3D Printer Material Prediction Using IBM Watson Studio

1. INTRODUCTION

1.1 Overview

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.

1.2 Purpose

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

There are various problems associated with a 3D Printer technology. Some of the biggest problems faced by 3D Printer technology are as follows :

- Equipment costs

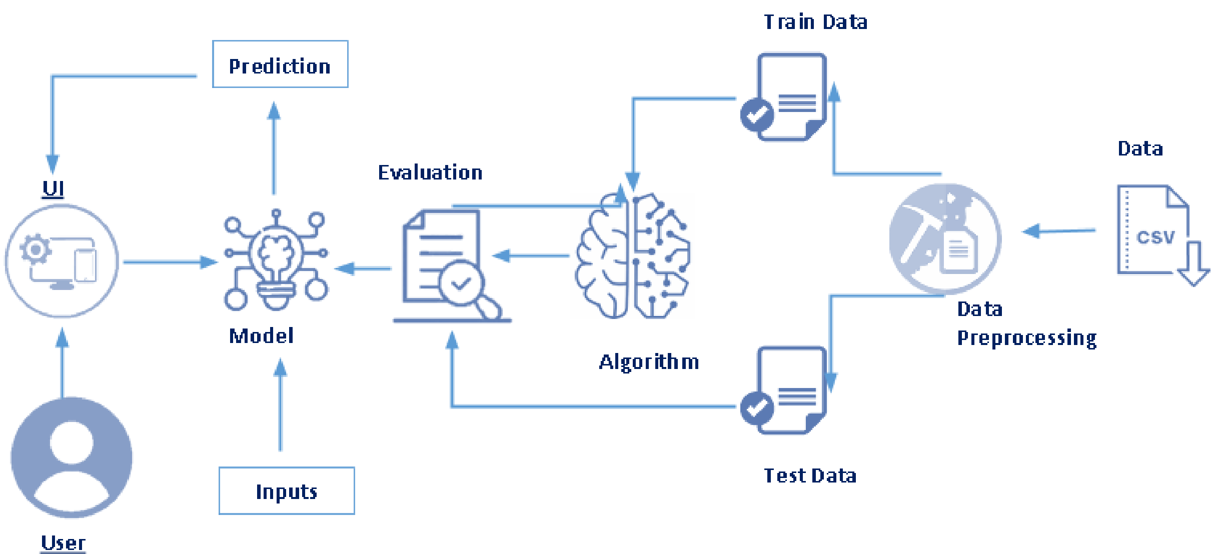
- Limited materials available
- Post-processing requirements
- Manufacturing costs
- Lack of in-house additive manufacturing resources
- Lack of expertise and/or training among workforce/employees
- Limited repeatability (accuracy from build to build)
- Lack of formal standards
- Lack of proven documentation of additive manufacturing’s capabilities
- Software development and capabilities
- Longer production timelines
- Limited recyclability
- Risk of litigation/legal implication
- Data storage requirements

2.2 Proposed solution

As can be seen, there are many problems that need to be solved for efficiently using 3D Printer. A lot of processing and documentation and manual analysis is involved to solve them. Instead if all the data required to print a 3D model is retrieved into a model , then the type of material needed can be predicted. The purpose here is to build a machine learning model and deploy it in Watson Studio by creating an endpoint. To interact with the model, Node-Red and scoring Endpoint will be used.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software designing

Software Requirements:

- Anaconda Navigator
- Tensor flow
- Keras
- Flask

Hardware Requirements:

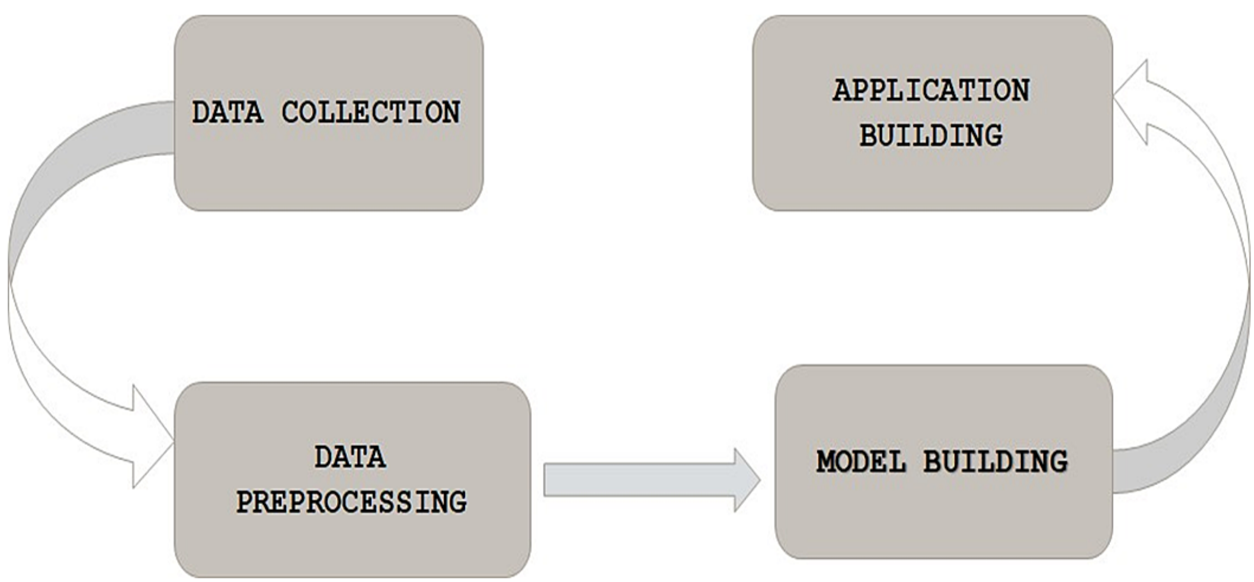
- Processor : Intel Core i3
- Hard Disk Space : Min 100 GB
- Ram : 4 GB
- Display : 14.1 “Color Monitor(LCD, CRT or LED)

  Clock Speed : 1.67 GHz

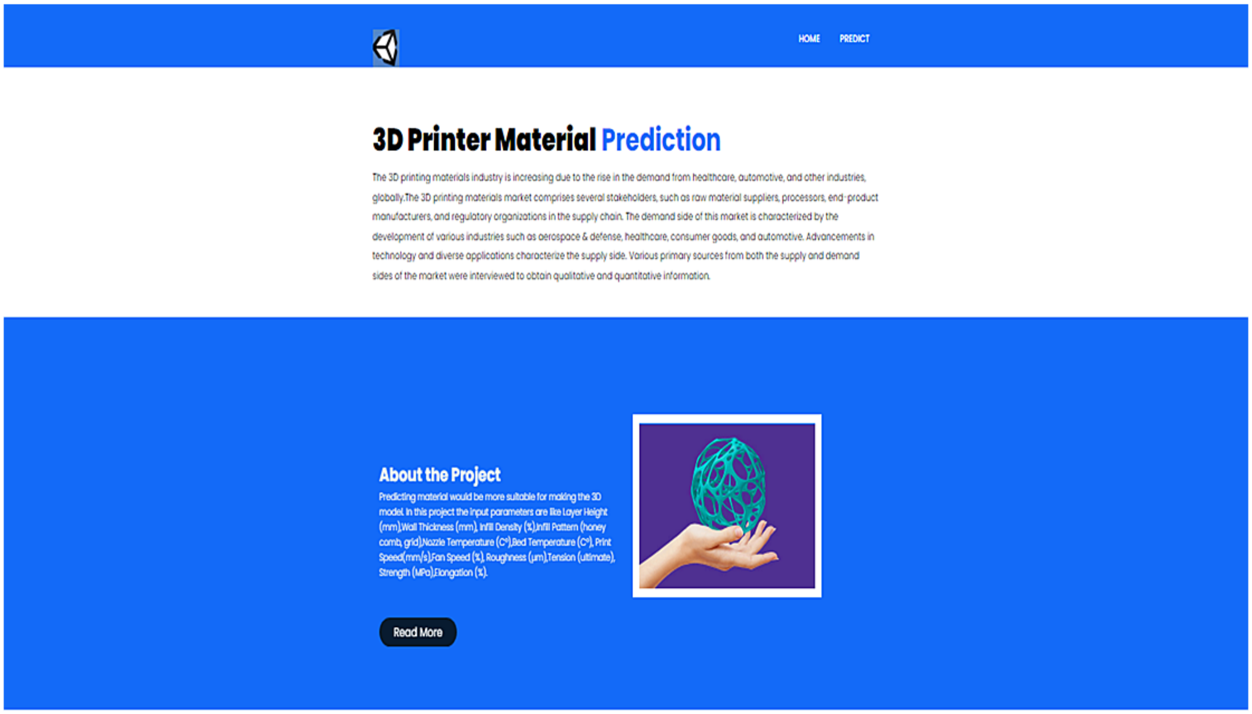
4. EXPERIMENTAL INVESTIGATIONS

Study shows that it provide with different data of a 3D model , the model detects, predicts the required material for the 3D model. When we enter the data for the 3D model and click the predict then it will show the predicted output.

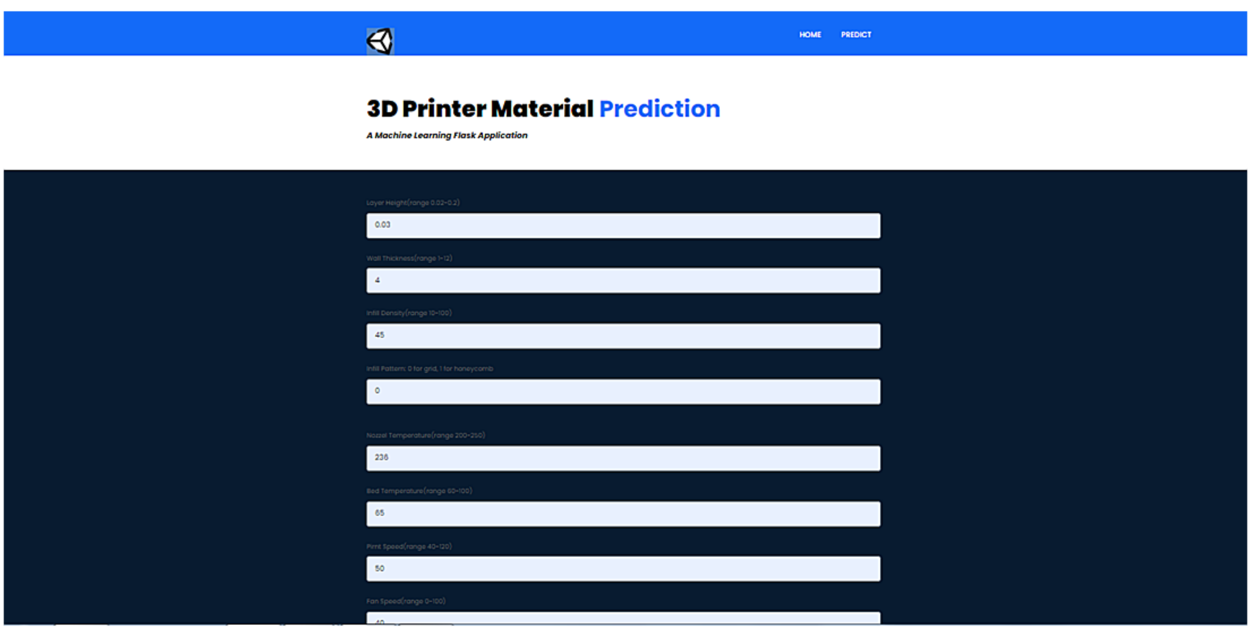
5. FLOWCHART



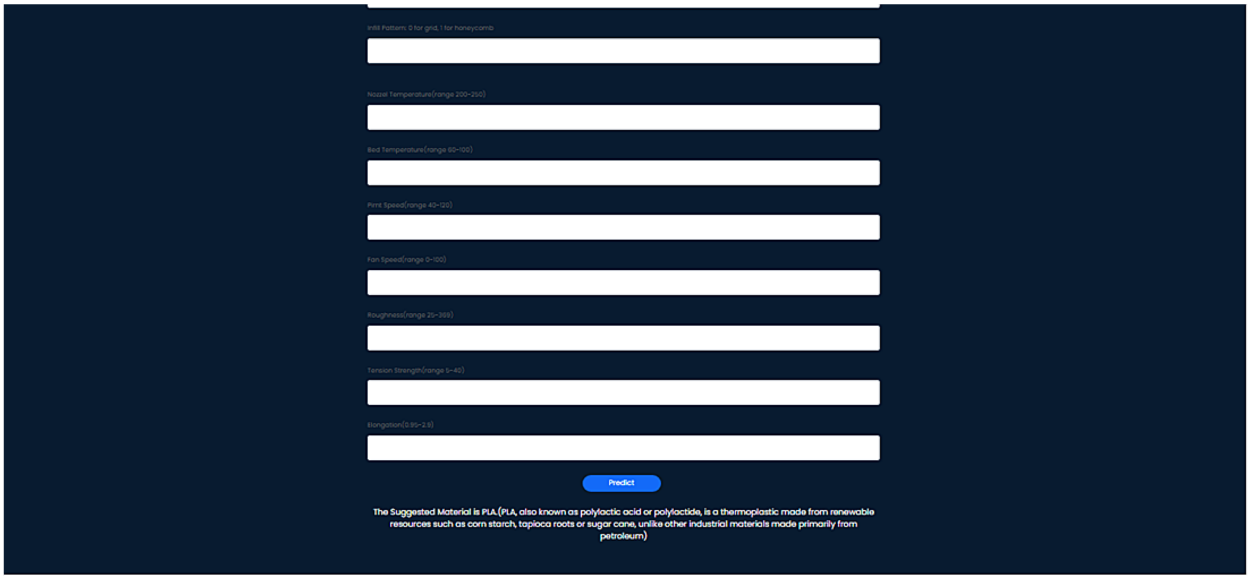
6. RESULT



HOME
PREDICT
3D Printer Material Prediction
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.
About the Project
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 (honey comb, grid),Nozzle Temperature (C°),Bed Temperature (C°), Print Speed(mm/s),Fan Speed (%), Roughness (µm),Tension (ultimate), Strength (MPa),Elongation (%).
Read More



HOME
PREDICT
3D Printer Material Prediction
A Machine Learning Flask Application
Layer Height(range 0.02-0.2)
0.03
Wall Thickness(range 1-12)
4
Infill Density(range 10-100)
45
Infill Pattern: 0 for grid, 1 for honeycomb
0
Nozzel Temperature(range 200-250)
236
Bed Temperature(range 60-100)
65
Print Speed(range 40-120)
50
Fan Speed(range 0-100)



7. ADVANTAGES & DISADVANTAGES

Advantages:

- Increased accuracy for Material prediction.
- Reduce the time complexity.

Disadvantages:

- Data mining techniques does not help to provide effective decision making.

8. APPLICATIONS

- Deep Learning technology is considered as one of the key technology used in 3D Printer Material Prediction. It presents the results obtained by processing input from the entered data.

9. CONCLUSION

In this project, we have established the application to predict from the data of a 3D Printer based on the IBM cloud application. 3D Printer Material Prediction can only use this web app to predict the material.

10. FUTURE SCOPE

The project can be further enhanced by deploying the deep learning model obtained using a web application and larger dataset cloud be used for prediction to give higher accuracy and produce better result.

11. BIBILOGRAPHY

- Ahmet Okudan, 3D Printer dataset for mechanical engineers, https://www.kaggle.com/datasets/afumetto/3dprinter

APPENDIX

Source Code

In [4]:
```python
##Importing libraries
import pandas as  pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

In [5]:
```python
##Loading the dataset

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

In [6]:
```python
##Printing the first five rows
ds.head()
```

Out[6]:

	layer_height	wall_thickness	infill_density	infill_pattern	nozzle_temperature	bed_temperature	print_speed	material	fan_speed	roughness	tension_strenght
0	0.02	8.0	90	grid	220	60	40	abs	0	25	18
1	0.02	7.0	90	honeycomb	225	65	40	abs	25	32	16
2	0.02	1.0	80	grid	230	70	40	abs	50	40	8
3	0.02	4.0	70	honeycomb	240	75	40	abs	75	68	10
4	0.02	6.0	90	grid	250	80	40	abs	100	92	5

In [7]:
```python
ds.tail()
```

Out[7]:

	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 [8]:
```python
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66 entries, 0 to 65
Data columns (total 12 columns):
layer_height          66 non-null float64
wall_thickness        66 non-null float64
infill_density        66 non-null int64
infill_pattern        66 non-null object
nozzle_temperature    66 non-null int64
bed_temperature       66 non-null int64
print_speed           66 non-null int64
material              66 non-null object
fan_speed             66 non-null int64
roughness             66 non-null int64
tension_strenght      66 non-null int64
elongation            66 non-null float64
dtypes: float64(3), int64(7), object(2)
memory usage: 5.7+ KB
```

In [9]:
```
##descripive statistics
ds.describe()
```

Out[9]:

	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 [10]:
```
##corr among the data
ds.corr()
```

Out[10]:

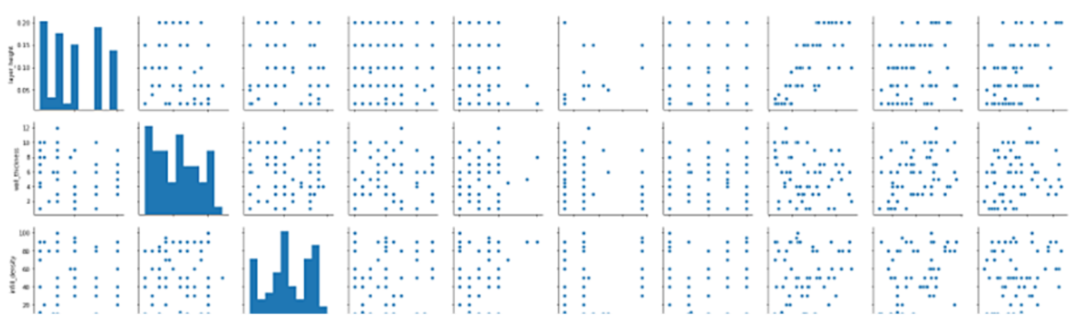
	layer_height	wall_thickness	infill_density	nozzle_temperature	bed_temperature	print_speed	fan_speed	roughness	tension_strenght	elon
layer_height	1.000000	-0.282933	-0.013763	-0.030562	-0.120838	0.044329	-0.040571	0.773096	0.325276	0.4
wall_thickness	-0.282933	1.000000	0.025534	-0.130299	0.061974	-0.341273	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

In [11]:
```
##Finding null values
ds.isnull().any()
```

Out[11]:
```
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
```

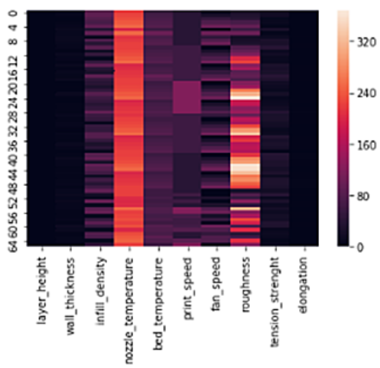
In [12]:
```
##Seaborn Pairplot
##plots a pairwise relationships in a dataset
sns.pairplot(ds)
```

Out[12]: <seaborn.axisgrid.PairGrid at 0x5808050>



In [13]:
```
##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[13]: `<matplotlib.axes._subplots.AxesSubplot at 0xd78e890>`



In [14]:
```
##Label Emcoding
from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

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

In [15]:
```
ds[:,3]=lb.fit_transform(ds[:,3])
```

In [16]:
```
da=pd.DataFrame(ds)
```

In [17]:
```
y=ds[:,7]

y=y.astype("int")
```

In [18]:
```
da.drop(columns=7,inplace=True)
```

In [19]:
```
x=da.iloc[:,:].values

x
```

Out[19]:
```
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],
```

In [20]:

```
# 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 [21]:

```
# FEATURE SCALING

from sklearn.preprocessing import MinMaxScaler

sc=MinMaxScaler()
```

In [22]:

```
x_train
```

Out[22]:

```
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 [23]:

```
x_train=sc.fit_transform(x_train)
```

```
C:\Users\ashz\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype object
was converted to float64 by MinMaxScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [24]:

```
x_train
```

Out[24]:

```
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 [25]:

```
x_test=sc.transform(x_test)

x_test
```

Out[25]:

```
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,
```

In [26]:
```
y_test
```
Out[26]:
```
array([1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1])
```

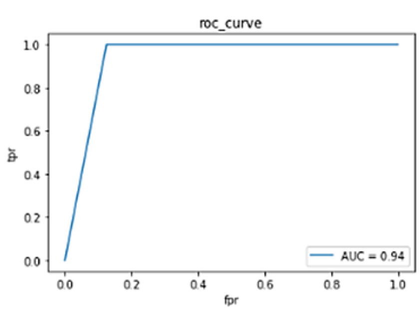
decision tree

training

In [27]:
```
from sklearn.tree import DecisionTreeClassifier
```
In [28]:
```
dt=DecisionTreeClassifier(criterion='entropy')
```
In [29]:
```
dt.fit(x_train,y_train)
```
Out[29]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

predicting

In [30]:
```
y_pred_dt=dt.predict(x_test)
```
In [31]:
```
y_pred_dt
```
Out[31]:
```
array([1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1])
```
In [32]:
```
import sklearn.metrics as metrics
```
In [33]:
```
fpr,tpr,threshold=metrics.roc_curve(y_test,y_pred_dt)
```
In [34]:
```
roc_auc_DT=metrics.auc(fpr,tpr)
```
In [35]:
```
roc_auc_DT
```
Out[35]:
```
0.9375
```
In [36]:
```
from sklearn.metrics import accuracy_score
```
In [37]:
```
accuracy_score(y_test,y_pred_dt)
```
Out[37]:
```
0.9285714285714286
```
In [38]:
```
plt.plot(fpr,tpr,label='AUC = %0.2f' % roc_auc_DT)
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.title("roc_curve")
plt.legend()
```
Out[38]:
```
<matplotlib.legend.Legend at 0x1239dd50>
```



In [39]:

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

In [40]:

```
pickle.dump(sc,open('sc.pkl','wb'))
pickle.dump(lb,open('lb.pkl','wb'))
```