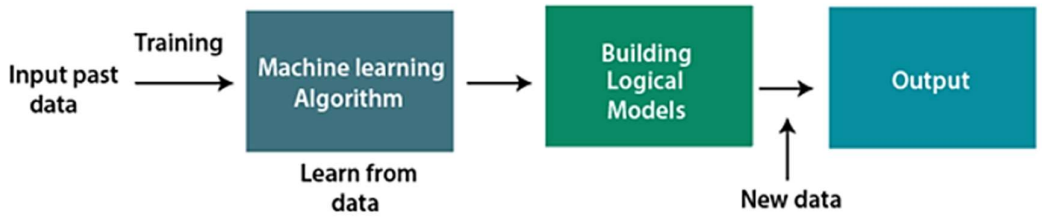
Machine Learning

In the actual world, we are surrounded by individuals who have the potential to learn from their experiences, as well as computers or robots that function on our orders. Can a machine, like a person, learn from prior experiences or data? So now comes Machine Learning's part.

Machine Learning work

A Machine Learning system learns from previous data, constructs prediction models, and forecasts the output for fresh data. The amount of data influences the accuracy of projected output since a large amount of data helps to construct a better model that predicts the output more precisely.

If we have a complex problem that requires certain predictions, instead of creating code for it, we can just input the data to generic algorithms, and the machine will develop the logic based on the data and forecast the outcome. Machine learning has altered our perspective on the issue. The following block diagram describes how the Machine Learning algorithm works:



Features of Machine Learning

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Need for Machine Learning

The need for machine learning is growing by the day. The necessity for machine learning stems from its ability to do tasks that are too complicated for a human to perform directly. As humans, we have some limits since we cannot access large amounts of data manually, therefore we need computer systems, and this is where machine learning comes in to help us.

We can train machine learning algorithms by feeding them massive amounts of data and allowing them to autonomously examine the data, develop models, and anticipate the desired output. The performance of the machine learning algorithm is defined by the cost function and is dependent on the amount of data.

The value of machine learning may be simply grasped by looking at its applications. Machine learning is being employed in self-driving cars, cyber fraud detection, facial recognition, and Facebook friend

recommendation, among other applications. Several leading firms, like Netflix and Amazon, have built machine learning algorithms that analyse user interest and propose products based on that data.

Classification of Machine Learning

At a broad level, machine learning can be classified into three types:

1. *Supervised learning*

   Supervised learning is a form of machine learning approach in which we feed sample labelled data to the machine learning system in order to train it, and it predicts the output based on that. The system builds a model using labelled data to interpret the datasets and learn about each data. After training and processing, we test the model by supplying sample data to see if it predicts the precise output or not. The purpose of supervised learning is to connect input and output data. Supervised learning is dependent on monitoring, such like when a student learns something under the observation of a teacher. Spam filtering is one example of supervised learning.

   Supervised learning can be grouped further in two categories of algorithms:

   - **Classification**
   - **Regression**

2. *Unsupervised learning*

   Unsupervised learning is a learning approach in which a machine learns without any supervision. The machine is trained given a collection of unlabeled, classified, or categorised data, and the algorithm must operate on the data without supervision. The purpose of unsupervised learning is to rearrange the incoming data into new features or a set of objects with similar patterns.

   There is no predefined outcome in unsupervised learning. The machine attempts to extract helpful insights from the massive volume of data. It is further divided into two types of algorithms:

   - **Clustering**
   - **Association**

3. *Reinforcement learning*

   Reinforcement learning is a feedback-based learning strategy in which a learning agent is rewarded for correct actions and penalized for incorrect actions. With these feedbacks, the agent automatically learns and improves its performance. The agent interacts with and investigates the environment in reinforcement learning. An agent's purpose is to earn the most reward points possible, so it enhances its performance.

   Reinforcement learning is demonstrated by the robotic dog, which automatically learns the movement of his arms.

Our Problem

The problem statement at hand involves the prediction of voltage gain at a specific angle for a given antenna data set using machine learning algorithms. This is a regression problem in supervised machine learning, where the objective is to predict a continuous output variable based on a set of input variables.

