GLOBAL ACADEMY OF TECHNOLOGY

Department of Artificial Intelligence and Machine Learning

Affiliated to VTU, Accredited by NAAC with 'A' grade

RR Nagar, Bengaluru – 560 098

AI TOOLS, FRAMEWORKS AND ITS APPLICATIONS I

MANUAL

IV Semester

Course Code: AML23407A

[As per the Autonomous Scheme]

Scheme: 2023

Version 1.0 w.e.f 3RD Febrauary, 2025

Editorial Committee, Faculty, Dept. of AI & ML

Approved by

Dr. Roopa B S

Document Log

Name of the Document	AI Tools, Frameworks and its Application I Manual
Scheme	2023
Current Version and Date	Version 2.0, 23.01.2025
Subject Code	23AML407A
Editorial Committee	Prof. Vani
Computer Programmer	--
Approved by	HOD, DEPT. OF AI & ML

Table of Contents

Sl. No.	Particulars		Page No.
1	**Vision and Mission of the Department**		
2	**PEOs, PSOs, Pos**		
3	• **Course Outcomes** • **Syllabus** • **CO-PO-PSO Mapping**		
4	**Lab Evaluation Process**		
5	**Rubrics**		
6	**Evaluation sheet**		
7		**Laboratory Experiments**	
		PART - A	
	Program 1	Simulate to measure the light intensity using Thinkercad	
	Program 2	Simulate an ultrasonic distance sensor to detect the distance from the object	
	Program 3	Simulate a DC motor with specified input that runs continuously and vary with the DC motor speed.	
	Program 4	Simulate the door sensor for smart homes using IoT sensor	
	Program 6	Installation of NLTK Library and working with Basic commands	
	Program 7	Write a python code for Tokenizing and count word frequency.	
	Program 8	Write a python code NLTK word stemming and Lemmatizing words using WordNet.	
		PART - B	
	1	**AI in Agriculture** **Smart Irrigation System** Build a smart irrigation system using Arduino or Raspberry Pi that incorporates soil moisture sensors and weather data. They will develop an AI algorithm to optimize irrigation based on real-time conditions.	
	2	**AI in Smart Cities** **Smart Street Lighting with Iot** Design a smart street lighting system using Ardino or Raspberry Pi and IoT sensors to detect light levels and motion. They will develop an AI algorithm to optimize street lighting based on real-time conditions.	

	3	**AI in Education** **Recommendation System for Online Courses** Create a basic recommendation system using collaborative filtering with TensorFlow. Use it to recommend online courses to students based on their previous choices.	
	4	**AI in Healthcare** **Predicting Disease Outcomes from Patient Data** Build a predictive model using TensorFlow to estimate the progression of a specific disease (e.g., diabetes) based on patient data such as age, BMI, and blood sugar levels.	

Vision of the Department

To provide progressive education and create transformative professionals and leaders to harness the power of technology and make a positive impact on the society.

Mission of the Department

M1 - Quality Education: To adopt a student centric curriculum delivery process with emphasis on problem solving and programming skills.

M2 - Innovation: To collaborate with industries and professional bodies and make the students industry ready. To drive innovation through multi-disciplinary research and development activities.

M3 - Skill Development & Ethics: To endorse additional skill development through student forums and experiential learning. Inculcate human values for a smarter and ethical world.

Program Educational Objectives (PEOs) of Department

Graduates in Artificial Intelligence and Data Science will be able to:

PEO1. Able to practice and implement their success skills like problem solving, communication and collaboration for providing innovative engineering solutions.

PEO2. Contribute their AI & ML expertise grounded in computer science as members and leaders of professional engineering teams in multidisciplinary applications.

PEO3. Demonstrate lifelong learning through continued professional development and higher education in top graduate technical, research and management programs.

PEO4. Demonstrate a commitment to society by applying the skills and knowledge for a smarter and ethical world.

Program Specific Outcomes (PSOs)

After successful completion of Artificial Intelligence and Machine Learning program

PSO1. Graduates will be Proficient in programming and problem-solving skills for developing, managing software and distributed systems.

PSO2. Graduates will be able to identify, formulate, predict and solve real world problems by applying principles of Artificial Intelligence & Machine learning.

