UNIVERSITY OF CAPE TOWN

Department of Mechanical Engineering

RONDEBOSCH, CAPE TOWN
SOUTH AFRICA

PROJECT No:
124

MEC4128S: FINAL REPORT 2022

Machine-learning clustering approaches for automated processing of stress-strain data in Python

KATLEHO MOFOKENG

MFKKAT007

Project Brief

During processing of stress-strain data, the curves need to be separated into regions, for example the elastic, transition and plastic regions. Test measurements might also contain spurious data, like leading and trailing measurements and various forms of noise. It can be difficult to clean this data in a robust, automated fashion, which is necessary for a large set of experiments.

Machine learning (ML) clustering methods provide a means of identifying groups of data within a given set of test measurements. These algorithms could be used to identify the regions of interest on each stress-strain curve, and to identify spurious noise in data. Automated post-processing could save researchers working in mechanical testing a lot of time and manual labour. Many ML clustering algorithms are readily available in existing Python libraries like scikit-learn. These can be easily implemented in a few lines of code, but need to be trained and optimised to perform specific tasks.

ML clustering algorithms will be selected and trained to perform automated processing on the given stress-strain dataset, before analysing and comparing the success and suitability of each algorithm.

SUPERVISORS: *Ernesto Ismail and Daniel Slater*

Word Count: 6605

Plagiarism declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each significant contribution to, and quotation in, this report/project from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report/project is my own work.
4. I have not allowed, and will not allow anyone to copy my work with the intention of passing it off as his or her own work.

Signature: MFKKAT007

Acknowledgements

I would like to acknowledge the contribution of my two supervisors, Ernesto Ismail and Daniel Slater. They provided me with support and guidance which enabled me to complete my research project.

Abstract

A uniaxial tension test is performed. During this test, the force applied to the specimen is recorded as a function of displacement between the grips of the testing machine. The data collected from the test is used to determine the important material properties such as Young's modulus, yield strength, ultimate tensile strength, elongation at break, Poisson's ratio, and Lankford coefficients. The issue is that the test measurements contain spurious data, like leading and trailing measurements and various forms of noise. Thus, before determining the material properties, the data must be cleaned. It is difficult to clean this data in a robust, automated fashion, which is necessary for a large set of experiments. Therefore, an algorithm should be built that can be trained to learn the behaviour of raw experimental data to identify regions of interest on each stress-strain curve, and to identify spurious noise in data. Automated post-processing will save researchers working in mechanical testing a lot of time and manual labour. The training of the algorithm will be done using the method of supervised learning.

The aim is to use machine learning clustering models to build an algorithm that can be trained to identify regions of interest on a stress strain curve like the elastic, transition and plastic region. But most importantly identify spurious noise in data. These will be done in the following steps: • Performing exploratory data analysis and identify regions of interests. • Comparing how different models cluster the data. • Pick models which you want to optimize based on how they cluster the regions of interest. • Train and optimize the models chosen. • Evaluate the performance of models and choose the model with the best performance.

Machine learning:

Machine learning is a branch of computer science that broadly aims to enable computers to "learn" without being directly programmed. It has origins in the artificial intelligence movement of the 1950s and emphasizes practical objectives and applications, particularly prediction and optimization [1]. The machine learning algorithms are classified into various categories i.e. supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning algorithms [2].

Supervised learning:

Supervised learning is a type of algorithm in machine learning in which both the input and desired output data are provided. As a teacher supervises the learning process in the same way the algorithm learns from the training dataset. We know the correct answers beforehand; the algorithm continuously makes predictions on the training data and is corrected by the teacher [2].

K means clustering:

The aim of this algorithm is to find clusters or groups in the given data, where variable k represents the number of groups. It is an iterative algorithm where each data point is allocated to one of the K groups based on the list of all the features which are given provided [3].

Contents

1. Introduction 4
   1.1 Background and description 4
   1.2 Scope and limitations 5
2. Literature review 6
3. Aims and objectives 12
4. Unsupervised learning 13
   4.1 Selecting a clustering algorithm 13
      4.1.1 Methodology for choosing a clustering algorithm 13
      4.1.2 Discussion 16
   4.2 Test the algorithm 19
      4.2.1 Methodology 19
      4.2.2 Results 22
      4.2.3 Discussion 23
   4.3 Optimize the algorithm and evaluate the performance 23
      4.3.1 Methodology 23
      4.3.2 Results 24
      4.3.3 Discussion 26
5. Supervised learning 26

5.1 Build a clustering algorithm 26
5.1.1 Methodology 26
5.1.2 Results 29
5.1.3 Discussion 30
5.2 Train and test the algorithm 30
5.2.1 Methodology 30
5.2.2 Results 31
5.2.3 Discussion 32
5.3 Optimize the algorithm and evaluate the performance 33
5.3.1 Methodology 33
5.3.2 Results 34
5.3.3 Discussion 35
5.4 Compare the two different clustering methods 35
5.4.1 Methodology 35
5.4.2 Results 36
5.4.3 Discussion 38
6. Project overall discussion 38
7. Recommendation 39
8. Bonus 39
9. Conclusion 39
10. References 44
Appendix 44
Appendix A 44
Appendix B 45
Appendix C 53
Appendix D 70

1. Introduction

1.1 Background and description

Designing and manufacturing are important parts of engineering. Thus, to ensure the safety of a final product mechanical testing has to be done. Mechanical testing is used to mitigate

against unexpected failures and as part of a failure investigation. Mechanical testing can also inform cost-effective design and assist with technological innovation.

During processing of stress-strain data, the curves need to be separated into regions, for example the elastic, transition and plastic regions. Test measurements might also contain spurious data, like leading and trailing measurements and various forms of noise. It can be difficult to clean this data in a robust, automated fashion, which is necessary for a large set of experiments.

This project was born due to the various forms of noise that occur during mechanical testing. Data is collected from multiple tests conducted, it will take a long time to remove all the noise manually. Thus, the project requires one find a method that can be automated to remove noise from all the collected data.

The method that will be explored for this project is machine learning. Because machine learning allows the researcher to feed a computer algorithm an immense amount of data and have the computer analyse and make data-driven recommendations and decisions based on only the input data.

The data is collected from the uniaxial compression strength test. The materials used are the following:

- AC – As Cost
- H560 – Homogenized at 560 °C
- H580 – Homogenized at 580 °C

The tests strain rates are either 10 or 1.

The nominal temperatures used for UC tests are the following:

- 300 °C
- 330 °C
- 360 °C
- 400 °C
- 450 °C
- 500 °C

The initial length of the part is 15 mm and the initial diameter is 10 mm.

Uniaxial compression test information is shown in **appendix A**.

1.2 Scope and limitations

The reason for this research is to use the automated post-processing clustering methods to remove the noise from the collected data and output ‘clean’ data.

Successfully completing this project will save researchers working in mechanical testing a lot of time and manual labour. It will also allow researchers to find parameters such as:

- Yield strength
- Elastic modulus

- Strength at failure

The aim is to use machine learning clustering algorithms that will be selected and trained to perform automated processing on the given stress-strain dataset.

The objective of the project will achieved by following these steps:

- Performing exploratory data analysis
- Identify regions of interests
- Comparing how different models cluster the data
- Pick the algorithm which you want to optimize based on how they cluster the regions of interest
- Optimize the clustering algorithm chosen
- Train the clustering algorithm chosen
- Evaluate the performance of the algorithm
- Choose the algorithm with the best performance

The limitations of the project are the following:

- The project must be completed in six weeks.
- The data used to build an algorithm is only collected from uniaxial compression strength test.
- Only machine learning algorithms will be used
- Only one student working on the project with the assistance of two supervisors.

2. Literature review

Mechanical testing

Mechanical testing is an essential part of any design or manufacturing process. The primary objective of mechanical testing is to ensure product safety, whether through characterising material properties or validating final products [4].

Mechanical testing is a process that is used to determine the mechanical properties of a material. It is used to ensure that supplied materials will perform as expected. It can be used to evaluate a material independent of its geometry and also at defined geometrical conditions. There are a wide variety of mechanical testing methods that can be performed when evaluating a material, some of which may determine the appropriateness of a material subject to corrosion [4].

Mechanical testing covers a wide range of tests, which can be divided broadly into two types:

1. Those that aim to determine a material's mechanical properties, independent of geometry.
2. Those that determine the response of a structure to a given action, e.g. testing of composite beams, aircraft structures to destruction, etc.

There is a wide variety of standardised tests used to determine the different mechanical properties of materials. These material property tests tend to be independent of factors such as machining or geometry and will usually be related to the bulk raw material. However, even in these instances, factors like specimen size can impact certain properties, such as for tensile tests. Because of this, these tests are often also conducted on finished parts and components rather than just base materials [5].

The most common tests can be split into the following types;

- Hardness
- Tensile
- Impact
- Fracture toughness,
- Creep
- Fatigue
- Non-destructive testing

Hardness testing

This test is used to determine the resistance a material exhibits to permanent deformation by penetration of another harder material. Hardness testing includes the following:

- Vickers Hardness Test (HV) (Testing at a wide scale)
- Barcol Testing (Used for composite materials)
- Brinell Hardness Test (HB)
- Knoop Hardness Test (HK) (Testing over small areas)
- Janka Hardness Test (Used for wood)
- Meyer Hardness Test
- Rockwell Hardness Test (HR) (Commonly used in the USA)
- Shore Durometer Test (Used for polymers)

Tensile testing

This test is used to find out how strong a material is and also how much it can be stretched before it breaks. This test method is used to determine the following properties:

- Yield stress,
- Tensile stress,
- Percentage elongation to failure and
- Young's Modulus
- Ultimate tensile strength
- Ductility
- Poisson's ratio
- Strain hardening

Impact testing

Impact tests are used in studying the toughness of material. A material's toughness is a factor of its ability to absorb energy during plastic deformation. Brittle materials have low toughness as a result of the small amount of plastic deformation that they can endure [5].

Impact testing includes the following:

- Izod Testing
- Charpy Testing

Fracture toughness testing

This test determines how well a material can resist the growth of a crack under an increasing load. The fracture toughness values identified by the test are useful in material selection and in determining whether there is a danger of component failure when a flaw is discovered in an existing structure [4].

Fracture toughness testing includes the following:

- Linear-Elastic (KIc)
- K–R Curve
- Elastic Plastic (JIc, CTOD)

Creep testing

This test determines the mechanical behaviour of materials at high temperatures relative to the melting point.

There are three main stages of creep:

- Primary creep - Starts at an increased rate and slows with time due to material hardening.
- Secondary creep - Has a relatively steady rate.
- Tertiary creep - Has an accelerated rate and ends when the material breaks.

Fatigue testing

A fatigue test helps determine a material’s ability to withstand cyclic fatigue loading conditions. By design, a material is selected to meet or exceed service loads that are anticipated in fatigue testing applications [4].

Testing materials under cyclic loading, including:

- Fatigue Crack Growth Testing
- Load-Controlled Smooth Specimen Testing
- Strain-Controlled Smooth Specimen Testing

Non-destructive testing

Non-destructive testing (NDT) is the process of inspecting, testing, or evaluating materials, components or assemblies for discontinuities, or differences in characteristics without destroying the serviceability of the part or system [5].

Here are the eight most commonly used NDT techniques:

- Visual NDT (VT)
- Ultrasonic NDT (UT)
- Radiography NDT (RT)
- Eddy Current NDT (ET)
- Magnetic Particle NDT (MT)
- Acoustic Emission NDT (AE)
- Dye Penetrant NDT (PT)
- Leak Testing (LT)

Machine learning

Machine learning is a subset of artificial intelligence, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and on the basis of training, they build the model & perform a specific task[1].

These machine learning algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning

Supervised Machine Learning

In supervised learning, the machine is taught by example. The operator provides the machine learning algorithm with a known dataset that includes desired inputs and outputs, and the algorithm must find a method to determine how to arrive at those inputs and outputs. While the operator knows the correct answers to the problem, the algorithm identifies patterns in data, learns from observations and makes predictions. The algorithm makes predictions and

is corrected by the operator – and this process continues until the algorithm achieves a high level of accuracy/performance[2].

Supervised machine learning can be classified into two types of problems, which are given below:

- Classification
- Regression

a) Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "Yes" or No, Male or Female, Red or Blue, etc. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are Spam Detection, Email filtering, etc[2].

Some popular classification algorithms are given below:

- Random Forest Algorithm
- Decision Tree Algorithm
- Logistic Regression Algorithm
- Support Vector Machine Algorithm

b) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc[3].

Some popular Regression algorithms are given below:

- Simple Linear Regression Algorithm
- Multivariate Regression Algorithm
- Decision Tree Algorithm
- Lasso Regression

Unsupervised Machine Learning

The machine learning algorithm studies data to identify patterns. There is no answer key or human operator to provide instruction. Instead, the machine determines the correlations and relationships by analysing available data. In an unsupervised learning process, the machine

learning algorithm is left to interpret large data sets and address that data accordingly. The algorithm tries to organise that data in some way to describe its structure. This might mean grouping the data into clusters or arranging it in a way that looks more organised[1].

Unsupervised Learning can be further classified into two types, which are given below:

- Clustering
- Association

a) Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behaviour[2].

Some of the popular clustering algorithms are given below:

- K-Means Clustering algorithm
- Mean-shift algorithm
- DBSCAN Algorithm
- Principal Component Analysis
- Independent Component Analysis

b) Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in Market Basket analysis, Web usage mining, continuous production, etc [3].

Some popular algorithms of Association rule learning are:

- Apriori Algorithm
- Eclat
- FP-growth algorithm

Semi-Supervised Machine Learning

Semi-supervised learning is similar to supervised learning, but instead uses both labelled and unlabelled data. Labelled data is essentially information that has meaningful tags so that the algorithm can understand the data, whilst unlabelled data lacks that information. By using this combination, machine learning algorithms can learn to label unlabelled data [3].

Although Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabelled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels [1].

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced. The main aim of semi-supervised learning

is to effectively use all the available data, rather than only labelled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabelled data into labelled data. It is because labelled data is a comparatively more expensive acquisition than unlabelled data [1].

Semi-Supervised Machine Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance. Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards [1].

In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only. Due to its way of working, reinforcement learning is employed in different fields such as Game theory, Operation Research, Information theory, multi-agent systems [1].

Reinforcement learning is categorized mainly into two types of methods/algorithms:

- Positive Reinforcement Learning: Positive reinforcement learning specifies increasing the tendency that the required behaviour would occur again by adding something. It enhances the strength of the behaviour of the agent and positively impacts it [1].
- Negative Reinforcement Learning: Negative reinforcement learning works exactly opposite to the positive RL. It increases the tendency that the specific behaviour would occur again by avoiding the negative condition [1].

3. Aims and objectives

As discussed in the literature review, there are four types of machine learning methods namely supervised, unsupervised, semi-supervised and reinforced. Due to limited time given to complete the project I will only explore two main types which are supervised and unsupervised machine learning.

I chose supervised learning because supervised learning work with the labelled dataset so one can have an exact idea about the classes of objects. These algorithms are helpful in predicting the output on the basis of prior experience.

I chose unsupervised learning because its algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabelled dataset. Unsupervised algorithms are preferable for various tasks as getting the unlabelled dataset is easier as compared to the labelled dataset [3].

The steps that will be followed for supervised learning are:

- Build a clustering algorithm
- Train and test the algorithm
- Optimize and test the algorithm

The steps that will be followed for unsupervised learning are:

- Choose a clustering algorithm
- Test the algorithm
- Optimize and test the algorithm

The goal is to compare the two different machine learning methods to see which one works best for the project.

4. Unsupervised learning

4.1 Selecting a clustering algorithm

4.1.1 Methodology for choosing a clustering algorithm

As discussed in the literature review that the purpose for unsupervised learning is to discover hidden and interesting patterns in unlabelled data. The list of clustering algorithms that I will pick from are:

- KMeans
- MiniBatchKMeans
- DBSCAN

- Agglomerative
- MeanShift
- Birch
- Optics
- GaussianMixture

Example of the data provided is shown below in figure 1.

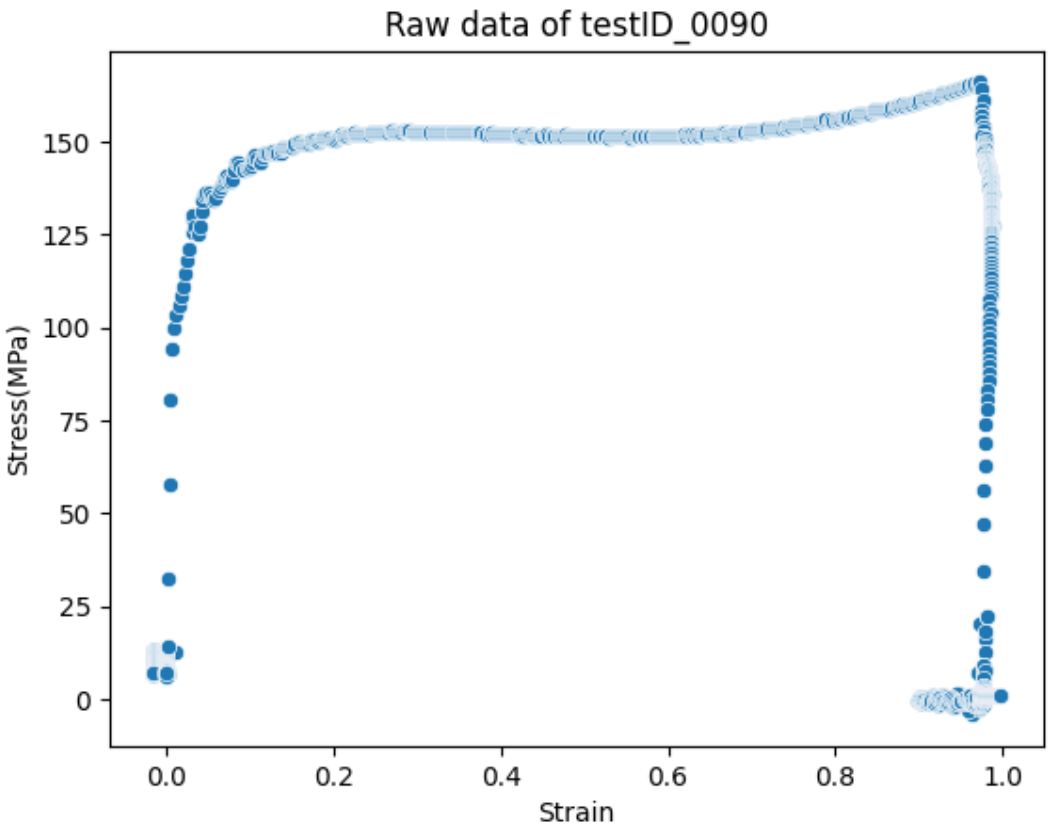


Figure 1 Raw data of test 0090

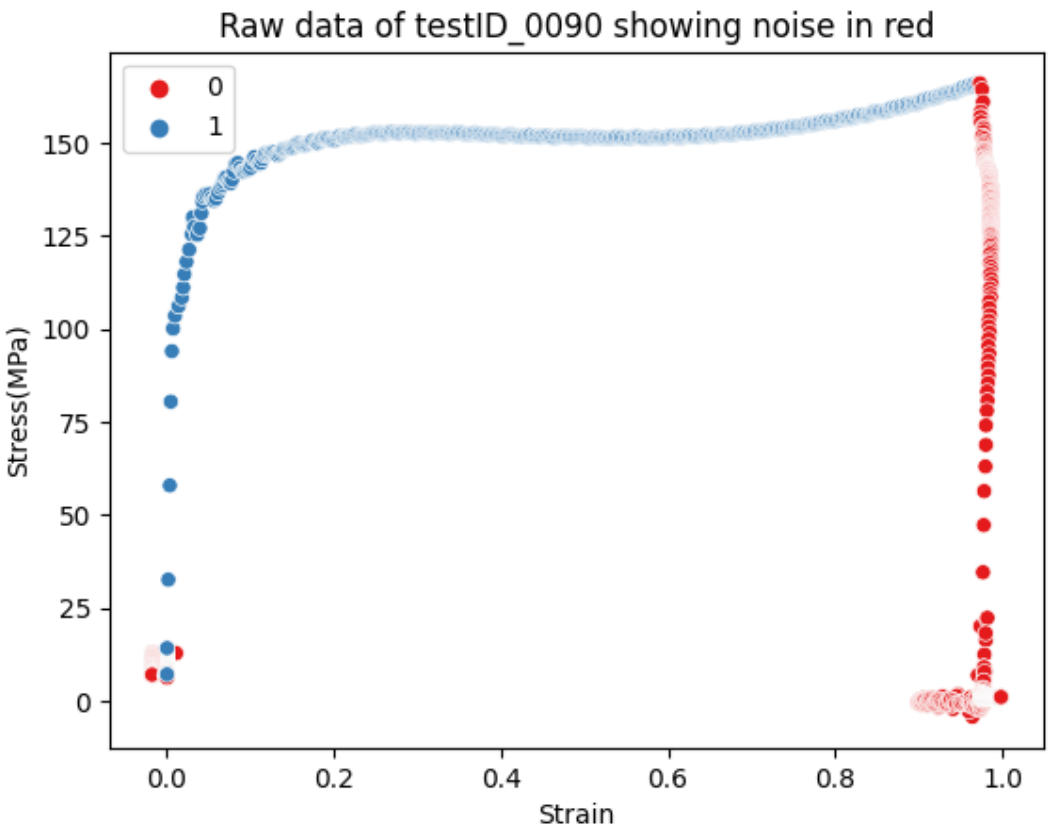


Figure 2 Raw data of test 0090 showing noise

Figure 1 shows raw data that collected from the uniaxial compression test. Figure 2 shows the noise in red from the raw data. Thus, the goal is to find clustering algorithms that are able to cluster spurious data together.

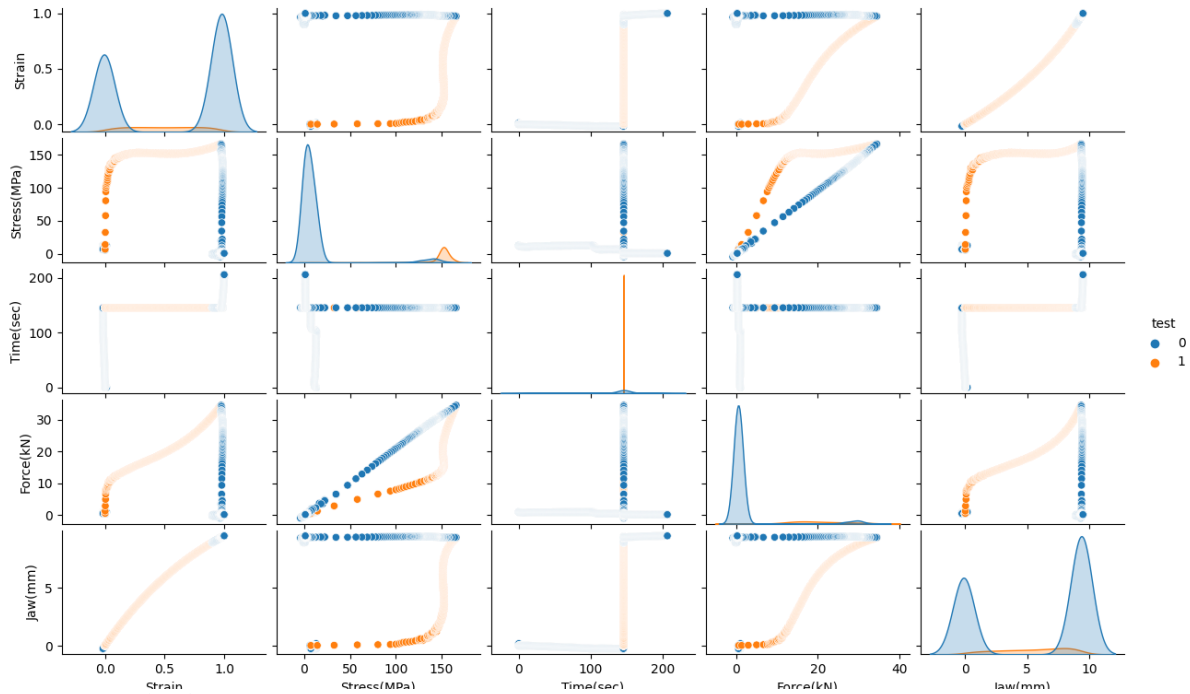


Figure 3 Pair plot showing noise

Figure 3 above shows the pair plot of raw data, and the spurious data that we are trying to cluster together is coloured in blue.

The method for selecting the best clustering algorithm is:

1. Perform exploratory data analysis. By analysing the pair plot of the clustering algorithms.
2. Develop a rating systems out of 10. To see which clustering algorithms are able to cluster the spurious data points together.
3. Pass multiple tests through the clustering algorithms.
4. Develop a rating system out of 10. To see which clustering algorithm perform best under different tests. Choose the best clustering algorithm that will be tested and optimized.

4.1.2 Discussion

The pair plots for all the clustering algorithms are in **appendix B**. The code used to generate the pair plots is in **appendix C**.

The rating systems will be used to eliminate the clustering algorithms that do not fit the criteria. Figure 2 shows that there is spurious data at the beginning and end. The pair-plots of the clustering algorithm will be compared to the Figure 3 pair-plot which is the goal.

The standard parameters are used for all the algorithms to develop. Pair-plots in appendix B. The algorithms that do not meet the below criteria will not be chosen.

There are two criteria:

1. Able to cluster spurious data in beginning
2. Able to cluster spurious data at the end

Clustering algorithm	cluster spurious data in beginning	cluster spurious data at the end	Total rating
KMeans	7	5	12
MiniBatchKMeans	8	6	14
DBSCAN	10	8	18
Agglomerative	9	3	12
MeanShift	9	7	16
Birch	8	7	15
Optics	1	0	1
GaussianMixture	9	8	17

The maximum total rating is 20. The algorithms that will be eliminated are those that have a total rating below 13. Because those that have a rating above 13 can be easily optimized to perform better.

Thus, the eliminated clustering algorithms are the following:

- KMeans (Total rating of 12)
- Agglomerative (Total rating of 12)
- Optics (Total rating of 1)

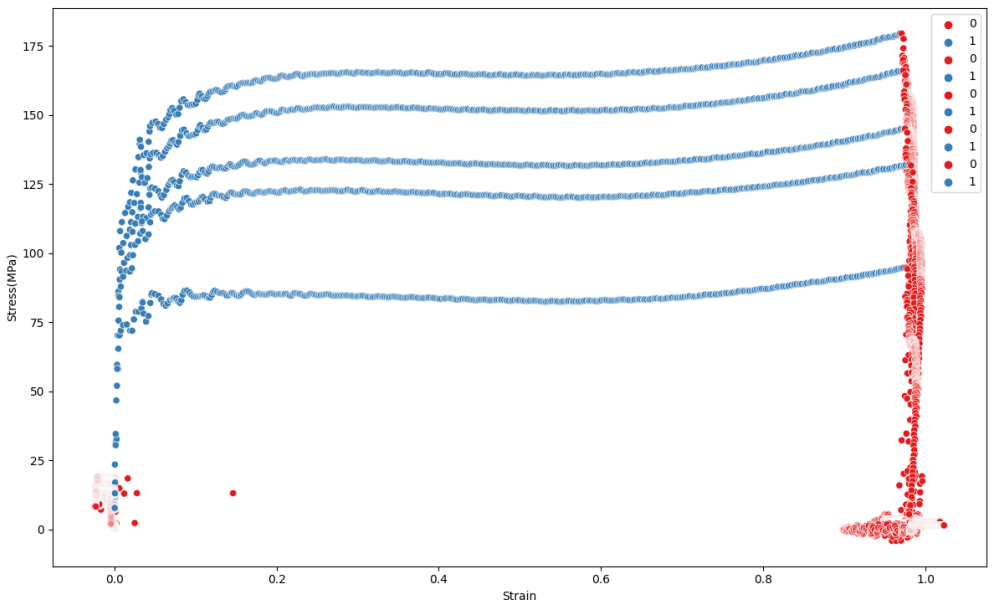


Figure 4 Multiple tests

The next step is to pass multiple test through the algorithms and see which one gets the highest total rating score. The multiple tests will be compared to figure 4 above which is the goal. The algorithm that best meets the below criteria will be chosen.

The criteria is the following:

- Able to cluster noise at the beginning for all tests
- Able to cluster noise at the end for all tests
- Maintains the same cluster label number for noise at the beginning
- Maintains the same cluster label number for noise at the end

The graphs showing multiple tests for different algorithms is in **appendix D** and the code is in **appendix E**.

Clustering algorithm	Cluster noise at the beginning	Cluster noise at the end	Maintain the same cluster number at the beginning	Maintain the same cluster number at the end	Total rating	Comments
MiniBatchKMeans	7	6	2	2	17	Unable to maintain the same cluster label number for spurious data.
DBSCAN	9	7	9	9	34	Able to cluster noise and maintain the same cluster label number for noise.
MeanShift	8	4	8	6	26	It struggles to cluster all spurious data at the end together.
Birch	8	4	5	5	22	It maintains the same cluster label number

						for noise half of the time.
GaussianMixture	8	9	3	4	24	Unable to maintain the same cluster label number for spurious data.

The clustering algorithm that has the highest total rating score DSCAN, with a rating of 34/40.

Thus, DBCAN clustering algorithm will be tested and optimized.

4.2 Test the algorithm

4.2.1 Methodology

Density Based Spatial Clustering of Application with Noise (DBSCAN) iterates through points and uses two key hyperparameters to assign cluster labels. Which are epsilon and minimum number of points.

Epsilon is the distance extended from a point (think of it like a radius). Minimum number of points refers to minimum points in the epsilon distance.

DBSCAN has three point types which are affected by its key hyperparameters, which are:

- Core
- Border
- Outlier

Core point is that which has minimum number of points in its epsilon range (including itself). See figure 5 below.

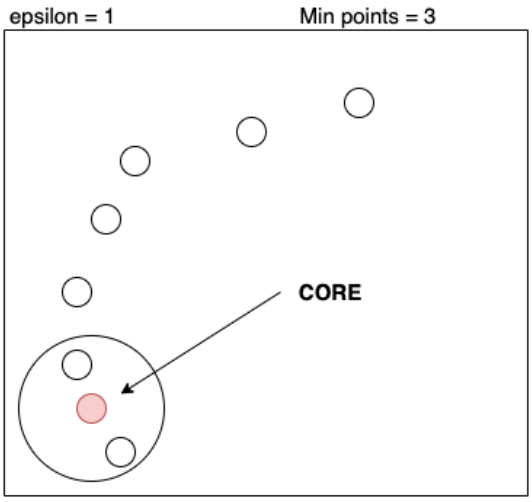


Figure 5 showing core point

Border point is that which is in the epsilon range of core point, but does not contain minimum number of points in its epsilon range. See figure 6 below.

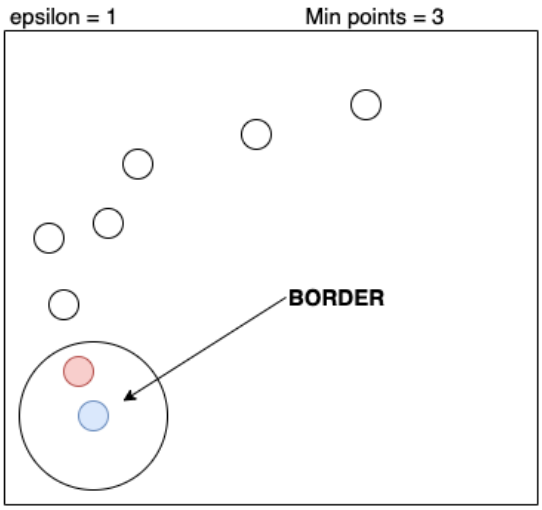


Figure 6 showing border point

Outlier point is that which cannot be reached by points in a cluster assignment. See figure 7 below.

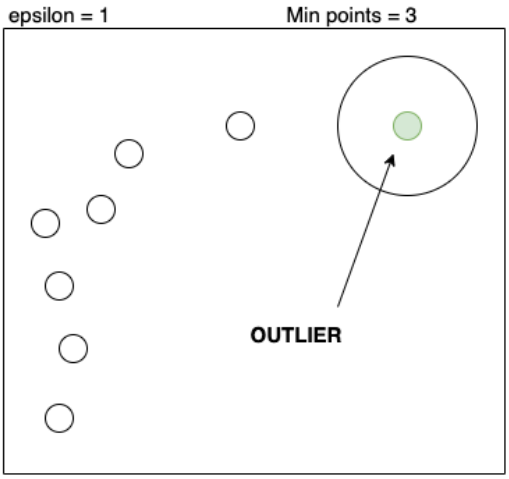


Figure 7 Showing the outlier

After optimization the values that will be used for the two key hyperparameter are the following:

1. Minimum number of points = 20
2. Epsilon = 1

The above parameters outputs the graph below in figure 8.

