ME6324 – Artificial Intelligence in Manufacturing

Project Report: Predicting the filament in 3D Printing

Presented by
Group 13 (Batch-2: Group 7)

D. Bhanu Reddy: ME18B130
N. Sashank: ME18B160
P. Manoj: ME18B162

TABLE OF CONTENTS:

A. Introduction

3D Printing is the process in which we make 3D objects from a digital file, through additive processes in which material is added layer by layer. Nowadays, it has a wide range of applications in fields like medicine, military, construction and architecture, computer industry, aerospace industry, and many others, due to its virtue of rapid prototyping and low budget setup. It is increasingly used for mass customization, production of any type of open-source design in various areas.

Many factors affect the print quality of a 3D Printer, and it is more important to understand how varying the print settings show an impact on the final print we get. For instance, we may encounter a case of over extrusion, which happens due to low cooling fan speed, and when if it is high, then we may see warping in the print layers (Fig. A.1). Similarly, if the extruder temperature is high, we see stringing of layers in the model, and when it is low, we encounter layer separation (Fig. A.2). This is why, the first print we do, is not always the perfect print, and we have to look upon these features. More importantly, these features depend on the material used as well, and its heat resistant properties among others. Even if one of these features goes off the required range, errors in the print creep in, which is commonly seen in small-scale 3D printing setups.

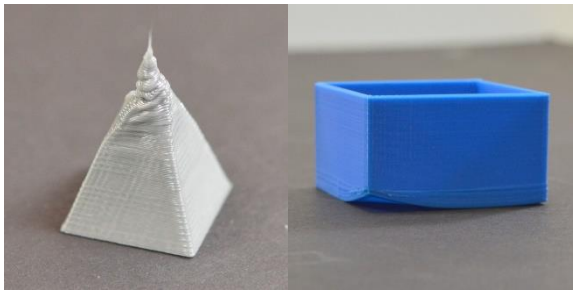

Fig. A.1 Over-extrusion and Warping

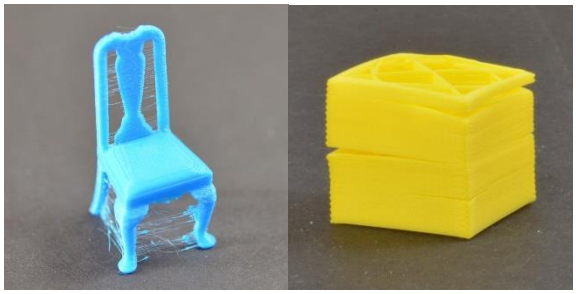

Fig. A.2 Stringing and Layer Separation

So through our project, we attempt to make an intelligent system that predicts the material used with all the parameters together to give accurate results that can be applied in real-time. For this, we have trained our model using a dataset that has 10 input features (Fig 1.3) which are used to predict the single output feature, that is, the material used: ABS (Acrylonitrile Butadiene Styrene) and PLA (Polylactic Acid). We have obtained this dataset from Kaggle, which was contributed by TR/Selcuk University, Turkey. These prints were performed on an Ultimaker S5 3D printer.

Input Features	
Layer Height	(mm)
Print Speed	($mm.s^{-1}$)
Fan Speed	(%)
Nozzle Temperature	(°C)
Bed Temperature	(°C)
In-fill Density	(%)
Wall Thickness	(mm)
Roughness	(µm)
Tensile Strength	(MPa)
Infill pattern	(Grid or Honeycomb)

→

Output Feature
Material: • ABS • PLA

Fig. A.3 Table diagram representing the input and output features with appropriate dimensions

Image courtesy Fig. A.1, A.2 - https://www.simplify3d.com/support/print-quality-troubleshooting/
Dataset csv file for download - https://drive.google.com/file/d/1BmbOqgQmbz06GHovRqURXRpeDXieFJVg/view?usp=sharing

B. Implementation

In the initial stage, we planned to use decision trees to predict the material used from the given parameters. But, there would be a problem of overfitting in that approach. We then rectified that possibility by extending our approach to random forests. (discussed below)

B.1. Decision Trees:

Decision Trees is a flowchart-like structure, that helps in deciding on a particular process. It is a supervised learning method used for classification and regression. Decision trees use entropy, Gini index, and information gain as the metrics.

Gini index or Gini impurity measures the degree of probability of a particular variable being wrongly classified when it is randomly chosen.

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

Where p_i is the probability of an element being classified into a specific class.

One of the most common problems, that we faced here, is that they tend to overfit a lot. These are situations where a tree is built with some features of the train set and the predictive power for unseen data is reduced. This happens when more weight is given to some set of features.

B.2. Random Forests:

Random Forest is a machine learning technique that is used to solve regression and classification problems. As the name implies, they consist of a large number of individual decision trees that operate as an ensemble. A large number of uncorrelated trees operate together and outperform any of the individual constituent models, and prevent the occurrence of overfitting. It uses bagging which is:

Bagging (Bootstrap Aggregation):
Decisions trees are very sensitive to the data they are trained on. small changes to the training set can result in significantly different tree structures. Random forest takes advantage of this by allowing each tree to randomly sample from the dataset with replacement, resulting in different trees. This process is known as bagging.

