'**: 4, **'cols'**: 22}), required=**True**, max\_length=250)  
 city = forms.CharField(widget=forms.TextInput(  
 attrs={**'autocomplete'**: **'off'**, **'pattern'**: **'[A-Za-z ]+'**, **'title'**: **'Enter Characters Only '**}), required=**True**,  
 max\_length=100)  
 state = forms.CharField(widget=forms.TextInput(  
 attrs={**'autocomplete'**: **'off'**, **'pattern'**: **'[A-Za-z ]+'**, **'title'**: **'Enter Characters Only '**}), required=**True**,  
 max\_length=100)  
 status = forms.CharField(widget=forms.HiddenInput(), initial=**'waiting'**, max\_length=100)  
  
 **class** Meta():  
 model = UserRegistrationModel  
 fields = **'\_\_all\_\_'**

Sape utility Class

**from** sklearn **import** svm, metrics, datasets  
**from** sklearn.utils **import** Bunch  
**from** sklearn.model\_selection **import** train\_test\_split  
**import** matplotlib.pyplot **as** plt  
**from** matplotlib **import** style  
*#style.use("ggplot")***from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** tree  
**from** sklearn.linear\_model **import** Perceptron  
**from** sklearn.metrics **import** mean\_squared\_error  
**from** math **import** sqrt  
**from** sklearn.pipeline **import** make\_pipeline  
**from** sklearn.svm **import** SVC  
**from** sklearn.preprocessing **import** StandardScaler  
**import** numpy **as** np  
**class** UserImageShape:  
 **def** startSvm(self,df):  
 X = df[[**'picHeight'**,**'picWidht'**]]  
 y = df[**'picbrightness'**]  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=109)  
 print(df.head())  
 X\_train = np.array(X\_train)  
 y\_train = np.array(y\_train)  
 clf = make\_pipeline(StandardScaler(), SVC(gamma=**'auto'**))  
 clf.fit(X\_train, y\_train)  
 y\_pred = clf.predict(X\_test)  
 meanRmaseSVM = sqrt(mean\_squared\_error(y\_test, y\_pred))  
 print(**"SVM RSME= "**, meanRmaseSVM)  
 **return** meanRmaseSVM  
  
 **def** startKnn(self,df):  
 X = df[[**'picHeight'**, **'picWidht'**]]  
 y = df[**'picbrightness'**]  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=109)  
 print(df.head())  
 X\_train = np.array(X\_train)  
 y\_train = np.array(y\_train)  
 clf = KNeighborsClassifier(n\_neighbors=3)  
 clf.fit(X\_train, y\_train)  
 y\_pred = clf.predict(X\_test)  
 print(**"KNN Classification report for - :\n{}\n"**.format(metrics.classification\_report(y\_test, y\_pred)))  
 meanRmaseSVM = sqrt(mean\_squared\_error(y\_test, y\_pred))  
 print(**"KNN Mean RSME= "**, meanRmaseSVM)  
 **return** meanRmaseSVM  
  
 **def** startDecisionTree(self,df):  
 X = df[[**'picHeight'**, **'picWidht'**]]  
 y = df[**'picbrightness'**]  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=109)  
 print(df.head())  
 X\_train = np.array(X\_train)  
 y\_train = np.array(y\_train)  
 clf = tree.DecisionTreeClassifier()  
 clf.fit(X\_train, y\_train)  
 y\_pred = clf.predict(X\_test)  
 print(**"DT Classification report for - :\n{}\n"**.format(metrics.classification\_report(y\_test, y\_pred)))  
 meanRmaseSVM = sqrt(mean\_squared\_error(y\_test, y\_pred))  
 print(**"DT Mean RSME= "**, meanRmaseSVM)  
 **return** meanRmaseSVM  
  
 **def** startSLP(self,df):  
 X = df[[**'picHeight'**, **'picWidht'**]]  
 y = df[**'picbrightness'**]  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=109)  
 print(df.head())  
 X\_train = np.array(X\_train)  
 y\_train = np.array(y\_train)  
 clf = Perceptron(tol=1e-3, random\_state=0)  
 clf.fit(X\_train, y\_train)  
 y\_pred = clf.predict(X\_test)  
 print(**"SLP Classification report for - :\n{}\n"**.format(metrics.classification\_report(y\_test, y\_pred)))  
 meanRmaseSVM = sqrt(mean\_squared\_error(y\_test, y\_pred))  
 print(**"SLP Mean RSME= "**, meanRmaseSVM)  
 **return** meanRmaseSVM

Image Processing

**from** django.conf **import** settings  
*# import the necessary packages***from** matplotlib **import** pyplot **as** plt  
**import** numpy **as** np  
**from** skimage.feature **import** hog  
**import** cv2  
**from** skimage.transform **import** resize  
**from** skimage **import** exposure  
**import** matplotlib  
*#matplotlib.use("TkAgg")***def** threshold\_slow(T, image):  
 *# grab the image dimensions* h = image.shape[0]  
 w = image.shape[1]  
  
 *# loop over the image, pixel by pixel* **for** y **in** range(0, h):  
 **for** x **in** range(0, w):  
 *# threshold the pixel* image[y, x] = 255 **if** image[y, x] >= T **else** 0  
  
 *# return the thresholded image* **return** image  
**class** StartProcess:  
 **def** process(self,imagepath):  
 picshape = **''** colorinfpo = **''** picbrightnisee = **''** print(**"Image File "**,imagepath)  
 filepath = settings.MEDIA\_ROOT + **"\\datasets\\"** + imagepath  
 image = cv2.imread(filepath)  
 resized\_img = resize(image, (150, 950))  
 fd, hog\_image = hog(resized\_img, orientations=8, pixels\_per\_cell=(8, 8), cells\_per\_block=(4, 4), visualize=**True**,  
 multichannel=**True**)  
 hog\_image\_rescaled = exposure.rescale\_intensity(hog\_image, in\_range=(0, 10))  
 fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8), sharex=**True**, sharey=**True**)  
  
 ax1.imshow(resized\_img, cmap=plt.cm.gray)  
 ax1.set\_title(**'Input image'**)  
 ax2.imshow(hog\_image\_rescaled, cmap=plt.cm.gray)  
 ax2.set\_title(**'Histogram of Oriented Gradients'**)  
  
 plt.show()  
 cv2.imshow(**"image"**, image)  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 cv2.imshow(**"gray"**, gray)  
 hist = cv2.calcHist([gray], [0], **None**, [256], [0, 256])  
 plt.figure()  
 plt.title(**"Grayscale Histogram"**)  
 plt.xlabel(**"Bins"**)  
 plt.ylabel(**"# of Pixels"**)  
 plt.plot(hist)  
 plt.xlim([0, 256])  
 plt.show()  
  
 *# grab the image channels, initialize the tuple of colors,  
 # the figure and the flattened feature vector* chans = cv2.split(image)  
 colors = (**"b"**, **"g"**, **"r"**)  
 plt.figure()  
 plt.title(**"'Flattened' Color Histogram"**)  
 plt.xlabel(**"Bins"**)  
 plt.ylabel(**"# of Pixels"**)  
 features = []  
 *# loop over the image channels* **for** (chan, color) **in** zip(chans, colors):  
 *# create a histogram for the current channel and  
 # concatenate the resulting histograms for each  
 # channel* hist = cv2.calcHist([chan], [0], **None**, [256], [0, 256])  
 features.extend(hist)  
 *# plot the histogram* plt.plot(hist, color=color)  
 plt.xlim([0, 256])  
  
 *# here we are simply showing the dimensionality of the  
 # flattened color histogram 256 bins for each channel  
 # x 3 channels = 768 total values -- in practice, we would  
 # normally not use 256 bins for each channel, a choice  
 # between 32-96 bins are normally used, but this tends  
 # to be application dependent* plt.show()  
 print(**"flattened feature vector size: %d"** % (np.array(features).flatten().shape))  
 image = cv2.imread(filepath)  
 img = image  
 print(**"Shape "**, image.shape)  
 picshape = image.shape  
 avg\_color\_per\_row = np.average(image, axis=0)  
 avg\_color = np.average(avg\_color\_per\_row, axis=0)  
 print(**"Color info="**, avg\_color)  
 colorinfpo = avg\_color  
 plt.hist(image.ravel(), 256, [0, 256])  
 plt.show()  
 image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 a = np.array(image)  
 print(**"BrightNess="**, a.max(), np.unravel\_index(a.argmax(), a.shape))  
 picbrightnisee = a.max()  
 plt.imshow(image, cmap=**"gray"**)  
 plt.show()  
 arr = np.asarray(image)  
 print(len(arr))  
 *#####################################################################  
 # Get Oriantations* hh, ww, cc = img.shape  
  
 *# convert to gray* gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  
 *# threshold the grayscale image* ret, thresh = cv2.threshold(gray, 0, 255, 0)  
  
 *# find outer contour* cntrs = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
 cntrs = cntrs[0] **if** len(cntrs) == 2 **else** cntrs[1]  
  
 *# get rotated rectangle from outer contour* rotrect = cv2.minAreaRect(cntrs[0])  
 box = cv2.boxPoints(rotrect)  
 box = np.int0(box)  
  
 *# draw rotated rectangle on copy of img as result* result = img.copy()  
 cv2.drawContours(result, [box], 0, (0, 0, 255), 2)  
  
 *# get angle from rotated rectangle* angle = rotrect[-1]  
  
 *# from https://www.pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/  
 # the `cv2.minAreaRect` function returns values in the  
 # range [-90, 0); as the rectangle rotates clockwise the  
 # returned angle trends to 0 -- in this special case we  
 # need to add 90 degrees to the angle* print(**"Orinatations is "**, angle)  
 **if** angle < -45:  
 angle = -(90 + angle)  
  
 *# otherwise, just take the inverse of the angle to make  
 # it positive* **else**:  
 angle = -angle  
  
 print(angle, **"deg"**)  
  
 *# write result to disk  
 #cv2.imwrite("Resultimage.png", result)  
  
 #cv2.imshow("THRESH", thresh)  
 #cv2.imshow("RESULT", result)  
 #cv2.waitKey(0)  
 #cv2.destroyAllWindows()  
 ##########################################################  
 # arr = np.split(arr, 20)  
 # arr = np.array([np.split(x, 20, 1) for x in arr])  
 # print("Arr Values ",arr)* image = threshold\_slow(5, image)  
 plt.imshow(image, cmap=**"gray"**)  
 plt.show()  
  
 **return** colorinfpo,picbrightnisee,picshape

Algorithm codes:

**from** pathlib **import** Path  
**from** django.conf **import** settings  
**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
*# %matplotlib notebook***from** sklearn **import** svm, metrics, datasets  
**from** sklearn.utils **import** Bunch  
**from** sklearn.model\_selection **import** GridSearchCV, train\_test\_split  
**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn **import** tree  
**from** sklearn.linear\_model **import** Perceptron  
**from** sklearn.metrics **import** mean\_squared\_error  
**from** math **import** sqrt  
**from** skimage.io **import** imread  
**from** skimage.transform **import** resize  
  
  
**def** load\_image\_files(container\_path, dimension=(64, 64)):  
  
 image\_dir = Path(container\_path)  
 folders = [directory **for** directory **in** image\_dir.iterdir() **if** directory.is\_dir()]  
 categories = [fo.name **for** fo **in** folders]  
  
 descr = **"A image classification dataset"** images = []  
 flat\_data = []  
 target = []  
 **for** i, direc **in** enumerate(folders):  
 **for** file **in** direc.iterdir():  
 img = imread(file)  
 img\_resized = resize(img, dimension, anti\_aliasing=**True**, mode=**'reflect'**)  
 flat\_data.append(img\_resized.flatten())  
 images.append(img\_resized)  
 target.append(i)  
 flat\_data = np.array(flat\_data)  
 target = np.array(target)  
 images = np.array(images)  
  
