International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Comparative Analysis of 3D Printing Support Structure Prediction Using Feature Selection Methods for Classification Algorithms

Sonali Patil[1], Yogesh Deshpande[2], Dattatraya Parle[3]

Submitted: 27/01/2024 **Revised**: 05/03/2024 **Accepted**: 13/03/2024

Abstract: The success of machine learning models heavily relies on the quality and diversity of the datasets used for training and evaluation. This paper discusses the process of dataset generation and building machine learning models. Support structures are essential for accurate 3D printing of complex geometries, preventing sagging and deformation, and ensuring stable, high-quality prints with careful adjustment of print settings. In this work, different Machine Learning techniques are used and evaluated based on their performance of classifying the need for support during the 3D printing process. The ultimate objective is to classify the model in terms of 'need support Y/N'.In pursuing this objective, different Machine Learning techniques are utilized to classify different CAD models. The different machine-learning classification techniques applied in this work are Logistic Regression, Support Vector Machine, Random Forest, K-Nearest Neighbours, Decision Tree, and Gradient Boosting. The comparative study based on 6 different performance measures suggests that the Random Forest algorithm works with an accuracy of 0.97 well for classifying the need for support into categories based on the values provided for the process parameters. Finally, SHAP& LIME analysis shows the significance of each feature in the prediction of the need for support. This study can be extended for independent variables including curvature/taper in the build direction and dependent variables as type of structure and type of build adhesion which may be a powerful tool to predict the mechanical properties better.

Keywords: *3D printing, Machine learning algorithm, Classification model, Support structure, Dataset, Prediction*

1. Introduction

1.1 Background

The increasing complexity of real-world problems necessitates advanced machine learning models. However, the efficacy of these models is intrinsically tied to the quality and representativeness of the datasets they are trained on. This paper aims to interpret the significance of dataset generation in the context of building robust machine learning models.

Support structures in 3D printing (3Dp) depend on the geometry of the object and the printer's capabilities[1]. Factors to consider include the overhang angle, geometry complexity, material and printer characteristics, layer adhesion, print orientation, printing software features, material considerations, print speed and temperature, post-processing considerations, and test prints[2]. Overhangs can be handled by most 3D printers, while complex geometries may require support to maintain print accuracy[3]. Materials and printers have varying capabilities, and support structures help maintain layer adhesion[4]. Print orientation can also minimize the need for supports[1]. Print speed and temperature can also influence the need for supports. Post-processing considerations involve determining how easy it will be to remove support structures after printing[5]. Test prints can help identify potential issues and refine settings before committing to a full-scale print[6].

A labeled dataset of 96CAD models with 5 different orientations used to test the binary classification approaches of Logistic Regression, Support Vector Machine, Random Forest,K-Nearest Neighbours, Decision Tree, and Gradient Boosting. Our objective is to construct binary classifiers that can distinguish between the printing model's need support or not. We evaluate how well these methods work to identify support requirements for the printing portion.

2. Dataset Generation

2.1 Data Collection Techniques

In 3D printing, various types of datasets are generated, depending on the nature of the printing process[7], the sensors or monitoring systems in use[8], and the desired information[2]. Here are several types of datasets commonly associated with 3D printing such as toolpath, deposition rate, and cooling time[9] , Temperature measurements at different points on the build plate, nozzle, or inside the print chamber during the printing process[10], Data on the movement speed and acceleration of the 3D printer's print head during the printing process[11] , Records of any detected defects, errors, or anomalies during the printing process [8]and Information about the

[1,2]*Department of Computer Engineering, Vishwakarma University,Pune, India;*
[3]*Nuclear Advanced Manufacturing Research Centre, Sheffield, United Kingdom*
** Corresponding Author Email: sonali.patil-960@vupune.ac.in*

printed object's geometry, dimensions, and accuracy[12][13]. And also the Point cloud data sets[14]. Some work done to extract the features by parsing the sliced STL file [15]. But none of the dataset can be utilized to predict the need of structure and build adhesion in prior.

In this study the dataset is generated using small alphabets, capital alphabets and numbers. The wall-based technique[16] used in this study helps in understanding the overall structure of the 3D print and provides insights into how the geometry is built up layer by layer. This method plays a pivotal role in adapting 3Dp settings to varying orientations of the alphabet, as illustrated in Fig.1a. It is essential to recognize that a block in a specific orientation cannot be successfully printed using the same 3Dp settings as another orientation. The unique characteristics of each orientation necessitate tailored settings for a successful 3Dp process. Fig.1b & Fig.1c exemplifies how the wall-based technique captures these distinctions.

A notable distinction in the wall-based technique lies in the number of walls, denoted as Wall 1 Wall 2 etc and its printing behavior. This distinction is critical as it directly influences the structural integrity and complexity of the printed object. The need for support structures and build plate adhesion settings further emphasizes the intricate adjustments demanded by the different build orientations.

The input feature vector undergoes substantial changes to accommodate the distinctions of different orientations. As the build orientation shifts to different degrees, additional considerations become pivotal for a successful 3D print. The introduction of the following features reflects the adaptability of the wall-based technique:

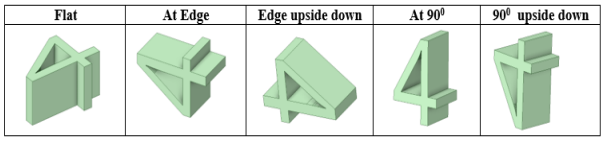
‘dX’,’dY’,’dZ’,'now','Fwall','nowp','Farea','offset','maxoffset_bed','Wanother','maxoffset_anotherW','Wair','Wair_partial','max_support'