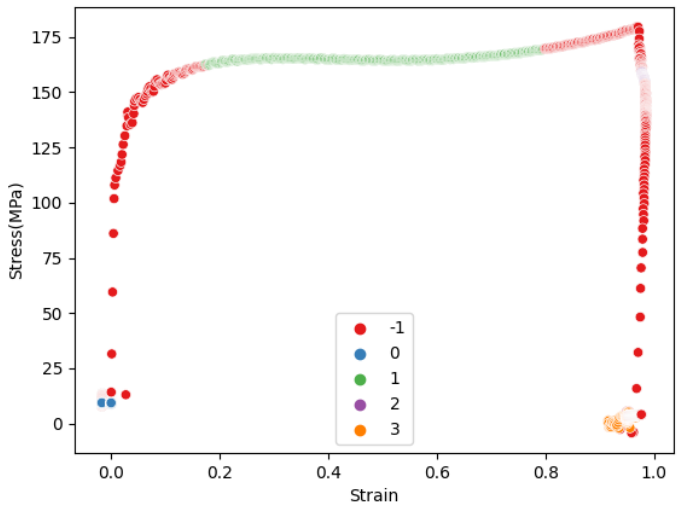
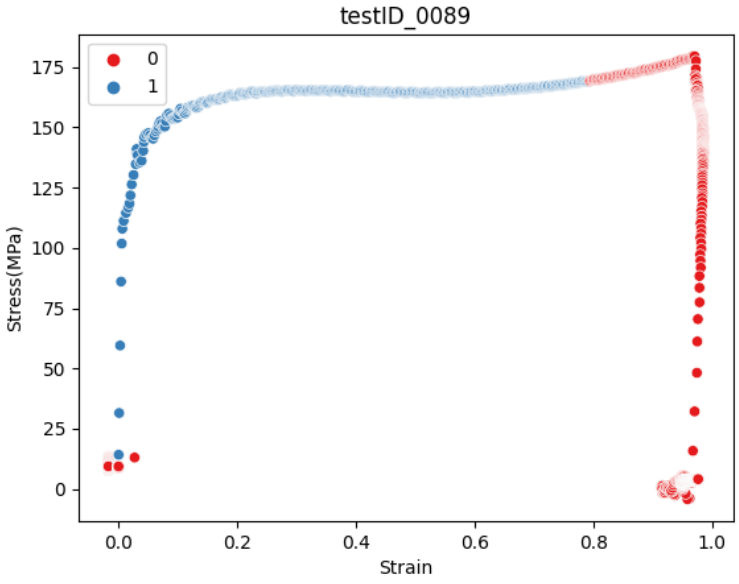
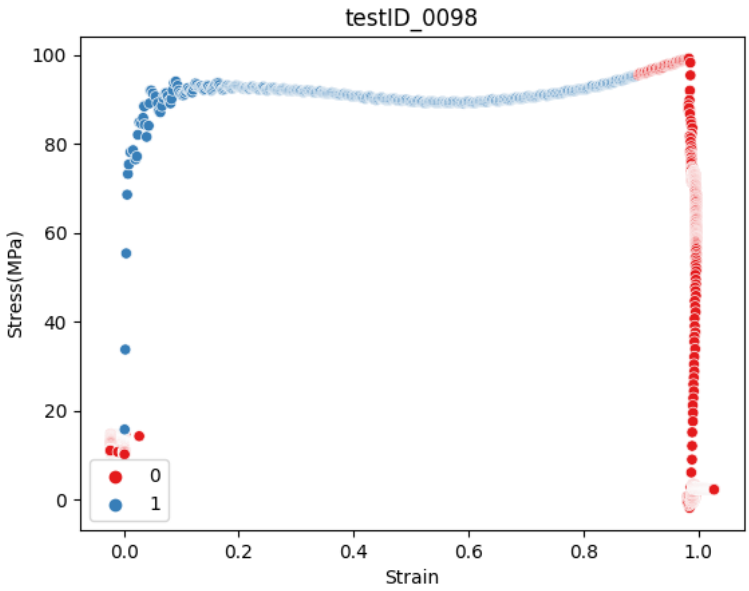


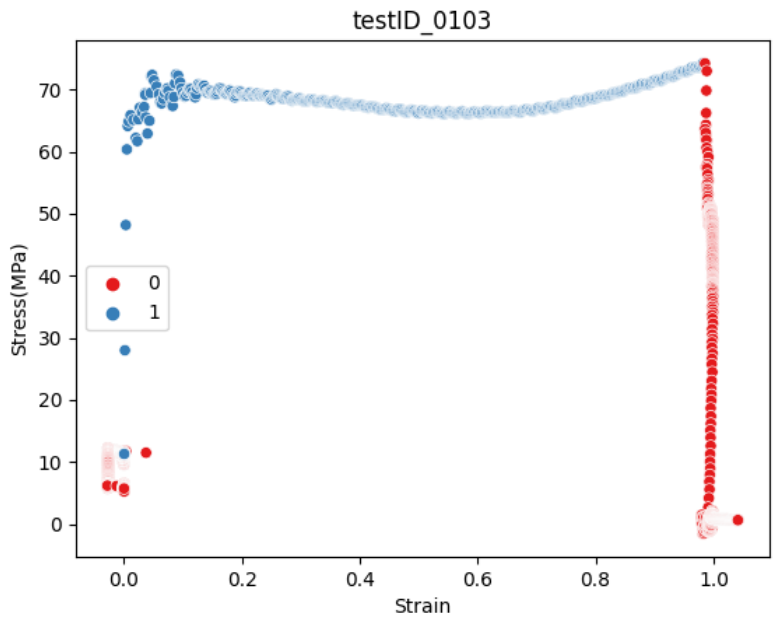
Figure 8

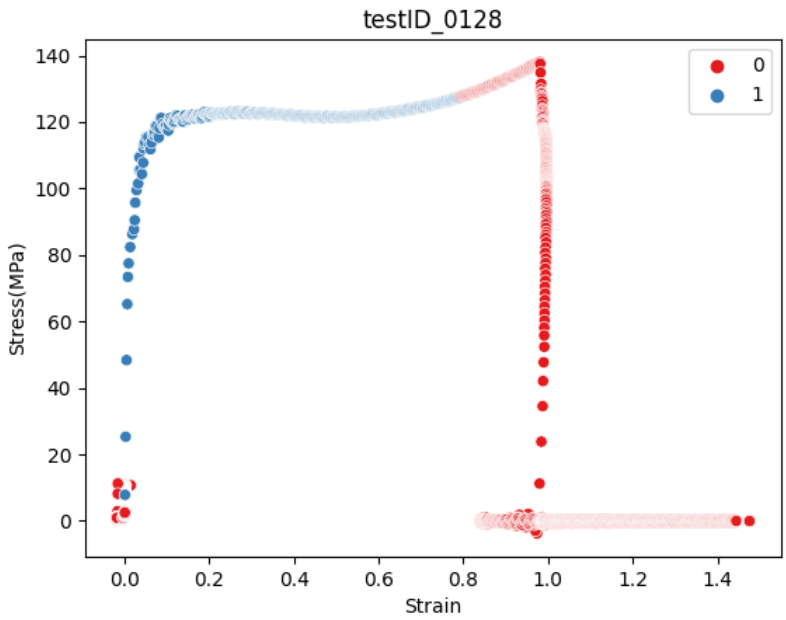
The goal is to have two clusters labels, the same as in figure 2. Thus, to do that the points in cluster "0" and cluster "-1" (after cluster '1') will be given the cluster label of "0", and the remaining points will be given the cluster label of "1".

4.2.2 Results


testID_0089
0
1
175
150
125
100
75
50
25
0
Stress(MPa)
0.0
0.2
0.4
0.6
0.8
1.0
Strain


testID_0098
100
80
60
40
20
0
Stress(MPa)
0
1
0.0
0.2
0.4
0.6
0.8
1.0
Strain





4.2.3 Discussion

The graphs in the results show that the DBSCAN algorithm is able to cluster the noise at the beginning perfectly. It is also able to cluster all the spurious points at the end together. But problem is that it clusters some of the valuable data points together with spurious data points in cluster label '0'.

Thus, the DBCAN clustering algorithm has to be optimized so that the valuable points at the end can be retained.

4.3 Optimize the algorithm and evaluate the performance

4.3.1 Methodology

As discussed above that the issue with the DBSCAN clustering algorithm is that it clusters valuable points together with the with spurious points at the end. This can be seen in figure 9 below.

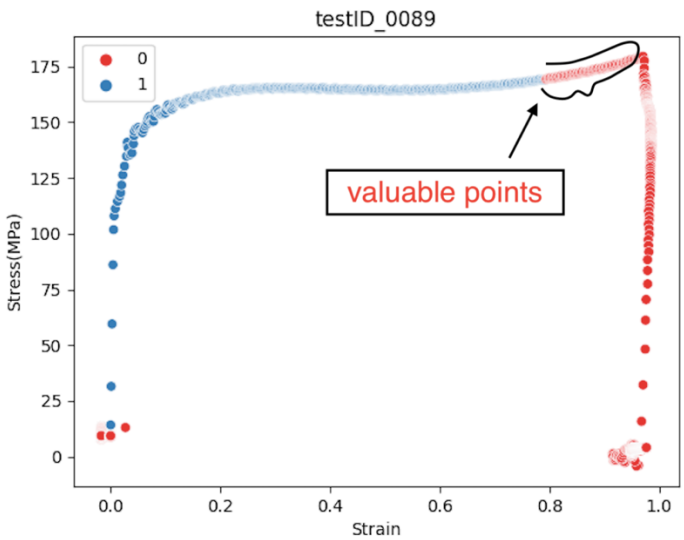


Figure 9

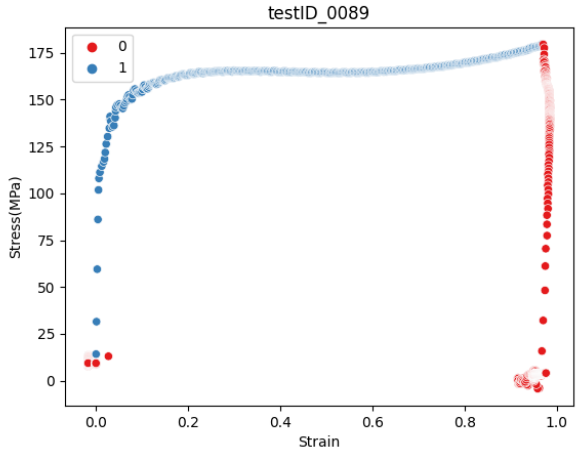
Thus, an optimization function will be built. The optimization function will be able to check which points in cluster label '0' are valuable, it will move valuable points from cluster label '0' to cluster label '1'.

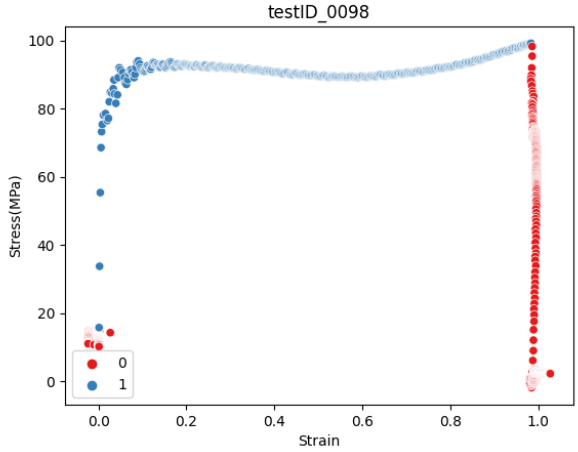
The optimization function will work by following the below steps:

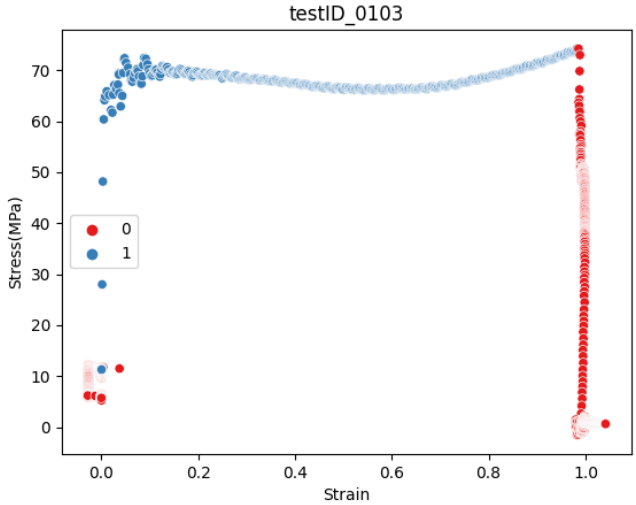
1. Extract the last Stress(MPa) value in cluster 1 (Stress value of the last blue point). Assign it to a variable "last_p"
2. Then it will check if the next ten stress values after "last_p" are all less than "last_p" in value.
3. If not all the next ten Stress(MPa) values are less than "last_p", the function will then iterate by assigning the next Stress value the variable "last_p".
4. It will iterate until the following ten Stress(MPa) values are all less than "last_p" value.
5. After the iteration stops, it will assign the points after the "last_p" to cluster '0'.

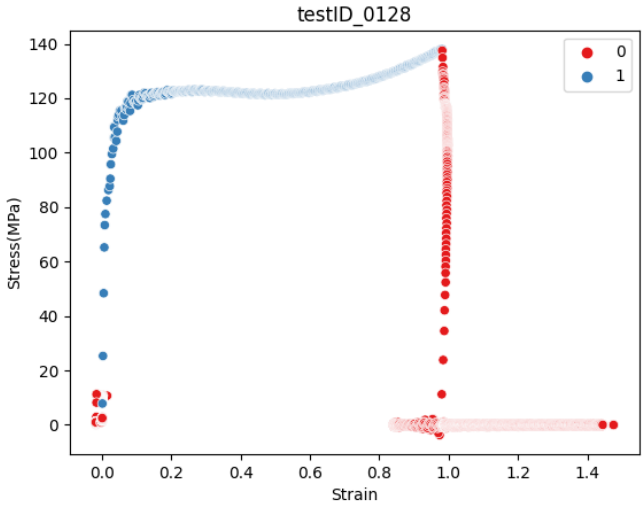
The code for the optimization function is in **appendix F**.

4.3.2 Results


testID_0089
0
1
175
150
125
100
75
50
25
0
Stress(MPa)
0.0
0.2
0.4
0.6
0.8
1.0
Strain


testID_0098
100
80
60
40
20
0
Stress(MPa)
0
1
0.0
0.2
0.4
0.6
0.8
1.0
Strain





4.3.3 Discussion

The DBSAN clustering algorithm works as expected after optimization by analysing the results. It is able to assign cluster label '0' to all the spurious data points and cluster label '1' to all the valuable data points.

Thus, no further optimization needs to be done.

5. Supervised learning

5.1 Build a clustering algorithm

5.1.1 Methodology

As discussed in the literature review that supervised learning is when the machine is taught by example. Raw and trimmed data is provided to build a clustering algorithm that will be trained and optimized to cluster the spurious data together and the valuable data together.

The steps that will be followed for supervised learning are:

1. Choose an algorithm
2. Train the algorithm
3. Evaluate the performance of the algorithm
4. Optimize the algorithm

The algorithms that will be used are the following:

- Polynomial Features
- Linear Regression

Polynomial Features algorithm is chosen because the stress strain curve has a non-linear relationship. The algorithm is able to raise existing features to an exponent, so that the points predicted can follow a specific polynomial curve. The algorithm does so by using its one key hyperparameter named "degree".

The graphs showing polynomial functions of different degrees is shown below in figure 10.

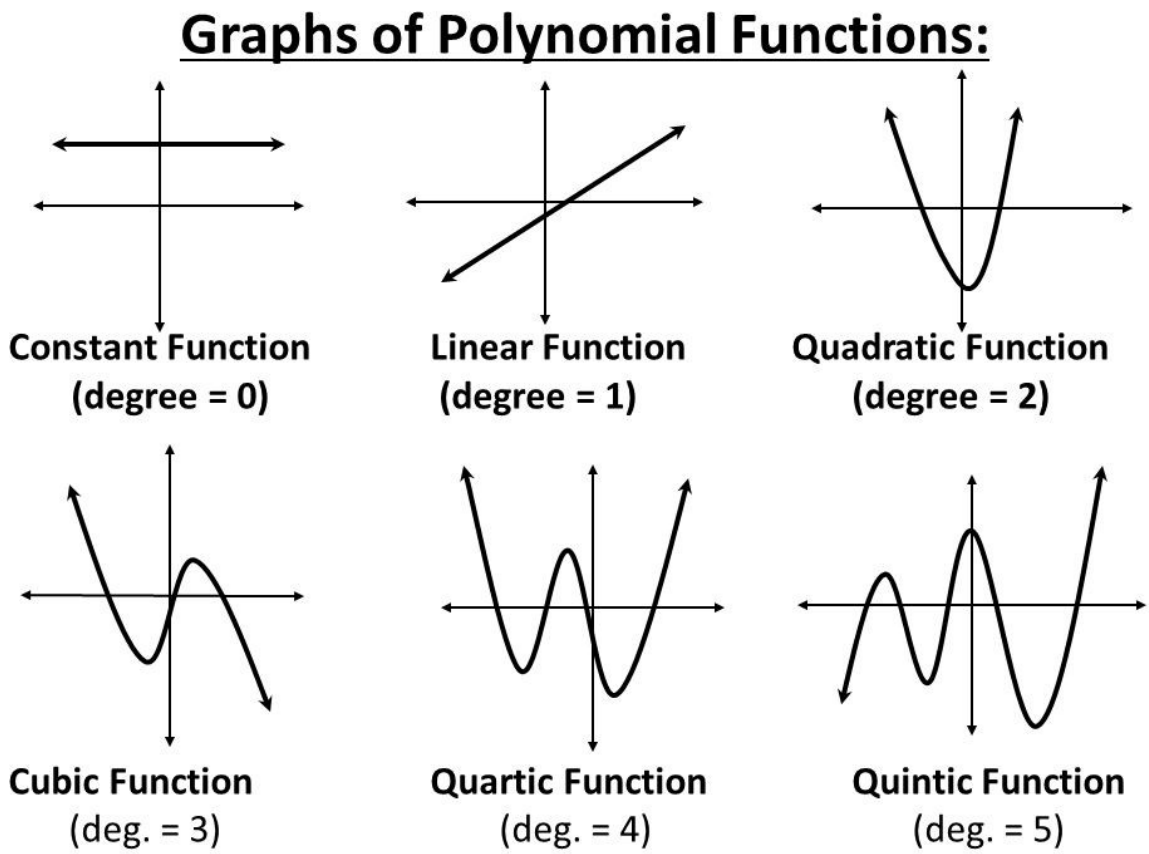


Figure 10

Linear Regression allows one to build a relationship between multiple features to estimate a target output.

Thus, using polynomial features and linear regression together provides us with the following advantages:

- The best approximation of the relationship between the dependent and independent variable.
- A broad range of functions can be fit under it.
- Fits a whole range of curvature.
- Improved the performance of Linear Regression.

Through optimization the chosen degree for Polynomial Features is 7.

The steps of building the algorithm are the following:

1. Give the raw unlabelled data a variable "X_unlabelled"
2. Call Polynomial features algorithm with degree 7, give it a variable "PolyFeatures"
3. Fit and transform the "X_unlabelled" to the "PolyFeatures" give it a variable "X_train".
4. Use the raw and trimmed data to find which data points in the raw data are not in the trimmed data. Assign those data points to a cluster label "0" and the rest of the data points to cluster label "1".
5. Assign the labelled data to a variable "Y_train"
6. Call a Linear Regression algorithm and give it a variable "Linear_R".
7. Fit the "X_train" and "Y_train" to the "Linear_R" algorithm.
8. Assign the raw unseen unlabelled data to a variable "X_t". Then fit and transform "X_t" to the "PolyFeatures" algorithm and give it a variable "X_test".
9. Assign the raw labelled data using number 4 method to a variable "Y_test".
10. Assign the variable "predict" to Linear_R.predict(X_test).

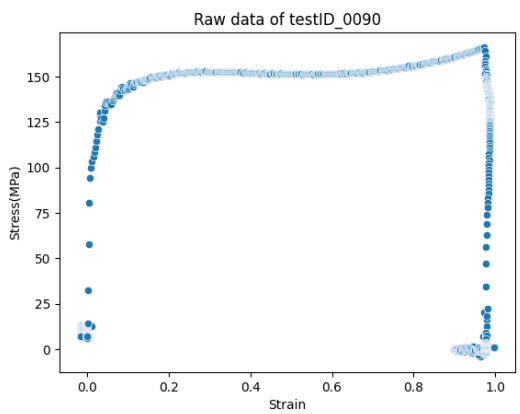
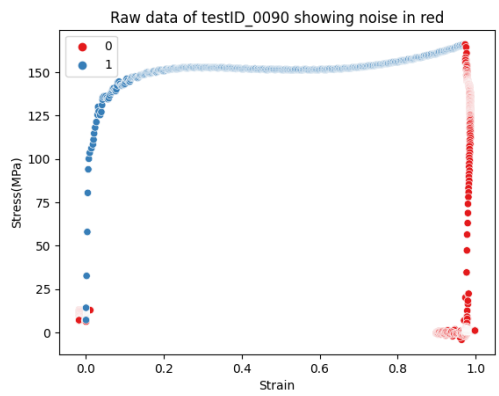
X Train	Y Train
	

Figure 11 Training

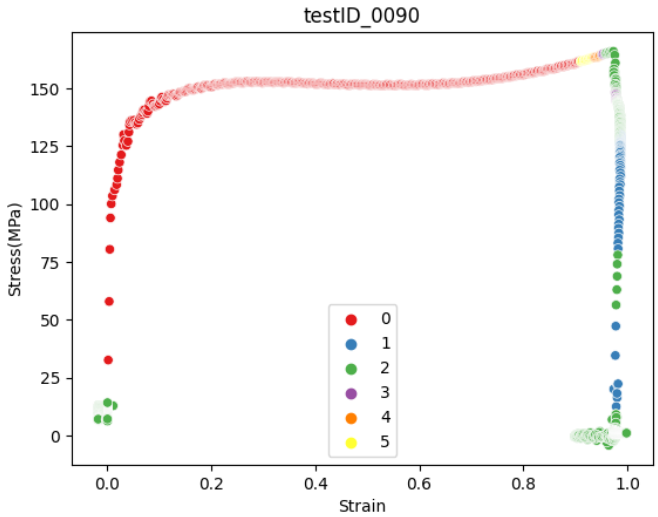
The X_train and Y_train is shown in the above Figure 12. The algorithm is the trained to only output '0' or '1'. The data points that have cluster label '0' are spurious and those that have a cluster label '1' are valuable points. The Linear Regression algorithm does not output either

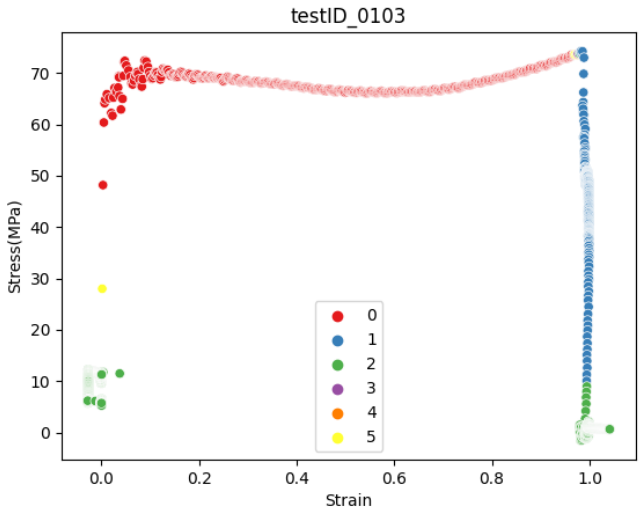
‘0’ or ‘1’, it output any real number between -1 and 1. Thus, the data points will be given the following cluster labels based on the output of the algorithm:

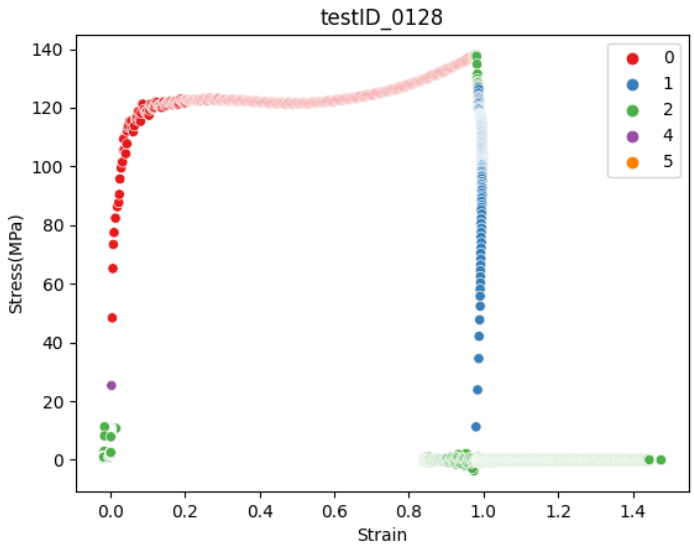
- Output less than -0.05 assign to a cluster label ‘1’.
- Output less than 0.1 assign to a cluster label ‘2’.
- Output less than 0.13 assign to a cluster label ‘3’.
- Output less than 0.2 assign to a cluster label ‘4’.
- Output less than 0.35 assign to a cluster label ‘5’.
- Output great than 0.35 assign to a cluster label ‘0’.

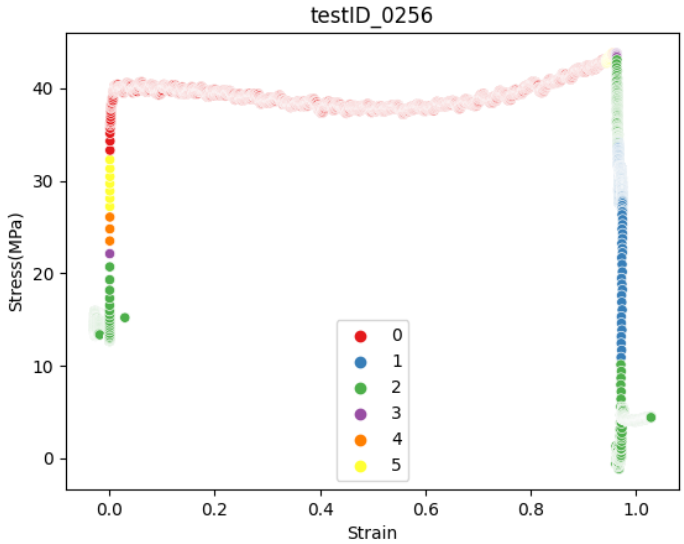
The code for this is in **appendix G**.

5.1.2 Results









5.1.3 Discussion

The above results of different graphs show that the valuable points are in cluster label ‘0’. Thus, the next step is to only have two cluster labels namely ‘0’ and ‘1’.

5.2 Train and test the algorithm

5.2.1 Methodology

When you pass the unseen data through the algorithm it assigns the each point to one of the six cluster labels.

The goal is to a build function that intakes the 6 cluster labels shown in figure 13 below and only output two cluster labels.

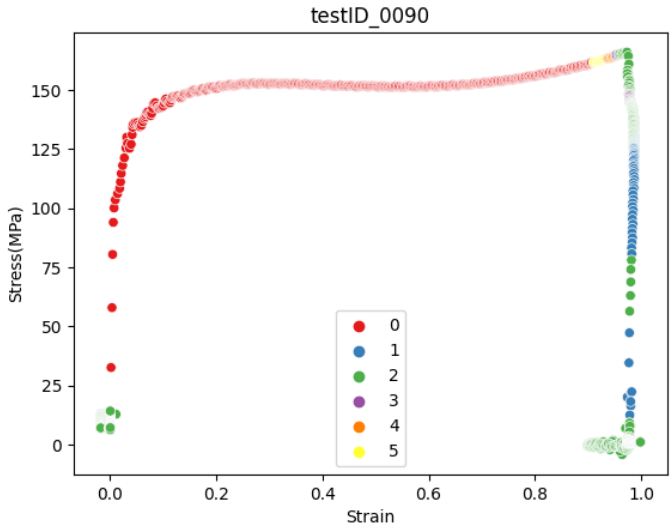


Figure 13

From the above graph shows that the valuable data points are in cluster '0'. Thus, the function will do the following:

- Assign all the data points in cluster '0' to cluster label '1'.
- Assign all the data points not in cluster '0' to cluster label '0'.

5.2.2 Results

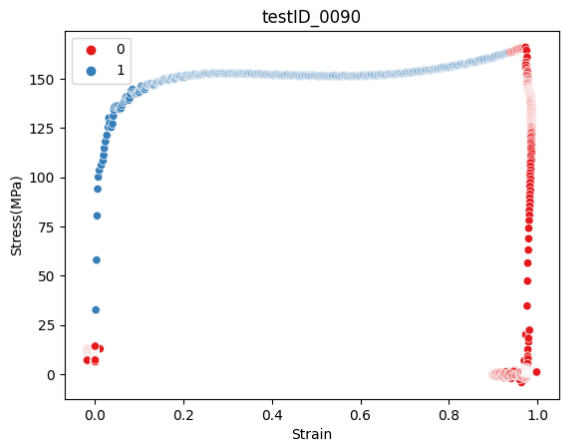
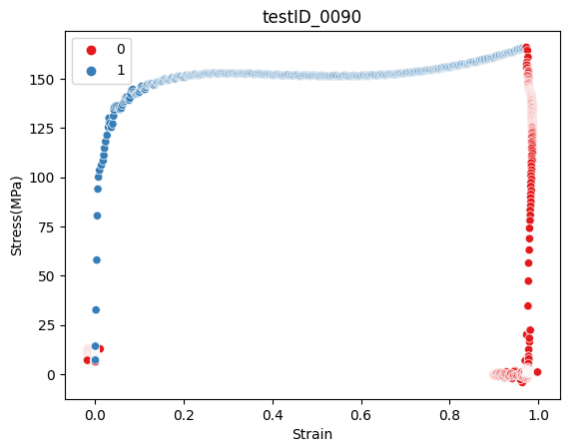
Predicted	Y test
	

Figure 14

Predicted	Y test

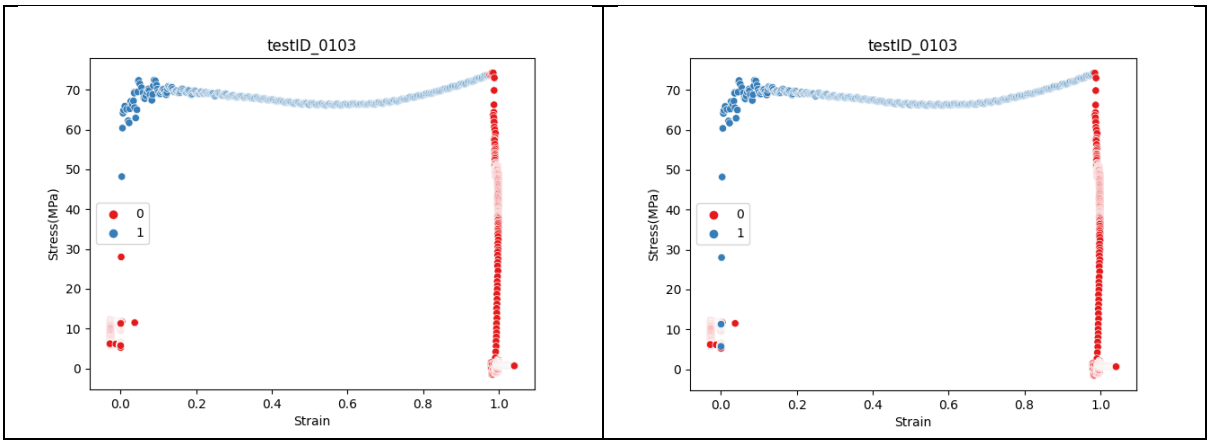


Figure 15

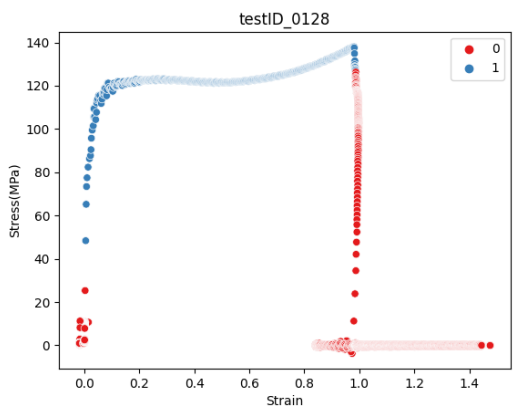
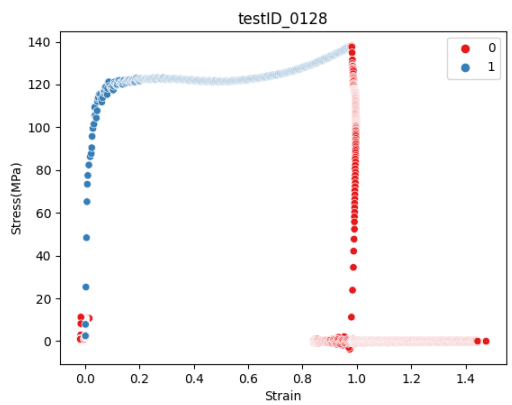
Predicted	Y test
	

Figure 16

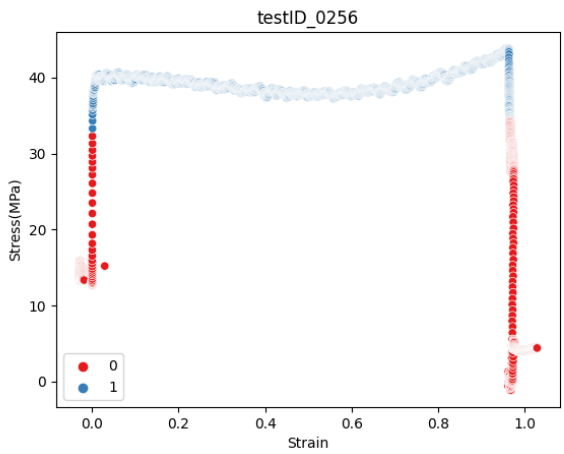
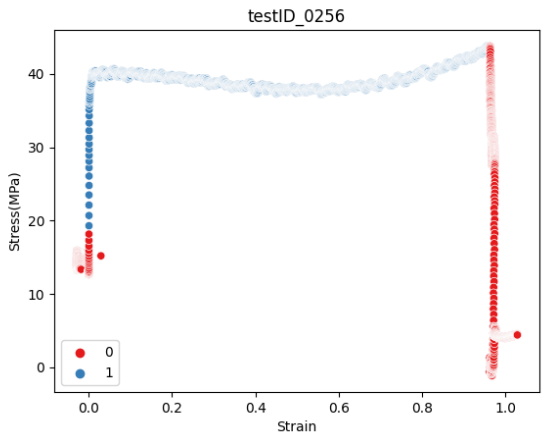
Predicted	Y test
	

Figure 17

5.2.3 Discussion

From the above results it can be seen the algorithm has the following problems:

1. The algorithm clusters some valuable points at the beginning in cluster label ‘0’ with spurious points. This is shown in figure 17.
2. The algorithm clusters some spurious points at the end in cluster label ‘1’ with valuable points. This is shown in figure 16.
3. The algorithm clusters some valuable points at end in cluster label ‘0’ with spurious points. This is shown in figure 14.