 **return** Bunch(data=flat\_data,  
 target=target,  
 target\_names=categories,  
 images=images,  
 DESCR=descr)  
  
**class** UserSVMCode:  
 **def** startSvm(self):  
 *#filepath = settings.MEDIA\_ROOT + "\\datasets\\"* filepath = settings.MEDIA\_ROOT  
 image\_dataset = load\_image\_files(filepath)  
  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 image\_dataset.data, image\_dataset.target, test\_size=0.3, random\_state=109)  
  
 param\_grid = [  
 {**'C'**: [1, 10, 100, 1000], **'kernel'**: [**'linear'**]},  
 {**'C'**: [1, 10, 100, 1000], **'gamma'**: [0.001, 0.0001], **'kernel'**: [**'rbf'**]},  
 ]  
 svc = svm.SVC()  
 clf = GridSearchCV(svc, param\_grid)  
 clf.fit(X\_train, y\_train)  
 y\_pred = clf.predict(X\_test)  
 *#print("Classification report for - \n{}:\n{}\n".format(clf, metrics.classification\_report(y\_test, y\_pred)))* print(**"SVM Classification report for - :\n{}\n"**.format(metrics.classification\_report(y\_test, y\_pred)))  
 meanRmaseSVM = sqrt(mean\_squared\_error(y\_test, y\_pred))  
 print(**"SVm RSME= "**,meanRmaseSVM)  
 **return** meanRmaseSVM  
  
 **def** startKnn(self):  
 filepath = settings.MEDIA\_ROOT  
 image\_dataset = load\_image\_files(filepath)  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(image\_dataset.data, image\_dataset.target, test\_size=0.3, random\_state=109)  
 neigh = KNeighborsClassifier(n\_neighbors=3)  
 neigh.fit(X\_train, y\_train)  
 y\_pred = neigh.predict(X\_test)  
 print(**"KNN Classification report for - :\n{}\n"**.format(metrics.classification\_report(y\_test, y\_pred)))  
 meanRmaseSVM = sqrt(mean\_squared\_error(y\_test, y\_pred))  
 print(**"KNN Mean RSME= "**, meanRmaseSVM)  
 **return** meanRmaseSVM  
  
 **def** startDecisionTree(self):  
 filepath = settings.MEDIA\_ROOT  
 image\_dataset = load\_image\_files(filepath)  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(image\_dataset.data, image\_dataset.target, test\_size=0.3, random\_state=109)  
 dt = tree.DecisionTreeClassifier()  
 dt.fit(X\_train, y\_train)  
 y\_pred = dt.predict(X\_test)  
 print(**"DT Classification report for - :\n{}\n"**.format(metrics.classification\_report(y\_test, y\_pred)))  
 meanRmaseSVM = sqrt(mean\_squared\_error(y\_test, y\_pred))  
 print(**"DT Mean RSME= "**, meanRmaseSVM)  
 **return** meanRmaseSVM  
  
 **def** startSLP(self):  
 filepath = settings.MEDIA\_ROOT  
 image\_dataset = load\_image\_files(filepath)  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(image\_dataset.data, image\_dataset.target, test\_size=0.3, random\_state=109)  
 clf = Perceptron(tol=1e-3, random\_state=0)  
 clf.fit(X\_train, y\_train)  
 y\_pred = clf.predict(X\_test)  
 print(**"SLP Classification report for - :\n{}\n"**.format(metrics.classification\_report(y\_test, y\_pred)))  
 meanRmaseSVM = sqrt(mean\_squared\_error(y\_test, y\_pred))  
 print(**"SLP Mean RSME= "**, meanRmaseSVM)  
 **return** meanRmaseSVM

Admin side views:

**from** django.shortcuts **import** render  
**from** django.contrib **import** messages  
**from** users.models **import** UserRegistrationModel,CoronaDischargeModel  
**from** .utility.AdminViewGLH **import** StartProcess  
**import** matplotlib  
*#matplotlib.use("Agg")  
  
# Create your views here.***def** AdminLoginCheck(request):  
 **if** request.method == **'POST'**:  
 usrid = request.POST.get(**'loginid'**)  
 pswd = request.POST.get(**'pswd'**)  
 print(**"User ID is = "**, usrid)  
 **if** usrid == **'admin' and** pswd == **'admin'**:  
 **return** render(request, **'admins/AdminHome.html'**)  
 **elif** usrid == **'Admin' and** pswd == **'Admin'**:  
 **return** render(request, **'admins/AdminHome.html'**)  
 **else**:  
 messages.success(request, **'Please Check Your Login Details'**)  
 **return** render(request, **'AdminLogin.html'**, {})  
  
  
**def** AdminHome(request):  
 **return** render(request, **'admins/AdminHome.html'**)  
  
  
**def** ViewRegisteredUsers(request):  
 data = UserRegistrationModel.objects.all()  
 **return** render(request, **'admins/RegisteredUsers.html'**, {**'data'**: data})  
  
  
**def** AdminActivaUsers(request):  
 **if** request.method == **'GET'**:  
 id = request.GET.get(**'uid'**)  
 status = **'activated'** print(**"PID = "**, id, status)  
 UserRegistrationModel.objects.filter(id=id).update(status=status)  
 data = UserRegistrationModel.objects.all()  
 **return** render(request, **'admins/RegisteredUsers.html'**, {**'data'**: data})  
  
**def** AdminViewCoroanDischarges(request):  
 data = CoronaDischargeModel.objects.all()  
 **return** render(request, **'admins/AdminViewImages.html'**,{**'data'**:data})  
  
**def** GetImageGLH(request):  
 **if** request.method == **'GET'**:  
 imgname = request.GET.get(**'imagename'**)  
 print(**"Values Must Be "**, imgname)  
 obj = StartProcess()  
 obj.process(imgname)  
 data = CoronaDischargeModel.objects.all()  
 **return** render(request, **'admins/AdminViewImages.html'**, {**'data'**: data})

All Urls.py

*"""StressDetection URL Configuration  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/2.0/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""***from** django.contrib **import** admin  
**from** django.urls **import** path  
**from** CoronaRecognition **import** views **as** mainView  
**from** users **import** views **as** usr  
**from** admins **import** views **as** admins  
**from** django.contrib.staticfiles.urls **import** static  
**from** django.contrib.staticfiles.urls **import** staticfiles\_urlpatterns  
**from** django.conf **import** settings  
  
urlpatterns = [  
 path(**'admin/'**, admin.site.urls),  
 path(**""**, mainView.index, name=**"index"**),  
 path(**"index/"**, mainView.index, name=**"index"**),  
 path(**"logout/"**, mainView.logout, name=**"logout"**),  
 path(**"UserLogin/"**, mainView.UserLogin, name=**"UserLogin"**),  
 path(**"AdminLogin/"**, mainView.AdminLogin, name=**"AdminLogin"**),  
 path(**"UserRegister/"**, mainView.UserRegister, name=**"UserRegister"**),  
  
 *### User Side Views* path(**"UserRegisterActions/"**, usr.UserRegisterActions, name=**"UserRegisterActions"**),  
 path(**"UserLoginCheck/"**, usr.UserLoginCheck, name=**"UserLoginCheck"**),  
 path(**"UserHome/"**, usr.UserHome, name=**"UserHome"**),  
 path(**"UploadImageForm/"**, usr.UploadImageForm, name=**"UploadImageForm"**),  
 path(**"UploadImageAction/"**, usr.UploadImageAction, name=**"UploadImageAction"**),  
 path(**"UserSVMTest/"**, usr.UserSVMTest, name=**"UserSVMTest"**),  
 path(**"UserShapeTest/"**, usr.UserShapeTest, name=**"UserShapeTest"**),  
 path(**"UserBrightness/"**, usr.UserBrightness, name=**"UserBrightness"**),  
 path(**"GetImageHOGRGBGLH/"**, usr.GetImageHOGRGBGLH, name=**"GetImageHOGRGBGLH"**),  
  
 *### Admin Side Views* path(**"AdminLoginCheck/"**, admins.AdminLoginCheck, name=**"AdminLoginCheck"**),  
 path(**"AdminHome/"**, admins.AdminHome, name=**"AdminHome"**),  
 path(**"ViewRegisteredUsers/"**, admins.ViewRegisteredUsers, name=**"ViewRegisteredUsers"**),  
 path(**"AdminActivaUsers/"**, admins.AdminActivaUsers, name=**"AdminActivaUsers"**),  
 path(**"AdminViewCoroanDischarges/"**, admins.AdminViewCoroanDischarges, name=**"AdminViewCoroanDischarges"**),  
 path(**"GetImageGLH/"**, admins.GetImageGLH, name=**"GetImageGLH"**),  
  
  
  
]  
  
urlpatterns += staticfiles\_urlpatterns()  
urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

User base.html

<!DOCTYPE **html**>  
{%load static%}  
<**html lang="en"**>  
 <**head**>  
 <**meta charset="utf-8"** />  
 <**meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"** />  
 <**meta name="description" content=""** />  
 <**meta name="author" content=""** />  
 <**title**>Electrical Discharge Of Image</**title**>  
 <**link rel="icon" type="image/x-icon" href="{%static 'img/favicon.ico'%}"** />  
 *<!-- Font Awesome icons (free version)-->* <**script src="https://use.fontawesome.com/releases/v5.13.0/js/all.js" crossorigin="anonymous"**></**script**>  
 *<!-- Google fonts-->* <**link href="https://fonts.googleapis.com/css?family=Varela+Round" rel="stylesheet"** />  
 <**link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,400i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet"** />  
 *<!-- Core theme CSS (includes Bootstrap)-->* <**link href="{%static 'css/styles.css'%}" rel="stylesheet"** />  
 </**head**>  
 <**body id="page-top"**>  
 *<!-- Navigation-->* <**nav class="navbar navbar-expand-lg navbar-light fixed-top" id="mainNav"**>  
 <**div class="container"**>  
 <**a class="navbar-brand js-scroll-trigger" href="#page-top"**>Corona Recognition Method Based on Visible Light Color</**a**>  
 <**button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation"**>  
 Menu  
 <**i class="fas fa-bars"**></**i**>  
 </**button**>  
 <**div class="collapse navbar-collapse" id="navbarResponsive"**>  
 <**ul class="navbar-nav ml-auto"**>  
 <**li class="nav-item"**><**a class="nav-link js-scroll-trigger" href="{%url 'index'%}"**>Home</**a**></**li**>  
 <**li class="nav-item"**><**a class="nav-link js-scroll-trigger" href="{%url 'UserLogin'%}"**>Users</**a**></**li**>  
 <**li class="nav-item"**><**a class="nav-link js-scroll-trigger" href="{%url 'AdminLogin'%}"**>Admin</**a**></**li**>  
 <**li class="nav-item"**><**a class="nav-link js-scroll-trigger" href="{%url 'UserRegister'%}"**>Registrations</**a**></**li**>  
 </**ul**>  
 </**div**>  
 </**div**>  
 </**nav**>  
 *<!-- Masthead-->*{%block contents%}  
{%endblock%}  
 *<!-- About-->* <**section class="about-section text-center" id="about"**>  
 <**div class="container"**>  
 <**div class="row"**>  
 <**div class="col-lg-8 mx-auto"**>  
 <**h2 class="text-white mb-4"**>Characteristic information extraction</**h2**>  
 <**p class="text-white-50"**>  
 Each image is composed of a large number of pixels, and in an effort to extract sufficient amount of meaningful color information from the selected areas of pixels, x (0, 150), y (0, 950), shown as Fig. 1(b), we chose the same area of (m × n = 150 × 950) pixels across all images.  
  