1. Bounding box X, bounding box Y, Bounding box Z (dX','dY','dZ'):- These parameters define the dimensions of the bounding box in the X, Y, and Z directions, respectively. They represent the maximum space available for 3D printing within the specified limits.
2. Number of walls (‘now’): - This refers to the count of walls being printed in the 3D model. Walls are typically structures that enclose the printed object
3. Is the first wall printing on bed? How many walls are printing on the bed? ('Fwall','nowp') - Indicates whether the first wall is in contact with the print bed and how many total walls are printing on the bed. The "bed" is the surface where the 3D printing process starts.
4. Minimum area of the first layer('Farea'):- This represents the smallest surface area of the initial layer being printed. It's a critical parameter for ensuring proper adhesion to the print bed.
5. Is there an offset in the first layer and last layer of wall printing on the bed? (offset') :- This question is asking if there is a positional difference (offset) between the first and last layers of the walls printed on the bed.
6. What is the maximum offset between the first layer and the last layer of walls printing on the bed? (’’maxoffset_bed',):-

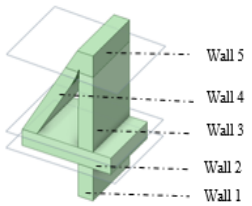
   If there is an offset, this specifies the maximum allowed distance between the first and last layers of walls printed on the bed.
7. How many walls are printing on another wall? ('Wanother')

   Refers to the number of walls that are being printed on top of or adjacent to other walls.
8. What is the maximum offset between the first layer and the last layer of walls printing on the other wall? ('maxoffset_anotherW'):-Similar to question 6 but specifically for walls printed on other walls.
9. How many walls are fully printing in the air? ('Wair') :- Indicates the number of walls that do not have any support from the print bed or other structures during the printing process.
10. How many walls are partially printing in the air? (Wair_partial') :-Refers to the number of walls that have some support but are not fully connected to the print bed or other structures.
11. What is the maximum unsupported area of the first layer of the wall printing in the air? ('max_support'):- Specifies the largest area of the first layer of walls that is not supported by the print bed or other structures.

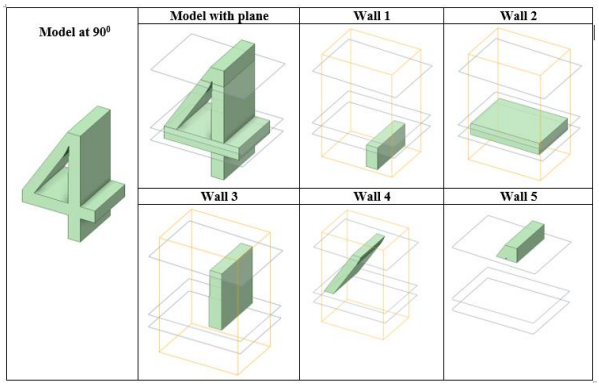
In summary, these parameters are essential for optimizing the 3D printing process, ensuring proper adhesion, and understanding the required support structural characteristics of the printed object. In the context of the wall-based technique, the specific characteristics of walls, with straight edges but tapering in the z-direction, introduce additional complexity. Notably, the sides of the start and end layers differ due to this tapering, underscoring the need for detailed consideration in the 3Dp settings. As the complexity of the printed part increases, the dataset adapts by incorporating additional wall features. This adaptive nature of the wall-based technique is a testament to its versatility in accommodating diverse geometries and orientations in 3Dp, providing a robust framework for capturing the intricacies of the layer-wise printing process.



(a)



(b)



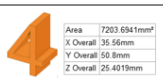
(c)

Fig.1: Wall based technique- (a) Rotation and Flip of the 3D model (b) No. of walls generated (c) Visualization of walls.

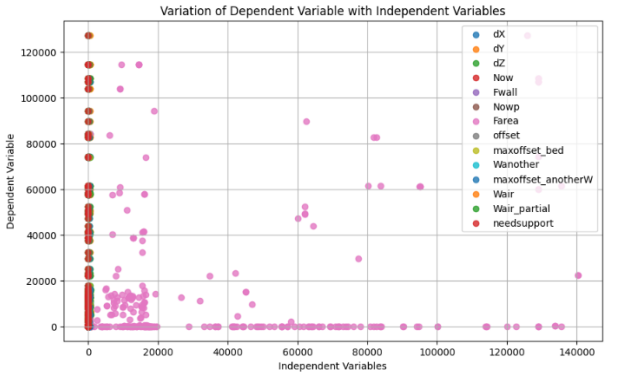
Understanding the variation of dependent and independent variables is crucial for feature selection, model building, and interpretation of results in machine learning tasks[17], [18]. Proper preprocessing, handling of missing values, normalization, and encoding categorical variables are some of the steps involved in managing the variation of variables effectively for model training and evaluation[19].

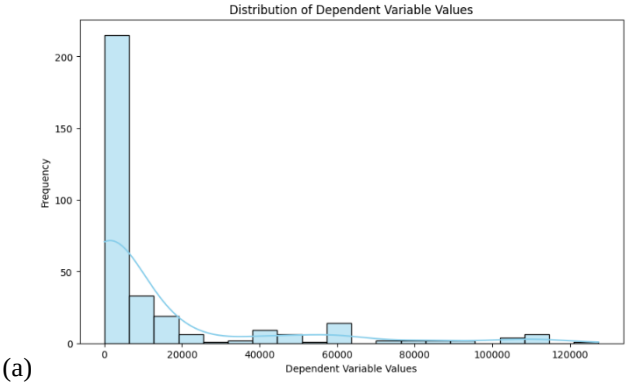
Below Table 1 shows sample dataset generated for the digit '4' at 90^0.

Table 1: Sample dataset generated for the model at 90^0

model	Bounding box X Bounding box y Bounding box Z		No. of walls	Is first wall printing on bed?	how many walls printing on the bed	Minimum area of first layer
						
Is there a offset in first layer and last layer of wall printing on the bed?	What is the maximum offset between first layer and last layer of walls printing on the bed	How many walls printing on another wall?	What is the maximum offset between first layer and last layer of walls printing on the another wall	How many walls fully printing in the air?	How many walls partially printing in the air?	What is the max unsupported area of the first layer of the wall printing in the air
Yes	15mm	4	17mm	0	1	2641

In summary, the variation of dependent and independent variables in experiments involves careful manipulation and observation of factors to understand their relationship. Fig.2 shows the variation of the distribution of dependent variable 'need support' values provides insights into the patterns and characteristics of the outcomes.





