Analysis of Different Electrical Components and Properties of a RLC circuit using Machine Learning

### A Major Project report for evaluation and partial fulfillment of the requirement for the award of

#### Integrated Dual Degree Programme B.Tech(ECE)-M.Tech (CSE)



### **Submitted by: Supervisor:**

### Ayush Gangwar (16/IEC/009) Mr. Jatin

### Shashank Singh (16/IEC/057)

### Rishabh Rathi (16/IEC/033)

#### SCHOOL OF INFORMATION & COMMUNICATION TECHNOLOGY GAUTAM BUDDHA UNIVERSITY

#### GREATER NOIDA Uttar Pradesh - 201312

#### SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

GAUTAM BUDDHA UNIVERSITY, GREATER NOIDA, 201312, U. P., (INDIA)

## Candidate’s Declaration

We, hereby, certify that the work embodied in this project report entitled **“Analysis of Different Electrical Components and Properties of a RLC circuit using Machine Learning*”*** in partial fulfillment of the requirements for the award of the dual degree of B. Tech (ECE) + M. Tech (CSE) Submitted to the School of Information and Communication Technology, Gautam Buddha University, Greater Noida is an authentic record of our own work carried out under the supervision of **Mr. Jatin,** School of ICT. The matter presented in this report has not been submitted in any other University / Institute for the award of any other degree or diploma. Responsibility for any plagiarism related issue stands solely with us.

#### NAME: - ROLL NO.:- SIGNATURE:-

#### AYUSH GANGWAR 16/IEC/009

#### SHASHANK SINGH 16/IEC/057

#### RISHABH RATHI 16/IEC/033

This is to certify that the above statement made by the candidates is correct to the best of my knowledge and belief. However, responsibility for any plagiarism related issue solely stands with the students.

Date:

Place: Greater Noida

Signature of the Supervisor:

Name : **Mr. Jatin**

**Acknowledgements**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

We are highly indebted to **Mr. Jatin** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project. We would like to express our gratitude towards our friends & family members for their kind co-operation and encouragement which helped us in completion of this project.

Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

**Abstract**

In the field of artificial intelligence and computer gaming, coined the term “Machine Learning”. Machine learning define as – “Field of study that gives computers the capability to learn without being explicitly programmed”. In a very layman manner, Machine Learning (ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance. The process starts with feeding good quality data and then training our machines(computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate.

In this report, we have collected datasets of resistance (based on wire length, wire width & temperature); capacitance (based on to wire length, wire width & wire thickness) and inductance (based on wire length, wire width & wire thickness) and used them to create machine learning models to predict one value out the 4(i.e., resistance, wire length, wire width & temperature) when the other 3 are feeded by the user with high accuracy in case of resistance. Same can be done for capacitance and inductance as well.

We have also collected datasets of different properties of a RLC circuit like quality factor, damping factor, resonance frequency and bandwidth (based on resistance, capacitance and inductance) and used them to design different RLC circuits when value of any one of the 4 properties mentioned above is feeded by the user using machine learning.

**CONTENTS**

Declaration

Acknowledgements Abstract

[List of Figures 7](#_TOC_250013)

1. [Introduction 8](#_TOC_250014)
   1. [Machine Learning 8](#_TOC_250013)
   2. [Process 10](#_TOC_250012)
   3. How Machine Learning is related to ICs 10
   4. Objectives 10
2. [Theory 11](#_TOC_250011)
   1. [Machine Learning 11](#_TOC_250010)
   2. [Applications of Machine Learning 12](#_TOC_250009)
   3. [Machine Learning Usage in Electronics 13](#_TOC_250008)
   4. [Machine Learning Usage in Computer Science 14](#_TOC_250007)
   5. [Machine Learning Approaches 14](#_TOC_250007)
3. [Hardware, Software & Language used 21](#_TOC_250006)
   1. [Hardware 21](#_TOC_250005)
   2. [Software 21](#_TOC_250004)
   3. [Language 21](#_TOC_250003)
   4. [Library 21](#_TOC_250003)
   5. [Dataset 21](#_TOC_250003)
4. Process…………………………………………………………………………22
   1. Procedural Steps ………………………………………………………………………...22
5. Implementation 24
   1. [Codes 24](#_TOC_250013)
6. [Result and Output 40](#_TOC_250011)
   1. [Results 40](#_TOC_250005)
   2. [Outputs 40](#_TOC_250005)
7. [Future Scope 46](#_TOC_250011)
8. [Conclusion 47](#_TOC_250011)
9. [References 48](#_TOC_250011)

List of Figures

[Figure 1.1 Traditional Programming and Machine Learning 8](#_TOC_250014)

[Figure 2.1 Splitting of Data in Machine Learning ..11](#_TOC_250012)

[Figure 2.2 Classification type Supervised Learning ..15](#_TOC_250011)

[Figure 2.3 Regression type Supervised Learning .16](#_TOC_250006)

[Figure 2.4 Unsupervised Learning .16](#_TOC_250006)

[Figure 2.5 Clustering in Unsupervised Learning .17](#_TOC_250006)

[Figure 2.6 Semi-Supervised Learning .18](#_TOC_250006)

[Figure 2.7 Reinforcement Learning .19](#_TOC_250006)

[Figure 2.8 Step 1 of Reinforcement Learning .19](#_TOC_250006)

[Figure 2.9 Step 2 of Reinforcement Learning .20](#_TOC_250006)

# 1. Introduction

**1.1 Machine Learning**

In the field of artificial intelligence and computer gaming, coined the term “Machine Learning”. Machine learning define as – “Field of study that gives computers the capability to learn without being explicitly programmed”. In a very layman manner, Machine Learning(ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance. The process starts with feeding good quality data and then training our machines(computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

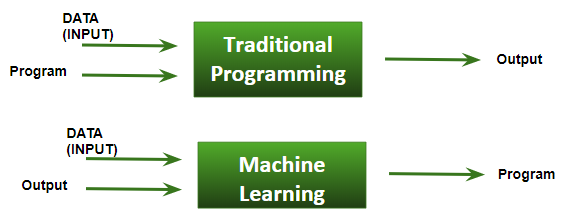


Fig. [1.1]

**Some machine learning methods**

Machine learning algorithms are often categorized as supervised or unsupervised.

* **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
* In contrast, **unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labelled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabelled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabelled data.
* **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabelled data for training – typically a small amount of labelled data and a large amount of unlabelled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labelled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabelled data generally doesn’t require additional resources.
* Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.
* Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

## Terminology

**Dataset**: A set of data examples, that contain features important to solving the problem.

**Features**: Important pieces of data that help us understand a problem. These are fed in to a Machine Learning algorithm to help it learn.

**Model**: The representation (internal model) of a phenomenon that a Machine Learning algorithm has learnt. It learns this from the data it is shown during training. The model is the output you get after training an algorithm. For example, a decision tree algorithm would be trained and produce a decision tree model.