 </**p**>  
 </**div**>  
 </**div**>  
 <**img class="img-fluid" src="{%static 'img/ipad.png'%}" alt=""** />  
 </**div**>  
 </**section**>  
 *<!-- Projects-->* <**section class="projects-section bg-light" id="projects"**>  
 <**div class="container"**>  
 *<!-- Featured Project Row-->* <**div class="row align-items-center no-gutters mb-4 mb-lg-5"**>  
 <**div class="col-xl-8 col-lg-7"**><**img class="img-fluid mb-3 mb-lg-0" src="{%static 'img/bg-masthead.jpg'%}" alt=""** /></**div**>  
 <**div class="col-xl-4 col-lg-5"**>  
 <**div class="featured-text text-center text-lg-left"**>  
 <**h4**>Shoreline</**h4**>  
 <**p class="text-black-50 mb-0"**>Grayscale is open source and MIT licensed. This means you can use it for any project - even commercial projects! Download it, customize it, and publish your website!</**p**>  
 </**div**>  
 </**div**>  
 </**div**>  
 *<!-- Project One Row-->* <**div class="row justify-content-center no-gutters mb-5 mb-lg-0"**>  
 <**div class="col-lg-6"**><**img class="img-fluid" src="{%static 'img/demo-image-01.jpg'%}" alt=""** /></**div**>  
 <**div class="col-lg-6"**>  
 <**div class="bg-black text-center h-100 project"**>  
 <**div class="d-flex h-100"**>  
 <**div class="project-text w-100 my-auto text-center text-lg-left"**>  
 <**h4 class="text-white"**>Misty</**h4**>  
 <**p class="mb-0 text-white-50"**>An example of where you can put an image of a project, or anything else, along with a description.</**p**>  
 <**hr class="d-none d-lg-block mb-0 ml-0"** />  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 *<!-- Project Two Row-->* <**div class="row justify-content-center no-gutters"**>  
 <**div class="col-lg-6"**><**img class="img-fluid" src="{%static 'img/demo-image-02.jpg'%}" alt=""** /></**div**>  
 <**div class="col-lg-6 order-lg-first"**>  
 <**div class="bg-black text-center h-100 project"**>  
 <**div class="d-flex h-100"**>  
 <**div class="project-text w-100 my-auto text-center text-lg-right"**>  
 <**h4 class="text-white"**>Mountains</**h4**>  
 <**p class="mb-0 text-white-50"**>Another example of a project with its respective description. These sections work well responsively as well, try this theme on a small screen!</**p**>  
 <**hr class="d-none d-lg-block mb-0 mr-0"** />  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**section**>  
 *<!-- Signup-->* <**section class="signup-section" id="signup"**>  
 <**div class="container"**>  
 <**div class="row"**>  
 <**div class="col-md-10 col-lg-8 mx-auto text-center"**>  
 <**i class="far fa-paper-plane fa-2x mb-2 text-white"**></**i**>  
 <**h2 class="text-white mb-5"**>Subscribe to receive updates!</**h2**>  
 <**form class="form-inline d-flex"**>  
 <**input class="form-control flex-fill mr-0 mr-sm-2 mb-3 mb-sm-0" id="inputEmail" type="email" placeholder="Enter email address..."** />  
 <**button class="btn btn-primary mx-auto" type="submit"**>Subscribe</**button**>  
 </**form**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**section**>  
 *<!-- Contact-->* <**section class="contact-section bg-black"**>  
 <**div class="container"**>  
 <**div class="row"**>  
 <**div class="col-md-4 mb-3 mb-md-0"**>  
 <**div class="card py-4 h-100"**>  
 <**div class="card-body text-center"**>  
 <**i class="fas fa-map-marked-alt text-primary mb-2"**></**i**>  
 <**h4 class="text-uppercase m-0"**>Address</**h4**>  
 <**hr class="my-4"** />  
 <**div class="small text-black-50"**>4923 Market Street, Orlando FL</**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="col-md-4 mb-3 mb-md-0"**>  
 <**div class="card py-4 h-100"**>  
 <**div class="card-body text-center"**>  
 <**i class="fas fa-envelope text-primary mb-2"**></**i**>  
 <**h4 class="text-uppercase m-0"**>Email</**h4**>  
 <**hr class="my-4"** />  
 <**div class="small text-black-50"**><**a href="#!"**>hello@yourdomain.com</**a**></**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="col-md-4 mb-3 mb-md-0"**>  
 <**div class="card py-4 h-100"**>  
 <**div class="card-body text-center"**>  
 <**i class="fas fa-mobile-alt text-primary mb-2"**></**i**>  
 <**h4 class="text-uppercase m-0"**>Phone</**h4**>  
 <**hr class="my-4"** />  
 <**div class="small text-black-50"**>+1 (555) 902-8832</**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**div**>  
 <**div class="social d-flex justify-content-center"**>  
 <**a class="mx-2" href="#!"**><**i class="fab fa-twitter"**></**i**></**a**>  
 <**a class="mx-2" href="#!"**><**i class="fab fa-facebook-f"**></**i**></**a**>  
 <**a class="mx-2" href="#!"**><**i class="fab fa-github"**></**i**></**a**>  
 </**div**>  
 </**div**>  
 </**section**>  
 *<!-- Footer-->* <**footer class="footer bg-black small text-center text-white-50"**><**div class="container"**>Copyright © Your Alex Corporation 2020</**div**></**footer**>  
 *<!-- Bootstrap core JS-->* <**script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"**></**script**>  
 <**script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.bundle.min.js"**></**script**>  
 *<!-- Third party plugin JS-->* <**script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.4.1/jquery.easing.min.js"**></**script**>  
 *<!-- Core theme JS-->* <**script src="{%static 'js/scripts.js'%}"**></**script**>  
 </**body**>  
</**html**>

