ISEN613 ENGINEERING DATA ANALYSIS
FINAL PROJECT REPORT

PREDICTIVE MODELLING TO PREDICT THE POROSITY USING MACHINE LEARNING

TEAM #18

#MEMBER	NAME	INDIVIDUAL CONTRIBUTION (%)
1	AKASH PANTHALAGUDI SUNDARRAJA SESHADRI	20%
2	AAKASH BALAJI BALA VIGNESH	20%
3	AJITH KUMAR SATHAPPA SUBRAMANIAN	20%
4	MOHAMED NIYAAS SIRAJUDEEN	20%
5	PRADEEP MANOHARAN	20%

CONTENTS

INTRODUCTION

Additive manufacturing has evolved as one of the widely used and accepted means of manufacturing and prototyping. This rapid development of AM in industries is for making manufacturing more convenient and time efficient when compared with traditional manufacturing. The applications of this are present across wide domains like engineering, medicine, marketing, etc., AM can be used to make plastics, ceramics, polymers, and metallic models of various shapes and features, this breakthrough in manufacturing is trivial for rapid improvement in product development.

IMPORTANCE OF THE PROBLEM:

For this project on additive manufacturing, we have made use of the hybrid additive machine. This hybrid machine combines additive manufacturing and machining process that uses lasers, metal powders, and heat treatments, to modify the structure and properties of the created materials. For collecting data about the material, this process can be automatically controlled by using a computer. The machine uses high-speed cameras, sensors, and other instruments that allow faster data collections which will speed up the discovery of the properties of the materials. Around one gigabyte of data is generated every min by this machine, this data along with the discoveries from the machine are speeding up smart material discovery for the future world.

OBJECTIVE:

The main objective of this project is to predict the surface porosity of the surface being machined. The feature data for printing, milling, and combined machining are then trained and tested using machine learning models like Random Forest, XGBoosting, LDA, QDA, and Logistic Regression. From these ML models, we choose the most accurate one for this project and predict the surface being machined as a porous or non-porous surface.

SPECIFIC, SCOPE OF WORK:

The Machine Learning model is built to predict the output response (% of porosity) based on the input tensor data. This model is highly useful in predicting the continuous characteristic of the surface being machined. If there are some wide deflections in the graph being plotted from the response generated we can know there is something wrong with the machine so that we can run a complete diagnosis of the hybrid additive machine and can make sure that the surface being machined is porous free.

PROJECT APPROACH

DATA DESCRIPTION:

As part of the dataset, we have been provided with a set of images (both black and white and raw images) of surfaces that have been machined using 3D printing and milling. These images give information about the porosity of the machined surface which may be caused by the machining process. A total of 100 images from each machining condition have been provided to us. The characteristic of the sample is as follows:

- **Sample 1** has been converted into Black and White, the machining condition for sample 1 is a Laser power of 250 W, a scan speed of 10 inches/min, and a Powder flow rate of 4 RPM
- **Sample 4** are raw images, the machining condition for sample 4 is a Laser power of 420 W, a scan speed of 20 inches/min, and a Powder flow rate of 4 RPM.

The dataset also contains two MATLAB files containing spectrogram data associated with each sample.

- The first MATLAB (Sample 1) file has spectrogram data in the following format: Tensor voxel_{index} and 129 x 15 x 6 which corresponds to 129 x 15 spectrograms for M1 P1 P2 M2 P3 P4 (M - Milling, P - Printing).
- The second MATLAB (Sample 4) file has a spectrogram in the following format: Tensor voxel_{index} and 129 x 6 x 4 which corresponds to 129 x 6 spectrograms for P1 P2 P3 P4 (M – Milling, P - Printing).

Here 129 corresponds to the Frequency bands for each spectrogram, 15 corresponds to the time steps and 6 corresponds to the channels of each spectrogram. An excel sheet provided to us contains properties corresponding to the 90 voxels given to us in the MATLAB file. The properties we have been given in that excel are:

1. Pore Count
2. Total Pore Area (Pixel 2)
3. Average Pore Size (Pixel 2)
4. % Area Porosity

There is a python code given to us for inputting all the spectrograms and implementing a CNN model which we have used as a foundation to develop our model. The dataset can be used to predict porosity in three different ways.

1. A Combined spectrogram using all six signals (4- Printing, 2-Milling)
2. A 4-signal spectrogram containing only printing signals indicating that the porosity of the surface can be predicted after the printing process.
3. A 2-signal spectrogram containing only milling signals indicating the porosity of the surface that can be predicted after the milling process.

The target y here is coded as 0 or 1 since we are doing a classification problem. We have used the %area porosity as the property to code as 0 or 1. From our research, we were able to infer that for a surface to be classified as a highly porous surface or low porous surface it depended on the base material used, the machining process, and the application of the product. We decided to use a 1.5% porosity area to classify it as high or low to avoid an imbalance class. We have divided the data into training and testing datasets in the ratio of 4:1. The dataset is picked out at random using a random seed value of 42 to maintain both randomness and to ensure repeatability. So, the 72 data points in the training dataset are picked out at random.

Apart from this, a python code is given to us to input all the spectrogram data and implement the CNN model, which we have used as a foundation to develop our ML model.

FLOWCHART:

With the help of the spectrogram data and code of the CNN model given to us, we have developed a feature extraction model using CNN to reduce the 4-dimensional data given to us into 2-dimensional data. Feature extraction builds a derived set of values that facilitates the model for the generalization of unseen data. Deriving the subset of the initial features is called feature extraction. The selected features are expected to contain relevant, nonredundant information which can help us in the fast convergence of the model. The extracted 2-dimensional features are then modeled using the following ML methods:

1. Random Forest
2. XGBoosting
3. LDA
4. QDA
5. Logistic Regression

We are going to separately use the above-mentioned 3 datasets and separately extract information from each of the data datasets modeling it first through feature extraction of shape (90 x 4096) 90 indicating the 90 data points given to us and 4096 representing the 129 x 6/4/2 x 15 features. This 90 x 4096 is used as the input for all the above-mentioned 5 classification methods.

The dataset is split into training and testing with 4:1. The training dataset is then split into 9 folds for stratified k-fold cross validation which is used in the case of imbalanced datasets.

REASON BEHIND THIS APPROACH:

Since this is a classification method we have decided to use these methods because prominently these methods have shown us the ability to give good results. Also, we have developed a CNN model for the same 4-dimensional dataset for all three conditions (Combined, Printing, and Milling), and the results we observed were not sufficient. Our model was predicting all 1s since the number of 1's in the dataset is greater than the number of 0's. This is a classic case of an imbalanced dataset. Even though the difference between the number of 1s and a number of 0's is 4 the case of imbalance arises because due to the random shuffling of the data points there are 41 1's and 31 0's in the training dataset and 6 1's and 12 0's in the testing dataset. But when we developed these ML models we observed comparatively better results, which we will explain in the later sections. Moreover, 4-dimensional datasets cannot be modeled using a lot of methods. We have also tried regressing the total area of the pores using the CNN model, but we were getting a negative R^2 score which shows us that our model performance is not sufficient. Then we tried the feature extraction technique using the CNN model and then using Random Forest and XGBoosting to regress the total pore area and the result we were getting was again negative R^2 scores for the test data which shows us that the model is not generalizing well.