## 1.2 Process

1. **Data Collection:** Collect the data that the algorithm will learn from.
2. **Data Preparation:** Format and engineer the data into the optimal format, extracting important features and performing dimensionality reduction.
3. **Training:** Also known as the fitting stage, this is where the Machine Learning algorithm actually learns by showing it the data that has been collected and prepared.
4. **Evaluation:** Test the model to see how well it performs.
5. **Tuning:** Fine tune the model to maximize it’s performance.

**1.3 How Machine Learning Is Related To ICs**

Machine Learning works by extracting information from the data in the form of a model consisting of a very complex equation of dependent variables representing the data. So long as one can represent the IC design problem with underlying data, one can extract the solution as a machine learning model.

Semiconductor engineers are already hearing AI’s footsteps as it encroaches on their design work. Consider the vast amount of design data and variability required of chip designers, especially when developing a variety of chips with different power, temperature and performance specs. Complex IC designs might well be one of the logical areas to apply machine learning.

**1.4 Objectives**

* Resistance analysis using Machine Learning
* Capacitance analysis using Machine Learning
* Inductance analysis using Machine Learning
* RLC series circuit analysis using Machine Learning.

**2. Theory**

**2.1 Machine Learning**

Machine learning (ML) is a category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use [statistical analysis](https://whatis.techtarget.com/definition/statistical-analysis) to predict an output while updating outputs as new data becomes available.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

# **How we split data in Machine Learning: -**

* **Training Data:**The part of data we use to train our model. This is the data which your model actually sees (both input and output) and learn from.
* **Validation Data:**The part of data which is used to do a frequent evaluation of model, fit on training dataset along with improving involved hyper parameters (initially set parameters before the model begins learning). This data plays its part when the model is actually training.
* **Testing Data:**Once our model is completely trained, testing data provides the unbiased evaluation. When we feed in the inputs of testing data, our model will predict some values (without seeing actual output). After prediction, we evaluate our model by comparing it with actual output present in the testing data. This is how we evaluate and see how much our model has learned from the experiences feed in as training data, set at the time of training.

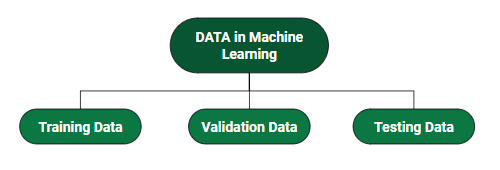
****

Fig. [2.1]

**Basic Difference in ML and Traditional Programming: -**

* **Traditional Programming:**We feed in DATA (Input) + PROGRAM (logic), run it on machine and get output.
* **Machine Learning:**We feed in DATA (Input) + Output, run it on machine during training and the machine creates its own program (logic), which can be evaluated while testing.

# 2.2 Applications of Machine Learning

# Machine learning technology does not need to be introduced as it has already made its place in the hearts of the people. But still, for the sake of the beginners, we would like to give a brief introduction to it.

# Artificial Intelligence (AI) is everywhere. Possibility is that you are using it in one way or the other and you don’t even know about it. One of the popular applications of AI is Machine Learning (ML), in which computers, software, and devices perform via cognition (very similar to human brain).

# ****1. Virtual Personal Assistants****

# Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice.

# Virtual Assistants are integrated to a variety of platforms. For example:

* Smart Speakers: Amazon Echo and Google Home
* Smartphones: Samsung Bixby on Samsung S8
* Mobile Apps: Google Allo

**2. Predictions while Commuting**

**Traffic Predictions**: We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic.

**Online Transportation Networks:** When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours. The answer is machine learning.

# ****3. Videos Surveillance****

# The video surveillance system nowadays is powered by AI that makes it possible to detect crime before they happen. They track unusual behavior of people like standing motionless for a long time, stumbling, or napping on benches etc.

# ****4. Social Media Services****

# From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits.

# ****5. Email Spam and Malware Filtering****

* There are a number of spam filtering approaches that email clients use. To ascertain that these spam filters are continuously updated, they are powered by machine learning.
* Over 325, 000 malwares are detected every day and each piece of code is 90–98% similar to its previous versions. The system security programs that are powered by machine learning understand the coding pattern.

**6. Online Customer Support**

A number of websites nowadays offer the option to chat with customer support representative while they are navigating within the site.

**7. Search Engine Result Refining**

Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results.

**8. Product Recommendations**

You shopped for a product online few days back and then you keep receiving emails for shopping suggestions. If not this, then you might have noticed that the shopping website or the app recommends you some items that somehow matches with your taste.

**9. Online Fraud Detection**

Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. For example: PayPal is using ML for protection against money laundering.

**2.3 Machine Learning Usage in Electronics**

The rise of machine learning (ML) has introduced many opportunities for computer-aided-design, VLSI design, and their intersection. Related to computer-aided design, we review several classical CAD algorithms which can benefit from ML, outline the key challenges, and discuss promising approaches. In particular, because some of the existing ML accelerators have used asynchronous design, we review the state-of-the-art in asynchronous CAD support, and identify opportunities for ML within these flows.

The 3-D integration helps improve performance and density of electronic systems. However, since electrical and thermal performance for 3-D integration is related to each other, their code sign is required. Machine learning, a promising approach in artificial intelligence, has recently shown promise for addressing engineering optimization problems.  We can apply machine learning for the optimization of 3-D integrated systems where the electrical performance and thermal performance need to be analyzed together for maximizing performance. In such systems, modelling can be challenging due to the multiscale geometries involved, which increases computation time per iteration. Machine learning can be applied to such systems where multiple parameters can be optimized to achieve the desired performance using the minimum number of iterations.

# 2.4 Machine Learning Usage in Computer Science

# In computer science, machine learning refers to a type of data analysis that uses algorithms that learn from data. It is a type of artificial intelligence (AI) that provides systems with the ability to learn without being explicitly programmed. This enables computers to find data within data without human intervention. What is important to know about machine learning is that data is being used to make predictions, not code. Data is dynamic so machine learning allows the system to learn and evolve with experience and the more data that is analyzed.

## 2.5 Machine Learning Approaches

There are many approaches that can be taken when conducting Machine Learning. They are:

* [Supervised Learning](https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0#6246)
* [Unsupervised Learning](https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0#0ea7)
* [Semi-supervised Learning](https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0#8316)
* [Reinforcement Learning](https://towardsdatascience.com/machine-learning-an-introduction-23b84d51e6d0#2745)

**2.5.1 Supervised Learning**

In supervised learning, the goal is to learn the **mapping** (the **rules**) between a set of **inputs** and **outputs.**

For example, the inputs could be the weather forecast, and the outputs would be the visitors to the beach. The goal in supervised learning would be to **learn the mapping that describes the relationship** between temperature and number of beach visitors.

**Example**

**Labelled** data is provided of past **input and output pairs** during the learning process to teach the model how it should behave, hence, ‘supervised’ learning. For the beach example, **new inputs** can then be fed in of forecast temperature and the Machine learning algorithm will then **output a future prediction** for the number of visitors.

The output from a supervised Machine Learning model could be a **category** from a finite set

e.g. [low, medium, high] for the number of visitors to the beach:

Input [temperature=**20**] -> *Model* -> Output = [visitors=**high**]

When this is the case, it’s is deciding how to **classify the input**, and so is known as **classification**.

