Predicting optimal 3D printing material using machine learning

Submitted by
Gayathry U
Arjun Sajan
Group : 27
Union Christian College

TABLE OF CONTENTS: -

1.INTRODUCTION

1.1OVERVIEW

The 3D printing materials industry is increasing due to the rise in the demand from healthcare, automotive, and other industries, globally. The 3D printing materials market comprises several stakeholders, such as raw material suppliers, processors, end-product manufacturers, and regulatory organizations in the supply chain. The demand side of this market is characterized by the development of various industries such as aerospace & defense, healthcare, consumer goods, and automotive. Advancements in technology and diverse applications characterize the supply side. Various primary sources from both the supply and demand sides of the market were interviewed to obtain qualitative and quantitative information.

1.2PURPOSE

Predicting material would be more suitable for making the 3D model. In this project, the input parameters are like Layer Height (mm), Wall Thickness (mm), Infill Density (%), Infill Pattern (honeycomb, grid), Nozzle Temperature (Cº), Bed Temperature (Cº), Print Speed(mm/s), Fan Speed (%), Roughness (µm), Tension (ultimate), Strength (MPa), Elongation (%).
Based on these parameters a supervised machine learning model is built to predict the best material to be used for building 3D models. A web application is build so that the user can type in the mentioned parameters and the material which suits the best is showcased on UI

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

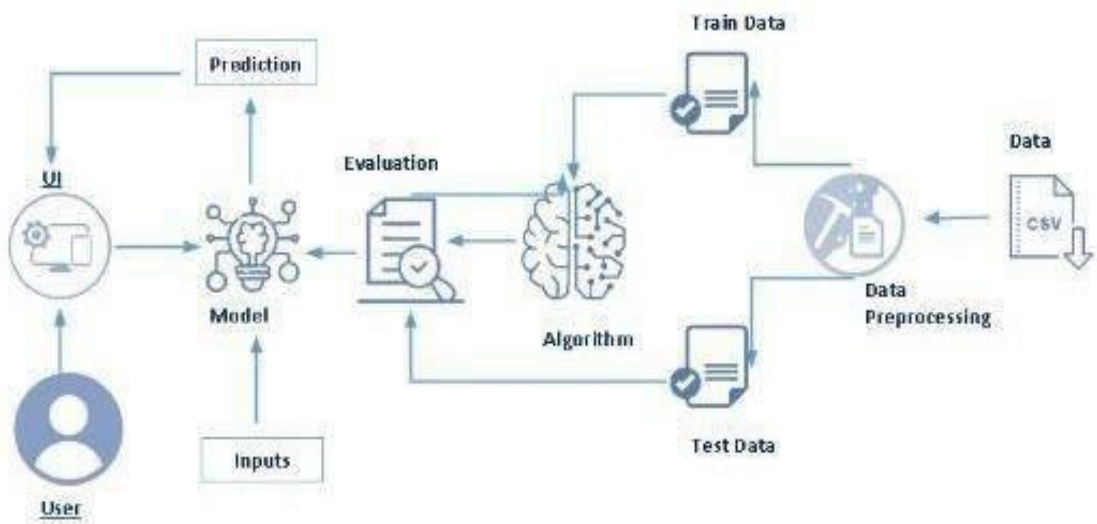
While 3D printing allows engineers to produce single items inexpensively, it sometimes comes at a cost to quality. Aside from high-end machines that costs millions of dollars to purchase, many 3D printers produce good that are inferior to those made through traditional manufacturing. One of the reasons for this is a lack of universal standards.
“Put simply, many manufacturers and end users have difficulty stating with certainty that parts or products produced via 3D printing—whether all on the same printer or across geographies—will be of consistent quality, strength, and reliability,” . “Without this guarantee, many manufacturers will remain leery of AM technology, judging the risks of uncertain quality to be too costly a trade-off for any gains they might realize.”

2.2 PROPOSED SYSTEM

The merging of artificial intelligence and 3D printing is an evolution of manufacturing paradigms. Prosthetics design, for instance, is one of the most important applications of 3D printing. As technology advances, artificial intelligence and 3D printers can be used to control 3D printers and increase the number of compatible materials for the process. By combining these two technologies, manufacturers can create new and improved products and production processes. Artificial intelligence and 3D printing will eventually help humans create better prosthetics.

3.THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE AND SOFTWARE DESIGNING

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was created by Guido van Rossum , and first released on February 20, 1991. Its high-level built in data structures, combined with dynamic typing and dynamic binding , make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Anaconda Navigator

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, crossplatform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder.

Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use

Spyder

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features. Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.Spyder is extensible with first-party and third party plugins includes support for interactive tools for data inspection and embeds Pythonspecific code. Spyder is also pre-installed in Anaconda Navigator, which is included in Anaconda.

Flask

Web frame work used for building. It is a web application framework written in python which will be running in local browser with a user interface. In this application, whenever the user interacts with UI and selects emoji, it will suggest the best and top movies of that genre to the use.