Thus, the algorithm will be optimized by fixing the above mentioned problems.

5.3 Optimize the algorithm and evaluate the performance

5.3.1 Methodology

As discussed in chapter 5.2.3 that to optimize the algorithm is fix the following problems:

1. The algorithm clusters some valuable points at the beginning in cluster label ‘0’ with spurious points. This is shown in figure 17.
2. The algorithm clusters some spurious points at the end in cluster label ‘1’ with valuable points. This is shown in figure 16.
3. The algorithm clusters some valuable points at end in cluster label ‘0’ with spurious points. This is shown in figure 14.

The optimization function will do the following to fix problem 1:

1. Assign the first Stress(MPa) value in cluster ‘1’ to a variable “first_v”
2. If the Stress(MPa) value before the “first_v” is less than “first_v”, then assign that value the variable “first_v”.
3. The function will iterate until the Stress(MPa) value before “first_v” is greater than “first_v” value.
4. After iteration stops, assign all the data points before “first_v” to cluster ‘0’.

The optimization function will do the following to fix problem 2:

1. Extract the last Stress(MPa) value in cluster 1 (Stress value of the last blue point). Assign it to a variable “last_p”
2. Then it will check if the next ten stress values after “last_p” are all less than “last_p” in value.
3. If not all the next ten Stress(MPa) values are less than “last_p”, the function will then iterate by assigning the next Stress value the variable “last_p”.
4. It will iterate until the following ten Stress(MPa) values are all less than “last_p” value.
5. After the iteration stops, assign the data points after the “last_p” to cluster ‘0’.

The optimization function will do the following to fix problem 3:

1. Extract the last Stress(MPa) value in cluster 1 (Stress value of the last blue point). Assign it to a variable “last_p”.
2. Check which data point occurs 20 points before “last_p”. And assign that point the variable “last_o”.
3. This will give rise to problem 2 above because some valuable points will clustered with spurious points in cluster ‘1’.
4. Follow the iteration method of problem 2.
5. After iteration stops, assign all the data points after “last_o” to cluster ‘0’.

The code used is in **appendix H**.

5.3.2 Results

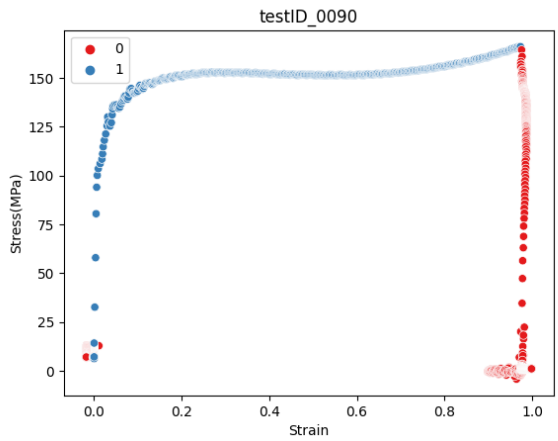
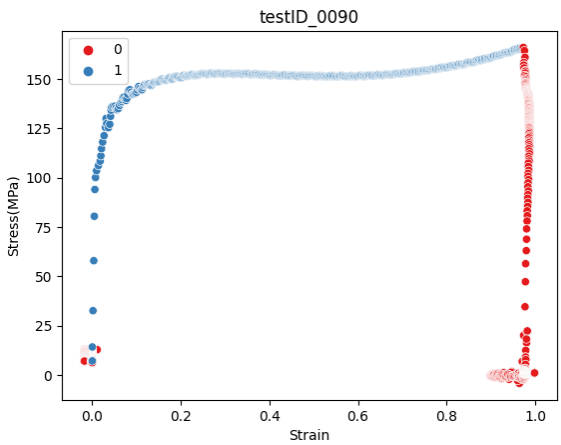
Predicted	Y test
	

Figure 18

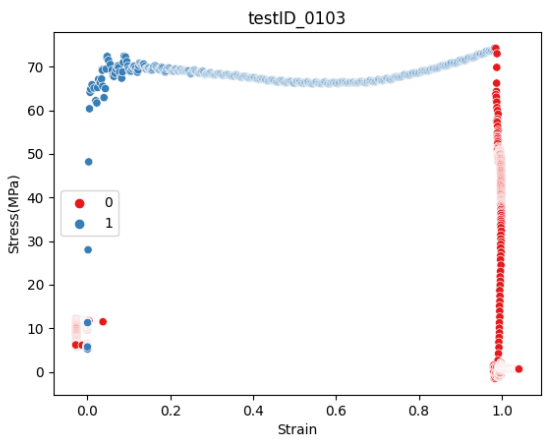
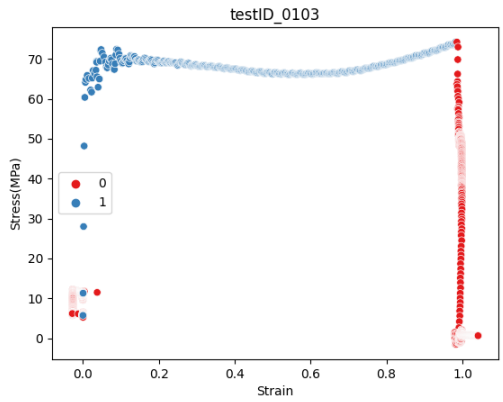
Predicted	Y test
	

Figure 19

Predicted	Y test

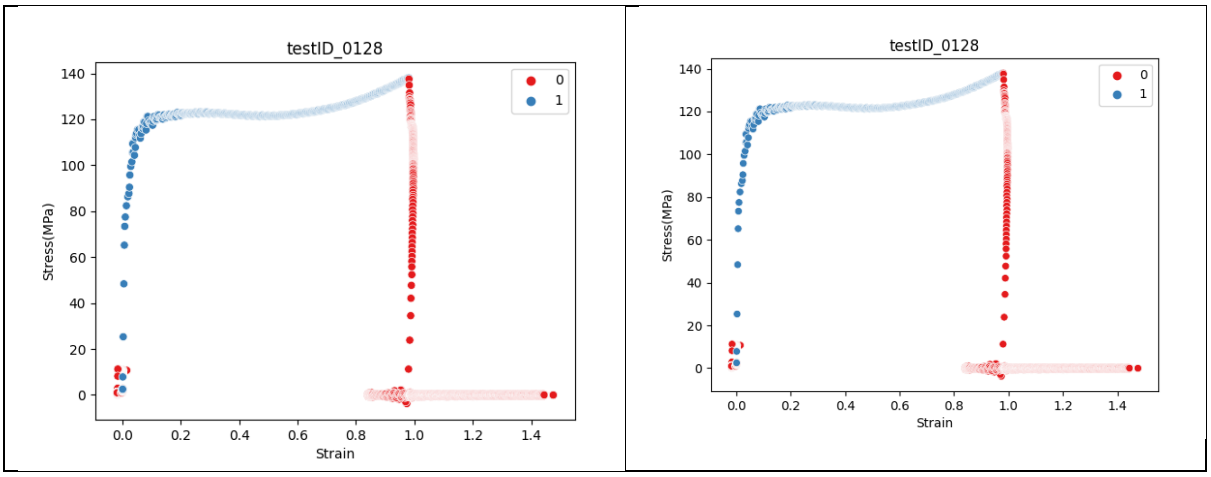


Figure 20

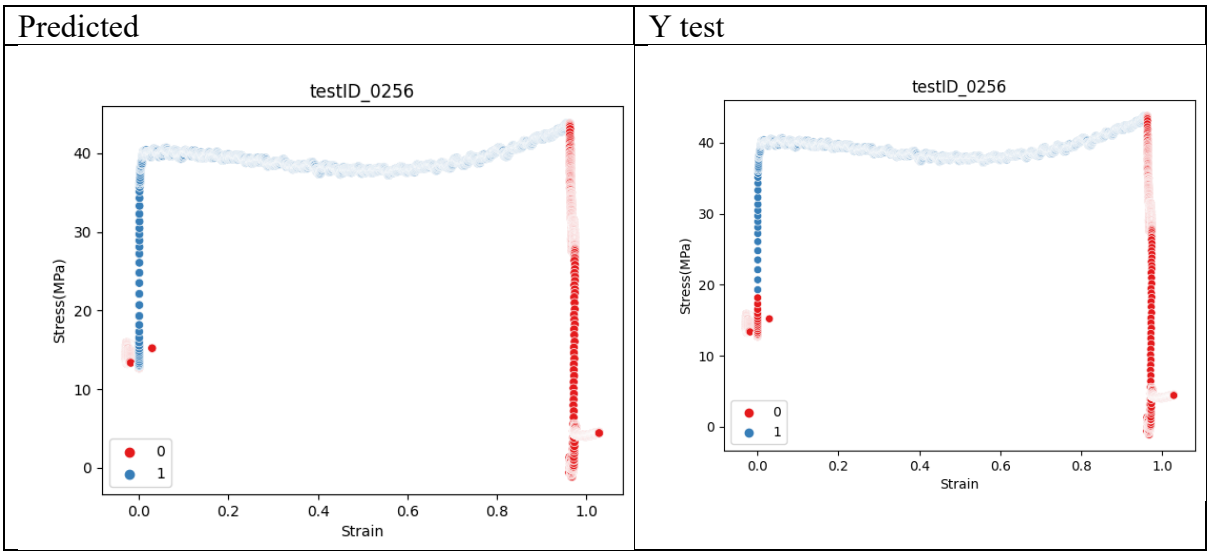


Figure 21

5.3.3 Discussion

The results in chapter 5.3.2 show that after optimization the algorithm is able to cluster all the spurious points at the end together. The small problem the algorithm has is that it cluster a few spurious data points in the beginning together with valuable points in cluster label '1'. This can be seen figure 21 above.

5.4 Compare the two different clustering methods

5.4.1 Methodology

There are two algorithms that have been built and optimized so far. The first algorithm was developed and optimized using unsupervised learning. The second algorithm was developed and optimized using supervised learning.

The performance of the two algorithms will be evaluated under the following conditions.

- Raw data with few spurious data points.
- Raw data with moderate number of spurious data points.
- Raw data with a lot of spurious data points.

5.4.2 Results

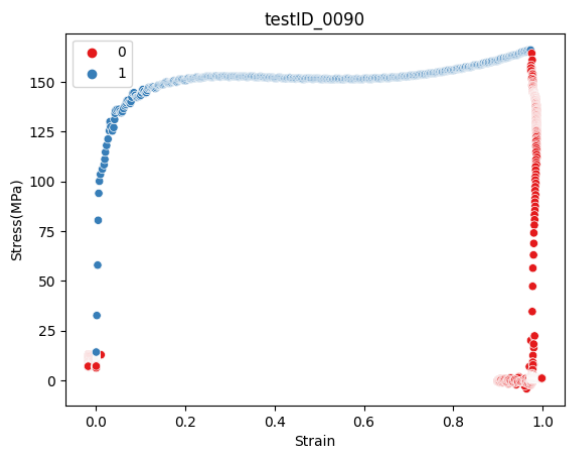
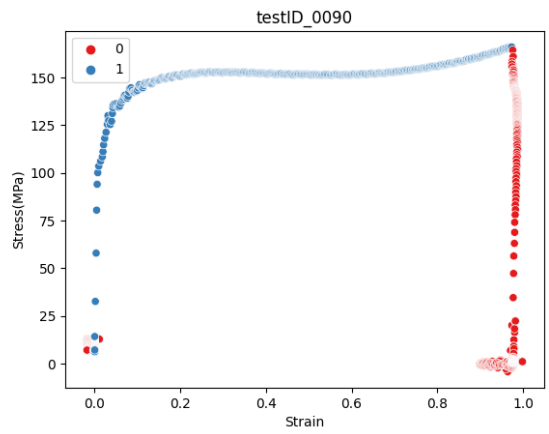
Unsupervised algorithm	Supervised algorithm
	

Figure 22

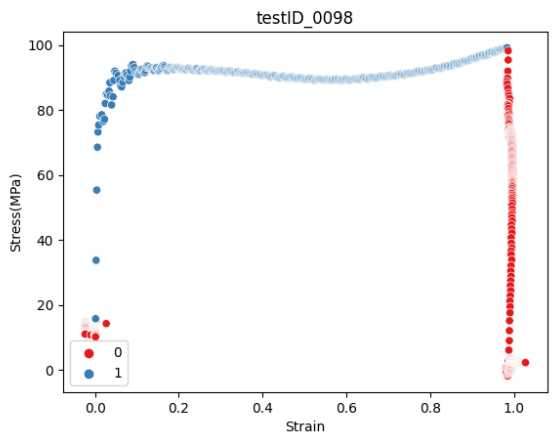
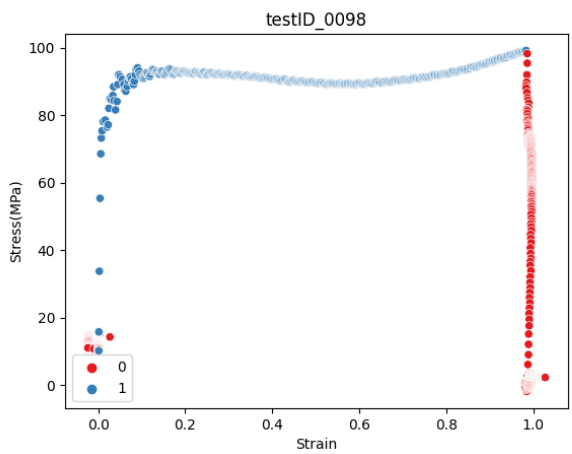
Unsupervised algorithm	Supervised algorithm
	

Figure 23

Unsupervised algorithm	Supervised algorithm

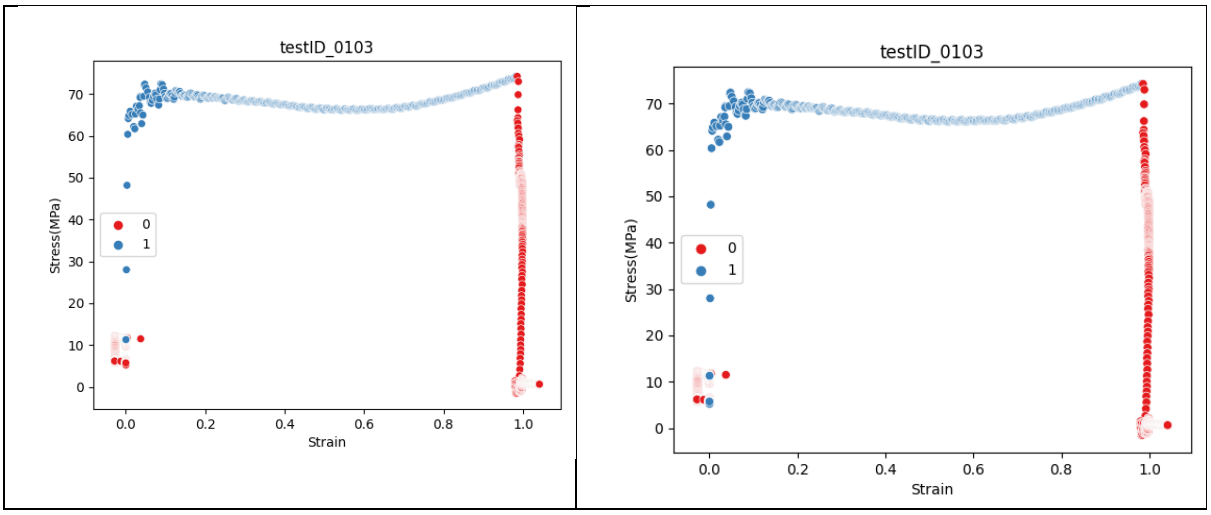


Figure 24

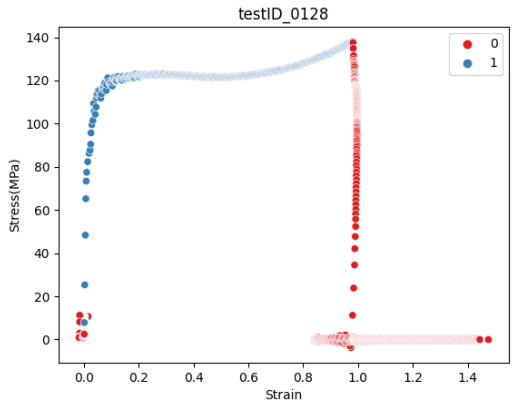
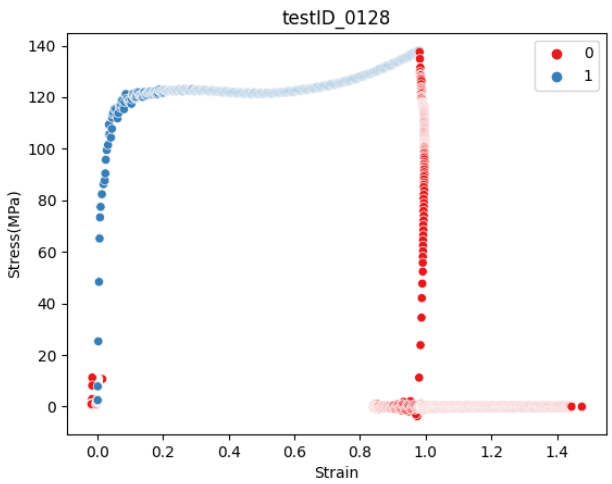
Unsupervised algorithm	Supervised algorithm
	

Figure 25

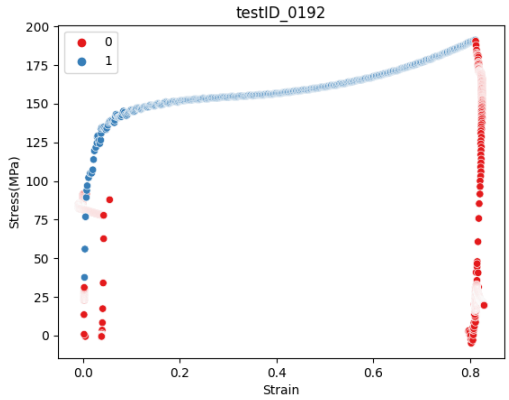
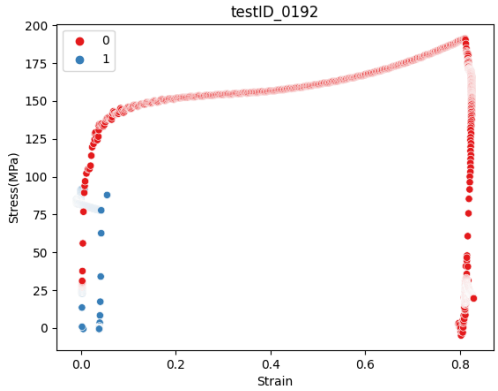
Unsupervised algorithm	Supervised algorithm
	

Figure 26

5.4.3 Discussion

The performance of the unsupervised algorithm is:

- It performs very well using raw data with few spurious data points. See figure 22 and 23.
- It performs very well using raw data moderate number of spurious data points. See figure 24.
- It performs very well using raw data with a lot spurious data points. See figure 25 and 26.

The performance of the unsupervised algorithm is:

- It performs well using raw data with few spurious data points. See figure 22 and 23.
- It performs good using raw data with moderate number of spurious data points. But it clusters few spurious data points at the beginning with the valuable points in cluster label '1'. See figure 24.
- It performs very poor using raw data with a lot spurious data points. See figure 26.

Thus, unsupervised algorithm is the best algorithm because it performs very well under all the conditions.

6. Project overall discussion

The objective of the project is achieved. Which is to use automated post-processing clustering methods to remove the noise from the collected data and output 'clean' data.

As discussed in chapter 5.4.3, the clustering algorithm that works best for this project is that which is built using unsupervised learning.

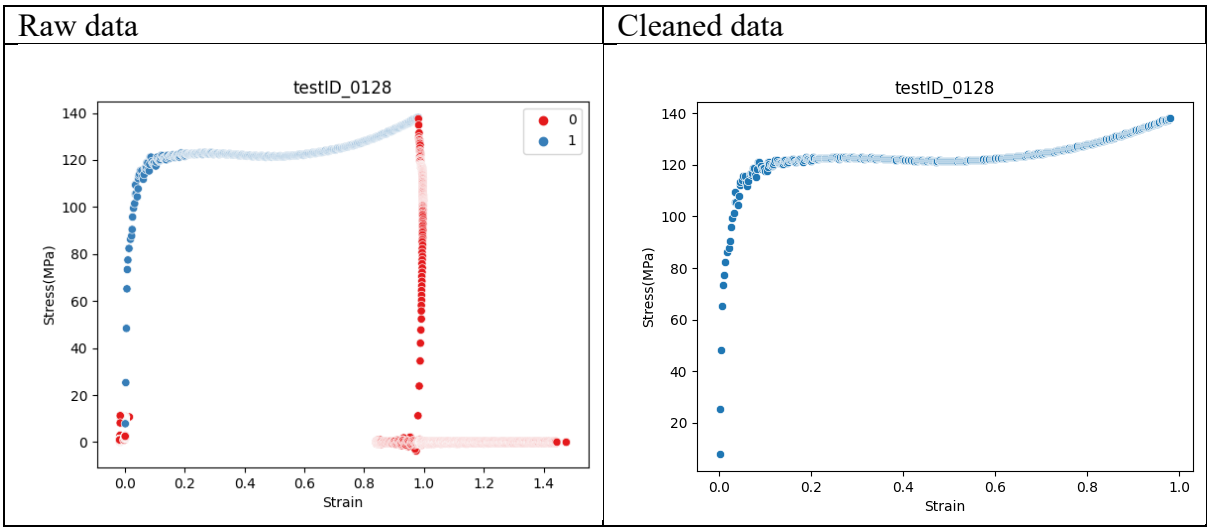


Figure 27

The optimized DSCAN clustering algorithm used for unsupervised learning is able to assign all the spurious data points to cluster label '0'. Thus, in order remove noise from the raw data, one just have to remove all the data points in the cluster label '0'. This is shown in the above figure 27.

Due to multiple tests conducted. A function has to be built that will allow the algorithm to remove the noise from all the tests conducted.

The function will work by following these steps:

1. Open a folder containing raw data
2. Call the first test in the raw data folder
3. Pass the test through the algorithm
4. Remove data points in cluster label '0'
5. Output the cleaned test data to a folder called "processed data"
6. Call the second test in the raw data folder. Repeat step 3 to 5. Iterate until all the raw data test are 'cleaned'.

The code used is shown in **appendix I**.
The few graphs of the cleaned tests is shown in **appendix J**.

7. Recommendation

Unsupervised algorithm works perfectly with the collected uniaxial compression test data.

Supervised algorithm has two issues namely:

1. It clusters few spurious data points at the beginning with the valuable points in cluster label '1'. See figure 24.
2. It performs very poor using raw data with a lot spurious data points.

Issue 1 can be fixed by:

- Building a function the find r squared value of the data points in the elastic region.
- Let the function iterate and stop when the r squared value is less than 0.6
- Assign the data point that gives the r squared value below 0.6 a variable "mark"
- Then cluster all the data points before "mark" to cluster label "0"
- Doing so will get rid of the spurious data points in cluster label "1".

Issue 2 can be fixed by:

- Using a cross validation method to find the parameter that allows the algorithm to perform well when using raw data with a lot of spurious data points.
- One can also use support vector machines to optimize the algorithm or build a new algorithm.

8. Further research

8.1 Information

As discussed in chapter 6 that at this point we have processed data which does not have noise.

The processed stress-strain curve provides design engineers with a long list of important parameters needed for application design. A stress-strain graph gives us many mechanical properties such as:

- Yield point

- Elastic modulus
- Failure

For tensile strength test one can also find the following parameters:

- The point where necking begins
- Ultimate tensile strength
- Flow stress
- Poisson ratio

8.2 Elastic modulus

Elastic modulus is the stiffness of a material. For the data used, the elastic modulus is represented as the slope of the linear part of the stress strain curve.

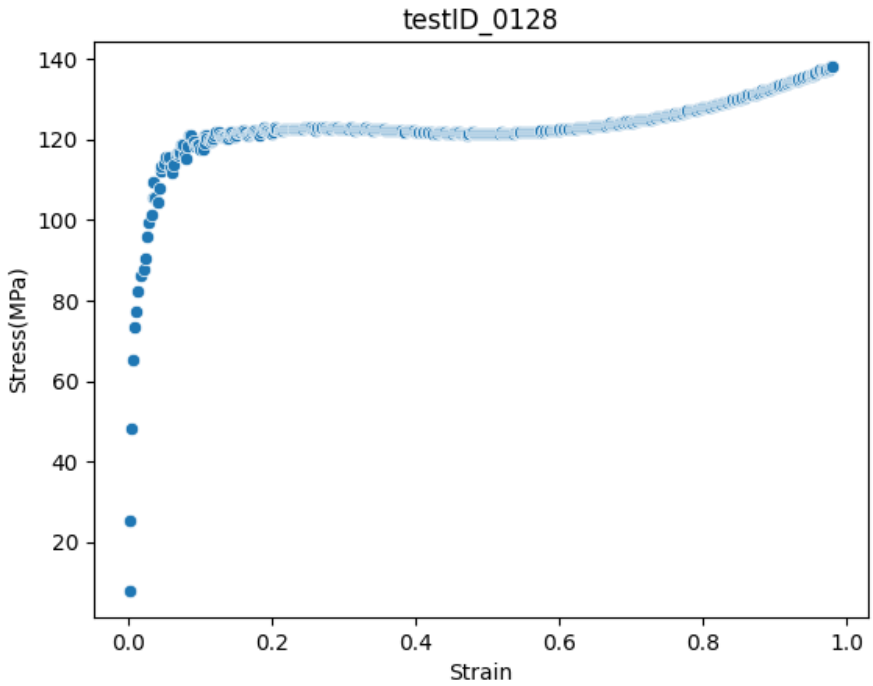


Figure 28

The elastic modulus calculated by dividing change in Stress(MPa) with change the in Strain.

Thus, the function used to find the elastic modulus of all the processed tests will work by following these steps:

1. Assign the first Stress(MPa) value a variable "y_1".
2. Assign the first Strain value a variable "x_1".
3. Assign the third Stress(MPa) value a variable "y_2"
4. Assign the third Strain value a variable "x_2"
5. Find the elastic modulus by using this equation (y_2 – y_1)/(x_2 – x_1).

The code using the above steps is in **appendix K**.

The graphs and their elastic modulus is **appendix L**.

Using the code in appendix . The elastic modulus of the graph in figure 28 is 11178.596983501393 MPa.

8.3 Yield point

Yield point is the point on a stress-strain curve that indicates the limit of elastic behaviour and the beginning of plastic behaviour. Below the yield point, a material will deform elastically and will return to its original shape when the applied stress is removed.
Thus, the function that will find yield point will work by following these steps:

1. Use the function from chapter 8.2 to find elastic modulus of the stress strain curve.
2. Offset the strain data points by 1%.
3. Plot a line that has a slope value equal to elastic modulus. The line should use offset strain values.
4. Assign the point that crosses the line the variable "Yield_point".

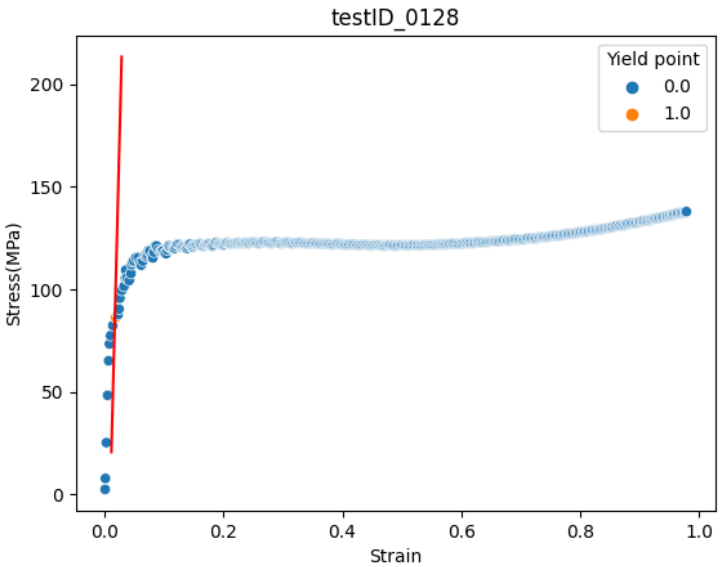


Figure 29

Figure 29 above shows how the function will work.

Thus, the function can be used to return strength and strain at yield.

Multiple graphs showing how the yield point function are in **appendix M**.

The code built and used for yield function is in **appendix N**.

8.4 Failure

Failure occurs when the specimen breaks or yields. If it fails by yielding, the specimen necks, thinning nonuniformly as shown in figure 30 below.

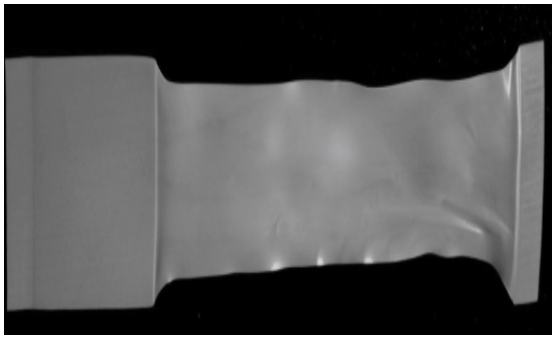


Figure 30

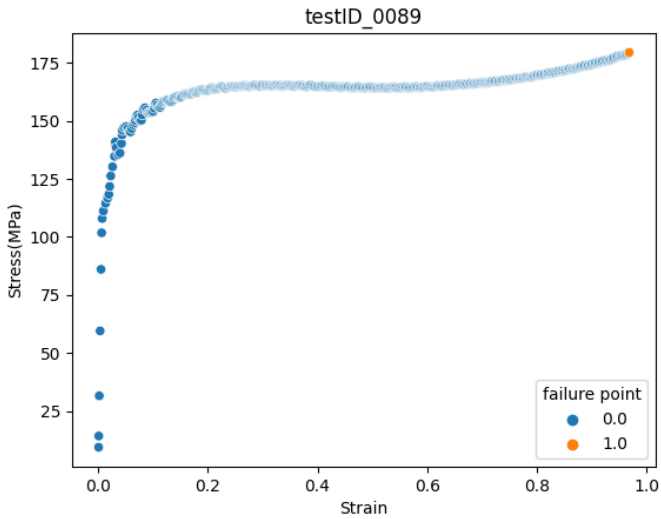


Figure 31

The failure point is easy to find because the it is the last point in the true stress strain curve. This point is shown in figure 31 above.

The failure point can provide the following information:

- Stress at failure
- Strain at failure
- Force at failure
- Specimen length at failure

9. Conclusion

The aim of the project was to use machine learning clustering algorithms that will be selected and trained to perform automated processing on the given stress-strain dataset.

The aim was achieved by using two types of machine learning methods namely unsupervised learning and supervised learning.

Unsupervised algorithm determines the correlations and relationships by analysing given raw unlabelled data. The algorithm was built following this steps:

1. Select a clustering algorithm based on how it meets the criteria discussed in chapter 4.1.
2. Test the algorithm
3. Optimize and test the algorithm

The selected clustering algorithm was DBSCAN as it best met the given criteria. DSCAN was tested and optimized to build unsupervised clustering algorithm.

A supervised learning algorithm takes a known set of input data (the learning set) and known responses to the data (the output), and forms a model to generate reasonable predictions for the response to the new input data. The steps followed to build the algorithm are:

1. Choose an algorithm
2. Train the algorithm
3. Evaluate the performance of the algorithm
4. Optimize the algorithm

Polynomial Features and Linear Regression models was trained and optimized to build supervised clustering algorithm.