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

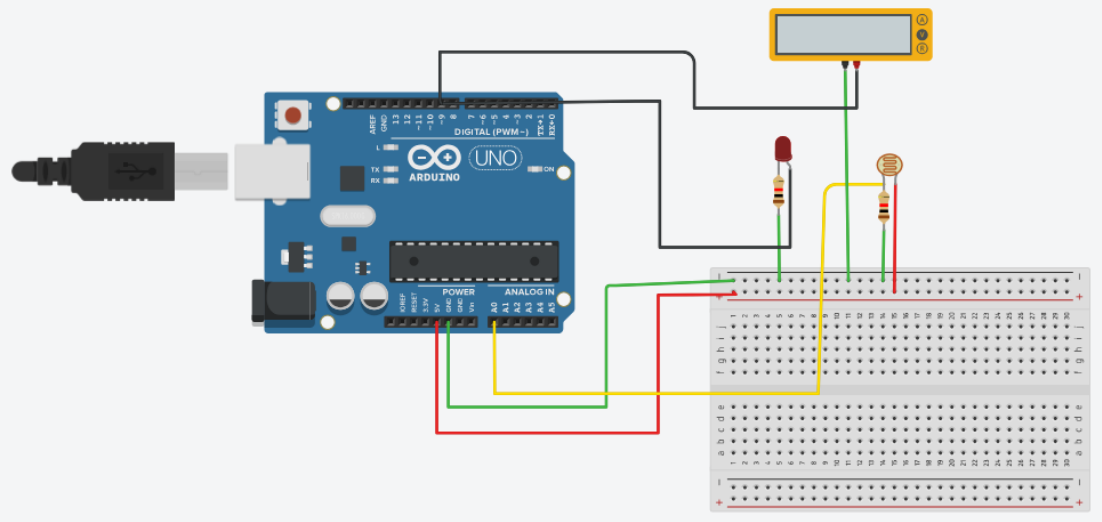
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

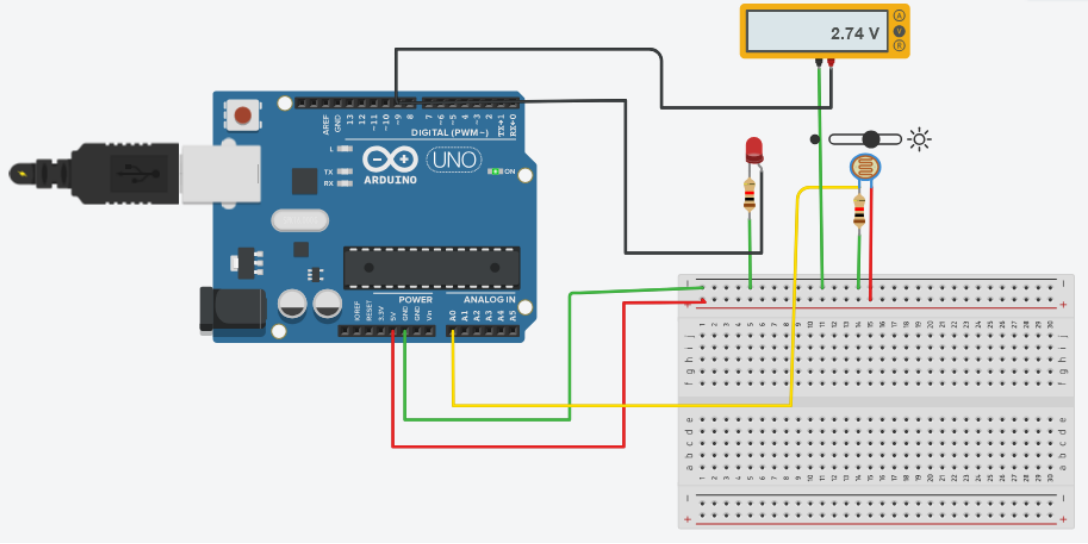
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Sl.NO	Experiments
1	Simulate to measure the light intensity using Thinkercad
2	Simulate an ultrasonic distance sensor to detect the distance from the object
3	Simulate a DC motor with specified input that runs continuously and vary with the DC motor speed.
4	Simulate the door sensor for smart homes using IoT sensor
5	Installation of NLTK Library and working with Basic commands
6	Write a python code for Tokenizing and count word frequency.
7	Write a python code NLTK word stemming and Lemmatizing words using WordNet.
9	**AI in Agriculture** **Smart Irrigation System** Build a smart irrigation system using Arduino or Raspberry Pi that incorporates soil moisture sensors and weather data. They will develop an AI algorithm to optimize irrigation based on real-time conditions.
10	**AI in Smart Cities** **Smart Street Lighting with Iot** Design a smart street lighting system using Ardino or Build a predictive model using TensorFlow to estimate the progression of a specific disease (e.g., diabetes) based on patient data such as age, BMI, and blood sugar levels.
11	**AI in Education** **Recommendation System for Online Courses** Create a basic recommendation system using collaborative filtering with TensorFlow. Use it to recommend online courses to students based on their previous choices.
12	**AI in Healthcare** **Predicting Disease Outcomes from Patient Data** Build a predictive model using TensorFlow to estimate the progression of a specific disease (e.g., diabetes) based on patient data such as age, BMI, and blood sugar levels.