(a)

(b)

Fig. 2. Shows (a) the variation of the dependent variables and the independent variable for the experiments and (b) distribution of dependent variables values.

The describe () method in pandas generates descriptive statistics that summarize the numerical attributes (columns) of a DataFrame. Table 2 statistics provide valuable insights into the distribution, central tendency, and spread of the numerical data within each column of the dataset generated.

Table 2. Average values, standard deviation, standard error of dependent variables and the independent variable for the experiments

	dX	dY	dZ	now	Fwall	nowp	Farea	offset	maxoffset_bed	Wanother	maxoffset_another W	Wair	Wair_partial	max_support	needsupport
count	480	480	480	480	480	480	480	480	480	480	480	480	480	480	480
mean	206.3	253.9	250.4	3.7	1.0	1.2	21608.2	0.8	104.7	2.3	92.0	0.3	0.5	9160.4	0.6
std	132.1	177.1	172.3	2.6	0.0	0.4	33122.3	0.4	99.0	2.3	121.1	0.5	0.7	22576.7	0.5
min	10.0	10.0	10.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	100.0	100.0	100.0	1.0	1.0	1.0	796.3	1.0	48.5	0.0	0.0	0.0	0.0	0.0	0.0
50%	254.0	254.0	254.0	3.0	1.0	1.0	7867.6	1.0	65.0	2.0	40.0	0.0	0.0	1.2	1.0
75%	304.8	381.0	381.0	5.0	1.0	1.0	17599.3	1.0	125.9	3.0	154.0	0.0	1.0	6000.0	1.0
max	610.0	688.0	686.0	15.0	1.0	3.0	140244.9	1.0	572.0	12.0	572.0	3.0	4.0	127387.0	1.0

2.2 Data Augmentation

Augmenting datasets through techniques such as rotation, and flipping to increase the diversity of the training data without collecting new samples. Data augmentation in the context of 3Dp involves creating variations of the existing dataset to enhance the robustness and diversity of the training data for a machine learning model. In the case described, where the dataset captures different orientations

and characteristics of 3D printed objects, data augmentation techniques can be applied to generate additional training examples. Here's how data augmentation worked for the given case:

1. Rotation:
   - Original Data: A model with a specific orientation (e.g., flat, at an edge).
   - Data Augmentation: Create variations by rotating the CAD model at different angles (e.g., 90 degrees, 180 degrees, etc.).
   - Impact: This helps the model generalize better to objects printed at various orientations.
2. Flipping:
   - Original Data: A model with specific characteristics.
   - Data Augmentation: Generate variations by flipping the CAD model horizontally and/or vertically.
   - Impact: Mimics different scenarios where the object might be flipped during printing, providing a more comprehensive training set.

2.3 Exploratory Data Analysis:

It refers to the critical process of performing initial investigations on data to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. Analysis done to understand the data first and try to gather as many insights as possible from it.

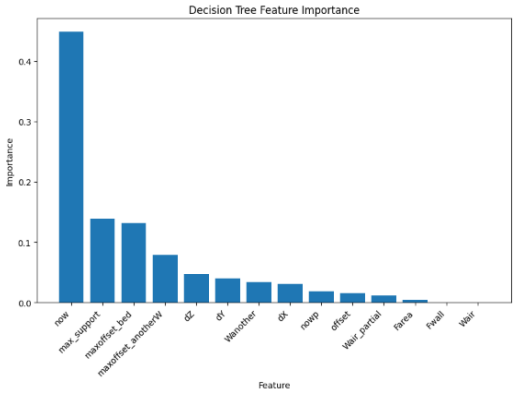
- Dataset comprises 480 observations and 14 characteristics.
- Out of which 13are independent variables and the rest 1aredependent variable.
- Data has only float and integer values.
- No variable column has null/missing values.

2.4 Explaining Algorithm:

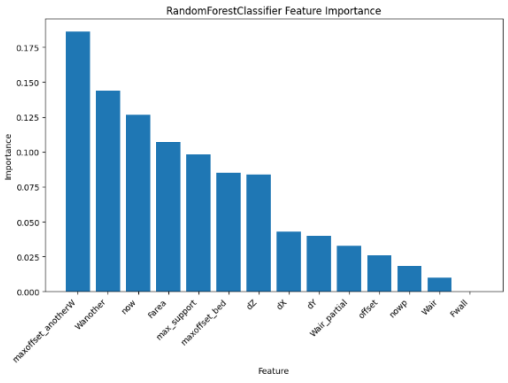
Feature Selection is the process used to select the input variables that are most important to Machine Learning task[20].

2.4.1 Feature importance

Feature importance can be extracted directly from Decision Trees and Random Forest Classifier[21][22], but it's not a standard concept for Naive Bayes and KNeighbors Classifier[18],. For Naive Bayes, the algorithm assumes that all features are conditionally independent, so there isn't a concept of feature importance in the same way as in decision tree-based models. Fig.3 (a) & (b) shows how feature importance extracted from Decision Trees and Random Forest Classifier.



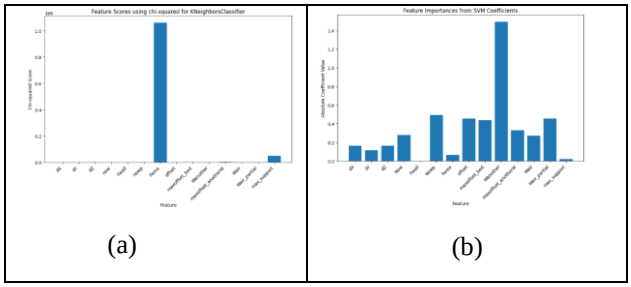
(a)



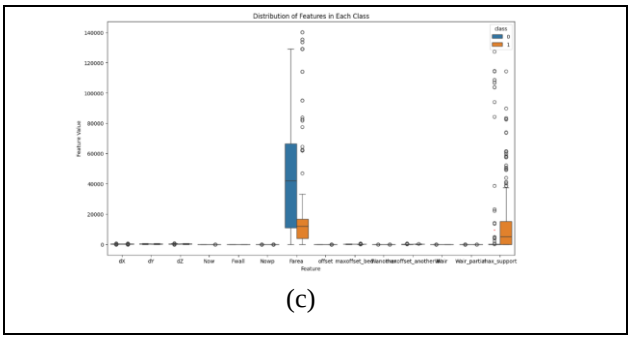
(b)

Fig. 3. Feature importance: (a) Decision Tree (b) Random Forest Classifier

For KNeighbors Classifier, there is no inherent feature importance as it doesn't build a model with coefficients for each feature. For KNeighbors Classifier, a common approach is to use univariate feature selection methods like Select KBest with statistical tests such as chi-squared, ANOVA, or mutual information [23]. The plot shown in Fig.4(a) is the chi-squared scores for each feature, and 'Farea' can be consider as a feature with higher scores as more important.