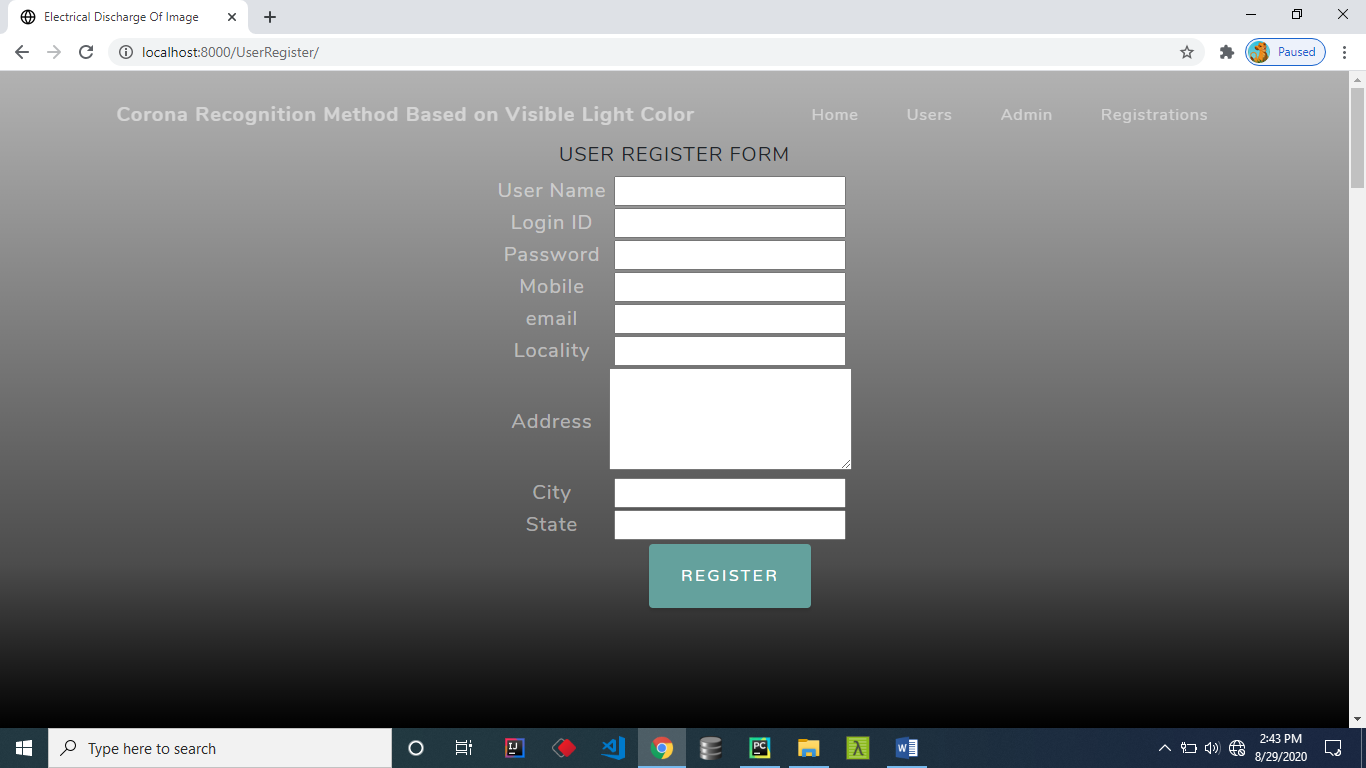
**IMPLEMENTATION**

**SCREEN SHOTS**

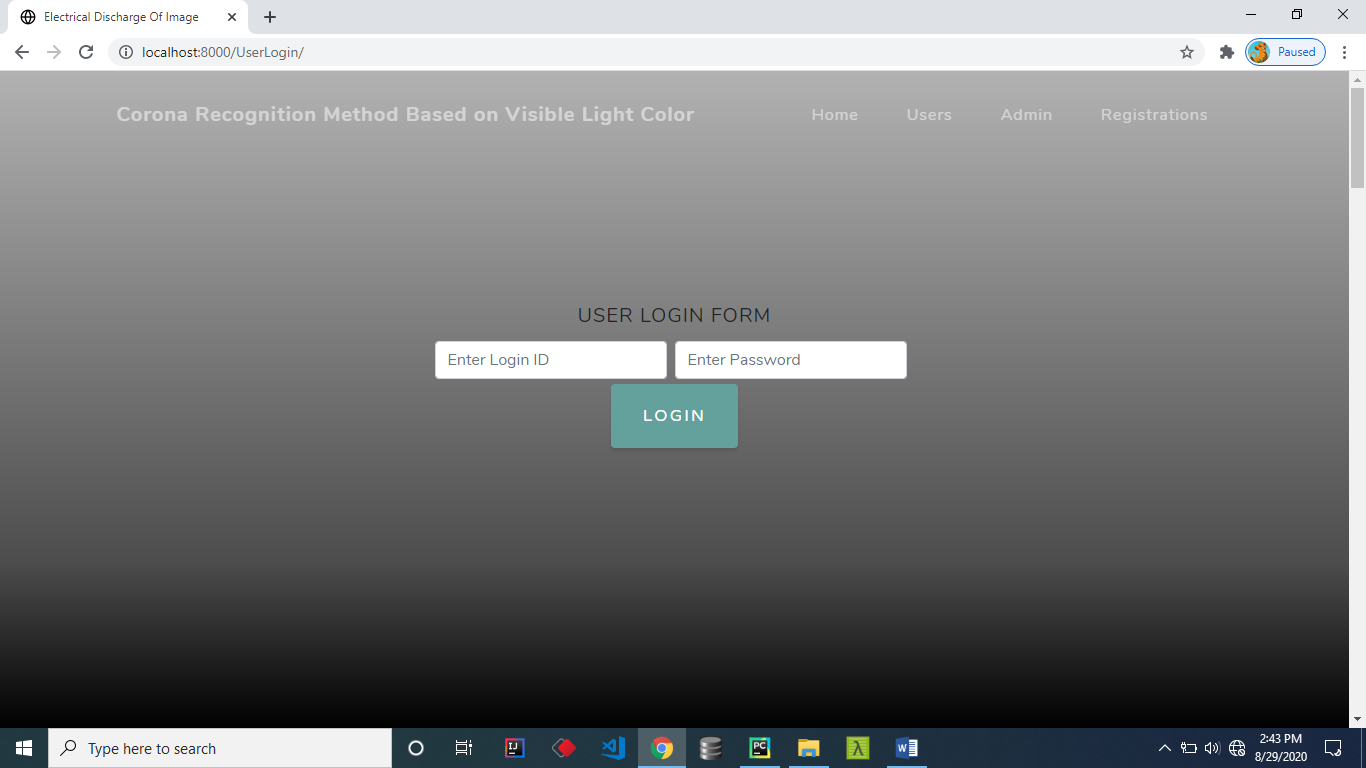
**Home page:**



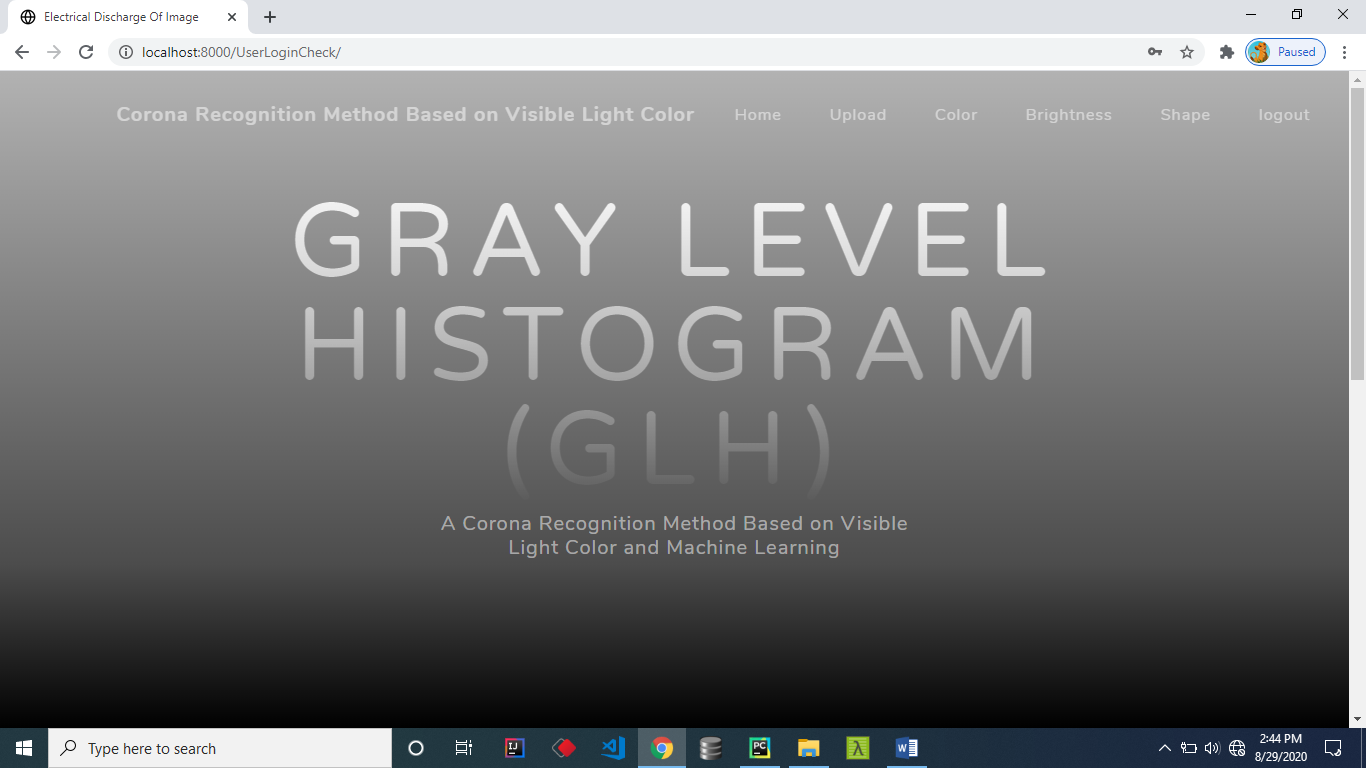
**User Register page:**



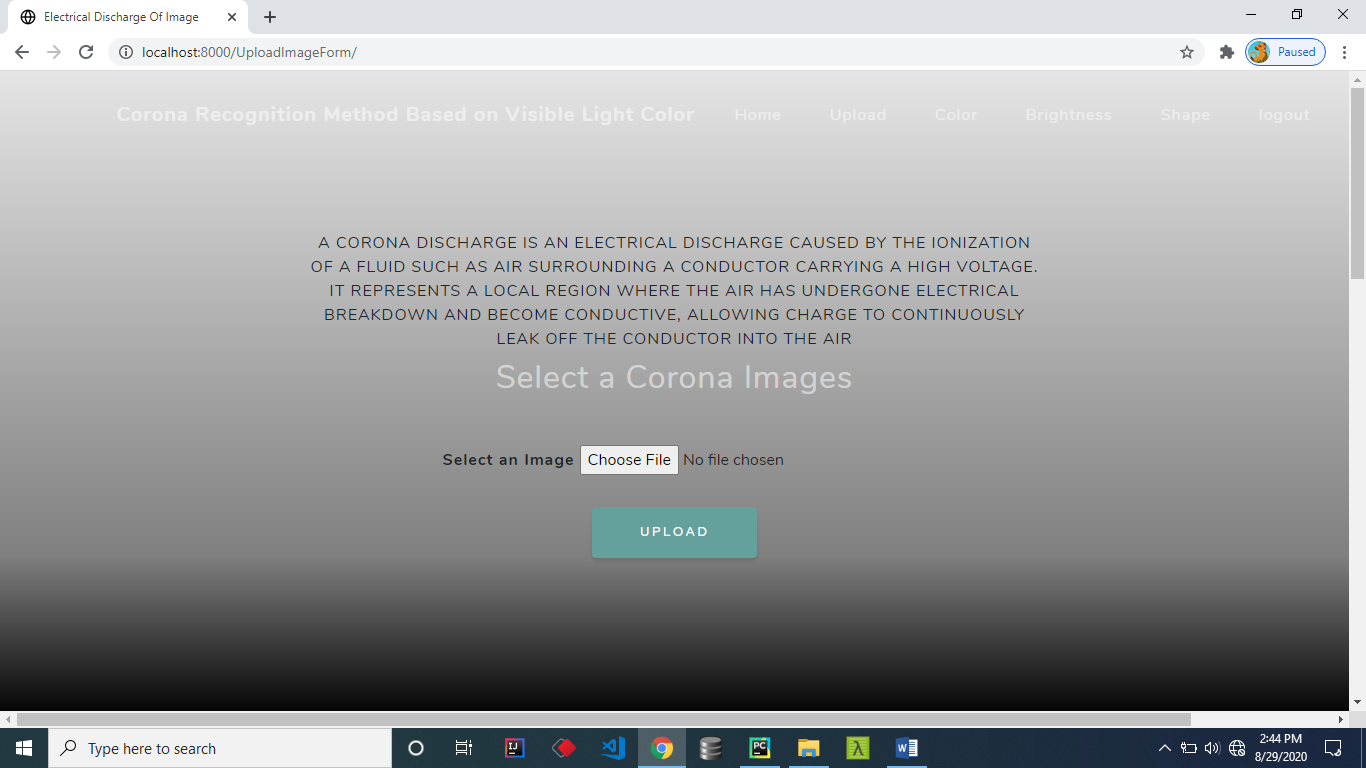
**User Login page:**



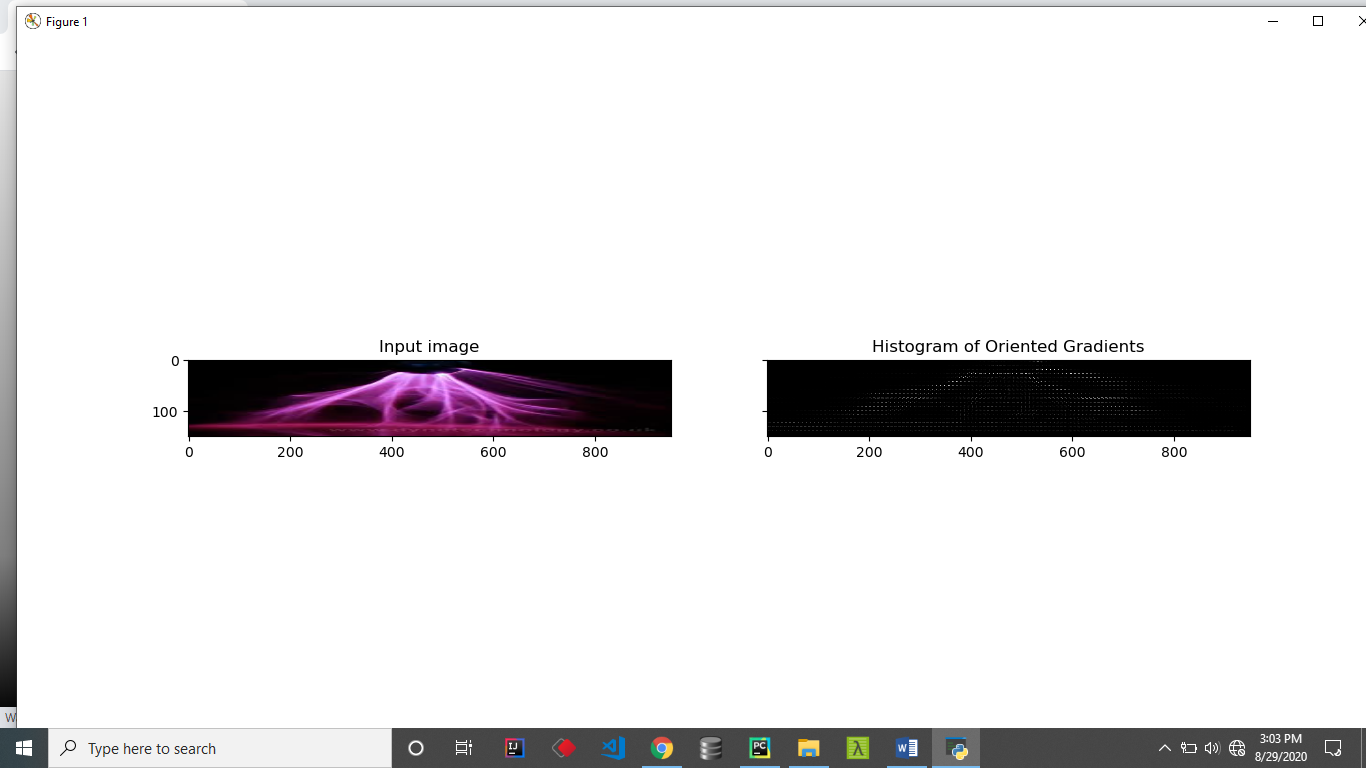
**User Home Page:**



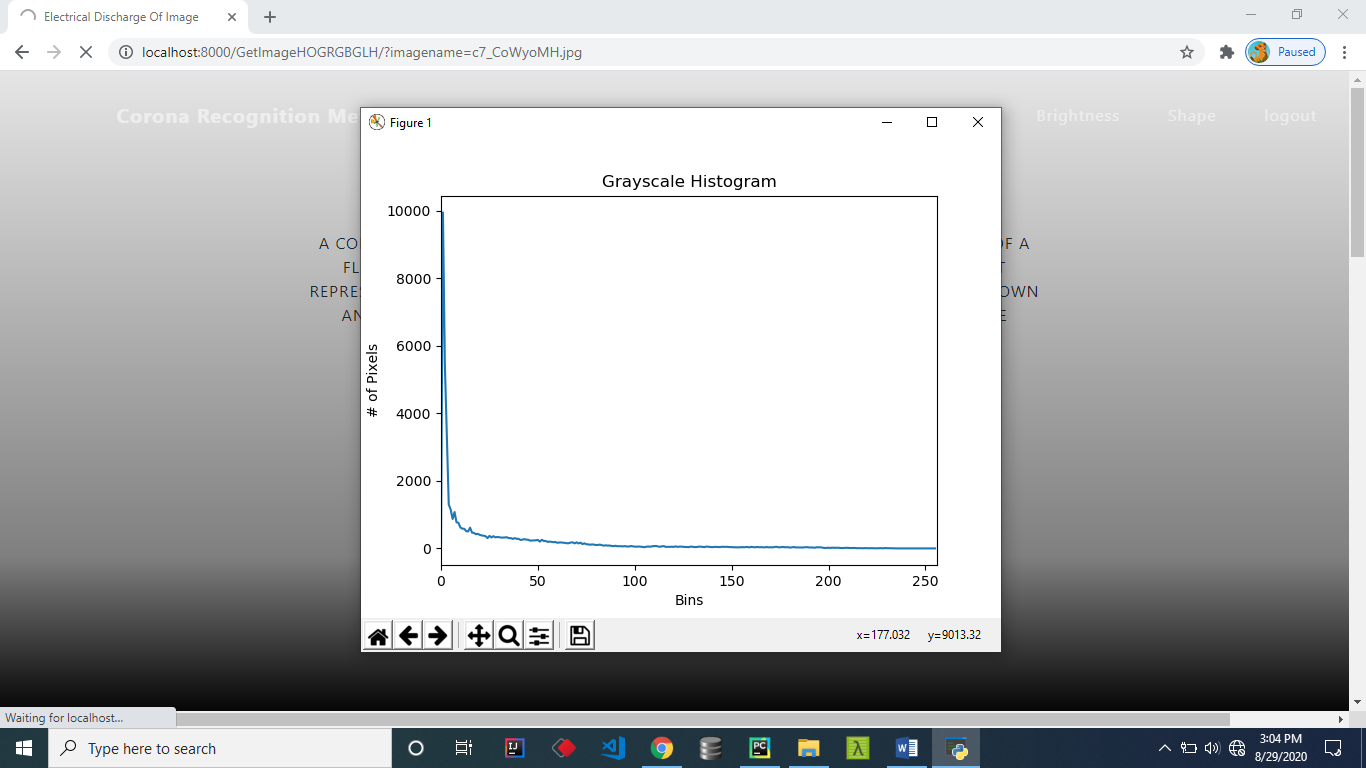
**Image Upload page:**



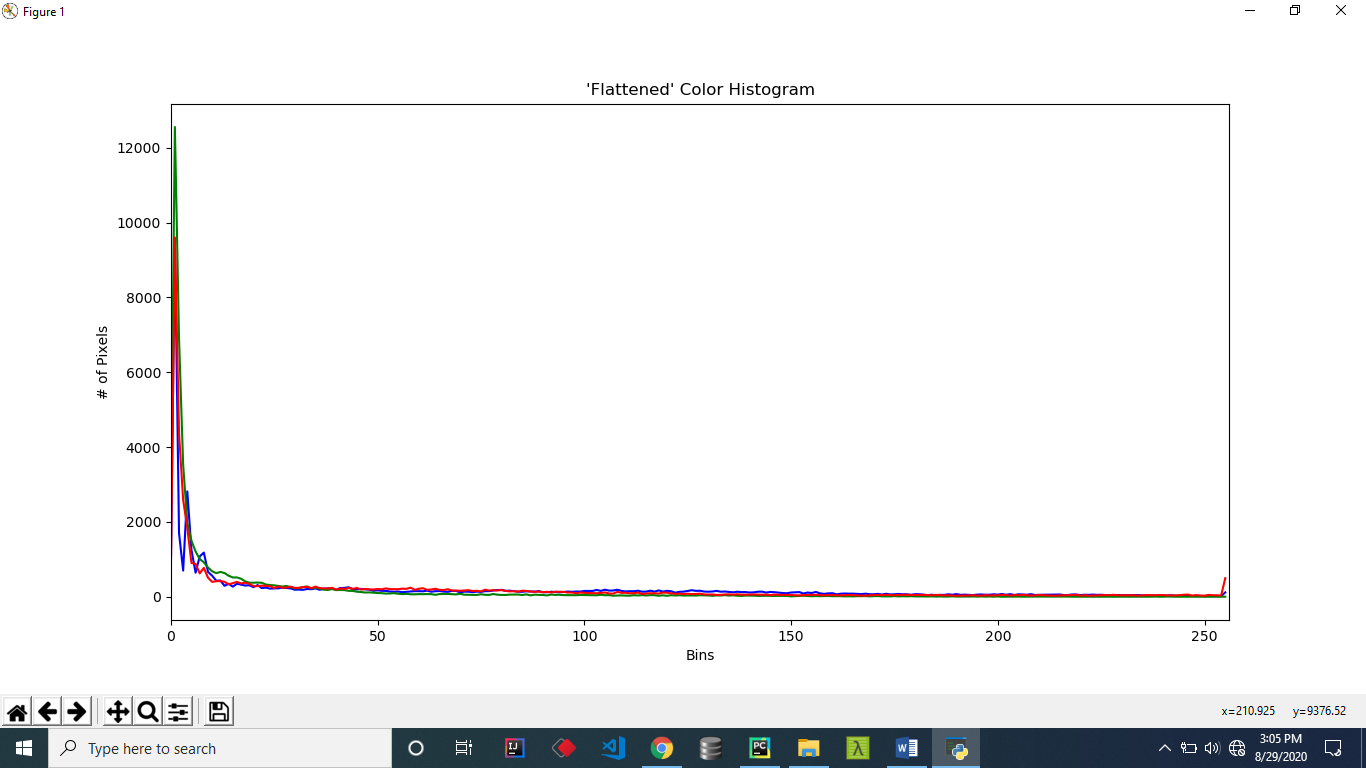
**Input Image**



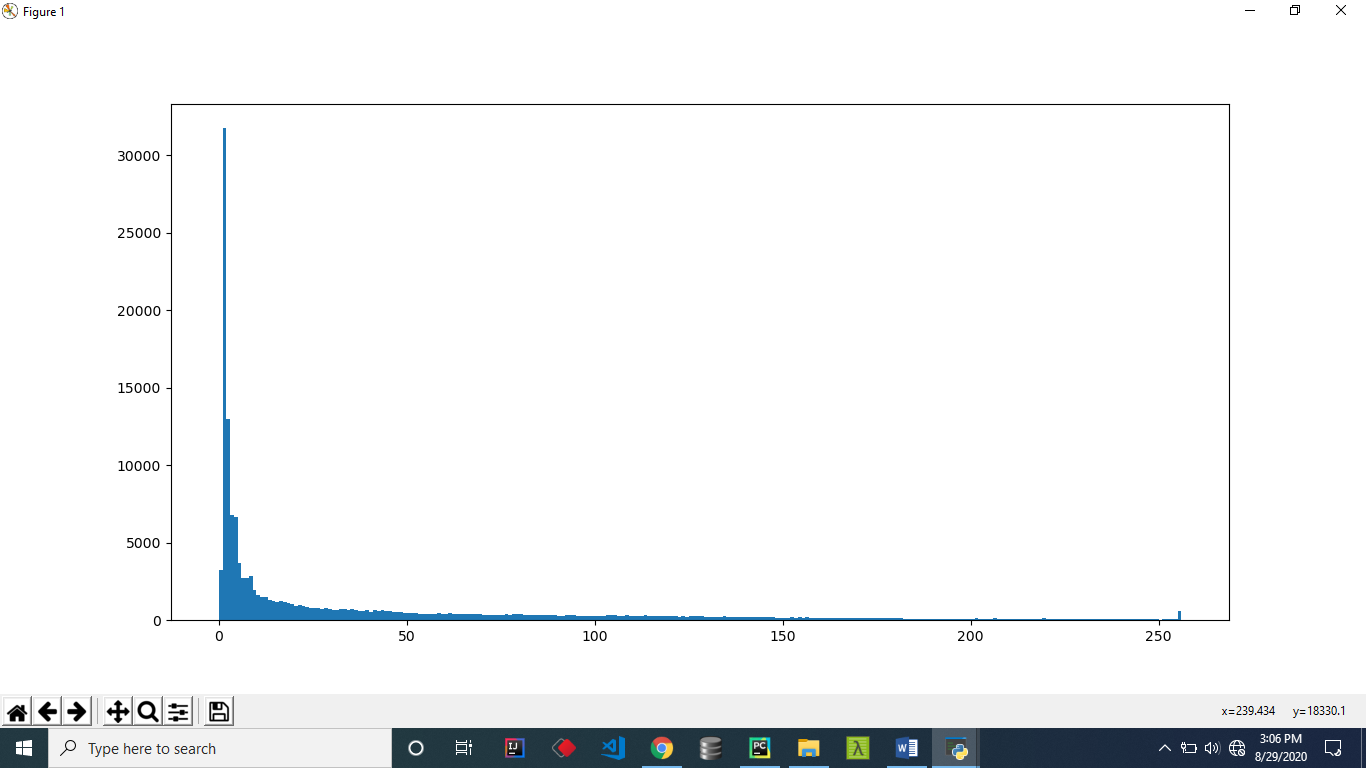
**Grayscale**



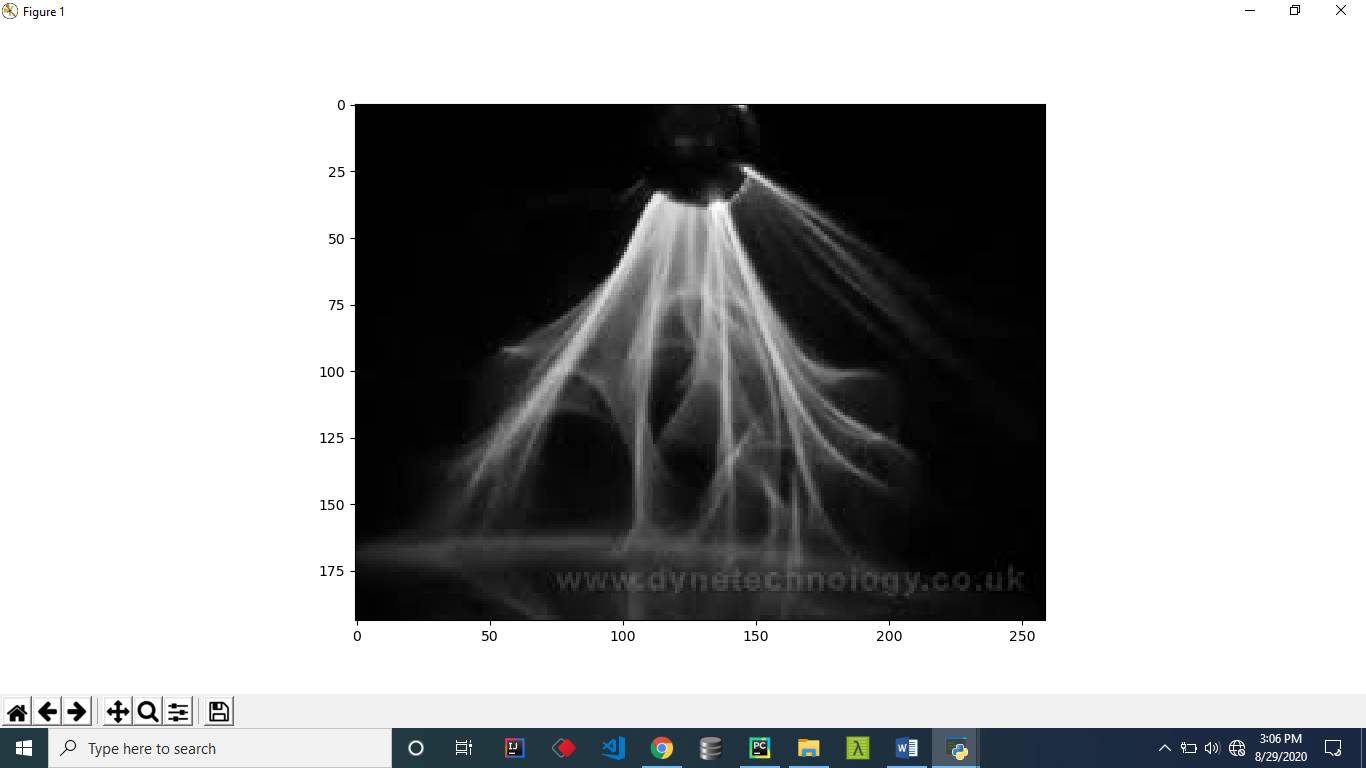
**RGB Orientations:**



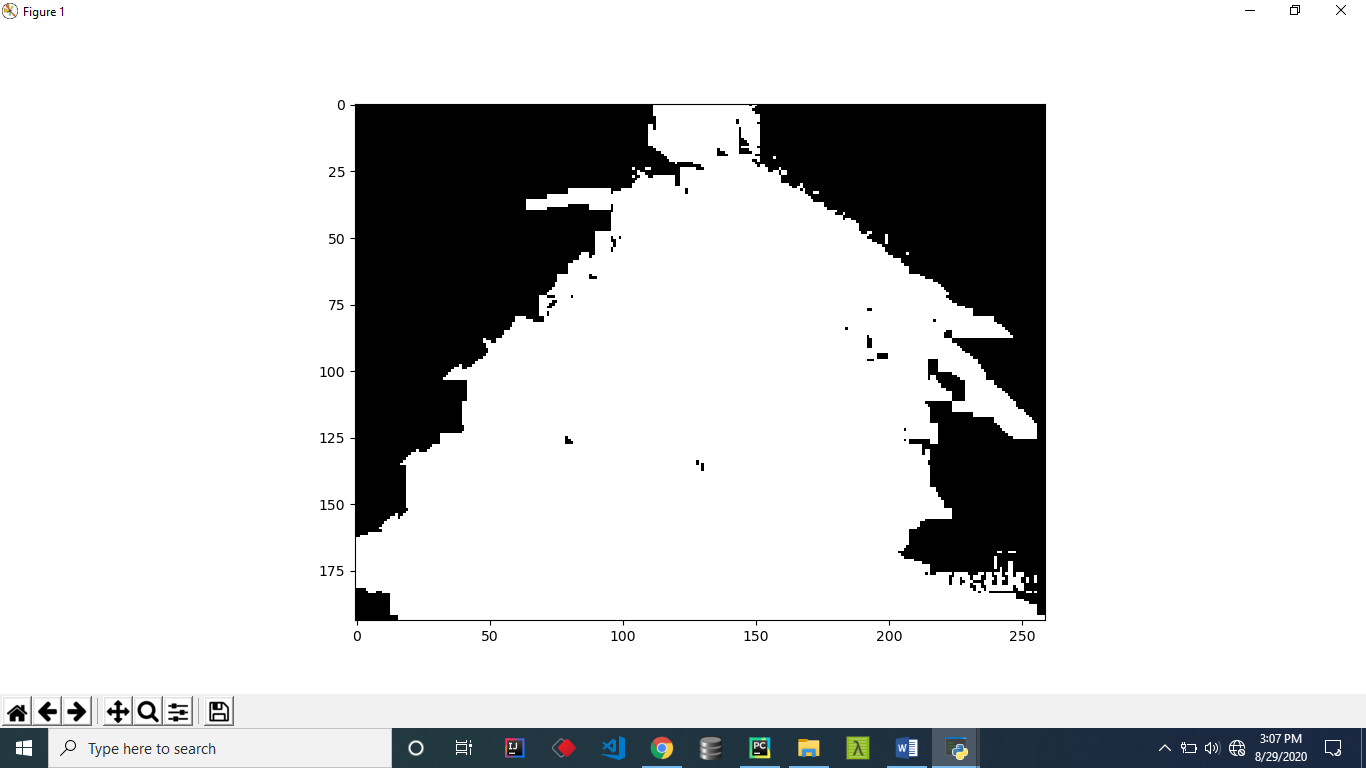
**HOG Codes:**



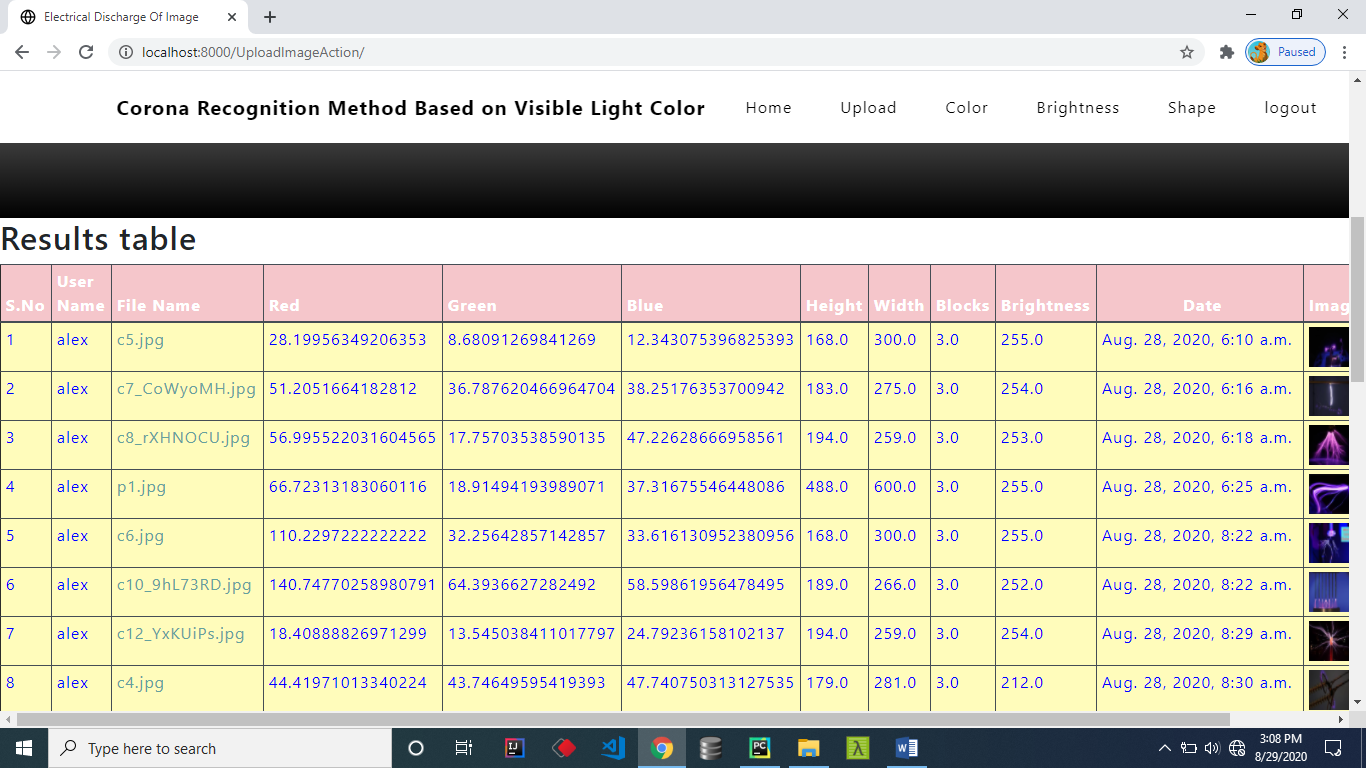
**Gray Scale:**



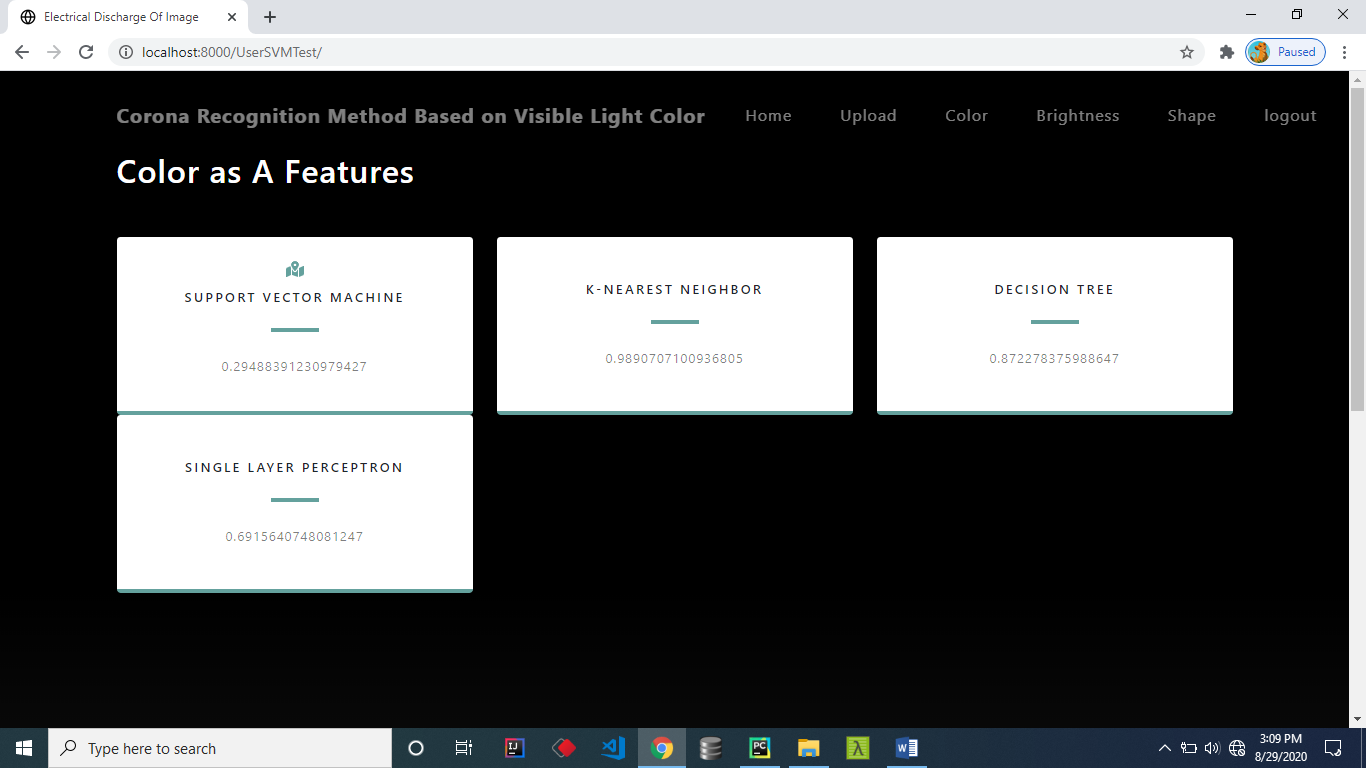
**8 block Orintation:**



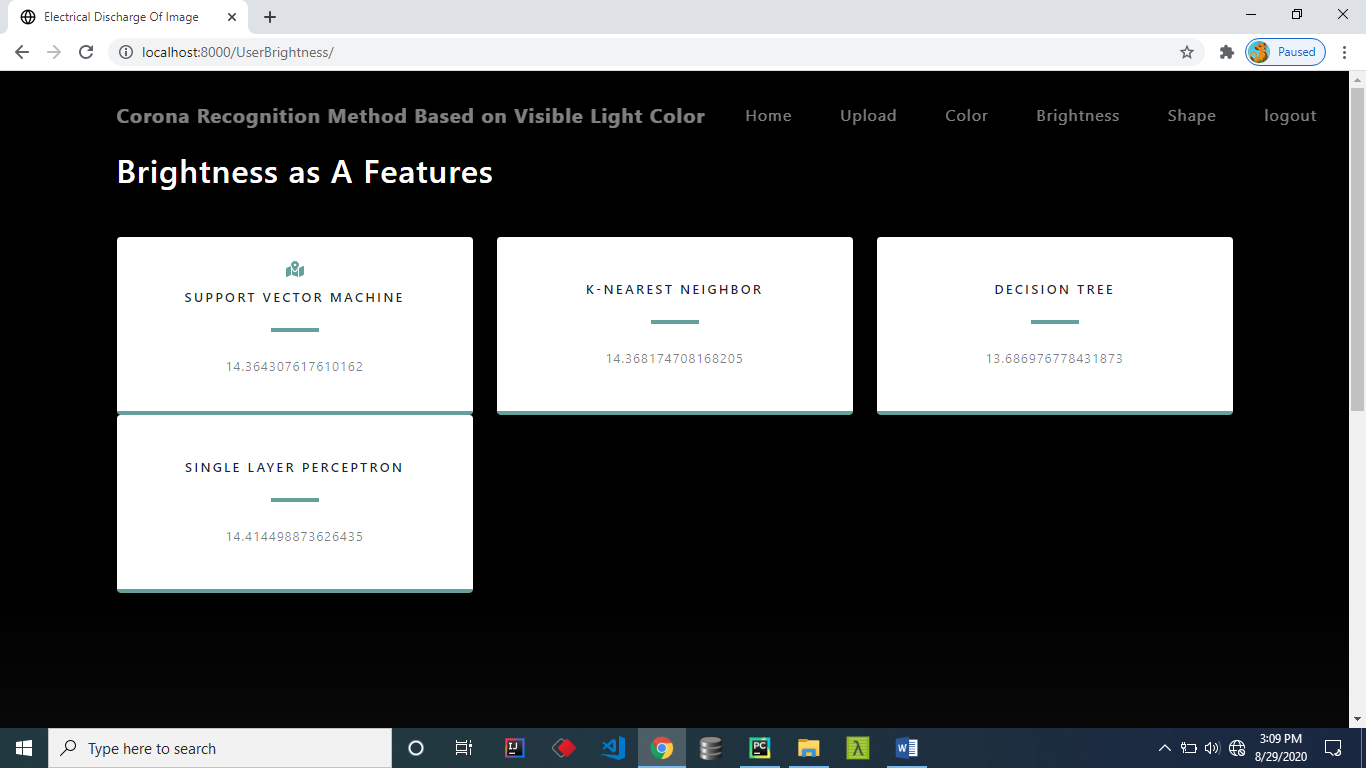
**Uploaded Images:**



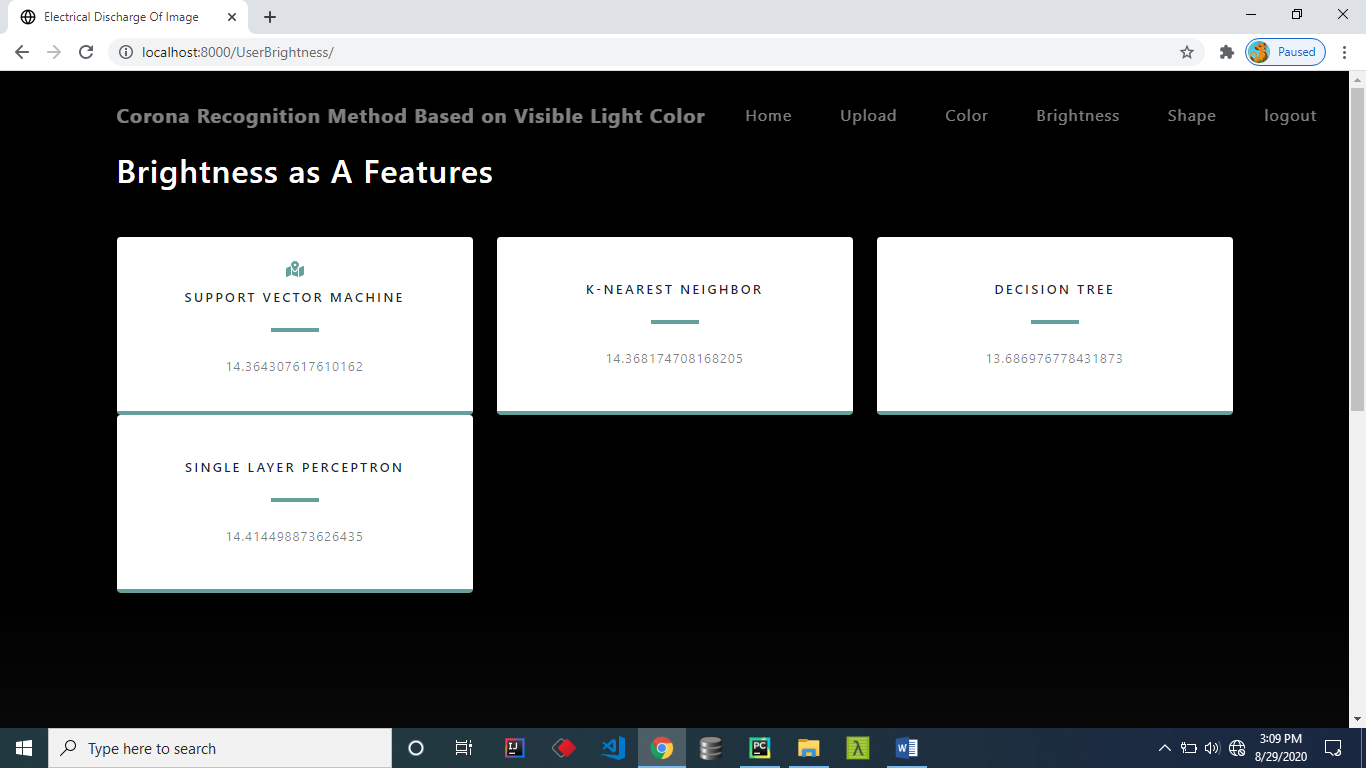
**Color Features**



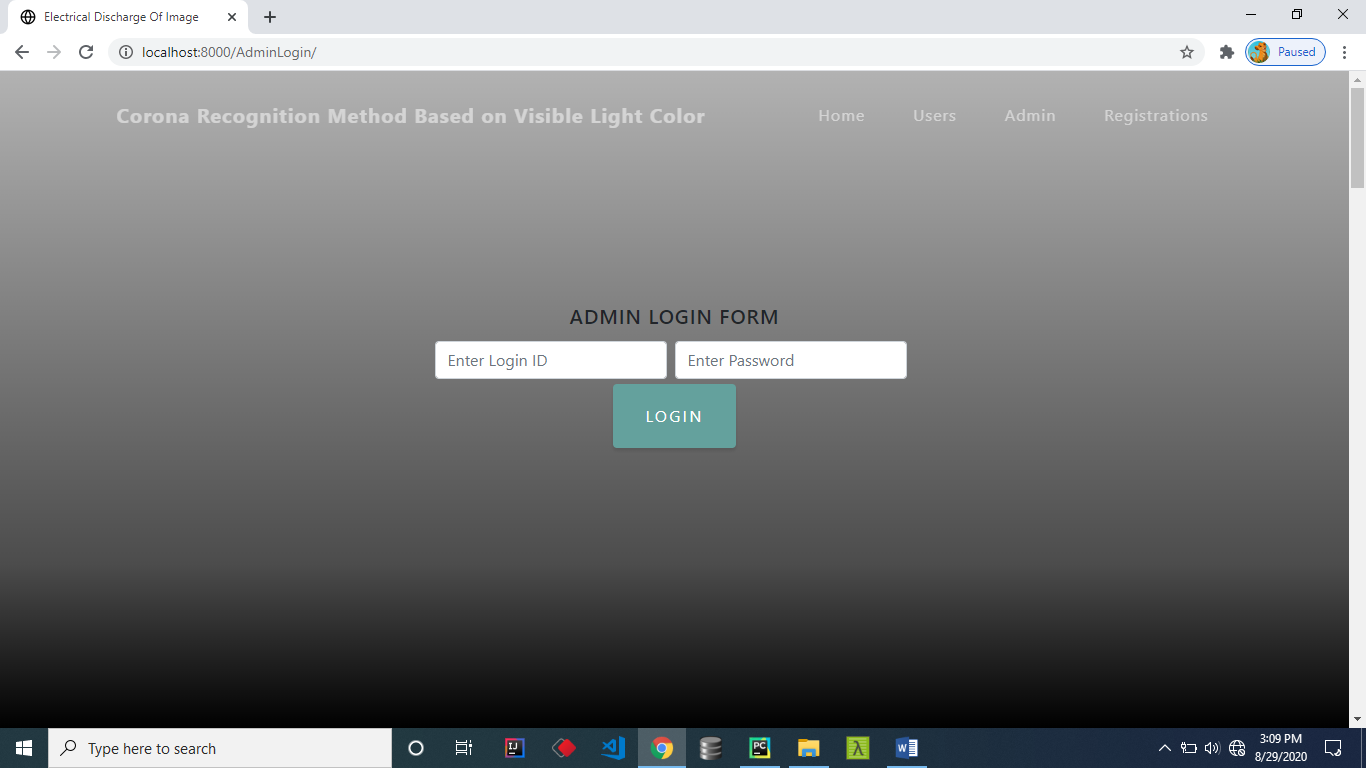
**Brightness Feature**



**Shape Features**



**Admin Login pag:**



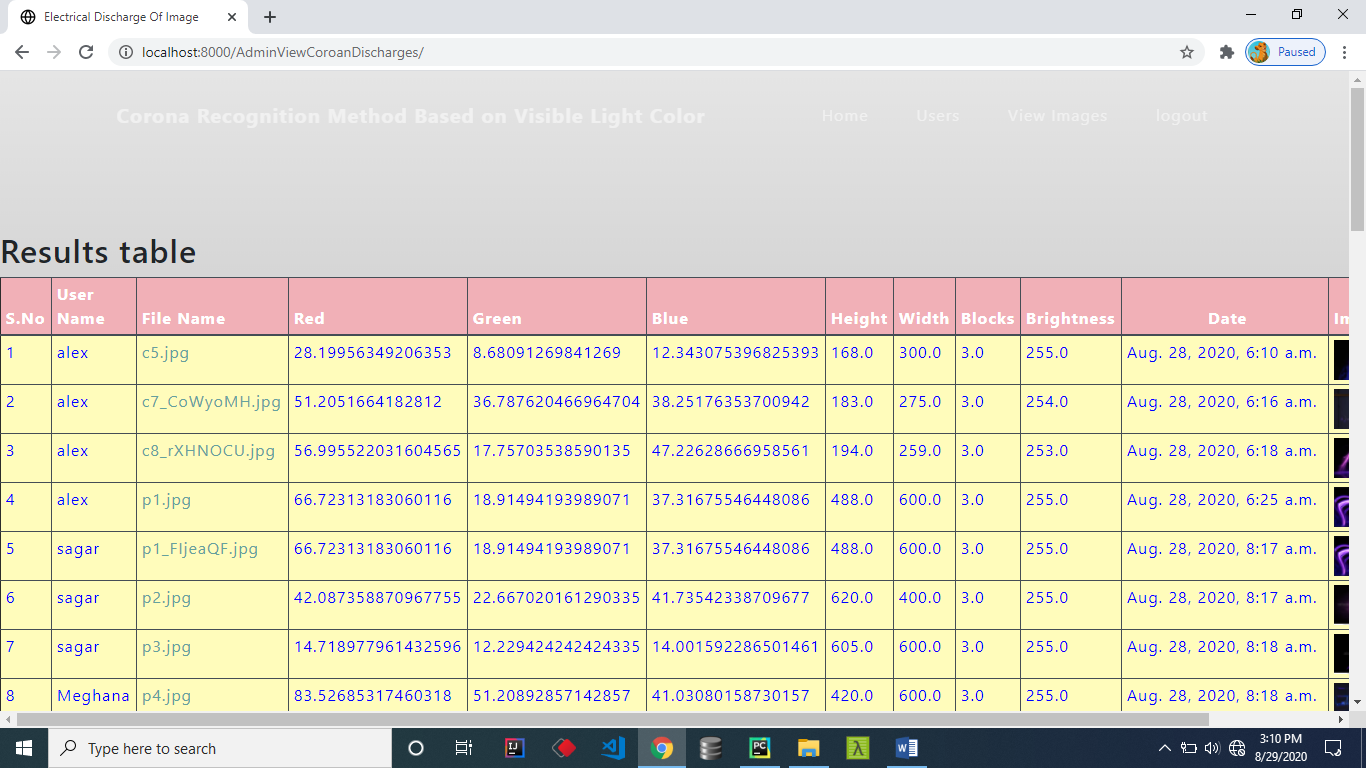
**Admin Home Page:**



**Activating Register users**



**User all Images:**



**TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### TYPES OF TESTS

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Sample Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no** | **Test Case** | **Excepted Result** | **Result** | **Remarks(IF Fails)** |
| 1. | User Register | If User registration successfully. | Pass | If already user email exist then it fails. |
| 2. | User Login | If User name and password is correct then it will getting valid page. | Pass | Un Register Users will not logged in. |
| 3. | Start Data Pre process | Image has toupload | Pass | Corona Discharge Image must be select |
| 4. | GLH Gray Level Hostorgram | Image Gray level Histogram will generate | Pass | Image GLH will create based on Python library |
| 5. | RGB Color Recognitions | Image RGB Score and generated on graph | Pass | Each image RGB Colors based on image |
| 6. | Image Brightness Calcuated | Per block and per black pixels calculated | Pass | Image pixel block will calculates |
| 7. | Mease RMSE Calculated | Measn RMSE first we need to find RMSE | Pass | Rmse Score calclayed based on this MeanRmase calculated |
