# Declaration of Authorship

We hereby declare that this thesis titled, Data Driven Modelling of Composites, and the work presented in it are our own. We confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.
- Where the thesis is based on work done by ourselves jointly with others, We have made clear exactly what was done by others and what we have contributed.

Date:

Signed:

Aman Shrivastava Mihir Rana Yash Sharma

# Abstract

Over the last thirty years, composite materials, plastics, and ceramics have been the dominant emerging materials. The volume and number of applications of composites have grown steadily, penetrating and conquering new markets relentlessly. Modern composites constitute a significant proportion of the engineered materials market ranging from everyday products to sophisticated niche applications.

The field of composite materials science, relies heavily on experiments and simulation based models in order to better predict their characteristics and discover new materials with improved properties. Lately, the big data generated by such experiments and simulations has offered unprecedented opportunities for application of data-driven techniques in this field, thereby opening up new avenues for accelerated materials discovery and design.

In this project, we have developed algorithmic frameworks for predicting properties of composites to aid in the meta-modelling process and in advancing the automated extraction of useful information from simulated data to make predictions at scales that are currently inaccessible. The frameworks designed are applicable to virtually any new composite. They have been designed in a way that allows instantaneous recommendation of composite materials and prediction of their aggregate properties.

# Acknowledgements

We take this opportunity to express our deepest gratitude to all those who encouraged and supported us to complete this project.

We cant emphasise how grateful we are to our mentor, Dr. Siladitya Pal, who in spite of being extraordinarily busy with his duties, took time out to hear and guide us and gave us immense confidence at every stage of our project.

We express our deepest gratitude to our Head of Department, Dr. Dinesh Kumar, for giving us the platform to pursue our ideas. It was a great learning experience that enabled us to use our engineering concepts to make a contribution to the industry.

We also wish to thank our professors, Dr. A. Parashar, Dr. I. V. Singh, and Dr. M. M. Joglekar for giving us critical feedback that helped us explore different dimensions of our project.

Lastly, we wish to thank our parents and friends for their constant support that helped us to deliver to the best of our abilities.

We perceive this opportunity as a big milestone in our career development. We will strive to use the gained skills and knowledge in the best possible way, and will continue to work on their improvement, in order to attain our desired career objectives.

# Contents

# List of Figures

# List of Tables

*Dedicated to*

*Dr. Siladitya Pal*

*and*
*The Data Science Community...*

# Chapter 1

# Introduction

## 1.1 Composites

A composite is a material made from two or more constituent materials with significantly different physical or chemical properties that, when combined, produce a material with characteristics different from the individual components. The individual components remain separate and distinct within the finished structure. The new material may be preferred for many reasons, primarily to maximize the useful properties and minimize the weaknesses of constituent materials; common examples include materials which are stronger, lighter, or less expensive when compared to traditional materials. One of the oldest and best-known composites, glass-fibre reinforced plastic (GRP), combines glass fibres (which are strong but brittle) with plastic (which is flexible) to make a composite material that is tough but not brittle. Composites are typically used in place of metals because they are equally strong but much lighter.

### 1.1.1 Need and Applications of Composites

Composites are one of the most widely used materials because of their adaptability to different situations and the relative ease of combination with other materials to serve specific purposes and exhibit desirable properties.

In surface transportation, reinforced plastics are the kind of composites used because of their huge size. They provide ample scope and receptiveness to design changes, materials and processes. The strength-weight ratio is higher than other materials.

Their stiffness and cost effectiveness offered, apart from easy availability of raw materials, make them the obvious choice for applications in varying fields. Some common examples are:

1. **Aircraft Industry:** More than 20% of the A380 is made of composite materials, mainly plastic reinforced with carbon fibres. The design is the first large-scale use of glass-fibre-reinforced aluminium, a new composite that is 25% stronger than conventional airframe aluminium but 20% lighter.

2. **Automobile Industry:** The automotive industry faces many challenges, including increased global competition, the need for higher-performance vehicles, a reduction in costs and tighter environmental and safety requirements. The materials used in automotive engineering play key roles in overcoming these issues: ultimately lighter materials mean lighter vehicles and lower emissions. Composites are being used increasingly in the automotive industry due to their strength, quality and light weight.

3. **Construction:** Concrete is a versatile and cheap material, with a vast range of applications around the home. Brick laying, constructing paths and driveways, foundations to buildings and walls, are some of the practical applications. Concrete has a similarly wide and varied range in industrial applications.

4. **Wind Mills:** Currently, carbon fibre is used primarily in the spar, or structural element, of wind blades longer than 45m, both for land-based and offshore systems. The higher stiffness and lower density of CF allows a thinner blade profile while producing stiffer, lighter blades.

### 1.1.2 Motivation

One of the main barriers to usage of composites across various domains is the complexity involved in modelling of composites, and thereby the prediction of its properties. Since precise material properties are often needed to satisfy industrial needs and criteria, design of composites with tailored properties is of utmost importance. The design and analysis of such composites faces following challenges -

1. Computational Expense

2. Complexity of Simulations

3. Rigorous Experimentation and Validation

4. Time Expensive Process

## 1.2 Data-Driven Modelling of Composites

The discussion in the previous section necessitates the need for an alternative, fast, and accurate solution to designing of composites. In this project, we present a Machine Learning based approach to this problem.

### 1.2.1 Introduction to Machine Learning

Machine Learning (ML) is a type of Artificial Intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data. Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field.

A subfield of ML, called **Supervised Learning**, is the task of inferring a function from labelled training data. The training data consists of a set of training examples. In general, the computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

**Cross-Validation (CV)** is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. In a prediction problem, a model is usually given a dataset of known data on which training is run (**training dataset**), and a dataset of unknown data against which the model is tested (**testing dataset**). The goal of CV is to define a dataset to "test" the model in the training phase (i.e., the **validation dataset**), in order to limit problems like overfitting (overreaction and specificity to minor fluctuations in the training data), give an insight on how the model will generalize to an independent dataset, etc.

In this project, a similar methodology of Supervised Learning has been followed wherein training data has been fed to the ML model, comprising aggregate properties of composites (Youngs Modulus, Poissons Ratio, Shear modulus, etc.) and its constituents, and predictions are made on unseen data (composites made from any two given materials) based on the historic fed data.

### 1.2.2 Advantage of Machine Learning in Designing Composites

Typically, the kind of data that is fed to a Machine Learning model is in the form of text, images, audio, or video. In this project, we input textual data to the model which is of the order of kilobytes. In such a case, a typical ML model takes time in the order of milliseconds to train to model. Furthermore, predictions are made instantaneously. From this discussion, it is evident that a data-driven approach is significantly superior to a traditional simulation technique in terms of computational time. Using cross-validation, if the same accuracy can be guaranteed, ML models would surpass any finite element simulations and counter all 4 challenges to traditional approaches that we presented earlier in this section.

In our analysis, we have considered data in the range of 1 GPa to 10 GPa (for matrix) and 46 GPa to 973 GPa (for fibres). This virtually covers any given composite that is used in practice. An FE simulation based approach would take time in the order of days to model such a wide variety of composites. On the contrary, ML models take only a few seconds; furthermore, they need to be run only during training. Once the parameters are optimized, predictions are made instantaneously.

### 1.2.3 Objective

The objective of this project is to develop an algorithmic framework to predict properties of composites to aid in modelling process for advancing the automated extraction of useful information from huge amounts of simulated data to make predictions at scales that are currently inaccessible. This can be achieved by the following subtasks:

- Generate and gather simulated and experimental data of various composite materials and feed it to the ML model

- Make accurate and fast predictions and validate the results
- Make predictions of properties on unknown materials, and recommend materials based on target properties

### 1.2.4 Scope

The project focuses on prediction and recommendation of composite material properties by learning the trends in materials fed to the model while training. Once trained, the model is able to predict properties for composites with varying:

- Constituent matrices and fibres, such as:
  - Youngs moduli of matrix and fibres
  - Poissons ratios of matrix and fibres
  - Diameter of fibres
- Volume fraction of fibres
- Spatial arrangement of fibres

The data generated and gathered must be reliable in the first place. This has been achieved by:

- Cross-validation against unseen data while training of ML model
- Validation against Finite Element simulations performed on different materials
- Validation against experimental data

### 1.2.5 Timeline

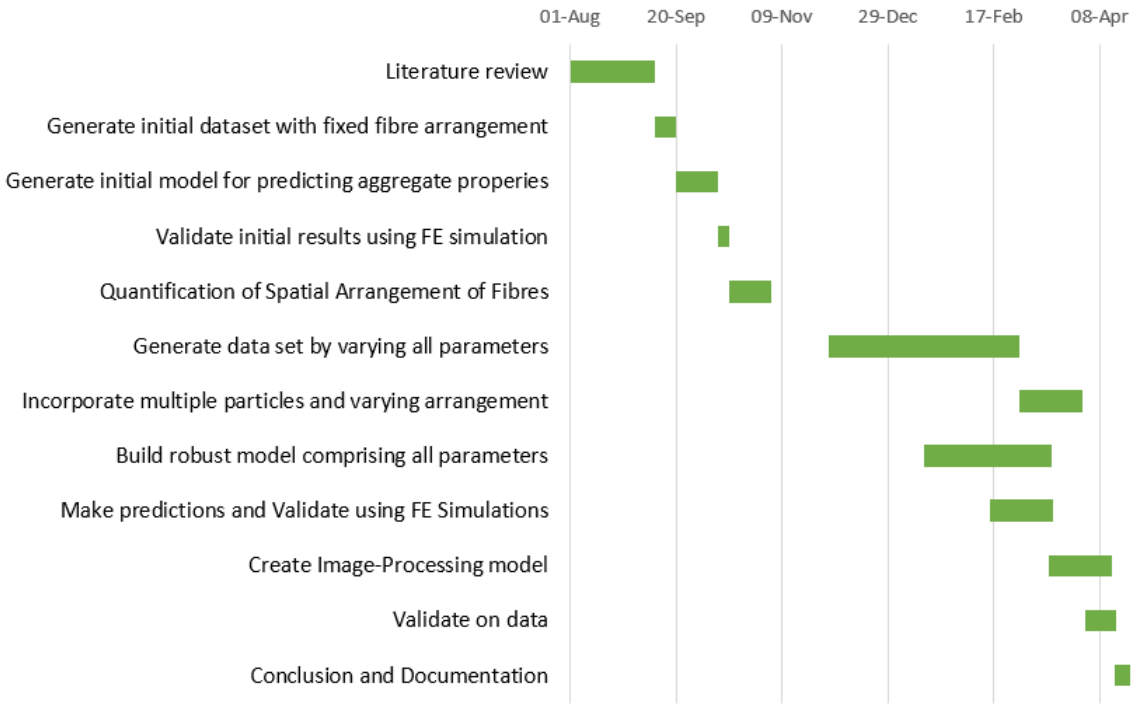


FIGURE 1.1: Gantt Chart showing the Timeline of the project

# Chapter 2

# Literature Review

Composites are of great importance and the need for new tailored composites that satisfy our needs thus have immense significance. Attributable to its more versatile and tuneable material properties, a range of composites have been widely used in aerospace, marine, vehicle and biomedical industry. Their different microstructures with two or more constituents allow achieving desirable properties such as multi-functionality and lightweight. To control the material properties, the spatial layout of the microstructure and/or the compositions of the constituent phases is extremely crucial.

## 2.1 Finite Element Method and Representative Volume Element

The effective elastic moduli of the composite are generally determined by finite element analysis of the RVE. It is paramount in such analyses that the correct boundary conditions be imposed such that they simulate the actual deformation within the composite. Sun and Vaidya [1] described a method based on the finite element analysis of an RVE to predict the effective mechanical properties of an unidirectional fiber composite and discussed the importance of the boundary conditions imposed on the RVE.

Most of the existing methods regard the micro-scale geometry as a periodic structure, assuming a deterministic and ordered distribution of fibers. However, the realistic distribution of fibers has been known to be non-uniform and randomly distributed.

Therefore, methods based on periodic fiber distributions cannot give accurate predictions of the effective properties of the composite, especially on the elastic-plastic behavior under transverse loading conditions, due to the inadequate modeling of the resin-rich region and the fiber-aggregate region [2]. Extensive studies on the effect of non-uniform fiber distribution on the overall composite behaviors have been done. Sun et al.[3] built random and periodic unit cell models for an E-glass particle reinforced composite and studied the lower bound and upper bound of Youngs modulus under iso-displacement loading and iso-stress loading, respectively. By comparing with experimental data, the random unit cell model shows better prediction of overall bounds for elastic properties. Trias et al. [4] demonstrated that random models must be considered for the simulation of local phenomena, as the use of periodic models leads to underestimation of matrix cracking and damage initiation. Zhang and Yan [5] found that the rectangular and hexagonal periodic models show inconsistency in predicting the transverse normal and shear modulus, and both rectangular and hexagonal periodic models show insufficient accuracy compared with the random model.

The general approach for modeling the non-uniform spatial arrangement of fibers includes characterization of the microstructure, statistical analysis of the microstructure, reconstruction of the random RVE model and micromechanical modeling of the RVE. Firstly, the realistic microstructure with fiber distribution is characterized using, for example, a scanning electron microscope (SEM). Digital image analysis is then carried out on the original image to identify the fibers based on a color threshold algorithm. From this image, the information such as the distribution of fiber radius and the distances between neighboring fibers can be extracted. The obtained statistical parameters are utilized to reconstruct a statistically equivalent RVE based on certain numerical algorithms. Finally, the reconstructed RVE is meshed to create a finite element model, which is used to predict the effective mechanical performance. To generate a statistically equivalent RVE, the hard-core model (also called the random sequential adsorption model) has been widely used [6–8].This method creates a set of randomly distributed points inside a square region, with the constraint that no pair of points may be closer than a certain minimum distance. Yang et al. [9] simulated the non-uniform microstructure of a ceramic matrix composite using the hardcore model and validated the statistical equivalency of the reconstructed random model against the realistic fiber distribution.