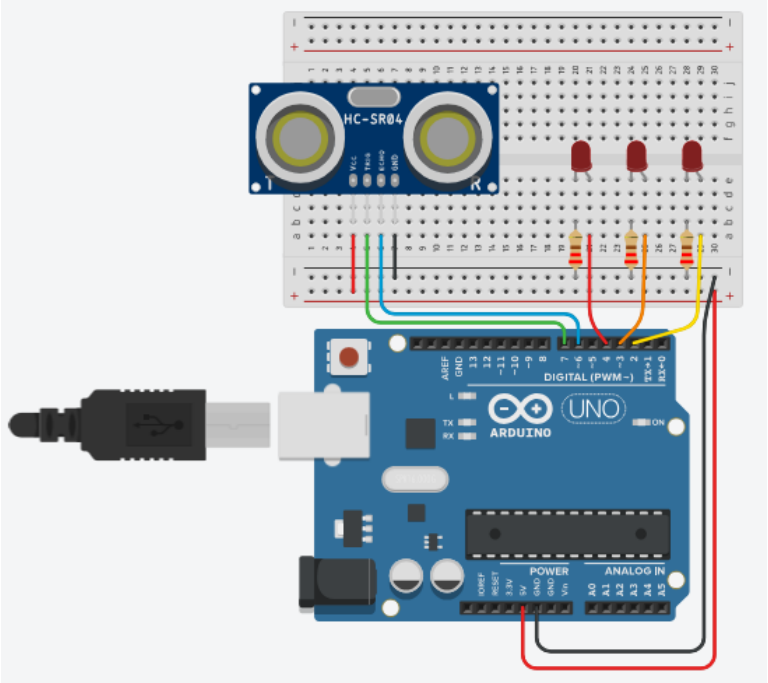
1. Simulate to measure the light intensity

```
int sensorValue = 0;
void setup()
{
pinMode(A0, INPUT);
Serial.begin(9600);
pinMode(9, OUTPUT);
}
void loop()
{
sensorValue = analogRead(A0);
Serial.println(sensorValue);
analogWrite(9, map(sensorValue,0,1023,0,255));
delay(1000);
}
```