| 8. | Models Executed | For out four algorithms has executed and calculated 3 features | Pass | Models executed and predicted the results |
| 9. | Admin login | Admin can login with his login credential. If success he get his home page | Pass | Invalid login details will not allowed here |
| 10. | Admin can activate the register users | Admin can activate the register user id | Pass | If user id not found then it won’t login. |

**CONCLUSION**

The prediction shows that the color features perform the best among all the three characteristics information and the KNN algorithm performs the best among all four algorithms The model shows consistently good performance with different cameras and camera settings as well. Discharge produces radiation of UV, visible, and nearinfrared wavelengths. Past studies focused mostly on the UV spectrum, yet the measurement of the light spectrum demonstrates that radiation intensity of the visible spectrum can be high as well. Objects that can produce radiation of visible spectrum is affected by radiation across all spectra. Because of this, even though our RGB-GLH method uses information only from the visible spectrum, it is still able to encompass discharge status-related information across spectra, thus enabling us to build a more successful model. The RGB color information characteristics method can also be applied to other discharge types other than corona.

**FURTHER ENHANCEMENT**

Corona discharge only forms when the electric field (potential gradient) at the surface of the conductor exceeds a critical value, the dielectric strength or disruptive potential gradient of the fluid. In air at atmospheric pressure, it is roughly 30 kilovolts per centimeter, but this decreases with pressure, so corona is more of a problem at high altitudes. A corona discharge is an electrical discharge possible because of the ionization of air surrounding a conductor that is electrically charged. The corona treatment is frequently used for polypropylene, PVC, PET, polyethylene, metallized surfaces, paper, and paperboard stock. Electric cables, automotive components, 3D parts, medical devices, pipes & tubes, board & foam, domestic appliances, extruded profiles are some components that are processed with corona.