However, the hard-core model does not permit the fiber volume fraction to be greater than 54.7% due to the presence of the so-called jamming [10]. To overcome the

jamming configuration of the hard-core model, a shaking process can be included to increase the number of inclusions through giving each fiber a small random displacement independent of its neighbor fibers. Wongsto and Li [11], Zhang and Yan [5] and Wang et al. [12] obtained non-uniform fiber distributions through disturbing the initially periodical hexagonal/square fiber arrangement. Melro et al. [13] developed a three-step procedure for generation of random fiber distribution, which includes the hardcore model (initial generation of fibers), stirring the fibers (assigning a small random disturbance to each initial generated fibers) and fibers in the outskirts (further generation of fibers in the matrix rich regions produced during the second step). Melro et al.s method, i.e., the random microstructure generation (RMG) algorithm, shows the capability of achieving a fiber volume ratio of 65%.

Vaughan and McCarthy [14] developed a combined experimental-numerical approach, the nearest neighbor algorithm (NNA) that generated fiber distribution with the same geometric features as the experimental samples determined using statistical analysis, reproducing both the short and long range interaction of fibers. The NNA model is fast in computational time, but, statistically, it produces a larger frequency of small inter-fiber distances and a lower frequency of large inter-fiber distances. The modified NNA algorithm was proposed by Wenzi Wang and Yonghui Dai[15] to compensate for the drawbacks of NNA algorithm.

## 2.2 Data Driven Modelling of Composites

The field of materials science relies on experiments and simulation-based models to understand the physics of different materials in order to better understand their characteristics and discover new materials with improved properties for use in society at all levels. Lately, the big data generated by such experiments and simulations has offered unprecedented opportunities for application of data-driven techniques in this field, thereby opening up new avenues for accelerated materials discovery and design. The need for such data analytics has also been emphasized by the Materials Genome Initiative (MGI)[16], which envisions the discovery, development, manufacturing, and deployment of advanced materials twice as fast and at a fraction of the cost.

In the past decade, great amount of work has been done in the field of data analytics with development of new preprocessing techniques, algorithms and data mining techniques. With advancement in this field, opportunity to use these algorithm in the

field of material science has also significantly increased. Ruoqian Liu and Abhishek Kumar[17] addressed the problem of identifying the complete space (or as much of it as possible) of microstructures that can be theoretically predicted to yield the desired combination of properties demanded by a selected application.

He proposed a route to address these challenges using a machine learning methodology improving the running time by 80%. Prasanna and Xue[18] came up with design strategies using regression model to obtain a material with desired elastic properties. Meredig, Agrawal[19] used machine learning model trained on a database of thousands of density functional theory calculations to predict the thermodynamic stability of arbitrary compositions without any other input and with six orders of magnitude less computer time than DFT. These are some of the papers we read to get the idea of how the advancement in data driven techniques is impacting material science industry[20, 21].

There was no literature available on generalised composite generation. The research which has been done in past is either on a particular composite combination or a group of composite combinations. Through our work, we propose to increase this to range of composites. So, if anyone wants to propose a composite with some unique combination of material. Instead of employing a hit and trial method of simulations and test to check the feasibility of composite. They can check the feasibility of that material using our trained model. This will decrease the checking time significantly and will make the process of creating new composite easier.

# Chapter 3

# Methodology

## 3.1 Assumptions

Before beginning the analysis, the assumptions in the underlying architecture of the project must be understood:

- The analysis is confined to 2-D space. This implies that fibres are considered (and not particles) of infinite length (or equal to length of material in that dimension).
- In the cross-section, all fibres are perfectly circular in shape. That is, the analysis is performed for cylindrical fibres.
- The analysis is confined to two phases of the composite.
- Very small particles will be ignored assuming they have negligible overall effect.

## 3.2 General Work Flow

The general flow of work of the project is as shown in Figure 3.1. First, data is generated using a Random Sequential Adsorption model, which generates statistically equivalent (SE) structures with different fibre arrangements. With this, 5 SE structures are generated which are isotropic. Next, full field simulations are performed on these 5 structures and their aggregate properties (Youngs Modulus, Poissons Ratio,

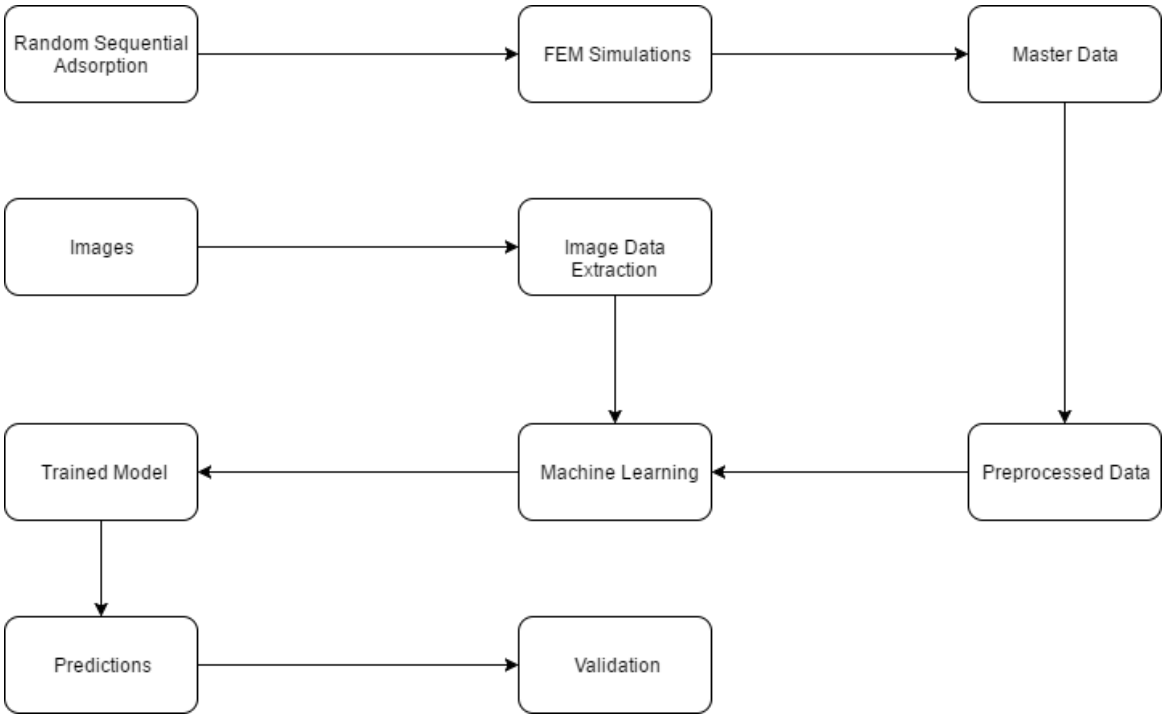


FIGURE 3.1: Overall Work Flow of the Project

Shear Modulus) are recorded. After that, these 5 values of each of the properties are averaged to arrive at a single value. This ensures robustness in the model, since there are small variations in properties when the ratio L/D $\geq$ 20 (L = length of unit cell; D = Diameter of fibre)[22]. The final properties are then fed to the training data of the ML model, where it is first prepared according to the desired format as is shown later.

Another way of gathering data for the ML model is through real images of fibres scattered in the matrix. The image is recognized and the desired properties (diameter, volume fraction) are directly extracted from the image. This data, coupled with the material properties, is sent for training of the ML model.

Once the data, whether it originated from images or textual data, is prepared and is model-ready, the Machine Learning model is trained using several different algorithms. Finally, predictions are made on unseen data, i.e., composite materials which were not used for training, and the results are validated against FEM and experimental data.

TABLE 3.1: Summary of Stress-Strain Data

| | Stress (Pa) | Matrix Young's Modulus (Pa) | Matrix Poisson's Ratio | Fibre Young's Modulus (Pa) | Fibre Poisson's Ratio | Strain | Volume Fraction (%) |
|---|---|---|---|---|---|---|---|
| count | 6.00e+04 | 6.00e+04 | 6.00e+04 | 6.00e+04 | 6.00e+04 | 6.00e+04 | 6.00e+04 |
| mean | 9.159537e+08 | 5.9948e+09 | 0.42476 | 5.063630e+11 | 0.320730 | 0.052192 | 17.500000 |
| std | 6.753535e+08 | 2.4735e+09 | 0.01544 | 2.830782e+11 | 0.018896 | 0.030326 | 8.539197 |
| min | 2.672000e+06 | 1.1500e+09 | 0.40000 | 1.030000e+10 | 0.282000 | 0.001001 | 5.000000 |
| 25% | 3.772325e+08 | 4.1600e+09 | 0.41000 | 2.590000e+11 | 0.306750 | 0.026082 | 10.000000 |
| 50% | 7.670250e+08 | 5.8600e+09 | 0.42400 | 4.845000e+11 | 0.322500 | 0.051775 | 17.500000 |
| 75% | 1.322200e+09 | 8.0975e+09 | 0.43825 | 7.527500e+11 | 0.338000 | 0.078081 | 25.000000 |
| max | 3.958600e+09 | 9.9600e+09 | 0.45000 | 9.990000e+11 | 0.350000 | 0.105000 | 30.000000 |

## 3.3 Data Generation

The entire analysis of this project relies on generation of accurate, and uniformly distributed data emulating realistic one. This data would comprise material properties of the matrix and fibres, along with diameter and volume fraction of fibres. For the purpose of the analysis, three volume fractions were considered: 30%, 40%, and 50%.

### 3.3.1 Data for Stress-Strain Curve

The first data set is created for building the stress-strain curves for various composites. For this, the following features are considered:

- Youngs Moduli of matrix and fibre
- Poissons Ratios of matrix and fibre
- Strain (normally distributed; 10 values considered)
- Volume fraction (5%-30%, in increments of 5%)

As seen in Table 3.1,

- Total size (rows) of data = 60,000
- Number of unique materials = 100

This data set is used for constructing the stress-strain curves of various composite materials and to check how closely they resemble the actual (FE simulated) curves.

The Stress-Strain Distributions of different materials in the training data set is represented below:

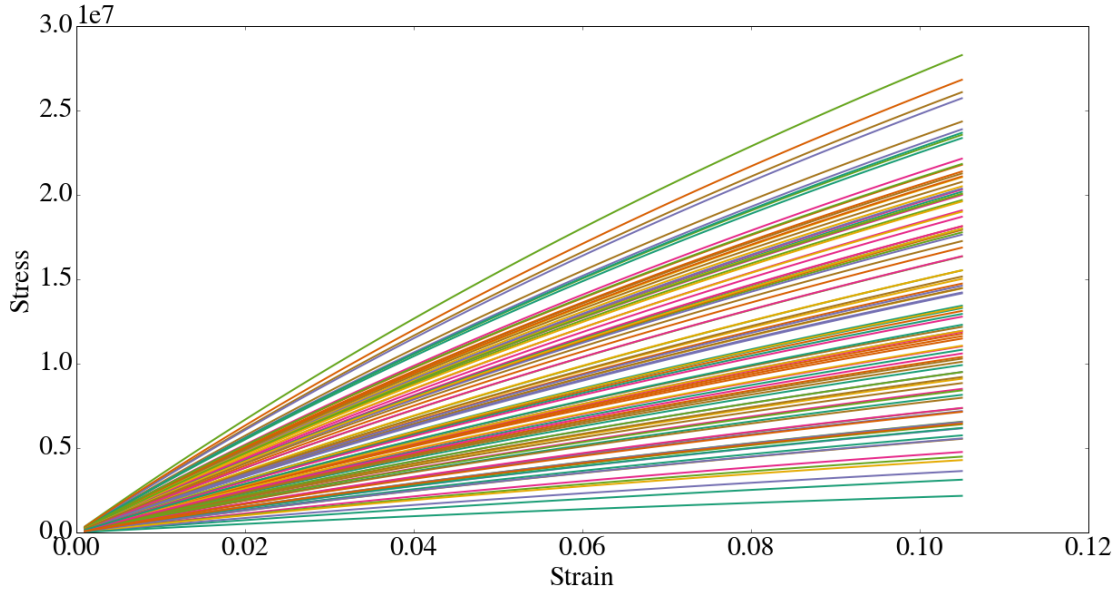


FIGURE 3.2: Stress-Strain Distribution for different materials (VF=5%)

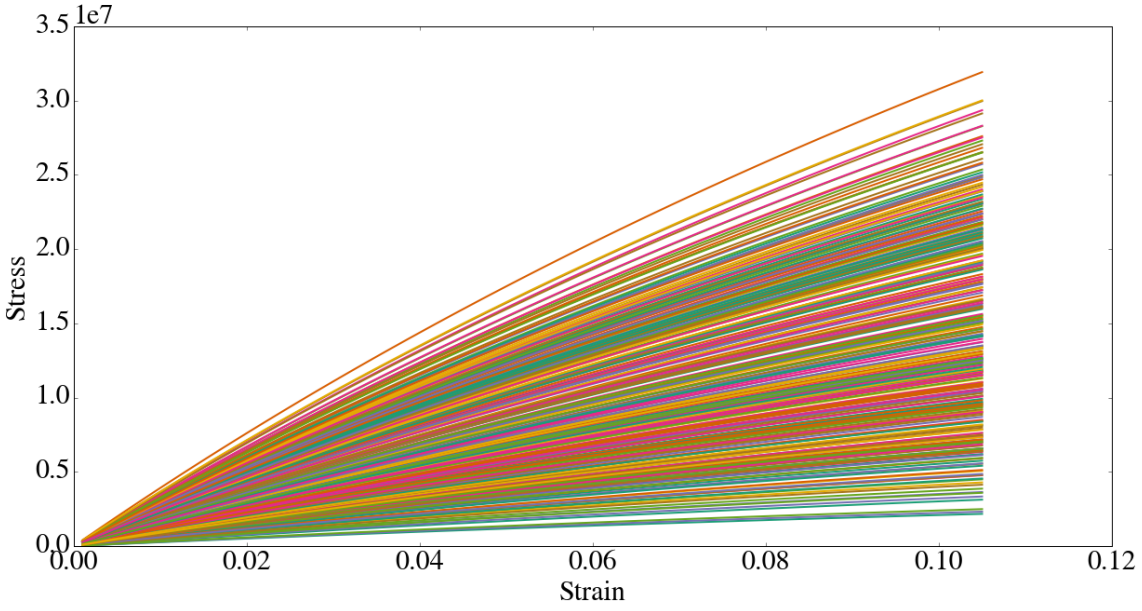


FIGURE 3.3: Stress-Strain Distribution for different materials (VF=15%)