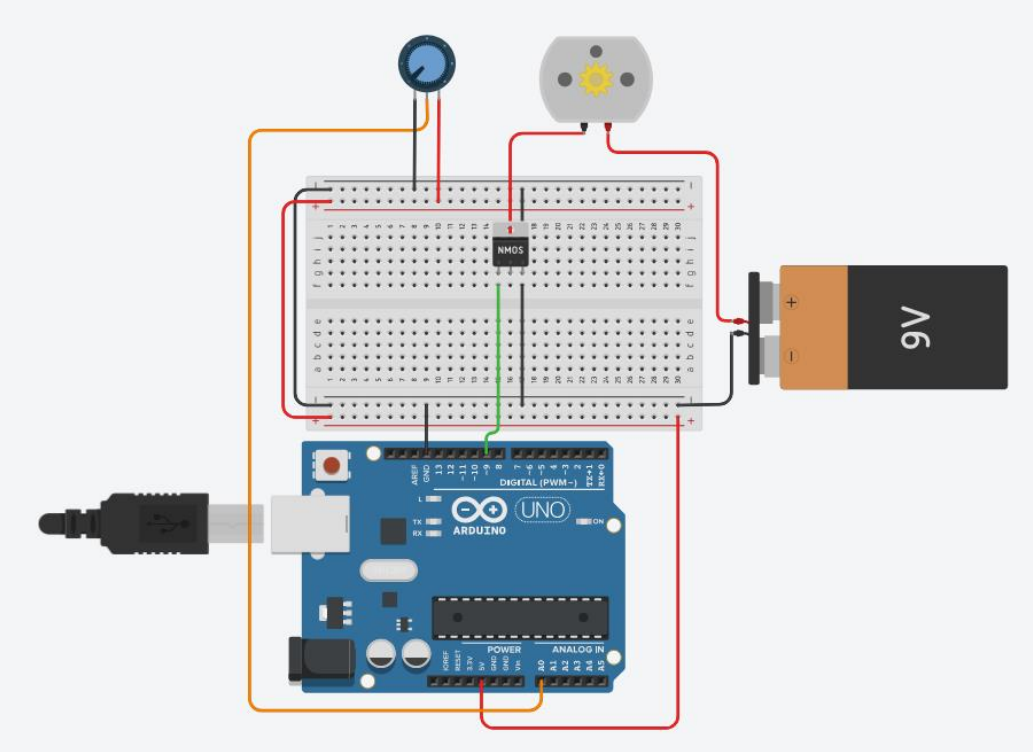


2. Simulate an ultrasonic distance sensor



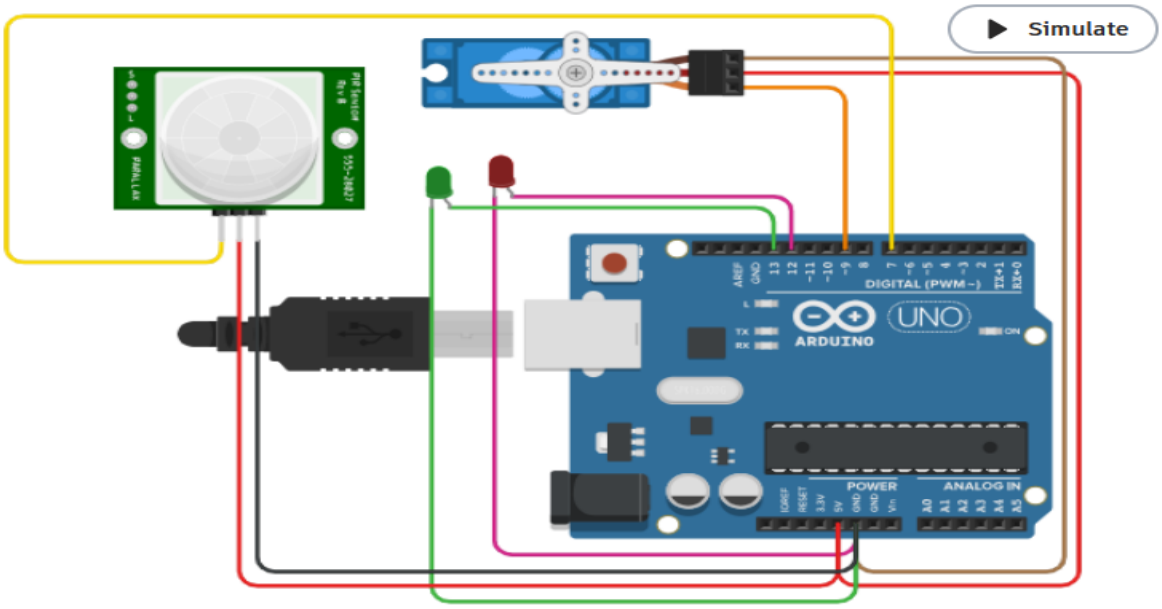

```
const int triggerPin = 9;  // Arduino digital pin connected to the trigger pin of the simulated sensor
const int echoPin = 10;    // Arduino digital pin connected to the echo pin of the simulated sensor

void setup() {
  Serial.begin(9600);
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Simulate the ultrasonic sensor by generating a distance value
  int simulatedDistance = generateSimulatedDistance();

  // Print the simulated distance value
  Serial.print("Simulated Distance: ");
  Serial.print(simulatedDistance);
  Serial.println(" cm");

  delay(1000);  // Wait for a moment before the next measurement
}

int generateSimulatedDistance() {
  // Replace this function with your desired simulation logic
  // In this example, a simple sine function is used
  // You can modify this function to simulate different behaviors
  return 50 + 50 * sin(millis() / 1000.0);  // Simulates a distance oscillating between 0 and 100 cm
      }
```

3. Simulate a DC motor with specified input that runs continuously and vary with the DC motor speed




```
int MPin = 9;   //motor pin
int Analog = A0;  // Analog Input
int Ainp = 0; // Read analog input, initialized to zero
void setup()
{
  Serial.begin(9600);
  pinMode(MPin, OUTPUT);
  pinMode(Analog, INPUT);
}
void loop()
{
      Ainp = analogRead(A0);
      int Speed = map(Ainp, 0, 1024, 0, 255);
      delay(100);
      Serial.println(Speed);
      analogWrite(MPin, Speed);
}
```

4. Simulate door sensor for smart homes using IoT sensor.




```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards
int poson = 1  ;   // on position(!!YOU WILL HAVE TO SET IT BY TRY AND ERROR ACCORDING TO YOUR SETUP!!)
int posoff =  100 ;  //off position(!!YOU WILL HAVE TO SET IT BY TRY AND ERROR ACCORDING TO YOUR SETUP!!)
int pirPin = 7;    //the digital pin connected to the PIR sensor's output
int ledPin = 13;
int led2Pin=12;

void setup() {
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(led2Pin,OUTPUT);
  digitalWrite(pirPin, LOW);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
 // delay(30000); //enough time for the sensor to calibrate
}

void loop() {
 if(  digitalRead(pirPin) == HIGH        ) {

    digitalWrite(ledPin, HIGH);
    digitalWrite(led2Pin,LOW);
    myservo.write(45);              // tell servo to go to position in variable 'poson'
    delay(5000);
    myservo.write(90);              // tell servo to go to position in variable 'posoff'
    digitalWrite(ledPin, LOW);
```

```
  digitalWrite(led2Pin,HIGH);
  delay(1000);   //delay so it wont start right again imidately
  digitalWrite(led2Pin,LOW);

}
}
```

5. Installation of NLTK Library and working with basic commands.

6. Write a python code for Tokenizing and count word frequency.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
import string

# Download NLTK resources (uncomment the following two lines if not already downloaded)
# nltk.download('punkt')
# nltk.download('stopwords')

def tokenize_and_count_frequency(text):
    # Tokenize the text into words
    words = word_tokenize(text)

    # Remove punctuation and convert to lowercase
    words = [word.lower() for word in words if word.isalpha()]

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words]

    # Calculate word frequencies
    frequency_dist = FreqDist(words)

    return frequency_dist

if __name__ == "__main__":
    # Sample text for testing
    sample_text = "This is a sample text. It is used for testing the NLTK word frequency counting functionality."