.

**REFERENCES**

[1] D. T. Markus and T. S. Lee, “A method of direct corona imaging on a dielectric transparency,” IEEE Trans. Ind. Appl., vol. 32, no. 4, pp. 832–836, Jul. 1996.

[2] Y. Arao, T. Fujii, and M. Yatsuzuka, “Image evaluation of atmospheric corona discharge applied for road tunnel high-velocity electrostatic precipitator,” IEEE Trans. Plasma Sci., vol. 33, no. 2, pp. 312–313, Apr. 2005.

[3] P. H. Ceccato, O. Guaitella, M. Le Gloahec, and A. Rousseau, “Timeresolved nanosecond imaging of the propagation of a corona-like plasma discharge in water at positive applied voltage polarity,” J. Phys. D: Appl. Phys., vol. 43, no. 17, 2010, Art. no. 175202.

[4] S. N. Abolmasov, K. Tachibana, and T. Shirafuji, “Mechanisms of pattern formation in dielectric barrier discharges,” IEEE Trans. Plasma Sci., vol. 39, no. 11, pp. 2090–2091, Nov. 2011.

[5] M. Liao, J. Huang, G. Ge, X. Duan, Z. Huang, and J. Zou, “Vacuum arc commutation characteristics of the DC microgrid hybrid circuit breakers,” IEEE Trans. Plasma Sci., vol. 45, no. 8, pp. 2172–2178, Aug. 2017.

[6] D. B.Watson, S. K. Kho, K. A. Samuels, L. Ma, and J. B. Chiu, “Impulse flashover trajectory in air in nonuniform fields,” IEEE Trans. Electr. Insul., vol. 28, no. 2, pp. 200–208, Apr. 1993.

[7] D. Amarasinghe, U. Sonnadara, M. Berg, and V. Cooray, “Correlation between brightness and channel currents of electrical discharges,” IEEE Trans. Dielectrics Electr. Insul., vol. 14, no. 5, pp. 1154–1160, Oct. 2007.

[8] J. M. K. MacAlpine, D. H. Qiu, and Z. Y. Li, “An analysis of spark paths in air using 3-dimensional image processing,” IEEE Trans. Dielectrics Electr. Insul., vol. 6, no. 3, pp. 331–336, Jun. 1999.

[9] D. H. Qiu, J. M. K. MacAlpine, and Z. Y. Li, “An incremental analysis of spark paths in air using 3-dimensional image processing,” IEEE Trans. Dielectrics Electr. Insul., vol. 7, no. 6, pp. 758–763, Dec. 2000.

[10] L. Kebbabi and A. Beroual, “Fractal analysis of creeping discharge patterns propagating at solid/liquid interfaces: Influence of the nature and geometry of solid insulators,” J. Phys. D: Appl. Phys., vol. 39, no. 1, pp. 177–183, Jan. 2006.