So in a random forest, we end up with trees that are not only trained on different sets of data but also use different features to make decisions.

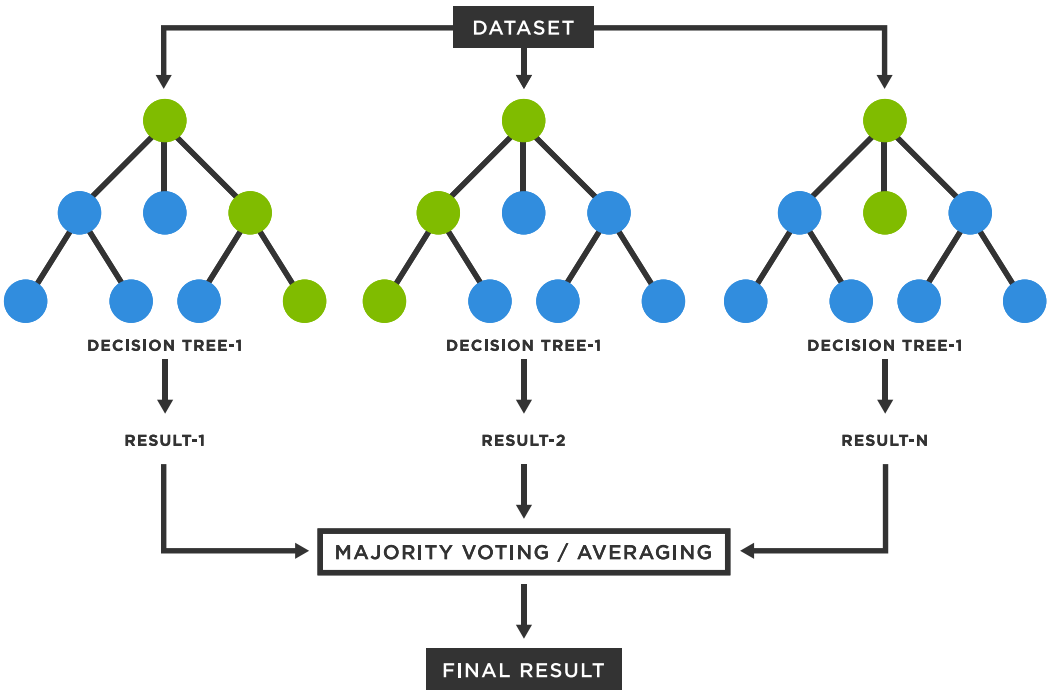

Fig. B.1 Schematic of Random Forests

C. Procedure

The tools and software we used are Python and Anaconda for libraries like *pandas*, *NumPy*, *sci-kit-learn*, *matplotlib* for computing and visualizing data. Below are the steps are taken while implementing our idea on code, with a few pictures of it to have a brief idea at the critical points.

1. Firstly, we load all the important libraries (as mentioned above) and load the data.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,DecisionTreeRegressor
from sklearn.metrics import classification_report,r2_score
```

Fig. C.1. Imported Libraries

2. After loading, we convert the categorical variables (material and in-fill pattern) into 0s and 1s.

	layer_height	print_speed	fan_speed	bed_temperature	nozzle_temperature	infill_density	wall_thickness	roughness	strength	infill_pattern	material
0	0.02	40	100	80	220	24	10	10	10	1	1
1	0.06	60	65	73	237	60	5	155	13	1	0
2	0.20	40	45	69	209	53	4	262	262	1	1
3	0.06	60	80	76	242	69	16	179	11	0	0
4	0.20	40	70	74	214	44	6	240	27	0	1

Fig. C.2 First five entries of the dataset

3. Now, we split the data into train and test sets with 75 and 25 percent of the data respectively.

4. Use the decision tree classifier now to fit X (train data excluding material attribute) and Y (train data's material attribute).

```
model  = DecisionTreeClassifier(random_state=0,min_samples_leaf=20)

features = list(train.columns)
features.remove('material')
X = train.loc[:,features]
y = train['material'] # pla = 1; abs = 0

model.fit(X,y)
```

Fig. C.3. Decision Tree Classifier

5. Then we test this model with the test (validation) data. We are using limited features here and giving more weight to them which causes overfitting. So, we now look into its extension, Random Forests.