Hardware Requirements:

o Operating system: window 7 and above with 64bit o Processor Type -Intel Core i3-3220

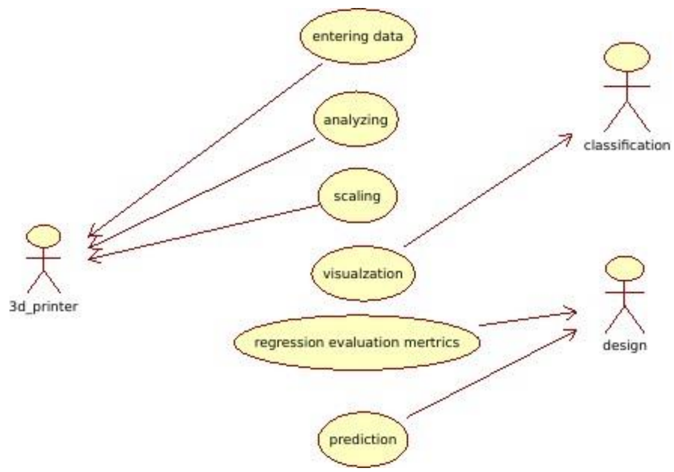
o RAM: 4Gb and above

o Hard disk: min 100GB

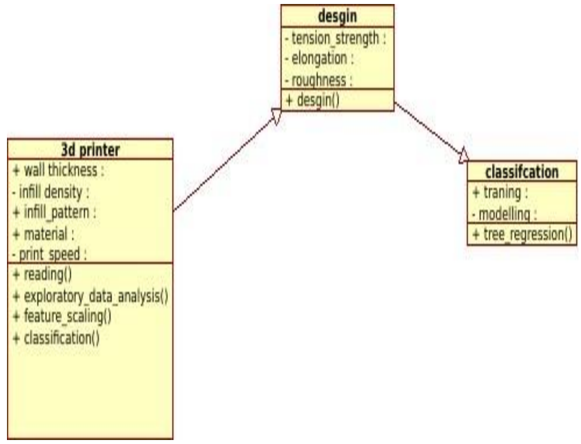
4.EXPERIMENTAL INVESTIGATION

Here we are going to build a machine learning model that predicts whether the given message is a spam or not, based on these parameters a supervised machine learning model is built to predict the best material to be used for building 3D models. A web application is build so that the user can type in the mentioned part a meters and the material which suits the best is showcased on UI.