Alternatively, the output could be a **real-world scalar** (output a number):

Input [temperature=**20**] -> *Model* -> Output = [visitors=**300**]

When this is the case, it is known as **regression**.

**Classification**

Classification is used to group the similar data points into different sections in order to classify them. Machine Learning is used to **find the rules** that explain how to separate the different data points.

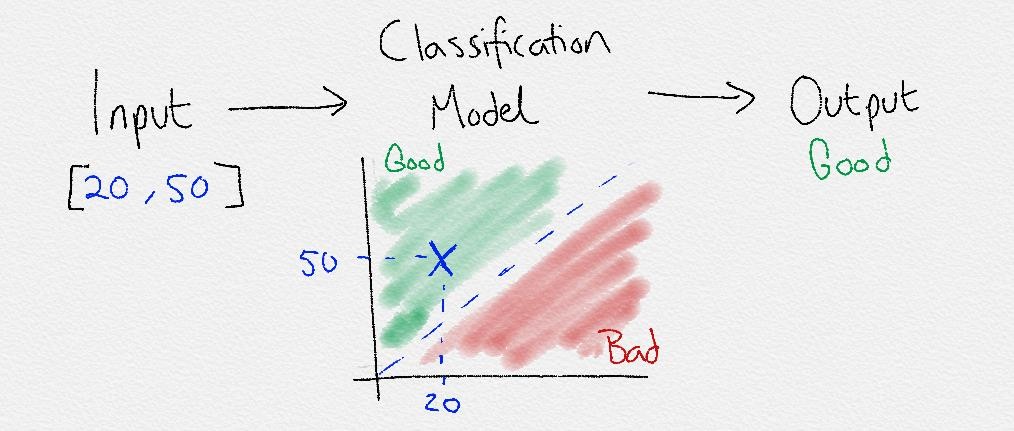


Fig. [2.2] Classification

#### Regression

Regression is another form of supervised learning. The difference between classification and regression is that regression **outputs a number** rather than a class. Therefore, regression is useful when predicting number-based problems like stock market prices, the temperature for a given day, or the probability of an event.

Both the classification and regression supervised learning techniques can be extended to much more complex tasks. For example, tasks involving speech and audio. Image classification, object detection and chat bots are some examples.

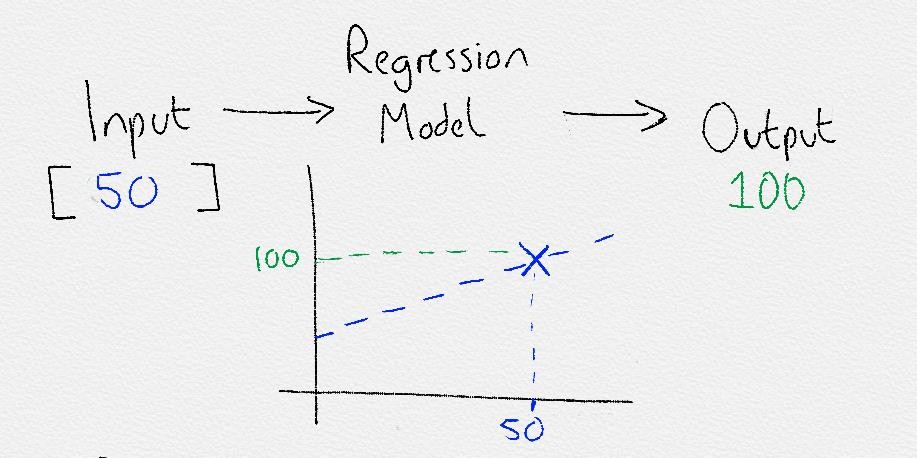


Fig. [2.3] Regression

#### Examples

Regression is used in financial trading to find the patterns in stocks and other assets to decide when to buy/sell and make a profit. For classification, it is already being used to classify if an email you receive is spam.

**2.5.2 Unsupervised Learnin****g**

Fig. [2.4] Unsupervised learning

In unsupervised learning, **only input data** is provided in the examples. There are no labelled example outputs to aim for. But it may be surprising to know that it is still possible to find many interesting and complex patterns hidden within data without any labels.

An example of unsupervised learning in real life would be sorting different color coins into separate piles. Nobody taught you how to separate them, but by just looking at their features such as color, you can see which color coins are **associated** and **cluster** them into their correct features.

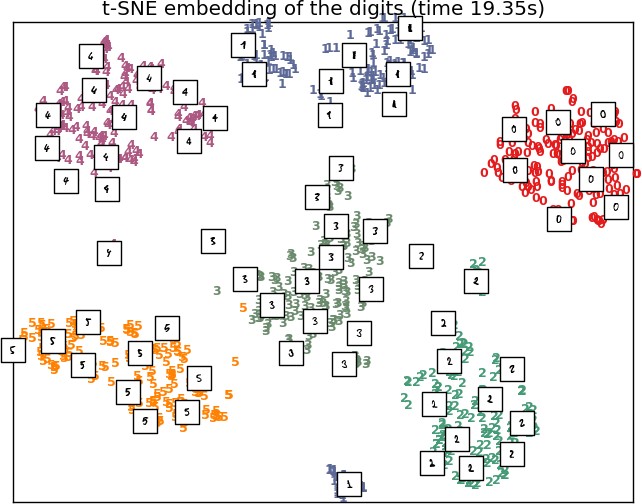


Fig. [2.5] An unsupervised learning algorithm (t-SNE) correctly clusters handwritten digits into groups, based only on their characteristics

#### Clustering

Unsupervised learning is mostly used for **clustering**. Clustering is the act of **creating groups with differing characteristics**. Clustering attempts to find various **subgroups** within a dataset.

#### Association

In Association Learning you want to **uncover the rules that describe your data**. For example, if a person watches video A they will likely watch video B. Association rules are perfect for examples such as this where you want to find **related items**.

#### Anomaly Detection

The **identification of rare or unusual items** that differ from the majority of data. For example, your bank will use this to detect fraudulent activity on your card.

#### Dimensionality Reduction

Dimensionality reduction aims to **find the most important features** to reduce the original feature set down into a smaller more efficient set that **still encodes the important data**.