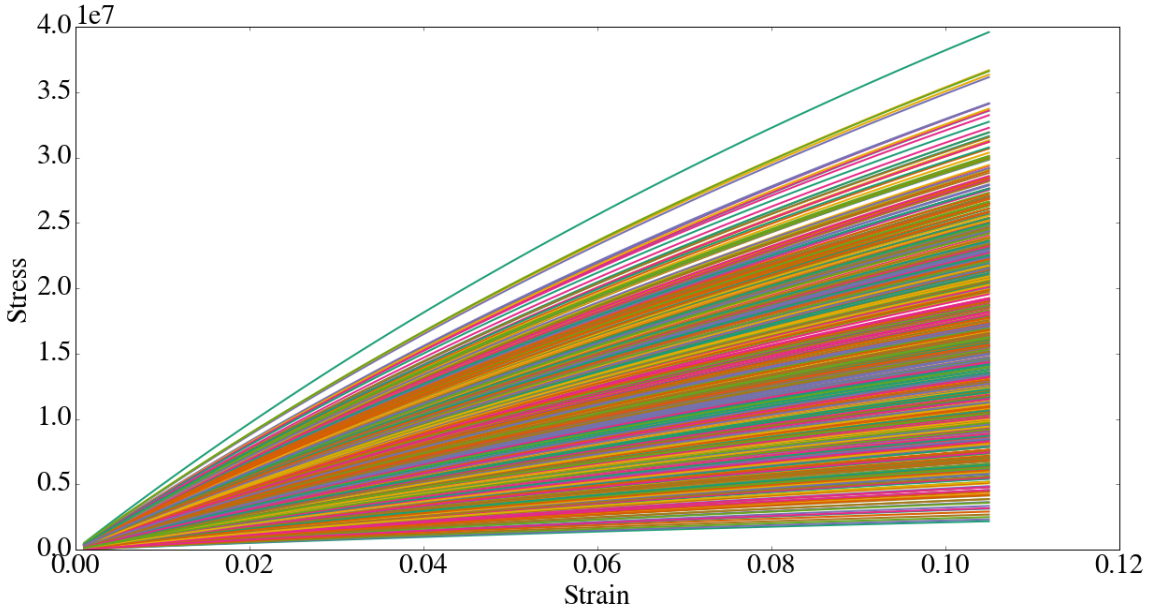


FIGURE 3.4: Stress-Strain Distribution for different materials (VF=30%)

### 3.3.2 Generation of Representative Volume Element

Once the accuracy of the model was verified, the next step was to generate data for higher volume fractions, considering only the final stress and strain values. To generate statistically equivalent Representative Volume Elements (RVE), the hard-core model, also called the **Random Sequential Adsorption (RSA)** model was used. This method creates a set of randomly distributed points inside a square region, with the constraint that no pair of points may be closer points inside a square region, and that no pair of points may be closer than a certain minimum distance.

In this model, structures are created by first setting the values of volume fraction, and limiting the distance between fibres (circles) and between circles and the walls of the unit cell. Thereafter, the required diameter of circles and their number is determined based on the information provided. Finally, the required arrangement is obtained in a random manner to ensure isotropic nature. The code is attached in the Code Snippets section.

Using the RSA model, for each matrix-fibre pair with specified volume fraction, 5 different structures were obtained in the form of images, with locations of centres of all circles in a 2D array. This data was further used as input for finite element simulations.

One drawback of the RSA model is that it does not permit the fibre volume fraction to be greater than 54.7% due to the presence of the so-called jamming limit. As most of the experimental data available was pertaining to composite materials with a greater volume fraction, countering this issue was imperative. To do so, first, the volume fraction was brought up to 54.7%, and then by using jittering functionalities, the fibre arrangements were varied randomly and additional fibres were introduced manually into the matrix till the volume fraction reached 60%. This was used only for validation.

### 3.3.3 Finite Element Method Simulations

Once 5 different statistically equivalent structures were obtained for each composite material for a particular volume fraction, all 5 structures were simulated using Finite Element Modelling on ANSYS. The features used were:

- Transverse Youngs Moduli
- Transverse Poissons Ratios
- Volume Fraction of fibres

For the analysis, plain strain conditions were used. Thus, principal strains were introduced along two perpendicular axes, and a third shear strain was introduced in the plane containing these two axes. This can be represented as follows:

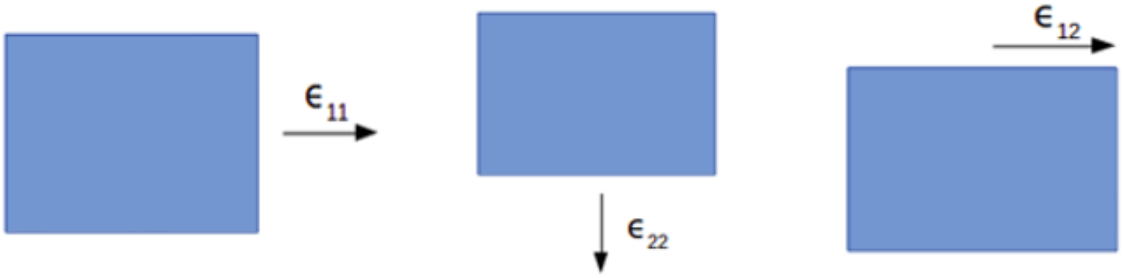


FIGURE 3.5: Plain strain conditions for a 2-D body

The stress field (von Mises) developed due to these strains ($\overline{E_{11}}$ and $\overline{E_{22}}$) for various materials is shown in Figures 3.6-3.8.

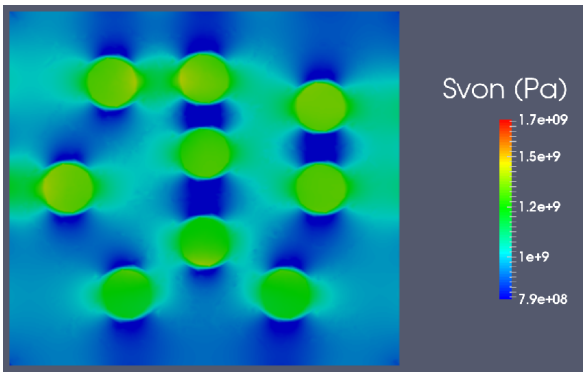


FIGURE 3.6: von Mises Stress Field for Composite 1

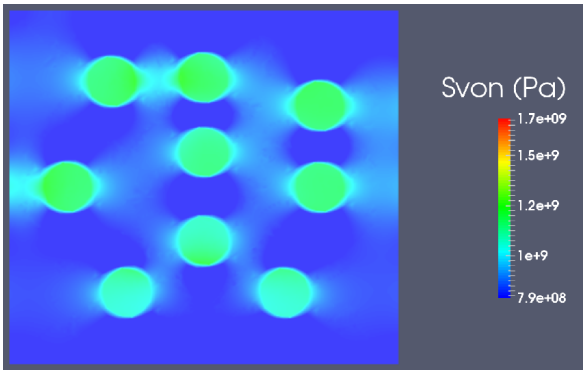


FIGURE 3.7: von Mises Stress Field for Composite 2

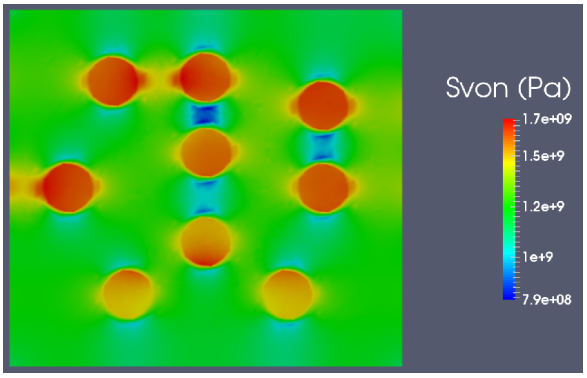


FIGURE 3.8: von Mises Stress Field for Composite 3

Lastly, from the simulations, the average stress values were recorded. From these values, using the tangent formulae (shown below), tangent moduli for the composites were calculated. These moduli were later used as a feature for the ML model.

TABLE 3.2: Description of Data from RSA

| | Matrix Young's Modulus (Pa) | Matrix Poisson's Ratio | Fibre Young's Modulus (Pa) | Fibre Poisson's Ratio | Volume Fraction |
|---|---|---|---|---|---|
| count | 3.000000e+02 | 300.000000 | 3.000000e+02 | 300.000000 | 300.000000 |
| mean | 5.574183e+09 | 0.305267 | 4.987511e+11 | 0.222166 | 0.400000 |
| std | 2.836223e+09 | 0.059083 | 2.654900e+11 | 0.073148 | 0.081786 |
| min | 1.078529e+09 | 0.201779 | 4.610544e+10 | 0.102230 | 0.300000 |
| 25% | 2.931881e+09 | 0.251675 | 2.931493e+11 | 0.158937 | 0.300000 |
| 50% | 5.475158e+09 | 0.312091 | 4.872371e+11 | 0.216204 | 0.400000 |
| 75% | 8.189232e+09 | 0.355053 | 6.915839e+11 | 0.285937 | 0.500000 |
| max | 9.960097e+09 | 0.398134 | 9.739165e+11 | 0.349943 | 0.500000 |

Tangent Moduli formulae:

$$\{\overline{\sigma}\} = \{\overline{D}\}\{\overline{\epsilon}\} \tag{3.1}$$

$$\overline{D_{1111}} = \frac{\partial \overline{\sigma_{11}}}{\partial \overline{\epsilon_{11}}} \tag{3.2}$$

$$\overline{D_{2222}} = \frac{\partial \overline{\sigma_{22}}}{\partial \overline{\epsilon_{22}}} \tag{3.3}$$

$$\overline{D_{1212}} = \frac{\partial \overline{\sigma_{12}}}{\partial \overline{\epsilon_{12}}} \tag{3.4}$$

The data obtained can be seen in Table 3.2.

## 3.4 Particle Arrangement Quantification

It is found that besides the volume fraction, shape, and orientation of the reinforcements, the distribution of fibers also plays a significant role in the mechanical properties of unidirectional composites.[23]
Thus two different methods were used to quantify structural properties are used.

### 3.4.1 Distance Energy Method

For quantifying particle arrangement of the composite the particles were considered to represent a graph which allowed the development of a model that helped quantifying the arrangement using the Distance Energy of the graph. The distance energy of a graph $G$ is a recently developed energy-type invariant, defined as the sum of absolute values of the eigenvalues of the distance matrix of $G$.

Let $G$ be a simple connected graph on vertex set $V(G) = v_1, v_2, ..., v_n$. The distance matrix of the graph $G$ is defined as a square matrix $D = D(G) = [d_{ij}]$, where $d_{ij}$ is the distance between the vertices $v_i$ and $v_j$ in $G$. The diameter of $G$, denoted by $diam(G)$, is the maximum distance between any two vertices of $G$. The girth of $G$ is the length of a shortest cycle in $G$. The eigenvalues of $D(G)$ are called the $D - eigenvalues$ of $G$. Since $D(G)$ is a real symmetric matrix, its eigenvalues are real numbers. So we can order so that $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n$ are $D - eigenvalues$ of $G$. The distance energy of the connected graph G is then defined as[24]:

$$E_D = E_D(G) = \sum_{1}^{n} |\mu_i| \tag{3.5}$$

Results demonstrated that this Distance Energy was significantly varied with the variation in particle arrangement hence the value could be used to computationally differentiate between different composite structures.

### 3.4.2 D-Energy Computation

For computation purposes the following functions were developed.

Generating random co-ordinates

```
def generateCoordinates(size):
    radius = 1
    rangeX = (0, 10)
    rangeY = (0, 10)
    qty = size
    deltas = set()
    for x in range(-radius, radius+1):
        for y in range(-radius, radius+1):
            if x*x + y*y <= radius*radius:
                deltas.add((x,y))

    randPoints = []
    excluded = set()
    i = 0
    while i<qty:
        x = random.randrange(*rangeX)
        y = random.randrange(*rangeY)
        if (x,y) in excluded: continue
        randPoints.append((x,y))
        i += 1
        excluded.update((x+dx, y+dy) for (dx,dy) in deltas)
    dataPoints = pd.DataFrame(randPoints)
    plt.scatter(dataPoints[0],dataPoints[1], s=500)
    return dataPoints[0], dataPoints[1]
```

The set of random coordinates hence generated were approximated to be the locations of the randomly distributed particles in the Representative Volume which was considered to be a standard size of 10 units by 10 units for relative comparisons in the generated values of the Distance Energy parameter.