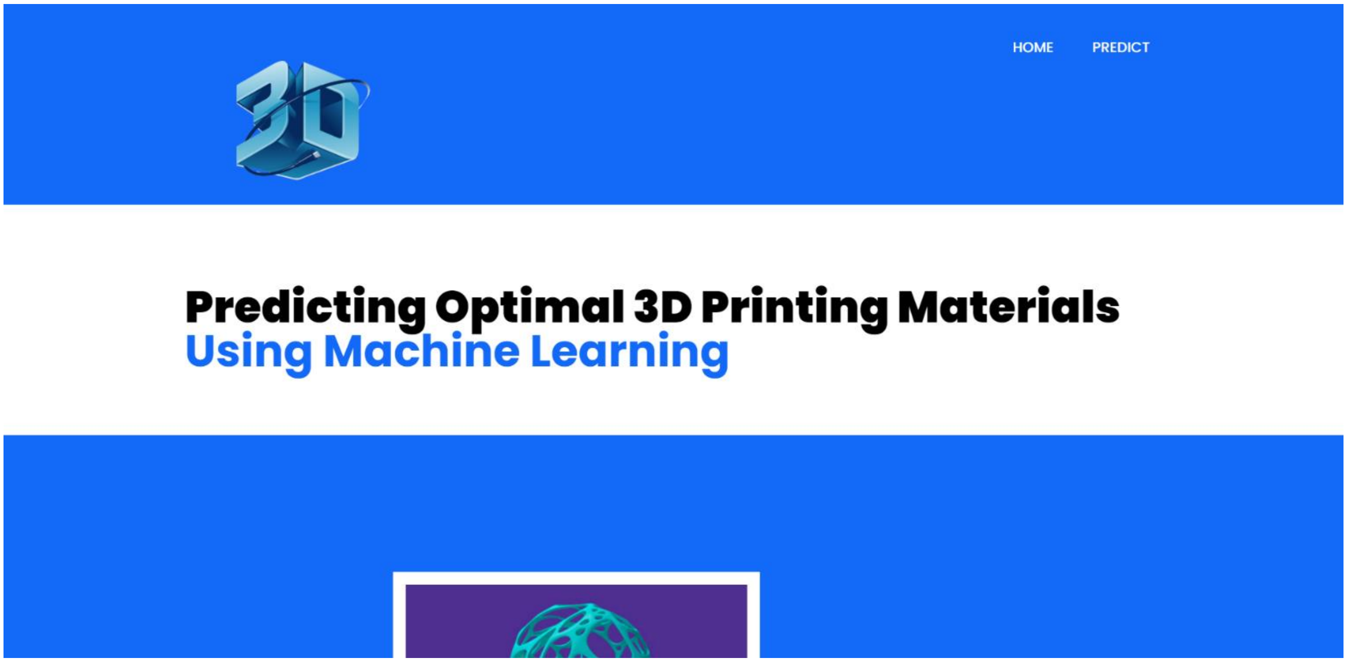
5.FLOWCHART

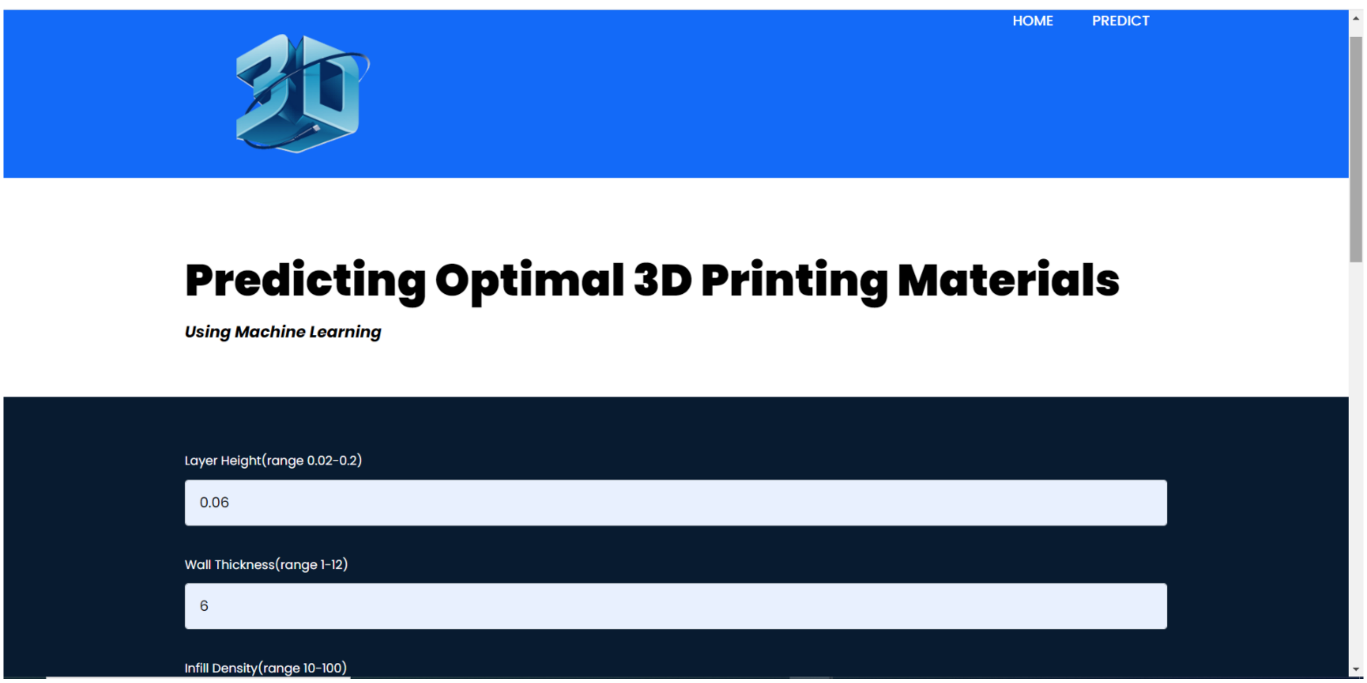


USE CASE DIAGRAM



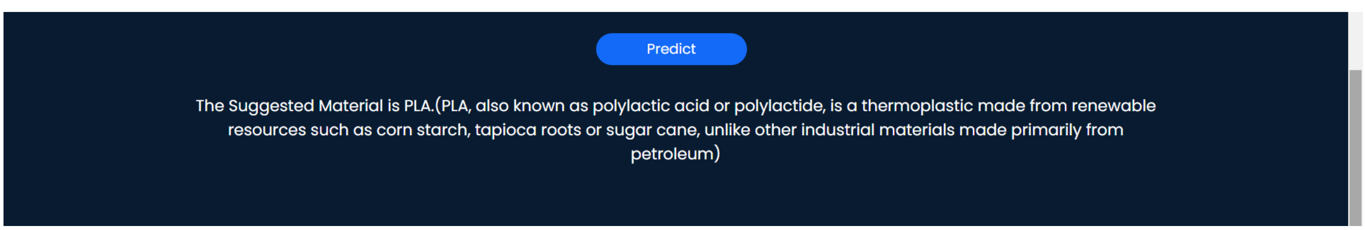
6.RESULT


HOME
PREDICT
Predicting Optimal 3D Printing Materials
Using Machine Learning


HOME
PREDICT
Predicting Optimal 3D Printing Materials
Using Machine Learning
Layer Height(range 0.02-0.2)
0.06
Wall Thickness(range 1-12)
6
Infill Density(range 10-100)


