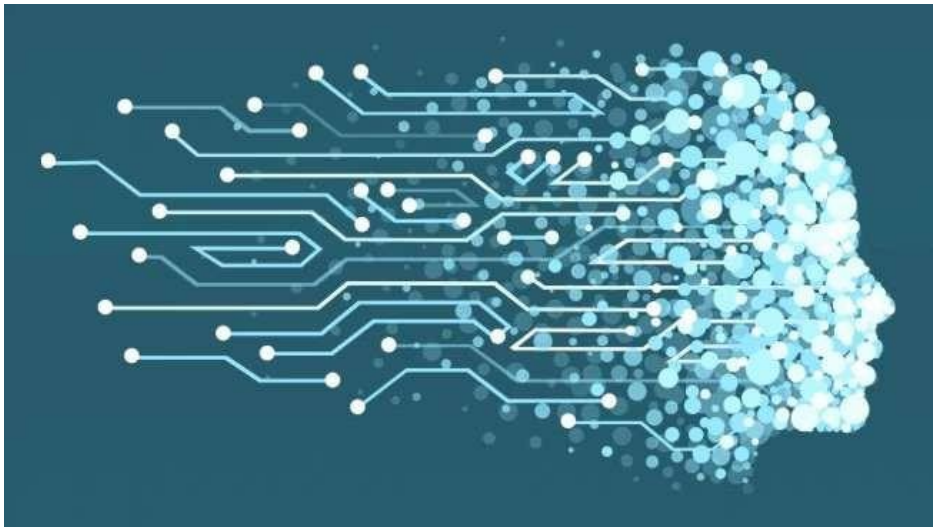
CS-33: Machine Learning with Python

BCA Semester – 6



Unit – 2

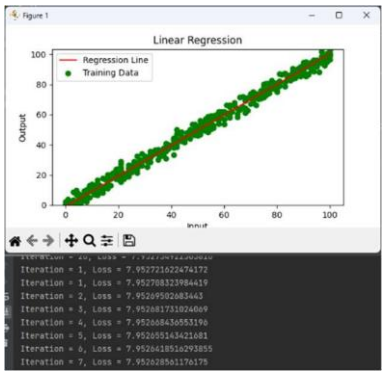
Supervised Learning

1. **Linear Regression Example**

```
#linear Regression Example

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

url = 'data_for_lr.csv'
data = pd.read_csv(url)
data
# Drop the missing values
data = data.dropna()

# training dataset and labels
train_input = np.array(data.x[0:500]).reshape(500, 1)
train_output = np.array(data.y[0:500]).reshape(500, 1)

# valid dataset and labels
test_input = np.array(data.x[500:700]).reshape(199, 1)
test_output = np.array(data.y[500:700]).reshape(199, 1)

class linearRegression:
    def __init__(self):
        self.parameters = {}
    def forward_propagation(self, train_input):
        m = self.parameters["m"]
        c = self.parameters["c"]
        predictions = np.multiply(m, train_input) + c
        return predictions
    def cost_function(self, predictions, train_output):
        cost = np.mean((train_output - predictions) ** 2)
        return cost

    def backward_propagation(self, train_input, train_output, predictions):
        derivatives = {}
        df = (predictions-train_output)
        # dm= 2/n * mean of (predictions-actual) * input
        dm = 2 * np.mean(np.multiply(train_input, df))
        # dc = 2/n * mean of (predictions-actual)
        dc = 2 * np.mean(df)
        derivatives["dm"] = dm
        derivatives['dc'] = dc
        return derivatives
    def update_parameters(self, derivatives, learning_rate):
       self.parameters["m"] = self.parameters["m"] - learning_rate * derivatives["dm"]
       self.parameters["c"] = self.parameters["c"] - learning_rate * derivatives['dc']
    def train(self, train_input, train_output, learning_rate, iters):
        # Initialize random parameters
        self.parameters["m"] = np.random.uniform(0, 1) * -1
        self.parameters["c"] = np.random.uniform(0, 1) * -1

        # Initialize loss
        self.loss = []

        # Initialize figure and axis for animation
        fig, ax = plt.subplots()
        x_vals = np.linspace(min(train_input), max(train_input), 100)
        line, = ax.plot(x_vals, self.parameters["m"] * x_vals +