IMPLEMENTATION DETAILS

PROJECT IMPLEMENTATION

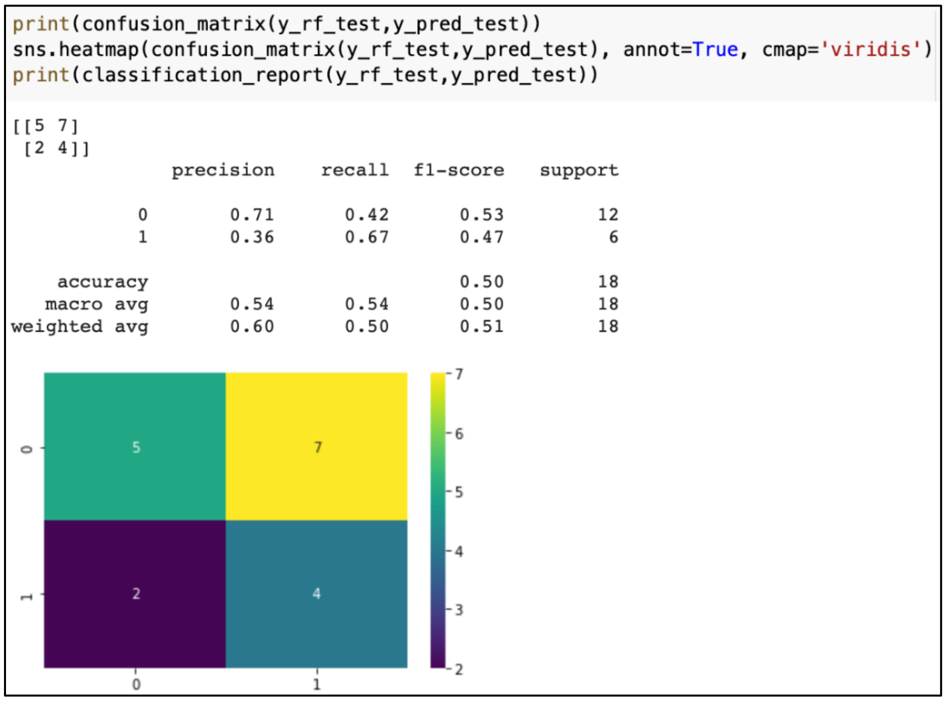
As mentioned above we have used the CNN model to extract features and 5 machine learning methods on the extracted features to predict whether the porosity is high or low on all three datasets and the results are explained below.

1. COMBINED DATASET

a) Random Forest

```
Cross Validation Scores:  [0.750 0.500 0.625 0.375 0.250 0.625 0.250 0.375 0.625]
Average CV Score:  0.4861111111111111
Number of CV Scores used in Average:  9
```

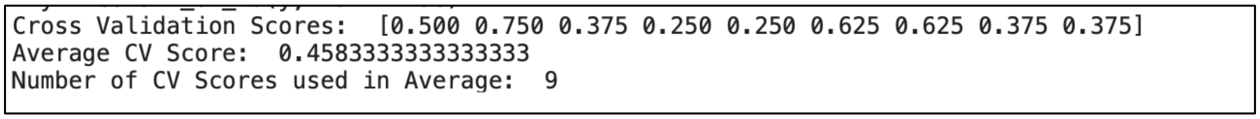
The average CV score obtained for the random forest is 0.486 which is the average accuracy obtained over all the folds and we can see that the accuracy has a range of 0.25-0.75 which indicated in some folds it is underfitting. From the test dataset implementation, we can infer that the accuracy is 0.5, and also the independent class accuracies also do not vary that much when compared to the overall test accuracy. But we can still note that the model tends to predict more 1s than 0’s which may be because there are more 1’s than 0's in the training dataset.



```
print(confusion_matrix(y_rf_test,y_pred_test))
sns.heatmap(confusion_matrix(y_rf_test,y_pred_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred_test))
```

```
[[5 7]
 [2 4]]
              precision    recall  f1-score   support

           0       0.71      0.42      0.53        12
           1       0.36      0.67      0.47         6

    accuracy                           0.50        18
   macro avg       0.54      0.54      0.50        18
weighted avg       0.60      0.50      0.51        18
```

b) XGBoosting

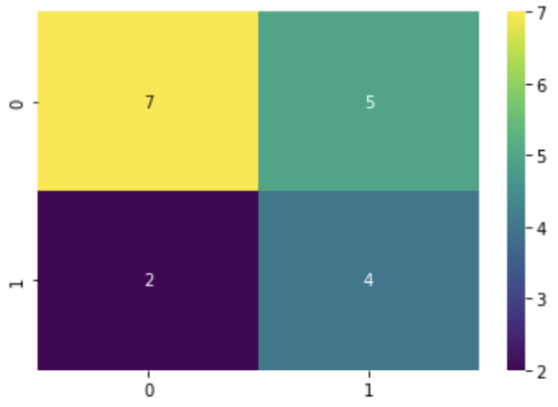


```
Cross Validation Scores:  [0.500 0.750 0.375 0.250 0.250 0.625 0.625 0.375 0.375]
Average CV Score:  0.4583333333333333
Number of CV Scores used in Average:  9
```

The average CV score obtained for XGBoosting is 0.458 which is the average accuracy obtained over all the folds and we can see that the accuracy has a range of 0.25-0.75 like a random forest which indicates that in some folds it is underfit. From the test dataset implementation, we can infer that the accuracy is 0.61 and the independent class accuracies are also very close to the overall test accuracy. Moreover, we can still note that the model tends to predict equal 1s and 0s.

```
print(confusion_matrix(y_rf_test,y_pred_test))
sns.heatmap(confusion_matrix(y_rf_test,y_pred_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred_test))
```

```
[[7 5]
 [2 4]]
              precision    recall  f1-score   support

           0       0.78      0.58      0.67        12
           1       0.44      0.67      0.53         6

    accuracy                           0.61        18
   macro avg       0.61      0.62      0.60        18
weighted avg       0.67      0.61      0.62        18
```