Infill Density(range 10-100)
50
Infill Pattern: 0 for grid, 1 for honeycomb
0
Nozzel Temperature(range 200-250)
230
Bed Temperature(range 60-100)
80
Pirnt Speed(range 40-120)
40
Fan Speed(range 0-100)
100


Roughness(range 25-369)
98
Tension Strength(range 5-40)
15
Elongation(0.95-2.9)
0.95
Predict


Predict
The Suggested Material is PLA.(PLA, also known as polylactic acid or polylactide, is a thermoplastic made from renewable resources such as corn starch, tapioca roots or sugar cane, unlike other industrial materials made primarily from petroleum)

7.ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Easy to use
- Cost efficient
- Time efficient

DISADVANTAGE

1. Initial costs of printer
2. Post processing
3. Printing time
4. Special skill required for 3D models
5. Manufacturing Job Losses

8. Applications

3D printing has gone through a number of changes over the years. In the early days, 3D printing was time-consuming and costly, and not very practical for applications outside of industry. However, with the advent of today’s more flexible and cost-effective 3D printing methods, there are areas where 3D printing has become a practical tool.

It is applicable in different sectors such as

- Engineering And Design
- Consumer products
- Manufacturing
- Education
- Aerospace
- Medical
- Movies / Theatres
- Architectures

9. CONCLUSION

3D printing technology could revolutionize and re-shape the world. Advance in 3D technology can significantly change and improve the way we manufacture products goods worldwide.
If the last industrial revolution brought us mass production and the advent of economics of scale – the digital 3D printing revolution could bring mass manufacturing back a full of circle – to an era of mass personalization, and return to individual craftsmanship.

10. FUTURE SCOPE

Future applications for 3D printing might include creating open-source scientific equipment to create opensource labs

Science-based applications like reconstructing fossils in palaeontology . Replicating ancient and priceless artifacts in archaeology

Reconstructing bones and body parts in forensic pathology. The technology currently being researched for building construction.

11. BIBILOGRAPHY

- http://mashable.com/2014/03/06/3d-printed-blood-vessels/
- http://www.3dprinter.net/

12.APPENDIX

In [1]:
```
##Importing libraries
import pandas as  pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

In [2]:
```
##Loading the dataset