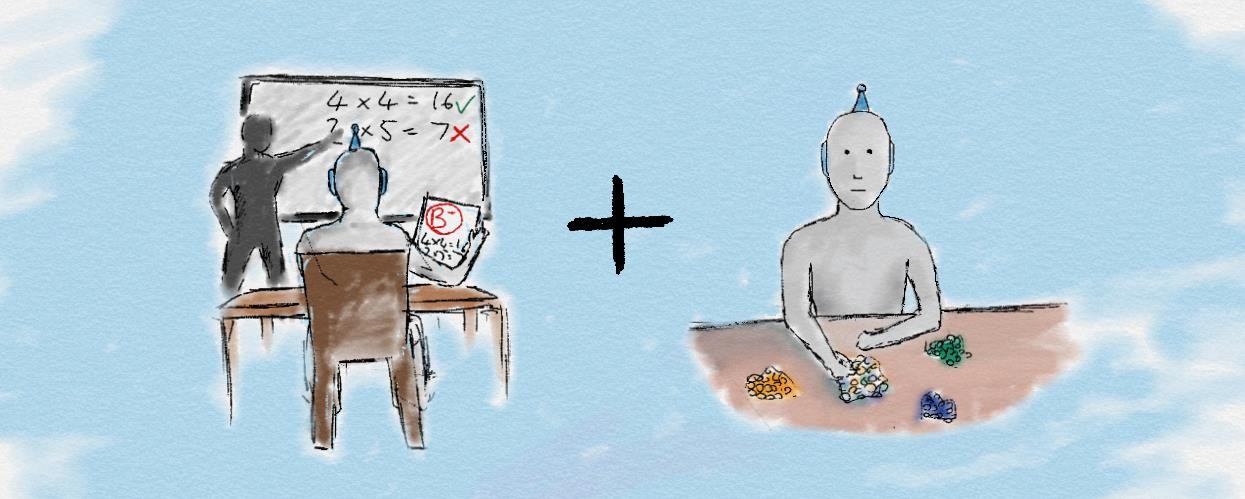
**2.5.3 Semi-supervised learning**

Fig. [2.6] Semi-supervised learning

Semi-supervised learning is a mix between supervised and unsupervised approaches. The learning process isn’t closely supervised with example outputs for every single input, but we also don’t let the algorithm do its own thing and provide no form of feedback. Semi-supervised learning takes the middle road.

By being able to **mix together a small amount of labelled data with a much larger unlabeled dataset** it reduces the burden of having enough labelled data. Therefore, it opens up many more problems to be solved with machine learning.

#### Examples

A perfect example is in medical scans, such as breast cancer scans. A trained expert is needed to label these which is time consuming and very expensive. Instead, an expert can **label just a small set** of breast cancer scans, and the semi-supervised algorithm would be able to leverage this small subset and **apply it to a larger set** of scans.

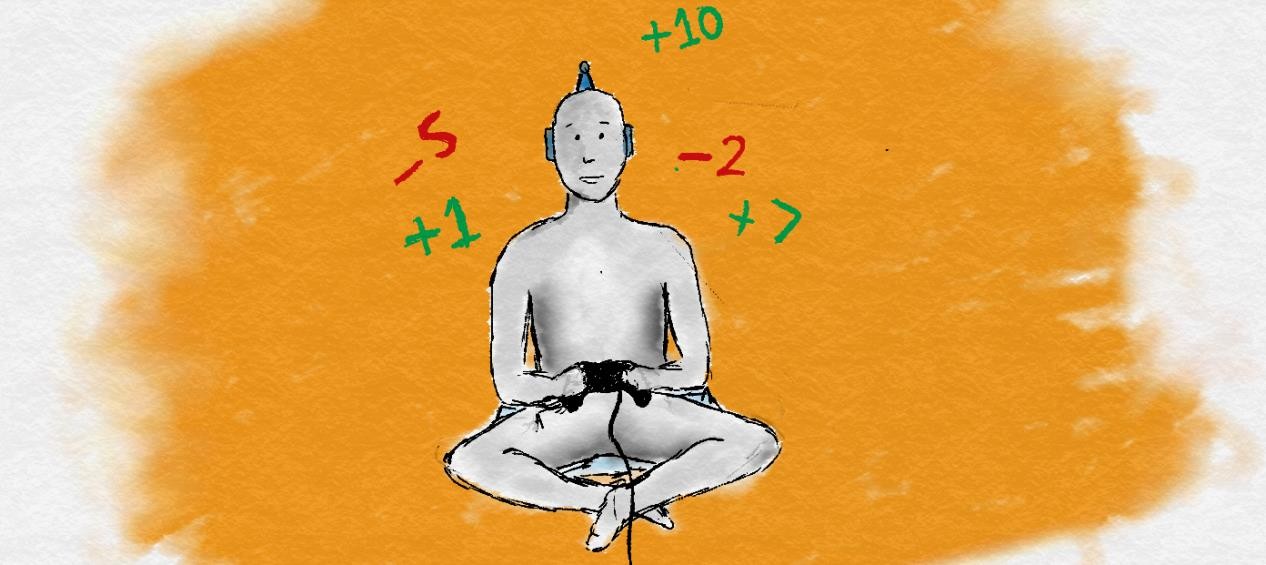
**2.5.4 Reinforcement Learning**

Fig. [2.7] Reinforcement Learning

The final type of machine learning is by far my favorite. It is less common and much more complex, but it has generated incredible results. It doesn’t use labels as such, and instead uses rewards to learn.

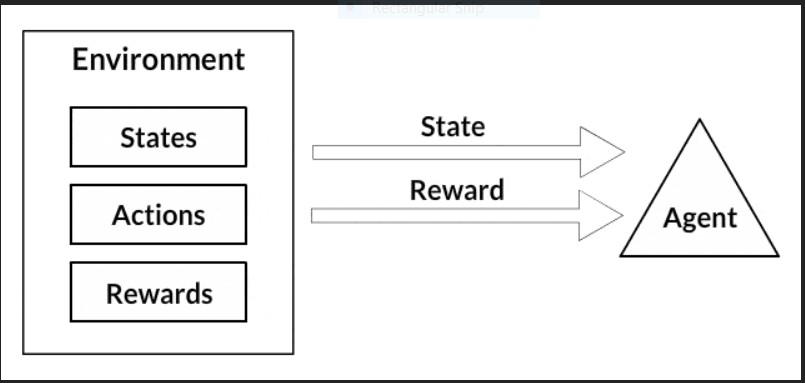


Fig. [2.8] step 1 of Reinforcement learning

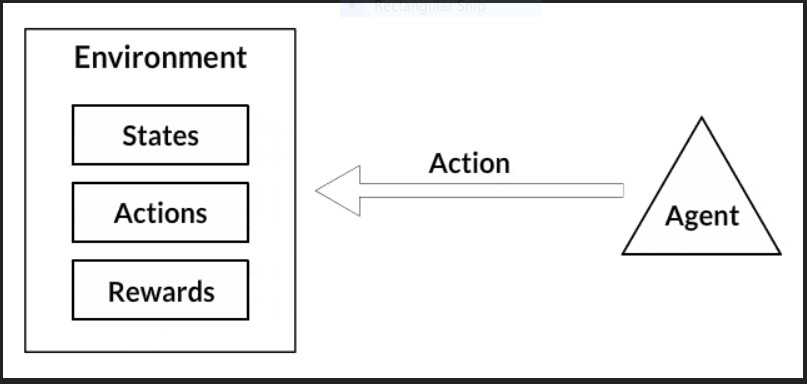


Fig. [2.9] step 2 of Reinforcement learning

This is very similar to how we as humans also learn. Throughout our lives, we receive positive and negative signals and constantly learn from them. The chemicals in our brain are one of many ways we get these signals. When something good happens, the neurons in our brains provide a hit of positive neurotransmitters such as dopamine which makes us feel good and we **become more likely to repeat that specific action**. We don’t need constant supervision to learn like in supervised learning. By only giving the occasional reinforcement signals, we still learn very effectively.