[11] C. Liu, D. Dobrynin, and A. Fridman, “Uniform and non-uniform modes of nanosecond-pulsed dielectric barrier discharge in atmospheric air: Fast imaging and spectroscopic measurements of electric fields,” J. Phys. D: Appl. Phys., vol. 47, no. 25, Jun. 2014, Art. no. 252003.

[12] B. Pongrác, H. H. Kim, M. Janda, V. Martišovitš, and Z. Machala, “Fast imaging of intermittent electrospraying of water with positive corona discharge,” J. Phys. D: Appl. Phys., vol. 47, no. 31, 2014, Art. no. 315202.

[13] J. M. Palomares, A. Kohut, G. Galbács, R. Engeln, and Z. Geretovszky, “A time-resolved imaging and electrical study on a high current atmospheric pressure spark discharge,” J. App. Phys., vol. 118, no. 23, 2015, Art. no. 233305.

[14] X. Li et al., “Statistical evaluation of AC corona images in long-time scale and characterization of short-gap leader,” IEEE Trans. Dielectr. Electr. Insul., vol. 23, no. 1, pp. 165–173, Feb. 2016.

[15] P. C. Russell and G. R. Jones, “Chromatic monitoring of plasma and plasma systems,” Vacuum, vol. 58, pp. 88–99, Aug. 2000.

[16] K. Koppisetty, M. Serkan, and H. Kirkici, “Image analysis: A tool for optical-emission characterization of partial-vacuum breakdown,” IEEE Trans. Plasma Sci., vol. 37, no. 1, pp. 153–158, Jan. 2009.

[17] J. M. Serrano, R. Ruiz-Lombera, J. J. Valdiande, A. Cobo, and J. M. Lopez-Higuera, “Colorimetric analysis for on-line arc-welding diagnostics by means of plasma optical spectroscopy,” IEEE Sensors J., vol. 16, no. 10, pp. 3465–3471, May 2016.

[18] Q. Ye, F. Li, and Y. Hu, “A method of dividing corona discharge stage by visible light image chromaticity processing,” CN Patent 106 778 855 A, Dec. 8, 2016.

[19] Q. Ye and Z. Guo, “A visible light image RGB recognition method for transmission equipment corona in sunlight environment,” CN Patent 107 862 317 A, Oct. 23, 2017.

[20] Q. Ye, X. Dong, and Y. Wang, “A method of diagnosing surface discharge based on visible light in sunlight environment,” CN Patent 108 230 408 A, Jan. 18, 2018.

[21] D. S. Prasad and B. S. Reddy, “Study on corona activity using an image processing approach,” IEEE Trans. Ind. Appl., vol. 53, no. 4, pp. 4008–4014, Jul./Aug. 2017.

[22] D. S. Prasad and B. S. Reddy, “Digital image processing techniques for estimating power released from the corona discharges,” IEEE Trans. Dielectr. Electr. Insul., vol. 24, no. 1, pp. 75–82, Feb. 2017.