ds=pd.read_csv(r'../dataset/3D_printer.csv')
```

In [3]:
```
##Printing the first five rows
ds.head()
```

Out[3]:

layer_height	wall_thickness	infill_density	infill_pattern	nozzle_temperature	bed_temperature	print_speed	material	fan_speed	roughness	tension_strenght	elo
0.02	8.0	90	grid	220	60	40	abs	0	25	18	
0.02	7.0	90	honeycomb	225	65	40	abs	25	32	16	
0.02	1.0	80	grid	230	70	40	abs	50	40	8	
0.02	4.0	70	honeycomb	240	75	40	abs	75	68	10	
0.02	6.0	90	grid	250	80	40	abs	100	92	5	

In [4]:
```
ds.tail()
```

Out[4]:

	layer_height	wall_thickness	infill_density	infill_pattern	nozzle_temperature	bed_temperature	print_speed	material	fan_speed	roughness	tension_strengh
61	0.06	9.0	10	honeycomb	200	75	80	abs	75	200	9
62	0.04	2.0	80	grid	230	70	40	abs	50	40	12
63	0.02	4.5	70	honeycomb	240	85	40	abs	75	68	10
64	0.05	6.0	10	honeycomb	245	75	85	abs	75	205	5
65	0.15	1.0	50	grid	220	60	120	abs	0	120	16

In [5]:
```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66 entries, 0 to 65
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   layer_height        66 non-null     float64
 1   wall_thickness      66 non-null     float64
 2   infill_density      66 non-null     int64
 3   infill_pattern      66 non-null     object
 4   nozzle_temperature  66 non-null     int64
 5   bed_temperature     66 non-null     int64
 6   print_speed         66 non-null     int64
 7   material            66 non-null     object
 8   fan_speed           66 non-null     int64
 9   roughness           66 non-null     int64
 10  tension_strenght    66 non-null     int64
 11  elongation          66 non-null     float64
dtypes: float64(3), int64(7), object(2)
memory usage: 6.3+ KB
```

In [6]:
```
##descripive statistics
ds.describe()
```

Out[6]:

	layer_height	wall_thickness	infill_density	nozzle_temperature	bed_temperature	print_speed	fan_speed	roughness	tension_strenght	elongation
count	66.000000	66.000000	66.000000	66.000000	66.000000	66.000000	66.000000	66.000000	66.000000	66.000000
mean	0.098182	5.583333	54.727273	222.272727	70.378788	64.242424	48.530303	160.545455	19.757576	1.625000
std	0.062608	2.952943	27.545512	15.094110	8.651839	28.598580	35.834328	95.703899	9.202108	0.762498
min	0.020000	1.000000	10.000000	200.000000	60.000000	40.000000	0.000000	21.000000	4.000000	0.400000
25%	0.052500	3.000000	40.000000	210.000000	65.000000	40.000000	25.000000	78.250000	12.000000	1.025000
50%	0.100000	6.000000	50.000000	220.000000	70.000000	60.000000	50.000000	149.500000	18.500000	1.500000
75%	0.150000	8.000000	80.000000	230.000000	75.000000	60.000000	75.000000	220.000000	27.000000	2.175000
max	0.200000	12.000000	100.000000	250.000000	100.000000	120.000000	100.000000	368.000000	38.000000	3.300000

In [7]:
```
##corr among the data
ds.corr()
```

Out[7]:

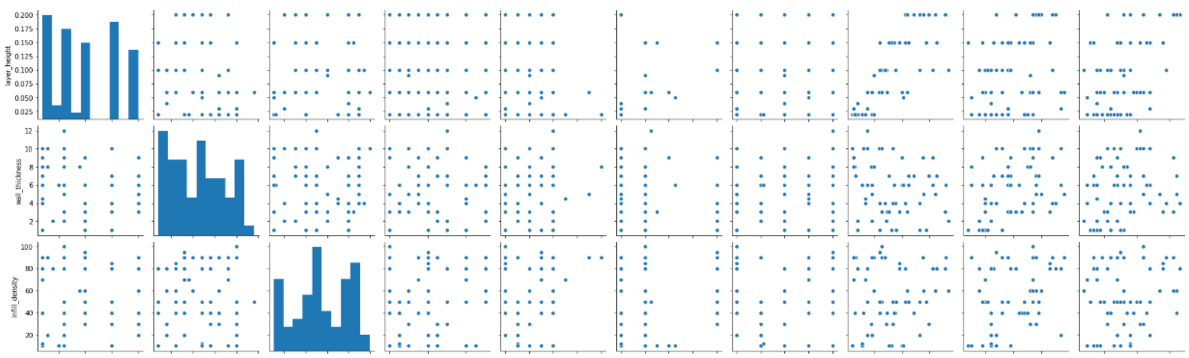
	layer_height	wall_thickness	infill_density	nozzle_temperature	bed_temperature	print_speed	fan_speed	roughness	tension_strenght	elon
layer_height	1.000000	-0.282933	-0.013763	-0.030562	-0.120838	0.044329	-0.040571	0.773096	0.325276	0.4
wall_thickness	-0.282933	1.000000	0.025534	-0.130299	0.061974	-0.341273	0.050462	-0.240834	0.336492	0.1
infill_density	-0.013763	0.025534	1.000000	0.213167	0.119221	-0.048114	0.035763	0.037378	0.278869	0.1
nozzle_temperature	-0.030562	-0.130299	0.213167	1.000000	0.552889	0.031671	0.580967	0.302494	-0.392501	-0.5
bed_temperature	-0.120838	0.061974	0.119221	0.552889	1.000000	-0.067218	0.906690	0.106675	-0.247139	-0.3
print_speed	0.044329	-0.341273	-0.048114	0.031671	-0.067218	1.000000	-0.000353	0.212711	-0.195963	-0.2
fan_speed	-0.040571	0.050462	0.035763	0.580967	0.906690	-0.000353	1.000000	0.202488	-0.299644	-0.3
roughness	0.773096	-0.240834	0.037378	0.302494	0.106675	0.212711	0.202488	1.000000	0.038829	0.0
tension_strenght	0.325276	0.336492	0.278869	-0.392501	-0.247139	-0.195963	-0.299644	0.038829	1.000000	0.8
elongation	0.482438	0.150234	0.118003	-0.524996	-0.310455	-0.213770	-0.347389	0.073683	0.834834	1.0

In [8]:
```python
##Finding null values
ds.isnull().any()
```

Out[8]:
```
layer_height          False
wall_thickness        False
infill_density        False
infill_pattern        False
nozzle_temperature    False
bed_temperature       False
print_speed           False
material              False
fan_speed             False
roughness             False
tension_strenght      False
elongation            False
dtype: bool
```

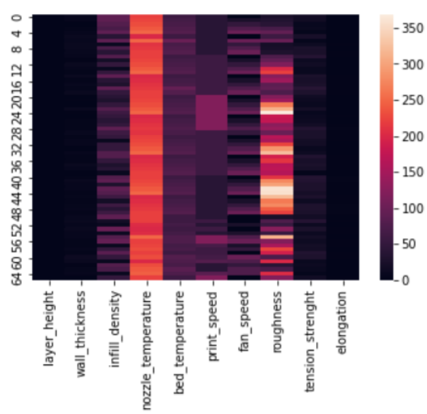
In [9]:
```python
##Seaborn Pairplot
##plots a pairwise relationships in a dataset
sns.pairplot(ds)
```

Out[9]: `<seaborn.axisgrid.PairGrid at 0x27b2c3d7280>`



In [10]:
```python
##Seaborn Heatmap
##A way of representing the data in 2-D form
sns.heatmap(ds[['layer_height','wall_thickness','infill_density','nozzle_temperature','bed_temperature','print_speed','fan_speed'
```

Out[10]: `<matplotlib.axes._subplots.AxesSubplot at 0x27b306c9640>`



In [11]:
```python
##Label Emcoding
from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

ds=ds.iloc[:,:].values
```

In [12]:
```python
ds[:,3]=lb.fit_transform(ds[:,3])
```