Computing Distance Array

```
def genDistArray (X, Y, size):
    arr = np.zeros(shape=(size,size))
    for i in range(size):
        for j in range(size):
            if (i == j):
                continue
            # print str(i) + " " + str(j)
            arr[i][j] = math.hypot(X[j] - X[i], Y[j] - Y[i])
    return arr
```

Calculating Distance Energy

```
def genDistanceEnergy (arr):
    e_vals, e_vecs = la.eig(arr)
    DistEnergy = 0
    for val in e_vals:
        if val < 0:
            val = -1 * (val)
        DistEnergy = DistEnergy + val
        return DistEnergy
```

### 3.4.3 D-Energy Results

To demonstrate the variation in the value of D-Energy with distribution of particles two different arrangements with respective computations are reported.

```
Distance Array =
[[  0.          4.          2.82842712  6.32455532  6.40312424
    7.21110255  7.28010989  2.23606798]
 [  4.          0.          6.32455532  8.48528137  5.          4.47213595
    3.60555128  5.38516481]
 [  2.82842712  6.32455532  0.          4.          6.70820393
    8.24621125  9.8488578   4.12310563]
 [  6.32455532  8.48528137  4.          0.          6.08276253
    8.24621125 12.04159458  8.06225775]
 [  6.40312424  5.          6.70820393  6.08276253  0.          2.23606798
    7.61577311  8.60232527]
 [  7.21110255  4.47213595  8.24621125  8.24621125  2.23606798  0.
    6.08276253  9.21954446]
 [  7.28010989  3.60555128  9.8488578  12.04159458  7.61577311
    6.08276253  0.          8.        ]
 [  2.23606798  5.38516481  4.12310563  8.06225775  8.60232527
    9.21954446  8.          0.        ]]

Distance Energy =
(45.4380186079+0j)
```

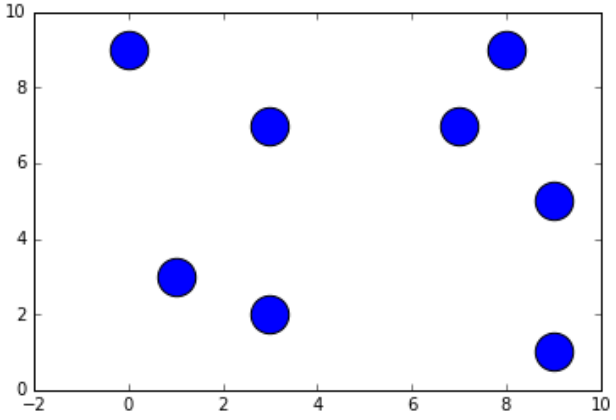


FIGURE 3.9: Sample 1 — D-energy = 45.438

```
Distance Array =
[[ 0.          4.12310563  2.82842712  6.70820393  7.28010989  7.81024968
   6.40312424  1.41421356]
 [ 4.12310563  0.          3.60555128  4.47213595  3.16227766  4.47213595
   5.09901951  5.        ]
 [ 2.82842712  3.60555128  0.          4.12310563  6.40312424  8.06225775
   3.60555128  4.24264069]
 [ 6.70820393  4.47213595  4.12310563  0.          5.09901951  8.
   1.41421356  8.06225775]
 [ 7.28010989  3.16227766  6.40312424  5.09901951  0.          3.16227766
   6.32455532  8.06225775]
 [ 7.81024968  4.47213595  8.06225775  8.          3.16227766  0.
   9.05538514  8.06225775]
 [ 6.40312424  5.09901951  3.60555128  1.41421356  6.32455532  9.05538514
   0.          7.81024968]
 [ 1.41421356  5.          4.24264069  8.06225775  8.06225775  8.06225775
   7.81024968  0.        ]]

Distance Energy =
(39.1196924458+0j)
```

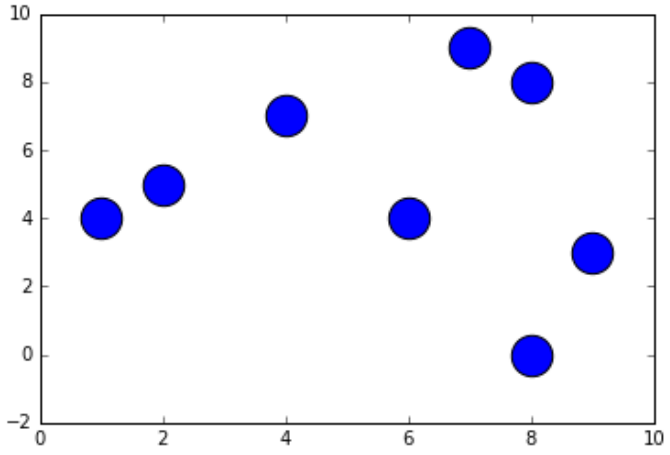


FIGURE 3.10: Sample 2 — D-energy = 39.119

For the same number of particles the D-energy gives a variation of about 5 units in account of different arrangement.

### 3.4.4 Drawbacks of D-Energy

It has been proved that there exist pairs of non-D-cospectral D-equienergetic graphs of order n for every n $\geq$ 6. This means that whenever n $\geq$ 6, there exists at least one pair of arrangements where D-energies are equal.

Thus to distinguish between these arrangements, we need one more parameter to differentiate between them.

### 3.4.5 Image Data Extraction

Aside from the arrangement, volume fraction also plays a very important role in determining the mechanical properties of the composite. Also in several cases, the composites are present to us in form of its magnified image captured during experimentation.

A model was developed to extract physical properties of the composite from these images.

### 3.4.6 Data Extraction Results

To calculate the volume fraction and the digitizing the experimental image OpenCV was used. The blobs in the image are approximated to be perfect circles. All such blobs are then detected and used to calculate the volume fraction of the composite and also generating a digital image representing the composite.

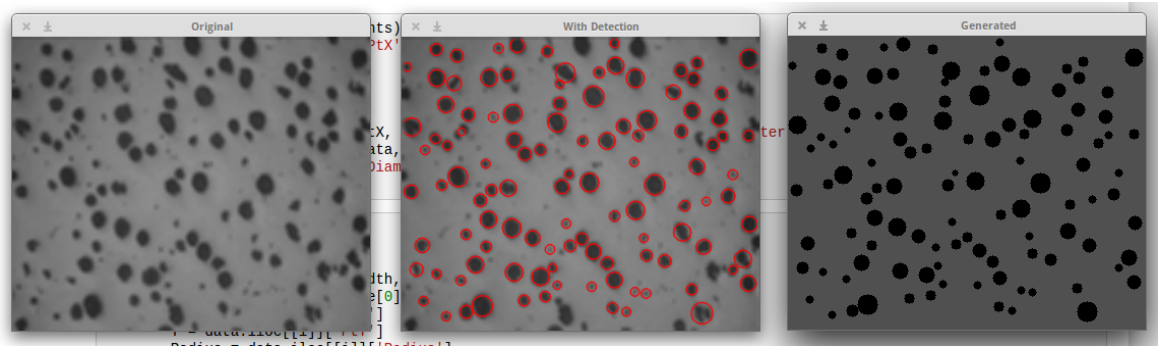


FIGURE 3.11: Image Data Extraction Sample

## 3.5 Machine Learning

Before the development of a predictive model the first step is to understand the data format and representation, perform necessary preprocessing to ensure the quality of the data before modeling, and remove or appropriately deal with noise, outliers, missing values, duplicate data instances, etc.
The next step involves applying machine learning model on the processed data. A data-driven model can, in principle, memorize every single instance of the dataset and thus result in a 100% accuracy on the same data, but wont be able to work well on unseen data values. Thus, tuning of models is a really important part of the procedure.
Techniques like train-test split and cross validation are used to accomplish this task. During validation an appropriate evaluation metric is chosen to check the predictive accuracy of the model.

### 3.5.1 Data Preprocessing and Transformation

The data we generally get is not in proper format. It needs to be processed and transformed before proceeding to model training part. Following are the commonly used data preprocessing steps:

- **Formatting**: The selected data may not be in a format that is suitable to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and should be in a relational database or a text file.
- **Cleaning**: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data that is needed to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.
- **Sampling**: There may be far more selected data available than needed. More data can result in much longer running times for algorithms and larger computational and memory requirements. One can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

After processing data, there may be need of transforming it based on the machine learning model we are planning to use or on the basis of specific problem domain we are working on. Following are the commonly used data transformation techniques:

- **Scaling**: The preprocessed data may contain attributes with a mixtures of scales for various quantities such as dollars, kilograms and sales volume. Many machine learning methods like data attributes to have the same scale such as between 0 and 1 for the smallest and largest value for a given feature.
- **Decomposition**: There may be features that represent a complex concept that may be more useful to a machine learning method when split into the constituent parts. An example is a date that may have day and time components that in turn could be split out further. Perhaps only the hour of day is relevant to the problem being solved. consider what feature decompositions can be performed.
- **Aggregation**: There may be features that can be aggregated into a single feature that would be more meaningful to the problem you are trying to solve. For example, there may be a data instances for each time a customer logged into a system that could be aggregated into a count for the number of logins allowing the additional instances to be discarded. Consider what type of feature aggregations could be performed.

The above-mentioned techniques also depend directly on the machine learning model we are planning to use. So, preprocessing data, transforming data, applying machine learning model and tuning it based on evaluation metric is an iterative process. The steps we followed on our dataset:

1. The data gathered in data generation part was in .dat format and is to be converted into .csv format. The code was written to automate this process on each of the datasets.
2. The .csv format data was loaded on the system in pandas data frame format. Loading this dataset in this format gives us the power to use in-built functions of pandas making the process of data preprocessing easier.
3. Different data frame was aggregated into a master data frame for processing.
4. Data was cleaned to remove outlier points and missing points and proper scaling of features was done.

5. Graphs were plotted to find the feature-feature relationship and feature-target relationship.

6. Based on these graphs, conclusion about the correlation and dependency was drawn and noted.

### 3.5.2 Algorithms

Once appropriate data preprocessing has been performed and the data are ready for modeling, one can employ supervised data mining techniques for predictive modeling. Caution needs to be exercised here to appropriately split the data into training and testing sets (or use cross validation), else the model may be prone to overfitting and show over optimistic accuracy. If the target attribute is numeric (e.g., fatigue strength, formation energy) regression techniques can be used for predictive modeling and if it is categorical (e.g., whether a compound is metallic or not), classification techniques can be used. Some techniques are capable of doing both classification and regression. There also exist several ensemble learning techniques that combine the results from base learners in different ways, and have shown to improve accuracy and robustness of the model in some cases. Apart from predictive modeling, one can also use other data mining techniques such as clustering and relationship mining depending on the goal of the project, for instance, to find group similar materials or discovering hidden patterns and associations in the data.

Our problem was a regression problem. We experimented with various algorithms before finally selecting the three models giving us the best result. The 3 models are explained in detail in the following sections.

### 3.5.3 Random Forest Regressor

The random forest is a bagging algorithm that works on the principle of Ensemble approach. Ensembles are a divide-and-conquer approach used to improve performance. The main principle behind ensemble methods is that a group of weak learners can come together to form a strong learner. The figure below provides an example. Each classifier, individually, is a weak learner, while all the classifiers taken together are a strong learner.

The data to be modeled are the blue circles. We assume that they represent some underlying function plus noise. Each individual learner is shown as a gray curve. Each gray curve (a weak learner) is a fair approximation to the underlying data. The red curve (the ensemble strong learner) can be seen to be a much better approximation to the underlying data.

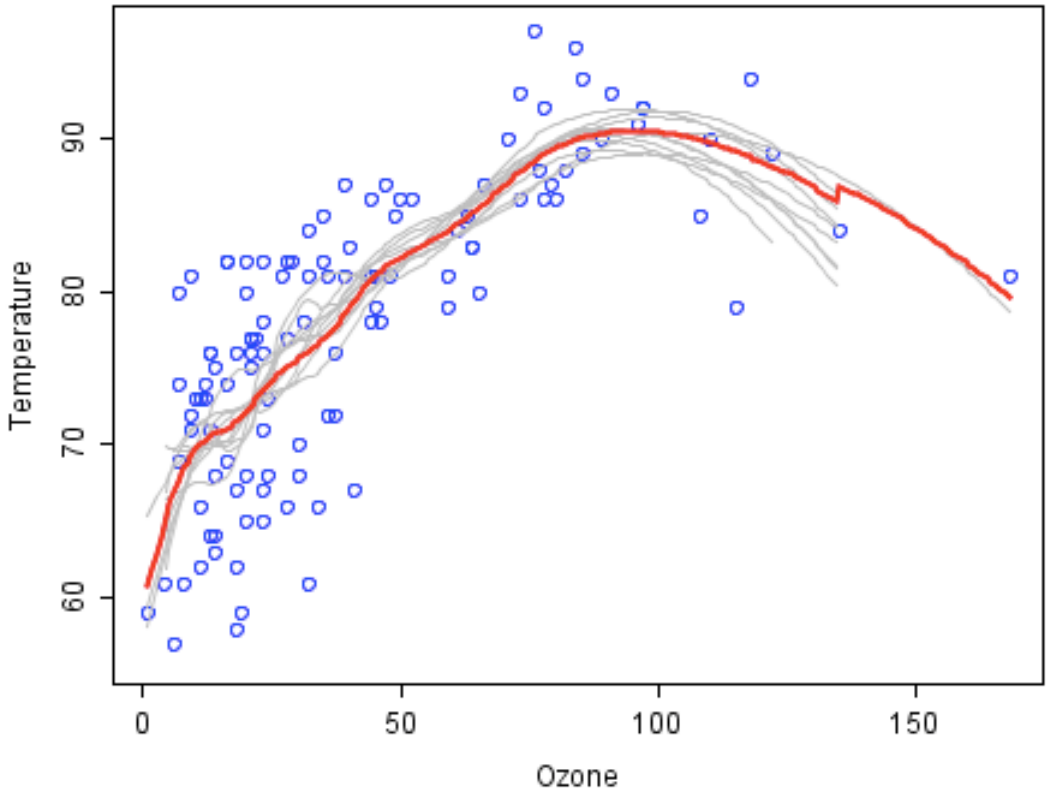


FIGURE 3.12: Random Forrest — Variation of Temperature

The random forest starts with a standard machine learning technique called a decision tree which, in ensemble terms, corresponds to our weak learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets bucketed into smaller and smaller sets.

In this example, the tree advises us, based upon weather conditions, whether to play ball. For example, if the outlook is sunny and the humidity is less than or equal to 70, then its probably OK to play.

The random forest takes this notion to the next level by combining trees with the notion of an ensemble. Thus, in ensemble terms, the trees are weak learners and the random forest is a strong learner.