```
model = RandomForestClassifier(random_state=0,min_samples_leaf=10)
model.fit(X,y)
y_pred = model.predict(X_test)
output = model.predict_proba(X_test)
```

Fig. C.4. Random Forest Classifier training on X and Y

6. Obtaining the tree diagrams and the accuracy scores.

Link to the google colab notebook: https://drive.google.com/file/d/18nxa3XrRoMay0pvquzILJ_x8yLoq7CSq/view?usp=sharing

D. Observations and Results

The Gini index is used for classification based on parameters. In the below tree diagram, Infill density <= 48.5 which is classified under ABS is used as an initial feature for classification. The 2nd layer is further classified based on nozzle temperature and strength. This classification based on the features further continues up to when then leaves become less than 10.

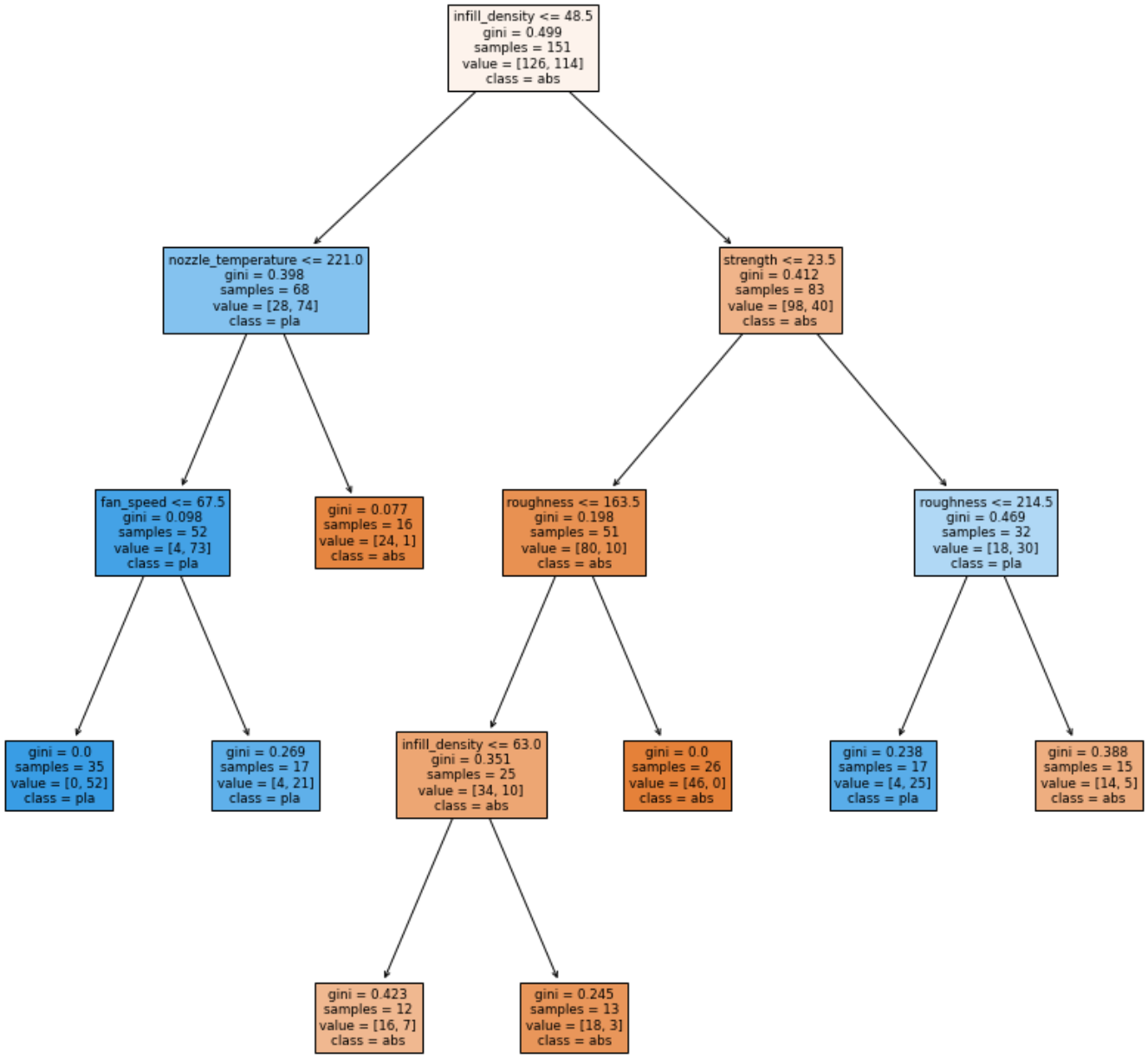

Fig. D.1: Output tree with in-fill density as the first chosen feature for splitting.

This is just one of the output trees shown as an example. The code we have written outputs multiple such trees taking different features as parameters and returning the trees with layers as it converges to a point with no further split.

The following figure shows the developed confusion matrix for the problem at hand.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Fig. D.2. Confusion Matrix

The TP, FP, FN, TN are obtained as 41, 0, 1, 38 respectively wrt abs (categorically 0) and 38, 1, 0, 41 concerning the categorical 1 variable (which is PLA here).

Formulae for the respective metrics:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Negative + True\ Negative + False\ Positive}$$

We have shown these calculations through the code as well:

```
print(classification_report(y_pred,y_test))
print("R2 Score is: ",r2_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        42
           1       0.97      1.00      0.99        38

    accuracy                           0.99        80
   macro avg       0.99      0.99      0.99        80
weighted avg       0.99      0.99      0.99        80

R2 Score is:  0.9499687304565353
```

The R2 Score of the prediction turned out to be 0.9499, which can be safely approximated as 0.95, and it is a really good score, indicating that our trained model has accurately predicted results for the test data set, and can be used to predict any further un-seen data parameters as an intelligent system in real-time as well.