Then the criteria was chosen to see how the two different algorithms will be perform under the following conditions:

- Raw data with few spurious data points.
- Raw data with moderate number of spurious data points.
- Raw data with a lot of spurious data points.

The unsupervised clustering algorithm performed very well under the above conditions.

A function was built to call all the tests in the "raw data" folder one by one to pass them through the unsupervised clustering algorithm. The algorithm then removes noise from the tests and outputs the cleaned test in a "processed data" folder.

Now that I manage to get processed data without noise. I did further research by building functions that will us mechanical properties such as:

- Yield point
- Elastic modulus
- Failure point

Thus, the project was completed successfully because the following objectives are met:

- Built a clustering algorithm that takes the raw data from all the uniaxial compression strength test collected.
- The algorithm then removes all the noise from the raw data.
- It then outputs the cleaned data into a "processed data" folder.

- Then it finds mechanical properties of all the tests in the “processed data” folder. And outputs the information in a folder named “processed data info”.

Therefore, the unsupervised clustering algorithm will save researchers working in mechanical testing a lot of time and manual labour. It will also allow researchers to find mechanical properties.

10. References

[1] Qifang Bi, Katherine E Goodman, Joshua Kaminsky, Justin Lessler, What is Machine Learning? A Primer for the Epidemiologist, American Journal of Epidemiology, Volume 188, Issue 12, December 2019, Pages 2222–2239

[2] Yang, Xin-She, and Xing-Shi He. Classification and Clustering Algorithms of Machine Learning with Their Applications. Vol. 855. Cham: Springer International Publishing :, 2020. Web.

[3] Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., S, R., Angela, Y.W.: An efficient kmeans clustering algorithm: analysis.

[4] F. R. Brotzen (1994) Mechanical testing of thin films, international materials reviews, 39:1, 24-45.

[5] Y. Hiramatsu, International Journal of Rock Mechanics and miming Sciences & Geomechanics abtracts. Volume 3, issue 2, May 1966, Pages 89-90.

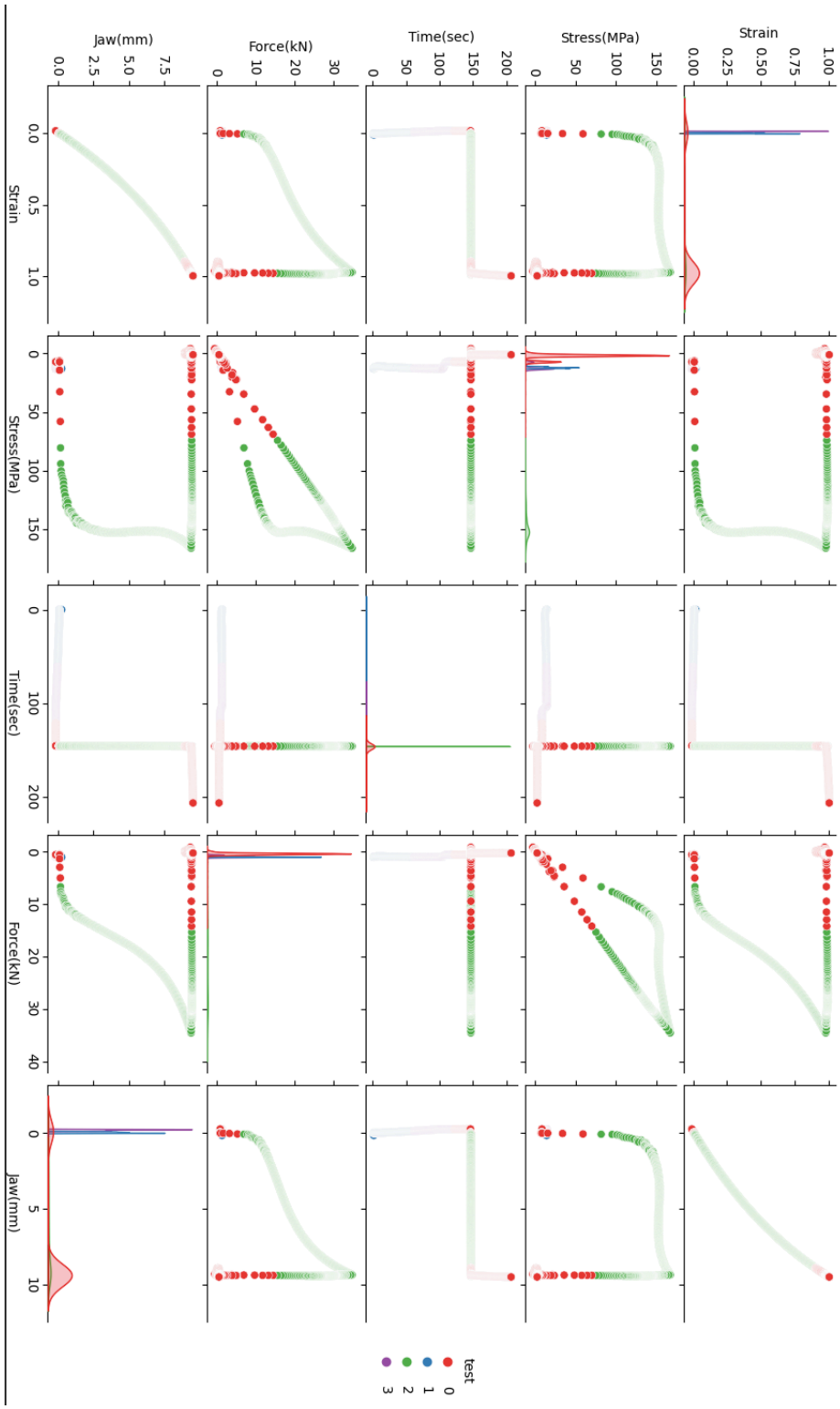
Appendix

Appendix A

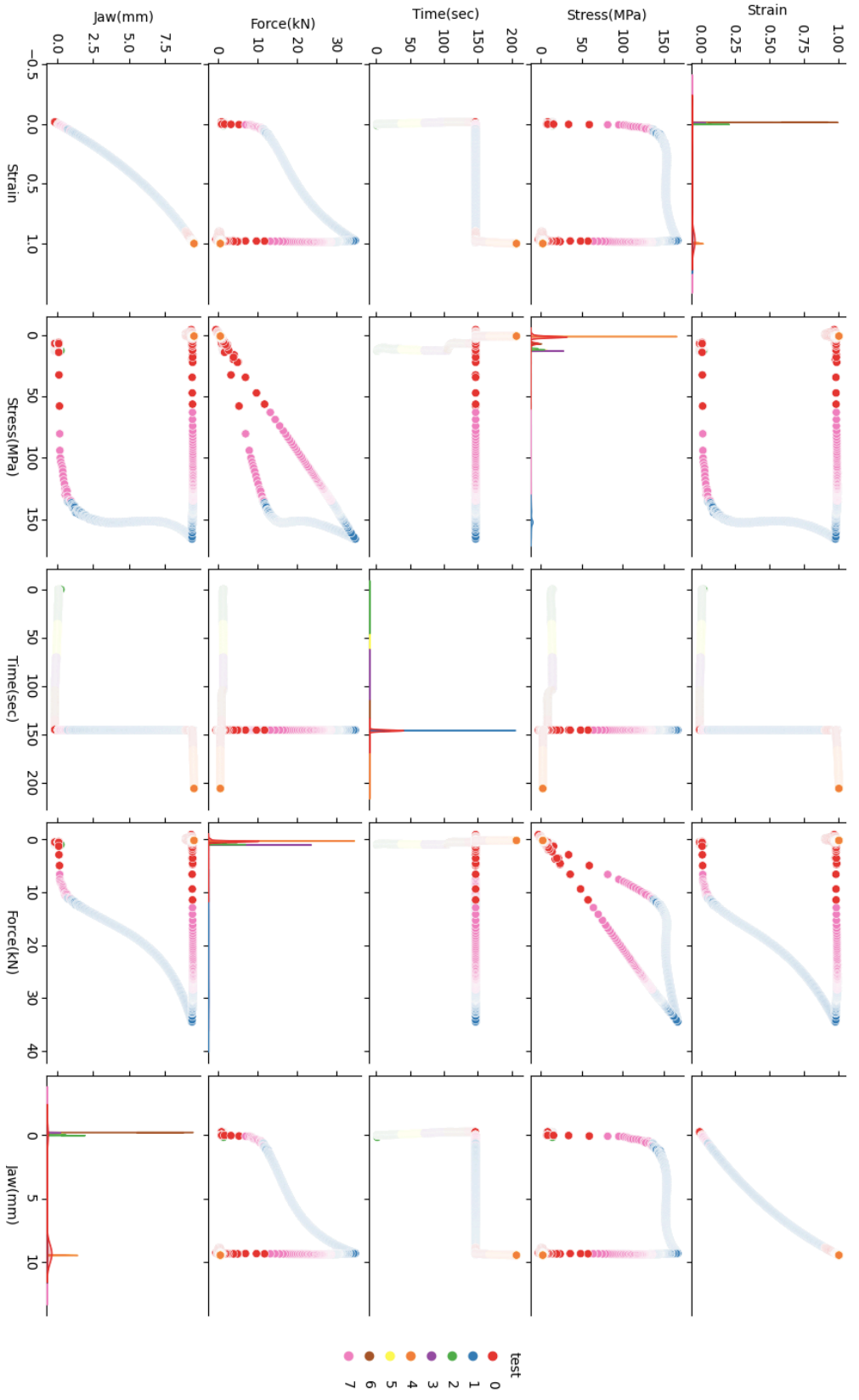
test id	reference	test type	material	rate	temperature	L_0 (mm)	D_0 (mm)
testID_0089	Baron	UC	AC	10	300	15	10
testID_0090	Baron	UC	AC	10	300	15	10
testID_0092	Baron	UC	AC	10	330	15	10
testID_0094	Baron	UC	AC	10	330	15	10
testID_0095	Baron	UC	AC	10	360	15	10
testID_0096	Baron	UC	AC	10	360	15	10
testID_0097	Baron	UC	AC	10	400	15	10
testID_0098	Baron	UC	AC	10	400	15	10
testID_0099	Baron	UC	AC	10	450	15	10
testID_0101	Baron	UC	AC	10	450	15	10
testID_0102	Baron	UC	AC	10	500	15	10
testID_0103	Baron	UC	AC	10	500	15	10
testID_0104	Baron	UC	AC	1	400	15	10
testID_0105	Baron	UC	AC	1	400	15	10
testID_0106	Baron	UC	AC	1	450	15	10
testID_0107	Baron	UC	AC	1	450	15	10
testID_0108	Baron	UC	AC	1	500	15	10
testID_0109	Baron	UC	AC	1	500	15	10
testID_0129	Baron	UC	H560	10	300	15	10
testID_0131	Baron	UC	H560	10	300	15	10
testID_0132	Baron	UC	H560	10	300	15	10
testID_0133	Baron	UC	H560	10	300	15	10
testID_0134	Baron	UC	H560	10	330	15	10
testID_0135	Baron	UC	H560	10	330	15	10
testID_0136	Baron	UC	H560	10	330	15	10
testID_0137	Baron	UC	H560	10	330	15	10
testID_0138	Baron	UC	H560	10	360	15	10
testID_0139	Baron	UC	H560	10	360	15	10
testID_0140	Baron	UC	H560	10	360	15	10
testID_0141	Baron	UC	H560	10	360	15	10