(a) (b)



(c)

Fig. 4. Feature importance: (a) KNeighbors Classifier (b) Support Vector Machines (c) Gaussian Naive Bayes

Feature importance interpretation with Support Vector Machines (SVMs) is not as straightforward as with some other models like decision trees or random forests. SVMs use a hyperplane to separate classes, and feature importance is derived from the coefficients of the hyperplane or support vectors. In summary, while SVMs offer powerful classification capabilities, interpreting feature importance requires careful consideration of the model's coefficients, support vectors, and feature selection techniques, balancing model complexity and interpretability. Fig.4 (b) shows "Wanother" feature as an important feature in building the model using SVM.

As mentioned earlier, Naive Bayes classifiers, specifically Gaussian Naive Bayes, do not provide a direct measure of feature importance like some other algorithms. Unlike some other algorithms, the GaussianNB class in scikit-learn does not directly expose the mean and variance values for each feature in each class after training. For Gaussian Naive Bayes, the model assumes that the features follow a Gaussian (normal) distribution within each class, and it estimates mean and variance parameters for each feature in each class during training. However, these parameters are not directly accessible from the GaussianNB instance in scikit-learn [24]. Fig.4(c) shows a descriptive statistic to explore the means and variances for each feature in each class. By Understanding and leveraging Fig.3 & Fig.4 statistics can lead to more effective and interpretable classification models. Fig.5 shows the means and variances of most important features like 'Farea','now','maxoffset_bed','Wanother','maxoffset_anotherW','Wair'within each class

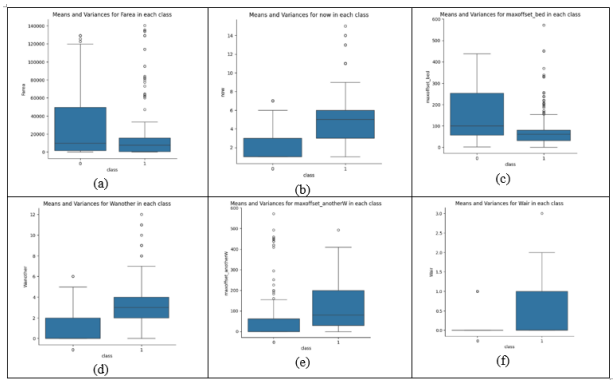


Fig. 5: Means and variances of most important features (a)Farea (b)now (c)maxoffset_bed (d)Wanother (e)maxoffset_anotherW (f)Wair

In summary, means and variances of features within each class play a pivotal role in various aspects of classification tasks, including feature selection, model building, interpretation, and ensuring the robustness of machine learning models.

2.4.2 Machine Learning Interpretability (SHAP & LIME)

SHAP (SHapley Additive exPlanations) values, based on game theory, provide a consistent and objective explanation of how each feature impacts a model's prediction[25]. Positive SHAP values enhance prediction, while negative values have a negative effect, with magnitude indicating the strength of the effect[26]. This section will calculate SHAP values, display feature importance, dependence, force, and decision plots. The model in has shown better performance for "1" label than "0" due to an unbalanced dataset. Overall, it is an acceptable result with 92% accuracy.

	precision	recall	f1-score	support
0	0.90	0.93	0.92	29
1	0.94	0.91	0.92	33
accuracy			0.92	62
macro avg	0.92	0.92	0.92	62
weighted avg	0.92	0.92	0.92	62

The model explainer will be created using a random forest classification model and SHAP value will be calculated using a testing set. In Fig.6(a) summary plot display the summary_plot using SHAP values and testing set.

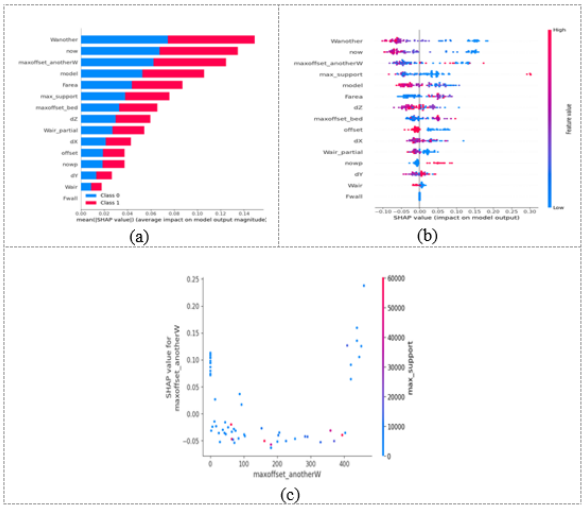


Fig. 6. SHAP analysis: (a) Summary plot (b) SHAP value (c) Effect of maxoffset_anotherW

The summary plot shows the feature importance of each feature in the model. The results show that 'Wanother','maxoffset_anotherW' and 'now' play major roles in determining the results. Y-axis indicates the feature names in order of importance from top to bottom. X-axis represents the SHAP value, which indicates the degree of change in log odds. The color of each point on the graph represents the value of the corresponding feature, with red indicating high values and blue indicating low values. Each point represents a row of data from the original dataset. The feature "'Wanother', shows that it is mostly high with a negative SHAP value. It means higher 'No. of walls printing on another wall' counts tend to negatively affect the output.

shap.dependence_plot ("maxoffset_anotherW", shap_values [0],test,interaction_index="max_support")

A dependence plot is a type of scatter plot that displays how a model's predictions are affected by a specific feature (maxoffset_anotherW). On average, maxoffset_anotherW have a mostly positive effect on the model.