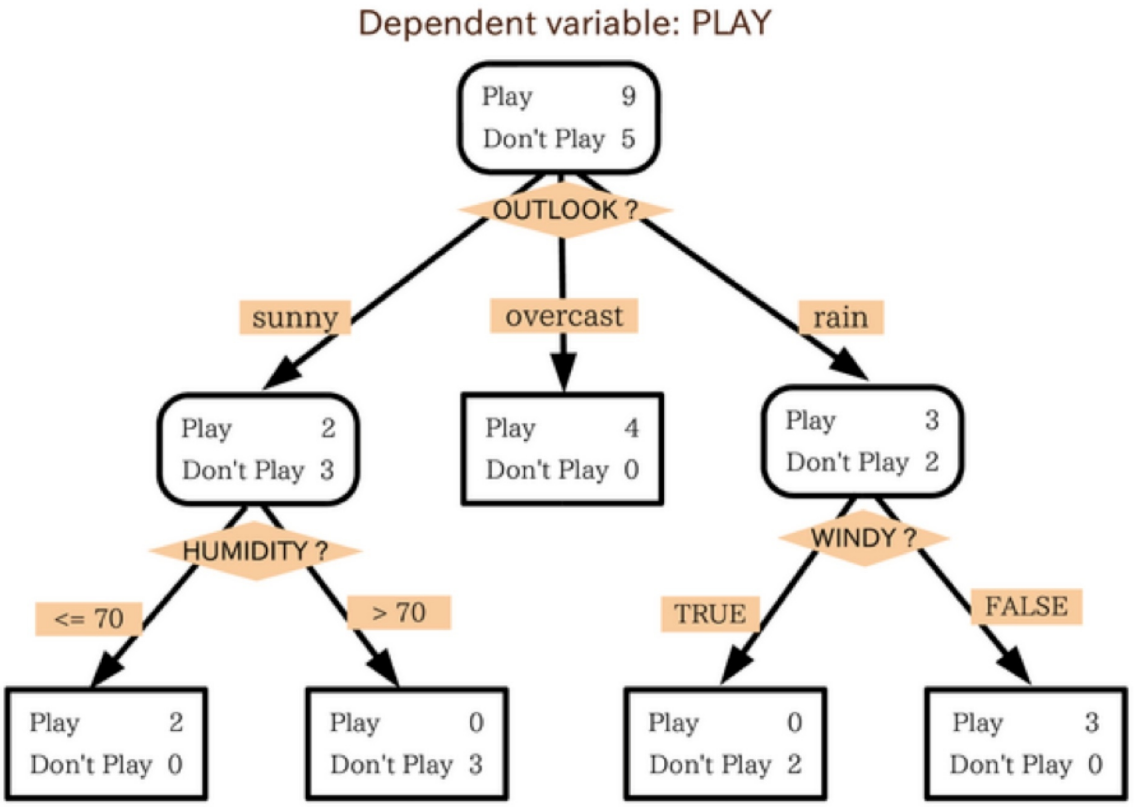


FIGURE 3.13: Random Forrest — Decision Tree

Here is how such a system is trained; for some number of trees T:

- Sample N cases at random with replacement to create a subset of the data. The subset should be about 66% of the total set.
- 2. At each node:
  - For some number m, m predictor variables are selected at random from all the predictor variables.
  - The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.
  - At the next node, choose another m variables at random from all predictor variables and do the same.

The Random Forest algorithm functions almost similarly for both classification and regression. The difference comes in the objective function that the algorithm is trying to optimize and for final prediction in the end, in classification voting is done and in regression averaging is done.

### 3.5.4 XGBoost Regressor

XGBoost is short for Extreme Gradient Boosting. XGBoost works using the same decision tree as explained in Random Forest. The difference between XGBoost and random forest comes in the way both algorithm optimizes the objective function. As explained Random forest is a bagging algorithm works by averaging the result of many decision tree, XGBoost is a boosting algorithm, which works on the principle of sequential learning.

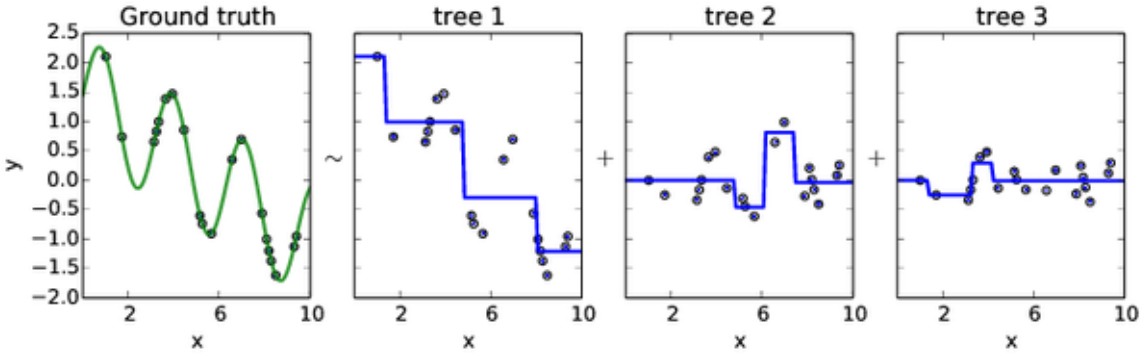


FIGURE 3.14: XGBoost

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weights data points that are hard to predict. Gradient boosting is an approach where new models are created that predicts the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

Extreme Gradient Boosting (XGBoost) is similar to gradient boosting framework but more efficient. Specifically, XGBoost used more regularised model formalization to control overfitting, which gives it better performance, which is why its also known as regularized boosting technique. It has both linear model solver and tree learning algorithms. Moreover, it has the capacity to do parallel computation on a single machine. This makes XGBoost at least 10 times faster than existing gradient boosting implementations. It supports various objective functions, including regression, classification and ranking.

### 3.5.5 Ensemble Averaging

In machine learning, ensemble averaging is the process of creating multiple models and combining them to produce a desired output, as opposed to creating just one model. Frequently an ensemble of models performs better than any individual model, because the various errors of the models "average out." Ensemble averaging can be modified further to do weighted average of models. Generally model with low bias and high variance are averaged to reduce the variance of the model. Each of the models is trained separately on the data and based on the objective function appropriate weights for each of the models is calculated. The final prediction is calculated using prediction of each of the model and the calculated weight. The risk of overfitting the data reduces significantly by averaging the model.

### 3.5.6 Error Metric

Predictive Modeling works on constructive feedback principle. You build a model. Get feedback from metrics, make improvements and continue until you achieve a desirable accuracy. Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results. Hence, selecting a proper evaluation metric is an important part of the learning process. The commonly used metrics for regression are:

- The Mean Absolute Error (or MAE) is the sum of the absolute differences between predictions and actual values. The measure gives an idea of the magnitude of the error, but no idea of the direction (e.g. over or under predicting).
- The Mean Squared Error (or MSE) is much like the mean absolute error in that it provides a gross idea of the magnitude of error. Taking the square root of the mean squared error converts the units back to the original units of the output variable and can be meaningful for description and presentation. This is called the Root Mean Squared Error (or RMSE).
- The $R^2$ (or R Squared) metric provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature, this measure is called the coefficient of determination. This is a value between 0 and 1 for no-fit and perfect fit respectively.

### 3.5.7 Algorithms Applied

During the project, the processed data was there after used to train the model. Different models and evaluation metrics were evaluated to check the suitability of the accuracy of result. Generalized framework followed was:

- Split the data into two parts: train data and test data.
- Train the model on train data and predict the results for the test data.
- Check accuracy of the test data prediction using selected evaluation metric.

This framework was iteratively followed to tune the hyper parameters of the data and to select the final model for prediction. XGBoost Regressor, Random Forest Regressor and weighted average model of XGBoost and Random Forest were the best performing models.

For training the models, the following parameters were used:

1. Features:
    - Youngs Modulus of Matrix ($E_m$)
    - Poissons Ratio of Matrix ($\nu_m$)
    - Youngs Modulus of Fibre ($E_f$)
    - Poissons Ratio of Fibre ($\nu_f$)
    - Volume Fraction ($\phi$) - obtained from image or set to $\{0.3, 0.4, 0.5\}$ during generation from FEM
    - Tangent Moduli (from FEM)
2. Target:
    - Tangent Moduli ($D_{1111}$, $D_{2222}$, $D_{1212}$)
3. After obtaining the tangent moduli, apply the formulae in equations 3.6-3.8 to get:
    - Youngs Modulus of composite ($E_m$)
    - Shear Modulus of composite $D_{1212}$

- Poissons ratio of composite ($\nu_c$)

Formulae for arriving at Young's and Shear Modulus of composite from tangent moduli:

$$x = \frac{2 \times D_{1212}}{D_{1111}} \tag{3.6}$$

$$\nu_c = \frac{x - 1}{x - 2} \tag{3.7}$$

$$E_c = D_{1212} \times 2 \times (1 + \nu_c) \tag{3.8}$$

# Chapter 4

# Analysis and Results

## 4.1 Results for Stress-Strain Curve

The model that was trained on the stress-strain curve data was used to predict values (stresses) on unknown composite materials for every volume fraction. This, in turn, was used to construct the corresponding stress-strain curve. A comparison has been made between the actual and predicted stress-strain curves for a particular material at different volume fractions. The resulting graphs are as follows:

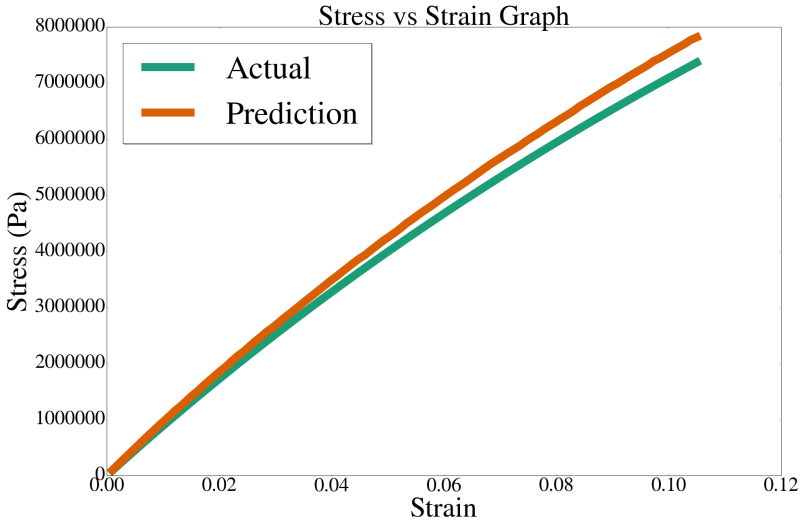


FIGURE 4.1: Stress-Strain Distribution at 5% Volume Fraction

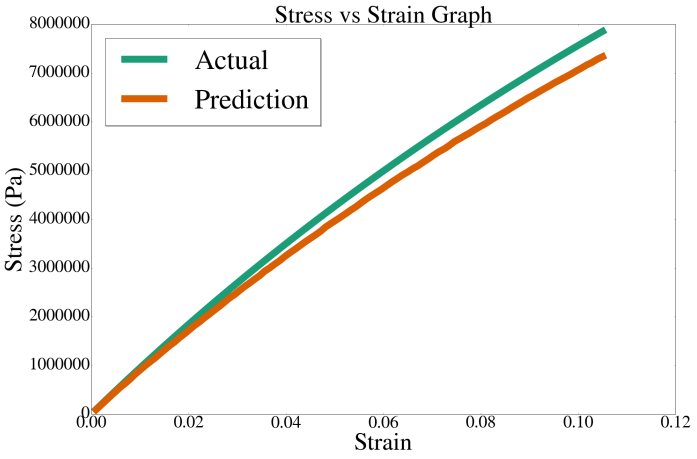


FIGURE 4.2: Stress-Strain Distribution at 10% Volume Fraction

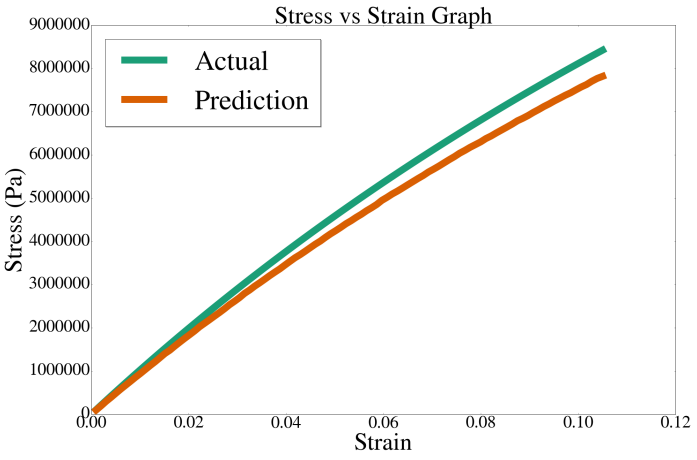


FIGURE 4.3: Stress-Strain Distribution at 15% Volume Fraction

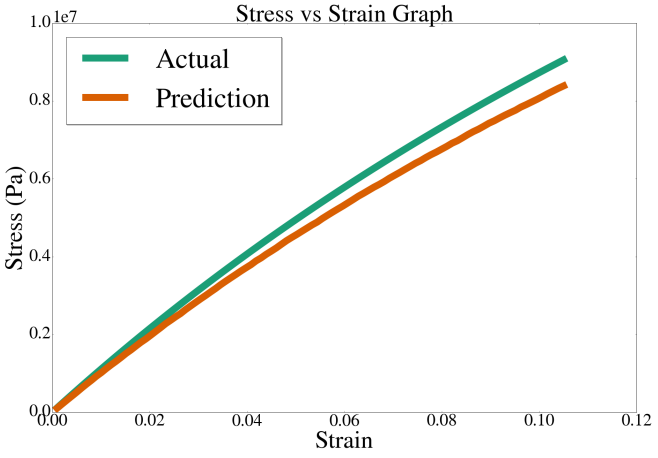


FIGURE 4.4: Stress-Strain Distribution at 20% Volume Fraction

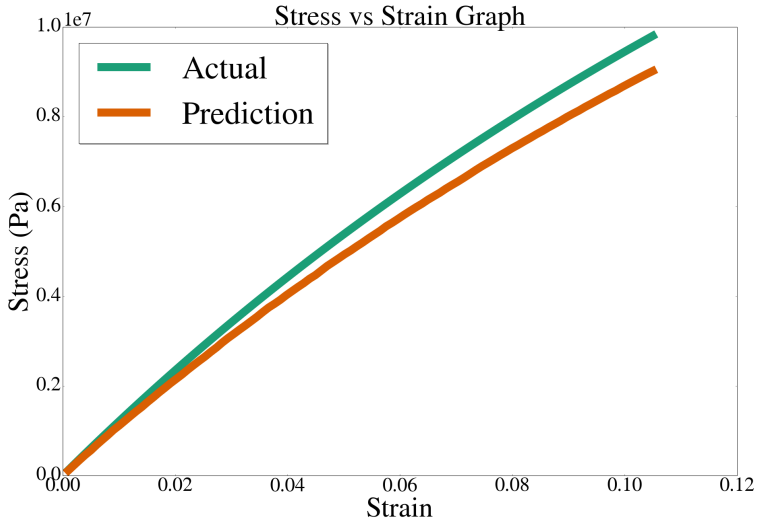


FIGURE 4.5: Stress-Strain Distribution at 25% Volume Fraction

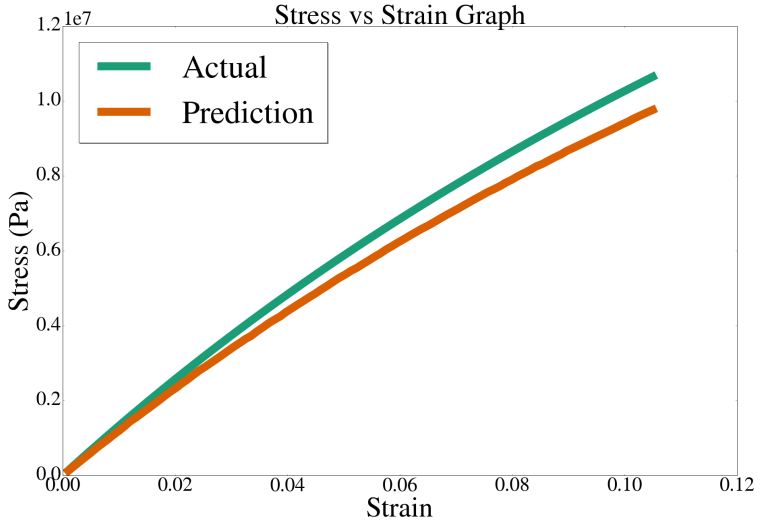


FIGURE 4.6: Stress-Strain Distribution at 30% Volume Fraction

TABLE 4.1: Accuracy of Different ML Models

| Volume Fraction (%) | Coefficient of Determination (R2) |
|---|---|
| 5 | 0.95 |
| 10 | 0.96 |
| 15 | 0.95 |
| 20 | 0.99 |
| 25 | 0.97 |
| 30 | 0.96 |

From the above graphs and corresponding R2 values, it is clear that the ML model works extremely well on unseen data and is able to capture the trend of the stress-strain curve satisfactorily.

## 4.2 Model trained on RSA Data

The trained model was evaluated againsta Finite Element Method using two composites Glass-Epoxy and Boron-Epoxy. Properties of the composites and corresponding results for FEM and Model Predictions are mentioned below.

| S.No. | E_m | nu_m | E_f | nu_f | phi |
|---|---|---|---|---|---|
| 1. | 4.00e+09 | 0.35 | 7.10e+10 | 0.22 | 0.3 |
| 2. | 4.00e+09 | 0.35 | 7.10e+10 | 0.22 | 0.4 |
| 3. | 4.00e+09 | 0.35 | 7.10e+10 | 0.22 | 0.5 |
| 4. | 4.00e+09 | 0.35 | 4.20e+11 | 0.2 | 0.3 |
| 5. | 4.00e+09 | 0.35 | 4.20e+11 | 0.2 | 0.4 |
| 6. | 4.00e+09 | 0.35 | 4.20e+11 | 0.2 | 0.5 |

| S.No. | E_fem | E_rf | E_xgb | E_average |
|---|---|---|---|---|
| 1. | 7.1439e+09 | 6.7000e+09 | 7.2151e+09 | 7.0728e+09 |
| 2. | 8.9053e+09 | 8.3700e+09 | 9.037e+09 | 8.7735e+09 |
| 3. | 11.7873e+09 | 10.9200e+09 | 11.9101e+09 | 11.6646e+09 |
| 4. | 7.217e+09 | 7.7090e+09 | 7.2272e+09 | 7.2068e+09 |
| 5. | 9.2233e+09 | 9.1600e+09 | 9.1192e+09 | 9.3273e+09 |
| 6. | 12.9028e+09 | 12.9100e+09 | 12.5892e+09 | 13.2164e+09 |

FIGURE 4.7: Properties and Results of Sample Composites

The Results are plotted in the following graphs:

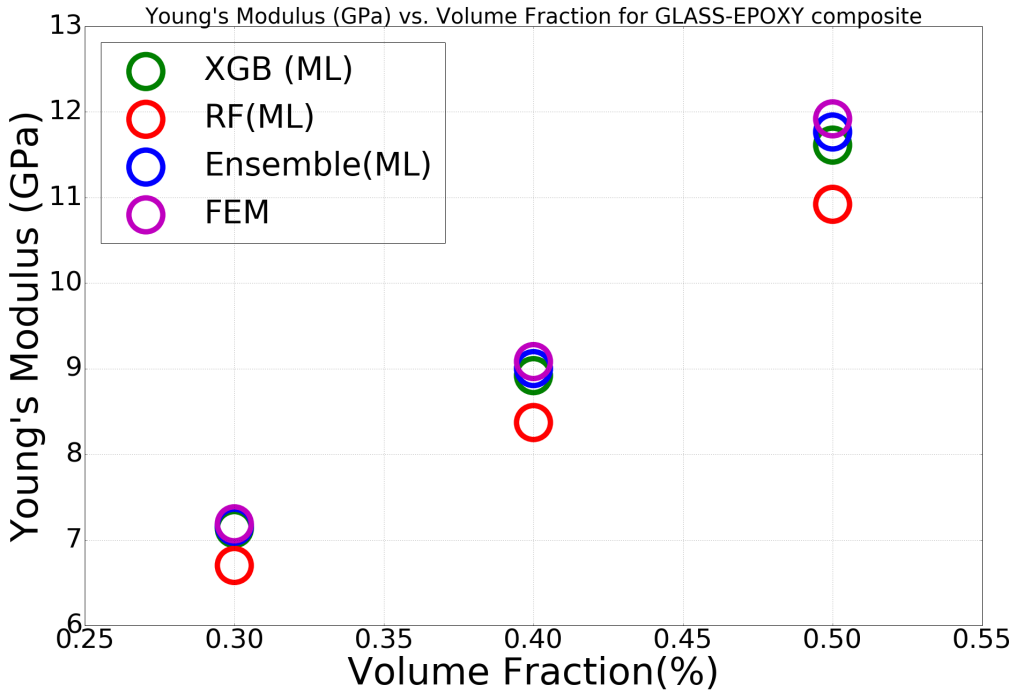


FIGURE 4.8: Young's Modulus Predictions wrt Vol. Fraction (Glass-Epoxy)

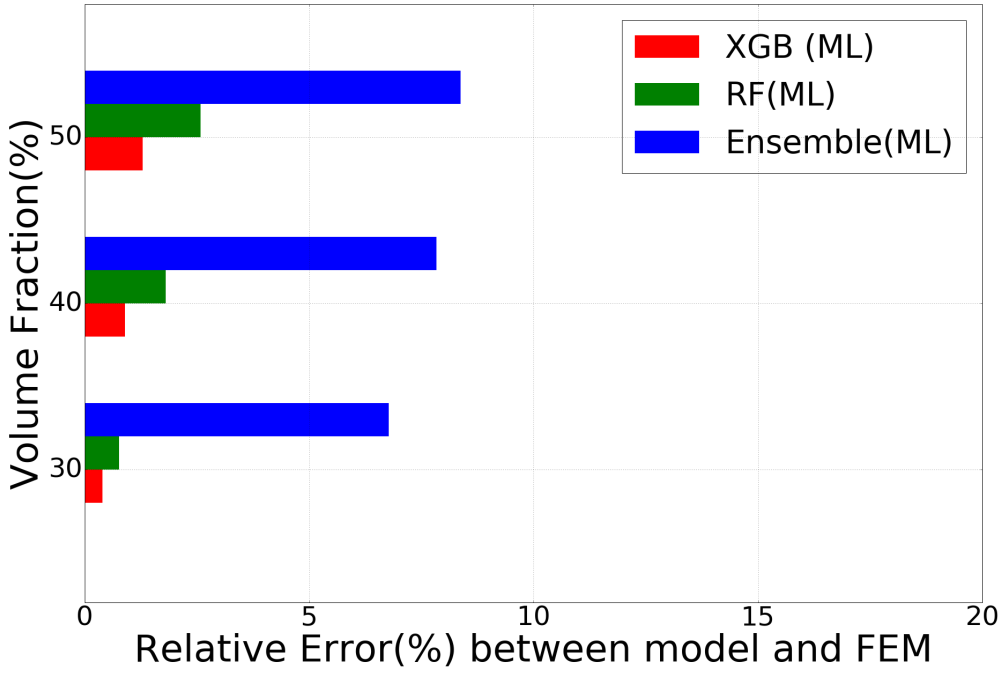


FIGURE 4.9: Relative Model Performance (Glass-Epoxy)

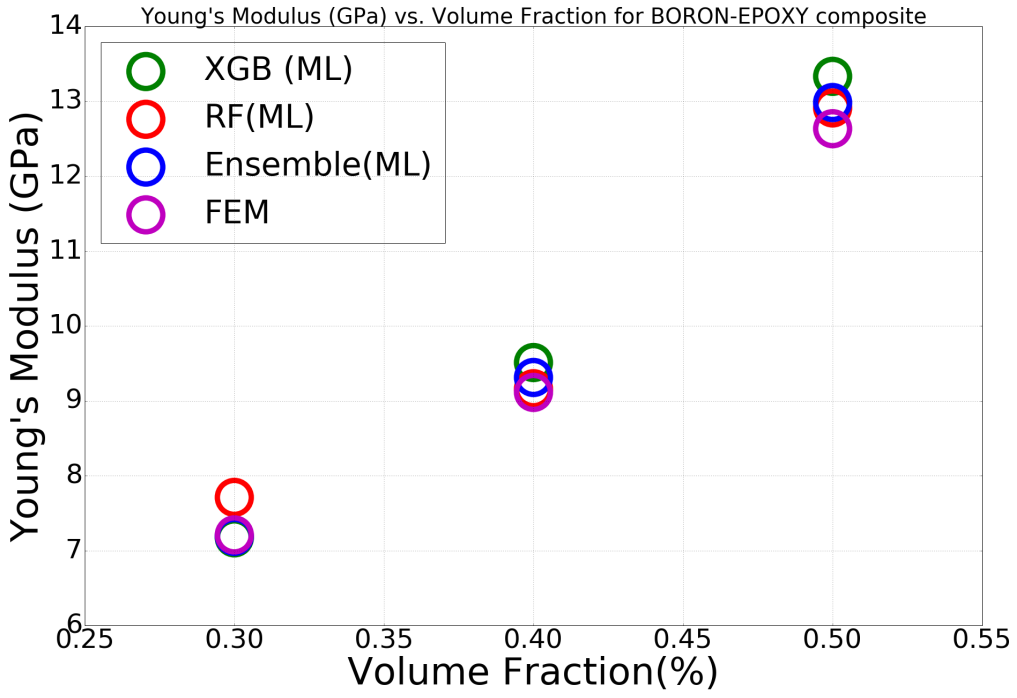


FIGURE 4.10: Young's Modulus Predictions wrt Vol. Fraction (Boron-Epoxy)

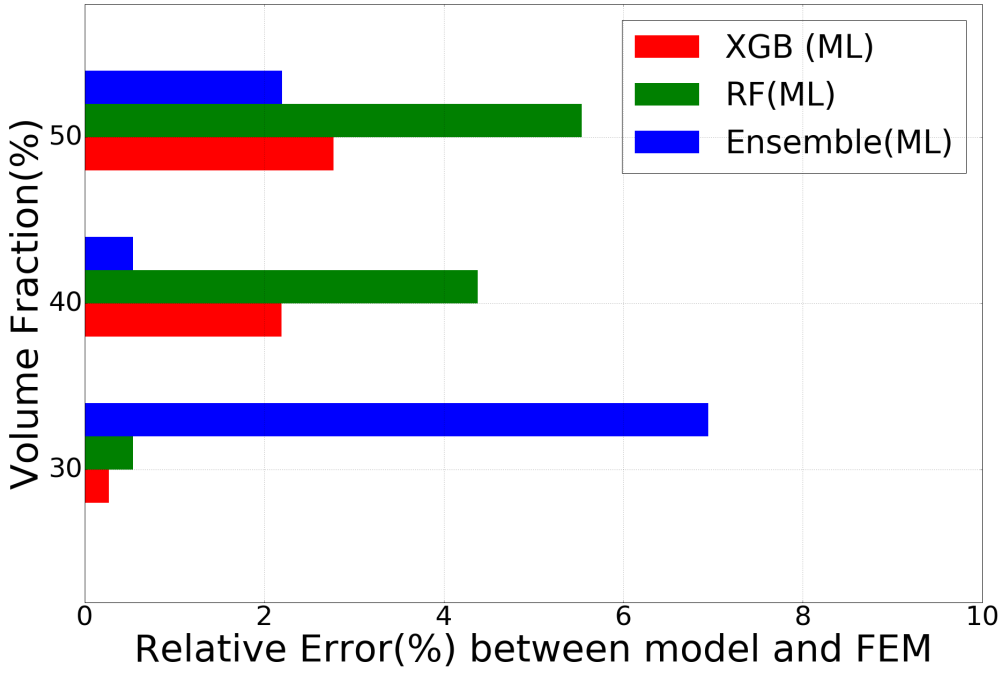


FIGURE 4.11: Young's Modulus Prediction Comparison (Boron-Epoxy)

# Chapter 5

# Conclusion

## 5.1 Findings of the Present Work

The present study was focused broadly on prediction of properties for new unseen materials with varying material properties. Towards this end, this project made variations in Youngs Moduli, Poissons Ratio, and volume fractions of the fibres. The following were the observations made from the analysis:

- A data set comprising nearly 300 different types of materials, when fed to the Machine Learning model, is satisfactorily sufficient in predicting properties of virtually all materials in the range of the data.
- Ensembles of several different Machine Learning models tend to perform better than more powerful single models in capturing trends in the data. Furthermore, it also helps in reducing overfitting and building a more robust model.
- A Machine Learning approach for constructing the stress-strain curve of a new composite material gives promising results on different materials with different volume fractions, provided the strain is in the range of the data.