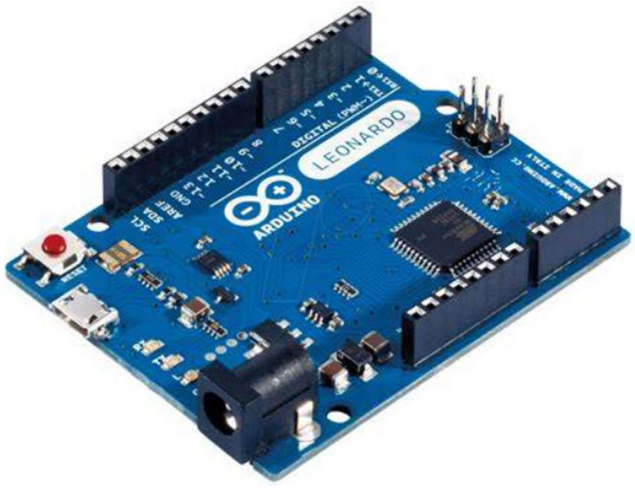
The dataset in question contains information on voltage gain at various angles for a given antenna. The aim is to use this dataset to predict the voltage gain at a particular angle using machine learning techniques. The challenge lies in identifying an appropriate machine-learning algorithm that can accurately predict the voltage gain based on the available input variables.

To address this challenge, our group worked with various machine learning algorithms and tested their performance on the dataset. The goal was to identify an algorithm that could provide the most accurate predictions of voltage gain at a particular angle. After thorough experimentation and evaluation, the group found that polynomial regression yielded the best results for this problem.

We trained the polynomial regression algorithm on the dataset to predict voltage gain at corresponding angles. The training process involved providing the algorithm with input data and corresponding output data, and the algorithm adjusted its parameters to minimize the difference between predicted and actual outputs. The resulting model was then tested on a validation set to evaluate its accuracy and generalization capability.

In conclusion, the problem statement involved predicting voltage gain at a specific angle for a given antenna data set using machine learning algorithms. The group experimented with various algorithms and identified polynomial regression as the most effective technique for this problem. The resulting model was trained on the dataset and evaluated on a validation set, achieving accurate predictions of voltage gain at corresponding angles.

Collection of Dataset using Arduino Microcontroller and Python:



We concern the collection of a dataset using an Arduino microcontroller connected to an antenna setup, and the use of a Python script to save the collected data in a CSV file. The objective of this data collection is to obtain information on the voltage gain at a particular angle for the antenna, to use this data for further analysis or prediction.

To collect the data, an Arduino microcontroller is used to interface with the antenna setup and obtain voltage gain measurements at specific angles. The microcontroller collects data for 20 different voltage gain readings, each taken at a particular angle, with a delay interval between each reading. This voltage gain data is transmitted via the COM3 port of a computer, where a Python script is used to receive and process the data.

In the Python script, the received data is saved in a CSV file, which can be used for further analysis or prediction. The CSV file contains the voltage gain readings collected by the Arduino microcontroller, as well as the corresponding angle for each reading.

The data collection process involves multiple steps, including setting up the Arduino microcontroller and the antenna setup, establishing communication between the microcontroller and the computer, and developing a Python script to receive and process the collected data. These steps require technical expertise and careful attention to detail to ensure that the collected data is accurate and reliable.

Adruino Code:

```
float inputValue=0;
float inputValue1=0;
float inputValue2=0;
float inputValue3=0;
float inputValue4=0;
float inputValue5=0;
int analogOutputPin = 11;

void setup()
{
    pinMode(11,OUTPUT);
    pinMode(A0,INPUT);
    pinMode(A1,INPUT);
    pinMode(A2,INPUT);
    pinMode(A3,INPUT);
    pinMode(A4,INPUT);
    pinMode(A5,INPUT);
    Serial.begin(9600);
}
void loop()
{
    inputValue = analogRead(A0);
    inputValue1 = analogRead(A1);
    inputValue2 = analogRead(A2);
    inputValue3 = analogRead(A3);
    inputValue4 = analogRead(A4);
    inputValue5 = analogRead(A5);
    float temp = inputValue*5/1023;
    float temp1 = inputValue1*5/1023;
    float temp2 = inputValue2*5/1023;
    float temp3 = inputValue3*5/1023;
    float temp4 = inputValue4*5/1023;
    float temp5 = inputValue5*5/1023;
    // convert all float to string to avoid delay problem in values
    String str = String(temp, 4);
    String str1 = String(temp1, 4);
    String str2 = String(temp2, 4);
    String str3 = String(temp3, 4);
    String str4 = String(temp4, 4);
    String str5 = String(temp5, 4);
    Serial.println(str+","+str1+","+str2+","+str3+","+str4+","+str5);
    delay(2000);
}
```