```

```
60                          self.parameters["c"],          color='red',
label='Regression line')
61          ax.scatter(train_input,  train_output,  marker="o",
62                  color='green', label='Training Data')
63
64          # Set y-axis limits to exclude negative values
65          ax.set_ylim(0, max(train_output) + 1)
66
67          def update(frame):
68              # Forward propagation
69              predictions = self.forward_propagation(train_input)
70
71              # Cost function
72              cost = self.cost_function(predictions, train_output)
73
74              # Back propagation
75              derivatives = self.backward_propagation(
76                  train_input, train_output, predictions)
77
78              # Update parameters
79              self.update_parameters(derivatives, learning_rate)
80
81              # Update the regression line
82              line.set_ydata(self.parameters["m"]
83                             * x_vals + self.parameters["c"])
84
85              # Append loss and print
86              self.loss.append(cost)
87              print("Iteration = {}, loss = {}".format(frame + 1,
cost))
88
89              return line,
90          # Create animation
91           ani = FuncAnimation(fig, update, frames=iters, interval=200,
blit=True)
92
93          # Save the animation as a video file (e.g., MP4)
94           ani.save('linear_regression_A.gif', writer="ffmpeg")
95
96          plt.xlabel('Input')
97          plt.ylabel("Output")
98          plt.title('Linear Regression')
99          plt.legend()
100         plt.show()
101         return self.parameters, self.loss
102
103 #Example usage
104 linear_reg = LinearRegression()
105 parameters, loss   =   linear_reg.train(train_input,    train_output,
0.0001, 20)
```

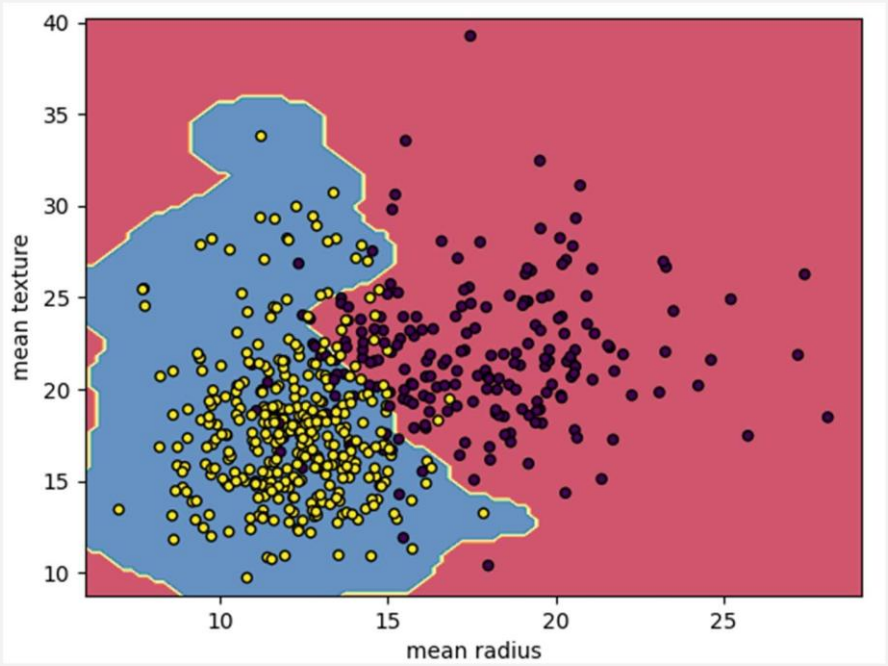
Output:



2. Classifier using Support Vector Machine example

```
#Classifier using Support Vector Machine example

# load the important packages
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.svm import SVC

# load the datasets
cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target

#Build the model
svm = SVC(kernel="rbf", gamma=0.5, C=1.0)
# Trained the model
svm.fit(X, y)

# Plot Decision Boundary
DecisionBoundaryDisplay.from_estimator(
        svm,
        X,
        response_method="predict",
        cmap=plt.cm.Spectral,
        alpha=0.8,
        xlabel=cancer.feature_names[0],
        ylabel=cancer.feature_names[1],
    )

# Scatter plot
plt.scatter(X[:, 0], X[:, 1],
            c=y,
            s=20, edgecolors="k")
plt.show()
```

Output:



3. Logistic Regression Example

```
#logistic Regression Example

from sklearn.model_selection import train_test_split
from sklearn import datasets, linear_model, metrics

digits = datasets.load_digits()
X = digits.data
y = digits.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.4, random_state=1)

reg = linear_model.logisticRegression(max_iter=10000,
random_state=0)
reg.fit(X_train, y_train)

y_pred = reg.predict(X_test)

print(f"logistic Regression model accuracy:
{metrics.accuracy_score(y_test, y_pred) * 100:.2f}%")
```

Output:

```
logistic Regression model accuracy: 96.66%
```

4. Implementation of Gaussian Naive Bayes

```
#Implementation of Gaussian Naive Bayes

import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import labelEncoder

iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['Species'] = iris.target
X = data.drop("Species", axis=1)
y = data['Species']
# Encoding the Species column to get numerical class
le = labelEncoder()
y = le.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
# Gaussian Naive Bayes classifier
gnb = GaussianNB()

# Train the classifier on the training data
gnb.fit(X_train, y_train)
# Make predictions on the testing data
y_pred = gnb.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"The Accuracy of Prediction on Iris Flower is: {accuracy}")
```