#### Examples

Google Deep Mind have used reinforcement learning in research to play Go and Atari games at superhuman levels.

# 3. Hardware, Software & Language used

* 1. **Hardware**
     + PC with Intel i3-3rd Gen Processor and 4 GB RAM
  2. **Software**
     + Windows 8
     + Spyder (Anaconda3)
  3. **Language**
     + Python 3

**3.4 Library**

* + - numpy
    - pandas
    - sklearn.preprocessing
    - sklearn.linear\_model
    - sklearn.model\_selection

**3.5 Dataset**

* + - Resistance dataset
    - Capacitance dataset
    - Inductance dataset
    - Quality Factor dataset
    - Damping Factor dataset
    - Resonance Frequency dataset
    - Bandwidth dataset



**4. Process**

**4.1 Procedural Steps**

Following steps were followed in this project:

1. Datasets are procured.

2. Datasets are divided into testing and training datasets.

3. ML models are created using ML algorithms and trained using training data.

4. ML models are tested using testing data through which accuracy of model is calculated.

5. When accuracy of model is greater than threshold then model is ready for predicting output.

1. Required output is calculated.

Flowchart

Check accuracy >= threshold accuracy

Delete values causing max error (top-5)

Calculate output

Model created

linear

ridge

Adjust the degree

Form ML Model

Data normalization

Data input

No

Yes

**5. Implementation**

**5.1 Codes**

**5.1.1 Resistance**

import pandas as pd

import numpy as np

train=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\training dataset.csv")

test=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\testing dataset.csv")

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression,Ridge

from sklearn.preprocessing import PolynomialFeatures

def datapreprocess(x1,values): *#to normalize data*

wltrain=train.drop(x1,axis=1)

Rtrain=train[x1]

scalar=StandardScaler().fit(wltrain,Rtrain)

wltrain=scalar.transform(wltrain)

values=scalar.transform(values)

Rtrain=np.array(Rtrain).reshape(-1,1)

wltest=test.drop(x1,axis=1)

wltest=scalar.transform(wltest)

Rtest=test[x1]

Rtest=np.array(Rtest).reshape(-1,1)

return wltrain,wltest,Rtest,Rtrain,values

def mlpredict(x1,values): *#to transfer normalize data obtained from datapreprocess to mlcalc*

wltrain,wltest,Rtest,Rtrain,values=datapreprocess(x1,values)

y=mlcalc(wltrain,wltest,Rtest,Rtrain,values)

return y

def mlcalc(wltrain,wltest,Rtest,Rtrain,values): *#for creating ML model*

*#Using Simple Linear Regression*

poly= PolynomialFeatures(5).fit(wltrain)

wltrain=poly.transform(wltrain)

wltest=poly.transform(wltest)

values=poly.transform(values)

i = 1

while i <10:

reg = LinearRegression().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor=reg.score(wltest,Rtest)

print("The accuracy score for Linear Regression is",scor,"/1")

*#Using Ridge Rigression*

i = 1

while i <10:

reg = Ridge().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor2=reg.score(wltest,Rtest)

print("The accuracy score for Ridge Regression is",scor2,"/1")

y=reg.predict(values)

return y

print("1. Wire-Length\n2. Wire-Width\n3. Temperature\n4. Resistance")

x1= input("Enter output string choice: ") *#taking input from user*

if(x1=='Wire-Length'):

w = float(input("Enter wire width b/w 2e-08 & 1.78e-07: "))

w2= float(input("Enter temperature b/w 9.91 & 101.51: "))

w3= float(input("Enter resistance b/w 22.68 & 486.27: "))

t="the wire length is:"

elif(x1=='Wire-Width'):

w = float(input("Enter wire length b/w 1.4e-06 & 8.84e-06: "))

w2= float(input("Enter temperature b/w 9.91 & 101.51: "))

w3= float(input("Enter resistance b/w 22.68 & 486.27: "))

t="the wire width is:"

elif(x1=='Temperature'):

w = float(input("Enter wire length b/w 1.4ee-08 & 1.78e-07: "))

w2= float(input("Enter wire width: "))

w3= float(input("Enter resistance: "))

t="the temperature is:"

else:

w = float(input("Enter wire length b/w 1.4e-06 & 8.84e-06: "))

w2= float(input("Enter wire width b/w 2e-08 & 1.78e-07: "))

w3= float(input("Enter temperature 9.91 & 101.51: "))

t="the resistance is:"

n=np.array([w,w2,w3]).reshape(1,-1)

y=mlpredict(x1,n)

print(t,y)

**5.1.2 Capacitance**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import PolynomialFeatures

data=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\cap1.csv")

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression,Ridge

def datapreprocess(x1,values):

label=data[x1]

data2=data.drop(x1,axis=1)

wltrain, wltest, Rtrain, Rtest = train\_test\_split(data2, label, test\_size=0.2)

scalar=StandardScaler().fit(wltrain,Rtrain)

wltrain=scalar.transform(wltrain)

values=scalar.transform(values)

Rtrain=np.array(Rtrain).reshape(-1,1)

wltest=scalar.transform(wltest)

Rtest=np.array(Rtest).reshape(-1,1)

return wltrain,wltest,Rtest,Rtrain,values

def mlpredict(x1,values):

wltrain,wltest,Rtest,Rtrain,values=datapreprocess(x1,values)

y=mlcalc(wltrain,wltest,Rtest,Rtrain,values)

return y

def mlcalc(wltrain,wltest,Rtest,Rtrain,values):

poly= PolynomialFeatures(3).fit(wltrain)

wltrain=poly.transform(wltrain)

wltest=poly.transform(wltest)

values=poly.transform(values)

#Using Simple Linear Regression

i = 1

while i <10:

reg = LinearRegression().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor=reg.score(wltest,Rtest)

print("The accuracy score for Linear Regression is",scor,"/1")

#Using Ridge Rigression

i = 1

while i <10:

reg = Ridge().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor2=reg.score(wltest,Rtest)

print("The accuracy score for Ridge Regression is",scor2,"/1")

y=reg.predict(values)

return y

print("1. Length\n2. Width\n3. Thickness\n4. Capacitance")

x1= input("Enter string choice: ")

if(x1=='Length'):

w = float(input("Enter width b/w 2.29e-08 & 1.76e-07: "))

w2= float(input("Enter thickness b/w 8.43e-10 & 1.14e-09: "))

w3= float(input("Enter capacitance b/w 1.77e-16 & 2.34e-15: "))

t="the length is:"

elif(x1=='Width'):

w = float(input("Enter length b/w 8.79e-07 & 8.85e-06: "))

w2= float(input("Enter thickness b/w 8.43e-10 & 1.14e-09: "))

w3= float(input("Enter capacitance 1.77e-16 & 2.34e-15: "))

t="the width is:"

elif(x1=='Thickness'):

w = float(input("Enter length b/w 8.79e-07 & 8.85e-06: "))

w2= float(input("Enter width b/w 2.29e-08 & 1.76e-07: "))

w3= float(input("Enter capacitance b/w 1.77e-16 & 2.34e-15: "))

t="the thickness is:"

elif(x1=='Capacitance'):

w = float(input("Enter length b/w 8.79e-07 & 8.85e-06: "))

w2= float(input("Enter width b/w 2.29e-08 & 1.76e-07: "))

w3= float(input("Enter thickness b/w 8.43e-10 & 1.14e-09: "))

t="the capacitance is:"

n=np.array([w,w2,w3]).reshape(1,-1)

y=mlpredict(x1,n)

print(t,y)