    # Tokenize and count word frequency
    word_freq = tokenize_and_count_frequency(sample_text)

    # Print the word frequencies
    print("Word frequencies:")
    for word, frequency in word_freq.items():
        print(f"{word}: {frequency}")
```

7. Write a python code NLTK word stemming and Lemmatizing words using WordNet.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import stopwords

# Download NLTK resources (uncomment the following two lines if not already downloaded)
# nltk.download('punkt')
# nltk.download('wordnet')
# nltk.download('stopwords')

def preprocess_text(text):
    # Tokenize the text into words
    words = word_tokenize(text)

    # Remove punctuation and convert to lowercase
    words = [word.lower() for word in words if word.isalpha()]

    # Remove stop words
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words]

    return words

def stem_text(words):
    # Initialize the Porter Stemmer
    stemmer = PorterStemmer()

    # Apply stemming to each word in the list
    stemmed_words = [stemmer.stem(word) for word in words]

    return stemmed_words

def lemmatize_text(words):
    # Initialize the WordNet Lemmatizer
    lemmatizer = WordNetLemmatizer()

    # Apply lemmatization to each word in the list
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]

    return lemmatized_words

if __name__ == "__main__":
    # Sample text for testing
    sample_text = "The quick brown foxes are jumping over the lazy dogs."

    # Preprocess the text
    processed_words = preprocess_text(sample_text)

    # Stem the words
    stemmed_words = stem_text(processed_words)

    # Lemmatize the words
```

```
lemmatized_words = lemmatize_text(processed_words)

# Print the results
print("Original Words:", processed_words)
print("Stemmed Words:", stemmed_words)
print("Lemmatized Words:", lemmatized_words)
```

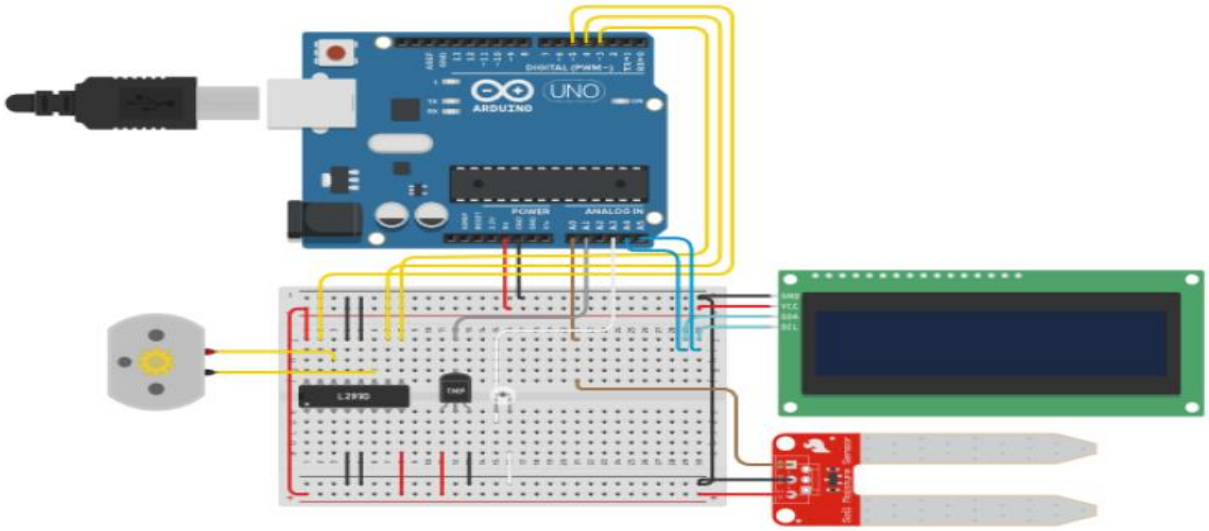
In this script:

- The `preprocess_text` function tokenizes the text, converts words to lowercase, and removes punctuation and stop words.
- The `stem_text` function uses the Porter Stemmer for stemming the words.
- The `lemmatize_text` function uses the WordNet Lemmatizer for lemmatizing the words.

9. AI in Agriculture

Smart Irrigation System

Build a smart irrigation system using Arduino or Raspberry Pi that incorporates soil moisture sensors and weather data. They will develop an AI algorithm to optimize irrigation based on real-time conditions.



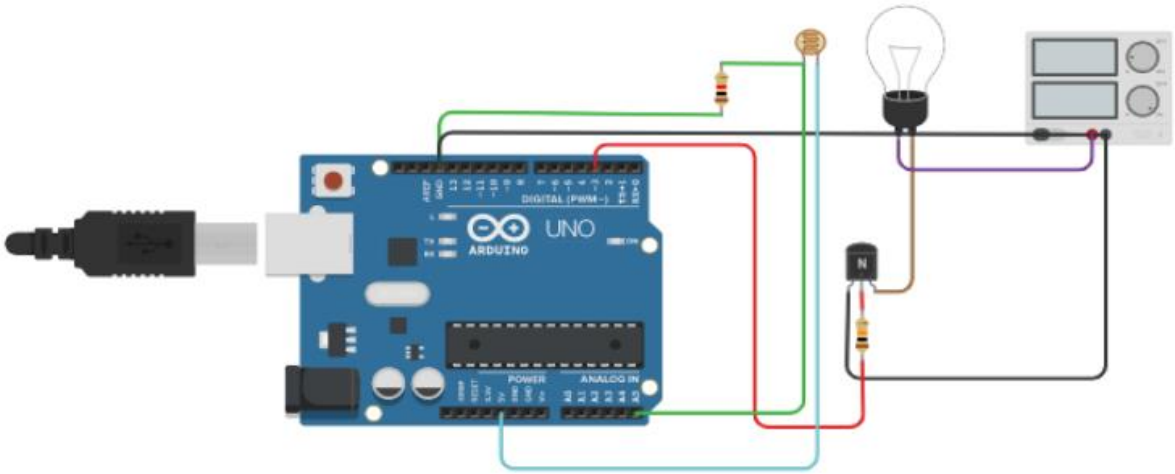

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Define the pins for the motor control
int enA = 5;  // Enable pin for the L293D motor driver
int in1 = 3;  // Input pin 1 for the L293D motor driver
int in2 = 4;  // Input pin 2 for the L293D motor driver