Python Script:

```
import serial
import time
import pandas as pd
import numpy as np
import re


list=['COM1','COM2','COM3','COM4','COM5','COM6','COM7','COM8',
'COM9','COM10','COM11','COM12','COM13','COM14','COM15','COM16','COM17','COM18',]

COM1='COM1'
COM2='COM2'
COM3='COM3'
COM4='COM4'
COM5='COM5'
COM6='COM6'
COM7='COM7'
COM8='COM8'
COM9='COM9'
COM10='COM10'
COM11='COM11'
COM12='COM12'
COM13='COM13'
COM14='COM14'
COM15='COM15'
COM16='COM16'
COM17='COM17'
COM18='COM18'
COM19='COM19'
```

```
time.sleep(1)
ser = serial.Serial()
ser.baudrate = 9600

i=1

while True:
    time.sleep(.2)
    print(i)
    ser.port = list[i]
    try:

        ser.open()
        if ser.isOpen()==True:
            print('Connected')
            #print('arduino is on COMPORT'.join(i))
            break
        break

    except:
        print('Waiting...')
        i=i+1
        if i==18:
            print('Kindly remove usb cable and try again')
            break


print('Loading...')
index_val = 0
```

```
try:
    DF1  = pd.read_csv("data1.csv",header=None)
    if DF1.shape[0]==1:
        DF1 = DF1.iloc[0]
    else:
        DF1 = DF1.iloc[-1]
    index_val = DF1.iloc[0]
    print(index_val)
    index_val+=5
except:
    print("Empty CSV file")


arr1 = [0.0,0.0,0.0,0.0,0.0,0.0]
index=0
ans = 0.0
```

```
while True:
    "a1,a2,a3"
    temp = re.findall(r"[-+]?\d*\.?\d+|[-+]?\d+", ser.readline().decode('utf-8'))
    if temp==[]:
        continue
    print(temp)
    for j in range(len(temp)):
        arr1[j]+=float(temp[j])
    index=index+1
    if index==21:
        break


for elem in range(len(arr1)):
    arr1[elem] /=20.0

arr1.insert(0, index_val)
DF = pd.DataFrame(arr1).transpose()
DF.to_csv('data1.csv', mode='a', index=False, header=False)
print(DF)
```

Model Training using Polynomial Regression:

The next step to train the collected data using the machine learning algorithm polynomial regression. This step involves inputting the collected data into the polynomial regression algorithm, which will learn the relationship between the input variables (voltage gain readings at specific angles) and the output variable (voltage gain at a particular angle). The aim is to use this trained model to make accurate predictions of voltage gain at any angle within the range of the collected data. This step is crucial in utilizing the collected data to its full potential and can have implications in antenna design and optimization.

Algorithm Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import os

base_url = os.getcwd()
df=pd.read_csv(base_url+'/training data/6_port_data.csv')

def train_modal():
    # Dataframes
    X=(df.iloc[:,1:])
    y=df['Angle']
    y = pd.DataFrame(y, columns = ['Angle'])
    print(X,y)
    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20, random_state=0)
    # Transform the input data into a polynomial feature space
    poly = PolynomialFeatures(degree=9)
    X_train_poly = poly.fit_transform(X_train)
    X_test_poly = poly.transform(X_test)
    # Initialize Linear Regression model
    reg = LinearRegression()
    # Fit the model to the training data
    reg.fit(X_train_poly, y_train)
    y_pred = reg.predict(X_test_poly)
    # Calculate R-squared score to evaluate the model
    r2 = r2_score(y_test, y_pred)
    print("R-squared score:", r2)
    print(y_test)
    print(y_pred)
    return reg,poly,y,X,X_test,y_pred,y_test
```

Plot Graph:

```
import matplotlib.pyplot as plt


def plot_Graph(y,X,X_test,y_pred,y_test,title):
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.suptitle(title)
    ax1.plot(y,X)
    ax2.plot(y_test,X_test,c='blue',linestyle='dotted')
    ax2.plot(y_pred,X_test,c='red',linestyle='dashed')
    plt.show(block=False)
```

Output Graph:

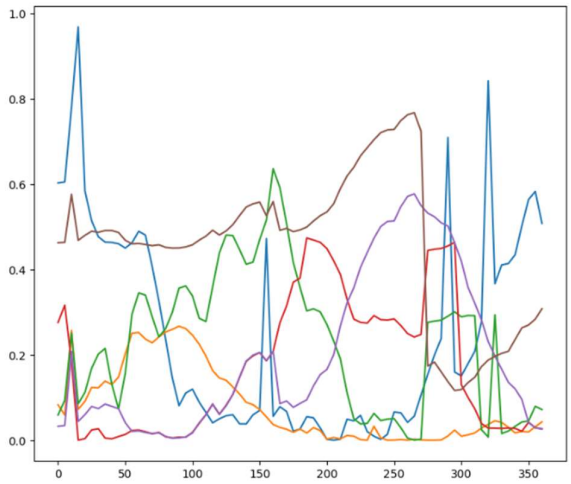


Fig: Input data of 6 ports voltage gain vs angle

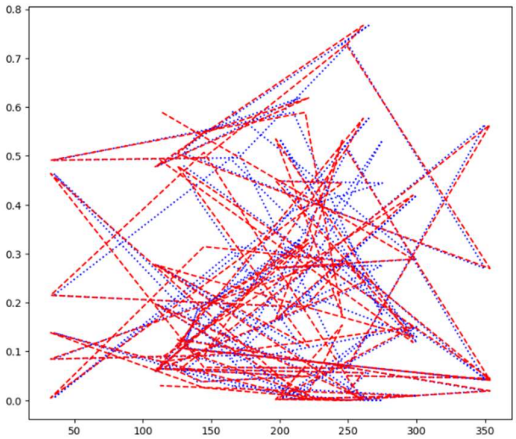


Fig: Trained Output Angle(Red) vs Actual Angle(Blue)

The graph represents the relationship between the trained output of a machine learning algorithm and the actual output of an antenna. It shows how well the trained model can predict the actual output of the antenna at different angles. The graph is an important tool for evaluating the performance of the machine learning algorithm and can be used to identify areas where the model is accurate and areas where it needs improvement. The graph can also be used to make decisions about antenna design and optimization based on the predicted output.