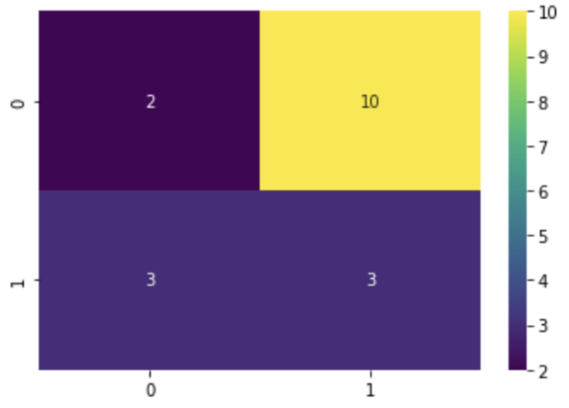
c) Linear Discriminant Analysis (LDA)

```
Cross Validation Scores:  [0.375 0.250 0.500 0.250 0.500 0.375 0.250 0.500 0.500]
Average CV Score:  0.3888888888888889
Number of CV Scores used in Average:  9
```

We can see from the above results that the average accuracy score is somewhat better than the rest of the two methods as we get a score of 0.389 which is the lowest when compared to the rest of the methods for the combined dataset and we can see that the accuracy has a range between 0.25-0.5. When we see the test dataset, we can infer that the overall test accuracy is very less when compared to the rest of the above methods. Moreover, we can notice that the class accuracies are also very less indicating that the model is not learning the relationship between the predictor and the target properly as we can see that the model predicts more 1s than 0's when the true value is 0. Also, the model predicts equals 1's and 0's when the true value is 1.

```
print(confusion_matrix(y_rf_test,y_pred_test))
sns.heatmap(confusion_matrix(y_rf_test,y_pred_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred_test))
```

```
[[ 2 10]
 [ 3  3]]
              precision    recall  f1-score   support

           0       0.40      0.17      0.24        12
           1       0.23      0.50      0.32         6

    accuracy                           0.28        18
   macro avg       0.32      0.33      0.28        18
weighted avg       0.34      0.28      0.26        18
```



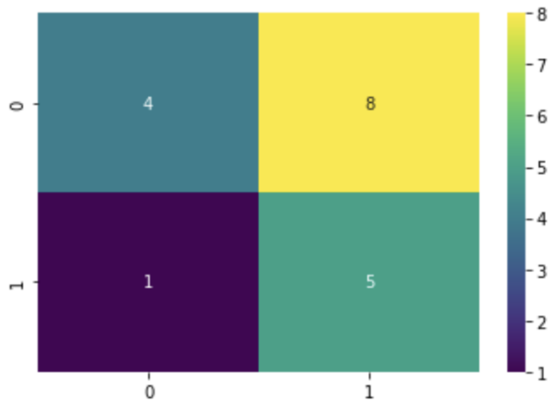
d) Quadratic Discriminant Analysis (QDA)

```
Cross Validation Scores:  [0.250 0.625 0.375 0.500 0.500 0.625 0.625 0.875 0.500]
Average CV Score:  0.5416666666666666
Number of CV Scores used in Average:  9
```

The average accuracy of the k-fold stratified cross-validation for QDA is 0.542 and the range is 0.25 -0.875. The overall test accuracy of QDA is like that of the random forest except that there is a lot of variation between the class accuracies with class 0 having an accuracy of 0.33 and class 1 having an accuracy of 0.83. Like LDA, QDA also predicts class 1 more often than class 0. This may be the reason for the difference in the individual accuracies of the k-fold cross-validation.

```
print(confusion_matrix(y_rf_test,y_predicted1_test))
sns.heatmap(confusion_matrix(y_rf_test,y_predicted1_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_predicted1_test))
```

```
[[4 8]
 [1 5]]
              precision    recall  f1-score   support

           0       0.80      0.33      0.47        12
           1       0.38      0.83      0.53         6

    accuracy                           0.50        18
   macro avg       0.59      0.58      0.50        18
weighted avg       0.66      0.50      0.49        18
```



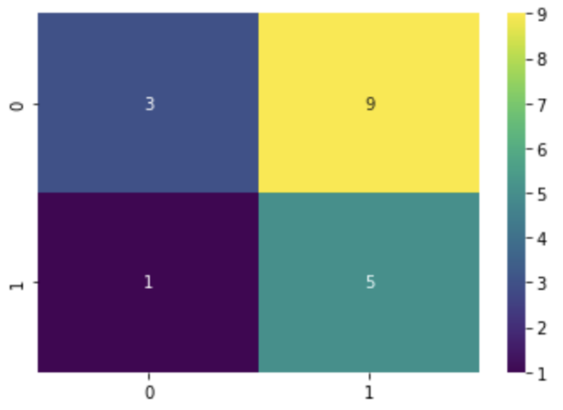
e) Logistic Regression

```
Cross Validation Scores:  [0.500 0.375 0.500 0.375 0.625 0.500 0.500 0.500 0.500]
Average CV Score:  0.4861111111111111
Number of CV Scores used in Average:  9
```

Logistic Regression has an average accuracy of 0.486 with a range of 0.375-0.625 and 5 of the 8 folds have an accuracy of 0.5. The overall test accuracy is 0.44 which is somewhat near the average CV score. However, like LDA and QDA it also suffers from a large variation in class accuracies as it also tends to predict more 1s than 0's more so than QDA.

```
print(confusion_matrix(y_rf_test,y_pred))
sns.heatmap(confusion_matrix(y_rf_test,y_pred), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred))
```

```
[[3 9]
 [1 5]]
              precision    recall  f1-score   support

           0       0.75      0.25      0.38        12
           1       0.36      0.83      0.50         6

    accuracy                           0.44        18
   macro avg       0.55      0.54      0.44        18
weighted avg       0.62      0.44      0.42        18
```



2. PRINTING DATASET

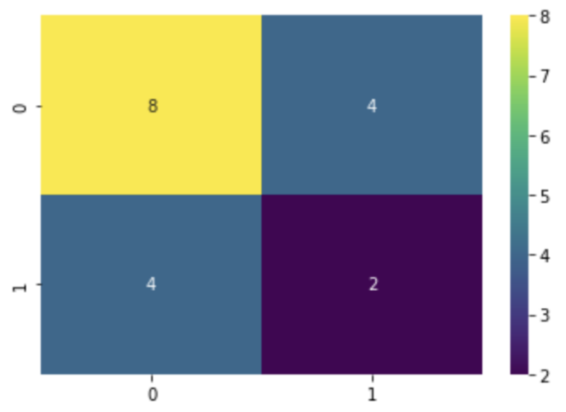
a) Random Forest

```
Cross Validation Scores:  [0.250 0.500 0.375 0.375 0.375 0.500 0.375 0.500 0.625]
Average CV Score:  0.4305555555555556
Number of CV Scores used in Average:  9
```