LiquidCrystal_I2C lcd(32,16,2);

// Define the pins for the sensors
int tempPin = A1;     // Temperature sensor is connected to A1
```

```
int lightPin = A2;    // Light sensor is connected to A2
int moisturePin = A0; // Soil moisture sensor is connected to A0

void setup() {
 // Initialize the Serial Monitor
            lcd.begin(16,2);
            lcd.init();
            lcd.backlight();
            Serial.begin(9600);


  // Set the motor control pins as outputs
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
}

void loop() {
  // Read the sensor values
  int tempValue = analogRead(tempPin);
  int lightValue = analogRead(lightPin);
  int moistureValue = analogRead(moisturePin);

  // Convert the sensor values to their corresponding units
  float temp = (((tempValue * 5.0) / 1024.0)-0.5)*100;
  float light = (lightValue * 5.0) / 1024.0;
  float moisture = (moistureValue * 5.0) / 1024.0;


  Serial.print("Temp ");
  Serial.print(temp);
  Serial.println(" C");
  Serial.print("Light:");
  Serial.print(light);
  Serial.println("V");
  Serial.print("Moist: ");
  Serial.print(moisture);
  Serial.println("V");

   // Display the sensor values on the LCD screen
  lcd.clear(); // Clear the LCD screen
  lcd.setCursor(0, 0); // Set the cursor to the top-left corner
  lcd.print("Temp:"); // Print the temperature label
  lcd.print(temp); // Print the temperature value
  lcd.print("C"); // Print the temperature unit
  lcd.setCursor(0, 1); // Set the cursor to the second row
  lcd.print("Light:"); // Print the light label
  lcd.print(light); // Print the light value
  lcd.print("V"); // Print the light unit
  lcd.setCursor(8, 1); // Set the cursor to the second row, ninth column
  lcd.print("Moist:"); // Print the moisture label
  lcd.print(moisture); // Print the moisture value
  lcd.print("V"); // Print the moisture unit

  // Control the motor based on the sensor readings
  if((temp > 30 || light < 2)&& moisture < 4 ) {
```

```
    // If it's hot, the soil is dry, and it's morning, turn on the motor
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    analogWrite(enA, 255); // Set the motor speed to maximum
    Serial.println("Turning on the motor...");
  } else {
    // Otherwise, turn off the motor
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    analogWrite(enA, 0); // Set the motor speed to 0
    Serial.println("Turning off the motor...");
  }

  // Wait for 1 second before taking another reading
  delay(1000);
}
```

10. Smart Street Lighting with IoT

Design a smart street lighting system using Ardino or Raspberry Pi and IoT sensors to detect light levels and motion. They will develop an AI algorithm to optimize street lighting based on real-time conditions.




```
int ldr_pin = A5;
  int ldr_value;

  int light = 3;

  void setup()
  {
    pinMode(light, OUTPUT); pinMode(ldr_pin, INPUT);
  }
  void loop()
```

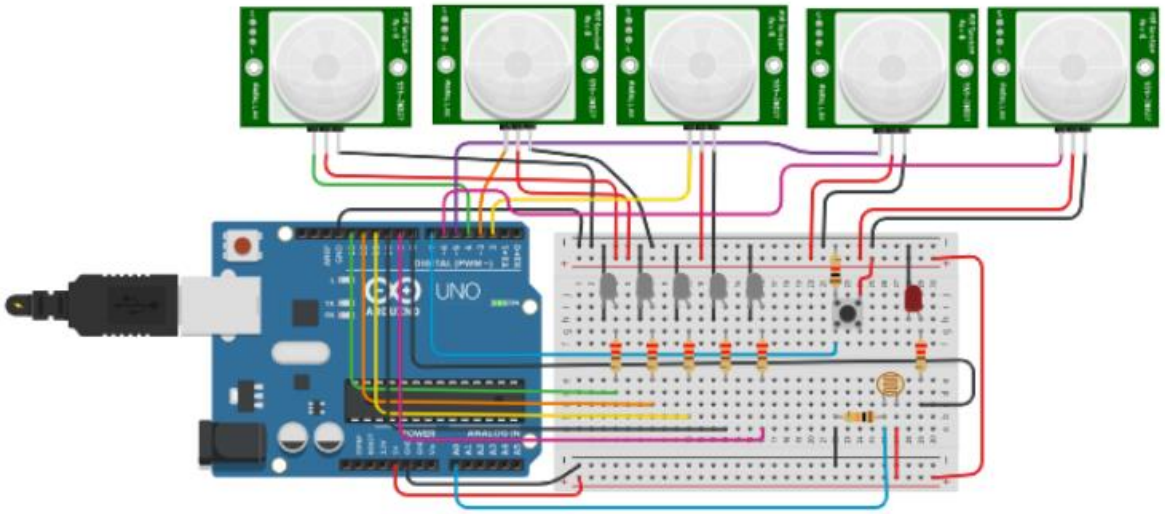
```
  {
    ldr_value = analogRead(ldr_pin);
    if (ldr_value > 512)
      digitalWrite(light, 0);
    else
      digitalWrite(light, 1);