```
ser = serial.Serial()
ser.baudrate = 9600
# ---:Try to connect with arduino microcontroller:---
is_connected = True
# is_connected = connect_arduino(ser)

if is_connected:
    # ---:Training of model before live testing with antenna:---
    reg,poly,y,X,X_test,y_pred,y_test =  train_modal()
    # plot_Graph(y,X,X_test,y_pred,y_test,'Antenna Data training using Polynomial Regression')
    index=0
    ans_20=[0.0,0.0,0.0,0.0,0.0,0.0]
    while True:
        # res = read_values(ser)
        # print(res)
        res = [0.44989,0.20209,0.156405,0.01347,0.041055,0.4682]
        for i in range(len(res)):
            res[i] = float(res[i])
            ans_20[i]+=res[i]
        index+=1
        if index==20:
            index=0
            for elem in range(len(ans_20)):
                ans_20[elem]/=20.0
            input_arr = [ans_20]
            input_ = np.asarray(input_arr)
            X = poly.fit_transform(input_)
            print("Angle at => "+get_time()+"=>")
            print(round(predict_angle(X,reg)[0][0], 3))
            for elem in range(6):
                ans_20[elem]=0.0
        time.sleep(0.08)
else:
    print("Unable to connect with arduino. Try again...")
```

Testing:

The final stage of the project involves testing the machine learning model by feeding it live input data and observing its performance in predicting the voltage gain at a particular angle. This stage is critical in evaluating the effectiveness of the trained model in real-world scenarios. The live input data is fed into the model, and the output is compared with the actual voltage gain measured at the given angle. This stage can reveal any discrepancies between the model's predictions and the actual output, allowing for further refinement of the model. Overall, the testing stage is crucial for determining the practical viability of the machine learning model for predicting voltage gain at a specific angle.

Python Code:

```
ser = serial.Serial()
ser.baudrate = 9600
# ---:Try to connect with arduino microcontroller:---
is_connected = True
# is_connected = connect_arduino(ser)

if is_connected:
    # ---:Training of model before live testing with antenna:---
    reg,poly,y,X,X_test,y_pred,y_test =  train_modal()
    # plot_Graph(y,X,X_test,y_pred,y_test,'Antenna Data training using Polynomial Regression')
    index=0
    ans_20=[0.0,0.0,0.0,0.0,0.0,0.0]
    while True:
        # res = read_values(ser)
        # print(res)
        res = [0.44989,0.20209,0.156405,0.01347,0.041055,0.4682]
        for i in range(len(res)):
            res[i] = float(res[i])
            ans_20[i]+=res[i]
        index+=1
        if index==20:
            index=0
            for elem in range(len(ans_20)):
                ans_20[elem]/=20.0
            input_arr = [ans_20]
            input_ = np.asarray(input_arr)
            X = poly.fit_transform(input_)
            print("Angle at => "+get_time()+"=>")
            print(round(predict_angle(X,reg)[0][0], 3))
            for elem in range(6):
                ans_20[elem]=0.0
        time.sleep(0.08)
else:
    print("Unable to connect with arduino. Try again...")
```

Conclusion

In conclusion, the problem statement involved collecting data on voltage gain at specific angles for an antenna setup using an Arduino microcontroller and saving it in a CSV file. The next step was to use machine learning algorithms to train this collected data and predict the voltage gain at any angle within the range of collected data. Polynomial regression was found to be the most effective algorithm for this regression problem. Finally, the trained model was tested using live input data to predict the voltage gain at a particular angle, providing a practical evaluation of the model's effectiveness.

This project showcases the potential of machine learning algorithms in solving complex problems in antenna design and optimization. It also highlights the importance of careful data collection and processing to ensure the accuracy and reliability of the trained model. Overall, the project provides valuable insights into the use of machine learning in antenna design and lays the groundwork for further research in this area.