**5.1.3 Inductance**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import PolynomialFeatures

data=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\induc.csv")

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression,Ridge

def datapreprocess(x1,values):

label=data[x1]

data2=data.drop(x1,axis=1)

wltrain, wltest, Rtrain, Rtest = train\_test\_split(data2, label, test\_size=0.2)

scalar=StandardScaler().fit(wltrain,Rtrain)

wltrain=scalar.transform(wltrain)

print("the values is",values)

values=scalar.transform(values)

Rtrain=np.array(Rtrain).reshape(-1,1)

wltest=scalar.transform(wltest)

Rtest=np.array(Rtest).reshape(-1,1)

return wltrain,wltest,Rtest,Rtrain,values

def mlpredict(x1,values):

wltrain,wltest,Rtest,Rtrain,values=datapreprocess(x1,values)

y,scor=mlcalc(wltrain,wltest,Rtest,Rtrain,values)

return y,scor

def mlcalc(wltrain,wltest,Rtest,Rtrain,values):

#Using Simple Linear Regression

poly= PolynomialFeatures(4).fit(wltrain)

wltrain=poly.transform(wltrain)

wltest=poly.transform(wltest)

values=poly.transform(values)

i = 1

while i <10:

reg = LinearRegression().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor=reg.score(wltest,Rtest)

print("The accuracy score for Linear Regression is",scor,"/1")

y=reg.predict(values)

return y,scor

print("1. Wire-Length\n2. Wire-Width\n3. Thickness\n4. Inductance")

x1= input("Enter string choice: ")

if(x1=='Wire-Length'):

w = float(input("Enter wire width b/w 2e-08 & 1.78e-07: "))

w2= float(input("Enter thickness b/w 8.43e-10 & 1.14e-09: "))

w3= float(input("Enter inductance b/w 9.42e-10 & 1.28e-08: "))

t="the wire length is:"

elif(x1=='Wire-Width'):

w = float(input("Enter wire length b/w 1.4e-06 & 8.83e-06: "))

w2= float(input("Enter thickness b/w 8.43e-10 & 1.14e-09: "))

w3= float(input("Enter inductance b/w 9.42e-10 & 1.28e-08: "))

t="the wire width is:"

elif(x1=='Thickness'):

w = float(input("Enter wire length b/w 1.4e-06 & 8.83e-06: "))

w2= float(input("Enter wire width b/w 2e-08 & 1.78e-07: "))

w3= float(input("Enter inductance b/w 9.42e-10 & 1.28e-08: "))

t="the thickness is:"

elif(x1=='Inductance'):

w = float(input("Enter wire length b/w 1.4e-06 & 8.83e-06: "))

w2= float(input("Enter wire width b/w 2e-08 & 1.78e-07: "))

w3= float(input("Enter thickness b/w 8.43e-10 & 1.14e-09: "))

t="the inductance is:"

n=np.array([w,w2,w3]).reshape(1,-1)

y,scor=mlpredict(x1,n)

print(t,y)

**5.1.4 Quality Factor**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import PolynomialFeatures

data=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\qfactor.csv")

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression,Ridge

def datapreprocess(x1,values):

label=data[x1]

data2=data.drop(x1,axis=1)

wltrain, wltest, Rtrain, Rtest = train\_test\_split(data2, label, test\_size=0.2)

scalar=StandardScaler().fit(wltrain,Rtrain)

wltrain=scalar.transform(wltrain)

values=scalar.transform(values)

Rtrain=np.array(Rtrain).reshape(-1,1)

wltest=scalar.transform(wltest)

Rtest=np.array(Rtest).reshape(-1,1)

return wltrain,wltest,Rtest,Rtrain,values

def mlpredict(x1,values):

wltrain,wltest,Rtest,Rtrain,values=datapreprocess(x1,values)

y=mlcalc(wltrain,wltest,Rtest,Rtrain,values)

return y

def mlcalc(wltrain,wltest,Rtest,Rtrain,values):

poly= PolynomialFeatures(4).fit(wltrain)

wltrain=poly.transform(wltrain)

wltest=poly.transform(wltest)

values=poly.transform(values)

#Using Simple Linear Regression

i = 1

while i <10:

reg = LinearRegression().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor=reg.score(wltest,Rtest)

print("The accuracy score for Linear Regression is",scor,"/1")

#Using Ridge Rigression

i = 1

while i <10:

reg = Ridge().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor2=reg.score(wltest,Rtest)

print("The accuracy score for Ridge Regression is",scor2,"/1")

y=reg.predict(values)

return y

d=data.drop(['Inductance','Quality factor'],axis=1)

w3=float(input("Enter qfactor b/w 2 & 100: "))

x1="Inductance"

n=np.ones(10000)\*w3

d.insert(2, "Q", n, True)

y=mlpredict(x1,d)

y2=np.where(y>0)[0]

y=y[y2]

d=d.drop(['Q'],axis=1)

d=d.iloc[y2]

d.insert(2,"Inductance",y,True)

x1="Quality factor"

y=mlpredict(x1,d)

y2=np.abs(np.subtract(y,w3))

y2=np.argsort(y2,axis=None)[0:5]

d=d.iloc[y2]

y=y[y2]

d.insert(3, "Q-Factor", y, True)

print(d)

**5.1.5 Damping Factor**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import PolynomialFeatures

data=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\dfactor.csv")

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression,Ridge

def datapreprocess(x1,values):

label=data[x1]

data2=data.drop(x1,axis=1)

wltrain, wltest, Rtrain, Rtest = train\_test\_split(data2, label, test\_size=0.2)

scalar=StandardScaler().fit(wltrain,Rtrain)

wltrain=scalar.transform(wltrain)

values=scalar.transform(values)

Rtrain=np.array(Rtrain).reshape(-1,1)

wltest=scalar.transform(wltest)

Rtest=np.array(Rtest).reshape(-1,1)

return wltrain,wltest,Rtest,Rtrain,values

def mlpredict(x1,values):

wltrain,wltest,Rtest,Rtrain,values=datapreprocess(x1,values)

y=mlcalc(wltrain,wltest,Rtest,Rtrain,values)

return y

def mlcalc(wltrain,wltest,Rtest,Rtrain,values):

poly= PolynomialFeatures(4).fit(wltrain)

wltrain=poly.transform(wltrain)

wltest=poly.transform(wltest)

values=poly.transform(values)

#Using Simple Linear Regression

i = 1

while i <10:

reg = LinearRegression().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor=reg.score(wltest,Rtest)

print("The accuracy score for Linear Regression is",scor,"/1")