  }
```

2nd

```
void setup()
{
  pinMode(4, INPUT);
  pinMode(13, OUTPUT);
   pinMode(3, INPUT);
  pinMode(12, OUTPUT);
   pinMode(2, INPUT);
   pinMode(7, INPUT);
  pinMode(11, OUTPUT);
  pinMode(5, INPUT);
  pinMode(10, OUTPUT);
  pinMode(6, INPUT);
  pinMode(9, OUTPUT);
  pinMode(A0, INPUT);
   pinMode(8, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  if(digitalRead(7)==1)
  {
    digitalWrite(9,LOW);
    digitalWrite(10,LOW);
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
     digitalWrite(13,LOW);

    digitalWrite(8,HIGH);
    Serial.println("EMERGENCY");
    delay(500);
    digitalWrite(8,LOW);
    delay(500);
    digitalWrite(8,HIGH);
    Serial.println("EMERGENCY");
    delay(500);
    digitalWrite(8,LOW);
    delay(500);
    digitalWrite(8,HIGH);
    Serial.println("EMERGENCY");
    delay(500);
    digitalWrite(8,LOW);
    delay(500);
    digitalWrite(7,HIGH);
    Serial.println("EMERGENCY");
    delay(500);
```

```
  digitalWrite(8,LOW);
  delay(500);
  digitalWrite(8,HIGH);
  Serial.println("EMERGENCY");
  delay(500);
  digitalWrite(8,LOW);
  delay(5000);
 }
 else
 {
  digitalWrite(8,LOW);
 }

 if(analogRead(A0)<300)
 {
if (digitalRead(4)==1)
{
 digitalWrite(13,HIGH);
 Serial.println("STATE – NIGHT ,  CHECKPOINT  – 1 , LED STATE – ON");
}
 else
  {
 digitalWrite(13,LOW);
}
 if (digitalRead(3)==1)
{
 digitalWrite(12,HIGH);
  Serial.println("STATE – NIGHT ,  CHECKPOINT  – 2 , LED STATE – ON");
}
 else
  {
 digitalWrite(12,LOW);
}
  if (digitalRead(2)==1)
{
 digitalWrite(11,HIGH);
    Serial.println("STATE – NIGHT ,  CHECKPOINT  – 3 , LED STATE – ON");
}
 else
  {
 digitalWrite(11,LOW);
}
    if (digitalRead(5)==1)
{
 digitalWrite(10,HIGH);
     Serial.println("STATE – NIGHT ,  CHECKPOINT  – 4 , LED STATE – ON");
}
 else
  {
 digitalWrite(10,LOW);
}
    if (digitalRead(6)==1)
{
 digitalWrite(9,HIGH);
     Serial.println("STATE – NIGHT ,  CHECKPOINT  – 5 , LED STATE – ON");
}
```

```
  else
    {
   digitalWrite(9,LOW);
 }
}
  else
  {
   digitalWrite(9,LOW);
    digitalWrite(10,LOW);
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
     digitalWrite(13,LOW);

    if (digitalRead(4)==1)
 {
Serial.println("STATE – DAY ,  CHECKPOINT  – 1 , LED STATE – OFF");
 }
   if (digitalRead(3)==1)
 {
Serial.println("STATE – DAY ,  CHECKPOINT  – 2 , LED STATE – OFF");
 }
    if (digitalRead(2)==1)
 {
Serial.println("STATE – DAY ,  CHECKPOINT  – 3 , LED STATE – OFF");
 }
     if (digitalRead(5)==1)
 {
 Serial.println("STATE – DAY ,  CHECKPOINT  – 4 , LED STATE – OFF");
 }
     if (digitalRead(6)==1)
 {
 Serial.println("STATE – DAY ,  CHECKPOINT  – 5 , LED STATE – OFF");