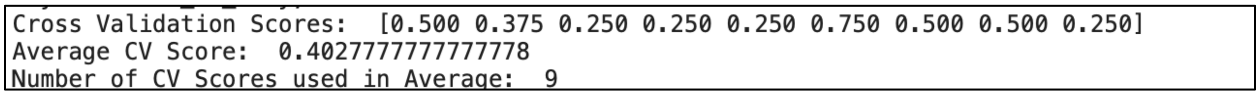
The average CV score obtained for the random forest is 0.430 for the printing dataset which is less than the accuracy we obtained for the combined dataset, and we can see that the accuracy has a range of 0.25-0.625. From the test dataset implementation, we can infer that the accuracy is 0.56 and the independent class accuracies have a lot of variation. But the interesting aspect is the model tends to predict more 0's than 1's due to which class 0 has an accuracy of 0.67 which is more when compared to class 1 accuracy of 0.33.

```
print(confusion_matrix(y_rf_test,y_pred_test))
sns.heatmap(confusion_matrix(y_rf_test,y_pred_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred_test))
```

```
[[8 4]
 [4 2]]
              precision    recall  f1-score   support

           0       0.67      0.67      0.67        12
           1       0.33      0.33      0.33         6

    accuracy                           0.56        18
   macro avg       0.50      0.50      0.50        18
weighted avg       0.56      0.56      0.56        18
```

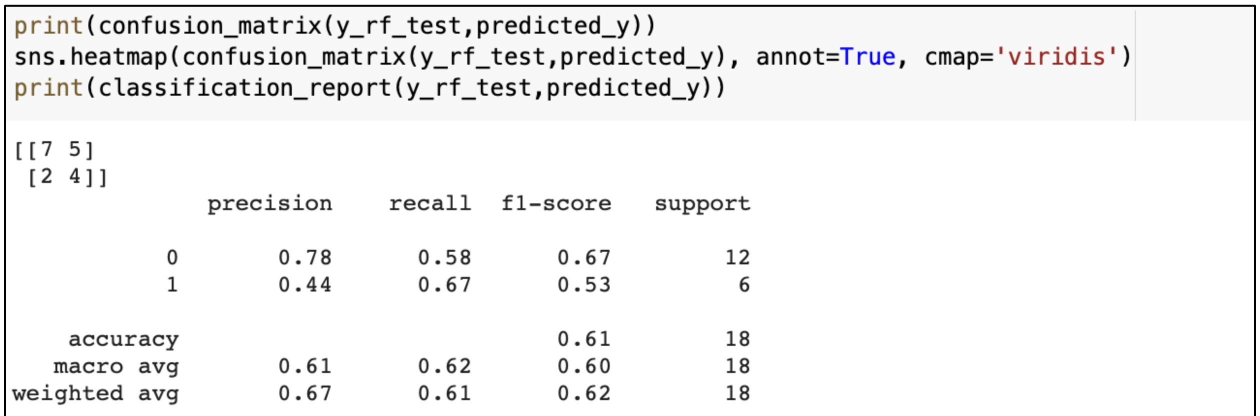


b) XGBoosting



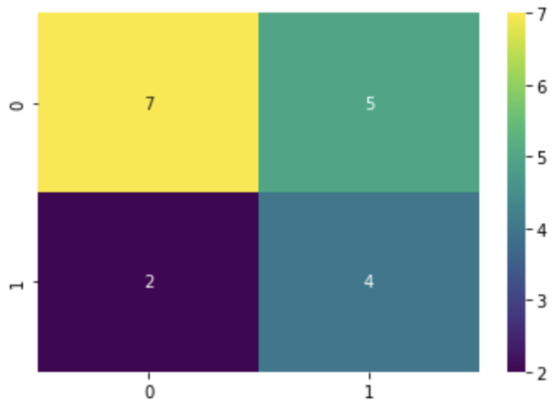
```
Cross Validation Scores:  [0.500 0.375 0.250 0.250 0.250 0.750 0.500 0.500 0.250]
Average CV Score:  0.40277777777777778
Number of CV Scores used in Average:  9
```

The average CV score obtained for XGBoosting is 0.403 with a range of 0.25-0.75. From the test dataset implementation, we can infer that the accuracy is 0.61 like the one we obtained for XGBoosting in the combined dataset and the independent class accuracies are also similar and close to the overall test accuracy. Moreover, we can also note that the model tends to predict equals 1's and 0s and the confusion matrix is very similar to the combined dataset test data confusion matrix.

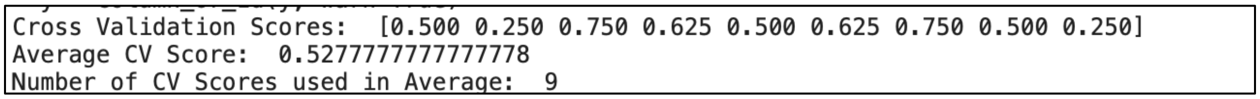


```
print(confusion_matrix(y_rf_test,predicted_y))
sns.heatmap(confusion_matrix(y_rf_test,predicted_y), annot=True, cmap='viridis')
print(classification_report(y_rf_test,predicted_y))
```

```
[[7 5]
 [2 4]]
              precision    recall  f1-score   support

           0       0.78      0.58      0.67        12
           1       0.44      0.67      0.53         6

    accuracy                           0.61        18
   macro avg       0.61      0.62      0.60        18
weighted avg       0.67      0.61      0.62        18
```



c) Linear Discriminant Analysis (LDA)

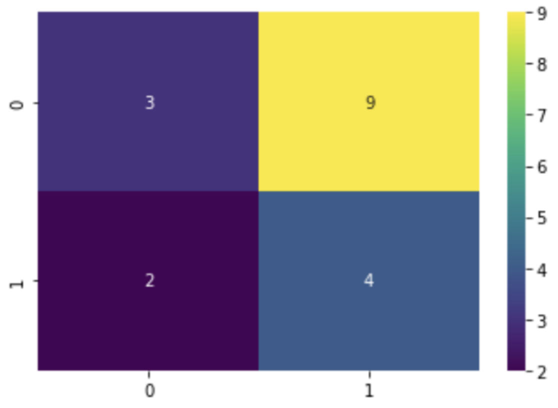


```
Cross Validation Scores:  [0.500 0.250 0.750 0.625 0.500 0.625 0.750 0.500 0.250]
Average CV Score:  0.5277777777777778
Number of CV Scores used in Average:  9
```

We can see from the above results that the average accuracy score is somewhat better than the random forest method and XGBoosting as we get a score of 0.528 with a range between 0.25 - 0.75 with 3 out of the 8 folds having a score of 0.5. We can note that still, the model is under it's for a few folds. When we see the test dataset, we can infer that the overall test accuracy is 0.39. Analogous to other models, we can infer that the model tends to predict more 1's than 0's as the class accuracies of class 0 and class 1 vary significantly. This may also be the reason for us getting a very high accuracy in some folds and very low accuracy in other folds.

```
print(confusion_matrix(y_rf_test,y_predicted_test))
sns.heatmap(confusion_matrix(y_rf_test,y_predicted_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_predicted_test))
```

```
[[3 9]
 [2 4]]
              precision    recall  f1-score   support

           0       0.60      0.25      0.35        12
           1       0.31      0.67      0.42         6

    accuracy                           0.39        18
   macro avg       0.45      0.46      0.39        18
weighted avg       0.50      0.39      0.38        18
```