#Using Ridge Rigression

i = 1

while i <10:

reg = Ridge().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor2=reg.score(wltest,Rtest)

print("The accuracy score for Ridge Regression is",scor2,"/1")

y=reg.predict(values)

return y

d=data.drop(['Inductance','Damping Factor'],axis=1)

w3=float(input("Enter dfactor b/w 0.003 & 0.09: "))

x1="Inductance"

n=np.ones(10000)\*w3

d.insert(2, "D", n, True)

y=mlpredict(x1,d)

y2=np.where(y>0)[0]

y3=y

y=y[y2]

d=d.drop(['D'],axis=1)

d=d.iloc[y2]

d.insert(2,"Inductance",y,True)

x1="Damping Factor"

y=mlpredict(x1,d)

y2=np.abs(np.subtract(y,w3))

y2=np.argsort(y2,axis=None)[0:4]

d=d.iloc[y2]

y=y[y2]

d.insert(3, "D-Factor", y, True)

print(d)

**5.1.6 Resonance Frequency**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import PolynomialFeatures

data=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\Resfreq.csv")

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression,Ridge

def datapreprocess(x1,values):

label=data[x1]

data2=data.drop(x1,axis=1)

wltrain, wltest, Rtrain, Rtest = train\_test\_split(data2, label, test\_size=0.2)

scalar=StandardScaler().fit(wltrain,Rtrain)

wltrain=scalar.transform(wltrain)

values=scalar.transform(values)

Rtrain=np.array(Rtrain).reshape(-1,1)

wltest=scalar.transform(wltest)

Rtest=np.array(Rtest).reshape(-1,1)

return wltrain,wltest,Rtest,Rtrain,values

def mlpredict(x1,values):

wltrain,wltest,Rtest,Rtrain,values=datapreprocess(x1,values)

y=mlcalc(wltrain,wltest,Rtest,Rtrain,values)

return y

def mlcalc(wltrain,wltest,Rtest,Rtrain,values):

poly= PolynomialFeatures(4).fit(wltrain)

wltrain=poly.transform(wltrain)

wltest=poly.transform(wltest)

values=poly.transform(values)

#Using Simple Linear Regression

i = 1

while i <10:

reg = LinearRegression().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor=reg.score(wltest,Rtest)

print("The accuracy score for Linear Regression is",scor,"/1")

#Using Ridge Rigression

i = 1

while i <10:

reg = Ridge().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor2=reg.score(wltest,Rtest)

print("The accuracy score for Ridge Regression is",scor2,"/1")

y=reg.predict(values)

return y

d=data.drop(['Inductance','Resonance Frequency'],axis=1)

d=d.Capacitance.unique()

d=pd.DataFrame(data=d,columns=["Capacitance"])

w3=float(input("Enter resfreq b/w 3.6e+10 & 2e+11: "))

x1="Inductance"

n=np.ones(d.size)\*w3

d.insert(1, "RF", n, True)

y=mlpredict(x1,d)

y2=np.where(y>0)[0]

y3=y

y=y[y2]

d=d.drop(['RF'],axis=1)

d=d.iloc[y2]

d.insert(1,"Inductance",y,True)

x1="Resonance Frequency"

y=mlpredict(x1,d)

y2=np.abs(np.subtract(y,w3))

y2=np.argsort(y2,axis=None)[0:4]

d=d.iloc[y2]

y=y[y2]

d.insert(2, "Res Freq", y, True)

print(d)

**5.1.7 Bandwidth**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import PolynomialFeatures

data=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\band.csv")

train=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\BWtrain.csv")

test=pd.read\_csv("C:\\Users\\rajesh\\Desktop\\BWtest.csv")

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression,Ridge

def datapreprocess(x1,values):

wltrain=train.drop(x1,axis=1)

Rtrain=train[x1]

wltest=test.drop(x1,axis=1)

Rtest=test[x1]

scalar=StandardScaler().fit(wltrain,Rtrain)

wltrain=scalar.transform(wltrain)

values=scalar.transform(values)

Rtrain=np.array(Rtrain).reshape(-1,1)

wltest=scalar.transform(wltest)

Rtest=np.array(Rtest).reshape(-1,1)

return wltrain,wltest,Rtest,Rtrain,values

def mlpredict(x1,values):

wltrain,wltest,Rtest,Rtrain,values=datapreprocess(x1,values)

y=mlcalc(wltrain,wltest,Rtest,Rtrain,values)

return y

def mlcalc(wltrain,wltest,Rtest,Rtrain,values):

poly= PolynomialFeatures(8).fit(wltrain)

wltrain=poly.transform(wltrain)

wltest=poly.transform(wltest)

values=poly.transform(values)

#Using Simple Linear Regression

i = 1

while i <10:

reg = LinearRegression().fit(wltrain,Rtrain)

y=reg.predict(wltrain)

y2=np.absolute(np.subtract(Rtrain,y))

y3=np.argsort(y2,axis=0)

y3=y3[-5:]

Rtrain=np.delete(Rtrain,y3).reshape(-1,1)

wltrain=np.delete(wltrain,y3,axis=0)

i=i+1

scor=reg.score(wltest,Rtest)

print("The accuracy score for Linear Regression is",scor,"/1")

y=reg.predict(values)

return y

d=data.drop(['Inductance','Bandwidth'],axis=1)

d=d.Resistance.unique()

d=pd.DataFrame(data=d,columns=["Resistance"])

w3=float(input("Enter bandwidth b/w 2.6e+09 & 1.5e+11: "))

x1="Inductance"

n=np.ones(d.size)\*w3

d.insert(1, "BW", n, True)

y=mlpredict(x1,d)

y2=np.where(y>0)[0]

y3=y

y=y[y2]

d=d.drop(['BW'],axis=1)

d=d.iloc[y2]

d.insert(1,"Inductance",y,True)

x1="Bandwidth"

y=mlpredict(x1,d)

y2=np.abs(np.subtract(y,w3))

y2=np.argsort(y2,axis=None)[0:4]

d=d.iloc[y2]

y=y[y2]

d.insert(2, "bandwidth", y, True)

print(d)

**6. Result and Output**

**6.1 Results**

Following can be inferred from the above outputs:

1. In the Resistance code, the ML model can predict desired value with up to 95% accuracy.

2. In the Capacitance code, the ML model can predict desired value with up to 99% accuracy.

3. In the Inductance code, the ML model can predict desired value with up to 99% accuracy.

4. In the Quality factor code, the ML model can predict desired value with up to 99% accuracy.

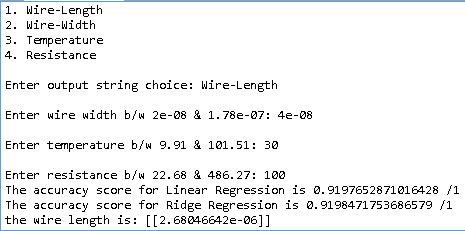
5. In the Damping factor code, the ML model can predict desired value with up to 99% accuracy.

6. In the Resonance frequency code, the ML model can predict desired value with up to 99% accuracy.

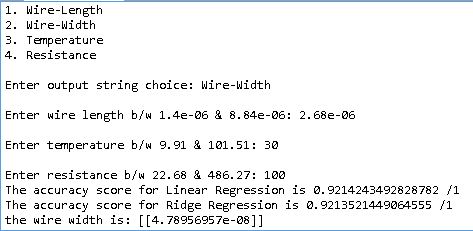
7. In the Bandwidth code, the ML model can predict desired value with up to 99% accuracy.