Force Plot

Force Plots provide a clear and intuitive visualization of how features influence individual predictions, enabling better understanding and trust in machine learning models. They help identify which features are driving predictions and why, which is crucial for model transparency and accountability.We will examine the first sample in the testing set to determine which features contributed to the "0" result. To do this, we will utilize a force plot and provide the expected value, SHAP value, and testing sample.

shap.plots.force(explainer.expected_value[0], shap_values[0][0,:], X_test.iloc[0, :], matplotlib = True)

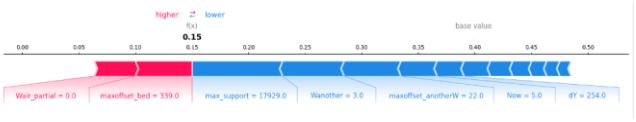


Fig. 7: Force Plot label "0"

We can clearly see that 2 no.of walls printing on the bed and 1 wall printing on another wall have contributed to negative to need of support.Let's look at need of support churn samples with label "1".

shap.plots.force(explainer.expected_value[1], shap_values[1][6, :], X_test.iloc[6, :],matplotlib = True)

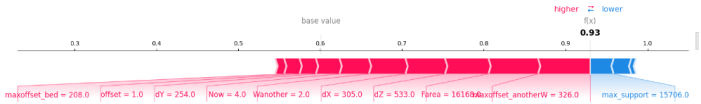


Fig. 8: Force Plot label "1"

Decision Plot

We will now display the decision_plot. It visually depicts the model decisions by mapping the cumulative SHAP values for each prediction.

shap.decision_plot(explainer.expected_value[1], shap_values[1], X_test.columns)

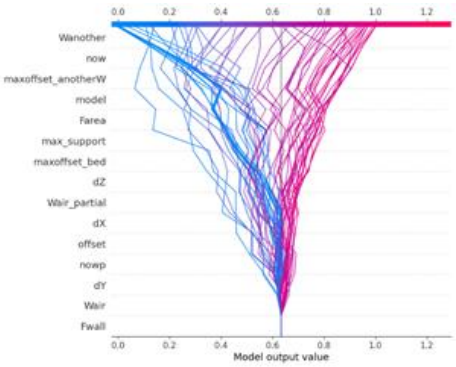


Fig. 9: Decision Plot

LIME, or Local Interpretable Model-Agnostic Explanations, offers localized insights into individual predictions made by complex machine learning models like Random Forests[27]. Analyzing the provided feature names, LIME elucidates the significance of each feature in determining the need for support during 3D printing. By highlighting key features such as "offset" and "maxoffset_bed", LIME reveals critical factors influencing the stability and support requirements of the printing process. Its interpretability empowers users to validate model predictions, identify areas for model improvement, and optimize 3D printing workflows effectively. Leveraging LIME's explanations in conjunction with domain expertise enables a comprehensive understanding of the intricate relationship between input features and the necessity for support structures in 3D printing.

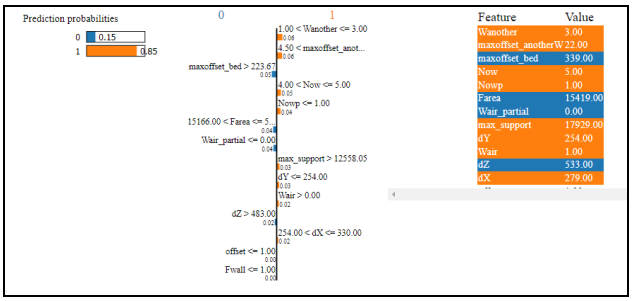


Fig. 10. : LIME analysis

3. Building Machine Learning Model

3.1 Split Dataset for Training and Testing

Machine learning involves three subsets: training, testing, and validation. Training trains the model, testing evaluates its performance on unseen data, and validation aids in hyper parameter tuning. A balanced distribution of classes is crucial to avoid biased training and evaluation. The ultimate goal is to develop a model that generalizes well to new, unseen data, using distinct train, test, and validation sets. In particular, we use Need support and No support for our experiments. For both types, we divided respective datasets into three splits - 70 percent for training, 15 percent for CV and 15 percent for testing purposes. We use stratified sampling in creating train/CV/test splits. Stratified sampling allows us to maintain the same proportion of each class in a split as in the original dataset. Details of the splits are mentioned in tables 3.

Table 3: Splitting accuracy

Algorithm	Training Accuracy	Testing Accuracy	Cross-Validation Mean Accuracy
Logistic Regression:	0.8	0.75	0.72 (+/- 0.07)
Support Vector Machine:	0.72	0.69	0.69 (+/- 0.22)
Random Forest:	1	0.91	0.79 (+/- 0.17)
K-Nearest Neighbours:	0.86	0.81	0.72 (+/- 0.14)
Decision Tree:	1	0.84	0.70 (+/- 0.32)
Gradient Boosting:	0.99	0.93	0.76 (+/- 0.30)

In this paper 6 classification algorithms Logistic Regression, Support Vector Machine, Random Forest, K-Nearest Neighbours, Decision Tree, and Gradient Boosting are used.

Algorithm	Description
Logistic Regression	Linear classification algorithm modeling the probability of class membership using the logistic function.
Support Vector Machine	Constructs an optimal hyperplane to separate classes, maximizing the margin while penalizing misclassifications.
Random Forest	Ensemble learning method using multiple decision trees, reducing overfitting and providing feature importance.
K-Nearest Neighbors	Classifies data based on the majority class among the K nearest neighbors, simple and intuitive.
Decision Tree	Hierarchical tree structure partitioning feature space based on decision rules to maximize class purity.
Gradient Boosting	Ensemble technique adding weak learners sequentially, focusing on minimizing errors to improve predictive power.