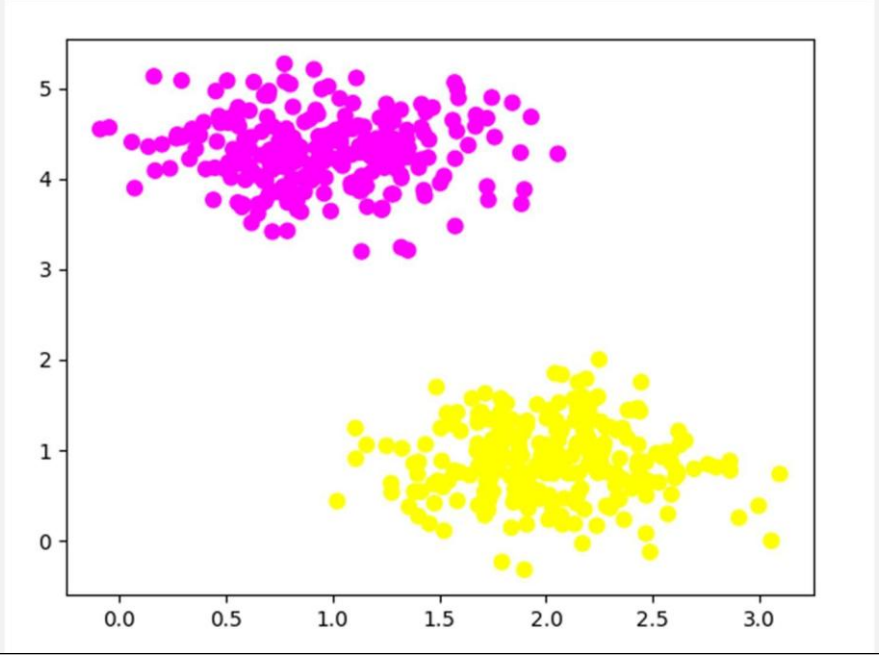
Output:

```
The Accuracy of Prediction on Iris Flower is: 0.9777777777
```

5. Predictive Model using Support Vector Machine

```
#Predictive Model using Support Vector Machine

# importing scikit learn with make_blobs
from sklearn.datasets import make_blobs
# creating datasets X containing n_samples
# Y containing two classes
X, Y = make_blobs(n_samples=500, centers=2, random_state=0,
cluster_std=0.40)
import matplotlib.pyplot as plt
# plotting scatters
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap="spring");
plt.show()
```

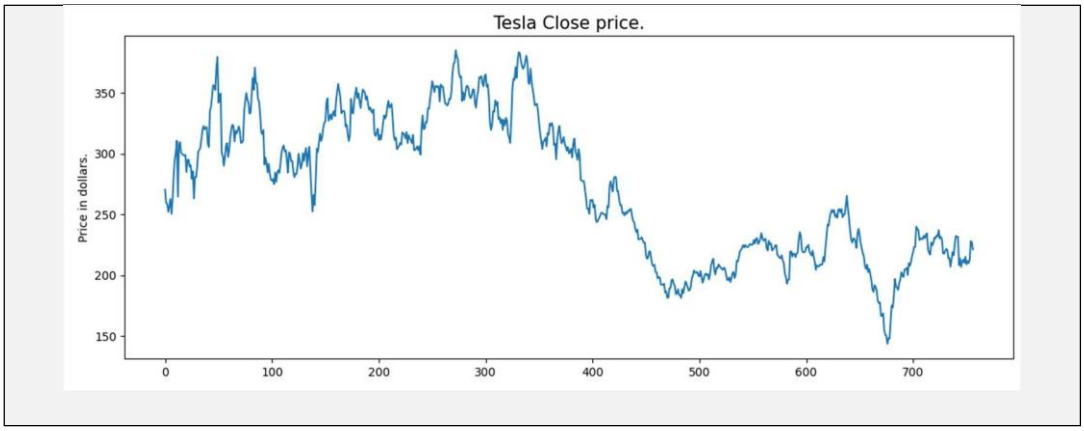
Output:



6. Stock Price Prediction using Machine Learning

```
#Stock Price Prediction using Machine learning in Python

import pandas as pd
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings("ignore")
df = pd.read_csv('tesla.csv')
df.head()
df.shape
df.describe()
df.info()
plt.figure(figsize=(15,5))
plt.plot(df['close'])
plt.title('Tesla Close price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()
```

Output:


Tesla Close price.
Price in dollars.
350
300
250
200
150
0
100
200
300
400
500
600
700