**6.2 Outputs**

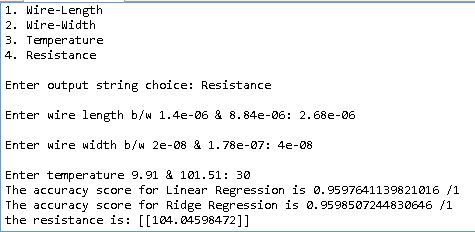
**6.2.1 Resistance**



Output-1

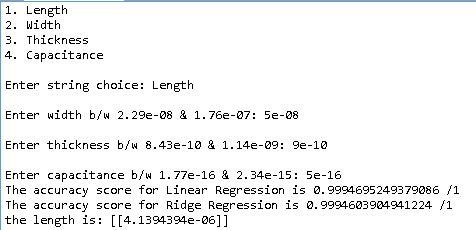


Output-2

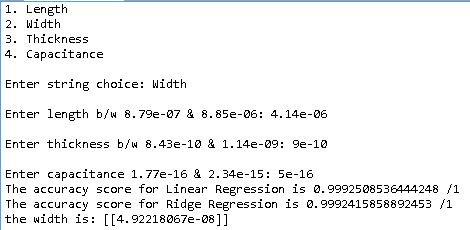


Output-3

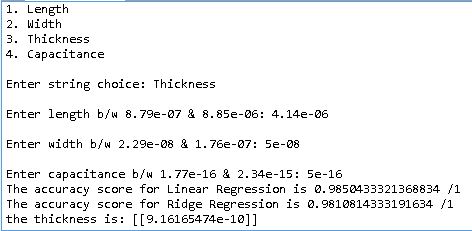
**6.2.2 Capacitance**



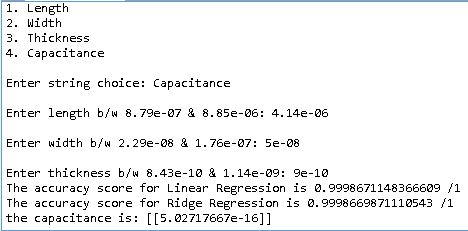
Output-4



Output-5

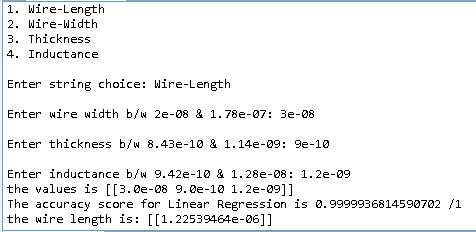


Output-6

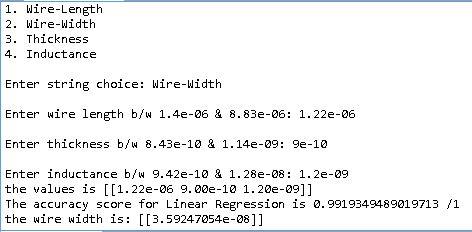


Output-7

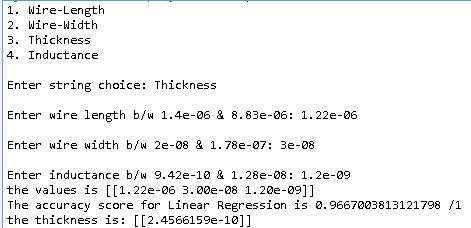
**6.2.3 Inductance**



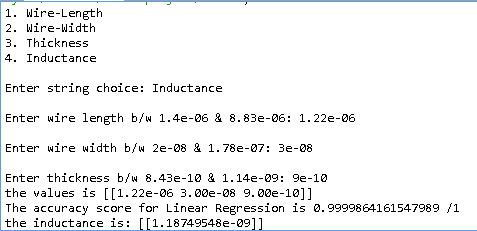
Output-8



Output-9

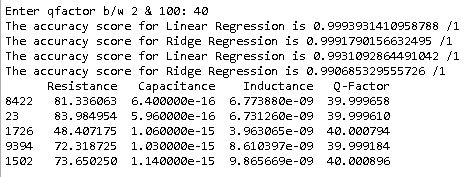


Output-10



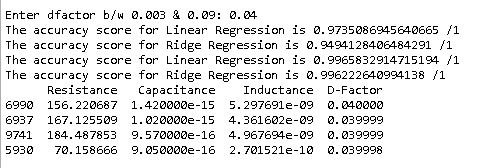
Output-11

**6.2.4 Quality Factor**



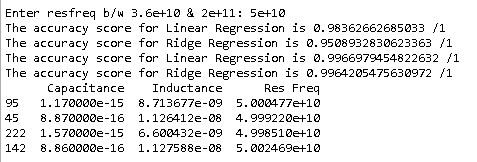
Output-12

**6.2.5 Damping Factor**



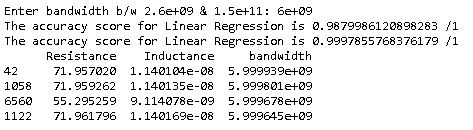
Output-13

**6.2.6 Resonance Frequency**



Output-14

**6.2.7 Bandwidth**



Output-15

**FUTURE SCOPE**

This project can be very helpful in designing much more complex circuits with many more circuit elements if good quality and large datasets can be procured and feeded to the Machine Learning Algorithms.

This project can also be used to design suitable circuit elements with respect to many different material properties and atmospheric conditions if a relationship can be established between them or good quality and large datasets can be obtained based on different variables.

These circuit elements can then be used to design desirable circuits depending on customer needs with optimum efficiency.

**CONCLUSION**

We have considered the task of designing different electrical components such as Resistance, Capacitance and Inductance based on different material properties such as Wire Length, Width & Thickness(for capacitance & inductance) and Temperature(for resistance) using machine learning algorithms.

We have also considered the task of designing different RLC circuits based on different properties such as Quality factor, Damping factor, Resonance frequency and Bandwidth using machine learning.

This project can be very useful to electrical circuit designing companies as it can learn from a large dataset and predict values for different components over a million possible values feeded by the user using machine learning.

**REFERENCES**

[1]For Resistance dataset: [https://www.kaggle.com/c/vlsi-wire-resistance- estimation/data](https://www.kaggle.com/c/vlsi-wire-resistance-%20%20estimation/data)

[2]For Capacitance dataset:

<https://github.com/srohit0/mida/blob/master/notebooks/Capacitance%20%20Estimation.ipynb>

[3]For Inductance dataset: <https://chemandy.com/calculators/flat-wire-inductor-calculator.htm>

[4]Gulden Kaya Uyanik, Nese Guler: A study on multiple linear regression analysis. [https://sciencedirect.com](https://sciencedirect.com/)

[5]**Khushbu Kumari, Suniti Yadav: Linear Regression Analysis Study**

**[6]Donald W. Marquardt, Ronald D. Snee: Ridge Regression in Practice**