```
In [12]: ds[:,3]=lb.fit_transform(ds[:,3])

         ds[:,7]=lb.fit_transform(ds[:,7])

In [13]: da=pd.DataFrame(ds)

In [14]: y=ds[:,7]

         y=y.astype("int")

In [15]:
         da.drop(columns=7,inplace=True)

In [16]: x=da.iloc[:,:].values

         x
```

```
Out[16]: array([[0.02, 8.0, 90, 0, 220, 60, 40, 0, 25, 18, 1.2],
                [0.02, 7.0, 90, 1, 225, 65, 40, 25, 32, 16, 1.4],
                [0.02, 1.0, 80, 0, 230, 70, 40, 50, 40, 8, 0.8],
                [0.02, 4.0, 70, 1, 240, 75, 40, 75, 68, 10, 0.5],
                [0.02, 6.0, 90, 0, 250, 80, 40, 100, 92, 5, 0.7],
                [0.02, 10.0, 40, 1, 200, 60, 40, 0, 60, 24, 1.1],
                [0.02, 8.0, 90, 0, 250, 100, 40, 100, 98, 5, 0.95],
                [0.02, 10.0, 10, 1, 210, 70, 40, 50, 21, 14, 1.5],
                [0.02, 9.0, 70, 0, 215, 75, 40, 75, 24, 27, 1.4],
                [0.02, 8.0, 40, 1, 220, 80, 40, 100, 30, 25, 1.7],
                [0.06, 6.0, 80, 0, 220, 60, 60, 0, 75, 37, 2.4],
                [0.06, 2.0, 20, 1, 225, 65, 60, 25, 92, 12, 1.4],
                [0.06, 10.0, 50, 0, 230, 70, 60, 50, 118, 16, 1.3],
                [0.06, 6.0, 10, 1, 240, 75, 60, 75, 200, 9, 0.8],
                [0.06, 3.0, 50, 0, 250, 80, 60, 100, 220, 10, 1.0],
                [0.06, 10.0, 90, 1, 200, 60, 60, 0, 126, 27, 2.2],
                [0.06, 3.0, 40, 0, 205, 65, 60, 25, 145, 23, 1.9],
                [0.06, 8.0, 30, 1, 210, 70, 60, 50, 88, 26, 1.6],
                [0.06, 5.0, 90, 0, 215, 95, 60, 75, 92, 38, 2.2],
                [0.06, 10.0, 50, 1, 220, 80, 60, 100, 74, 29, 2.0],
```


```
In [16]: x=da.iloc[:,:].values

         x

Out[16]: array([[0.02, 8.0, 90, 0, 220, 60, 40, 0, 25, 18, 1.2],
                [0.02, 7.0, 90, 1, 225, 65, 40, 25, 32, 16, 1.4],
                [0.02, 1.0, 80, 0, 230, 70, 40, 50, 40, 8, 0.8],
                [0.02, 4.0, 70, 1, 240, 75, 40, 75, 68, 10, 0.5],
                [0.02, 6.0, 90, 0, 250, 80, 40, 100, 92, 5, 0.7],
                [0.02, 10.0, 40, 1, 200, 60, 40, 0, 60, 24, 1.1],
                [0.02, 8.0, 90, 0, 250, 100, 40, 100, 98, 5, 0.95],
                [0.02, 10.0, 10, 1, 210, 70, 40, 50, 21, 14, 1.5],
                [0.02, 9.0, 70, 0, 215, 75, 40, 75, 24, 27, 1.4],
                [0.02, 8.0, 40, 1, 220, 80, 40, 100, 30, 25, 1.7],
                [0.06, 6.0, 80, 0, 220, 60, 60, 0, 75, 37, 2.4],
                [0.06, 2.0, 20, 1, 225, 65, 60, 25, 92, 12, 1.4],
                [0.06, 10.0, 50, 0, 230, 70, 60, 50, 118, 16, 1.3],
                [0.06, 6.0, 10, 1, 240, 75, 60, 75, 200, 9, 0.8],
                [0.06, 3.0, 50, 0, 250, 80, 60, 100, 220, 10, 1.0],
                [0.06, 10.0, 90, 1, 200, 60, 60, 0, 126, 27, 2.2],
                [0.06, 3.0, 40, 0, 205, 65, 60, 25, 145, 23, 1.9],
                [0.06, 8.0, 30, 1, 210, 70, 60, 50, 88, 26, 1.6],
                [0.06, 5.0, 90, 0, 215, 95, 60, 75, 92, 38, 2.2],
                [0.06, 10.0, 50, 1, 220, 80, 60, 100, 74, 29, 2.0],
```