Appendix B

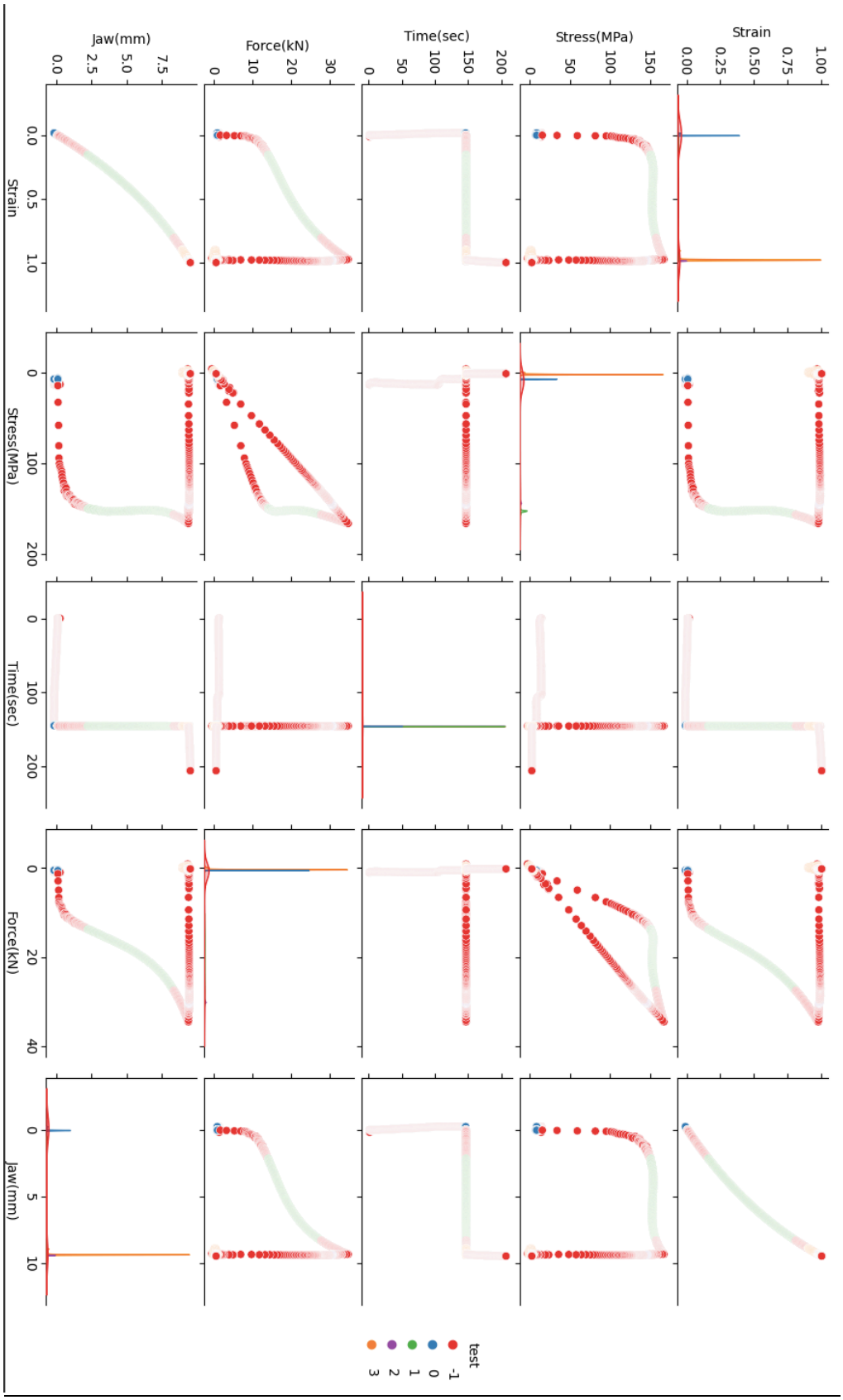
KMeans



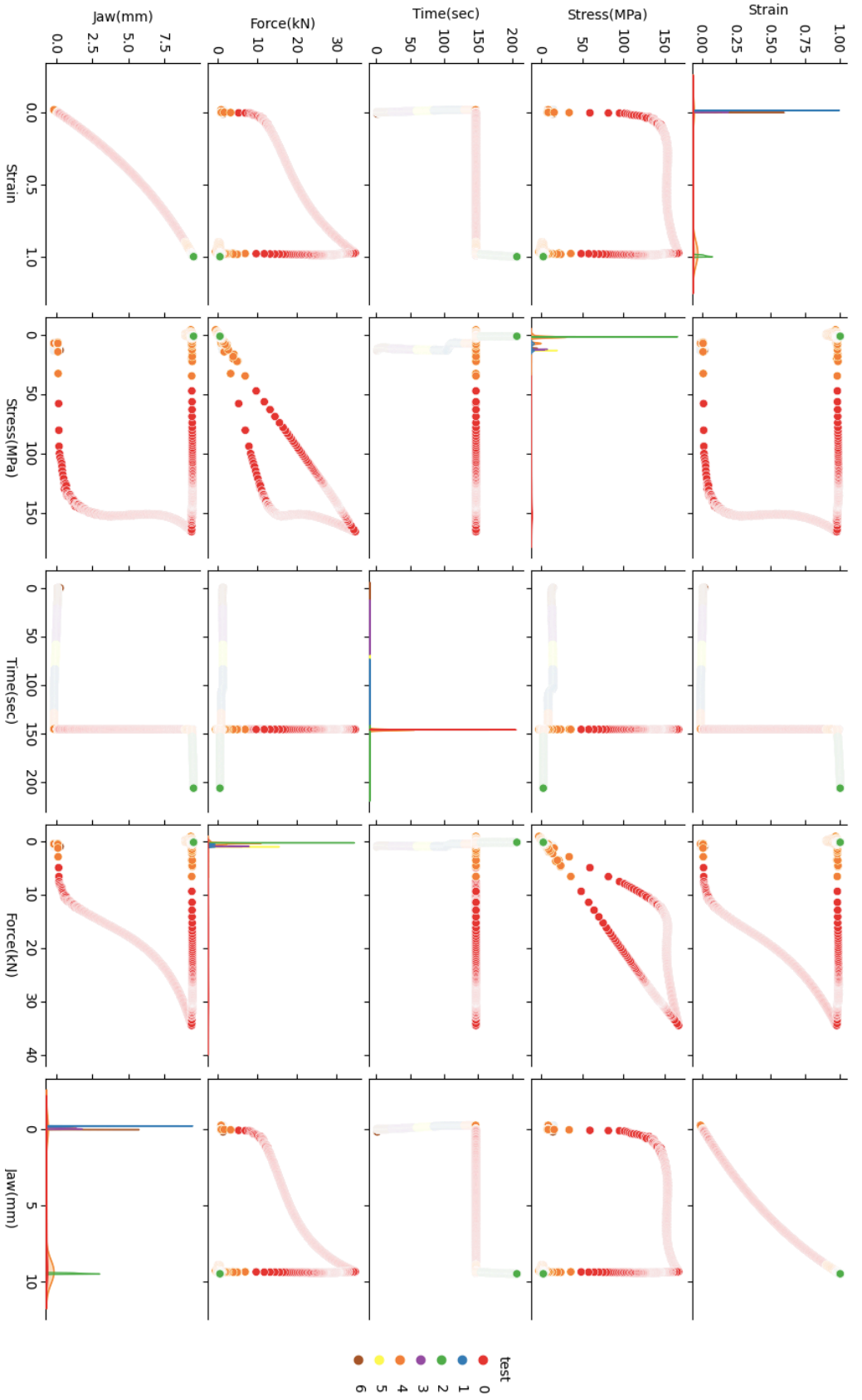
MiniBatchKMeans



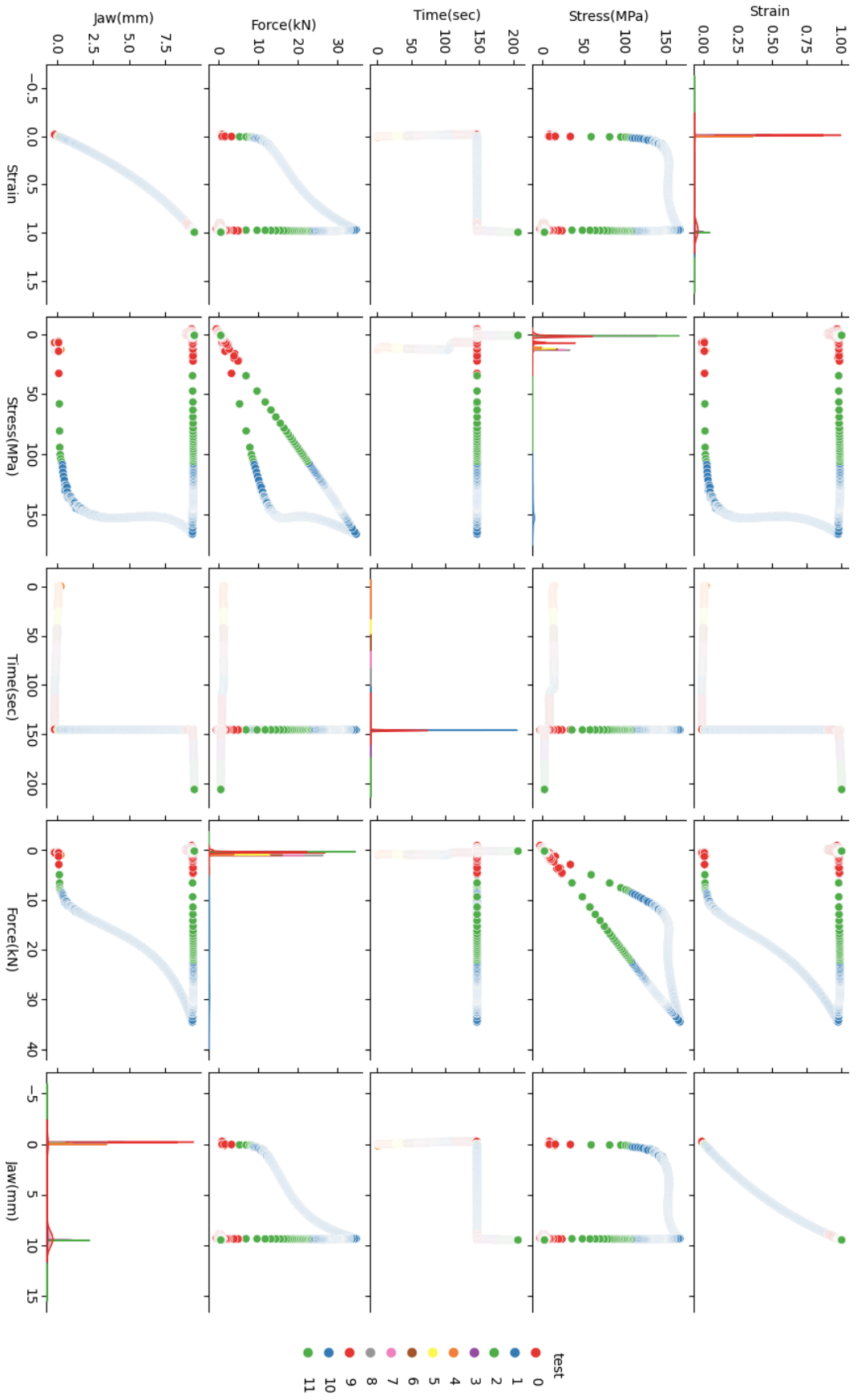
DSCAN



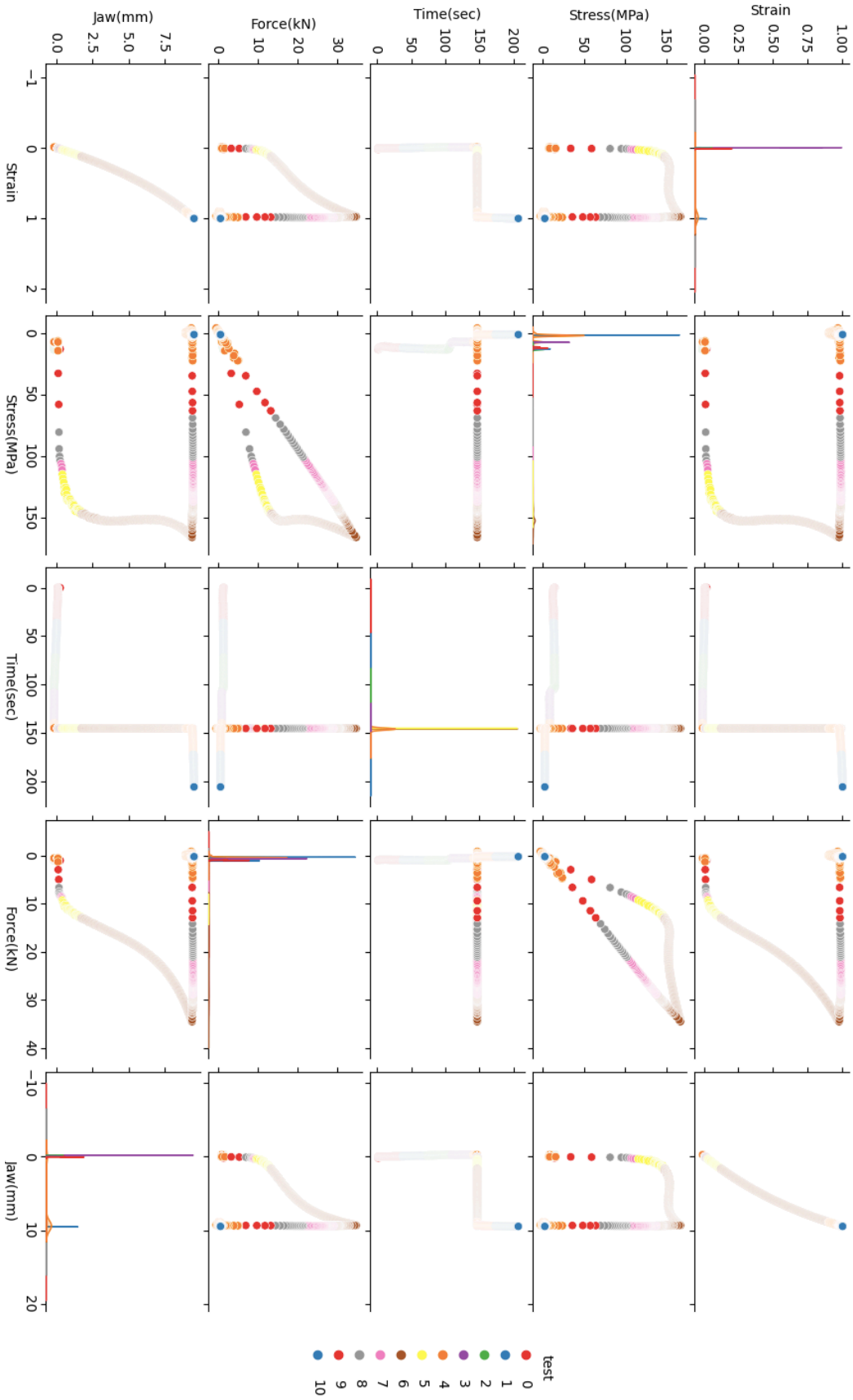
Agglomerative



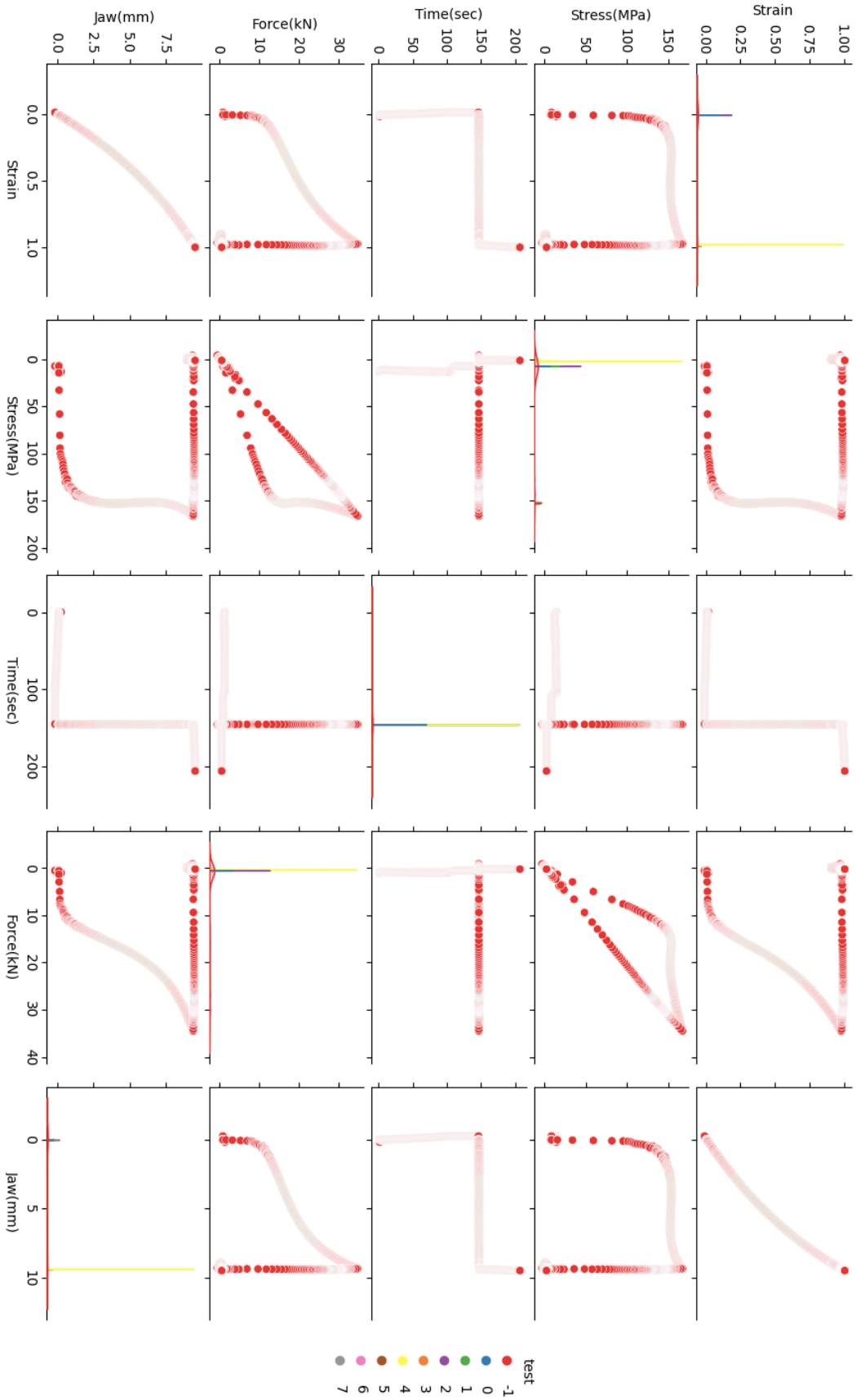
MeanShift


Strain
Stress(MPa)
Time(sec)
Force(kN)
Jaw(mm)
test
0
1
2
3
4
5
6
7
8
9
10
11

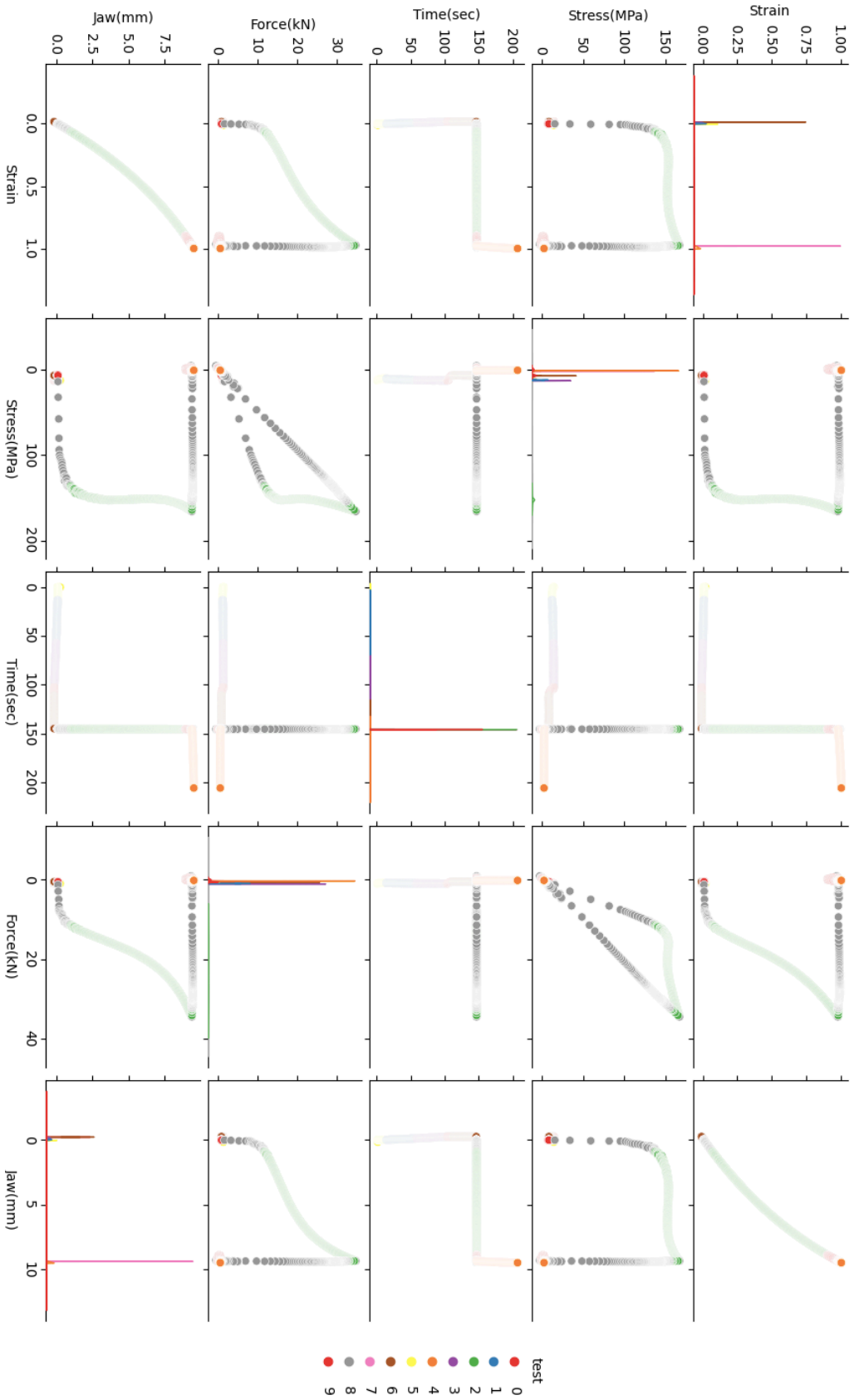
Birch


Strain
Stress(MPa)
Time(sec)
Force(kN)
Jaw(mm)
test
0
1
2
3
4
5
6
7
8
9
10

Optics



GaussianMixture



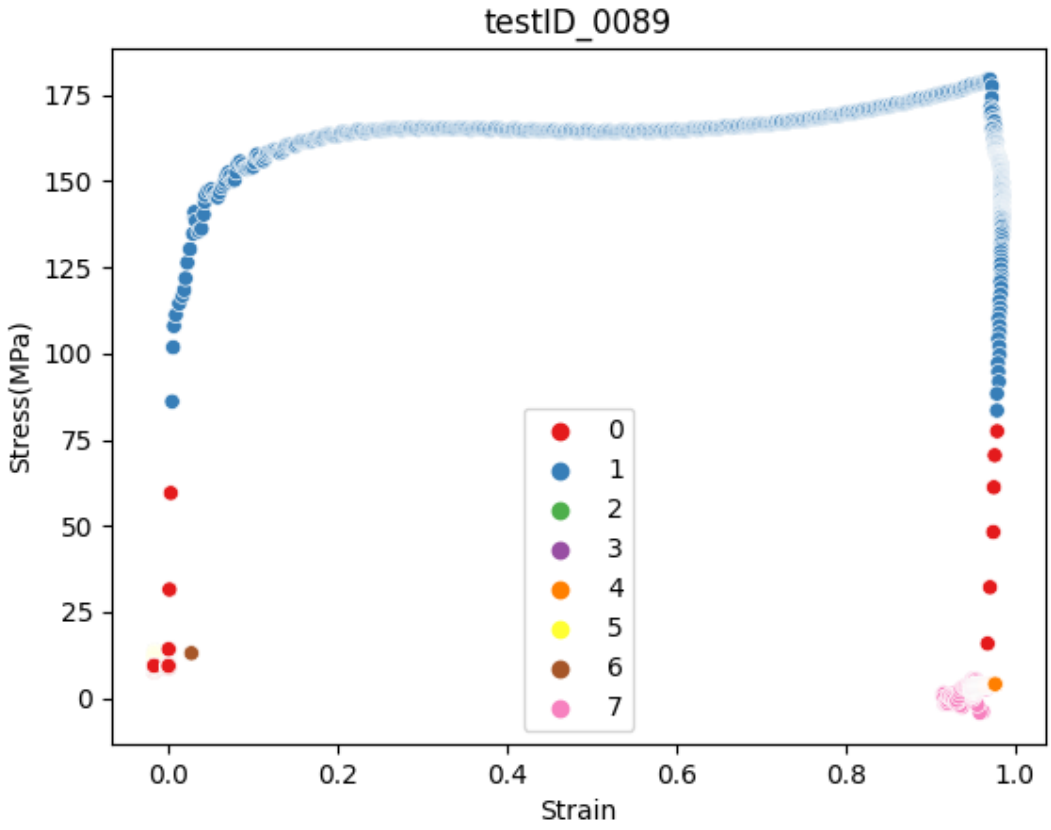
Appendix C

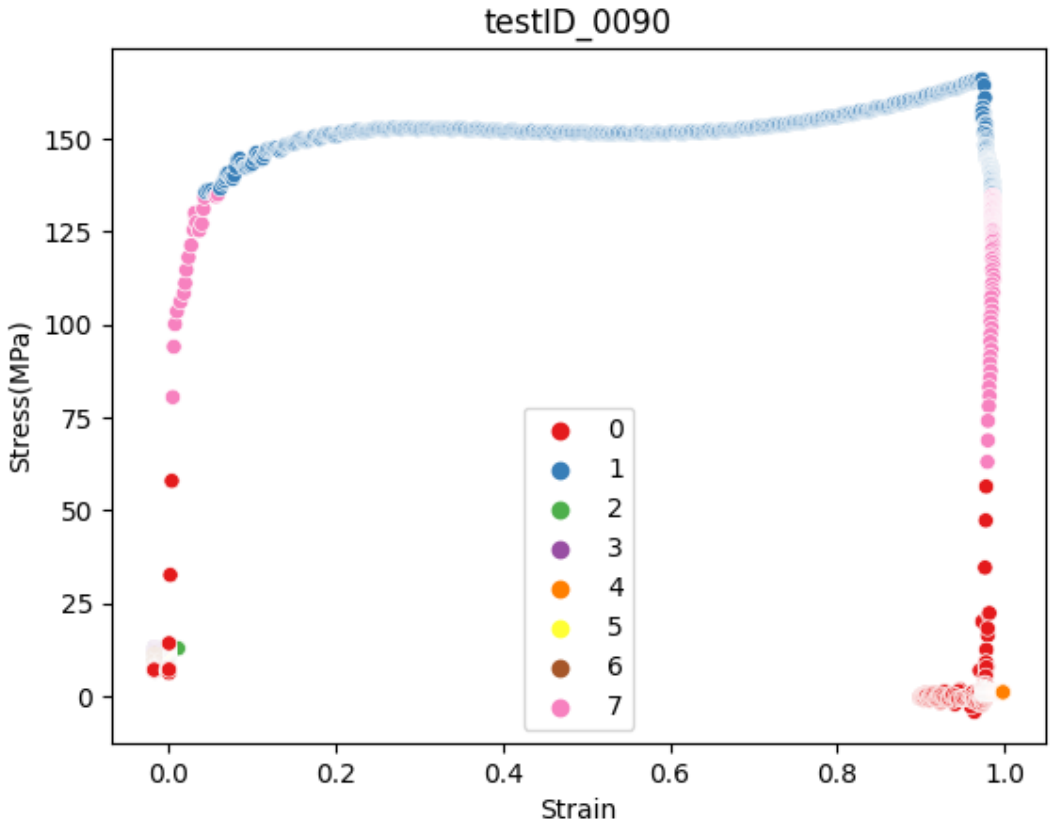
```
"""
This code was built by Katleho Mofokeng
```

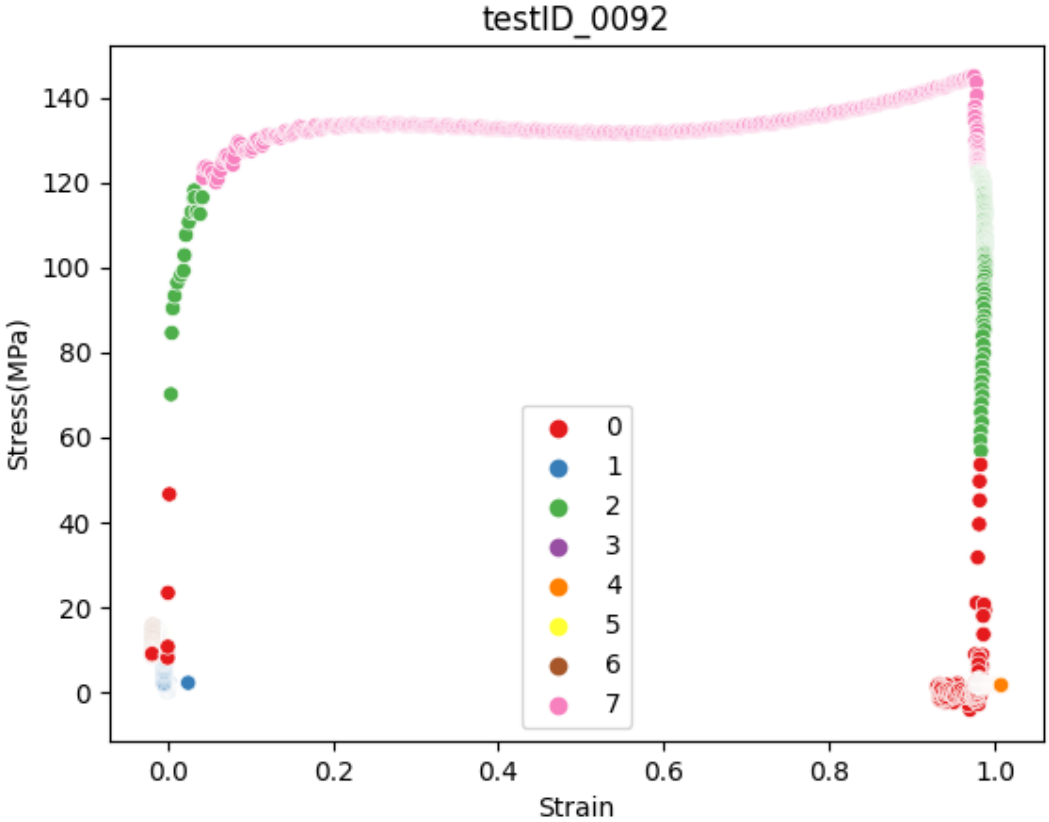
```
Katleho's student number is MFKKAT007
"""

import pandas as pd
# from matplotlib import pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import *
from sklearn.mixture import GaussianMixture


raw_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/01 screened data"


def screened_data(num):
    data = pd.read_csv(
        f"{raw_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


model_KMeans = KMeans(n_clusters=4, random_state=15)
model_MiniBatchKMeans = MiniBatchKMeans(n_clusters=8, random_state=15)
model_DBSCAN = DBSCAN(min_samples=20, eps=1)
model_AffinityPropagation = AffinityPropagation(preference=-100,
random_state=15)
model_Agglomerative = AgglomerativeClustering(n_clusters=7)
model_MeanShift = MeanShift(bandwidth=17)
model_Spectral = SpectralClustering(n_clusters=2, random_state=15)
model_Birch = Birch(branching_factor=100, n_clusters=None, threshold=10.5)
model_Optics = OPTICS(min_samples=25)  #
model_GaussianMixture = GaussianMixture(n_components=10, random_state=15)


def compare(model, data_entry):
    data = data_entry
    remove = model.fit_predict(data_entry)
    data["test"] = remove

    plt.figure(figsize=(8, 8))
    plt.title("Pair plot of test 0090")
    sns.pairplot(data_entry, hue='test', palette='Set1')
    plt.show()


compare(model_GaussianMixture, screened_data('0089'))
```

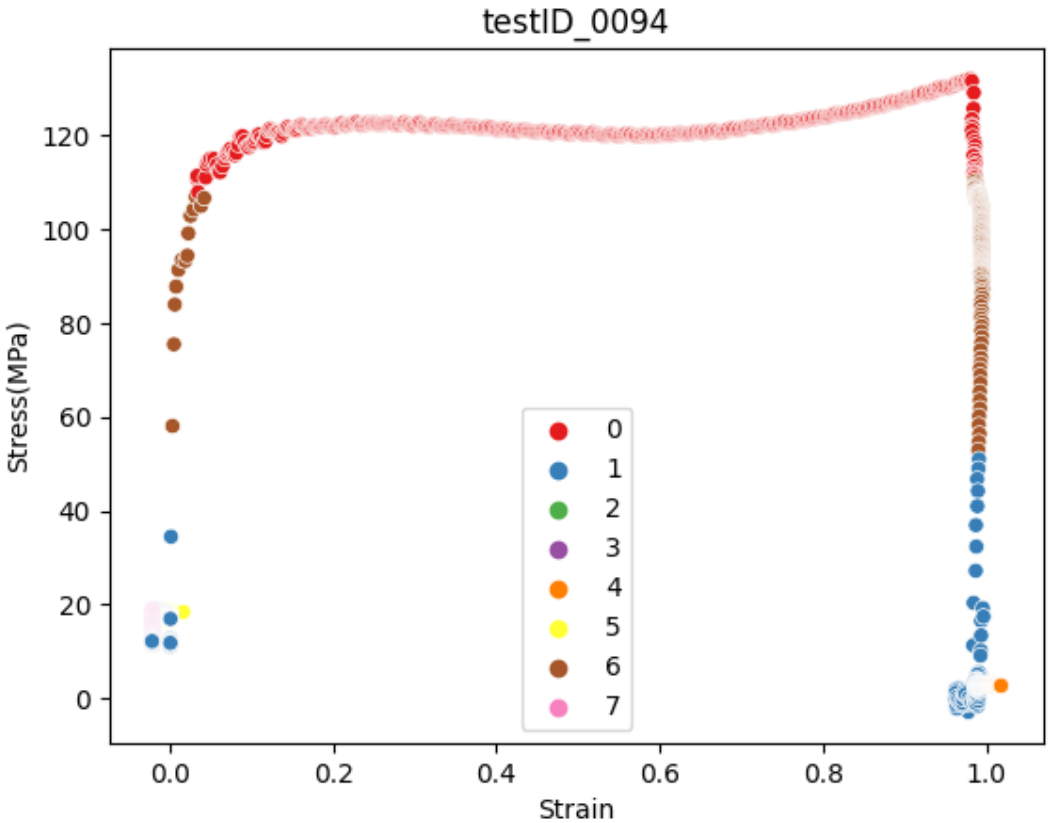
Appendix D

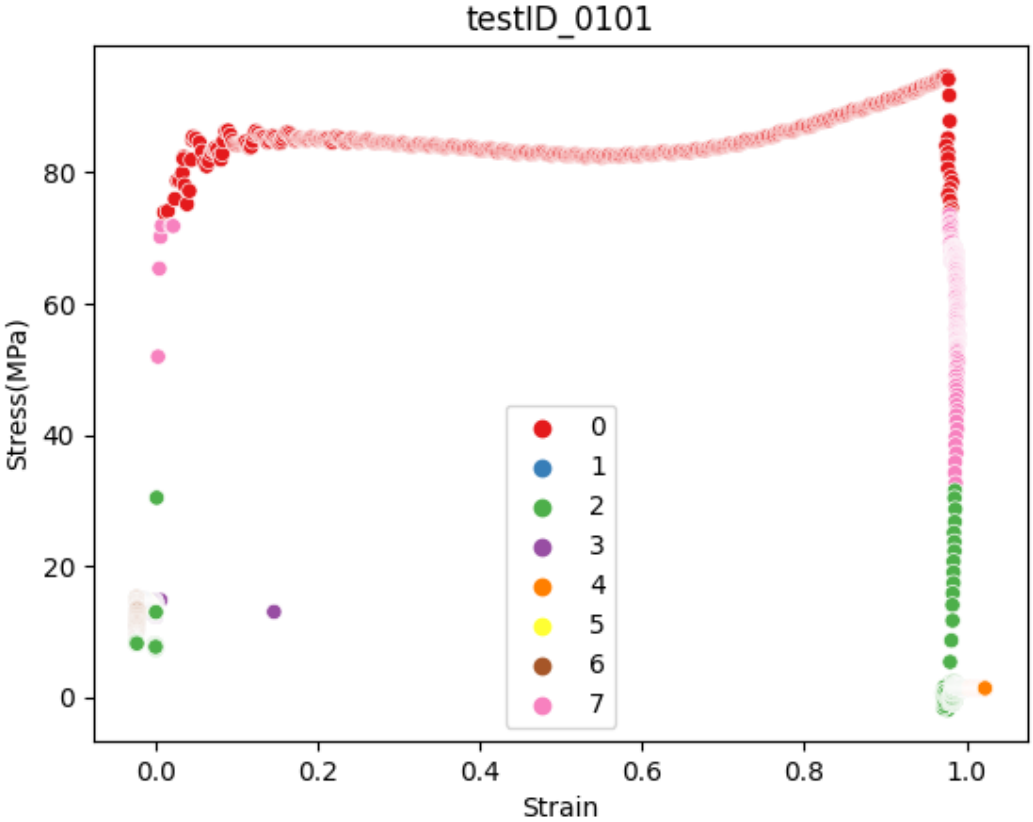
MiniBatchKMeans


testID_0089
Stress(MPa)
Strain
0
1
2
3
4
5
6
7

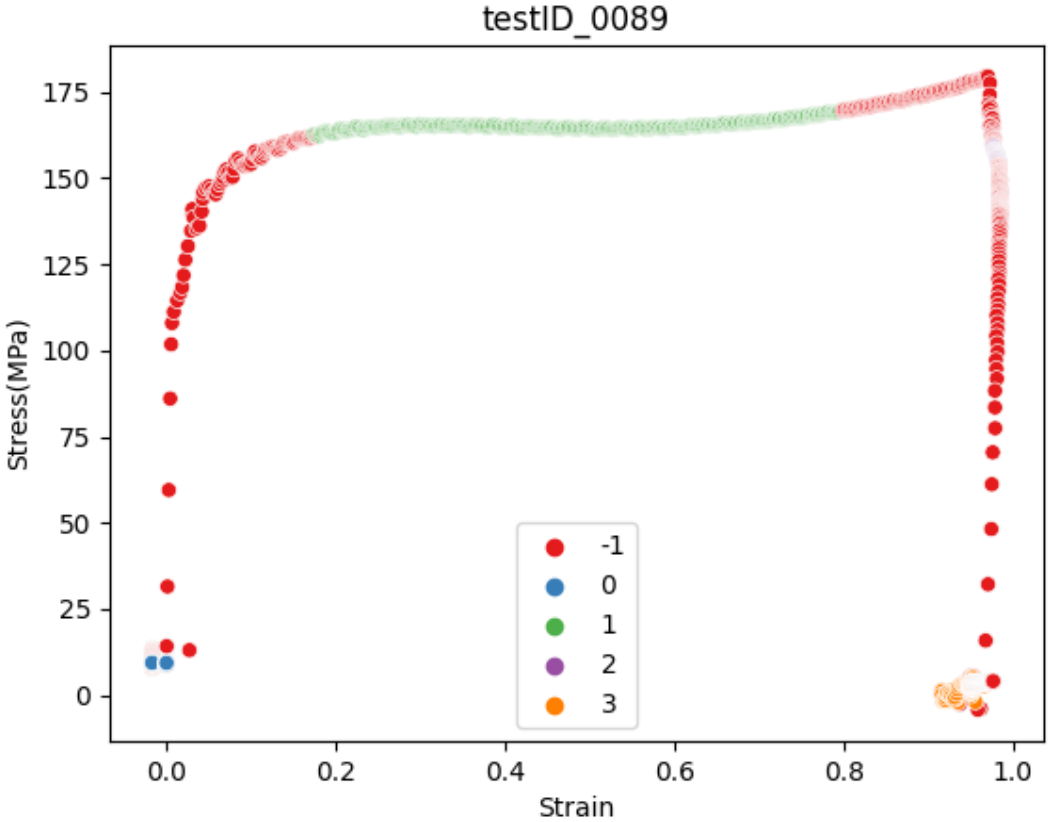

testID_0090
Stress(MPa)
Strain
0
1
2
3
4
5
6
7

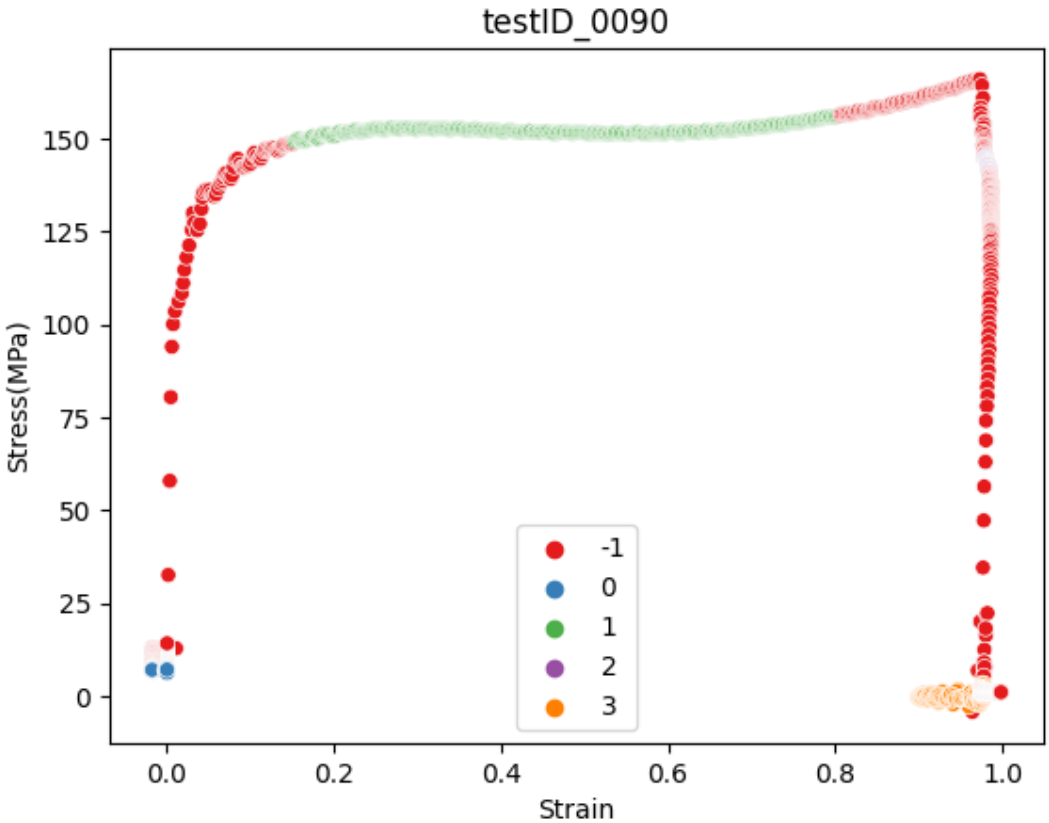

testID_0092
Stress(MPa)
Strain
0
1
2
3
4
5
6
7

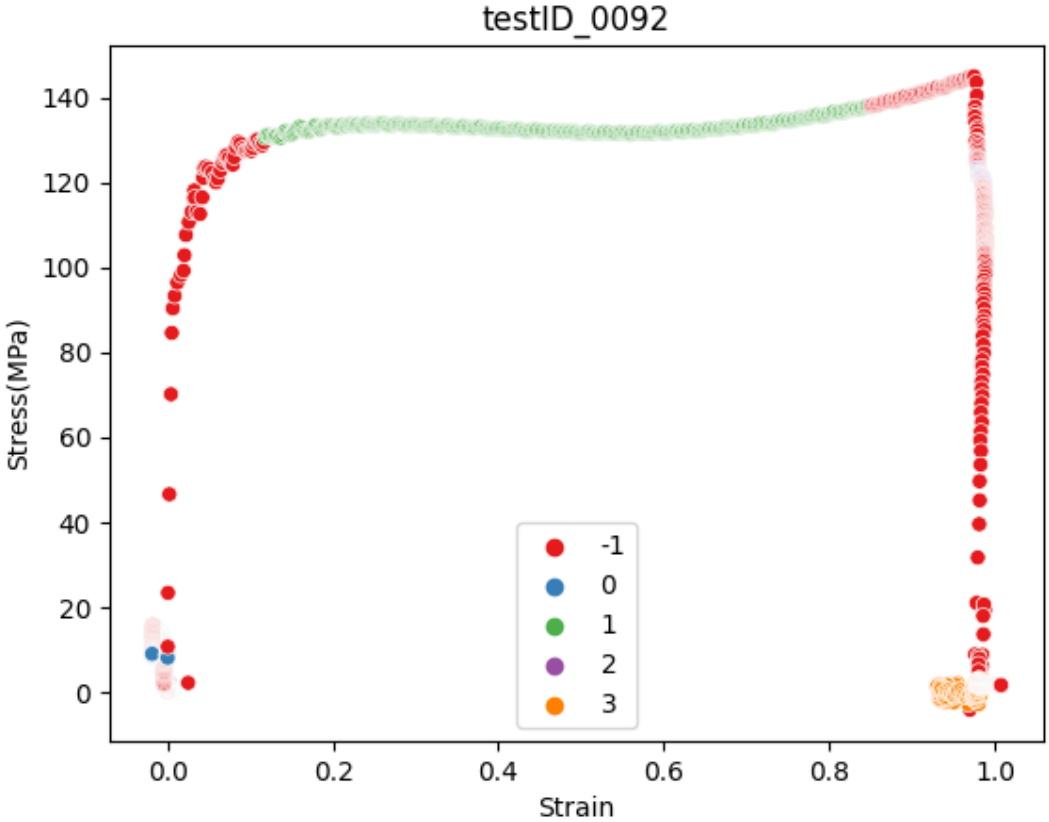

testID_0094
Stress(MPa)
Strain
0
1
2
3
4
5
6
7

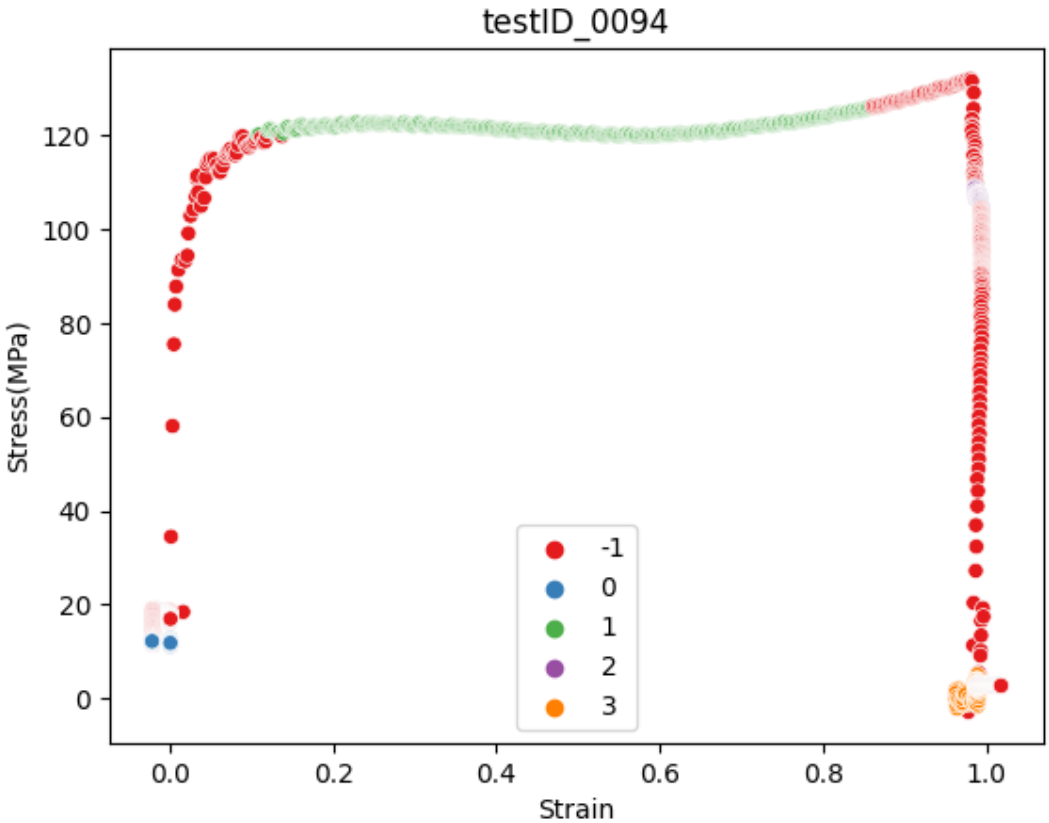

testID_0101
Stress(MPa)
Strain
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7

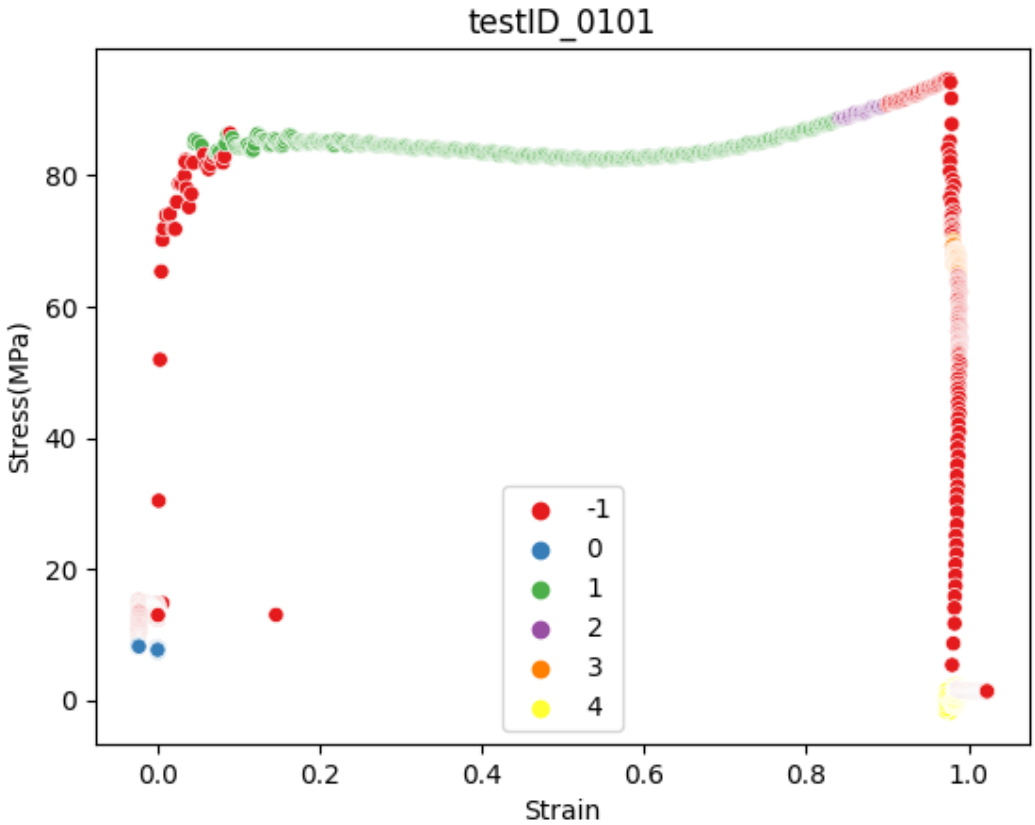
DBSCAN


testID_0089
Stress(MPa)
Strain
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0
-1
0
1
2
3

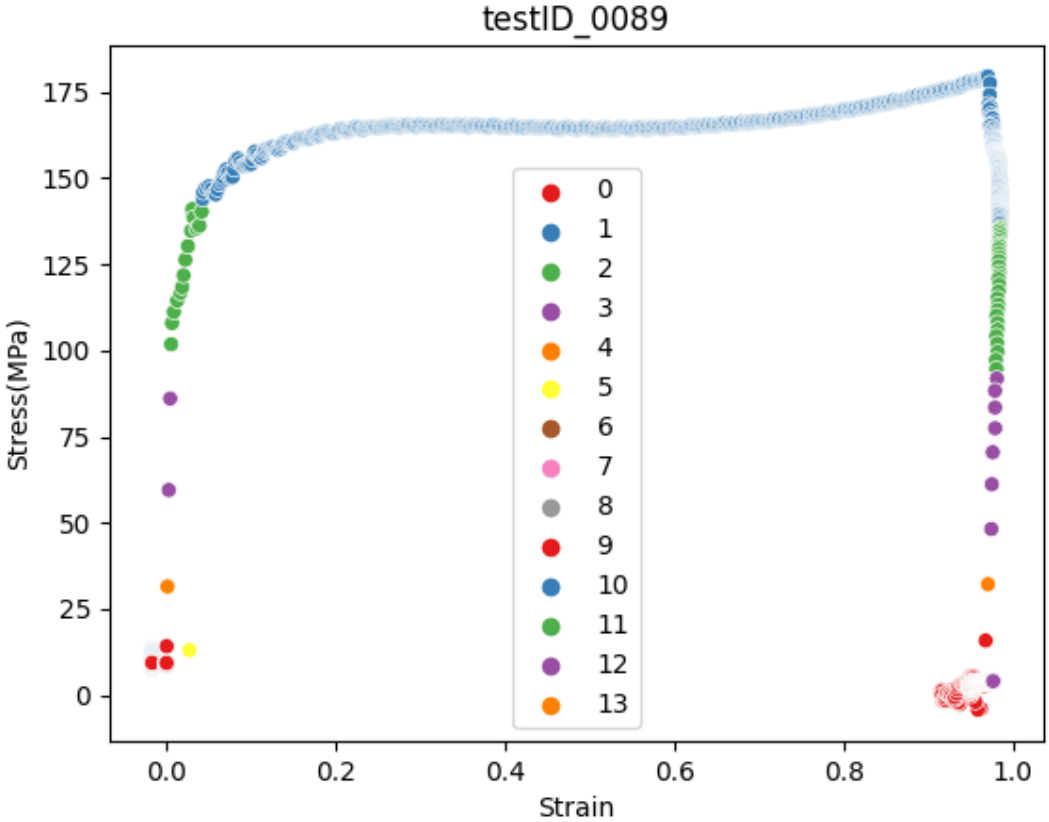

testID_0090
Stress(MPa)
Strain
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0
-1
0
1
2
3