The table summarizes the key characteristics and roles of each classification algorithm, aiding in understanding their differences and applications. In summary, each classification algorithm has its strengths and weaknesses, making them suitable for different types of datasets and problem scenarios. Understanding the underlying principles and characteristics of each algorithm is essential for choosing the most appropriate one for a given classification task. Additionally, ensemble methods like Random Forest and Gradient Boosting often provide improved predictive performance by combining the strengths of multiple base learners.

3.2 Evaluation metrics

After making sure the data is good and ready then can continue to building our model. In this paper tried to build 6 different models with different algorithm. First step is to create a baseline model for each algorithm using the default parameters set by sklearn and after building all 6 of models compare them to see which works best for our case. The confusion matrix will be used as the base for the evaluation of model. An exploration of various evaluation metrics, such as accuracy, precision, recall, and F1 score, and Cohen Kappa Score.

		True Class: Positive	True Class: Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * TP}{2 * TP + FP + FN}$$

Where: TP = True Positive; FP = False Positive; TN = True Negative; FN = False Negative.

Cohen Kappa Score: Cohen's kappa measures the agreement between two raters who each classify N items into C mutually exclusive categories.

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

Where Po is the empirical probability of agreement on the label assigned to any sample (the observed agreement ratio), and Pe is the expected agreement when both annotators assign labels randomly. Pe is estimated using a per-annotator empirical prior over the class labels

Area Under Curve (AUC): indicates how well the probabilities from the positive classes are separated from the negative classesSince our aim is to anticipate as many genuine good outcomes as possible, in this instance, we would like to concentrate on our model's recall value. due to a model's incorrect classification, which requires support in reality.

4. Results and Discussion

Building Model

Utilizing the training dataset build the models using Logistic Regression, Support Vector Machine,Random Forest,K-Nearest Neighbours, Decision Tree, and Gradient Boosting. Calculate evaluation matrix for each of the model.

Real-world examples demonstrating the impact of dataset quality on model performance. Evaluation Matrix for Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, Gradient Boosting, and K-Nearest Neighbors

Table 4: Model performance

Algorithm	Accuracy	Precision	Recall	F1 Score	Cohens Kappa Score	Area Under Curve
Logistic Regression:	0.72	0.72	0.72	0.72	0.43	0.71
Support Vector Machine:	0.69	0.78	0.69	0.65	0.34	0.66
Random Forest:	0.88	0.88	0.88	0.87	0.75	0.87
K-Nearest Neighbors:	0.81	0.81	0.81	0.81	0.62	0.81
Decision Tree:	0.90	0.90	0.90	0.90	0.79	0.90
Gradient Boosting:	0.89	0.89	0.89	0.89	0.77	0.88

Model Comparison

After building all of 6 model, compared how well each model perform. To do this create two chart, first is a grouped bar chart to display the value of accuracy, precision, recall, f1, and kappa score of our model, and second a line chart to show the AUC of all our models.

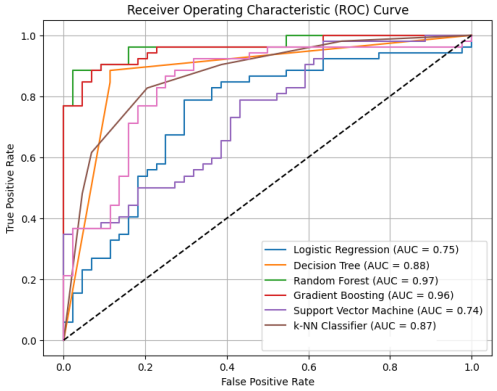


Fig. 11: Model Comparison

From the Fig.s above we can see that our Random Forest model tops the other models in 6 of the 6 metrics we evaluate, except precision. So we can assume that Random Forest is the right choice to solve our problem

Model Optimization

On the next part, optimizeRandomForest model by tuning the hyper parameters available from the scikit-learn library. After finding the optimal parameters evaluate new model by comparing it against our base line model before. Below shows the Overview of input parameter grid.

Decision Tree - Best Hyperparameters: {'max_depth': 2, 'min_samples_split': 5}

RandomForestClassifier - Best Hyperparameters: {'max_depth': 20, 'n_estimators': 100}

KNeighborsClassifier - Best Hyperparameters: {'n_neighbors': 1}

Decision Tree - Test Accuracy: 0.7846153846153846

RandomForestClassifier - Test Accuracy: 0.8769230769230769

KNeighborsClassifier - Test Accuracy: 0.7230769230769231

Tuning Hyperparameter with GridSearchCV

GridSearchCV functionality used from sklearn to find the optimal parameter for the model. Baseline model (named rf_grids) provided, scoring method (in this case recall is used as explained before), and also various parameters value. The GridSearchCV function will then iterate through each parameters combination to find the best scoring parameters.

Additionally, this function enables us to train the model using cross-validation, in which each iteration of our data is divided into five (the number of folds is customizable based on the parameter). The last fold will be used as validation data once the models have been trained on 4/5 of the data. This process will be done 5 times until all of the folds have been used.

The best_params_ attribute from our grid search object allows us to view the combination of parameters that yields the best results. The longer the procedure takes, the more combinations that are given. In an effort to save running time, RandomizedSearchCV can also try to just randomly select a predetermined amount of parameters.

Fitting 5 folds for each of 405 candidates, totalling 2025 fits

{'max_depth': 50,

'max_features': 2,

'min_samples_leaf': 3,

'min_samples_split': 8,

'n_estimators': 1000}

Evaluating Optimized Model

Once the model's optimal parameter has been determined, we can save our optimized model into the best_grid variable by using the GridSearchCV object's best_estimator_ property. And then the helper function is used to calculate the six assessment metrics and compare them with base model.

Accuracy: 0.9193548387096774

Precision: 0.8837209302325582

Recall: 1.0

F1 Score: 0.9382716049382717

Cohens Kappa Score: 0.8232611174458381

Area Under Curve: 0.9188596491228069

Model Comparison

The script has created a plot identical to the previous one, but used both the optimized and original Random Forest model. Additionally, shown the difference on each assessment metric so that we can determine whether our optimized model outperforms the original.