```
                [0.1, 1.0, 40, 0, 220, 60, 120, 0, 120, 16, 1.2],
                [0.1, 2.0, 30, 1, 225, 65, 120, 25, 144, 12, 1.1],
                [0.1, 1.0, 50, 0, 230, 70, 120, 50, 265, 10, 0.9],
                [0.1, 9.0, 80, 1, 240, 75, 120, 75, 312, 19, 0.8],
                [0.1, 2.0, 60, 0, 250, 80, 120, 100, 368, 8, 0.4],
                [0.1, 1.0, 50, 1, 200, 60, 120, 0, 180, 11, 1.6],
                [0.1, 4.0, 40, 0, 205, 65, 120, 25, 176, 12, 1.2],
                [0.1, 3.0, 50, 1, 210, 70, 120, 50, 128, 18, 1.8],
                [0.1, 4.0, 90, 0, 215, 75, 120, 75, 138, 34, 2.9],
                [0.09, 8.0, 60, 1, 210, 70, 60, 50, 98, 26, 1.6],
                [0.15, 4.0, 50, 0, 220, 60, 60, 0, 168, 27, 2.4],
                [0.15, 7.0, 10, 1, 225, 65, 60, 25, 154, 19, 1.8],
                [0.15, 6.0, 50, 0, 230, 70, 60, 50, 225, 18, 1.4],
                [0.15, 1.0, 50, 1, 240, 75, 60, 75, 289, 9, 0.6],
                [0.15, 7.0, 80, 0, 250, 80, 60, 100, 326, 13, 0.7],
                [0.15, 3.0, 80, 1, 200, 60, 60, 0, 192, 33, 2.8],
```

In [17]:
```
# TRAIN TEST SPLIT

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [18]:
```
# FEATURE SCALING

from sklearn.preprocessing import MinMaxScaler

sc=MinMaxScaler()
```

In [19]:
```
x_train
```

Out[19]:
```
array([[0.15, 1.0, 50, 0, 220, 60, 120, 0, 120, 16, 1.5],
       [0.02, 1.0, 80, 0, 230, 70, 40, 50, 40, 8, 0.8],
       [0.06, 2.0, 20, 1, 225, 65, 60, 25, 92, 12, 1.4],
       [0.15, 4.0, 50, 0, 220, 60, 60, 0, 168, 27, 2.4],
       [0.06, 6.0, 80, 0, 220, 60, 60, 0, 75, 37, 2.4],
       [0.1, 1.0, 50, 0, 230, 70, 120, 50, 265, 10, 0.9],
       [0.2, 9.0, 90, 1, 225, 65, 40, 25, 276, 34, 3.1],
       [0.05, 6.0, 10, 1, 245, 75, 85, 75, 205, 5, 0.5],
       [0.15, 6.0, 50, 0, 230, 70, 60, 50, 225, 18, 1.4],
       [0.02, 10.0, 10, 1, 210, 70, 40, 50, 21, 14, 1.5],
       [0.06, 3.0, 50, 0, 250, 80, 60, 100, 220, 10, 1.0],
       [0.1, 4.0, 40, 0, 205, 65, 120, 25, 176, 12, 1.2],
       [0.1, 3.0, 50, 1, 210, 70, 120, 50, 128, 18, 1.8],
       [0.1, 4.0, 95, 0, 220, 75, 120, 100, 121, 14, 1.5],
       [0.2, 7.0, 30, 0, 230, 70, 40, 50, 298, 28, 2.2],
       [0.2, 6.0, 90, 1, 240, 75, 40, 75, 360, 28, 1.6],
       [0.06, 12.0, 50, 1, 230, 80, 65, 100, 74, 29, 2.1],
       [0.06, 5.0, 90, 0, 215, 95, 60, 75, 92, 38, 2.2],
       [0.04, 2.0, 80, 0, 230, 70, 40, 50, 40, 12, 0.8],
       [0.06, 10.0, 90, 1, 200, 60, 60, 0, 126, 27, 2.2],
       [0.02, 10.0, 40, 1, 200, 60, 40, 0, 60, 24, 1.1],
       [0.06, 3.0, 40, 0, 205, 65, 60, 25, 145, 23, 1.9],
```

In [20]:
```
x_train=sc.fit_transform(x_train)
```

In [21]:
```
x_train
```

Out[21]:
```
array([[0.72222222, 0.        , 0.44444444, 0.        , 0.4       ,
        0.        , 1.        , 0.        , 0.28530259, 0.35294118,
        0.39285714],
       [0.        , 0.        , 0.77777778, 0.        , 0.6       ,
        0.25      , 0.        , 0.5       , 0.05475504, 0.11764706,
        0.14285714],
       [0.22222222, 0.09090909, 0.11111111, 1.        , 0.5       ,
        0.125     , 0.25      , 0.25      , 0.20461095, 0.23529412,
        0.35714286],
       [0.72222222, 0.27272727, 0.44444444, 0.        , 0.4       ,
        0.        , 0.25      , 0.        , 0.42363112, 0.67647059,
        0.71428571],
       [0.22222222, 0.45454545, 0.77777778, 0.        , 0.4       ,
        0.        , 0.25      , 0.        , 0.1556196 , 0.97058824,
        0.71428571],
       [0.44444444, 0.        , 0.44444444, 0.        , 0.6       ,
        0.25      , 1.        , 0.5       , 0.70317003, 0.17647059,
        0.17857143],
       [1.        , 0.72727273, 0.88888889, 1.        , 0.5       ,
```