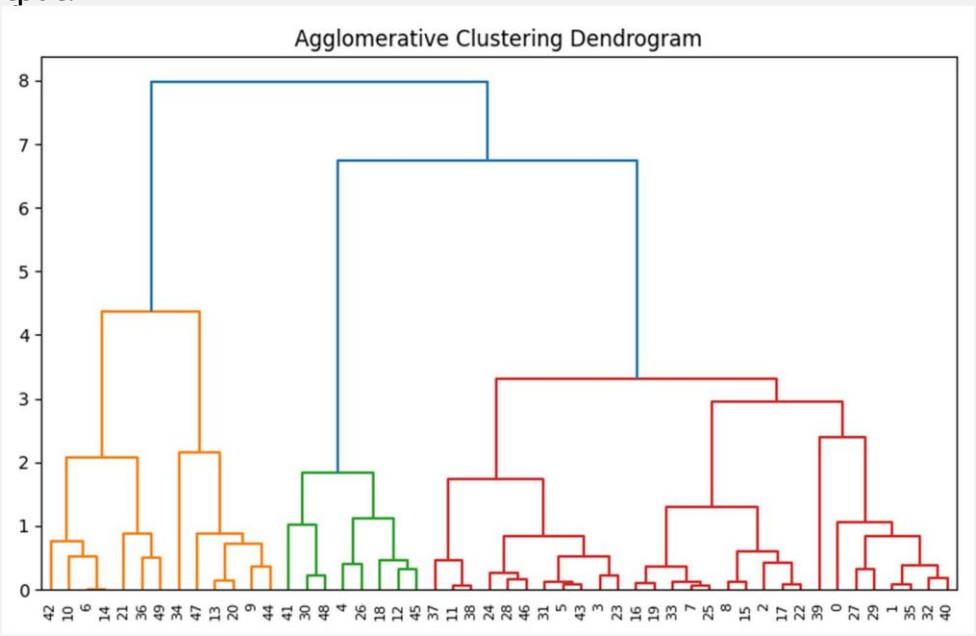
Unit – 3

Unsupervised Learning

7. Agglomerative Clustering Example

```
1  #Agglomerative Clustering Example
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from scipy.cluster.hierarchy import dendrogram, linkage
6
7  data = np.random.randn(50, 2)
8
9  Z = linkage(data, "ward")
10
11 # Plot dendrogram
12 plt.figure(figsize=(10, 7))
13 dendrogram(Z)
14 plt.title("Agglomerative Clustering Dendrogram")
15 plt.show()
```

Output:

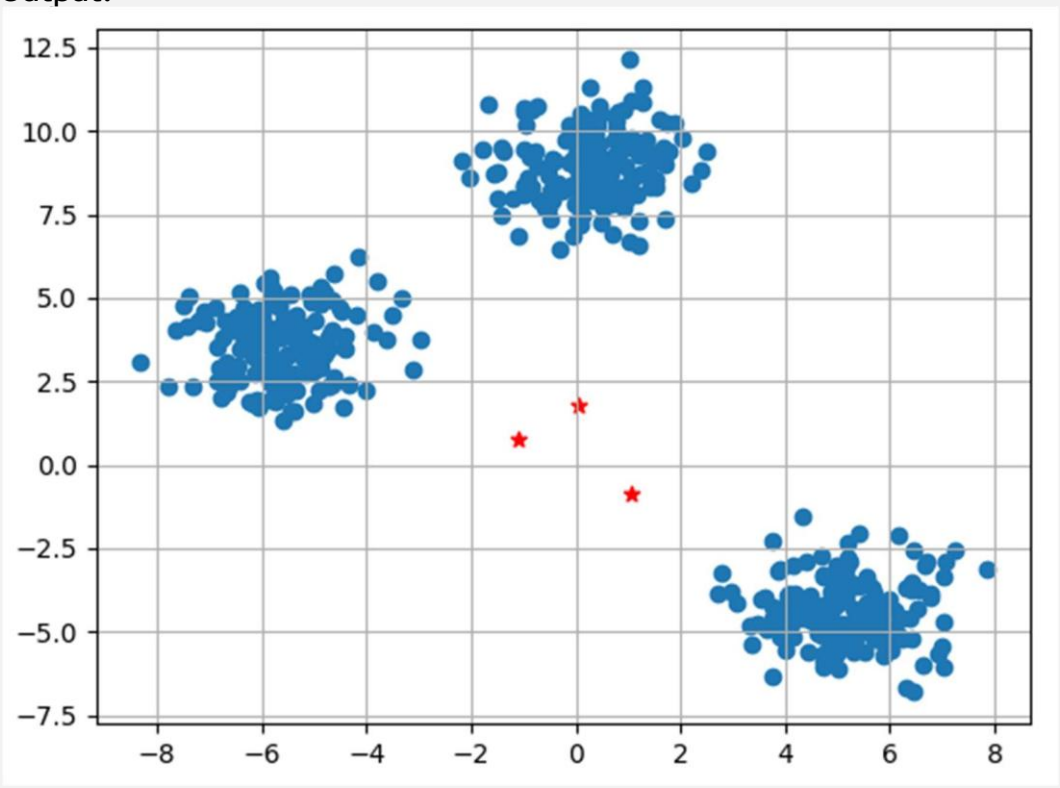


8. K means clustering Example

```
1  #K means clustering Example
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from sklearn.datasets import make_blobs
6
7  X,y = make_blobs(n_samples = 500,n_features = 2,centers =
8  3,random_state = 23)
9
10 # fig = plt.figure(0)
11 # plt.grid(True)
12 # plt.scatter(X[:,0],X[:,1])
13 # plt.show()
14
15 k = 3
16 clusters = {}
17 np.random.seed(23)
18
19 for idx in range(k):
20     center = 2 * (2 * np.random.random((X.shape[1],)) - 1)
```

```
21      points = []
22      cluster = {
23          'center': center,
24          'points': []
25      }
26      clusters[idx] = cluster
27  clusters
28
29  plt.scatter(X[:,0],X[:,1])
30  plt.grid(True)
31  for i in clusters:
32      center = clusters[i]['center']
33      plt.scatter(center[0],center[1],marker = '*',c = 'red')
34  plt.show()
```

Output:



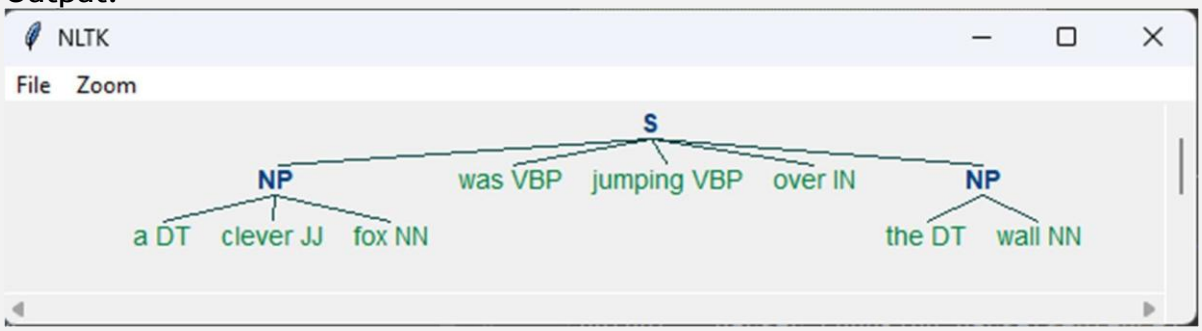
Unit – 4

Natural Language Processing

9. Implement Noun-Phrase chunking

```
#Implement Noun-Phrase chunking

import nltk
sentence = [("a", "DT"),("clever","JJ"),("fox"," NN"),("was","VBP"),
    ("jumping","VBP"),("over","IN"),("the","DT"),("wall","NN")]
grammar = "NP:{<DT>?<JJ>*< NN>}"
parser_chunking = nltk.RegexpParser(grammar)
parser_chunking.parse(sentence)
Output = parser_chunking.parse(sentence)
Output.draw()
```

Output:



10. Implementation of Snowball Stemmer

```
#implementation of Snowball Stemmer

import nltk
from nltk.stem.snowball import SnowballStemmer

# the stemmer requires a language parameter
snow_stemmer = SnowballStemmer(language="english")

# list of tokenized words
words = ['cared', 'university', 'fairly', 'easily', 'singing',
         'sings', 'sung', 'singer', 'sportingly']

# stem"s of each word
stem_words = []
for w in words:
    x = snow_stemmer.stem(w)
    stem_words.append(x)

# print stemming results
for e1, e2 in zip(words, stem_words):
    print(e1 + " ----> " + e2)
```

Output:

```
cared ----> care
university ----> univers
fairly ----> fair
easily ----> easili
singing ----> sing
sings ----> sing
sung ----> sung
singer ----> singer
sportingly ----> sport
```

11. Implementation of Porter Stemmer

```
1  #Implementation of Porter Stemmer
2
3  from nltk.stem import PorterStemmer
4
5  # Create a Porter Stemmer instance
6  porter_stemmer = PorterStemmer()
7
8  # Example words for stemming
9  words = ["running", "jumps", "happily", "running", "happily"]
10
11 # Apply stemming to each word
12 stemmed_words = [porter_stemmer.stem(word) for word in words]
13
14 # Print the results
15 print("Original words:", words)
16 print("Stemmed words:", stemmed_words)
```

Output:

```
Original words: ['running', 'jumps', 'happily', 'running', 'happily']
Stemmed words: ['run', 'jump', 'happili', 'run', 'happili']
```