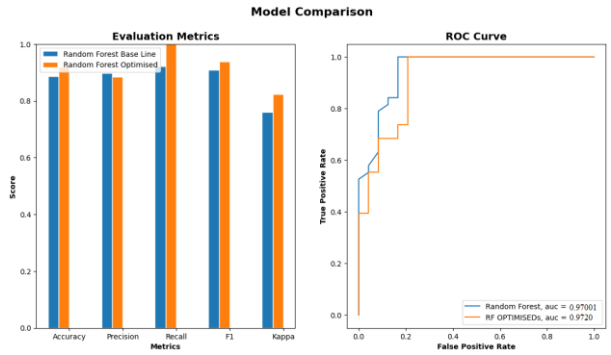


Fig. 12: Optimized Model Comparison

Change in Accuracy: 0.032258064516129004

Change in Precision: -0.013714967203339312

Change in Recall: 0.07894736842105265

Change in F1 Score: 0.029180695847362603

Change in Cohens Kappa Score: 0.06304012297069994

Change in Area Under Curve: -0.030701754385965008

The result show that our optimised performed little bit better than the original model. The optimised models show an increase in 4 out of the 6 metrics but perform worse in the other metrics, especially the Precision with -0.013 decrease. Because this paper focuses on predicting as many actual positive values as possible we should stick with our original model for the prediction because it has higher recall score.

Output

What will happen after our model is ready? Now need to explore the capacity to develop a model with significant reusability is essential. So at last it is discussed how to create a prediction based on new data and how to save (and load).To save time and avoid having to repeat the entire process, joblib is used to construct a model and then use it in production.

Making Predictions

In this step we will predict the expected outcome of all the row from ouroriginal dataset using the Random Forest model and then save it into a csvfile for easier access in the future.

```
df['need support'] = rf.predict(X)
df['need support'] = df['need support'].apply(lambda x: 'yes' if x==0 else 'no')

# Save new dataframe into csv file
df. to_csv('dneed support.csv', index=False)

df.head(10)
```

(a)

	model	dX	dY	dZ	now	Fwall	nowp	Farea	offset	maxoffset_bed	Wanother	maxoffset_anotherW	Wair	Wair_partial	max_support	need support
0	0	330.0	254.0	508.0	1	1	1	71339.0	0	254.0	0	0.0	0	0	0.0	yes
1	0	508.0	254.0	330.0	4	1	1	3913.0	0	2.0	3	199.0	0	1	0.0	no
2	0	508.0	254.0	330.0	4	1	1	3913.0	1	2.0	3	199.0	0	1	0.0	no
3	0	330.0	254.0	508.0	4	1	1	14000.0	0	51.0	3	403.0	0	0	0.0	no
4	0	330.0	254.0	508.0	4	1	1	14000.0	1	51.0	3	403.0	0	0	0.0	no
5	1	178.0	483.0	254.0	1	1	1	34722.0	0	254.0	0	0.0	0	0	0.0	yes
6	1	499.0	287.0	178.0	2	1	1	517.0	0	118.0	1	59.0	0	1	84277.0	yes
7	1	499.0	287.0	178.0	2	1	1	122580.0	0	59.0	1	103.0	0	0	0.0	yes
8	1	178.0	254.0	483.0	3	1	1	14890.0	0	420.0	2	62.0	0	0	0.0	yes
9	1	178.0	254.0	483.0	3	1	1	12761.0	0	62.0	2	420.0	0	0	0.0	yes

(b)

Fig. 13: (a) Code for prediction (b) New dataset with prediction result

Saving Model We can save our model for further model re-usability. This model can then be loaded on another machine to make new prediction without doing the whole training process again.

```
from joblib import dump, load
 # Saving model
dump(rf, 'need support_classification.joblib')
 # Loading model
clf = load('need support_classification.joblib')
```

5. Future Directions

In the future, this research proposes directions for investigation, such as adding more independent variables (curvature/taper in the construction direction) and dependent variables (structure type and build adhesion). The knowledge and optimization of 3D printing processes could be advanced by these expansions, which could greatly enhance the prediction of mechanical properties. In conclusion, the study adds to the expanding corpus of research on the subject of machine learning and 3D printing, providing insightful analysis and useful techniques that can improve the effectiveness and caliber of additive manufacturing procedures.

6. Conclusion

In summary, this work emphasizes how important diverse and high-quality datasets are to machine learning model performance, especially when it comes to 3D printing support structure categorization. The research attempted to categories CAD models according to the requirement for assistance during the 3D printing process by utilizing a variety of machine learning approaches, including Logistic Regression, assistance Vector Machine, Random Forest, K-Nearest Neighbours, Decision Tree, and Gradient Boosting.

The study's conclusions highlight the efficacy of the Random Forest algorithm, which classified the requirement for help with an accuracy of 0.97 based on given process characteristics. Additionally, the model's interpretability is improved by applying SHAP & LIME analysis, which provides insights into the importance of specific variables in forecasting the need for support.

Author contributions

Yogesh Deshpande: Conceptualization, Writing-Reviewing and Editing. **Sonali Patil**: Data curation, Writing-Original draft preparation, Software, Validation., Field study **Dattatraya Parle** : Methodology ,Visualization, Investigation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] A. Dey and N. Yodo, "A systematic survey of FDM process parameter optimization and their influence on part characteristics," *J. Manuf. Mater. Process.*, vol. 3, no. 3, 2019, doi: 10.3390/jmmp3030064.

[2] W. Sobieski and W. Kiński, "Geometry extraction from GCODE files destined for 3D printers," *Tech. Sci.*, no. 2019/2020, 2020, doi: 10.31648/ts.5644.

[3] S. Chowdhury, K. Mhapsekar, and S. Anand, "Part Build Orientation Optimization and Neural Network-Based Geometry Compensation for Additive Manufacturing Process," *J. Manuf. Sci. Eng. Trans. ASME*, vol. 140, no. 3, 2018, doi: 10.1115/1.4038293.

[4] V. Kadam, S. Kumar, A. Bongale, S. Wazarkar, P. Kamat, and S. Patil, "Enhancing surface fault detection using machine learning for 3d printed products," *Appl. Syst. Innov.*, vol. 4, no. 2, 2021, doi: 10.3390/asi4020034.