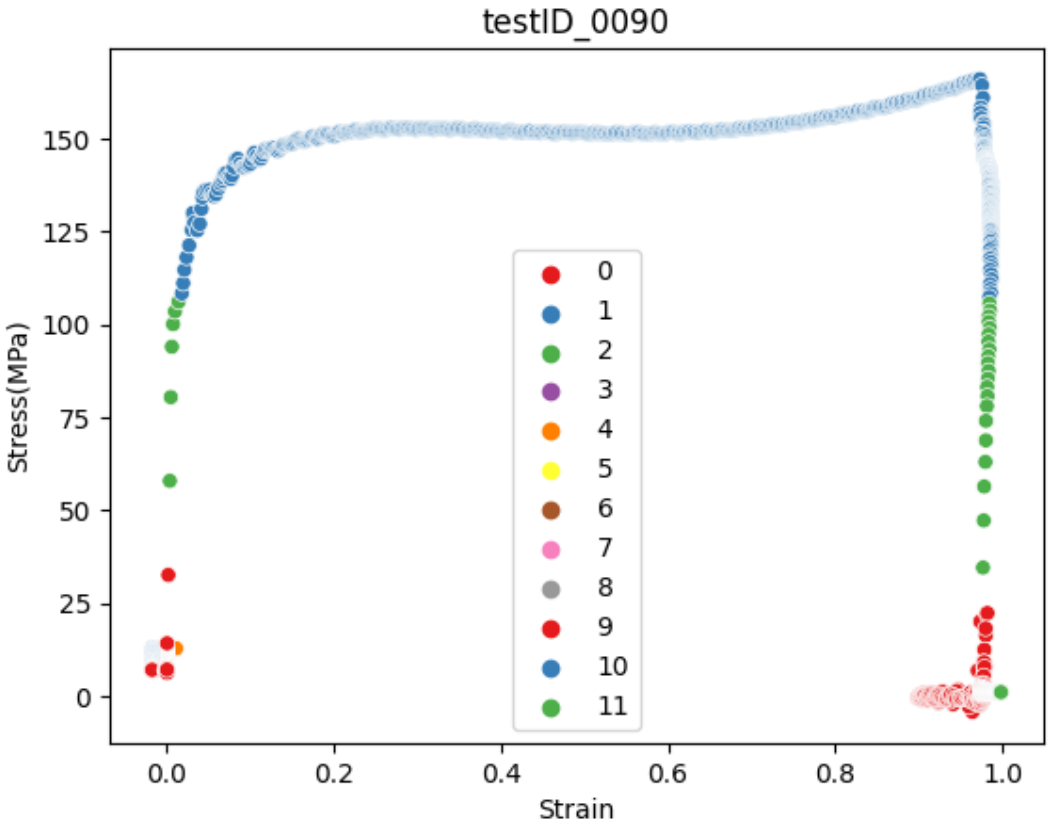

testID_0092
Stress(MPa)
Strain
-1
0
1
2
3

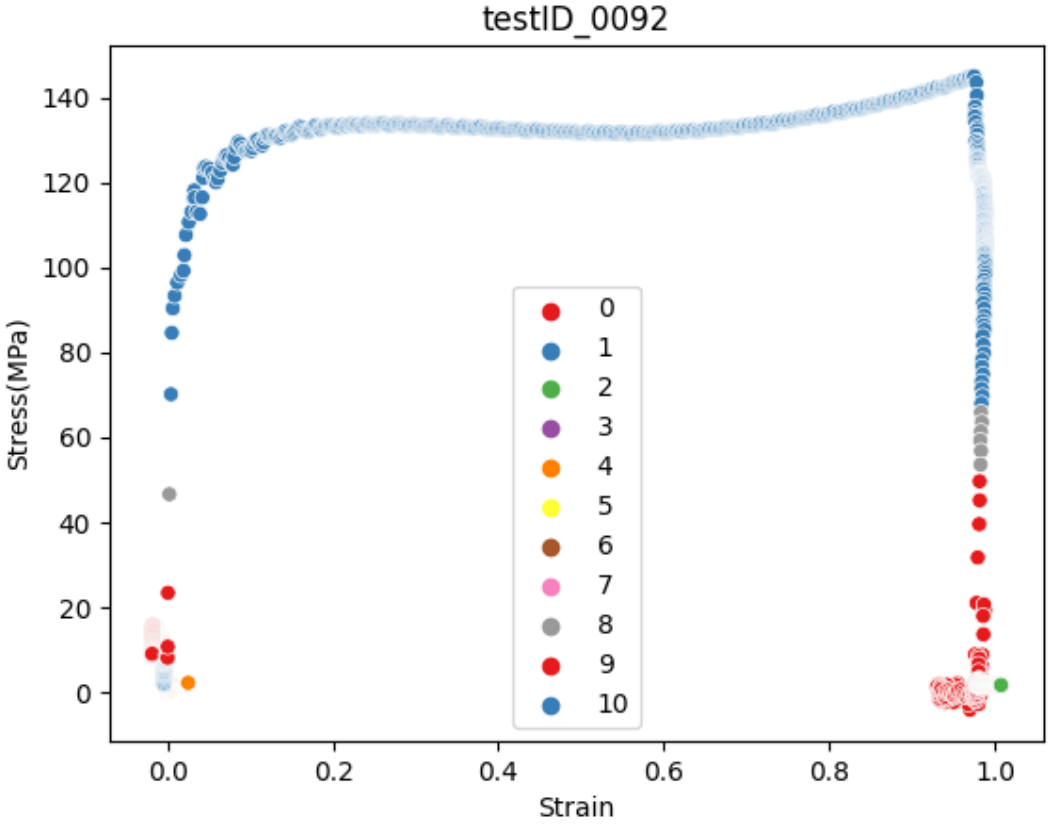

testID_0094
Stress(MPa)
Strain
-1
0
1
2
3

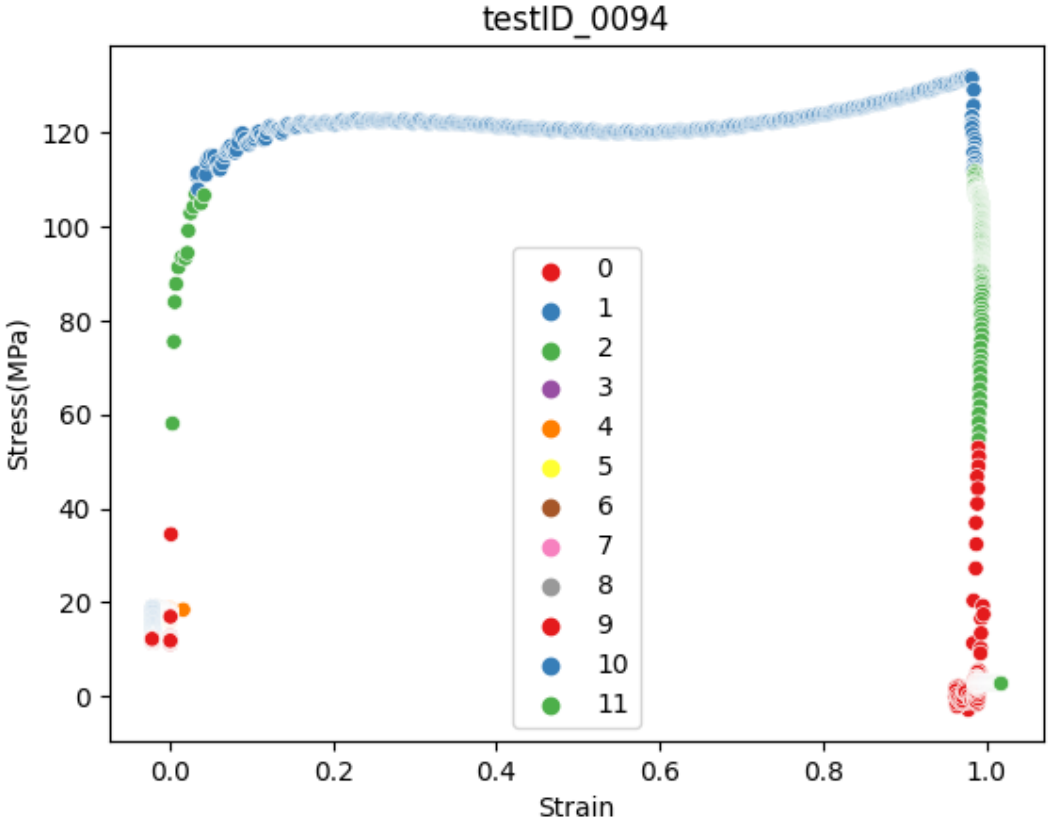

testID_0101
Stress(MPa)
Strain
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
-1
0
1
2
3
4

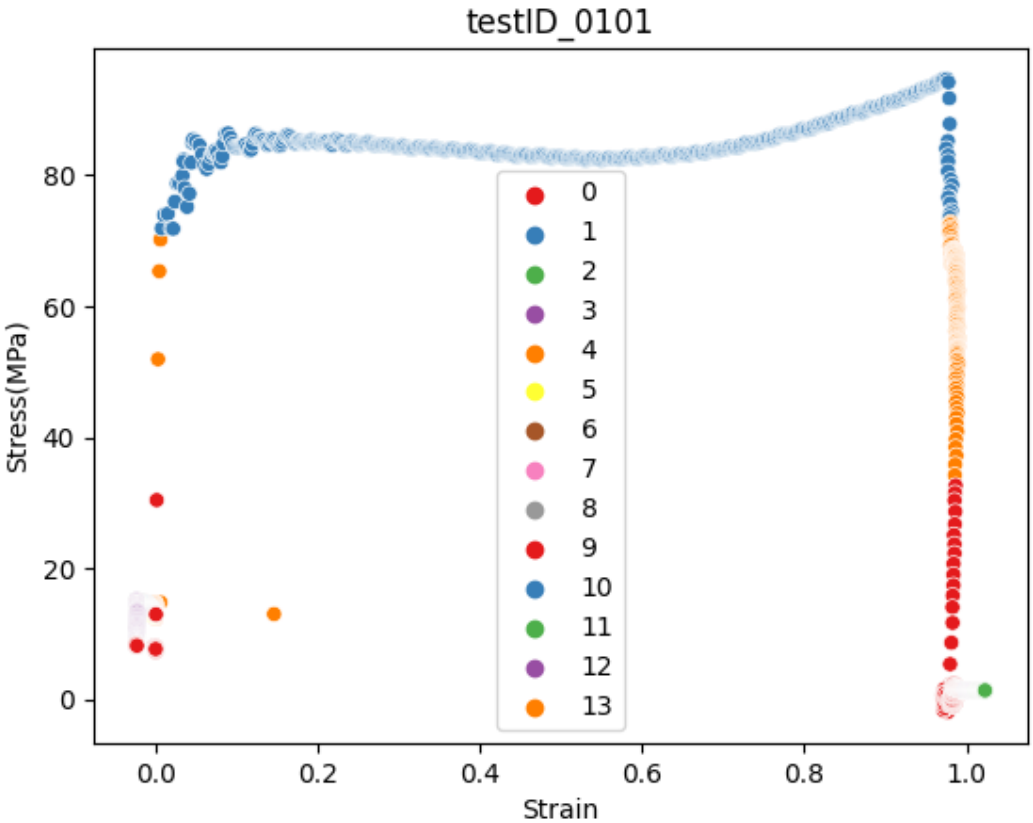
MeanShift


testID_0089
Stress(MPa)
Strain
0
1
2
3
4
5
6
7
8
9
10
11
12
13

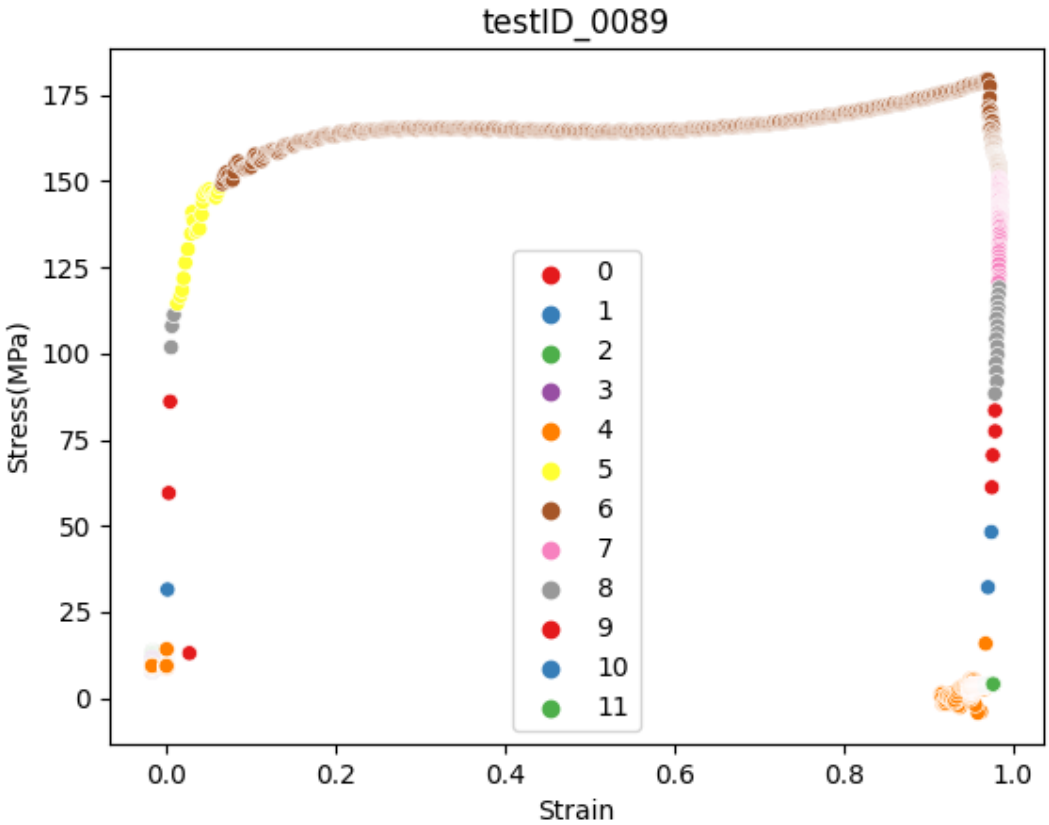

testID_0090
Stress(MPa)
Strain
0
1
2
3
4
5
6
7
8
9
10
11

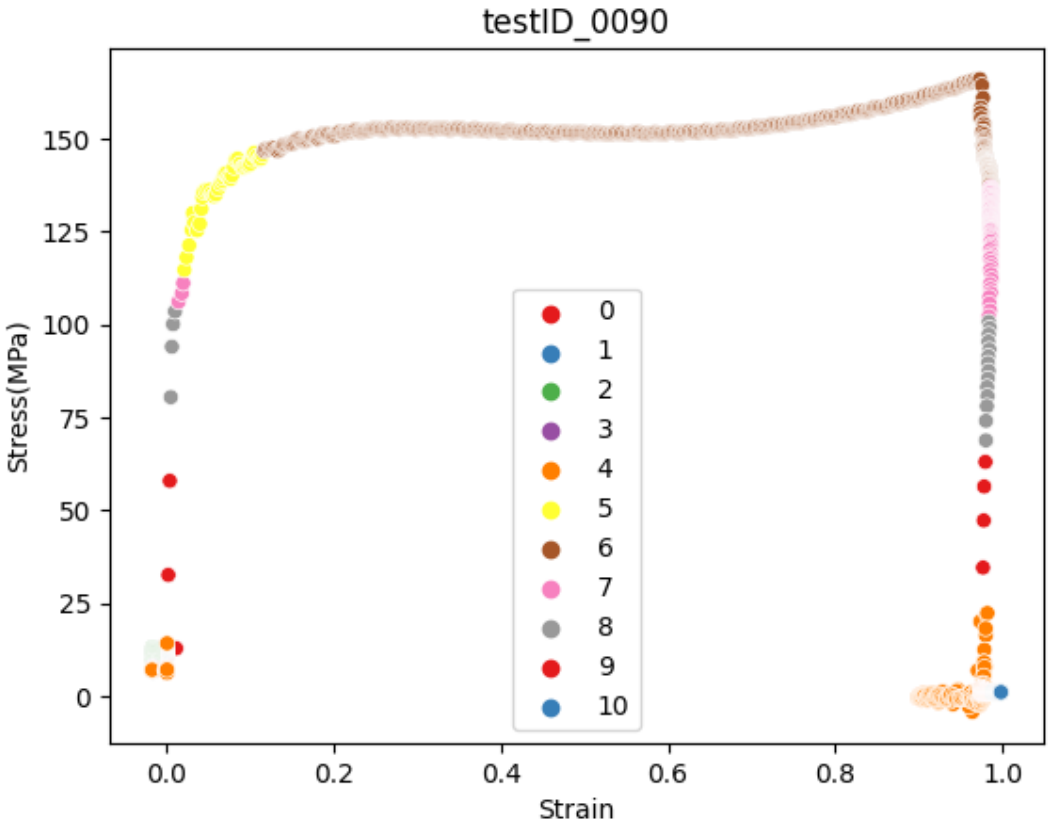

testID_0092
Stress(MPa)
Strain
140
120
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7
8
9
10

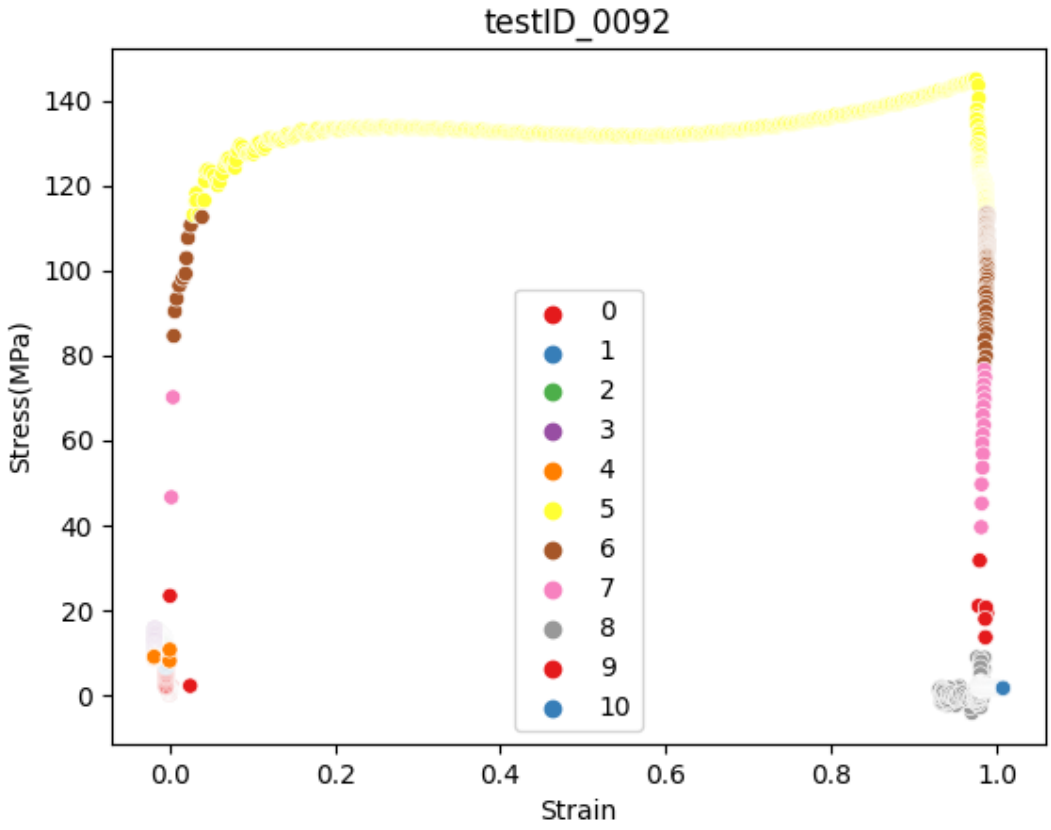

testID_0094
Stress(MPa)
Strain
0
20
40
60
80
100
120
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7
8
9
10
11

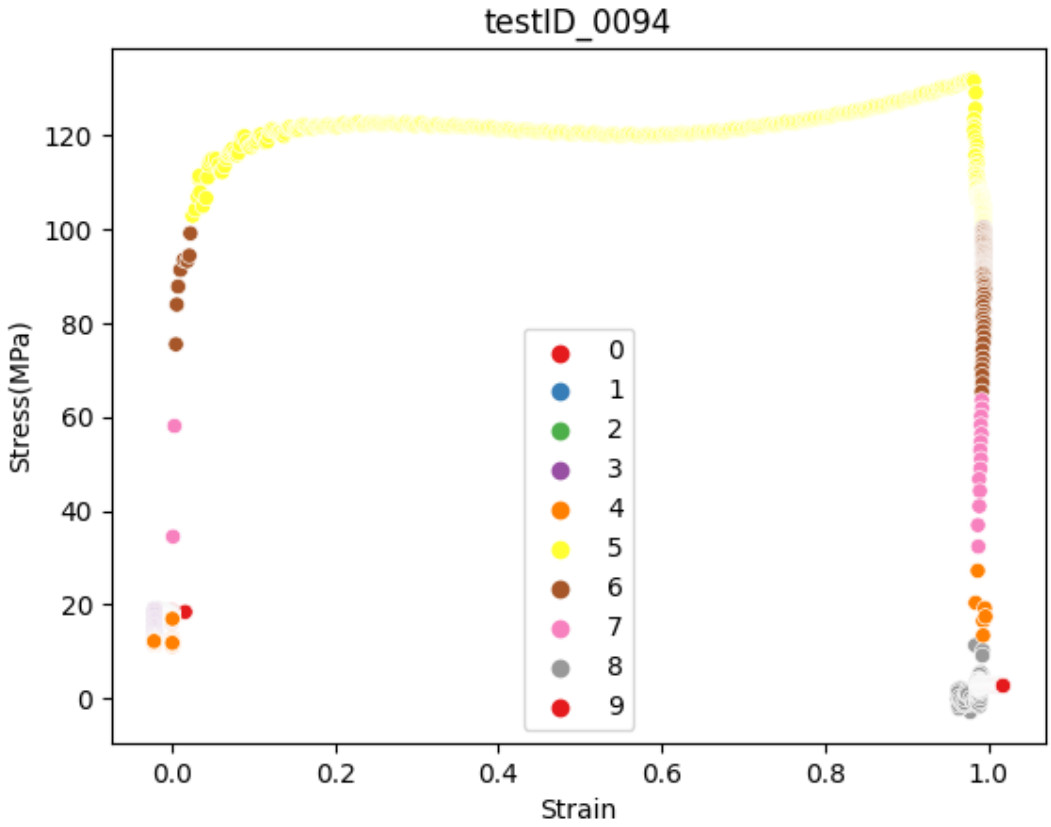

testID_0101
Stress(MPa)
Strain
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13