12. Text classifier example

```
1  #Text classifier example
2
3  # import regex
4  import re
5
   # input string
   string = "    Python 3.0, released in 2008, was a major revision of
   the language that is not completely backward compatible and much
   Python 2 code does not run unmodified on Python 3. With Python 2's
   end-of-life, only Python 3.6.x[30] and later are supported, with
   older versions still supporting e.g. Windows 7 (and old installers
   not restricted to 64-bit Windows)."
6
7  # convert to lower case
8  lower_string = string.lower()
9
10 # remove numbers
11 no_number_string = re.sub(r"\d+","",lower_string)
12 print(no_number_string)
13
14 #---------------- Remove punctuations ------------------------------
15
16 lower_string = string.lower()
17
18 # remove numbers
19 no_number_string = re.sub(r'\d+', '', lower_string)
20
21 # remove all punctuation except words and space
22 no_punc_string = re.sub(r"[^\w\s]", '', no_number_string)
23 print(no_punc_string)
```

Output:

```
python ., released in , was a major revision of the language that is
not completely backward compatible and much python code does not run
unmodified on python . with python 's end-of-life, only python ..x[]
and later are supported, with older versions still supporting e.g.
windows (and old installers not restricted to -bit windows).
```

python released in was a major revision of the language that is not completely backward compatible and much python code does not run unmodified on python with python s endoflife only python x and later are supported with older versions still supporting eg windows and old installers not restricted to bit windows

Unit – 5

Computer Vision with OpenCV

13. Test OpenCV installation

```
#Test  OpenCV  installation

import cv2

# Check OpenCV version
print("OpenCV version:", cv2.__version__)

# Simple test to see if the library loads correctly
img = cv2.imread("f.jpg")     # Replace with an actual image path
if img is None:
    print("Image not loaded correctly!")
else:
    print("OpenCV is installed and working correctly!")
```

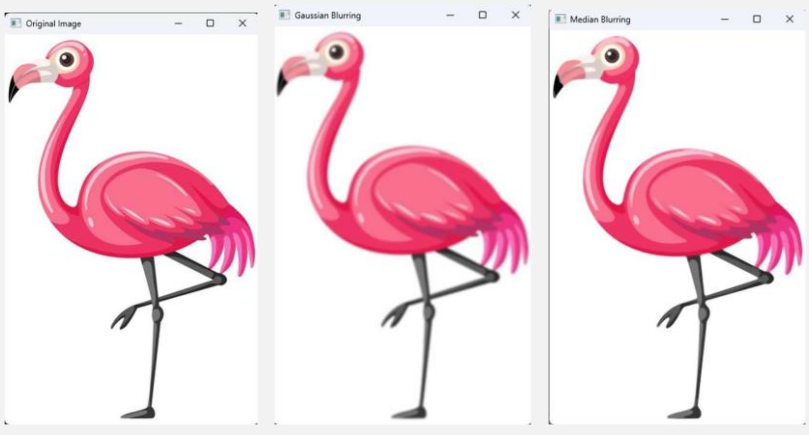
Output:

```
OpenCV version: 4.11.0
OpenCV is installed and working correctly!
```

14. Blur Image

```
#Blur Image

import cv2
import numpy as np

image   =   cv2.imread("f.jpg")
cv2.imshow("Original  Image",  image)
cv2.waitKey(0)

# Gaussian Blur
Gaussian = cv2.GaussianBlur(image, (7, 7), 0)
cv2.imshow("Gaussian Blurring', Gaussian)
cv2.waitKey(0)

# Median Blur
median = cv2.medianBlur(image, 5)
cv2.imshow("Median Blurring', median)
cv2.waitKey(0)

# Bilateral Blur
bilateral = cv2.bilateralFilter(image, 9, 75, 75)
cv2.imshow("Bilateral Blurring', bilateral)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

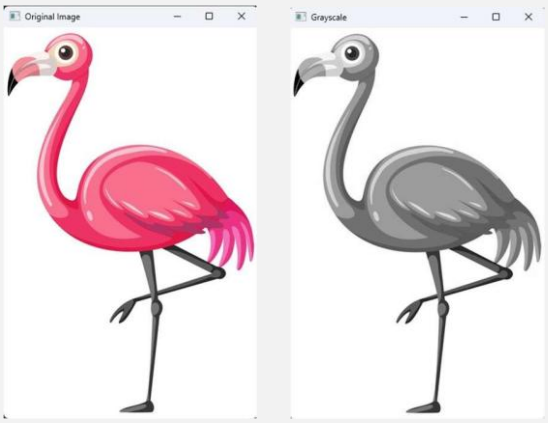
Output:



15. Grayscale image