In [22]:
```
x_test=sc.transform(x_test)

x_test
```

Out[22]:
```
array([[1.        , 0.36363636, 0.55555556, 1.        , 0.        ,
        0.        , 0.        , 0.        , 0.86455331, 0.70588235,
        0.82142857],
       [0.44444444, 0.27272727, 0.88888889, 0.        , 0.3       ,
        0.375     , 1.        , 0.75      , 0.33717579, 0.88235294,
        0.89285714],
       [0.38888889, 0.63636364, 0.55555556, 1.        , 0.2       ,
        0.25      , 0.25      , 0.5       , 0.22190202, 0.64705882,
        0.42857143],
       [0.44444444, 0.45454545, 0.77777778, 1.        , 1.        ,
        0.375     , 1.        , 0.75      , 0.83861671, 0.44117647,
        0.14285714],
       [0.        , 0.31818182, 0.66666667, 1.        , 0.8       ,
        0.625     , 0.        , 0.75      , 0.13544669, 0.17647059,
        0.14285714],
       [0.72222222, 0.54545455, 0.        , 1.        , 0.5       ,
        0.125     , 0.25      , 0.25      , 0.3832853 , 0.44117647,
        0.5       ],
       [0.05555556, 0.81818182, 0.11111111, 1.        , 0.4       ,
        0.        , 0.25      , 0.        , 0.1556196 , 0.97058824,
        0.71428571],
       [1.        , 0.27272727, 0.11111111, 0.        , 0.1       ,
        0.125     , 0.        , 0.25      , 0.70317003, 0.29411765,
        0.5       ],
       [0.72222222, 0.54545455, 0.77777778, 0.        , 1.        ,
        0.5       , 0.25      , 1.        , 0.87896254, 0.26470588,
        0.10714286],
       [0.        , 0.45454545, 0.88888889, 0.        , 1.        ,
        0.5       , 0.        , 1.        , 0.20461095, 0.02941176,
        0.10714286],
       [0.72222222, 0.18181818, 0.77777778, 1.        , 0.        ,
        0.        , 0.25      , 0.        , 0.49279539, 0.85294118,
        0.85714286],
       [1.        , 0.27272727, 0.77777778, 0.        , 0.4       ,
        0.        , 0.        , 0.        , 0.55043228, 0.91176471,
```

In [23]:
```
y_test
```

Out[23]:
```
array([1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1])
```

decision tree

training

In [24]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [25]:
```
dt=DecisionTreeClassifier(criterion='entropy')
```

In [26]:
```
dt.fit(x_train,y_train)
```

Out[26]:
```
DecisionTreeClassifier(criterion='entropy')
```

predicting

In [27]:
```
y_pred_dt=dt.predict(x_test)
```

In [28]:
```
y_pred_dt
```

Out[28]:
```
array([1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1])
```

In [29]:
```
import sklearn.metrics as metrics
```

In [30]:
```
fpr,tpr,threshold=metrics.roc_curve(y_test,y_pred_dt)
```

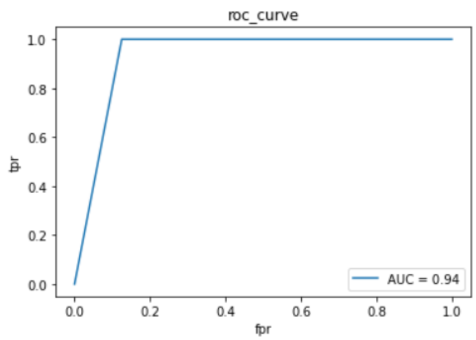
In [31]:
```
roc_auc_DT=metrics.auc(fpr,tpr)
```

```
In [32]: roc_auc_DT
Out[32]: 0.9375
```

```
In [33]: from sklearn.metrics import accuracy_score
```

```
In [34]: accuracy_score(y_test,y_pred_dt)
Out[34]: 0.9285714285714286
```

```
In [35]: plt.plot(fpr,tpr,label='AUC = %0.2f' % roc_auc_DT)
         plt.xlabel("fpr")
         plt.ylabel("tpr")
         plt.title("roc_curve")
         plt.legend()
Out[35]: <matplotlib.legend.Legend at 0x27b31949f70>
```




```
In [36]: #saving our model into a file
         import pickle
         pickle.dump(dt,open('PRJ.pkl','wb'))
```

```
In [37]: pickle.dump(sc,open('sc.pkl','wb'))
         pickle.dump(lb,open('lb.pkl','wb'))
```