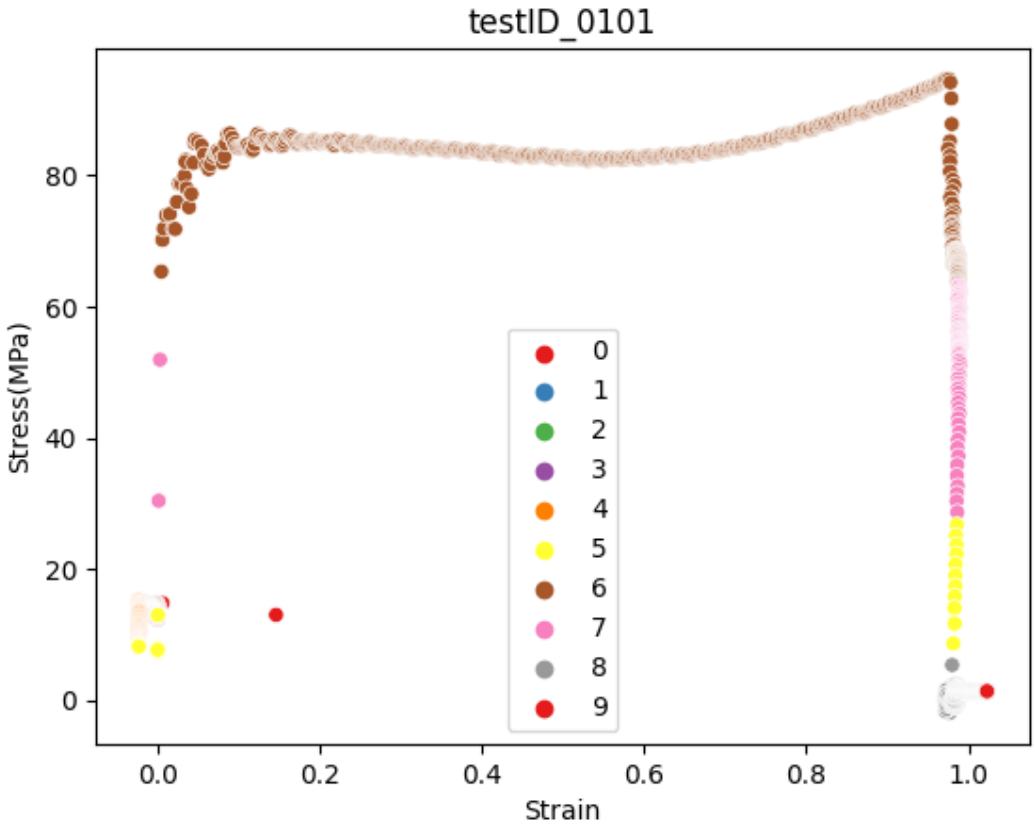
Birch


testID_0089
Stress(MPa)
Strain
0
1
2
3
4
5
6
7
8
9
10
11

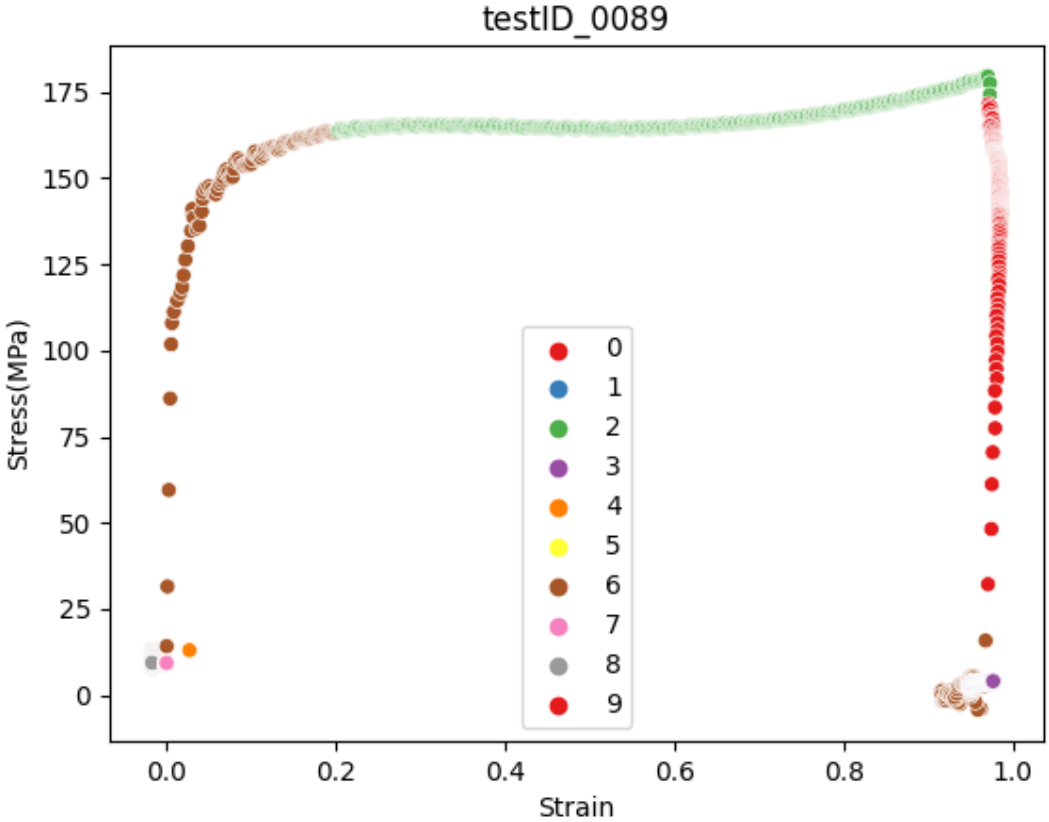

testID_0090
Stress(MPa)
Strain
0
1
2
3
4
5
6
7
8
9
10

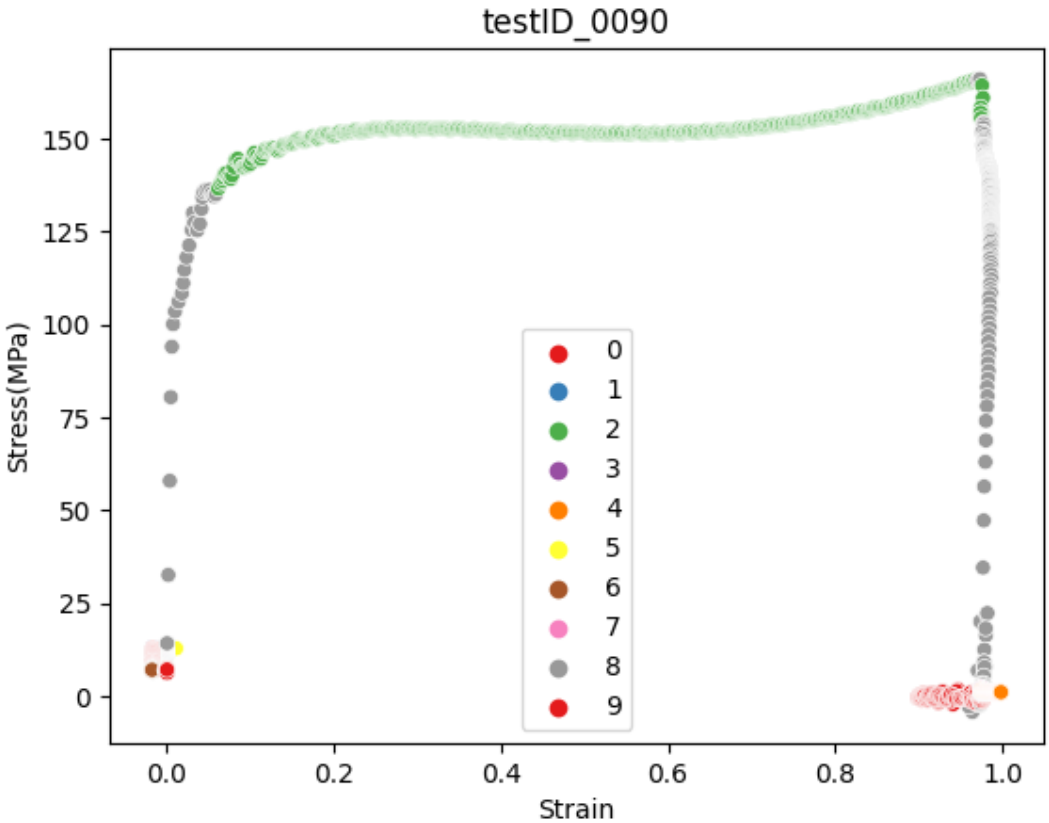

testID_0092
Stress(MPa)
Strain
0
1
2
3
4
5
6
7
8
9
10

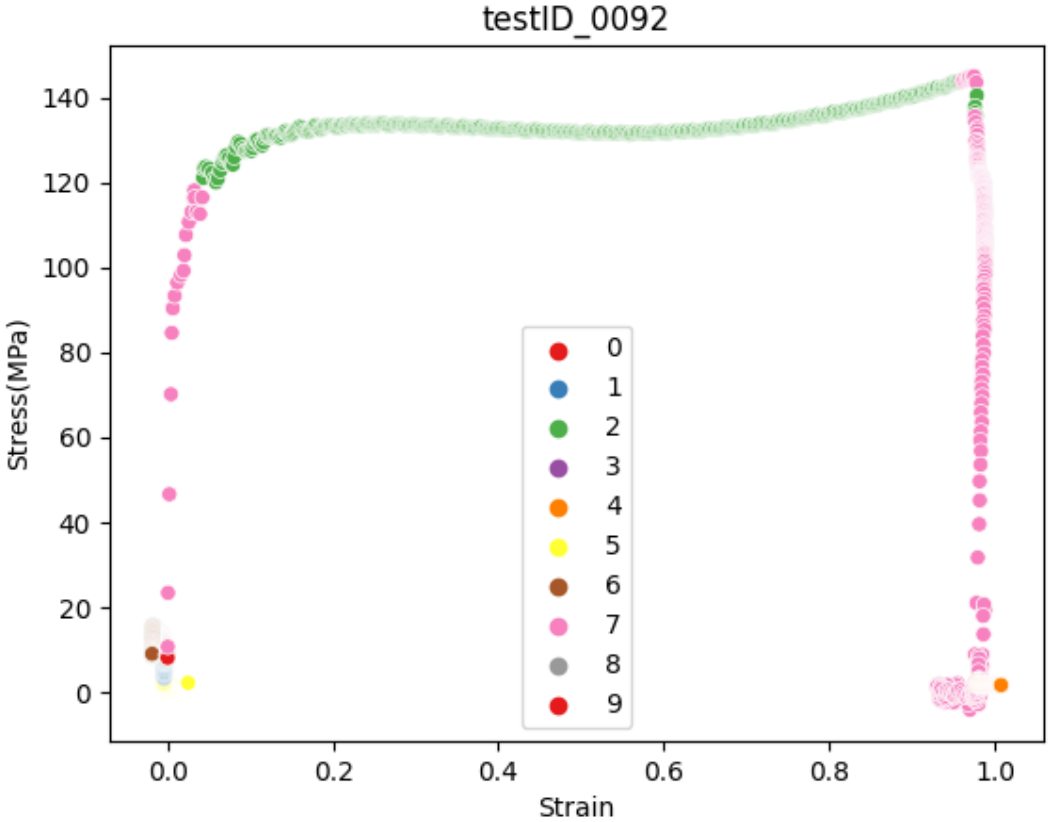

testID_0094
Stress(MPa)
Strain
0
1
2
3
4
5
6
7
8
9

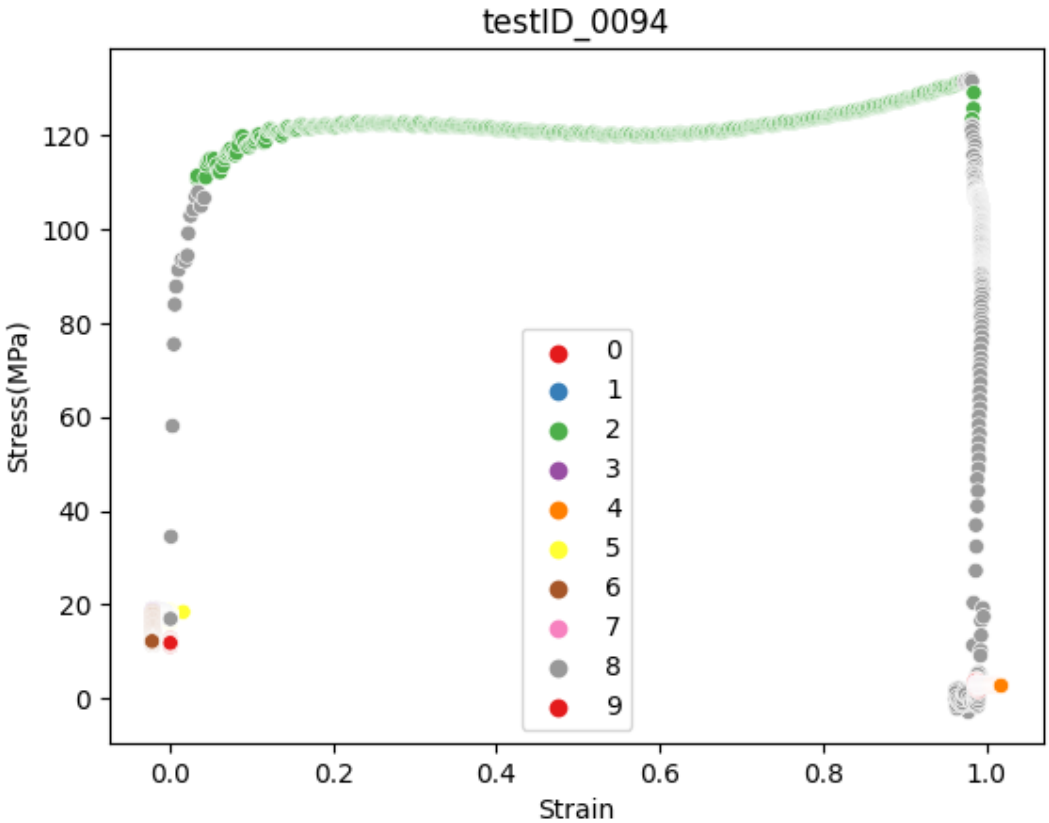


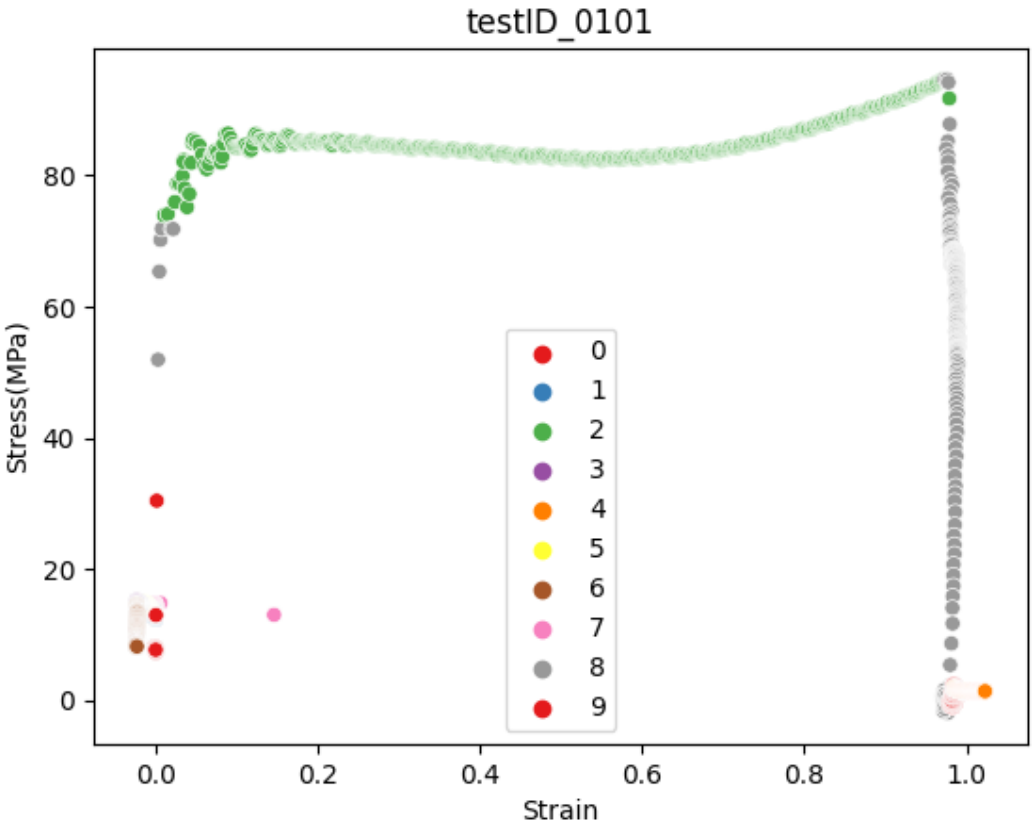
GaussianMixture


testID_0089
Stress(MPa)
Strain
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7
8
9


testID_0090
Stress(MPa)
Strain
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6
7
8
9


testID_0092
Stress(MPa)
Strain
0
1
2
3
4
5
6
7
8
9


testID_0094
Stress(MPa)
Strain
0
1
2
3
4
5
6
7
8
9



Appendix E

```
"""
This code was built by Katleho Mofokeng
Katleho's student number is MFKKAT007
"""

import pandas as pd
# from matplotlib import pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import *
from sklearn.mixture import GaussianMixture


raw_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/01 screened data"


def screened_data(num):
    data = pd.read_csv(
        f"{raw_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data
```

```
model_KMeans = KMeans(n_clusters=4, random_state=15)
model_MiniBatchKMeans = MiniBatchKMeans(n_clusters=8, random_state=15)
model_DBSCAN = DBSCAN(min_samples=20, eps=1)
model_AffinityPropagation = AffinityPropagation(preference=-100,
random_state=15)
model_Agglomerative = AgglomerativeClustering(n_clusters=7)
model_MeanShift = MeanShift(bandwidth=17)
model_Spectral = SpectralClustering(n_clusters=2, random_state=15)
model_Birch = Birch(branching_factor=100, n_clusters=None, threshold=10.5)
model_Optics = OPTICS(min_samples=25)  #
model_GaussianMixture = GaussianMixture(n_components=10, random_state=15)


def display(model, data):
    groups = model.fit_predict(data)
    plt.title('0089')
    sns.scatterplot(x="Strain", y="Stress(MPa)", data=data, hue=groups,
palette='Set1')
    plt.show()


display(model_MiniBatchKMeans, screened_data('0089'))
```

Appendix F

```
"""
This code was built by Katleho Mofokeng
Katleho's student number is MFKKATOO7
"""

import pandas as pd
# from matplotlib import pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import *

raw_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/01 screened data"


def screened_data(num):
    data = pd.read_csv(
        f"{raw_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def opt_1(start, data):
    yes = []
    no = []

    num_1 = start
    num_2 = start + 1

    before = data["Stress(MPa)"][num_1]
    after = data["Stress(MPa)"][num_2]
```

```
    count = 0
    while True:
        if float(before) > float(after):
            yes.append(1)
        else:
            no.append(0)

        num_1 = num_1 + 1
        num_2 = num_2 + 1

        before = data["Stress(MPa)"][num_1]
        after = data["Stress(MPa)"][num_2]

        if (len(yes) + len(no)) == 6:
            if len(no) > 0:
                count = count + 1
                num_1 = start + count
                num_2 = start + count + 1

                before = data["Stress(MPa)"][num_1]
                after = data["Stress(MPa)"][num_2]

                yes = []
                no = []
            else:
                break

    remove = count + start
    return remove


def opt_lower(start, data):
    num_1 = start
    num_2 = start - 1
    after = data["Stress(MPa)"][num_1]
    before = data["Stress(MPa)"][num_2]
    count = 0

    while True:
        if float(before) > float(after):
            count = count + 1
            num_1 = start - count
            num_2 = start - 1 - count

            before = data["Stress(MPa)"][num_1]
            after = data["Stress(MPa)"][num_2]

        else:
            # print("Done, count is =", count)
            break

    remove = start - count
    # print("end =", remove)
    return remove


def objective_2(predict, data):
    mark = -1
    while True:
        mark = mark + 1
```

```
        if predict[mark] == 0:
            break

        mark = mark
    while True:
        mark = mark + 1
        if predict[mark] != 0:
            break

    median = (data["Strain"][len(data) - 1]) / 2

    mid = 0
    while True:
        mid = mid + 1
        if (abs(float(median - data['Strain'][mid]))) < 0.01:
            break

    num = predict[mid]
    upper = mid
    while True:
        upper = upper + 1
        if predict[upper] != num:
            break

    new_list = []
    remove = upper - 40

    remove = opt_1(remove, data)
    count = -1
    while count < len(predict) - 1:
        count = count + 1
        if count < mark:
            new_list.append(0)
        elif count > remove:
            new_list.append(0)
        else:
            new_list.append(1)
    return new_list


def clean_noise(data):
    model = DBSCAN(min_samples=20, eps=1)
    predict = model.fit_predict(data)

    predict = objective_2(predict, data)

    data["Clusters"] = predict
    data_2 = data[data.Clusters != 0]
    data_2 = data_2.drop(['Clusters'], axis=1)

    plt.title("testID_0128")
    sns.scatterplot(x="Strain", y="Stress(MPa)", data=data, hue=predict,
palette='Set1')
    plt.show()
    sns.scatterplot(x="Strain", y="Stress(MPa)", data=data_2,
palette='Set1')
    plt.show()


clean_noise(screened_data('0128'))
```

Appendix G

```
"""
This code was built by Katleho Mofokeng
Katleho's student number is MFKKAT007
"""

import pandas as pd
# from matplotlib import pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression


raw_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/01 screened data"

trimmed_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/02 trimmed " \
                      "data"


def screened_data(num):
    data = pd.read_csv(
        f"{raw_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def trimmed_data(num):
    data = pd.read_csv(f"{trimmed_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def clusters(data):
    removed = []
    for i in data:
        if i < -0.05:
            removed.append(1)
        elif i <= 0.1:
            removed.append(2)
        elif i <= 0.13:
            removed.append(3)
        elif i < 0.2:
            removed.append(4)
        elif i < 0.35:
            removed.append(5)
        else:
            removed.append(0)

    return removed
```

```
def y_point(data_s, data_t):
    data_trim = []
    for i in data_t['Strain']:
        data_trim.append(i)

    removed = []
    for i in data_s['Strain']:
        if i in data_trim:
            removed.append(1)
        else:
            removed.append(0)

    return removed


def clean_noise(data_s_2):
    X_train = screened_data('0089')[["Strain", "Stress(MPa)"]]
    y_train = y_point(screened_data('0089'), trimmed_data('0089'))

    X_test = data_s_2[["Strain", "Stress(MPa)"]]
    y_test = y_point(screened_data("0095"), trimmed_data("0095"))

    model = PolynomialFeatures(degree=7)

    X_train = model.fit_transform(X_train)
    X_test = model.fit_transform(X_test)

    algo = LinearRegression()
    algo.fit(X_train, y_train)

    predict = algo.predict(X_test)
    predict = clusters(predict)

    data_s_2["Clusters"] = predict
    data_2 = data_s_2[data_s_2.Clusters != 0]
    data_2 = data_2.drop(['Clusters'], axis=1)

    plt.title("testID_0090")
    sns.scatterplot(x="Strain", y="Stress(MPa)", data=data_s_2,
hue=predict, palette='Set1')
    plt.show()


clean_noise(screened_data('0090'))
```

Appendix H

```
"""
This code was built by Katleho Mofokeng
Katleho's student number is MFKKAT007
"""

import pandas as pd
# from matplotlib import pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

```
raw_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/01 screened data"

trimmed_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/02 trimmed " \
                      "data"


def screened_data(num):
    data = pd.read_csv(
        f"{raw_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def trimmed_data(num):
    data = pd.read_csv(f"{trimmed_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def clusters(data):
    removed = []
    for i in data:
        if i < -0.05:
            removed.append(1)
        elif i <= 0.1:
            removed.append(2)
        elif i <= 0.13:
            removed.append(3)
        elif i < 0.2:
            removed.append(4)
        elif i < 0.35:
            removed.append(5)
        else:
            removed.append(0)

    return removed


def y_point(data_s, data_t):
    data_trim = []
    for i in data_t['Strain']:
        data_trim.append(i)

    removed = []
    for i in data_s['Strain']:
        if i in data_trim:
            removed.append(1)
        else:
            removed.append(0)

    return removed


def opt_1(start, data):
    yes = []
```

```
    no = []

    num_1 = start
    num_2 = start + 1

    before = data["Stress(MPa)"][num_1]
    after = data["Stress(MPa)"][num_2]
    #print("start ", start)
    count = 0
    while True:
        if float(before) > float(after):
            yes.append(1)
        else:
            no.append(0)

        num_1 = num_1 + 1
        num_2 = num_2 + 1

        before = data["Stress(MPa)"][num_1]
        after = data["Stress(MPa)"][num_2]

        if (len(yes) + len(no)) == 6:
            if len(no) > 0:
                count = count + 1
                num_1 = start + count
                num_2 = start + count + 1

                before = data["Stress(MPa)"][num_1]
                after = data["Stress(MPa)"][num_2]

                yes = []
                no = []
            else:
                #print("mark =", start)
                #print("stop =", num_1)
                break

    remove = count + start
    #print("end =", remove)
    return remove


def opt_lower(start, data):

    num_1 = start
    num_2 = start - 1
    #print("check")
    before = data["Stress(MPa)"][num_1]
    after = data["Stress(MPa)"][num_2]
    #print("start ", start)
    count = 0

    while True:
        if float(before) > float(after):
            count = count + 1
            num_1 = start - count
            num_2 = start - 1 - count

            before = data["Stress(MPa)"][num_1]
            after = data["Stress(MPa)"][num_2]
```

```
        else:
            #print("Done, count is =", count)
            break

    remove = start - count
    #print("end =", remove)
    return remove


def objective_2(data_entry, data):
    mark = -1
    while True:
        mark = mark + 1
        if data_entry[mark] == 0:
            break

    if set(data_entry) == {0, 1, 2, 3, 4, 5}:
        remove1 = mark
        while True:
            remove1 = remove1 + 1
            if data_entry[remove1] == 5:
                break

        remove1 = opt_1(remove1, data)
        #mark = opt_lower(mark, data)
        try:
            mark = opt_lower(mark, data)
        except Exception:
            mark = mark
        new_list = []
        count = -1
        while count < len(data_entry) - 1:
            count = count + 1
            if count < mark:
                new_list.append(0)
            elif count > remove1:
                new_list.append(0)
            else:
                new_list.append(1)
    else:
        remove1 = mark
        while True:
            remove1 = remove1 + 1
            if data_entry[remove1] == 5:
                break

        remove1 = opt_1(remove1, data)
        #mark = opt_lower(mark, data)
        try:
            mark = opt_lower(mark, data)
        except Exception:
            mark = mark
        new_list = []
        count = -1
        while count < len(data_entry) - 1:
            count = count + 1
            if count < mark:
                new_list.append(0)
            elif count > remove1:
                new_list.append(0)
            else:
```

```
                new_list.append(1)

    return new_list


def clean_noise(data_s_2):
    X_train = screened_data('0089')[["Strain", "Stress(MPa)"]]
    y_train = y_point(screened_data('0089'), trimmed_data('0089'))

    X_test = data_s_2[["Strain", "Stress(MPa)"]]
    y_test = y_point(screened_data("0095"), trimmed_data("0095"))

    model = PolynomialFeatures(degree=7)

    X_train = model.fit_transform(X_train)
    X_test = model.fit_transform(X_test)

    algo = LinearRegression()
    algo.fit(X_train, y_train)

    predict = algo.predict(X_test)
    predict = clusters(predict)
    predict = objective_2(predict, data_s_2)

    data_s_2["Clusters"] = predict
    data_2 = data_s_2[data_s_2.Clusters != 0]
    data_2 = data_2.drop(['Clusters'], axis=1)

    plt.title("testID_0090")
    sns.scatterplot(x="Strain", y="Stress(MPa)", data=data_s_2,
hue=predict, palette='Set1')
    plt.show()


clean_noise(screened_data('0090'))
```

Appendix I

```
"""
This code was built by Katleho Mofokeng
Katleho's student number is MFKKAT007
"""

import pandas as pd
# from matplotlib import pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import *
import os


raw_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/01 screened data"

trimmed_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/02 trimmed " \
                      "data"
```

```
def screened_data(num):
    data = pd.read_csv(
        f"{raw_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def trimmed_data(num):
    data = pd.read_csv(f"{trimmed_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def clusters(data):
    removed = []
    for i in data:
        if i < -0.05:
            removed.append(1)
        elif i <= 0.1:
            removed.append(2)
        elif i <= 0.13:
            removed.append(3)
        elif i < 0.2:
            removed.append(4)
        elif i < 0.35:
            removed.append(5)
        else:
            removed.append(0)

    return removed


def y_point(data_s, data_t):
    data_trim = []
    for i in data_t['Strain']:
        data_trim.append(i)

    removed = []
    for i in data_s['Strain']:
        if i in data_trim:
            removed.append(1)
        else:
            removed.append(0)

    return removed


def opt_1(start, data):
    yes = []
    no = []

    num_1 = start
    num_2 = start + 1

    before = data["Stress(MPa)"][num_1]
    after = data["Stress(MPa)"][num_2]
    #print("start ", start)
```

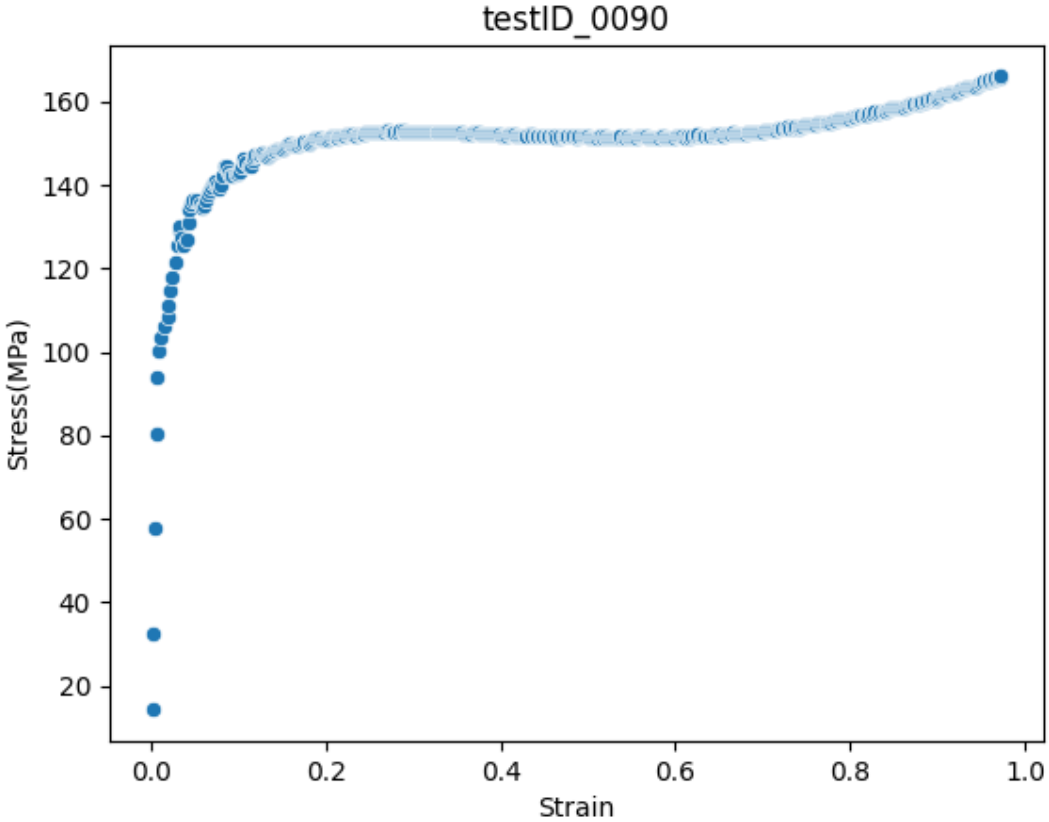
```
    count = 0
    while True:
        if float(before) > float(after):
            yes.append(1)
        else:
            no.append(0)

        num_1 = num_1 + 1
        num_2 = num_2 + 1

        before = data["Stress(MPa)"][num_1]
        after = data["Stress(MPa)"][num_2]

        if (len(yes) + len(no)) == 6:
            if len(no) > 0:
                count = count + 1
                num_1 = start + count
                num_2 = start + count + 1

                before = data["Stress(MPa)"][num_1]
                after = data["Stress(MPa)"][num_2]

                yes = []
                no = []
            else:
                #print("mark =", start)
                #print("stop =", num_1)
                break

    remove = count + start
    #print("end =", remove)
    return remove


def opt_lower(start, data):

    num_1 = start
    num_2 = start - 1
    #print("check")
    before = data["Stress(MPa)"][num_1]
    after = data["Stress(MPa)"][num_2]
    #print("start ", start)
    count = 0

    while True:
        if float(before) > float(after):
            count = count + 1
            num_1 = start - count
            num_2 = start - 1 - count

            before = data["Stress(MPa)"][num_1]
            after = data["Stress(MPa)"][num_2]

        else:
            #print("Done, count is =", count)
            break

    remove = start - count
    #print("end =", remove)
    return remove
```

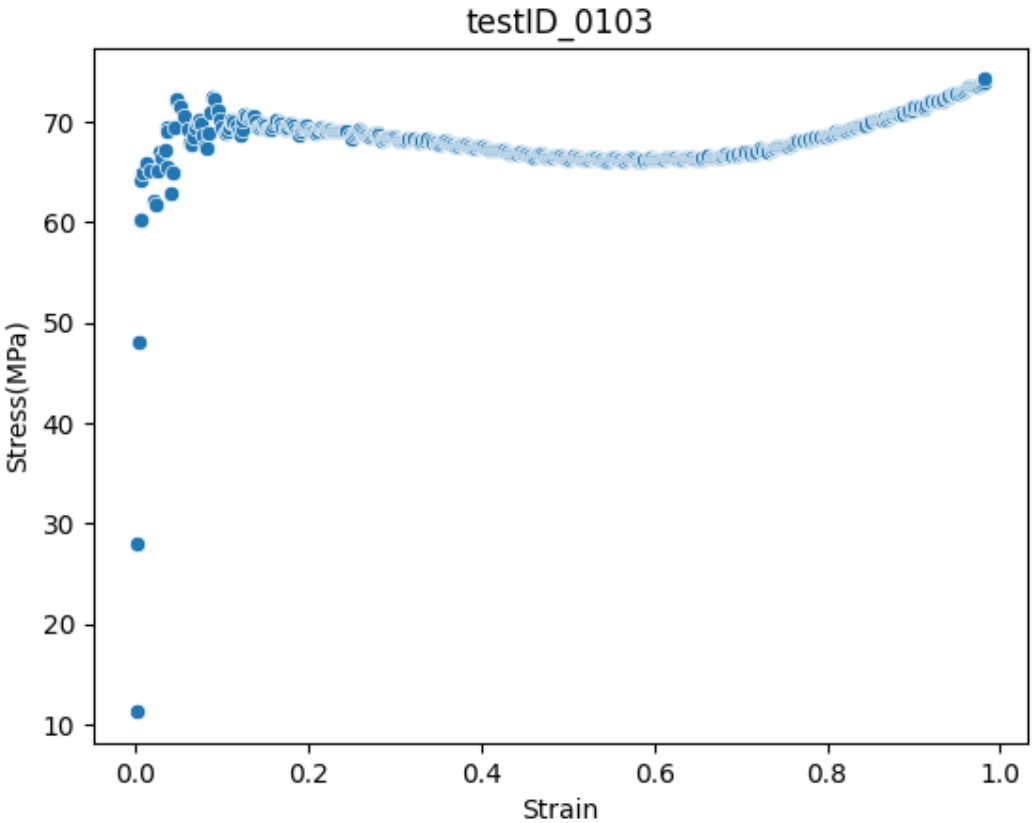
```python
def objective_2(data_entry, data):
    mark = -1
    while True:
        mark = mark + 1
        if data_entry[mark] == 0:
            break

    if set(data_entry) == {0, 1, 2, 3, 4, 5}:
        remove1 = mark
        while True:
            remove1 = remove1 + 1
            if data_entry[remove1] == 5:
                break

        remove1 = opt_1(remove1, data)
        #mark = opt_lower(mark, data)
        try:
            mark = opt_lower(mark, data)
        except Exception:
            mark = mark
        new_list = []
        count = -1
        while count < len(data_entry) - 1:
            count = count + 1
            if count < mark:
                new_list.append(0)
            elif count > remove1:
                new_list.append(0)
            else:
                new_list.append(1)
    else:
        remove1 = mark
        while True:
            remove1 = remove1 + 1
            if data_entry[remove1] == 5:
                break

        remove1 = opt_1(remove1, data)
        #mark = opt_lower(mark, data)
        try:
            mark = opt_lower(mark, data)
        except Exception:
            mark = mark
        new_list = []
        count = -1
        while count < len(data_entry) - 1:
            count = count + 1
            if count < mark:
                new_list.append(0)
            elif count > remove1:
                new_list.append(0)
            else:
                new_list.append(1)

    return new_list


def clean_noise(data_s_2):
    X_train = screened_data('0089')[["Strain", "Stress(MPa)"]]
    y_train = y_point(screened_data('0089'), trimmed_data('0089'))
```

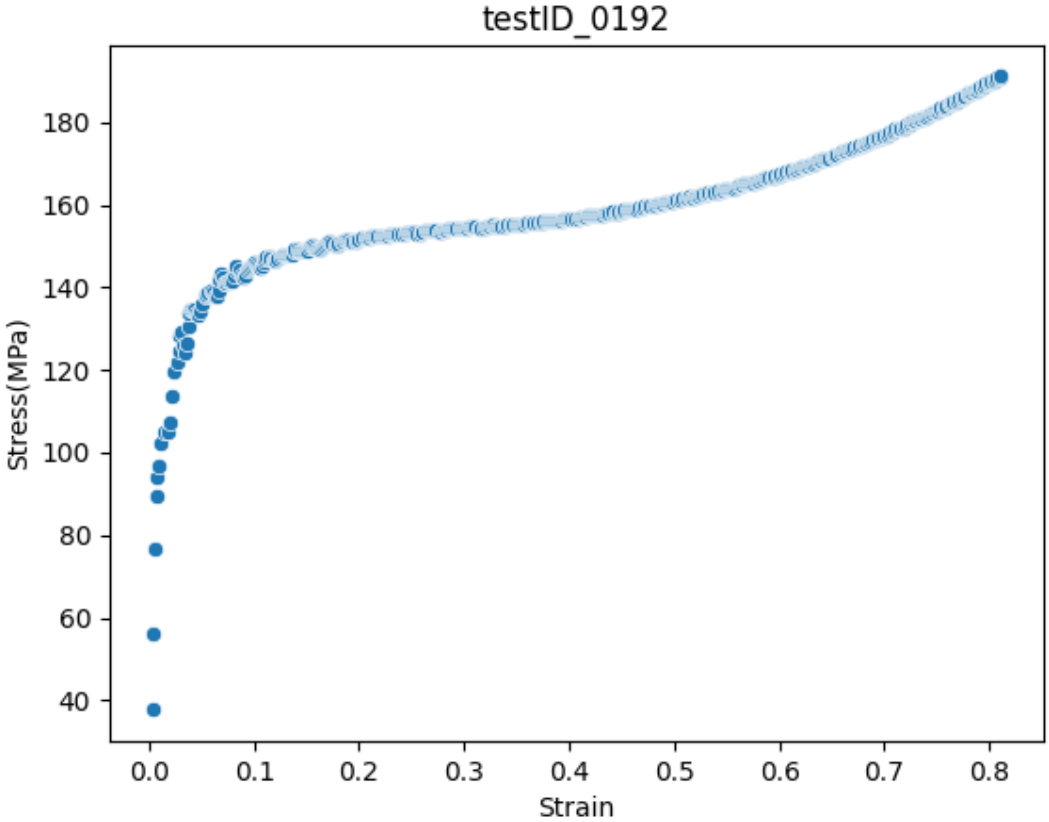
```
    X_test = data_s_2[["Strain", "Stress(MPa)"]]
    y_test = y_point(screened_data("0095"), trimmed_data("0095"))

    model = PolynomialFeatures(degree=7)

    X_train = model.fit_transform(X_train)
    X_test = model.fit_transform(X_test)

    algo = LinearRegression()
    algo.fit(X_train, y_train)

    predict = algo.predict(X_test)
    predict = clusters(predict)
    predict = objective_2(predict, data_s_2)

    data_s_2["Clusters"] = predict
    data_2 = data_s_2[data_s_2.Clusters != 0]
    data_2 = data_2.drop(['Clusters'], axis=1)

    plt.title("testID_0090")
    sns.scatterplot(x="Strain", y="Stress(MPa)", data=data_s_2,
hue=predict, palette='Set1')
    plt.show()


# clean_noise(screened_data('0090'))


def screened_data2(num):
    data = pd.read_csv(
        f"{raw_data_folder}/{num}")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


processed_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/DID " \
                        "WORK/supervised/cleaned data"


def results():
    dir_list = os.listdir(
        f"{raw_data_folder}")
    print(dir_list)
    X_train = screened_data('0089')[["Strain", "Stress(MPa)"]]
    X_test = y_point(screened_data('0089'), trimmed_data('0089'))

    for num in dir_list:
        print(num)
        data_s_2 = screened_data2(num)

        y_train = data_s_2[["Strain", "Stress(MPa)"]]

        model = PolynomialFeatures(degree=7)