```
1  #Grayscale image
2
3  import cv2
4
5  # load the input image
6  image = cv2.imread("f.jpg")
7  cv2.imshow("Original", image)
8  cv2.waitKey(0)
9
10 # Use the cvtColor() function to grayscale the image
11 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
12
13 cv2.imshow("Grayscale", gray_image)
14 cv2.waitKey(0)
15
16 # Window shown waits for any key pressing event
17 cv2.destroyAllWindows()
```

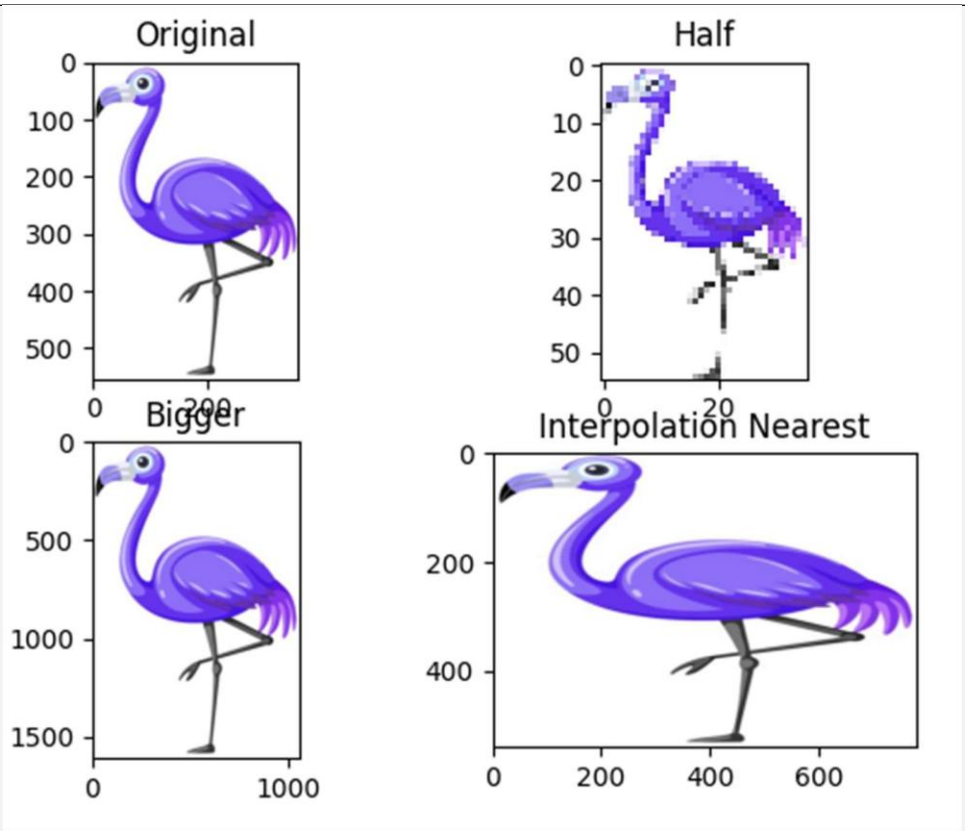
Output:



16. Resize image

```
1  #Resize image
2
3  import cv2
4  import matplotlib.pyplot as plt
5
6  image = cv2.imread("f.jpg", 1)
7
8  half = cv2.resize(image, (0, 0), fx = 0.1, fy = 0.1)
9  bigger = cv2.resize(image, (1050, 1610))
10 stretch_near = cv2.resize(image, (780, 540), interpolation =
11 cv2.INTER_LINEAR)
12
13 Titles =["Original", "Half", "Bigger", "Interpolation Nearest"]
14 images =[image, half, bigger, stretch_near]
15 count = 4
16
17 for i in range(count):
18     plt.subplot(2, 2, i + 1)
19     plt.title(Titles[i])
20     plt.imshow(images[i])
21
22 plt.show()
```

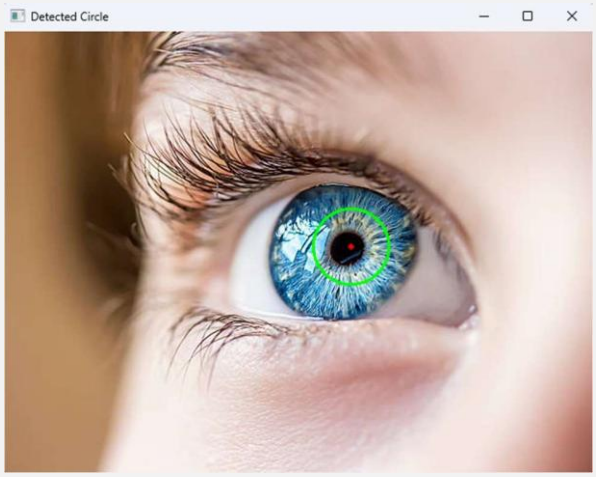
Output:



17. Detect pupil

```
#Detect pupil

import cv2
import numpy as np

# Read image.
img = cv2.imread("eye.jpg", cv2.IMREAD_COLOR)

# Convert to grayscale.
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Blur using 3 * 3 kernel.
gray_blurred = cv2.blur(gray, (3, 3))