[23] V. S. Murthy, K. Tarakanath, D. K. Mohanta, and S. Gupta, “Insulator condition analysis for overhead distribution lines using combined wavelet support vector machine (SVM),” IEEE Trans. Dielectr. Electr. Insul., vol. 17, no. 1, pp. 89–99, Feb. 2010.

[24] H. Janani, B. Kordi, and M. J. Jozani, “Classification of simultaneous multiple partial discharge sources based on probabilistic interpretation using a two-step logistic regression algorithm,” IEEE Trans. Dielectr. Electr. Insul., vol. 24, no. 1, pp. 54–65, Feb. 2017.

[25] K. Wang et al., “A new image-oriented feature extraction method for partial discharges,” IEEE Trans. Dielectr. Electr. Insul., vol. 22, no. 2, pp. 1015–1024, Apr. 2015.

[26] F. N. Budiman, Y. Khan, N. H. Malik, A. A. Al-Arainy, and A. Beroual, “Utilization of artificial neural network for the estimation of size and position of metallic particle adhering to spacer in GIS,” IEEE Trans. Dielectr. Electr. Insul., vol. 20, no. 6, pp. 2143–2151, Dec. 2013.

[27] J. Tang, D. Wang, L. Fan, R. Zhuo, and X. Zhang, “Feature parameters extraction of GIS partial discharge signal with multifractal detrended fluctuation analysis,” IEEE Trans. Dielectr. Electr. Insul., vol. 22, no. 5, pp. 3037–3045, Oct. 2015.

[28] M. Odstrcil, A. Murari, and J. Mlynar, “Comparison of advanced machine learning tools for disruption prediction and disruption studies,” IEEE Trans. Plasma Sci., vol. 41, no. 7, pp. 1751–1759, Jul. 2013.

[29] A. Shabbir, G. Verdoolaege, J. Vega, and A. Murari, “Elm regime classification by conformal prediction on an information manifold,” IEEE Trans. Plasma Sci., vol. 43, no. 12, pp. 4190–4199, Dec. 2015.

[30] C. Cortes and V. Vapnik, “Support-vector networks,” Mach. Learn., vol. 20, no. 3, pp. 273–297, 1995.

[31] S. Raudys, “Evolution and generalization of a single neurone: I. Singlelayer perceptron as seven statistical classifiers,” Neural Netw., vol. 11, pp. 283–296, Mar. 1998.

[32] T. Hastie and R. Tibshirani, “Discriminant adaptive nearest neighbor classification,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 18, no. 6, pp. 607–616, Jun. 1996.

[33] L. A. Breslow and D. W. Aha, “Simplifying decision trees: A survey,” Knowl. Eng. Rev., vol. 12, pp. 1–40, Jan. 1997.