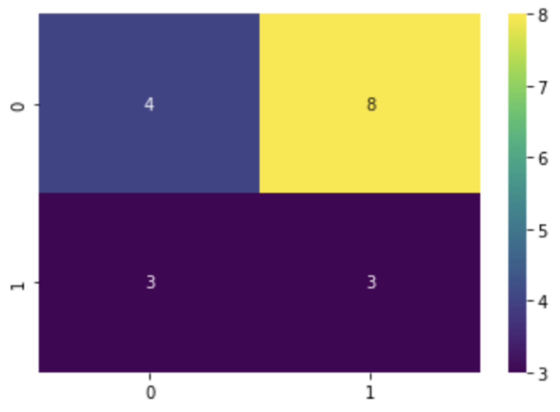
d) Quadratic Discriminant Analysis (QDA)

```
Cross Validation Scores:  [0.500 0.500 0.375 0.625 0.250 0.875 0.250 0.500 0.500]
Average CV Score:  0.4861111111111111
Number of CV Scores used in Average:  9
```

We can see from the above results that the average accuracy score is somewhat better than the random forest method and XGBoosting but not better than LDA as we get a score of 0.486 with a range between 0.25-0.875 with 4 out of the 8 folds having a score of 0.5. We can note that still, the model is under its few folds. When we see the test dataset, we can infer that the overall test accuracy which is 0.39 is equal to the one we obtained in the LDA method. Analogous to the LDA model, we can infer that the model tends to predict more 1’s than 0’s. However, the class accuracies do not vary that much since the model predicts 0’s when the true value is 1. This may also be the reason for us getting a very high accuracy in some folds and very low accuracy in other folds.

```
print(confusion_matrix(y_rf_test,y_predicted1_test))
sns.heatmap(confusion_matrix(y_rf_test,y_predicted1_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_predicted1_test))
```

```
[[4 8]
 [3 3]]
              precision    recall  f1-score   support

           0       0.57      0.33      0.42        12
           1       0.27      0.50      0.35         6

    accuracy                           0.39        18
   macro avg       0.42      0.42      0.39        18
weighted avg       0.47      0.39      0.40        18
```



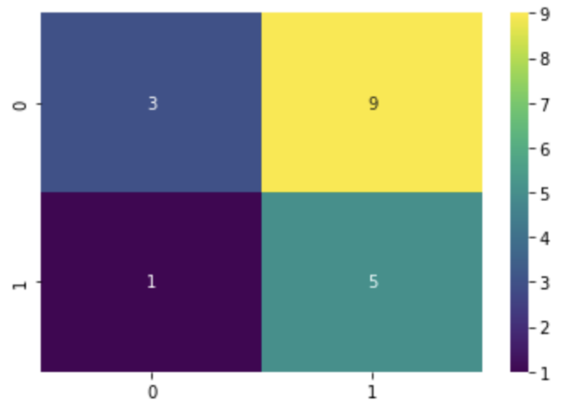
e) Logistic Regression

```
Cross Validation Scores:  [0.500 0.375 0.125 0.500 0.500 0.250 0.375 0.250 0.625]
Average CV Score:  0.3888888888888889
Number of CV Scores used in Average:  9
```

The Logistic regression model has the lowest average accuracy for k-fold stratified cross-validation when compared to the other models in the printing dataset which is 0.38 and has a range of 0.125-0.625 with 4 of the 8 folds having a score lesser than 0.5. The model tends to have a large variation in the class accuracies for model testing as class 0 has an accuracy of 0.25 whereas for class 1 it is 0.83 having an imbalance in the number of 1 and 0's predicted. Also, we can infer that there is a big difference in the tendency of the model to predict more 1's than 0's with 14 of the 18 predicted values being 1. This may be the reason for the accuracy of the training dataset for some folds being very less as the model is not learning the relationship between x and y properly.

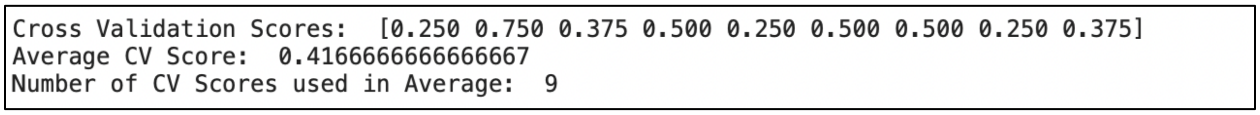
```
print(confusion_matrix(y_rf_test,y_test_pred))
sns.heatmap(confusion_matrix(y_rf_test,y_test_pred), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_test_pred))
```

```
[[3 9]
 [1 5]]
              precision    recall  f1-score   support

           0       0.75      0.25      0.38        12
           1       0.36      0.83      0.50         6

    accuracy                           0.44        18
   macro avg       0.55      0.54      0.44        18
weighted avg       0.62      0.44      0.42        18
```



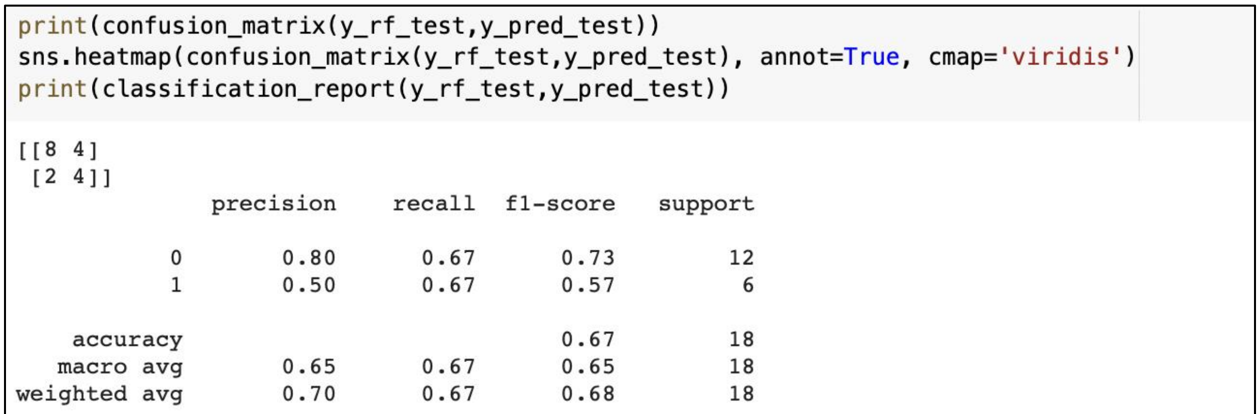
3. MILLING DATASET

a) Random Forest

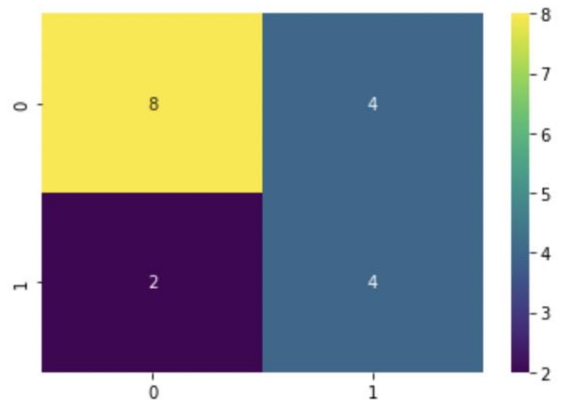


```
Cross Validation Scores:  [0.250 0.750 0.375 0.500 0.250 0.500 0.500 0.250 0.375]
Average CV Score:  0.4166666666666667
Number of CV Scores used in Average:  9
```