 }
}
}
```



11. AI in Education

Recommendation System for Online Courses

Create a basic recommendation system using collaborative filtering with TensorFlow. Use it to recommend online courses to students based on their previous choices.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.naive_bayes import MultinomialNB
from sklearn.multiclass import OneVsRestClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
from pandas.plotting import scatter_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

from google.colab import drive
drive.mount('/content/drive')

resumeDataSet = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/UpdatedResumeDataSet.csv'
,encoding='utf-8')
resumeDataSet['cleaned_resume'] = ''
resumeDataSet.head()

print ("Displaying the distinct categories of resume -")
print (resumeDataSet['Category'].unique())
```

```
print ("Displaying the distinct categories of resume and the number of records belonging to each category
-")
print (resumeDataSet['Category'].value_counts())

import seaborn as sns
plt.figure(figsize=(15,15))
plt.xticks(rotation=90)
sns.countplot(y="Category", data=resumeDataSet)

import re
def cleanResume(resumeText):
    resumeText = re.sub('http\S+\s*', ' ', resumeText)  # remove URLs
    resumeText = re.sub('RT|cc', ' ', resumeText)  # remove RT and cc
    resumeText = re.sub('#\S+', '', resumeText)  # remove hashtags
    resumeText = re.sub('@\S+', '  ', resumeText)  # remove mentions
    resumeText = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""), ' ', resumeText)  #
remove punctuations
    resumeText = re.sub(r'[^\x00-\x7f]',r' ', resumeText)
    resumeText = re.sub('\s+', ' ', resumeText)  # remove extra whitespace
    return resumeText

resumeDataSet['cleaned_resume'] = resumeDataSet.Resume.apply(lambda x: cleanResume(x))


import nltk

# Download the stopwords resource
nltk.download('stopwords')

import nltk
from nltk.corpus import stopwords
import string
from wordcloud import WordCloud
import matplotlib.pyplot as plt

import nltk
from nltk.corpus import stopwords
import string
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Download the stopwords resource
nltk.download('stopwords')
nltk.download('punkt')

# Rest of your code
oneSetOfStopWords = set(stopwords.words('english')+['``',"''"])
totalWords =[]
Sentences = resumeDataSet['Resume'].values
cleanedSentences = ""
for i in range(0, 160):
```

```
    cleanedText = cleanResume(Sentences[i])
    cleanedSentences += cleanedText
    requiredWords = nltk.word_tokenize(cleanedText)
    for word in requiredWords:
        if word not in oneSetOfStopWords and word not in string.punctuation:
            totalWords.append(word)

wordfreqdist = nltk.FreqDist(totalWords)
mostcommon = wordfreqdist.most_common(50)
print(mostcommon)

wc = WordCloud().generate(cleanedSentences)
plt.figure(figsize=(15,15))
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()

from sklearn.preprocessing import LabelEncoder

var_mod = ['Category']
le = LabelEncoder()
for i in var_mod:
    resumeDataSet[i] = le.fit_transform(resumeDataSet[i])

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.sparse import hstack

requiredText = resumeDataSet['cleaned_resume'].values
requiredTarget = resumeDataSet['Category'].values

word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    stop_words='english',
    max_features=1500)
word_vectorizer.fit(requiredText)
WordFeatures = word_vectorizer.transform(requiredText)

print ("Feature completed .....")

X_train,X_test,y_train,y_test = train_test_split(WordFeatures,requiredTarget,random_state=0,
test_size=0.2)
print(X_train.shape)
print(X_test.shape)


clf = OneVsRestClassifier(KNeighborsClassifier())
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
print('Accuracy of KNeighbors Classifier on training set: {:.2f}'.format(clf.score(X_train, y_train)))
print('Accuracy of KNeighbors Classifier on test set: {:.2f}'.format(clf.score(X_test, y_test)))
```

```
print("\n Classification report for classifier %s:\n%s\n" % (clf, metrics.classification_report(y_test, prediction)))
```

10. AI in Healthcare

Predicting Disease Outcomes from Patient Data

Build a predictive model using TensorFlow to estimate the progression of a specific disease (e.g., diabetes) based on patient data such as age, BMI, and blood sugar levels.

```
import numpy as np
import pandas as pd
import re #Regular expressions
import nltk
import matplotlib.pyplot as plt

from nltk.corpus import stopwords

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

from google.colab import files
uploaded = files.upload()


dataset = pd.read_csv('/content/Coronavirus Tweets.csv', encoding='ISO-8859-1')
print(dataset.shape)
print(dataset.head(5))


features = dataset.iloc[:, 4].values
labels = dataset.iloc[:, 5].values
print(labels)

processed_features = []

for sentence in range(0, len(features)):
    # Remove all the special characters
    processed_feature = re.sub(r'\W', ' ', str(features[sentence]))

    # remove all single characters
    processed_feature= re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)

    # Remove single characters from the start
    processed_feature = re.sub(r'\^[a-zA-Z]\s+', ' ', processed_feature)

    # Substituting multiple spaces with single space
    processed_feature = re.sub(r'\s+', ' ', processed_feature, flags=re.I)
```

```
    # Removing prefixed 'b'
    processed_feature = re.sub(r'^b\s+', '', processed_feature)

    # Converting to Lowercase
    processed_feature = processed_feature.lower()

    processed_features.append(processed_feature)


nltk.download('stopwords')
vectorizer = TfidfVectorizer (max_features=2500, min_df=7, max_df=0.8, stop_words=stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()
print(processed_features)

X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2, random_state=0)

text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)


predictions = text_classifier.predict(X_test)

print(accuracy_score(y_test, predictions))


from sklearn import metrics
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm = metrics.confusion_matrix(y_test, predictions, labels=['negative', 'neutral', 'positive','Extremely Negative','Extremely Positive'])
plot_confusion_matrix(cm, classes=['negative', 'neutral', 'positive','Extremely Negative','Extremely Positive'])
```