        X_train = model.fit_transform(X_train)
        y_train = model.fit_transform(y_train)

        algo = LinearRegression()
```

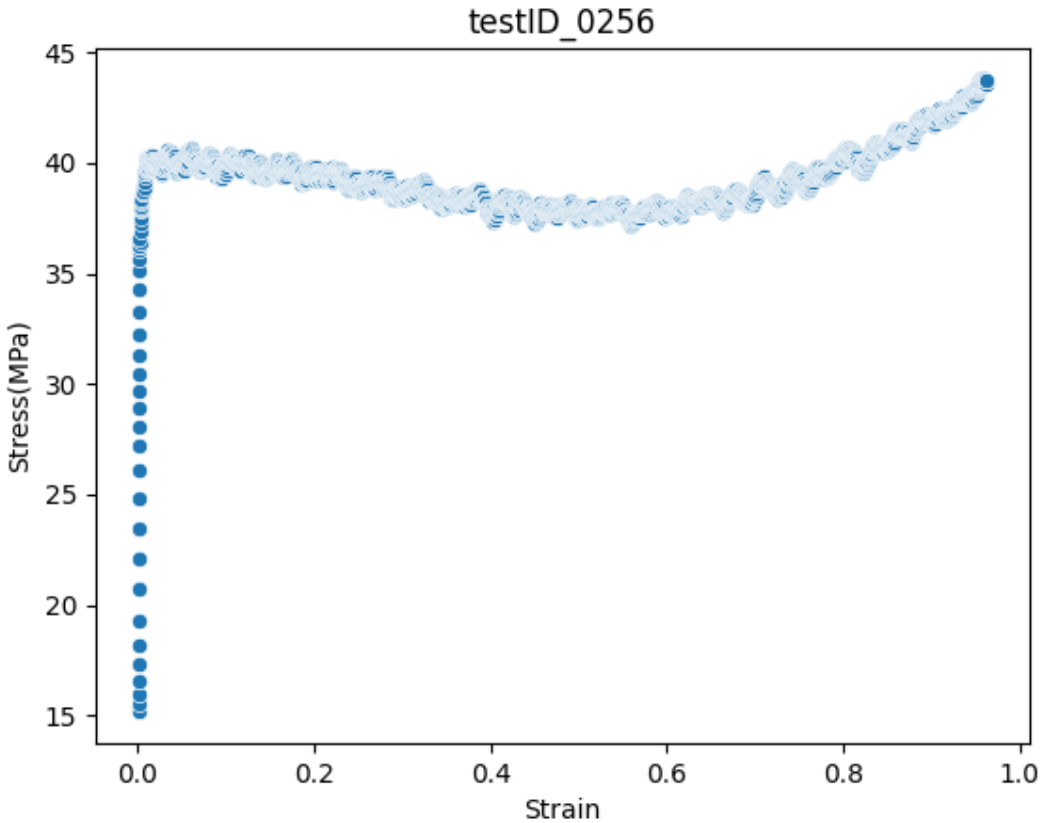
```
        algo.fit(X_train, X_test)

        predict = algo.predict(y_train)
        predict = clusters(predict)
        predict = objective_2(predict, data_s_2)

        data_s_2["Clusters"] = predict
        data_2 = data_s_2[data_s_2.Clusters != 0]
        data_2 = data_2.drop(['Clusters'], axis=1)
        df = data_2

        df.to_csv(
            f'{processed_data_folder}/{num}')


results()
```

Appendix J


testID_0090
Stress(MPa)
Strain
160
140
120
100
80
60
40
20
0.0
0.2
0.4
0.6
0.8
1.0


testID_0103
Stress(MPa)
Strain
70
60
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0

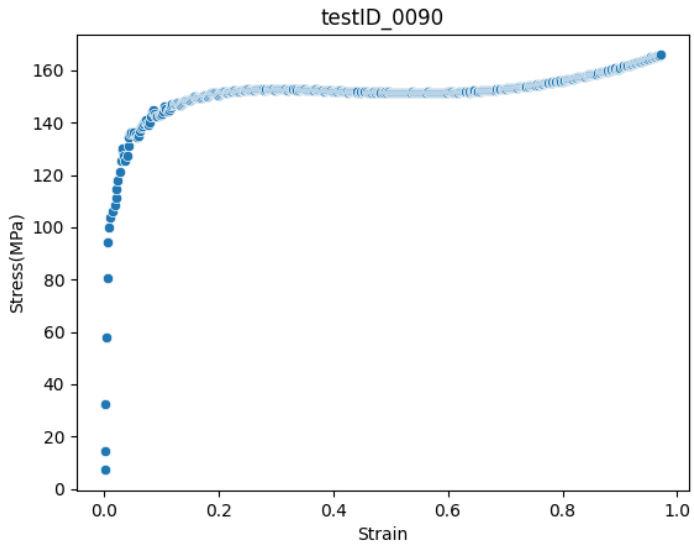

testID_0192
Stress(MPa)
Strain
180
160
140
120
100
80
60
40
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


testID_0256
Stress(MPa)
Strain
45
40
35
30
25
20
15
0.0
0.2
0.4
0.6
0.8
1.0

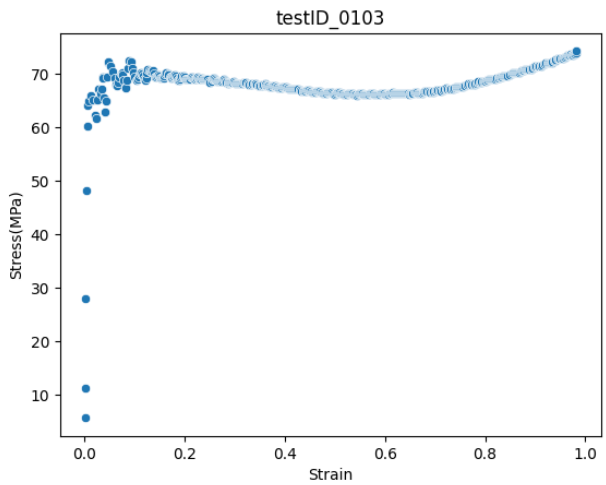
Appendix K

```
"""
This code was built by Katleho Mofokeng
Katleho's student number is MFKKAT007
"""

import pandas as pd
# from matplotlib import pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns

trimmed_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/02 trimmed " \
                      "data"


def trimmed_data(num):
    data = pd.read_csv(f"{trimmed_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def elastic_modulus(data):
    y1 = data['Stress(MPa)'][0]
    y2 = data['Stress(MPa)'][3]

    x1 = data['Strain'][0]
    x2 = data['Strain'][3]

    E_modulus = (y2 - y1) / (x2 - x1)

    print('Elastic modulus =', E_modulus, "MPa")

    plt.title("testID_0090")
    sns.scatterplot(x="Strain", y="Stress(MPa)", data=data, palette='Set1')
    plt.show()


elastic_modulus(trimmed_data('0090'))
```

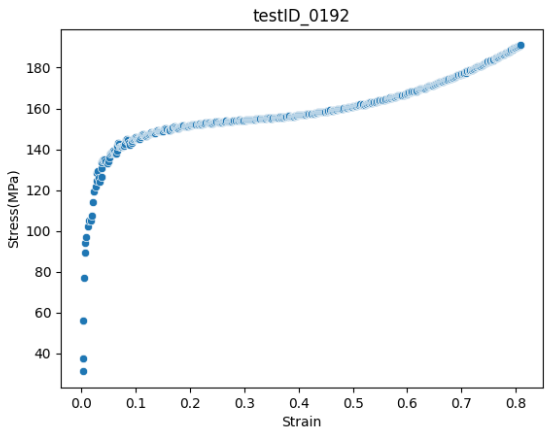
Appendix L



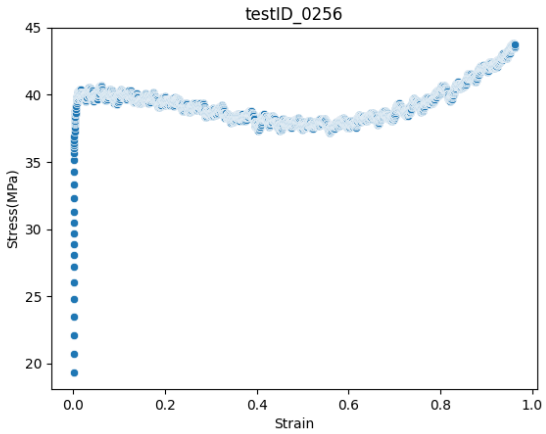
Elastic modulus = 15275.005465466707 MPa



Elastic modulus = 13774.738985630665 MPa

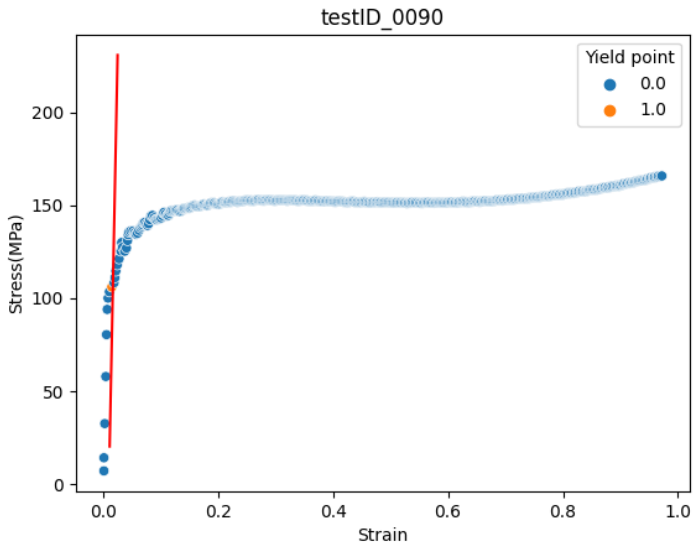


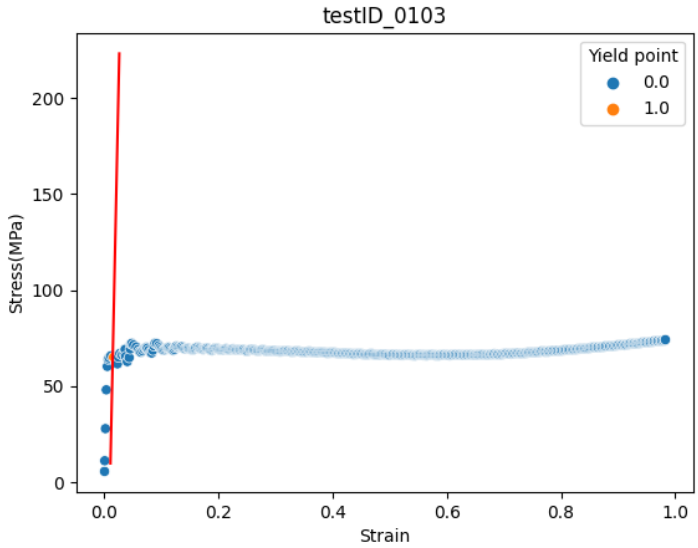
Elastic modulus = 17016.792282864135 MPa

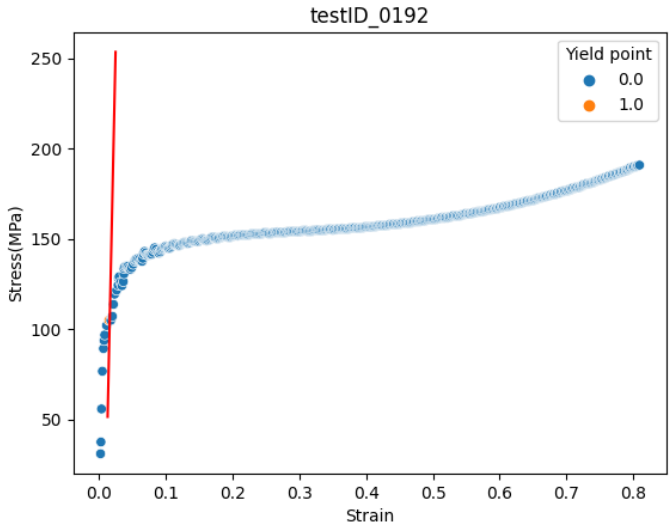


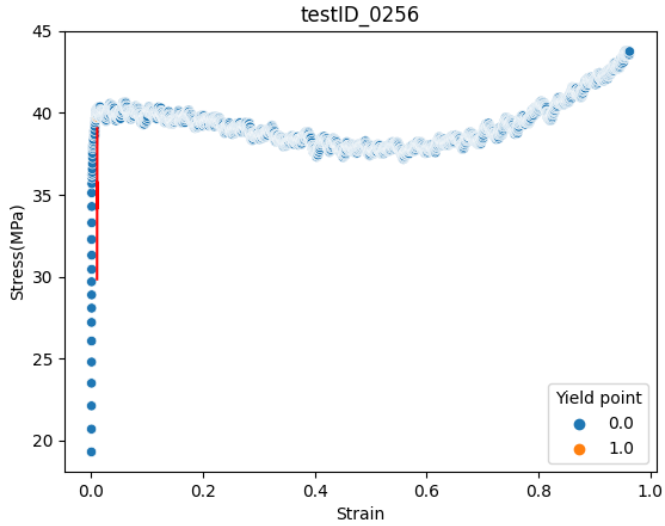
Elastic modulus = 28887.474015999367 MPa

Appendix M


testID_0090
Yield point
0.0
1.0
Stress(MPa)
Strain
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
1.0


testID_0103
Yield point
0.0
1.0
Stress(MPa)
Strain
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
1.0





Appendix N

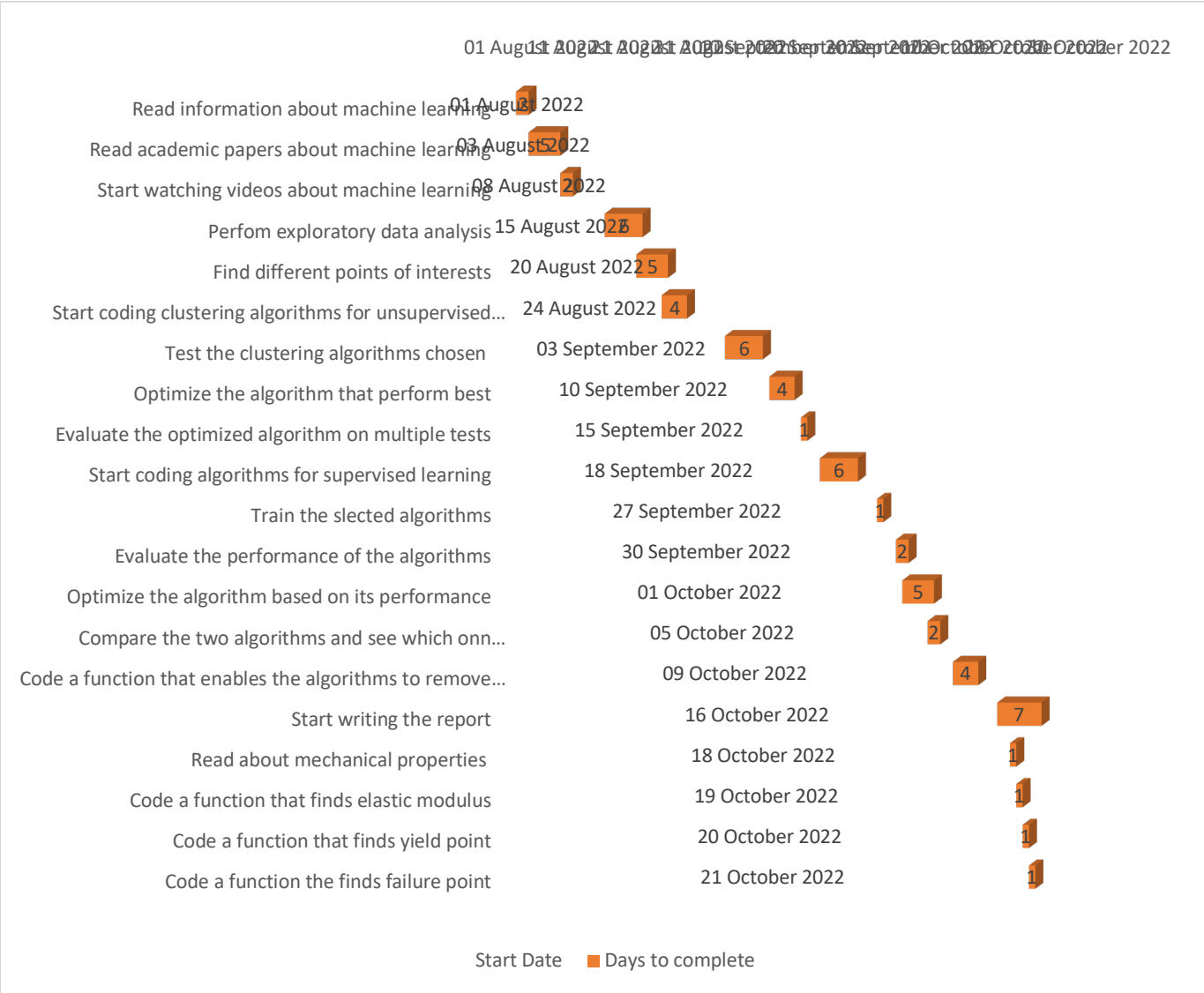
```
"""
This code was built by Katleho Mofokeng
Katleho's student number is MFKKATOO7
"""

import numpy as np
import pandas as pd
# from matplotlib import pyplot as plt
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import seaborn as sns

raw_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/01 screened data"

trimmed_data_folder = "/Users/admin/Library/CloudStorage/OneDrive-
UniversityofCapeTown/final year project/02 trimmed " \
                      "data"
```

```
def screened_data(num):
    data = pd.read_csv(
        f"{raw_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def trimmed_data(num):
    data = pd.read_csv(f"{trimmed_data_folder}/testID_{num}.csv")
    data = data[["Strain", "Stress(MPa)", "Time(sec)", "Force(kN)",
"Jaw(mm)"]]
    return data


def yield_method_2(data_entry):
    data = data_entry

    y1 = data['Stress(MPa)'][0]
    y2 = data['Stress(MPa)'][3]

    x1 = data['Strain'][0]
    x2 = data['Strain'][3]

    E_modulus = (y2 - y1) / (x2 - x1)

    data['strain offset'] = data['Strain'] + 0.01
    data['stress offset'] = data['Strain'] * E_modulus

    y1 = data['stress offset'][0]
    y2 = data['stress offset'][3]

    x1 = data['strain offset'][0]
    x2 = data['strain offset'][3]

    E_modulus = (y2 - y1) / (x2 - x1)
    y_intercept = y2 - (E_modulus * x2)

    print('E_modulus =', E_modulus)

    Eqn = E_modulus * data['Strain'] + y_intercept

    count = 0
    dis_list = []

    while count < len(data['Strain']) - 2:
        distance = abs(
            E_modulus * data['Strain'][count] - data['Stress(MPa)'][count]
+ y_intercept) / (
                                E_modulus ** 2 + 1)

        dis_list.append(distance)
        count = count + 1

    t = 1
    count = 0
    while t != min(dis_list):
        t = dis_list[count]
        count = count + 1
```

```
    yield_point = np.zeros(len(data['Strain']))
    yield_point[count - 1] = 1

    data["Yield point"] = yield_point

    plt.title("testID_0256")
    sns.scatterplot(x="Strain", y="Stress(MPa)", data=data, hue="Yield
point")
    plt.plot(data['strain offset'][1:9], data['stress offset'][1:9], "r")
    plt.show()


yield_method_2(trimmed_data("0256"))
```

Appendix O

Gantt chart



Task	Start Date	Days to complete
Read information about machine learning	01 August 2022	2
Read academic papers about machine learning	03 August 2022	5
Start watching videos about machine learning	08 August 2022	2
Perfom exploratory data analysis	15 August 2022	6
Find different points of interests	20 August 2022	5
Start coding clustering algorithms for unsupervised learning	24 August 2022	4
Test the clustering algorithms chosen	03 September 2022	6
Optimize the algorithm that perform best	10 September 2022	4
Evaluate the optimized algorithm on multiple tests	15 September 2022	1
Start coding algorithms for supervised learning	18 September 2022	6
Train the slected algorithms	27 September 2022	1
Evaluate the performance of the algorithms	30 September 2022	2
Optimize the algorithm based on its performance	01 October 2022	5
Compare the two algorithms and see which onn performes better	05 October 2022	2
Code a function that enables the algorithms to remove all the noise	09 October 2022	4
Start writing the report	16 October 2022	7
Read about mechanical properties	18 October 2022	1
Code a function that finds elastic modulus	19 October 2022	1
Code a function that finds yield point	20 October 2022	1
Code a function the finds failure point	21 October 2022	1

Appendix P

Meeting Agenda

Location:	Microsoft Teams
Date:	11 August 2022
Time:	10:00

Agenda details:

Topic 1: K means method

- How do I choose reasonable K number clusters?
- Is there any way I can evaluate how good my current K value is at determining clusters?

Topic 2: Cleaning data

- What are the criteria for identifying “spurious” data points?

Topic 3: ML Algorithm

- How do I evaluate the performance of the algorithm?

Topic 4: Data

- Do I only clean the stress and strain data from the given data?

Meeting Agenda

Location:	Microsoft Teams
Date:	16 August 2022
Time:	12:30

Agenda details:

Topic 1: Exploratory data analysis

- Different points of interests

Topic 2: Clustering methods

- Discuss different clustering methods

Topic 3: Choose the best algorithm

- How do I evaluate the performance of the algorithm?

Topic 4: Labelling

- Label different points of interests.

Meeting Agenda

Location: Microsoft Teams

Date: 23 August 2022

Time: 12:30

Agenda details:

Topic 1: Optimizing a ML algorithm

- How to optimize if using more than one model

Topic 2: test-train split

- Explanation

Topic 3: Evaluation

- How accurate should the algorithm be at removing noise

Topic 4: Objectives of the project

Topic 5: Literature review

- What information do I need to consider for literature review.

Meeting Agenda

Location:	Microsoft Teams
Date:	30 August 2022
Time:	12:30

Agenda details:

Topic 1: Check the method for identifying yield strength

Topic 2: Check the models I used

Topic 3: Discuss supervised learning

Meeting Agenda

Location:	Microsoft Teams
Date:	13 September 2022
Time:	12:30

Agenda details:

Topic 1: Check week6_submission document

Topic 2: Check the linear regression method for identifying the transition point.

Topic 3: Check the model in wee6_submission document

Action items of previous meeting:

1. Add a review of the previous meeting.
2. Add the action items
3. Find the elastic region using linear regression method.
4. Shows how the model works by including the stress strain plot to show the points removed.

Review of the previous meeting:

1. For identifying the elastic region of the stress strain curve, you can use two different methods. The first method is using linear regression to draw the best fit line. The second method is to use clustering algorithm to identify the points in the elastic region.
2. Reference the method I used that I saw from other literature.
3. In the meeting agenda review the last meeting and add the action items.
4. While showing how the model works, include the plot of stress strain to see which data was removed and which one is left.
5. Remember the true strain and true stress is logarithmic.

Meeting Agenda

Location:	Microsoft Teams
Date:	27 September 2022
Time:	12:30

Agenda details:

Topic 1: Check **yield info** excel file in “Week9_submission” folder.

Topic 2: Check **yield info 2** excel file in “Week9_submission” folder.

Topic 3: Check **compare** excel file in ““Week9_submission/info” folder.

Action items of previous meeting:

1. Improve the yield function
2. Check why the errors occurred in the code
3. Start measuring the performance of the algorithm

Review of the previous meeting:

1. More objectives:
    - UTS is not for compression test
    - Consider flow stress
    - Consider 0.2 Strain
2. Algorithm:
    - Keep many points in the elastic region as possible
    - Measure the performance
3. Yield:
    - Some values are too low
    - Keep the error code

Meeting Agenda

Location:	Microsoft Teams
Date:	20 September 2022
Time:	12:30

Agenda details:

Topic 1: Check more objectives pdf in “Week7_submission” folder.

Topic 2: Check algorithm pdf in “Week7_submission” folder.

Topic 3: Check the yield info excel in ““Week7_submission/info” folder.

Action items of previous meeting:

1. List other objectives that can be achieved.
2. Start training and testing.
3. Show for multiple tests.

Review of the previous meeting:

1. For yield strength check
    - Repeatability
    - R squared
    - Look at the error
2. Clustering algorithm: Plot against time

Meeting Agenda

Location:	Microsoft Teams
Date:	04 October 2022
Time:	12:30

Agenda details:

Topic 1: Discuss **table of contents** word file in “Week10_submission” folder.

Topic 2: Discuss **evaluating performance** pdf file in “Week10_submission” folder.

Action items of previous meeting:

1.Upload table of contents
2.Think of how else to measure the performance of the algorithm
3.Ensure for every test, there’s a pdf with all the measurements and plots

Review of the previous meeting:

1.Yield info:
- •The error can be seen

2.Yield info 2:
- •Function identifies the point yield starts a bit late

3.Compare:
- •Try to improve the performance
- •Measure performance if it was unsupervised learning

Meeting Agenda

Location:	Microsoft Teams
Date:	11 October 2022
Time:	12:30

Agenda details:

Topic 1: Discuss **table of contents** word file in “Week11_submission” folder.

Topic 2: Discuss **literature review topics** word file in “Week11_submission” folder.

Action items of previous meeting:

1.Upload the updated table of contents

Review of the previous meeting:

For introduction:

- Include background, aims and objectives
- Write about 500 words For literature review:

For literature review:

• Include types of tests, mechanical testing Conclusion:

For conclusion:

- Add conclusion to the table of contents
- Write about 500 words

Include “evaluate the performance” in “train test”. Thus, combine the two topics.

Appendix Q

Meeting Minutes - [Project 124]

Location: EM building
Date: 4 August 2022
Time: 11:00

Attendance

Katleho Mofokeng (student)

Ernesto Ismail (supervisor)

Agenda Items

1. Bringing project description.
2. Summary of the key tasks of the project.

Action Items

1. Get data.
2. Invite Daniel Slater to the next meeting.
3. Review the data.
4. Start labelling data.

Other Notes

Tasks I will perform in the coming weeks:

- Find ML Clustering algorithms
- Review them and select the appropriate ones
- Train the selected algorithms
    - Can use cross validation method.
- Evaluate the performance of the algorithms
    - It identifies regions of interest on each stress-strain curve.
    - Identifies spurious noise in data
    - Does not over-fit

Meeting Minutes Template © 2014 Vertex42 LLC https://www.vertex42.com/WordTemplates/meeting-minutes.html

Meeting Minutes - [Project 124]

Location: Microsoft Teams
Date: 11 August 2022
Time: 10:00

Attendance

Katleho Mofokeng (student)

Ernesto Ismail (supervisor)

Daniel Slater (Co-supervisor)

Agenda Items

1. Different clustering methods
2. Labelling data.

Action Items

1. Label data
2. Exploratory data analysis
3. Summary of different clustering techniques
4. Identify different points of interest
5. Learn the theory behind the clustering methods

Meeting Minutes Template © 2014 Vertex42 LLC https://www.vertex42.com/WordTemplates/meeting-minutes.html

Meeting Minutes - [Project 124]

Location: Microsoft Teams
Date: 16 August 2022
Time: 12:30

Attendance

Katleho Mofokeng (student)

Ernesto Ismail (supervisor)

Agenda Items

1. Exploratory data analysis
2. Evaluating the performance of a model
3. Discuss different clustering methods
4. Labelling

Notes

1. Compare different plots. By plotting time against force, jaw, stain and stress.
2. To evaluate the model, you can compare your cleaned data to your trimmed.
3. See how different models cluster the data. You can also use more than one model.
4. Identify different points of interests from exploratory data.

Action Items

1. Write the objectives of the project and ask the supervisors to check if their correct.
2. Don't just look at stress strain data but also time and force data.
3. Continue with exploratory data analysis.
4. Compare different models.

Update on what have been done since last previous meeting

1. I have done a summary of different clustering algorithms
2. I got the data that was removed using the trimmed and screened data.
3. I mapped the removed data to the screened data to see which points exactly were removed.
4. I studied the theory of different clustering algorithms.

Meeting Minutes Template © 2014 Vertex42 LLC https://www.vertex42.com/WordTemplates/meeting-minutes.html

Meeting Minutes - [Project 124]

Location: UCT Red lab
Date: 18 August 2022
Time: 16:20

Attendance

Katleho Mofokeng (student)

Daniel Slater (Co-supervisor)

Agenda Items

1. Exploratory data analysis
2. Understanding the information from excel info sheet.
3. How to train and test the data
4. How to optimize, if using more than one model.

Notes

1. Look into how a pair plot works.
2. From the excel sheet.
    a. UC: Uniaxial Compression
    b. AC: As-Cost
    c. H560: Homogenized at 560 °C
    d. H580: Homogenized at 580 °C
    e. Rate: strain rate
    f. Temperature: It is in degrees Celsius
    g. L_o: Initial length
    h. D_o: Initial diameter
    i. Jaw: Displacement
3. Take the screened data as training data and trimmed data as testing data.
4. You can use scipy.optimize for optimization.

Action Items

1. Study how a pair plot works.
2. Start thinking about the project report.
3. Reasons for why we are using clustering algorithms.

Meeting Minutes Template © 2014 Vertex42 LLC https://www.vertex42.com/WordTemplates/meeting-minutes.html

Meeting Minutes - [Project 124]

Location: MS Teams
Date: 23 August 2022
Time: 12:30

Attendance

Katleho Mofokeng (student)

Enersto Ismail (Supervisor)

Daniel Slater (Co-supervisor)

Agenda Items

1. Literature review
2. Objectives of the project
3. Test train split
4. Evaluating the performance of the algorithm

Notes

1. For literature review, use the funnel approach:
    a. Introduction
    b. Locate
    c. Broad Gap(s)
    d. Targeted Gap
    e. Focus of Approach
    f. Selection / Details
    g. Argue
    h. Conclude
2. Modify the objectives in the following way:
    a. Add what I found from exploratory data analysis
    b. Pretest & posttest, yield strength & ultimate tensile strength
    c. Train the model first before optimizing it.
    d. Explain to the reader what I am working with, how it works, how I made it work & results.
3. Label the raw data as training data and trimmed data as testing data. Then divide them in 70:30 ratio. 70% used for training and testing, 30% used for evaluating the performance. I am doing

Meeting Minutes - [Project 124]

Location: Microsoft teams
Date: 30 August 2022
Time: 12:30

Attendance

Katleho Mofokeng (student)

Enersto Ismail (Supervisor)

Daniel Slater (Co-supervisor)

Agenda Items

1. Check the method for identifying yield strength
2. Check the models used
3. Discuss supervised learning

Notes

1. For identifying the elastic region of the stress strain curve, you can use two different methods. The first method is using linear regression to draw the best fit line. The second method is to use clustering algorithm to identify the points in the elastic region.
2. Reference the method I used that I saw from other literature.
3. In the meeting agenda review the last meeting and add the action items.
4. While showing how the model works, include the plot of stress strain to see which data was removed and which one is left.
5. Remember the true strain and true stress is logarithmic.

Action Items

1. Add a review of the previous meeting.
2. Add the action items
3. Find the elastic region using linear regression method.
4. Shows how the model works by including the stress strain plot to show the points removed.

Update on what have been done since last previous meeting

Meeting Minutes - [Project 124]

Location: Microsoft teams
Date: 20 September 2022
Time: 12:30

Attendance

Katleho Mofokeng (student)

Enersto Ismail (Supervisor)

Daniel Slater (Co-supervisor)

Agenda Items

Topic 1: Check more objectives pdf in “Week7_submission” folder.

Topic 2: Check algorithm pdf in “Week7_submission” folder.

Topic 3: Check the yield info excel in ““Week7_submission/info” folder.

Notes

1. More objectives:
    - UTS is not for compression test
    - Consider flow stress
    - Consider 0.2 Strain
2. Algorithm:
    - Keep many points in the elastic region as possible
    - Measure the performance
3. Yield:
    - Some values are too low
    - Keep the error code
    - Modify the error code
    - Use standard deviation for error check
    - Standard deviation will allow us to look at the spread of the data.

Action Items

Meeting Minutes Template © 2014 Vertex42 LLC https://www.vertex42.com/WordTemplates/meeting-minutes.html

Meeting Minutes - [Project 124]

Location: Microsoft teams
Date: 13 September 2022
Time: 12:30

Attendance

Katleho Mofokeng (student)

Enersto Ismail (Supervisor)

Daniel Slater (Co-supervisor)

Agenda Items

1. Check week6_submission document
2. Check the linear regression method for identifying the transition point.
3. Check the model in wee6_submission document

Notes

1. For yield strength check
    - Repeatability
    - R squared
    - Look at the error
2. Clustering algorithm: Plot against time

Action Items

1. List other objectives that can be achieved.
2. Start training and testing.
3. Show for multiple tests.

Update on what have been done since last previous meeting

1. I coded a clustering algorithm that identifies spurious data in one step.
2. I coded an algorithm that identifies that identifies the transition point.
3. I started training the algorithm.

.

Meeting Minutes Template © 2014 Vertex42 LLC https://www.vertex42.com/WordTemplates/meeting-minutes.html

Meeting Minutes - [Project 124]

Location: Microsoft teams
Date: 27 September 2022
Time: 12:30

Attendance

Katleho Mofokeng (student)

Enersto Ismail (Supervisor)

Daniel Slater (Co-supervisor)

Agenda Items

Topic 1: Check **yield info** excel file in “Week9_submission” folder.

Topic 2: Check **yield info 2** excel file in “Week9_submission” folder.

Topic 3: Check **compare** excel file in ““Week9_submission/info” folder.

Notes

1. Yield info:
    - The error can be seen
2. Yield info 2:
    - Function identifies the point yield starts a bit
3. Compare:
    - Try to improve the performance
    - Measure performance as if it was unsupervised learning

Action Items

1. Upload table of contents
2. Think of how else to measure the performance of the algorithm
3. Ensure for every test, there’s a pdf with all the measurements and plots

Update on what have been done since last previous meeting

1. I was to improve the performance of the yield function

Meeting Minutes Template © 2014 Vertex42 LLC https://www.vertex42.com/WordTemplates/meeting-minutes.html

Meeting Minutes - [Project 124]

Location: Microsoft teams
Date: 04 October 2022
Time: 12:30

Attendance

Katleho Mofokeng (student)

Enersto Ismail (Supervisor)

Daniel Slater (Co-supervisor)

Agenda Items

Topic 1: Discuss table of contents word file in “Week10_submission” folder.

Topic 2: Discuss evaluating performance pdf file in “Week10_submission” folder.

Notes

For introduction:

- Include background, aims and objectives
- Write about 500 words

For literature review:

- Include types of tests, mechanical testing

Conclusion:

- Add conclusion to the table of contents
- Write about 500 words

Action Items

1. Upload the updated table of contents
2. You can show diagrams
3. You can also show methodologies

Meeting Minutes Template © 2014 Vertex42 LLC https://www.vertex42.com/WordTemplates/meeting-minutes.html