The average CV score obtained for the random forest is 0.416 for the milling dataset which is less than the accuracy we obtained for both the combined dataset and the printing dataset. The accuracy of the CV has a range of 0.25-0.75. From the test dataset implementation, we can infer that the overall accuracy of this method is 0.67 which is better than the scores obtained with the combined and printing datasets. The independent class accuracies are equal unlike the ones obtained with the other two datasets. But the noteworthy aspect is that the model tends to predict more 0s than 1’s which may be the reason for having a higher overall test accuracy.



```
print(confusion_matrix(y_rf_test,y_pred_test))
sns.heatmap(confusion_matrix(y_rf_test,y_pred_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred_test))

[[8 4]
 [2 4]]
              precision    recall  f1-score   support

           0       0.80      0.67      0.73        12
           1       0.50      0.67      0.57         6

    accuracy                           0.67        18
   macro avg       0.65      0.67      0.65        18
weighted avg       0.70      0.67      0.68        18
```



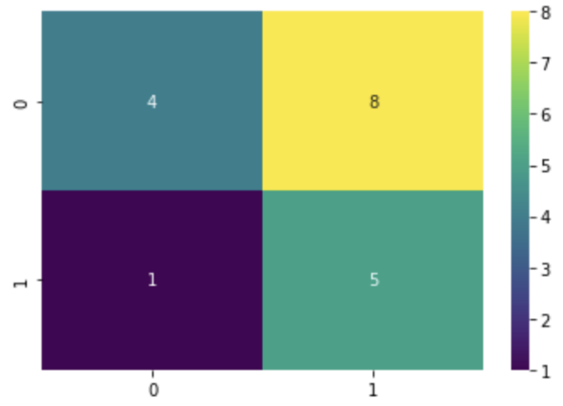
b) XGBoosting

```
Cross Validation Scores:  [0.500 0.375 0.375 0.375 0.500 0.375 0.250 0.375 0.375]
Average CV Score:  0.3888888888888889
Number of CV Scores used in Average:  9
```

The average CV score obtained for XGBoosting is 0.388 with a range of 0.25-0.5 which has a CV score of 0.375 for 6 folds out of 9 folds. From the test dataset implementation, we can infer that the overall accuracy of the model is 0.50 which is lower than the accuracy obtained with random forest. Due to an imbalance in the number of 1 and 0's predicted, the model has a large variation in class accuracies for model testing, with class 0 having an accuracy of 0.33 and class 1 having an accuracy of 0.83. The model tends to predict more 1s than 0’s, unlike the random forest method which may be due to the training dataset having more 1s than 0's.

```
print(confusion_matrix(y_rf_test,predicted_y))
sns.heatmap(confusion_matrix(y_rf_test,predicted_y), annot=True, cmap='viridis')
print(classification_report(y_rf_test,predicted_y))
```

```
[[4 8]
 [1 5]]
              precision    recall  f1-score   support

           0       0.80      0.33      0.47        12
           1       0.38      0.83      0.53         6

    accuracy                           0.50        18
   macro avg       0.59      0.58      0.50        18
weighted avg       0.66      0.50      0.49        18
```



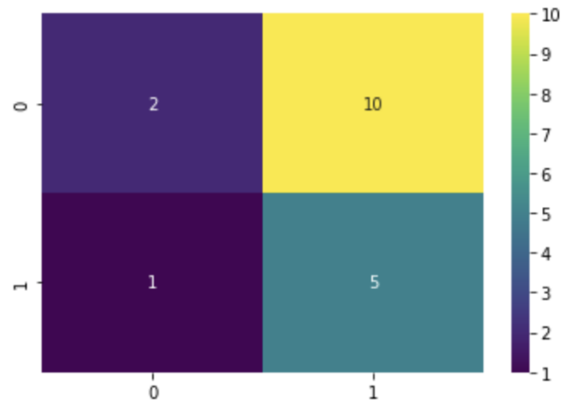
c) Linear Discriminant Analysis (LDA)

```
Cross Validation Scores:  [0.750 0.625 0.500 0.625 0.500 0.500 0.500 0.750 0.500]
Average CV Score:  0.5833333333333334
Number of CV Scores used in Average:  9
```

We can see from the above results that the average accuracy score of CV is better than any other method executed with the milling dataset as we get a score of 0.583 with a range between 0.50-0.75. 5 out of the 9 folds have a score of 0.5. We can note that still the model overfits for a few folds. When we observe the test dataset, we can infer that the overall test accuracy is 0.39. Analogous to the XGBoosting model, the LDA model tends to predict more 1's than 0's as the class accuracies of class 0 and class 1 vary significantly. But unlike the XGBoosting model, the ratio of numbers 1's predicted to the number of 0's predicted is much larger in the LDA model. This may also be the reason for us getting a very high accuracy in some folds and very low accuracy in other folds in the training dataset.

```
print(confusion_matrix(y_rf_test,y_predicted1_test))
sns.heatmap(confusion_matrix(y_rf_test,y_predicted1_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_predicted1_test))
```

```
[[ 2 10]
 [ 1  5]]
              precision    recall  f1-score   support

           0       0.67      0.17      0.27        12
           1       0.33      0.83      0.48         6

    accuracy                           0.39        18
   macro avg       0.50      0.50      0.37        18
weighted avg       0.56      0.39      0.34        18
```



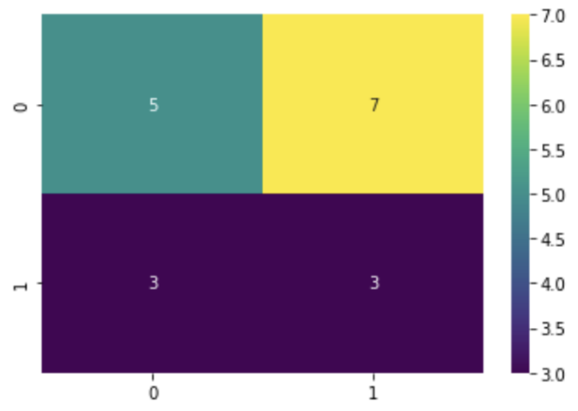
d) Quadratic Discriminant Analysis (QDA)

```
Cross Validation Scores:  [0.500 0.375 0.500 0.500 0.625 0.375 0.500 0.250 0.375]
Average CV Score:  0.4444444444444444
Number of CV Scores used in Average:  9
```