# Apply Hough transform on the blurred image.
detected_circles = cv2.HoughCircles(gray_blurred,
                                    cv2.HOUGH_GRADIENT, 1, 20, param1 = 50,
                                param2 = 30, minRadius = 1, maxRadius = 40)

# Draw circles that are detected.
if detected_circles is not None:

        # Convert the circle parameters a, b and r to integers.
        detected_circles = np.uint16(np.around(detected_circles))

        for pt in detected_circles[0, :]:
                a, b, r = pt[0], pt[1], pt[2]
                 # Draw the circumference of the circle.
                cv2.circle(img, (a, b), r, (0, 255, 0), 2)
                # Draw a small circle (of radius 1) to show the center.
                cv2.circle(img, (a, b), 1, (0, 0, 255), 3)
                cv2.imshow("Detected Circle", img)
                cv2.waitKey(0)
```

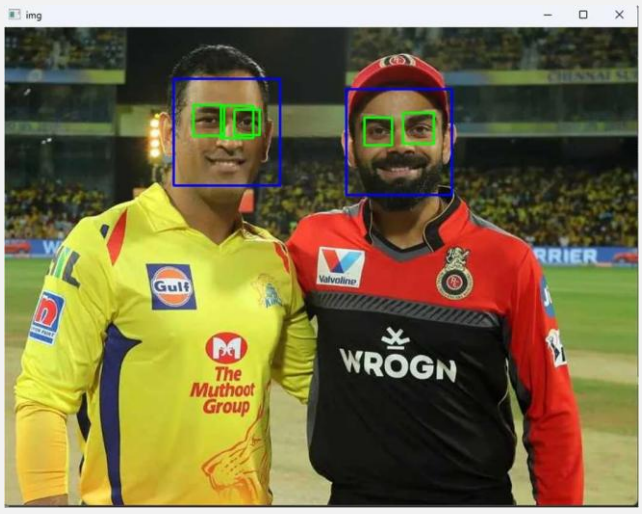
Output:



18. Detect face and eyes

```
1  #Detect face and eyes
2  import numpy as np
3  import cv2
4
5  face_cascade                                                    =
6  cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
7  eye_cascade      =      cv2.CascadeClassifier("haarcascade_eye.xml")
8
9  img   =   cv2.imread("img_3.png")
10 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
11 faces = face_cascade.detectMultiScale(gray, 1.3, 5)
12 for (x,y,w,h) in faces:
13     img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
14     roi_gray = gray[y:y+h, x:x+w]
15     roi_color = img[y:y+h, x:x+w]
16     eyes = eye_cascade.detectMultiScale(roi_gray)
17     for (ex,ey,ew,eh) in eyes:
18         cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
19
20 cv2.imshow("img",img)
21 cv2.waitKey(0)
   cv2.destroyAllWindows()
```

Output:



19. Play video using OpenCV

```
#Play video using OpenCV

import cv2
from ffpyplayer.player import MediaPlayer

file="sample.mp4"
video=cv2.VideoCapture(file)
player = MediaPlayer(file)
while True:
    ret, frame=video.read()
    audio_frame, val = player.get_frame()
    if not ret:
        print("End of video")
        break
    if cv2.waitKey(1) == ord("q"):
        break
    cv2.imshow("Video", frame)
    if val != 'eof' and audio_frame is not None:
        #audio
        img, t = audio_frame
video.release()
cv2.destroyAllWindows()
```

Output:

20. Detect face from Webcam

```
#Detect face from Webcam
import cv2
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +"haarcascade_frontalface_default.xml")
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +"haarcascade_eye.xml")
smile_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +"haarcascade_smile.xml")

def detect(gray, frame):
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y), ((x + w), (y + h)), (255, 0, 0), 2)
                roi_gray = gray[y:y + h, x:x + w]
                roi_color = frame[y:y + h, x:x + w]
                 smiles = smile_cascade.detectMultiScale(roi_gray, 1.8, 20)

                for (sx, sy, sw, sh) in smiles:
                        cv2.rectangle(roi_color, (sx, sy), ((sx + sw), (sy + sh)), (0, 0, 255), 2)
        return frame


video_capture = cv2.VideoCapture(0)
while video_capture.isOpened():
    # Captures video_capture frame by frame
    _, frame = video_capture.read()

    # To capture image in monochrome
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # calls the detect() function
    canvas = detect(gray, frame)
```

```
30
31      # Displays the result on camera feed
32      cv2.imshow("Video",  canvas)
33
34      # The control breaks once q key is pressed
35      if cv2.waitKey(1) fi 0xff == ord('q'):
36          break
37
38  # Release the capture once all the processing is done.
39  video_capture.release()
40  cv2.destroyAllWindows()
```

Output: