3D Printer Material Prediction UsingMachine Learning

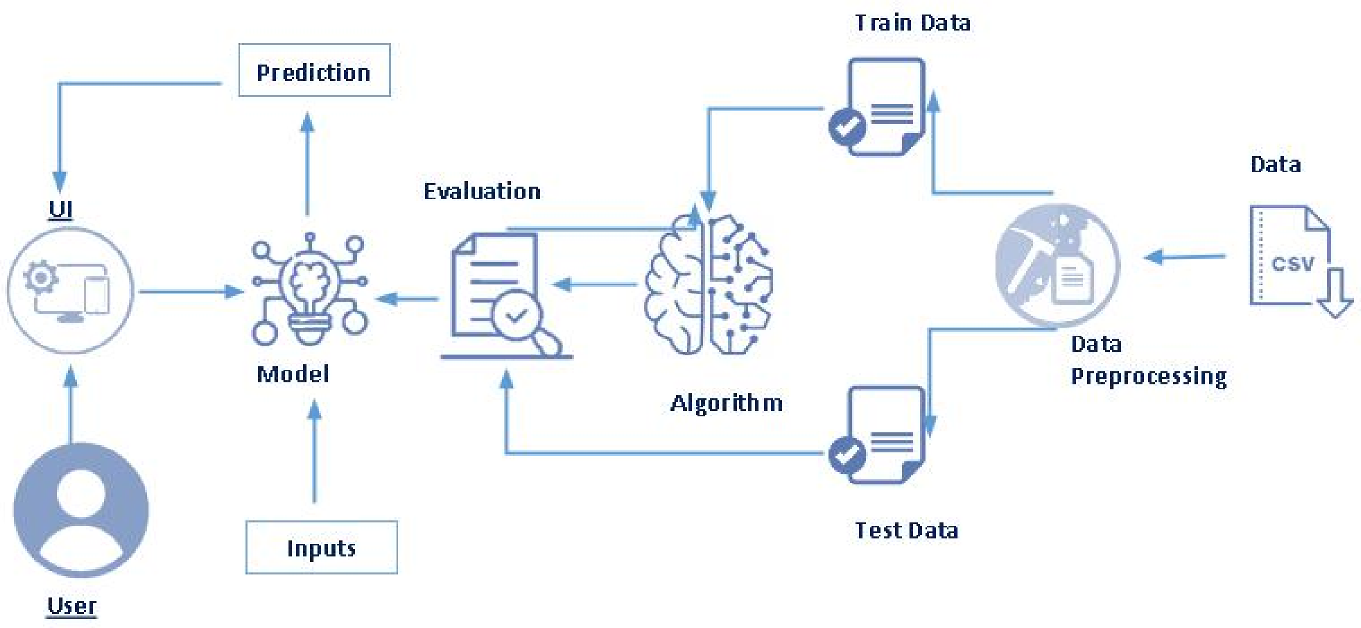
3D Printer Material Prediction Using Machine Learning

This project creates a smart tool that uses machine learning to predict the best 3D printer material for any job.

Instead of testing different materials (like PLA, ABS, Nylon) and wasting time and plastic, you just enter what you need—like how strong the part must be, or how much heat it should handle.

The system instantly analyzes this data and tells you the perfect material to use. This makes 3D printing faster, cheaper, and more reliable, ensuring your part is right the first time.

Technical Architecture:



Project Flow

- User interacts with the UI (User Interface) to enter Data
- The entered data is analyzed by the model which is integrated
- Once model analyses the input the prediction is showcased on the UI

You will go through all the steps mentioned below to complete the project.
To accomplish this, we have to complete all the activities and tasks listed below

Data Collection.
- Collect the dataset or Create the dataset

Data Preprocessing.
- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.

Data Visualization.
- Taking care of Missing Data.
- Label encoding.
- One Hot Encoding.
- Feature Scaling.
- Splitting Data into Train and Test.

Model Building
- Training and testing the model
- Evaluation of Model

Application Building
- Create an HTML file
- Build a Python Code

Prior Knowledge:

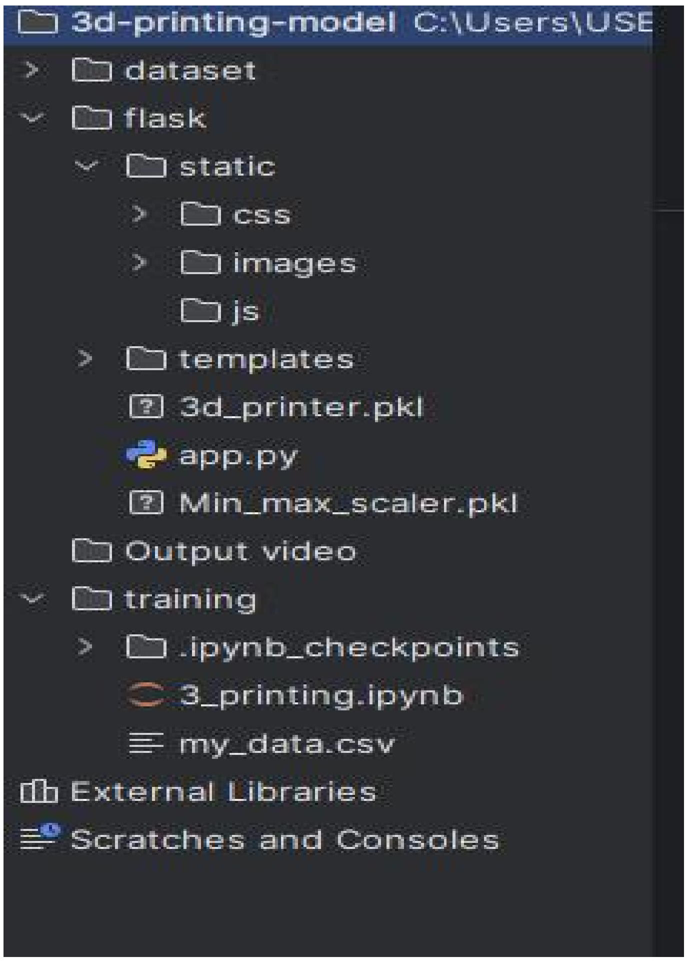
You must have prior knowledge of following topics to complete this project.

ML Concepts

- **Supervised learning:** https://www.javatpoint.com/supervised- machine-learning
- **Unsupervised learning:** https://www.javatpoint.com/unsupervised- machine-learning
- **Decision tree:** https://www.javatpoint.com/machine- learning-decision-tree-classification- algorithm
- **Random forest:** https://www.javatpoint.com/machine-learning- random-forest-algorithm
- **KNN:** https://www.javatpoint.com/k-nearest-neighbor-algorithm- for-machine-learning
- **Xgboost:**https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to- understand-the-math-behind- xgboost/
- **Evaluation metrics:** https://www.analyticsvidhya.com/blog/2019/08/11- important-model- evaluation-error-metrics/
- **Flask Basics :** https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below.



```
3d-printing-model C:\Users\USE
  > dataset
  v flask
    v static
      > css
      > images
        js
    > templates
      3d_printer.pkl
      app.py
      Min_max_scaler.pkl
    Output video
  v training
    > .ipynb_checkpoints
      3_printing.ipynb
      my_data.csv
External Libraries
Scratches and Consoles
```

- We have four folders dataset, Flask, Output video and Training.
- Dataset has dataset dataset.csv.
- A python file called app.py for server-side scipting.
- We need the model which is saved and the saved model in this content is 3d_printer.pkl.
- Templates folder which contains home.html, about.html and result.html files.
- The training folder has 3_printing.ipynb where the model is created and saved.
- Static folder which contains css(styling), fontawesome(styling), img(images), js(Java script) folders to enhance the features of web page.

Pre-requesties:

In order to develop this project we need to install the following software/packages:

Step 1:

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, cross-platform, package management system. Anaconda comes with great tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code.

- For this project, we will be using Jupyter notebook and Spyder.

- To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video.

Step 2:

- To build Machine learning models you must require the following packages
- **Sklearn:** Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.
- **NumPy :** NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n- dimensional array object.
- **Pandas :** pandas is a fast, powerful, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.
- **Seaborn :** It is a powerful Python data visualization library built on Matplotlib that provides a high-level interface for creating attractive and informative statistical graphics.
- **Matplotlib :** Matplotlib is a versatile Python library for creating static, animated, and interactive visualizations.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

- The core problem is the time- consuming and inefficient selection of 3D printing.
- materials for complex parts based on specific requirements (e.g., strength, heat resistance).
- The ML system provides an automated, data-driven recommendation to minimize material waste, reduce the number of trial prints, decrease lead time, and ensure the printed component meets all performance specifications.
- The current manual process of selecting a 3D printing material is a major source of inefficiency and cost.
- Engineers and designers often rely on historical experience or time-consuming physical testing to determine if a material is suitable for their part's functional requirements.
- This trial-and-error approach frequently results in expensive material waste and lengthy development cycles as parts must be reprinted multiple times to achieve the desired mechanical properties (like strength or temperature resistance).

Activity 2: Business requirements

- The system must meet several critical requirements to be adopted and valuable to users (designers and engineers):
- **Accuracy and Reliability:** The ML model must achieve a prediction accuracy of 95% or higher in recommending the correct material category (e.g., PLA, ABS, PETG). The system must provide a confidence score with every prediction, allowing the user to trust the recommendation for critical parts.

- **Input Flexibility:** The user interface (UI) must allow for input of various types of criteria, including:
- **Environmental Requirements:** Desired Heat Deflection Temperature (C) and UV/Water Resistance rating.
- **Printer & Cost Constraints:** Input fields for available printer type and maximum acceptable material cost per kilogram.
-
- **Speed and Responsiveness:** The prediction from the time a user submits their requirements to receiving the recommended material must take less than 5 seconds. Waiting longer would defeat the goal of accelerating the design process.
- **Scalability and Maintenance:** The backend must be designed to easily add new material data (new polymers) and adjust for updated printer parameters without requiring a complete model retraining or system overhaul.
- **User Experience (UX):** The final web application must be intuitive and simple. The results page must clearly display the recommended material, along with a list of supporting evidence (the key properties that led to the decision) and a clear call-to-action for sourcing the material.

.

Activity 3: Literature Survey

The survey focuses on three key areas:

- **3D Printing Material Science:** Reviewing the properties (tensile strength, flexibility, thermal resistance) of common 3D printing materials (e.g., PLA, ABS, PETG, Nylon).
- **Machine Learning in Manufacturing:** Examining existing research on using ML for material selection, quality control, and process optimization in additive manufacturing.
- **Classification Systems:** Identifying standard and advanced classification models best suited for predicting categorical material types based on numerical and categorical input features.

Activity 4: Social and Business Impact.

Business Impact:

- **Cost Reduction & Efficiency:** By eliminating the guesswork in material selection, we drastically reduce the number of failed prints and the associated waste of high-value polymers. This directly cuts material costs and shortens the product development cycle by days or even weeks.
- **Accelerated Time-to-Market:** Engineers can finalize designs faster because they receive an instant, accurate material recommendation instead of waiting for lengthy lab tests. This acceleration allows us to launch new products faster than our competitors.
- **Data-Driven Decision Making:** We are moving away from relying on "tribal knowledge" to a scalable, data-driven approach. This ensures consistent quality across all projects, reduces dependency on senior staff, and makes our manufacturing process more robust.

Social Impact :

- **Environmental Sustainability:** Reducing material waste is a significant environmental win. By ensuring that almost every print is successful on the first try, we minimize plastic consumption and the energy associated with failed prints and disposal.
- **Empowering Innovation:** By making material selection easy and accessible, we enable smaller firms, educational institutions, and independent innovators to experiment with advanced manufacturing techniques.
- **Safety and Reliability:** In critical applications (like medical or aerospace parts), the model ensures the part meets strict safety specifications by accurately predicting mechanical performance.

Milestone 2 : Data Collection & Preparation

- The success of this entire project hinges on the quality and comprehensiveness of the data we feed the machine learning model. In the real world, the relationship between a part's required properties (like high tensile strength) and the print settings (like slower speed or higher temperature) is complex and non-linear. Therefore, we cannot rely on small, anecdotal datasets.

- Our goal in this initial phase is to curate a master dataset that is both broad and deep, covering the spectrum of common polymers (PLA, ABS, PETG, Nylon, Resin) and their corresponding performance metrics. This dataset must not only list material specifications from manufacturers but also incorporate empirical data that accounts for real-world printing variables—a factor often missed in simpler models. Inadequate or biased data at this stage will lead to a predictive model that works well in theory but fails when presented with a new, unique design requirement in practice.

Activity 1: Collect the dataset

- There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

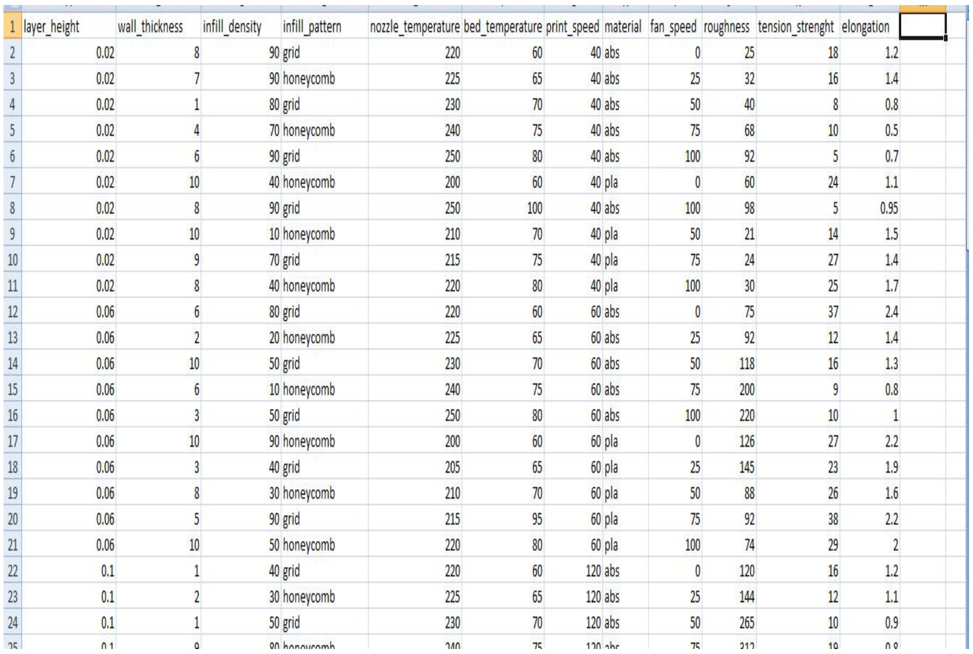
- In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

  Link : https://www.kaggle.com/vinaynomula/3d-printer-material-dataset

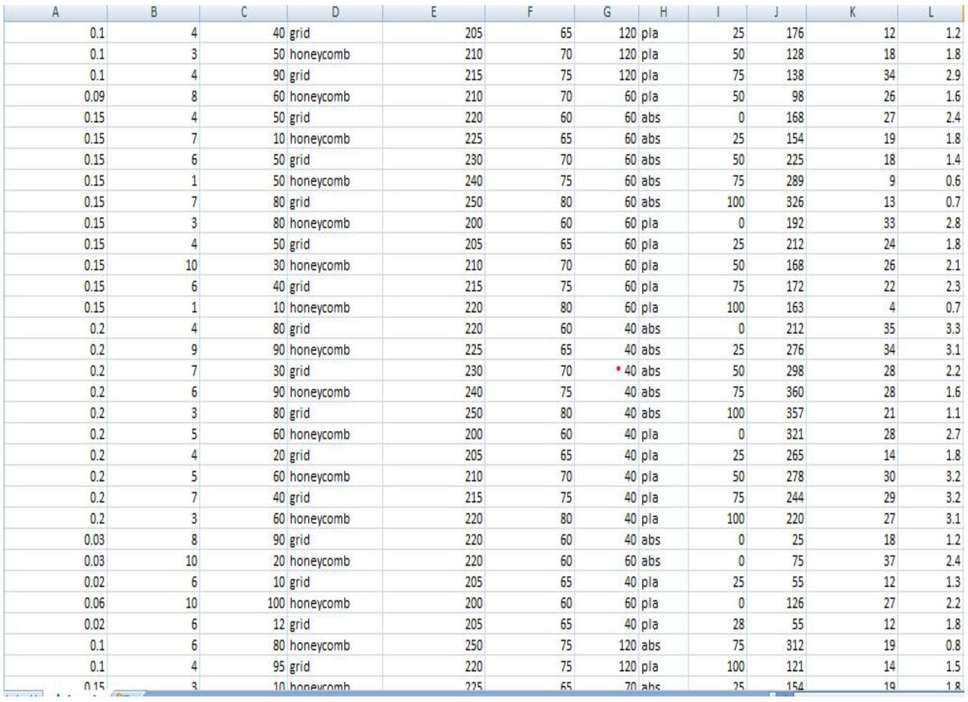
- As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

- Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Dataset Screenshots:



1	layer_height	wall_thickness	infill_density	infill_pattern	nozzle_temperature	bed_temperature	print_speed	material	fan_speed	roughness	tension_strenght	elongation
2	0.02	8	90	grid	220	60	40	abs	0	25	18	1.2
3	0.02	7	90	honeycomb	225	65	40	abs	25	32	16	1.4
4	0.02	1	80	grid	230	70	40	abs	50	40	8	0.8
5	0.02	4	70	honeycomb	240	75	40	abs	75	68	10	0.5
6	0.02	6	90	grid	250	80	40	abs	100	92	5	0.7
7	0.02	10	40	honeycomb	200	60	40	pla	0	60	24	1.1
8	0.02	8	90	grid	250	100	40	abs	100	98	5	0.95
9	0.02	10	10	honeycomb	210	70	40	pla	50	21	14	1.5
10	0.02	9	70	grid	215	75	40	pla	75	24	27	1.4
11	0.02	8	40	honeycomb	220	80	40	pla	100	30	25	1.7
12	0.06	6	80	grid	220	60	60	abs	0	75	37	2.4
13	0.06	2	20	honeycomb	225	65	60	abs	25	92	12	1.4
14	0.06	10	50	grid	230	70	60	abs	50	118	16	1.3
15	0.06	6	10	honeycomb	240	75	60	abs	75	200	9	0.8
16	0.06	3	50	grid	250	80	60	abs	100	220	10	1
17	0.06	10	90	honeycomb	200	60	60	pla	0	126	27	2.2
18	0.06	3	40	grid	205	65	60	pla	25	145	23	1.9
19	0.06	8	30	honeycomb	210	70	60	pla	50	88	26	1.6
20	0.06	5	90	grid	215	95	60	pla	75	92	38	2.2
21	0.06	10	50	honeycomb	220	80	60	pla	100	74	29	2
22	0.1	1	40	grid	220	60	120	abs	0	120	16	1.2
23	0.1	2	30	honeycomb	225	65	120	abs	25	144	12	1.1
24	0.1	1	50	grid	230	70	120	abs	50	265	10	0.9
25	0.1	9	80	honeycomb	240	75	120	abs	75	312	19	0.8



A	B	C	D	E	F	G	H	I	J	K	L
0.1	4	40	grid	205	65	120	pla	25	176	12	1.2
0.1	3	50	honeycomb	210	70	120	pla	50	128	18	1.8
0.1	4	90	grid	215	75	120	pla	75	138	34	2.9
0.09	8	60	honeycomb	210	70	60	pla	50	98	26	1.6
0.15	4	50	grid	220	60	60	abs	0	168	27	2.4
0.15	7	10	honeycomb	225	65	60	abs	25	154	19	1.8
0.15	6	50	grid	230	70	60	abs	50	225	18	1.4
0.15	1	50	honeycomb	240	75	60	abs	75	289	9	0.6
0.15	7	80	grid	250	80	60	abs	100	326	13	0.7
0.15	3	80	honeycomb	200	60	60	pla	0	192	33	2.8
0.15	4	50	grid	205	65	60	pla	25	212	24	1.8
0.15	10	30	honeycomb	210	70	60	pla	50	168	26	2.1
0.15	6	40	grid	215	75	60	pla	75	172	22	2.3
0.15	1	10	honeycomb	220	80	60	pla	100	163	4	0.7
0.2	4	80	grid	220	60	40	abs	0	212	35	3.3
0.2	9	90	honeycomb	225	65	40	abs	25	276	34	3.1
0.2	7	30	grid	230	70	40	abs	50	298	28	2.2
0.2	6	90	honeycomb	240	75	40	abs	75	360	28	1.6
0.2	3	80	grid	250	80	40	abs	100	357	21	1.1
0.2	5	60	honeycomb	200	60	40	pla	0	321	28	2.7
0.2	4	20	grid	205	65	40	pla	25	265	14	1.8
0.2	5	60	honeycomb	210	70	40	pla	50	278	30	3.2
0.2	7	40	grid	215	75	40	pla	75	244	29	3.2
0.2	3	60	honeycomb	220	80	40	pla	100	220	27	3.1
0.03	8	90	grid	220	60	40	abs	0	25	18	1.2
0.03	10	20	honeycomb	220	60	60	abs	0	75	37	2.4
0.02	6	10	grid	205	65	40	pla	25	55	12	1.3
0.06	10	100	honeycomb	200	60	60	pla	0	126	27	2.2
0.02	6	12	grid	205	65	40	pla	28	55	12	1.8
0.1	6	80	honeycomb	250	75	120	abs	75	312	19	0.8
0.1	4	95	grid	220	75	120	pla	100	121	14	1.5
0.15	3	10	honeycomb	225	65	70	abs	25	154	19	1.8

Activity 1.1: Importing the libraries

- Import the necessary libraries as shown in the image. Here we have used visualisation style as fivethirtyeight.

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix,
roc_auc_score,recall_score,precision_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score, learning_curve
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings("ignore")
```

- **Purpose:** Importing libraries allows you to access pre-written code for tasks like data analysis (pandas), visualization (matplotlib, seaborn), or machine learning (scikit-learn), saving time and effort.

- **Efficiency:** Libraries modularize functionality, enabling you to write concise, powerful code without reinventing the wheel.

- **Modularity:** Libraries promote modular programming by letting you import only what you need, reducing memory usage and improving code clarity.

- **Namespace Management:** Using aliases like import numpy as np helps avoid naming conflicts and makes code more concise and readable.

- **Custom Imports:** You can import your own Python files (modules) using import filename, enabling reuse of functions and classes across projects.

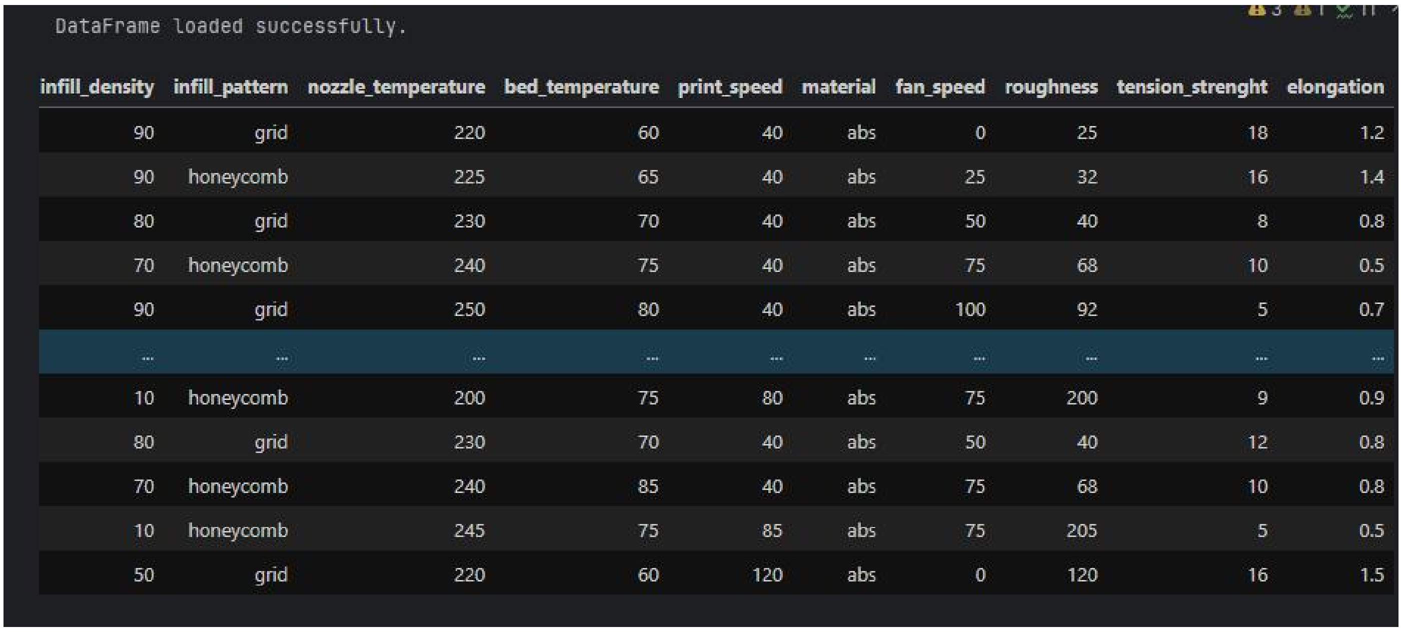
Activity 1.2: Read the Dataset

- Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

- In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
datapath = r"C:\Users\USER\3d-printing-model\dataset\dataset.csv"
DataFrame= pd.read_csv(datapath)
```

```
print("DataFrame loaded successfully.")
DataFrame=pd.DataFrame(DataFrame)
DataFrame
```



```
DataFrame loaded successfully.
```

infill_density	infill_pattern	nozzle_temperature	bed_temperature	print_speed	material	fan_speed	roughness	tension_strenght	elongation
90	grid	220	60	40	abs	0	25	18	1.2
90	honeycomb	225	65	40	abs	25	32	16	1.4
80	grid	230	70	40	abs	50	40	8	0.8
70	honeycomb	240	75	40	abs	75	68	10	0.5
90	grid	250	80	40	abs	100	92	5	0.7
...	...	...	...	...	...	...	...	...	...
10	honeycomb	200	75	80	abs	75	200	9	0.9
80	grid	230	70	40	abs	50	40	12	0.8
70	honeycomb	240	85	40	abs	75	68	10	0.8
10	honeycomb	245	75	85	abs	75	205	5	0.5
50	grid	220	60	120	abs	0	120	16	1.5

Activity 2: Data Preparation

- As we have understood how the data is, let's pre-process the collected data.

- The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling Outliers
- Checking duplicate values

- Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

- For checking the null values, DataFrame.isnull() function is used.

- To sum those null values we use .sum() function.

- From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
null_values = DataFrame.isnull().sum()
print("="*50)
print("Null values in each column".center(50))
print("="*50)
print(null_values)
```

Activity 2.2: Checking duplicate values

- **Detection:** Use DataFrame.duplicated() to identify duplicate rows; it returns a Boolean Series indicating whether each row is a duplicate of a previous one.

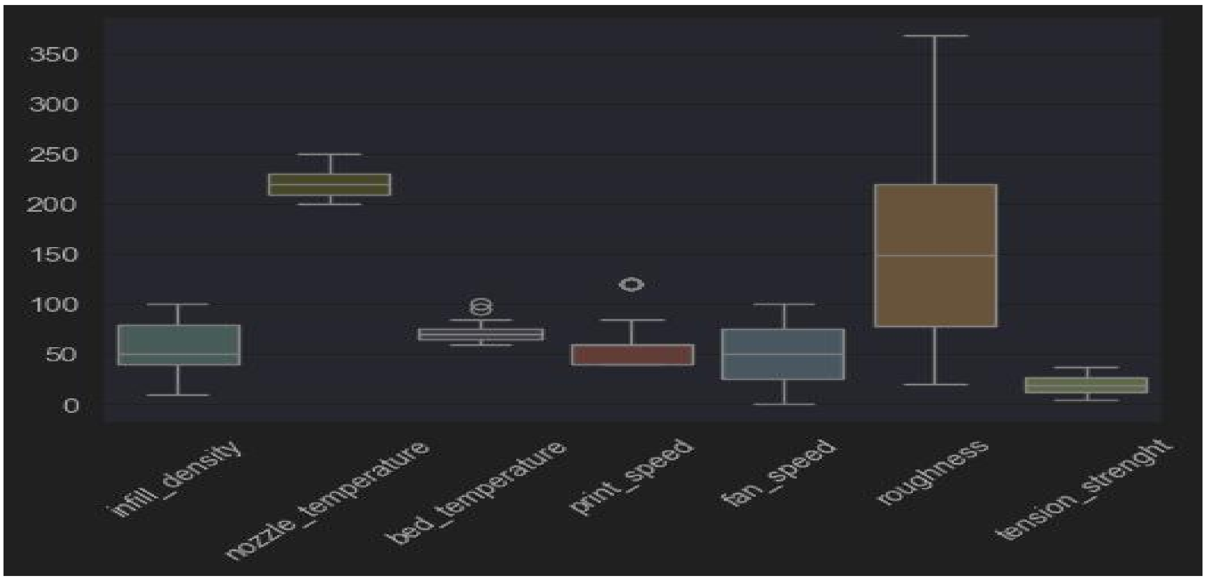
- **Removal:** Apply df.drop_duplicates() to remove duplicate rows and keep only the first occurrence by default, helping clean and streamline your dataset.

```
## checking duplicate values
print("="*50)
print("DUPLICATE VALUES CHECK".center(50))
print("="*50)

duplicate_count = DataFrame.duplicated().sum()
print(f"Number of duplicate rows in the DataFrame: {duplicate_count}")
```

Activity 2.3: Handling Outliers

- With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound all numerical feature with some mathematical formula.

- From the below diagram, we could visualize that bed temperature and print speed has outliers. Boxplot from seaborn library is used here to plot them.

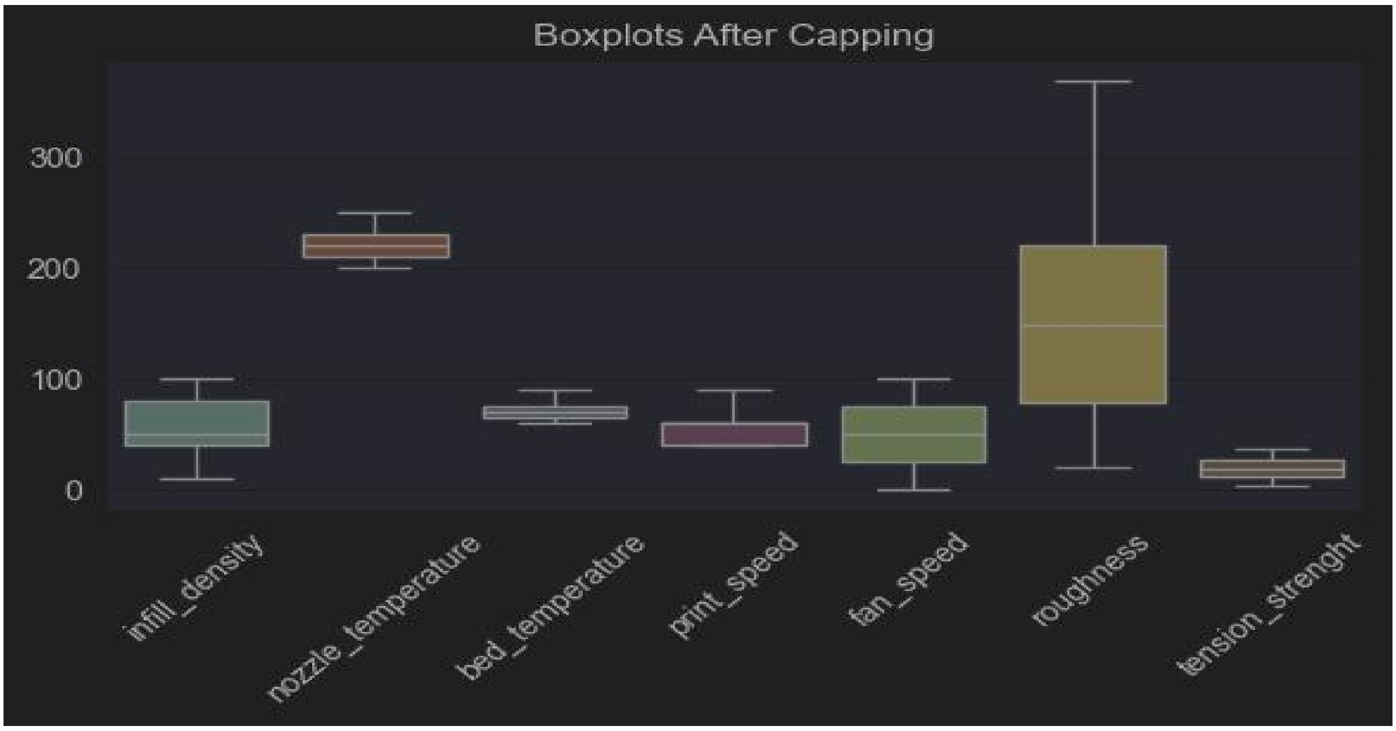


- To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.

```
Q1 = DataFrame[numerical_cols].quantile(0.25)
Q3 = DataFrame[numerical_cols].quantile(0.75)
IQR = Q3 - Q1
outliers = ((DataFrame[numerical_cols] < (Q1 - 1.5 * IQR)) | (DataFrame[numerical_cols] > (Q3 + 1.5 * IQR))).sum()
print("Number of outliers in each numerical column:")
print(outliers)
plt.figure(figsize=(6,4))
cols_to_plot = [col for col in numerical_cols if col not in ['elongation','layer_height','wall_thickness','infill_pattern','material']]
sns.boxplot(data=DataFrame[cols_to_plot], palette='Set3')
plt.xticks(rotation=45)
plt.show()
```

To handle the outliers, a capping technique is used. Here, the IQR method is applied to clip extreme values beyond 1.5 times the interquartile range. We have created a function to cap outliers and visualize the distribution and probability plot of the numerical columns.

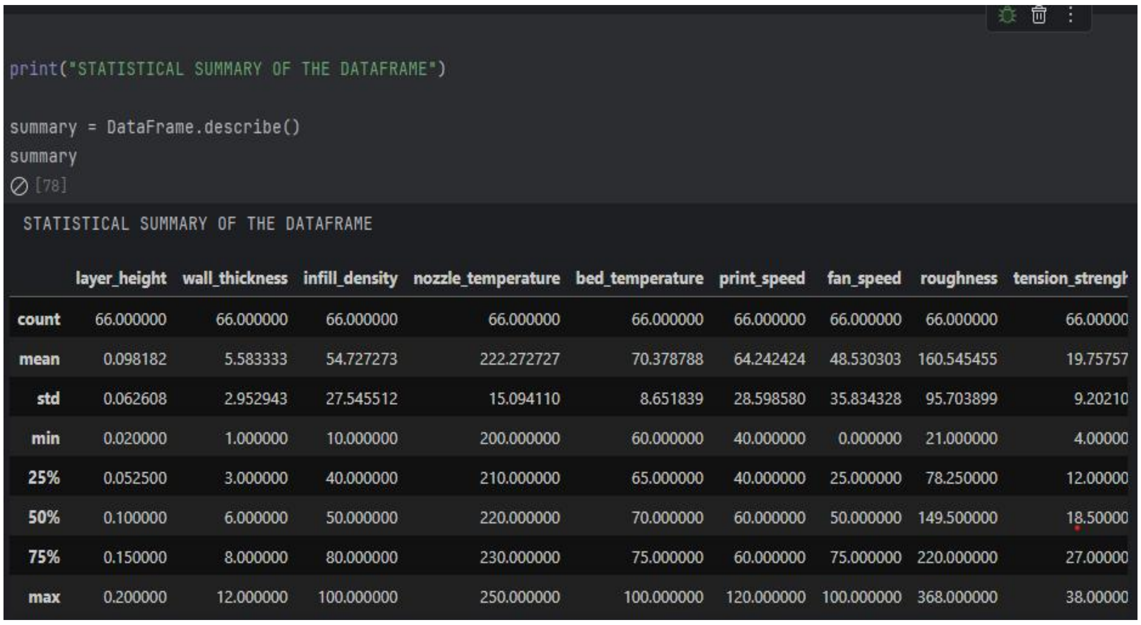
```
for col in numerical_cols:
    Q1 = DataFrame[col].quantile(0.25)
    Q3 = DataFrame[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    DataFrame[col] = np.where(DataFrame[col] < lower, lower, DataFrame[col])
    DataFrame[col] = np.where(DataFrame[col] > upper, upper, DataFrame[col])
```



Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.



```
print("STATISTICAL SUMMARY OF THE DATAFRAME")

summary = DataFrame.describe()
summary
[78]
```

```
STATISTICAL SUMMARY OF THE DATAFRAME
```

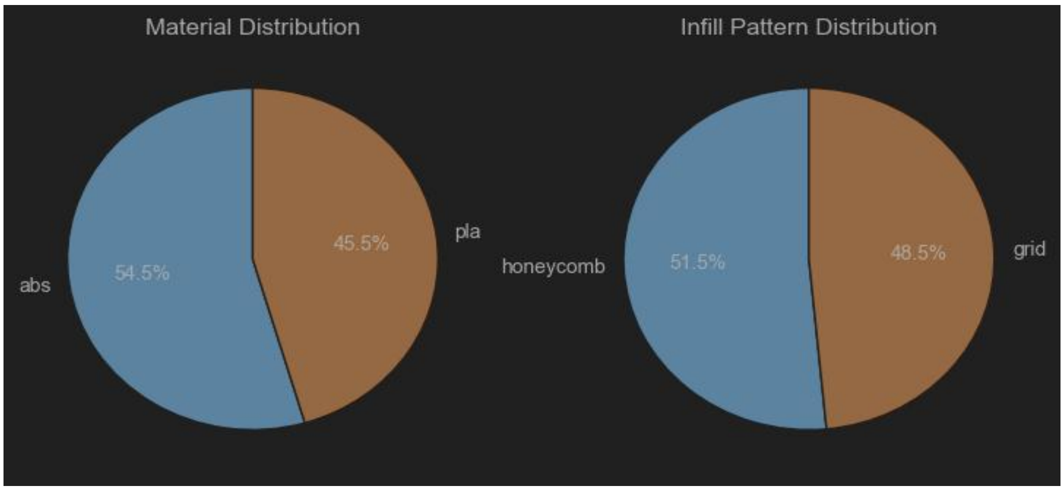
	layer_height	wall_thickness	infill_density	nozzle_temperature	bed_temperature	print_speed	fan_speed	roughness	tension_strengt
count	66.000000	66.000000	66.000000	66.000000	66.000000	66.000000	66.000000	66.000000	66.00000
mean	0.098182	5.583333	54.727273	222.272727	70.378788	64.242424	48.530303	160.545455	19.75757
std	0.062608	2.952943	27.545512	15.094110	8.651839	28.598580	35.834328	95.703899	9.20210
min	0.020000	1.000000	10.000000	200.000000	60.000000	40.000000	0.000000	21.000000	4.00000
25%	0.052500	3.000000	40.000000	210.000000	65.000000	40.000000	25.000000	78.250000	12.00000
50%	0.100000	6.000000	50.000000	220.000000	70.000000	60.000000	50.000000	149.500000	18.50000
75%	0.150000	8.000000	80.000000	230.000000	75.000000	60.000000	75.000000	220.000000	27.00000
max	0.200000	12.000000	100.000000	250.000000	100.000000	120.000000	100.000000	368.000000	38.00000

Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

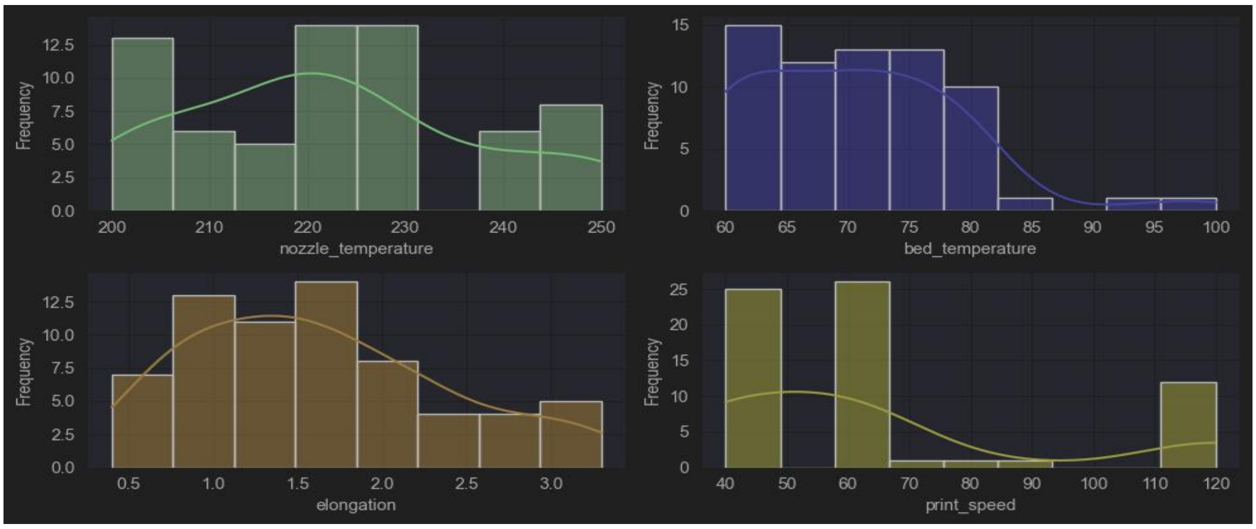
Activity 2.1: Univariate analysis

- In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as Piechart and countplot.
- In this section we created a two pie charts both describe about the distribution values in features like material and infill pattern.



- The left pie chart shows that the dataset contains two types of printing materials ABS (54.5%) and PLA (45.5%). This indicates a fairly balanced usage, with a slight preference for ABS.

- The right pie chart illustrates the proportions of infill patterns Honeycomb (51.5%) and Grid (48.5%). Again, the distribution is nearly even, indicating no strong bias toward one pattern.

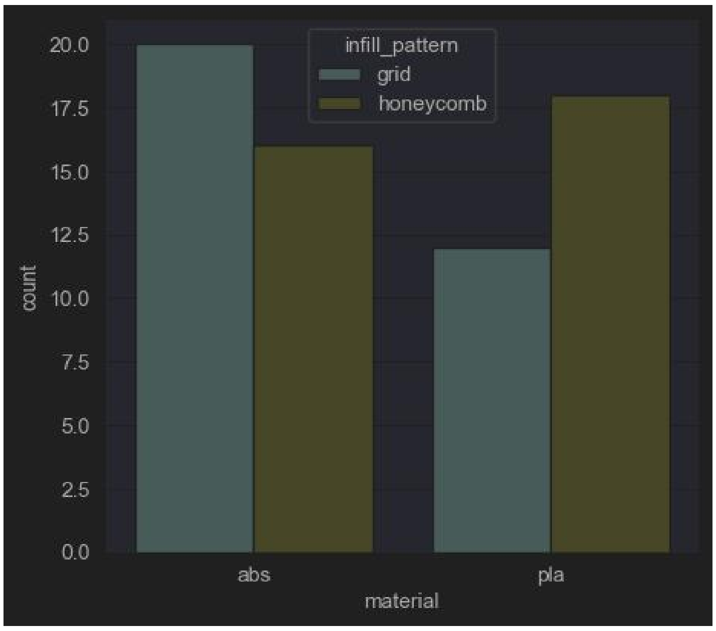


- **'Nozzle Temperature:** The distribution is unimodal and fairly symmetric, peaking around 220 to 230°C. This suggests a consistent operating range across most prints, likely optimized for material performance.
- **Bed Temperature:** The distribution is right-skewed, with most values concentrated between 60 to 70°C.This indicates a preference for lower bed temperatures, possibly to reduce warping or material degradation.
- **Elongation**: The elongation values show a unimodal distribution centered around 1.5to 2.0, implying that most materials exhibit moderate stretchability, which could be key for durability or flexibility.
- **Print Speed:** The distribution is bimodal, with peaks around 50to 60 mm/s and 110 to 120 mm/s.This suggests two distinct printing strategies perhaps one for precision and another for speed depending on material or design requirements.'

Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we can used barplot and boxplots.

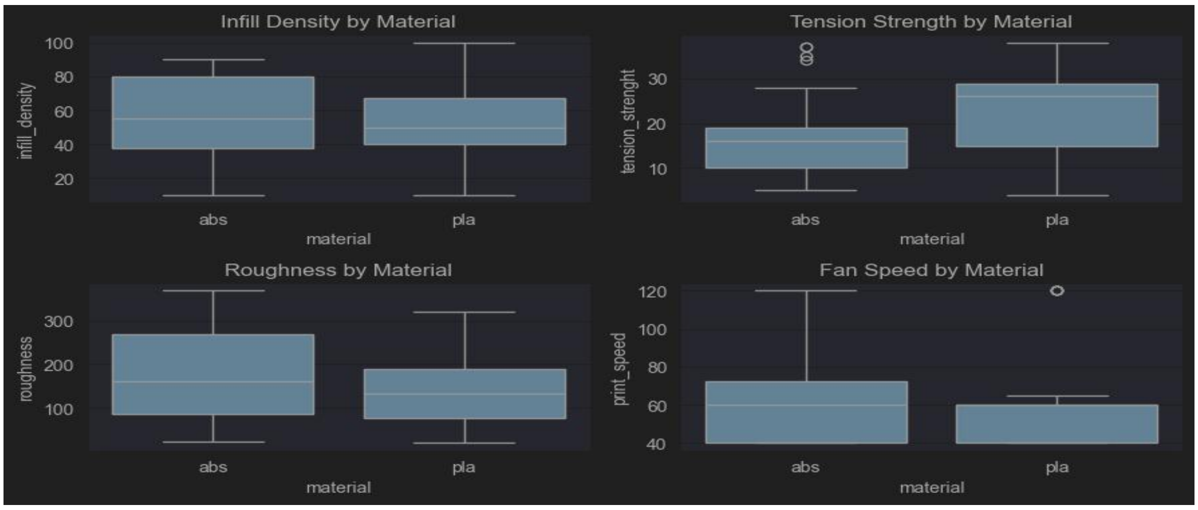
```
plt.figure(figsize=(5,5))
sns.countplot(data=DataFrame, x='material', hue='infill_pattern', palette='Set3')
plt.show()
#cross table for material and infill_pattern
cross_tab = pd.crosstab(DataFrame['material'], DataFrame['infill_pattern'])
print(cross_tab)
```



- ABS material exhibits a predominantly grid infill configuration compared to honeycomb structures, which typically consists of approximately 15 count lines; this choice may reflect an emphasis on simpler construction principles or expedited printing processes due to its characteristics.

- Polylactic Acid (PLA): This material shows an inverted trend where honeycomb patterns appear 18 times as frequently compared to grids at approximately 12 occurrences each. It might be because of PLA’s ability to support complex fill-in structures which improve both structural integrity and visual appeal.

- This analysis reveals that selecting materials impacts fill strategies significantly, guiding optimizations in areas like printing efficiency, longevity, or robustness.



- For infill density, PLA tends to have a higher median, meaning it’s usually printed with more internal fill than ABS.
- When it comes to tension strength, ABS clearly stands out—it’s stronger on average than PLA, though there are a few outliers.
- Roughness is also higher in ABS prints, which might mean they feel less smooth compared to PLA.
- And for fan speed, PLA prints generally use higher speeds, possibly to cool faster and avoid warping.

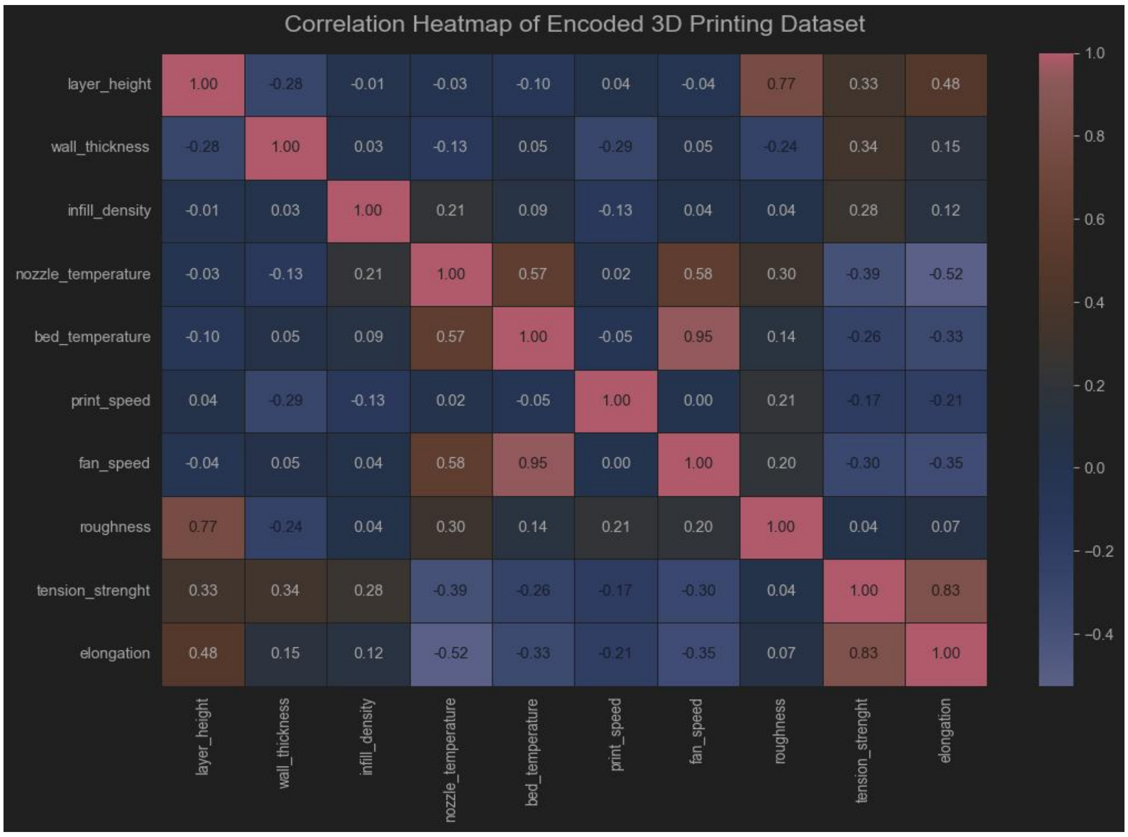
Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.

- This chart shows how different 3D printing parameters are related to each other. It’s a correlation heatmap—so the closer a value is to 1, the stronger the positive relationship; closer to -1 means a strong negative relationship.

- You can see that tension strength is highly correlated with wall density and infill density. That makes sense—more material packed in usually means stronger prints. Also, roughness seems to have some connection with fan speed and print speed, which could affect surface finish.

- Some features like layer height and bed temperature don’t show strong correlations with mechanical properties, which might mean they’re less influential in this dataset.



Encoding the Categorical Features:

```
df_encoded = pd.get_dummies(DataFrame, columns=['material', 'infill_pattern'], drop_first=True)
```

- We use one-hot encoding to convert categorical features like material and infill_pattern into numerical format.
- pd.get_dummies() creates binary columns for each category, making the data model-ready.
- Setting drop_first=True avoids multicollinearity by removing the first category from each feature.

Splitting data into train and test

- Now let’s split the Dataset into train and test sets. First split the dataset into x and y and then split the data set.

- Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
x = df_encoded.drop(['material_pla'], axis=1)
y = df_encoded['material_pla']
```

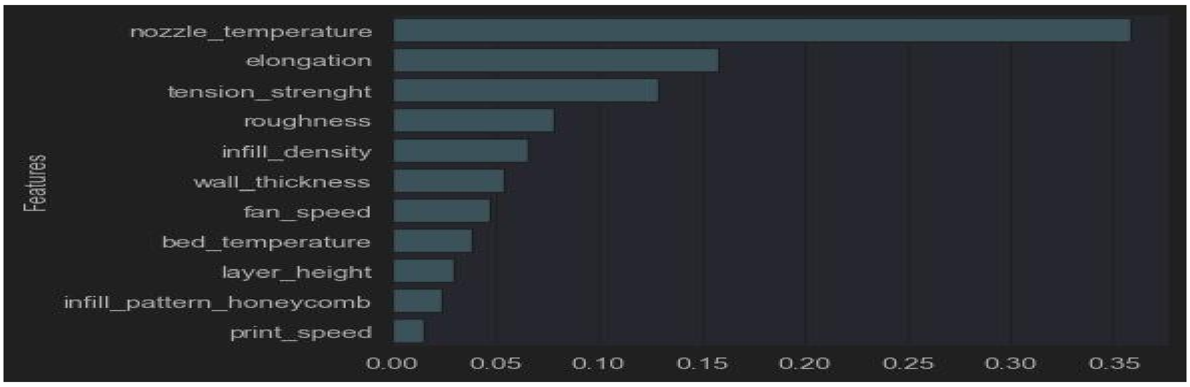
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Scaling

- Scaling is a technique used to transform the values of a dataset to a similar scale to improve the performance of machine learning algorithms. Scaling is important because many machine learning algorithms are sensitive to the scale of the input features.
- Here we are using MinMaxScaler.
- Min-Max Scaling transforms data to a fixed range using the formula: X_text_scaled= {X - X_min/X_max - X_Min } and Min-Max Scaling transforms data to a fixed range—typically [0, 1].

```
#Scaling of the data
scaler = MinMaxScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Feature selections



Using the random forest best estimator we ranked the 12 features in dataset as the dataset is small all features are considerd for the model.

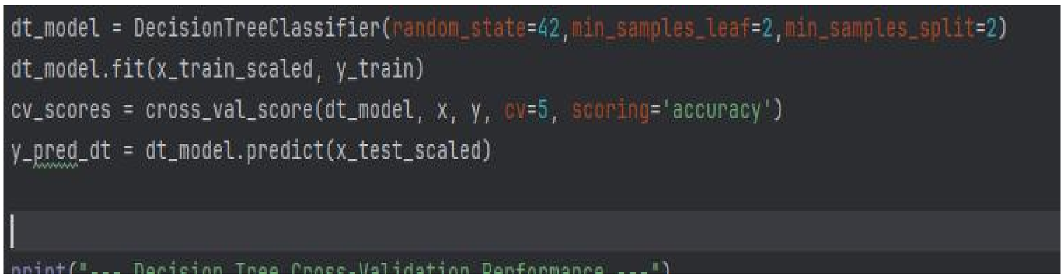
Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and its time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Decision tree model

- First Decision Tree is imported from sklearn Library then DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function.
- Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X_train and X_test.

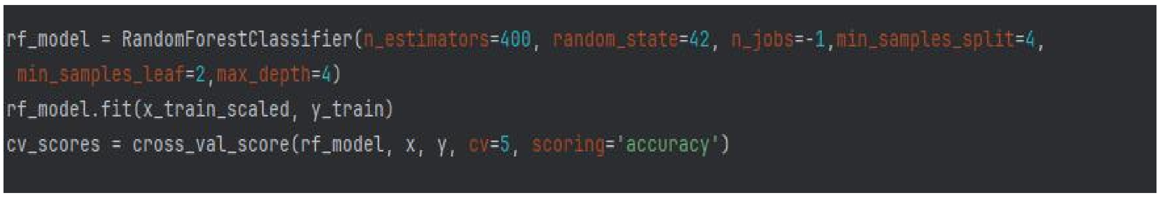


```
dt_model = DecisionTreeClassifier(random_state=42,min_samples_leaf=2,min_samples_split=2)
dt_model.fit(x_train_scaled, y_train)
cv_scores = cross_val_score(dt_model, x, y, cv=5, scoring='accuracy')
y_pred_dt = dt_model.predict(x_test_scaled)


print("--- Decision Tree Cross-Validation Performance ---")
```

Activity 1.2: Random forest model

- First Random Forest Model is imported from sklearn Library then RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function.

- Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X_train and X_test.



```
rf_model = RandomForestClassifier(n_estimators=400, random_state=42, n_jobs=-1,min_samples_split=4,
 min_samples_leaf=2,max_depth=4)
rf_model.fit(x_train_scaled, y_train)
cv_scores = cross_val_score(rf_model, x, y, cv=5, scoring='accuracy')
```

Activity 1.3: Logistic Regression model

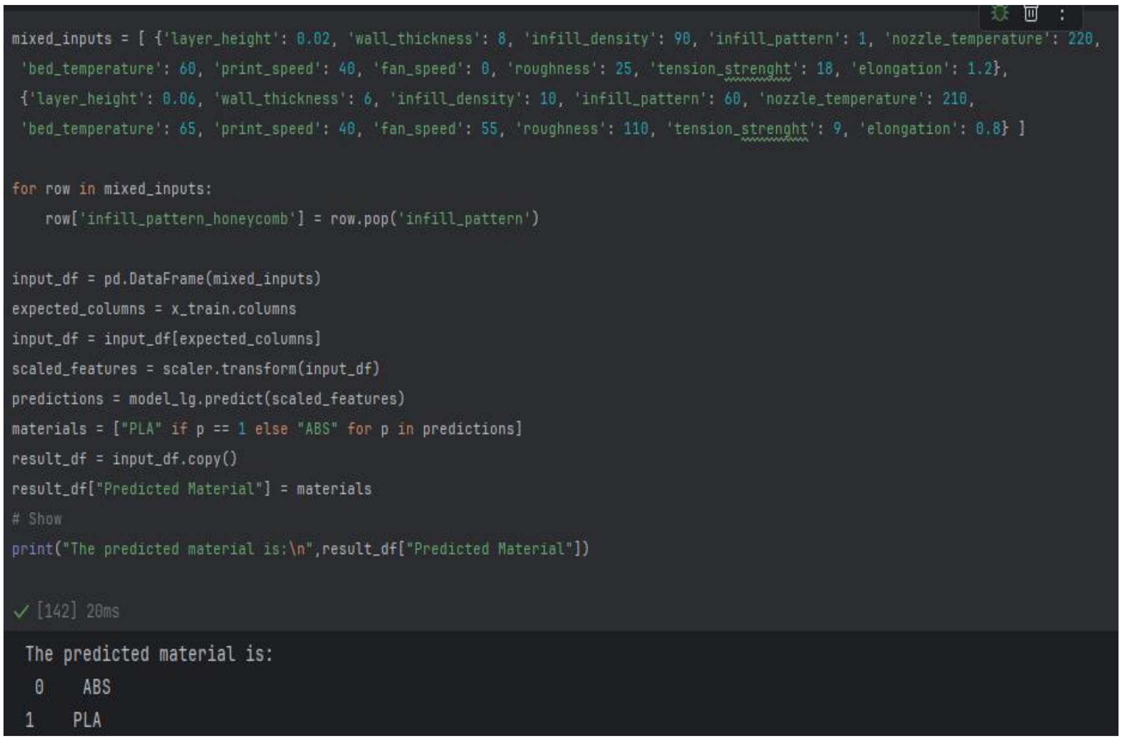
- Logistic Regression Model is imported from sklearn Library then Logistic Regression algorithm is initialised and training data is passed to the model with .fit() function.

- Scaled Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix is done.

```
#Logistic regression
model_lg = LogisticRegression(max_iter=5000, random_state=42,solver='lbfgs')
model_lg.fit(x_train_scaled, y_train)
y_pred_lg= model_lg.predict(x_test_scaled)
```

Activity 2: Testing the model

Here we have tested with Decision Tree algorithm . You can test with all algorithm. With the help of predict() function.



```
mixed_inputs = [ {'layer_height': 0.02, 'wall_thickness': 8, 'infill_density': 90, 'infill_pattern': 1, 'nozzle_temperature': 220,
 'bed_temperature': 60, 'print_speed': 40, 'fan_speed': 0, 'roughness': 25, 'tension_strenght': 18, 'elongation': 1.2},
 {'layer_height': 0.06, 'wall_thickness': 6, 'infill_density': 10, 'infill_pattern': 60, 'nozzle_temperature': 210,
 'bed_temperature': 65, 'print_speed': 40, 'fan_speed': 55, 'roughness': 110, 'tension_strenght': 9, 'elongation': 0.8} ]

for row in mixed_inputs:
    row['infill_pattern_honeycomb'] = row.pop('infill_pattern')

input_df = pd.DataFrame(mixed_inputs)
expected_columns = x_train.columns
input_df = input_df[expected_columns]
scaled_features = scaler.transform(input_df)
predictions = model_lg.predict(scaled_features)
materials = ["PLA" if p == 1 else "ABS" for p in predictions]
result_df = input_df.copy()
result_df["Predicted Material"] = materials
# Show
print("The predicted material is:\n",result_df["Predicted Material"])

✓ [142] 20ms

The predicted material is:
 0    ABS
1    PLA
```

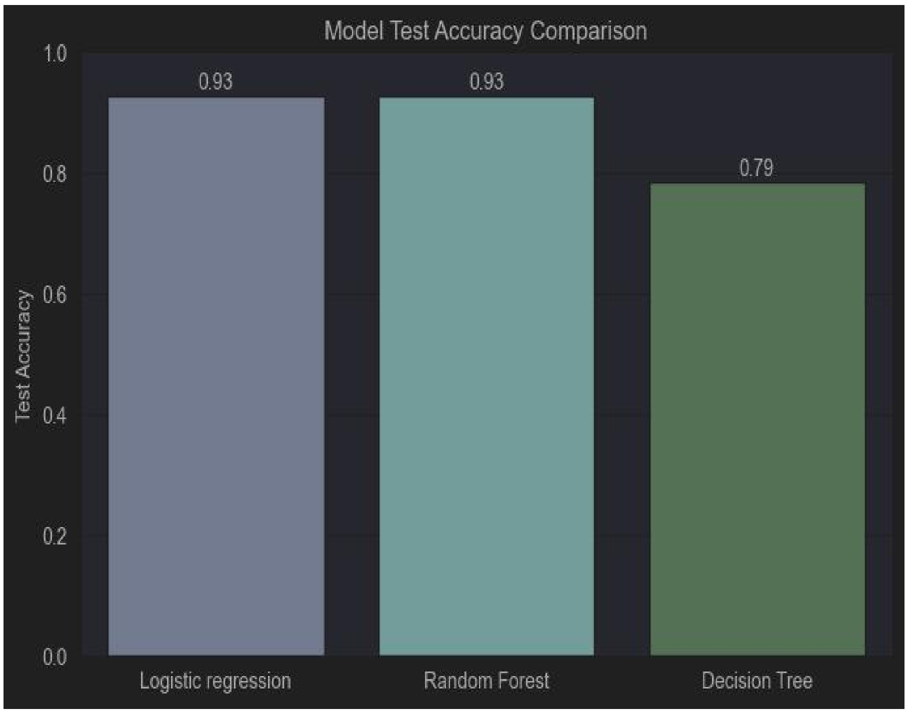
Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with multiple evaluation metrics

- Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures.
- This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

- Three models decision tree, logistic regression and random forest are compared.
- But decision tree classifier have low accuracy 79%.But both random forest and logistic regression have 93%.



- After model comparision the results of models are displayed as output.

  From the above models Logistic Regression is performing well.

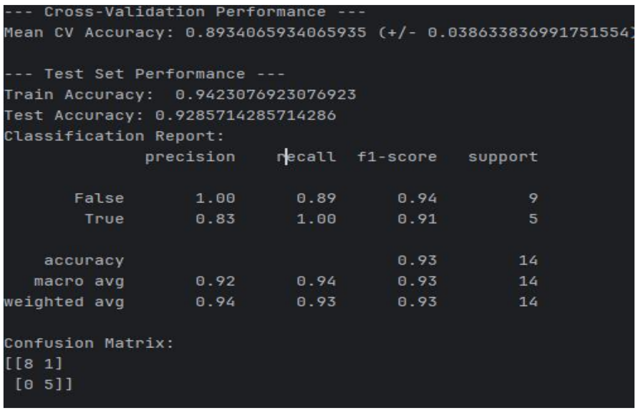
- Because cross validation is performed with cv value 5. Random forest

  with 0.09 which is not stable, but logistic regression with 0.03 is stable.

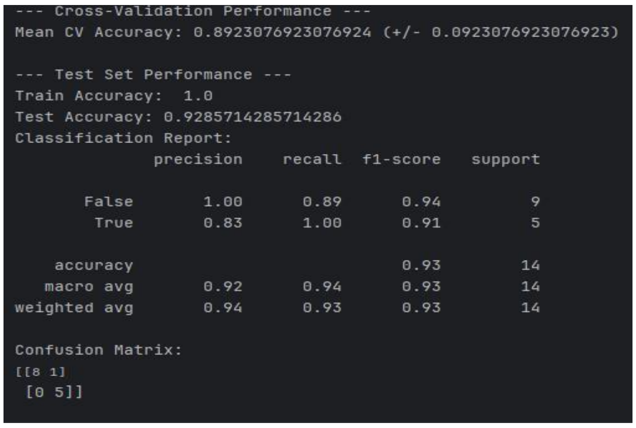
Activity 2: Comparing model accuracy before & after applying hyperparameter tuning

- Hyperparameter tuning technique is optional, for this project hyper-parameter tuning technique is not used.

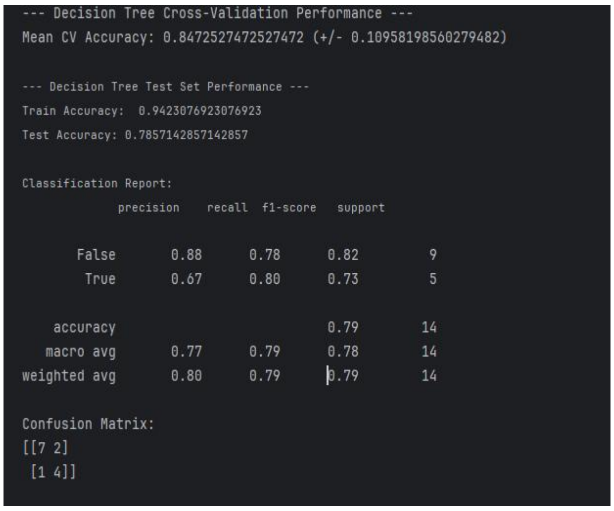
- Evaluating performance of the model From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given Random forest (model name), x, y, cv (as 5 folds). Our Logistic Regression model is performing wellso it is selected.



```
--- Cross-Validation Performance ---
Mean CV Accuracy: 0.8934065934065935 (+/- 0.038633836991751554)

--- Test Set Performance ---
Train Accuracy:  0.9423076923076923
Test Accuracy: 0.9285714285714286
Classification Report:
              precision    recall  f1-score   support

       False       1.00      0.89      0.94         9
        True       0.83      1.00      0.91         5

    accuracy                           0.93        14
   macro avg       0.92      0.94      0.93        14
weighted avg       0.94      0.93      0.93        14

Confusion Matrix:
[[8 1]
 [0 5]]
```



```
--- Cross-Validation Performance ---
Mean CV Accuracy: 0.8923076923076924 (+/- 0.0923076923076923)

--- Test Set Performance ---
Train Accuracy:  1.0
Test Accuracy: 0.9285714285714286
Classification Report:
              precision    recall  f1-score   support

       False       1.00      0.89      0.94         9
        True       0.83      1.00      0.91         5

    accuracy                           0.93        14
   macro avg       0.92      0.94      0.93        14
weighted avg       0.94      0.93      0.93        14

Confusion Matrix:
[[8 1]
 [0 5]]
```



```
--- Decision Tree Cross-Validation Performance ---
Mean CV Accuracy: 0.8472527472527472 (+/- 0.10958198560279482)

--- Decision Tree Test Set Performance ---
Train Accuracy:  0.9423076923076923
Test Accuracy: 0.7857142857142857

Classification Report:
              precision    recall  f1-score   support

       False       0.88      0.78      0.82         9
        True       0.67      0.80      0.73         5

    accuracy                           0.79        14
   macro avg       0.77      0.79      0.78        14
weighted avg       0.80      0.79      0.79        14

Confusion Matrix:
[[7 2]
 [1 4]]
```

Milestone 6: Model Deployment

Activity 1: Save the best model

- Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance.
- This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
import joblib
joblib.dump(model_lg,r"..\flask\3d_printer.pkl")
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Page:

For this project create HTML file namely index.html, home.html, about.html and result.html and save them in the templates folder.

Activity 2.2: Build Python code:

Import the libraries

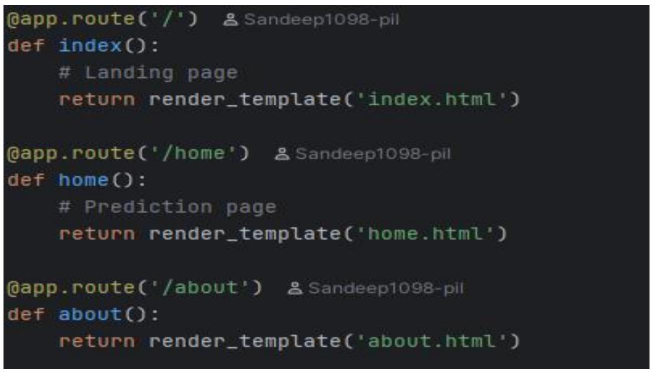
```
from flask import Flask, render_template, request
import pandas as pd
import joblib
```

Load the saved model and scaler . Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument.

- -

```
app = Flask(__name__)
model = joblib.load('3d_printer.pkl')
scaler = joblib.load('Min_max_scaler.pkl')
```

Render HTML page:

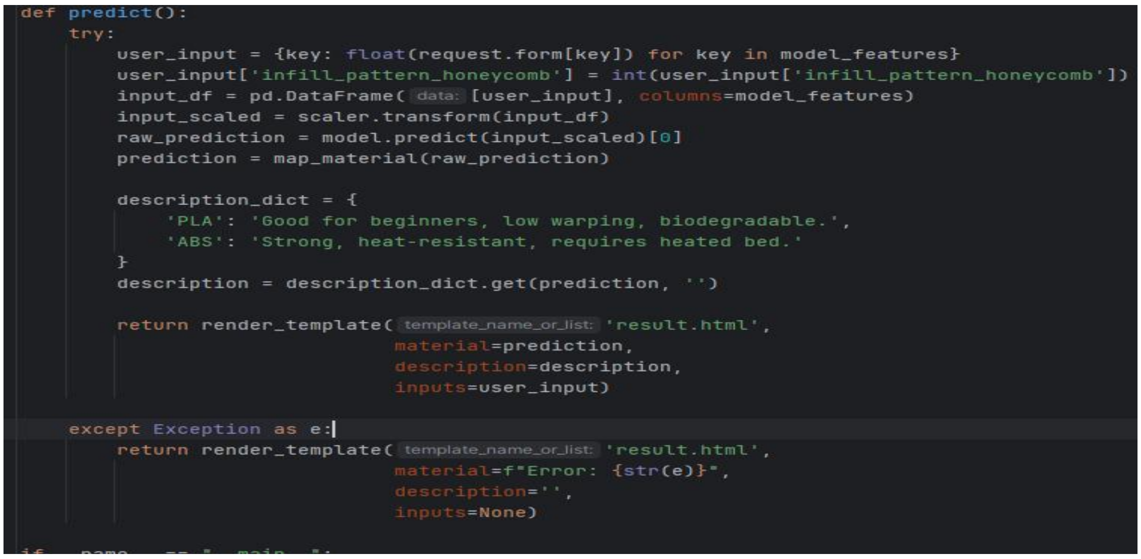


```
@app.route('/')  Sandeep1098-pil
def index():
    # Landing page
    return render_template('index.html')

@app.route('/home')  Sandeep1098-pil
def home():
    # Prediction page
    return render_template('home.html')

@app.route('/about')  Sandeep1098-pil
def about():
    return render_template('about.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



```
def predict():
    try:
        user_input = {key: float(request.form[key]) for key in model_features}
        user_input['infill_pattern_honeycomb'] = int(user_input['infill_pattern_honeycomb'])
        input_df = pd.DataFrame( data: [user_input], columns=model_features)
        input_scaled = scaler.transform(input_df)
        raw_prediction = model.predict(input_scaled)[0]
        prediction = map_material(raw_prediction)

        description_dict = {
            'PLA': 'Good for beginners, low warping, biodegradable.',
            'ABS': 'Strong, heat-resistant, requires heated bed.'
        }
        description = description_dict.get(prediction, '')

        return render_template( template_name_or_list: 'result.html',
                               material=prediction,
                               description=description,
                               inputs=user_input)

    except Exception as e:
        return render_template( template_name_or_list: 'result.html',
                               material=f"Error: {str(e)}",
                               description='',
                               inputs=None)
```

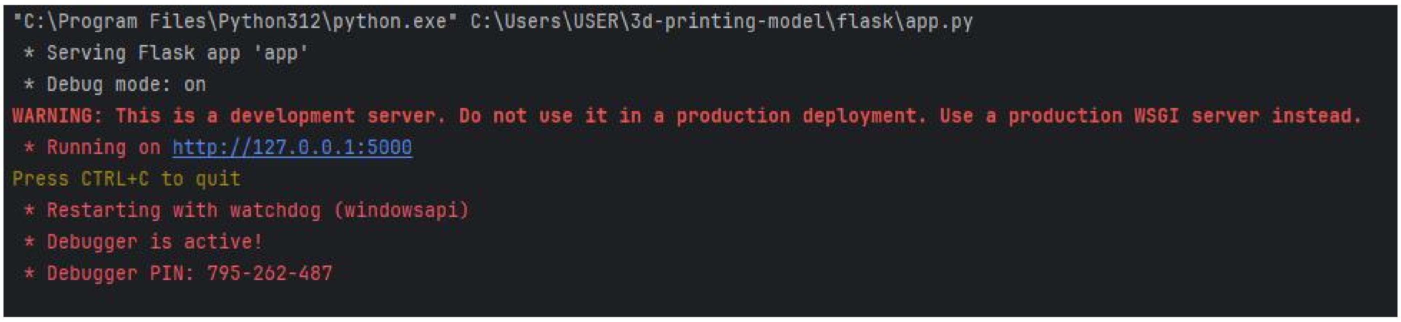
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```
if __name__ == "__main__":
    app.run(debug=True)
```

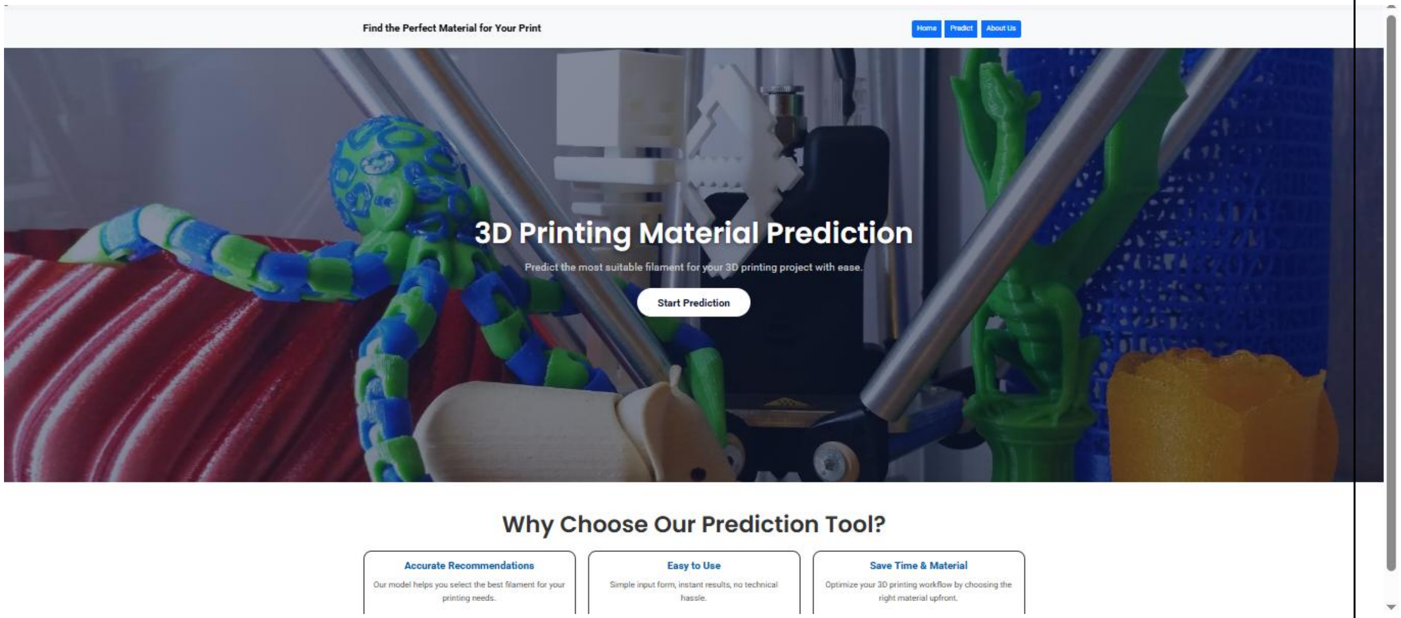
Activity 2.3: Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



```
"C:\Program Files\Python312\python.exe" C:\Users\USER\3d-printing-model\flask\app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 795-262-487
```

Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result.



Find the Perfect Material for Your Print

3D Printing Material Prediction

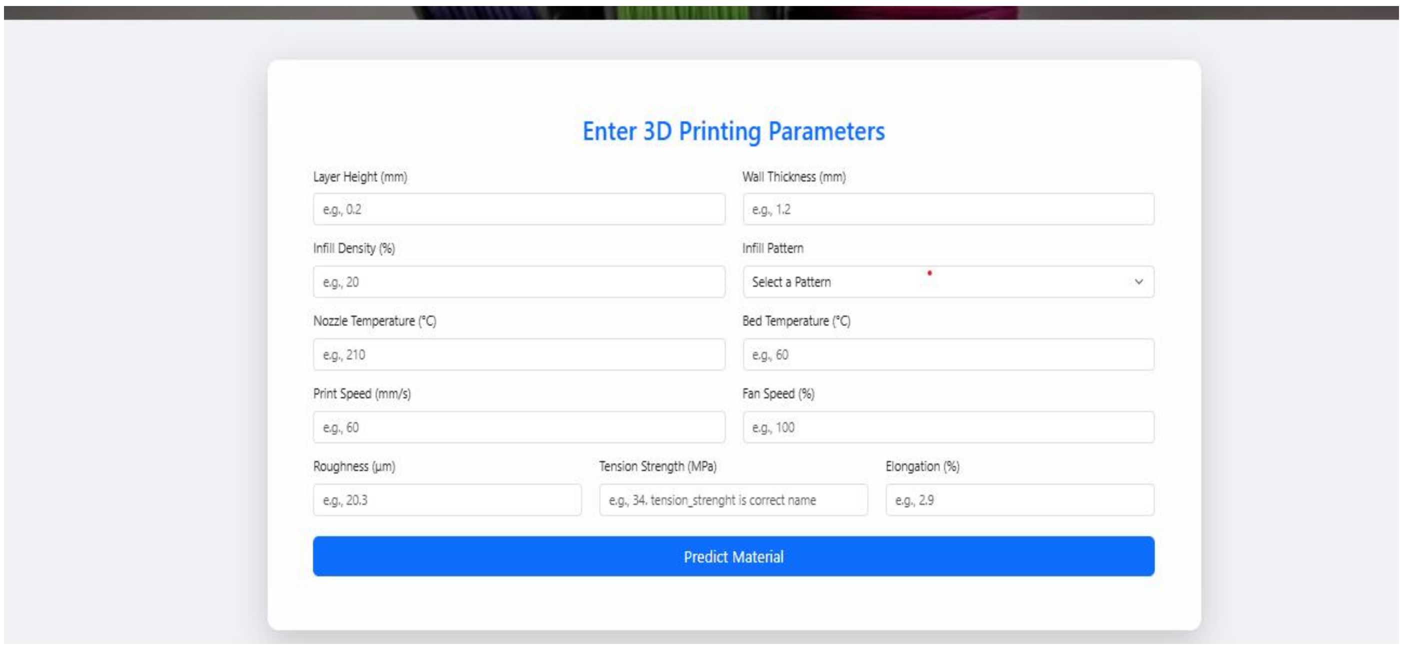
Predict the most suitable filament for your 3D printing project with ease.

Why Choose Our Prediction Tool?

Accurate Recommendations
Our model helps you select the best filament for your printing needs.

Easy to Use
Simple input form, instant results, no technical hassle.

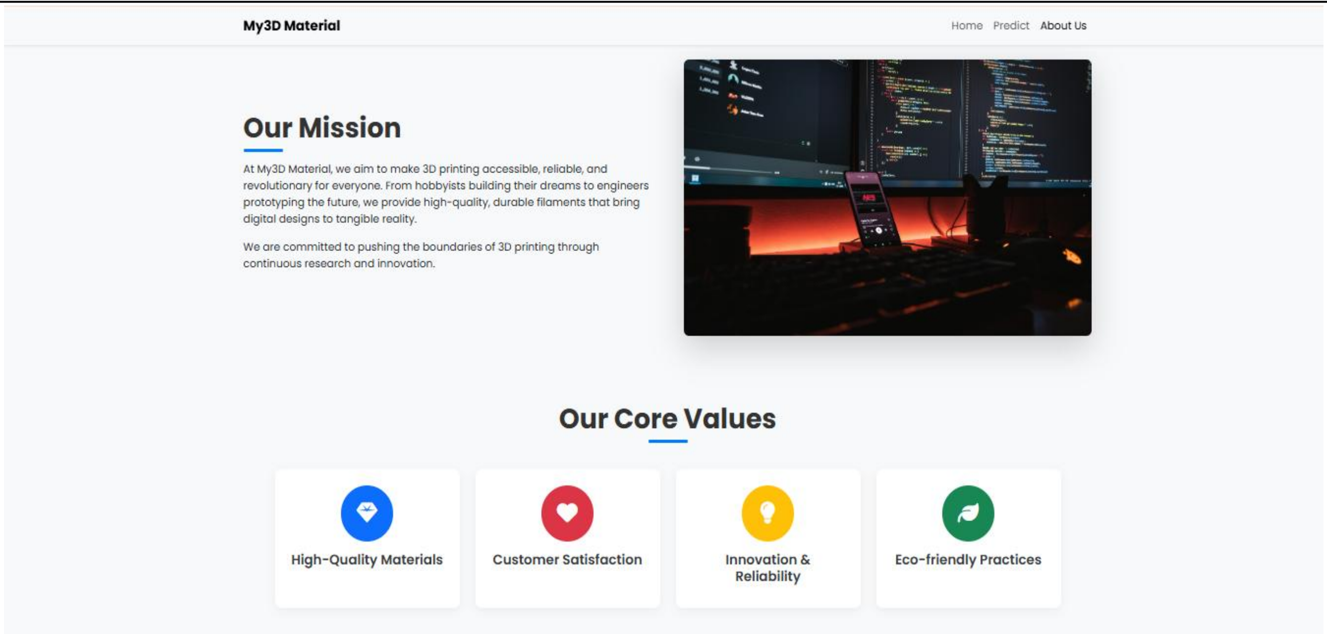
Save Time & Material
Optimize your 3D printing workflow by choosing the right material upfront.



Enter 3D Printing Parameters

Layer Height (mm): e.g., 0.2	Wall Thickness (mm): e.g., 1.2
Infill Density (%): e.g., 20	Infill Pattern: Select a Pattern
Nozzle Temperature (°C): e.g., 210	Bed Temperature (°C): e.g., 60
Print Speed (mm/s): e.g., 60	Fan Speed (%): e.g., 100

Roughness (µm): e.g., 20.3	Tension Strength (MPa): e.g., 34, tension_strenght is correct name	Elongation (%): e.g., 2.9



My3D Material

Home Predict About Us

Our Mission

At My3D Material, we aim to make 3D printing accessible, reliable, and revolutionary for everyone. From hobbyists building their dreams to engineers prototyping the future, we provide high-quality, durable filaments that bring digital designs to tangible reality.

We are committed to pushing the boundaries of 3D printing through continuous research and innovation.

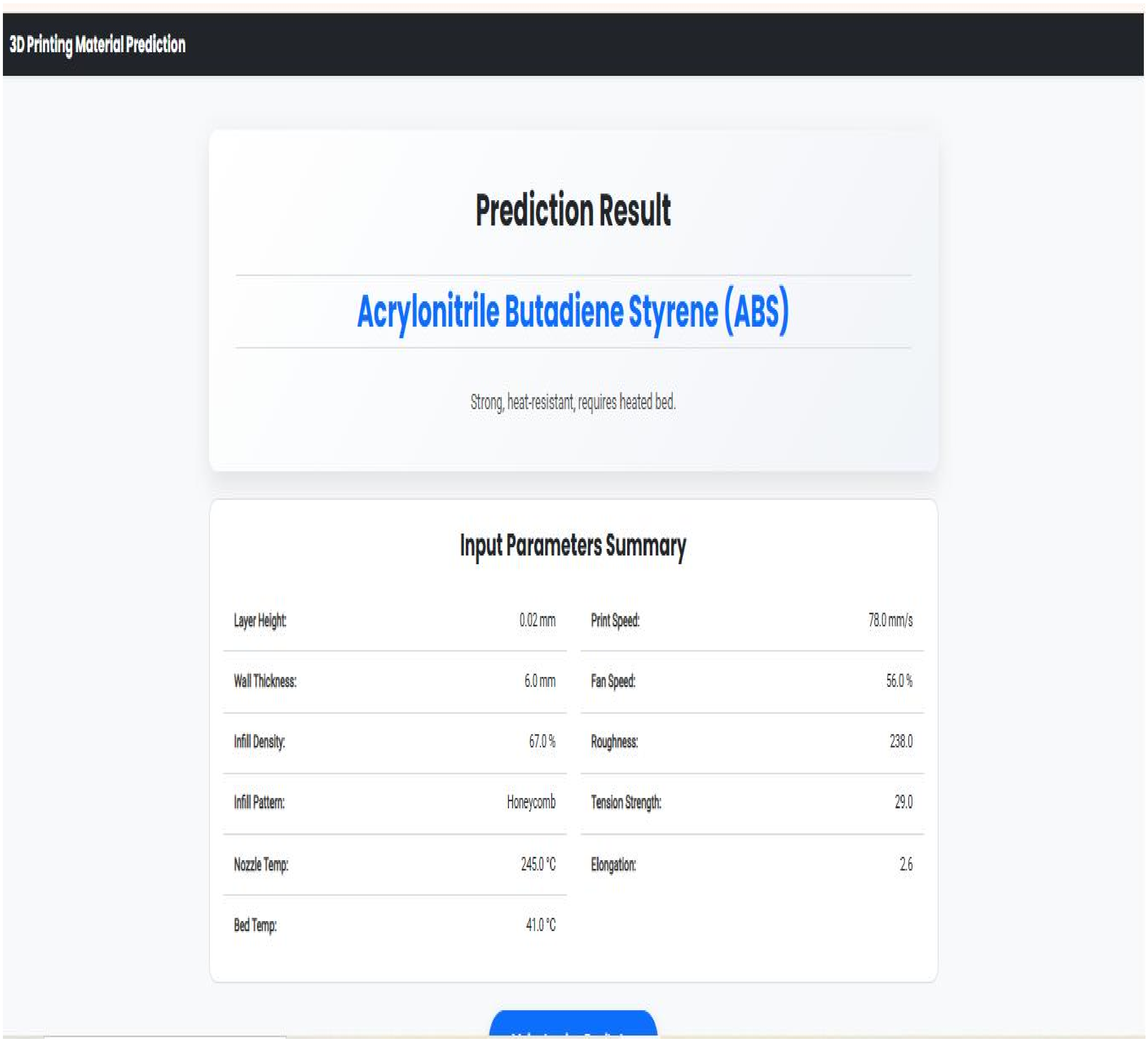
Our Core Values

High-Quality Materials

Customer Satisfaction

Innovation & Reliability

Eco-friendly Practices



3D Printing Material Prediction

Prediction Result

Acrylonitrile Butadiene Styrene (ABS)

Strong, heat-resistant, requires heated bed.

Input Parameters Summary

Layer Height:	0.02 mm	Print Speed:	78.0 mm/s
Wall Thickness:	6.0 mm	Fan Speed:	56.0 %
Infill Density:	67.0 %	Roughness:	238.0
Infill Pattern:	Honeycomb	Tension Strength:	29.0
Nozzle Temp:	245.0 °C	Elongation:	2.6
Bed Temp:	41.0 °C		

Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure

Create document as per the template provided