We can see from the above results that the average accuracy score is somewhat better than the random forest method and XGBoosting but not better than LDA as we get a score of 0.44 with a range between 0.25-0.625. We can note that still, the model is under it for a few folds. When we observe the test dataset, we can infer that the overall test accuracy which is also 0.44 is the same as the one obtained for logistic regression. Analogous to the LDA model and XGBoosting model, we can infer that the model tends to predict an equal number of 1's and 0's. As a result, the class accuracies do not vary that much.

```
print(confusion_matrix(y_rf_test,y_predicted1_test))
sns.heatmap(confusion_matrix(y_rf_test,y_predicted1_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_predicted1_test))
```

```
[[5 7]
 [3 3]]
              precision    recall  f1-score   support

           0       0.62      0.42      0.50        12
           1       0.30      0.50      0.37         6

    accuracy                           0.44        18
   macro avg       0.46      0.46      0.44        18
weighted avg       0.52      0.44      0.46        18
```



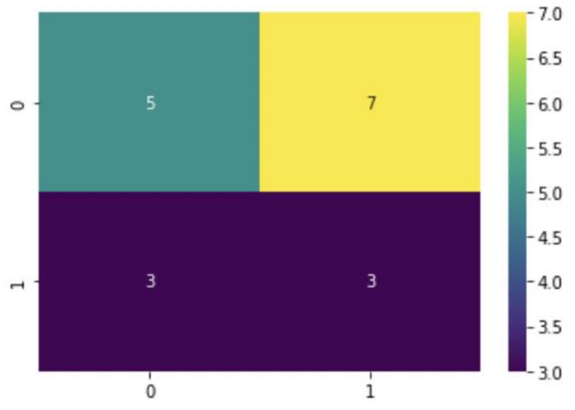
e) Logistic Regression

```
Cross Validation Scores:  [0.500 0.375 0.125 0.500 0.500 0.250 0.375 0.250 0.625]
Average CV Score:  0.3888888888888889
Number of CV Scores used in Average:  9
```

The Logistic regression model has the same average accuracy for k-fold stratified cross-validation as the XGBoosting model in the milling dataset which is 0.38 and has a range of 0.125-0.625 with 5 of the 9. So, we can note that the model is under its few folds. From the test dataset results, we can infer that the overall test accuracy is 0.44 which is like the accuracies obtained with the combined dataset and printing dataset for the logistic regression model. The model does not have a large variation in the class accuracies for model testing as class 0 has an accuracy of 0.42 whereas for class 1 it is 0.50.

```
print(confusion_matrix(y_rf_test,y_pred))
sns.heatmap(confusion_matrix(y_rf_test,y_pred), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred))
```

```
[[5 7]
 [3 3]]
              precision    recall  f1-score   support

           0       0.62      0.42      0.50        12
           1       0.30      0.50      0.37         6

    accuracy                           0.44        18
   macro avg       0.46      0.46      0.44        18
weighted avg       0.52      0.44      0.46        18
```



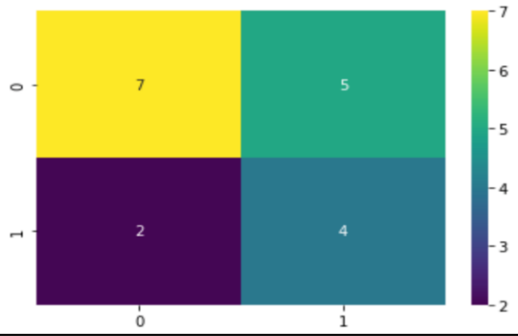
COMPARISON

For each dataset i.e., combined, printing and milling, we will consider those methods which have the highest overall test accuracy.

1. In ***Combined Dataset***, XGBoosting has the highest overall test accuracy of 0.61.

```
print(confusion_matrix(y_rf_test,y_pred_test))
sns.heatmap(confusion_matrix(y_rf_test,y_pred_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred_test))
```

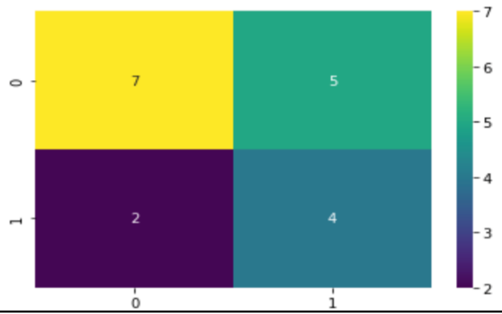
```
[[7 5]
 [2 4]]
              precision    recall  f1-score   support

           0       0.78      0.58      0.67        12
           1       0.44      0.67      0.53         6

    accuracy                           0.61        18
   macro avg       0.61      0.62      0.60        18
weighted avg       0.67      0.61      0.62        18
```



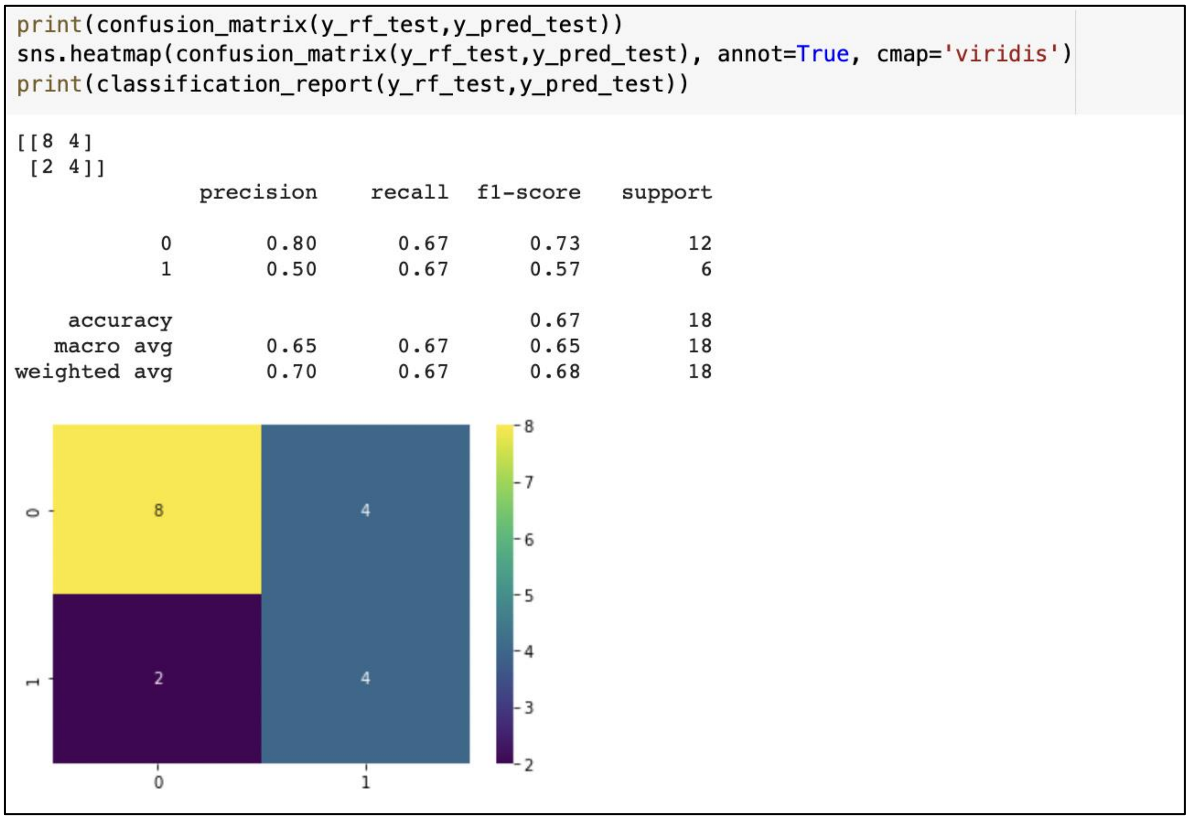
2. For ***Printing Dataset,*** XGBoosting again produced better results when compared to all the other methods and its overall test accuracy is the same as the combined dataset with an overall test accuracy of 0.61.

```
print(confusion_matrix(y_rf_test,predicted_y))
sns.heatmap(confusion_matrix(y_rf_test,predicted_y), annot=True, cmap='viridis')
print(classification_report(y_rf_test,predicted_y))
```

```
[[7 5]
 [2 4]]
              precision    recall  f1-score   support

           0       0.78      0.58      0.67        12
           1       0.44      0.67      0.53         6

    accuracy                           0.61        18
   macro avg       0.61      0.62      0.60        18
weighted avg       0.67      0.61      0.62        18
```