## 5.2 Future Scope

So far, the scope of the project was limited to variation in properties of constituent materials of a composite and the volume fraction of fibres. Upon observing the level of accuracy of the results, the following extensions can be made to the scope of the project:

- Variation in several other properties of materials (like physical properties), debonding of particles/fibres, and matrix failure (which can be used to calculate toughness and strength) can be introduced.
- A more comprehensive database containing characteristic values as well as information on the manufacturing process of a material, shape, size, and test conditions of a test sample, can be constructed and organized to enable physical, chemical, and engineering fundamental data to be used in industry.
- Development of faster and automated techniques for generation of composite materials with high volume fractions.
- The analysis can be extended to particles/fibres of different diameters.
- The analysis can be extended to 3-D with the introduction of particles. However, it would be significantly more computationally expensive.

# Appendix A

# Code Snippets

## A.1 Model for Data with Varying Vol. Fractions

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

from matplotlib import rcParams

# colorbrewer2 Dark2 qualitative color table
dark2_colors = [(0.10588235294117647, 0.6196078431372549, 0.4666666666666667),
                (0.8509803921568627, 0.37254901960784315, 0.00784313725490196),
                (0.4588235294117647, 0.4392156862745098, 0.7019607843137254),
                (0.9058823529411765, 0.1607843137254902, 0.5411764705882353),
                (0.4, 0.6509803921568628, 0.11764705882352941),
                (0.9019607843137255, 0.6705882352941176, 0.00784313725490196),
                (0.6509803921568628, 0.4627450980392157, 0.11372549019607843)]

rcParams['figure.figsize'] = (10, 6)
rcParams['figure.dpi'] = 150
rcParams['axes.color_cycle'] = dark2_colors
rcParams['lines.linewidth'] = 2
rcParams['axes.facecolor'] = 'white'
rcParams['font.size'] = 14
rcParams['patch.edgecolor'] = 'white'
rcParams['patch.facecolor'] = dark2_colors[0]
rcParams['font.family'] = 'StixGeneral'
features = pd.read_csv('../data/Paticlulate Composite.csv',
delimiter=',', header=None)

# Import All Train Data

strain = pd.DataFrame(strain.values.reshape(100,1))

stress_vf5 = pd.DataFrame(stress_vf5.values.reshape(10000,1))
```

```
stress_vf10 = pd.DataFrame(stress_vf10.values.reshape(10000,1))
stress_vf15 = pd.DataFrame(stress_vf15.values.reshape(10000,1))
stress_vf20 = pd.DataFrame(stress_vf20.values.reshape(10000,1))
stress_vf25 = pd.DataFrame(stress_vf25.values.reshape(10000,1))
stress_vf30 = pd.DataFrame(stress_vf30.values.reshape(10000,1))

features.columns = ['e_m', 'nu_m', 'e_p', 'nu_p']
features.head()

features[0] = features.index
features.columns = ['e_m', 'nu_m', 'e_p', 'nu_p', 'merge_on']

strain['merge_on'] = strain.index
strain.columns = ['strain', 'merge_on']

def create_merging_column(df):
    df['merge_on'] = df.index
    df['merge_on'] /= 100
    df['merge_on'] = stress_vf5['merge_on'].astype(int)
    df.columns = ['stress', 'merge_on']
    return df
stress_vf5 = create_merging_column(stress_vf5)
stress_vf10 = create_merging_column(stress_vf10)
stress_vf15 = create_merging_column(stress_vf15)
stress_vf20 = create_merging_column(stress_vf20)
stress_vf25 = create_merging_column(stress_vf25)
stress_vf30 = create_merging_column(stress_vf30)

def merge_stress_features(df):
    df['merge_on'] = df.index%100
    df = pd.merge(df, strain, on='merge_on', how='left')
    df.drop('merge_on', axis=1, inplace=True)
    return df
ata_vf5['vf'] = 5
data_vf10['vf'] = 10
data_vf15['vf'] = 15
data_vf20['vf'] = 20
data_vf25['vf'] = 25
data_vf30['vf'] = 30

data = pd.concat([data_vf5,data_vf10,data_vf15,data_vf20,data_vf25,data_vf30])

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.cross_validation import cross_val_score
from sklearn.metrics import mean_squared_error

clf = LinearRegression()
print np.mean(cross_val_score(clf,
data.drop('stress', axis=1),
data['stress'], cv = 4, scoring='r2'))

test = data[data['vf'] == 20]
train = data[data['vf'] != 20]
```

```
rf = RandomForestRegressor(n_estimators=200)
rf.fit(train.drop('stress', axis=1), train['stress'])

test['predict'] = rf.predict(test.drop('stress', axis=1))
np.sqrt(mean_squared_error(test['predict'], test['stress']))
import matplotlib.patches as mpatches

# plt.plot(test['strain'],test['stress'], color = 'g')
# plt.plot(test['strain'],test['predict'], color = 'b')
fig, ax = plt.subplots()
ax.plot(to_print['strain'],to_print['stress'],label='Actual')
ax.plot(to_print['strain'],to_print['predict'],label='Prediction')
# Now add the legend with some customizations.
legend = ax.legend(loc='upper left', shadow=True)
# predicted = mpatches.Patch(color='blue', label='Predicted')
# actual = mpatches.Patch(color='green', label='Actual')
# plt.legend(handles=[actual], loc = 'upper left')
plt.title('Stress vs Strain Graph')
plt.xlabel('Strain')
plt.ylabel('Stress (Pa)')
plt.show()
```

## A.2 Calculating Distance Energy

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import linalg as la
import scipy as sp
from matplotlib import rcParams
import matplotlib.cm as cm
import matplotlib as mpl
import random
import math

def generateCoordinates(size):
    X = []
    Y = []
    while len(X) < size:
        X.append(random.randint(1, 4))
        Y.append(random.randint(1, 4))
    plt.scatter(X, Y, s=500)
    return X, Y

def genDistArray (X, Y, size):
    arr = np.zeros(shape=(size,size))
    for i in range(size):
        for j in range(size):
            if (i == j):
                continue
            # print str(i) + " " + str(j)
            arr[i][j] = math.hypot(X[j] - X[i], Y[j] - Y[i])
    return arr

def genDistanceEnergy (arr):
    e_vals, e_vecs = la.eig(arr)
    DistEnergy = 0
    for val in e_vals:
        if val < 0:
            val = -1 * (val)
        DistEnergy = DistEnergy + val
        return DistEnergy

def run (size):
    X, Y = generateCoordinates(size)
    arr = genDistArray(X, Y, size)
    DistanceEnergy = genDistanceEnergy(arr)
    print
    print "Distance Array = "
    print arr
    print
    print "Distance Energy = "
    print DistanceEnergy

run(8)
```

## A.3 Image Data Extraction

```
import numpy as np
import pandas as pd
from pandas import DataFrame, Series
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import math
import cv2
import numpy as np

def initialiseBlobDetector():
    params = cv2.SimpleBlobDetector_Params()
    # Change thresholds
    params.blobColor = 0
    # Create a detector with the parameters
    ver = (cv2.__version__).split('.')
    if int(ver[0]) < 3 :
        detector = cv2.SimpleBlobDetector(params)
    else :
        detector = cv2.SimpleBlobDetector_create(params)
    return detector

def smothenImage(im):
    kernel = np.ones((5,5),np.float32)/22
    dst = cv2.filter2D(im,-1,kernel)
    return dst

def showImages(im, im2, im3):
    cv2.imshow("Original", im)
    cv2.imshow("With Detection",im2);
    cv2.imshow("Generated",im3);
    cv2.waitKey(0)
    cv2.imwrite("Original.BMP", im)
    cv2.imwrite("WithDetection.BMP", im2)
    cv2.imwrite("Generated.BMP", im3)

def showDetection(im, keypoints, keypointsData):
    print "Number of Blobs : " + str(len(keypoints))
    # Show keypoints
    im2 = createImageCopy(im, keypointsData)
    showImages(im, im_with_keypoints, im2)

def createImageCopy(im, data):
    im2 = im.copy()
    height, width = im.shape
    cv2.rectangle(im2,(0,0),(width,height),(80,80,81),-1)
    for i in range(0, data.shape[0]):
        X = data.iloc[[i]]['PtX']
        Y = data.iloc[[i]]['PtY']
        Radius = data.iloc[[i]]['Radius']
        cv2.circle(im2,(X,Y), Radius, (0,0,0), -1)
    return im2
```

```
im = cv2.imread("1002.BMP", cv2.IMREAD_GRAYSCALE)
im = smothenImage(im)
detector = initialiseBlobDetector()

keypoints = detector.detect(im)
keypointsData = extractKeypointsData(keypoints)
height, width = im.shape
area = height*width
keypointsData['AreaOfBlobs'] = np.pi*keypointsData['Radius']**2
volumeFraction = keypointsData['AreaOfBlobs'].sum() / area
print 'Volume Fraction of the input Composite image is ' + str(volumeFraction)
```

## A.4 Random Particle Generation

```
function h = circle(x,y,r,Lx,Ly)
hold on
th = 0:pi/50:2*pi;
xunit = r * cos(th) + x;
yunit = r * sin(th) + y;
h = plot(xunit, yunit);
patch(xunit, yunit,'r','EdgeColor',[1 1 1]);
hold on
clc;
clear all;
clf;
Lx=1;
Ly=1;

Divx=200;
Divy=200;


Porosity=30;
Order=10;
DistHoles=1.5;
Tol_edge=0.01;

AreaHole=Lx*Ly*Porosity/100;
Lxhole=sqrt(AreaHole);
Lyhole=sqrt(AreaHole);

% Lenx=sort(Lxhole*abs(rand(Order,1)));
% Leny=sort(Lyhole*abs(rand(Order,1)));

Lenx=(0:Lxhole/Order:Lxhole);
Leny=(0:Lyhole/Order:Lyhole);

Lenx(1)=0;
Lenx(Order+1)=Lxhole;
Leny(1)=0;
Leny(Order+1)=Lyhole;
```

```
for i=1:Order
    for j=1:Order
        Area(i,j)=(Lenx(j+1)-Lenx(j))*(Leny(i+1)-Leny(i));
        Radius(i,j)=sqrt(Area(i,j)/pi);
    end
end

for j=1:Order*Order
    for i=1:Order*Order-1
        ai=floor(i/Order)+1;
        bi=mod(i,Order);
        if(mod(i,Order)==0)
            bi=Order;
            ai=ai-1;
        end
        ai1=floor((i+1)/Order)+1;
        bi1=mod((i+1),Order);
        if(mod(i+1,Order)==0)
            bi1=Order;
            ai1=ai1-1;
        end
        if (Radius(ai,bi)<Radius(ai1,bi1))
            tmp=Radius(ai,bi);
            Radius(ai,bi)=Radius(ai1,bi1);
            Radius(ai1,bi1)=tmp;
        end
    end
end

count=1;
for i=1:Order
    for j=1:Order
        centre(count,:)=[rand*Lx rand*Ly i j];
        flag=1;
        flag1=0;
        while(flag==1)
            flag=0;
            for k=1:count-1
                if(dist<=DH || dist2<=DH || dist3<=DH)
                     flag=1;
                end
            end
            if((Radius(centre(count,3),centre(count,4)))+centre(count,1)>Lx )
                if (Radius(centre(count,3),centre(count,4))+centre(count,2)>Ly)
                flag1=1;
            end
            end
            if(flag==1)
                centre(count,:)=[rand*Lx rand*Ly i j];
            end
            % if the hole overlapping and border distance conditions are not
            % satisfied a new hole position is selected.
        end
```

```
        if(flag1==1)

            if(Radius(centre(count,3),centre(count,4))+centre(count,1)>Lx)
                centre(count+1,:)=centre(count,:);
                centre(count+2,:)=centre(count,:);
                centre(count+3,:)=centre(count,:);
                centre(count+1,1)=centre(count,1)-Lx;
                centre(count+2,2)=centre(count,2)-Ly;
                centre(count+3,1)=centre(count,1)-Lx;
                centre(count+3,2)=centre(count,2)-Ly;
                count=count+3;

                %if hole is at a corner then it has to be added to all the
                %other corners
            else
                centre(count+1,:)=centre(count,:);
                if(Radius(centre(count,3),centre(count,4))+centre(count,1)>Lx)
                    centre(count+1,1)=centre(count,1)-Lx;
                end
                if(Radius(centre(count,3),centre(count,4))+centre(count,2)>Ly)
                    centre(count+1,2)=centre(count,2)-Ly;
                end
                count=count+1;
                % if a hole is at an edge, it is added to the other
                % edge only
            end
        end
        %Condition to add a hole on the opposite side if a hole crosses the
        %domain boundary
        count=count+1;
    end
end

figure(1)
count=1;
for i=1:length(centre)
    circle(centre(i,1),centre(i,2),Radius(centre(i,3),centre(i,4)));
end
%Plot the holes and domain

hold on;
rectangle('Position',[0 Ly Lx 0.1*Ly],'facecolor',[1 1 1],'edgecolor',[1 1 1]);
rectangle('Position',[0 0 Lx Ly],'edgecolor',[0 0 0]);
hold off;
% Erase of the leftover portion of holes at the edges by using rectangles

Porositycheck=sum(sum(Area))/(Lx*Ly)*100;

nnx= Divx+1;%input('Enter the number of nodes in x-direction :-');
nny= Divy+1;%input('Enter the number of nodes in y-direction :-');
%filename='/home/asingh/Research/Diffusion2D_Petsc/tension/tension'
for count=1;
    for j=1:nny-1;
        for i=1:nnx-1;
            kN=[i+nnx*(j-1) i+nnx*(j-1)+1 nnx+i+1+nnx*(j-1) nnx+i+nnx*(j-1)];
```

```
            CN(count,:)=kN; % storing connectivity values for each element
            count=count+1;
        end
    end
end

numel=length(CN(:,1));
% co-ordinates

a=Lx/(nnx-1);
b=Ly/(nny-1);

for count=1;
for j=0:nny-1;
    for i=0:nnx-1;
        coor=[i*a j*b];
        co_or(count,:)=coor;
        count=count+1;
    end
end
end
numnp=length(co_or(:,1));

for i=1:numel
    nd=CN(i,:);
    xarr=co_or(nd,1);
    yarr=co_or(nd,2);
    xc=sum(xarr)/4.0;
    yc=sum(yarr)/4.0;
    elem_centre(i,1)=xc;
    elem_centre(i,2)=yc;
end
Gridx=0:Lx/(Divx):Lx;
Gridy=0:Ly/(Divy):Ly;
coord=zeros(Divx*Divx,8);
M=zeros(Divx,Divy);
count=1;
numcircle=length(centre(:,1));
CellDataArray(1:numel)=0.0;
for i=1:numel
        xc=elem_centre(i,1);
        yc=elem_centre(i,2);

        for k=1:numcircle
            circ_cen_x=centre(k,1);
            circ_cen_y=centre(k,2);
            rad=sqrt((xc-circ_cen_x)^2+(yc-circ_cen_y)^2);

            if(rad<(Radius(centre(k,3),centre(k,4))))
             CellDataArray(i)=1.0;
            end
        end
end

Cell_Fibre=reshape(CellDataArray,[Divx,Divy]);
```

```
Cell_Matrix=1.0-Cell_Fibre;

FFT_Cell_Fibre=fft2(Cell_Fibre);
Conjugate_FFT_Cell_Fibre=conj(FFT_Cell_Fibre);

FFT_Cell_Matrix=fft2(Cell_Matrix);
Conjugate_FFT_Cell_Matrix=conj(FFT_Cell_Matrix);
% S_fm
figure(2)
ISfm=FFT_Cell_Fibre*Conjugate_FFT_Cell_Matrix;
Sfm=real(ifft2(ISfm));
surf(Sfm);
legend('S_{fm}');

% S_mm
figure(3)
ISmm=FFT_Cell_Matrix*Conjugate_FFT_Cell_Matrix;
Smm=real(ifft2(ISmm));
surf(Smm);
legend('S_{mm}');

% S_mf
figure(4)
ISmf=FFT_Cell_Matrix*Conjugate_FFT_Cell_Fibre;
Smf=real(ifft2(ISmf));
surf(Smf);
legend('S_{mf}');

%S_ff
figure(5)
ISff=FFT_Cell_Fibre*Conjugate_FFT_Cell_Fibre;
Sff=real(ifft2(ISff));
surf(Sff);
legend('S_{ff}');

dat=fopen('CellIndexData.dat','wb');

for k=1:numel;
 fprintf(dat,'%4d %7f %7f  %4d\n',k, elem_centre(k,1:2), CellDataArray(k));
end
```