[5] T. D. Ngo, A. Kashani, G. Imbalzano, K. T. Q. Nguyen, and D. Hui, "Additive manufacturing (3D printing): A review of materials, methods, applications and challenges," *Compos. Part B Eng.*, vol. 143, pp. 172–196, 2018, doi: 10.1016/j.compositesb.2018.02.012.

[6] H. Gonabadi, A. Yadav, and S. J. Bull, "The effect of processing parameters on the mechanical characteristics of PLA produced by a 3D FFF printer," *Int. J. Adv. Manuf. Technol.*, vol. 111, no. 3–4, pp. 695–709, 2020, doi: 10.1007/s00170-020-06138-4.

[7] S. A. Langeland, "Automatic Error Detection in 3D Pritning using Computer Vision," no. January, 2020.

[8] Q. Huang, Y. Wang, M. Lyu, and W. Lin, "Shape Deviation Generator-A Convolution Framework for Learning and Predicting 3-D Printing Shape Accuracy," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1486–1500, 2020, doi: 10.1109/TASE.2019.2959211.

[9] M. Khanzadeh, P. Rao, R. Jafari-Marandi, B. K. Smith, M. A. Tschopp, and L. Bian, "Quantifying Geometric Accuracy with Unsupervised Machine Learning: Using Self-Organizing Map on Fused Filament Fabrication Additive Manufacturing Parts," *J. Manuf. Sci. Eng. Trans. ASME*, vol. 140, no. 3, 2018, doi: 10.1115/1.4038598.

[10] N. Cappetti, A. Naddeo, and G. Salerno, "Influence of control parameters on consumer FDM 3D printing," *Adv. Transdiscipl. Eng.*, vol. 7, no. April 2019, pp. 165–177, 2018, doi: 10.3233/978-1-61499-898-3-165.

[11] A. Omairi and Z. H. Ismail, "Towards machine learning for error compensation in additive manufacturing," *Appl. Sci.*, vol. 11, no. 5, pp. 1–27, 2021, doi: 10.3390/app11052375.

[12] N. Decker and Q. Huang, "Geometric accuracy prediction for additive manufacturing through machine learning of triangular mesh data," *ASME 2019 14th Int. Manuf. Sci. Eng. Conf. MSEC 2019*, vol. 1, no. June, 2019, doi: 10.1115/MSEC2019-3050.

[13] U. Delli and S. Chang, "Automated Process Monitoring in 3D Printing Using Supervised Machine Learning," *Procedia Manuf.*, vol. 26, pp. 865–870, 2018, doi: 10.1016/j.promfg.2018.07.111.

[14] H. S. Kokab and R. J. Urbanic, "Extracting of Cross Section Profiles from Complex Point Cloud Data Sets," *IFAC-PapersOnLine*, vol. 52, no. 10, pp. 346–351, 2019, doi: 10.1016/j.ifacol.2019.10.055.

[15] S. Patil, Y. Deshpande, and D. Parle, "Extracting Slicer Parameters from STL file in 3D Printing," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 14s, pp. 192–204, 2024.

[16] S. Patil, Y. Deshpande, and D. Parle, "Dataset Generation Using a Wall-Based Technique for FDM 3D Printing," *Proc. 3rd Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2023*, 2023, doi: 10.1109/ICACTA58201.2023.10392620.

[17] H. Mamdouh Farghaly and T. Abd El-Hafeez, "A high-quality feature selection method based on frequent and correlated items for text classification," *Soft Comput.*, vol. 27, no. 16, pp. 11259–11274, 2023, doi: 10.1007/s00500-023-08587-x.

[18] N. Pudjihartono, T. Fadason, A. W. Kempa-Liehr, and J. M. O'Sullivan, "A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction," *Front. Bioinforma.*, vol. 2, no. June, pp. 1–17, 2022, doi: 10.3389/fbinf.2022.927312.

[19] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Comput. Sci.*, vol. 2, no. 3, pp. 1–21, 2021, doi: 10.1007/s42979-021-00592-x.

[20] B. Remeseiro and V. Bolon-Canedo, "A review of feature selection methods in medical applications," *Comput. Biol. Med.*, vol. 112, no. February, p. 103375, 2019, doi: 10.1016/j.compbiomed.2019.103375.

[21] R. C. Chen, C. Dewi, S. W. Huang, and R. E. Caraka, "Selecting critical features for data classification based on machine learning methods," *J. Big Data*, vol. 7, no. 1, 2020, doi: 10.1186/s40537-020-00327-4.

[22] M. S. Joshi, A. Flood, T. Sparks, and F. W. Liou, "Applications of supervised machine learning algorithms in additive manufacturing: A review," *Solid Free. Fabr. 2019 Proc. 30th Annu. Int. Solid Free. Fabr. Symp. - An Addit. Manuf. Conf. SFF 2019*, pp. 213–224, 2019.

[23] M. S. Zulfiker, N. Kabir, A. A. Biswas, T. Nazneen, and M. S. Uddin, "An in-depth analysis of machine learning approaches to predict depression," *Curr. Res. Behav. Sci.*, vol. 2, no. February, p. 100044, 2021, doi: 10.1016/j.crbeha.2021.100044.

[24] K. Dissanayake and M. G. M. Johar, "Comparative study on heart disease prediction using feature selection techniques on classification algorithms," *Appl. Comput. Intell. Soft Comput.*, vol. 2021, 2021, doi: 10.1155/2021/5581806.

[25] K. Aas, M. Jullum, and A. Løland, "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values," *Artif. Intell.*, vol. 298, p. 103502, 2021, doi: 10.1016/j.artint.2021.103502.

[26] Y. Lu, X. Fan, Y. Zhang, Y. Wang, and X. Jiang, "Machine Learning Models Using SHapley Additive exPlanation for Fire Risk Assessment Mode and Effects Analysis of Stadiums," *Sensors*, vol. 23, no. 4, 2023, doi: 10.3390/s23042151.

[27] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, pp. 1–45, 2021, doi: 10.3390/e23010018.