3. For ***Milling Dataset*** Random Forest produced better results when compared to all the other methods with an overall test accuracy of 0.67.



```
print(confusion_matrix(y_rf_test,y_pred_test))
sns.heatmap(confusion_matrix(y_rf_test,y_pred_test), annot=True, cmap='viridis')
print(classification_report(y_rf_test,y_pred_test))
```

```
[[8 4]
 [2 4]]
              precision    recall  f1-score   support

           0       0.80      0.67      0.73        12
           1       0.50      0.67      0.57         6

    accuracy                           0.67        18
   macro avg       0.65      0.67      0.65        18
weighted avg       0.70      0.67      0.68        18
```

When we compare all the results with each other we can see that random forest for the milling dataset is better than the other two methods not only because it has the highest overall test accuracy but because it also has an equal class accuracy which is important for an imbalanced classification problem.

Moreover, theoretically speaking, we cannot just predict the porosity of the part just after the end of the printing process as we do not have a set of output values after the end of the printing process. This output is taken only after the end of the milling process. So, predicting the output only after the end of the milling process seems logical. The issue with the combined dataset is that sometimes the model may overfit when we use both printing and milling datasets to predict the output. That is the reason we think that using the milling dataset in a random forest model may give better results.

EXECUTIVE SUMMARY

Machine Learning (ML) is defined as a computer programming method to optimize the performance criterion using past data. For ML in Additive Manufacturing (AM), is extensive research being done where they are trying to integrate ML and Artificial Intelligent into AM. Machine Learning algorithms and methods are used in the AM process to improve the quality of the product optimize the manufacturing process and reduce manufacturing costs. One of the major challenges of AM processes is that porosity obtained which may make the product unusable in certain applications. With the help of the given dataset, we were able to construct a model which will predict the porosity of the product after the end of the milling process with the help of the input parameters taken from the sensors in the machine. With the help of the model we built, anyone will be able to tell concerning a particular parameter and its corresponding values in real-time whether the porosity will be high or low with a certain accuracy. Although this accuracy may vary for different data points, we can tell for certain values with high accuracy that the porosity will be high. This classification of high and low accuracy may change concerning the base material, processes used, and application of the product which we can change in the model accordingly. So, we can fix a sensor in the machine which senses during real-time when the model predicts with high accuracy that the porosity will be high at a certain value which can be used for a human intervention to stop the process and correct the error. With regards to the objective, we have set for the problem, we were able to conclude that the random forest model with the help of the milling dataset has high accuracy when compared to the rest of the datasets and ML methods.

CONCLUSION

From all the above interpretations and results, we can predict the level of porosity in the surface being machined, and thus we can declare a surface as porous or non-porous during the machining process.

For future work, we see this as a continuous process of data generation and data interpretation by which we could be able to predict the parts that are either going to be printed or machined will have a porosity or not. The level and intensity of porosity which is going to be classified as high will depend on various parameters. This Machine Learning model can also be used to predict some other features like porosity, vortex, discoloration, voids, cracks, etc. For this to function, a small change in the model must be performed and executed.

This Machine Learning model can be paired up with Digital Twin to predict the compliance of the machine being used. Digital Twin is an emerging technology used to digitally replicate an entire process in manufacturing and to determine the breaking point of any machine or process. With the help of ML and AI, digital twin can be applied to real-time scenarios.

REFERENCE

1. Liang, H., Sun, X., Sun, Y. et al. Text feature extraction based on deep learning: a review. J Wireless Com Network 2017, 211 (2017). https://doi.org/10.1186/s13638-017-0993-1
2. Dr. Pablo Rivas, Deep Learning for Beginners: A Beginner's Guide to Getting Up and Running with Deep Learning from Scratch Using Python, Published by Packt Publishing, (2020) ISBN 1838640851, 9781838640859.
3. Some Generalized data on Hybrid additive Machines has been taken from https://smtamu.wixsite.com/about
4. An Introduction to Statistical Learning with Applications in R by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, ISSN 1431-875X
5. Meng, L., McWilliams, B., Jarosinski, W. et al. Machine Learning in Additive Manufacturing: A Review. JOM 72, 2363–2377 (2020). https://doi.org/10.1007/s11837-020-04155-y
6. Jian Qin, Fu Hu, Ying Liu, Paul Witherell, Charlie C.L. Wang, David W. Rosen, Timothy W. Simpson, Yan Lu, Qian Tang, Research and application of machine learning for additive manufacturing, Additive Manufacturing, Volume 52, 2022, 102691, ISSN 2214-8604, https://doi.org/10.1016/j.addma.2022.102691. (https://www.sciencedirect.com/science/article/pii/S2214860422000963)
7. Xinbo Qi, Guofeng Chen, Yong Li, Xuan Cheng, Changpeng Li, Applying Neural-Network-Based Machine Learning to Additive Manufacturing: Current Applications, Challenges, and Future Perspectives, Engineering, Volume 5, Issue 4, 2019, Pages 721-729, ISSN 2095-8099, https://doi.org/10.1016/j.eng.2019.04.012. (https://www.sciencedirect.com/science/article/pii/S2095809918307732)