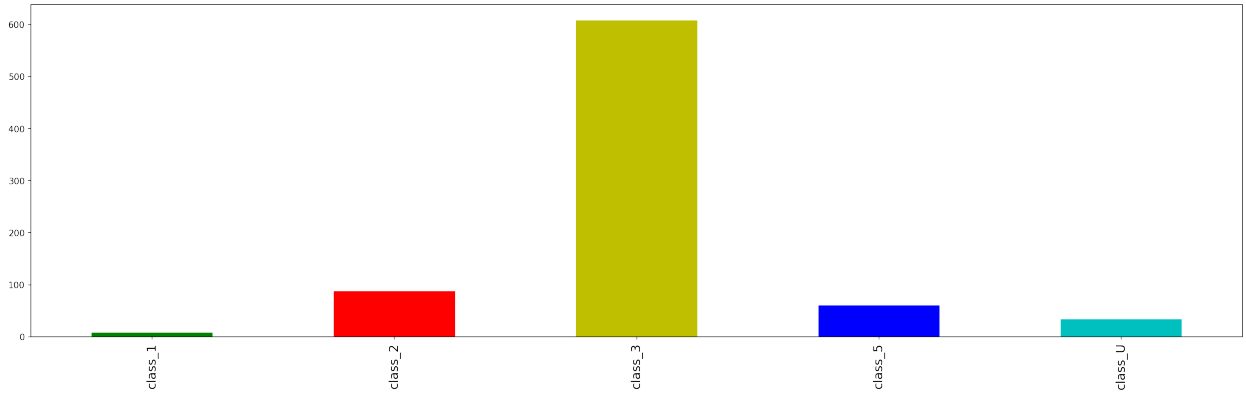
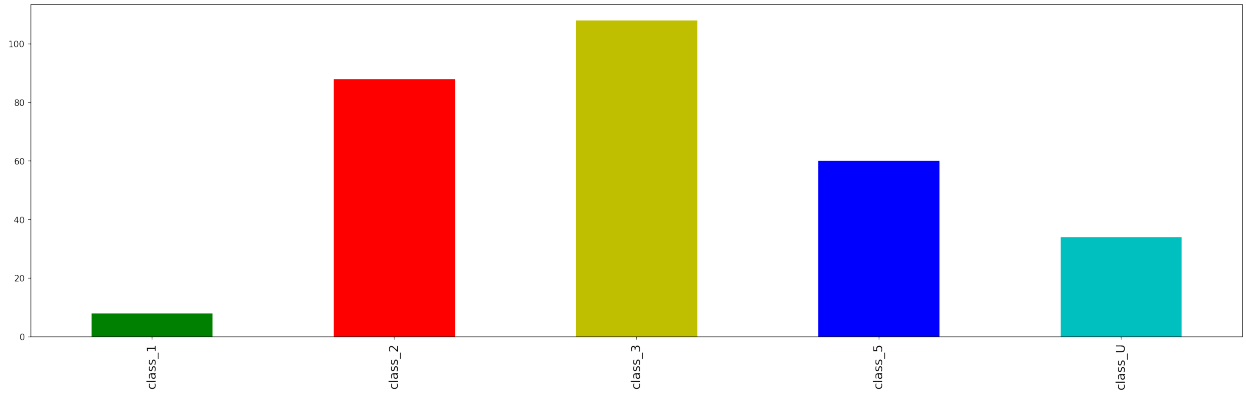
Pre Processing Data :

The conventional model evaluation methods do not accurately measure model performance when faced with imbalanced datasets. Standard classifier algorithms like Decision Tree and Logistic Regression have a bias towards classes which have number of instances. They tend to only predict the majority class data. The features of the minority class are treated as noise and are often ignored. Thus, there is a high probability of misclassification of the minority class as compared to the majority class.

The dataset is obviously imbalanced, As we see by visualizing data's distribution:



So first step is to balance the data.
Two Strategies to deal with imbalanced data sets are undersampling and oversampling. we choose undersampling as we see class_3 is Unbalanced we should randomly drop some rows of it.
Result of balancing has illustrated below:

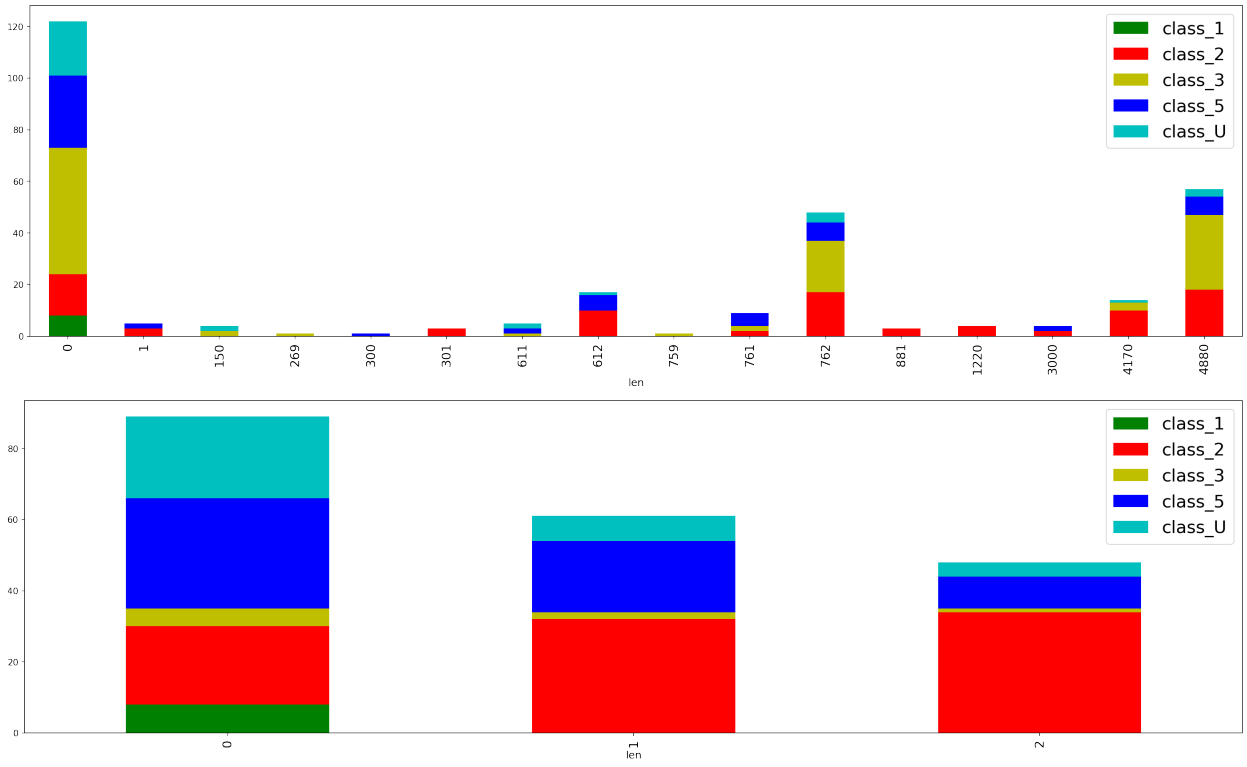


For second step we should drop ineffectual columns. By analyzing the dataset we see some columns has a very small correct information so we should drop them :

```
delete column: m
delete column: marvi
delete column: corr
delete column: jurofm
delete column: s
delete column: p
```

The column "product-type" has only value 'c' so it's useless and should be dropped too. The column "packing" would be dropped too because after data balancing it's not effective.

Now the Data should be provided to a Decision Tree.
first we separate continues data columns, for example the "len" column has been decided in 3 classes :



the columns hardness, strength, 4, thick and width have been categorized too. we tried to group them based on classes conversion.

then categorical values mapped to numeric classes to use related functions on the dataset.

Missing values are a common occurrence, and we need to have a strategy for treating them. for most columns we replace them with a Zero to make them effectless.
Surface-quality missing values has been inferred from 'temper_rolling' and 'strength'; because it's related to them.

	temper_rolling	strength	surface-quality
0	0	0	2.0
1	0	1	NaN
2	0	2	4.0
3	0	3	NaN
4	0	4	4.0
5	0	5	NaN
6	1	0	4.0

`DecisionTreeClassifier` is a class capable of performing multi-class classification on a dataset. grid search has been used for choosing decision tree parameter settings.
The dictionary para_grid provides the different parameter settings to test. The keys are the parameter name and the values are a list of settings to try.

```
param_grid = {"criterion": ["gini", "entropy"],
              "min_samples_split": [2, 10, 20],
              "max_depth": [2, 5, 10],
              "min_samples_leaf": [1, 5, 10],
              "max_leaf_nodes": [None, 5, 10, 20],
              }
```

Results shown below:

```
GridSearchCV took 4.82 seconds for 288 candidate parameter settings.
Model with rank: 1
Mean validation score: 0.960 (std: 0.025)
Parameters: {'criterion': 'entropy', 'max_depth': 10, 'max_leaf_nodes': 20, 'min_samples_leaf': 1, 'min_samples_split': 2}

Model with rank: 2
Mean validation score: 0.956 (std: 0.032)
Parameters: {'criterion': 'entropy', 'max_depth': None, 'max_leaf_nodes': 20, 'min_samples_leaf': 1, 'min_samples_split': 2}

Model with rank: 3
Mean validation score: 0.953 (std: 0.023)
Parameters: {'criterion': 'entropy', 'max_depth': None, 'max_leaf_nodes': 20, 'min_samples_leaf': 1, 'min_samples_split': 10}


Process finished with exit code 0
```

A Graph and sudo code has been generated in the code too.