## A.5 Predicting composite properties with ML Models

```
import numpy as np
import pandas as pd
from fnmatch import fnmatch
import matplotlib.pyplot as plt
from matplotlib import rcParams

#these colors come from colorbrewer2.org. Each is an RGB triplet
dark2_colors = [(0.10588235294117647, 0.6196078431372549, 0.4666666666666667),
                (0.8509803921568627, 0.37254901960784315, 0.00784313725490196),
                (0.4588235294117647, 0.4392156862745098, 0.7019607843137254),
                (0.9058823529411765, 0.1607843137254902, 0.5411764705882353),
                (0.4, 0.6509803921568628, 0.11764705882352941),
                (0.9019607843137255, 0.6705882352941176, 0.00784313725490196),
                (0.6509803921568628, 0.4627450980392157, 0.11372549019607843),
                (0.4, 0.4, 0.4)]

rcParams['figure.figsize'] = (20, 10)
rcParams['figure.dpi'] = 150
rcParams['axes.color_cycle'] = dark2_colors
rcParams['lines.linewidth'] = 2
rcParams['axes.grid'] = True
rcParams['axes.facecolor'] = '#eeeeee'
rcParams['font.size'] = 14
rcParams['patch.edgecolor'] = 'none'

# Import Required Data

data.drop([0, 5], axis=1, inplace=True)

result[4] = 0

from sklearn.ensemble import RandomForestRegressor
from sklearn.cross_validation import cross_val_score

rf = RandomForestRegressor(n_estimators = 200)
np.mean(cross_val_score(rf, data, result[1], cv = 4))

for i in [1,2,3,5,6,9]:
    print i, np.mean(cross_val_score(rf,data, result[i], cv = 4))

import xgboost as xgb
params = {"objective": "reg:linear",
          "booster" : "gbtree",
          "eta": 0.1,
          "max_depth": 3,
          "subsample": 0.8,
          'gamma':1.0,
          "silent": 1,
          "seed": 1
          }
```

```
num_boost_round = 1000

# Import Data

to_predict2.columns = [1,2,3,4,6]
data.columns = [1,2,3,4,6]

res11 = []
res12 = []
for i in [1,2,3,5,6,9]:
    dtrain = xgb.DMatrix(data, result[i])
    gbm = xgb.train(params, dtrain, num_boost_round)
    pred11 = gbm.predict(xgb.DMatrix(to_predict1))
    pred12 = gbm.predict(xgb.DMatrix(to_predict2))
    print i, pred11
    print i, pred12
    res11.append(pred11)
    res12.append(pred12)

phi11 = 2*res11[5]/res11[0]

phi12 = 2*res12[5]/res12[0]
nu_c11 = (phi11-1)/(phi11-2)
nu_c12 = (phi12-1)/(phi12-2)

E11 = 2*res11[5]*(1+nu_c11)
E12 = 2*res12[5]*(1+nu_c12)

a11 = pd.DataFrame(E11/(1e+09))
a11.columns = ['E']

a12 = pd.DataFrame(E12/(1e+09))
a12.columns = ['E']

a11['phi'] = pd.Series([0.3,0.4,0.5]*3)
a12['phi'] = pd.Series([0.3,0.4,0.5]*3)

res21 = []
res22 = []
rf = RandomForestRegressor(n_estimators = 200)
for i in [1,2,3,5,6,9]:
    rf.fit(data, result[i])
    pred21 = rf.predict(to_predict1)
    pred22 = rf.predict(to_predict2)
    print i, pred21
    print i, pred22
    res21.append(pred21)
    res22.append(pred22)
phi21 = 2*res21[5]/res21[0]
phi22 = 2*res22[5]/res22[0]
nu_c21 = (phi21-1)/(phi21-2)
nu_c22 = (phi22-1)/(phi22-2)

E21 = 2*res21[5]*(1+nu_c21)
E22 = 2*res22[5]*(1+nu_c22)
```

```
a21 = pd.DataFrame(E21/(1e+9))
a21.columns = ['E']
a22 = pd.DataFrame(E22/(1e+9))
a22.columns = ['E']
a21['phi'] = pd.Series([0.3,0.4,0.5]*3)
a22['phi'] = pd.Series([0.3,0.4,0.5]*3)
a31 = pd.DataFrame(a11['phi'])
a31['E'] = (a11['E'] + a21['E'])/2
a31 = a31[['E','phi']]
a32 = pd.DataFrame(a12['phi'])
a32['E'] = (a12['E'] + a22['E'])/2
a32 = a32[['E','phi']]
a31.loc[a31.shape[0]] = [6.7, 0.3]
a31.loc[a31.shape[0]] = [8.37, 0.4]
a31.loc[a31.shape[0]] = [10.92, 0.5]
a32.loc[a32.shape[0]] = [7.709, 0.3]
a32.loc[a32.shape[0]] = [9.16, 0.4]
a32.loc[a32.shape[0]] = [12.91, 0.5]
a1 = pd.concat([a11, a21, a31], ignore_index=True)
a2 = pd.concat([a12, a22, a32], ignore_index=True)


mu11 = pd.DataFrame(a11['phi'])
mu11['mu'] = res11[5]/1e+09
mu11 = mu11[['mu', 'phi']]

mu21 = pd.DataFrame(a21['phi'])
mu21['mu'] = res21[5]/1e+09
mu21 = mu21[['mu', 'phi']]

mu12 = pd.DataFrame(a12['phi'])
mu12['mu'] = res12[5]/1e+09
mu12 = mu12[['mu', 'phi']]

mu22 = pd.DataFrame(a22['phi'])
mu22['mu'] = res22[5]/1e+09
mu22 = mu22[['mu', 'phi']]

mu31 = pd.DataFrame(a11['phi'])
mu31['mu'] = (mu11['mu'] + mu21['mu'])/2
mu31 = mu31[['mu','phi']]

mu32 = pd.DataFrame(a11['phi'])
mu32['mu'] = (mu12['mu'] + mu22['mu'])/2
mu32 = mu32[['mu','phi']]

mu31.loc[mu31.shape[0]] = [2.5, 0.3]
mu31.loc[mu31.shape[0]] = [3.15, 0.4]
mu31.loc[mu31.shape[0]] = [4.14, 0.5]

mu32.loc[mu32.shape[0]] = [2.648, 0.3]
mu32.loc[mu32.shape[0]] = [3.44, 0.4]
mu32.loc[mu32.shape[0]] = [4.87, 0.5]
```

```
mu1 = pd.concat([mu11, mu21, mu31], ignore_index=True)

mu2 = pd.concat([mu12, mu22, mu32], ignore_index=True)
```

# Bibliography


[1] C. Sun and R. Vaidya. Prediction of composite properties from a representative volume element. *Compos. Sci. Technol.*, pages 171–179, 1996.

[2] Suresh S. Brockenbrough, J. and H. Wienecke. Deformation of metal-matrix composites with continuous fibers: Geometrical effects of fiber distribution and shape. *Acta Metall. Mater.*, pages 735–752, 1991.

[3] Saffari P. Ranade R. Sadeghipour K. Sun, C. and G Baran. Finite element analysis of elastic property bounds of a composite with randomly distributed particles. *Compos. Part A Appl. Sci. Manuf.*, pages 80–86, 2007.

[4] Costa J. Mayugo J. Trias, D. and J. Hurtado. Random models versus periodic models for fibre reinforced composites. *Comput. Mater. Sci.*, pages 316–324, 2006.

[5] T. Zhang and Y. Yan. A comparison between random model and periodic model for fiber-reinforced composites based on a new method for generating fiber distributions. *Polym. Compos.*, 2015.

[6] Feder J. Hinrichsen, E.L. and T. Jssang. Geometry of random sequential adsorption. *J. Stat. Phys.*, pages 793–827, 1986.

[7] J. Feder. Random sequential adsorption. *J. Theor. Biol.*, pages 237–254, 1980.

[8] J. Ramsden. Review of new experimental techniques for investigating random sequential adsorption. *J. Stat. Phys.*, pages 853–877, 1993.

[9] Tewari A. Yang, S. and A. M. Gokhale. Modeling of non-uniform spatial arrangement of fibers in a ceramic matrix composite. *Acta Mater.*, pages 3059–3069, 1997.

[10] Pagano N. Kim R. Buryachenko, V. and J. Spowart. Quantitative description and numerical simulation of random microstructures of composites and their effective elastic moduli. *Int. J. Solids Struct.*, pages 47–72, 2003.

[11] A. Wongsto and S. Li. Micromechanical fe analysis of ud fibre-reinforced composites with fibres distributed at random over the transverse cross-section. *Compos. Part A Appl. Sci. Manuf.*, pages 246–1266, 2005.

[12] Wang X. Zhang J. Liang W. Wang, Z. and L. Zhou. Automatic generation of random distribution of fibers in long-fiber-reinforced composites and mesomechanical simulation. *Mater. Des.*, pages 885–891, 2011.

[13] Camanho P. Melro, A. and S. Pinho. Generation of random distribution of fibres in long-fibre reinforced composites. *Compos. Sci. Technol.*, pages 2092–2102, 2008.

[14] T. Vaughan and C. McCarthy. A combined experimental-numerical approach for generating statistically equivalent fibre distributions for high strength laminated composite materials. *Compos. Sci. Technol.*, pages 291–297, 2010.

[15] Chao Zhang Xiaosheng Gao Wenzhi Wang, Yonghui Dai and Meiying Zhao. Micromechanical modeling of fiber-reinforced composites with statistically equivalent random fiber distribution. 2008.

[16] Materials genome initiative for global competitiveness. *OSTP*, pages 2092–2102, 2011.

[17] Zhengzhang Chen Ankit Agrawal Veera Sundararaghavan Ruoqian Liu, Abhishek Kumar and Alok Choudhary. A predictive machine learning approach for microstructure optimization and materials design. *Compos. Sci. Technol.*, pages 2092–2102, 2008.

[18] James Theiler John Hogden Prasanna V. Balachandran, Dezhen Xue and Turab Lookman. Adaptive strategies for materials design using uncertainties.

[19] S. Kirklin J. E. Saal J. W. Doak A. Thompson K. Zhang A. Choudhary B. Meredig, A. Agrawal and C. Wolverton. Combinatorial screening for new materials in unconstrained composition space with machine learning.

[20] Ahmet Cecen Gautham P Basavarsu Alok N Choudhary Ankit Agrawal, Parijat D Deshpande and Surya R Kalidindi. Exploration of data science techniques to predict fatigue strength of steel from composition and processing parameters.

[21] Doreswamy and Hemanth. K. S. Performance evaluation of predictive classifiers for knowledge discovery from engineering materials data sets.

[22] T.J. Vaughan and C.T. McCarthy. A combined experimentalnumerical approach for generating statistically equivalent fibre distributions for high strength laminated composite materials. *Composites Science and Technology*, pages 291–297, 2010.

[23] Z.Q. Zhang H.F. Lei and B. Liu. Effect of fiber arrangement on mechanical properties of short fiber reinforced composites. *Compos. Sci. Technol.*, pages 506–514, 2011.

[24] Ivan Gutmanb Gopalapillai Indulal and Ambat Vijayakumar. On distance energy of graphs. *Communications in Mathematical and in Computer Chemistry*, pages 461–472, 2008.