Team : Mutants303
Anshuman Chakravarty
Arnab Manna
Sushovan Halder

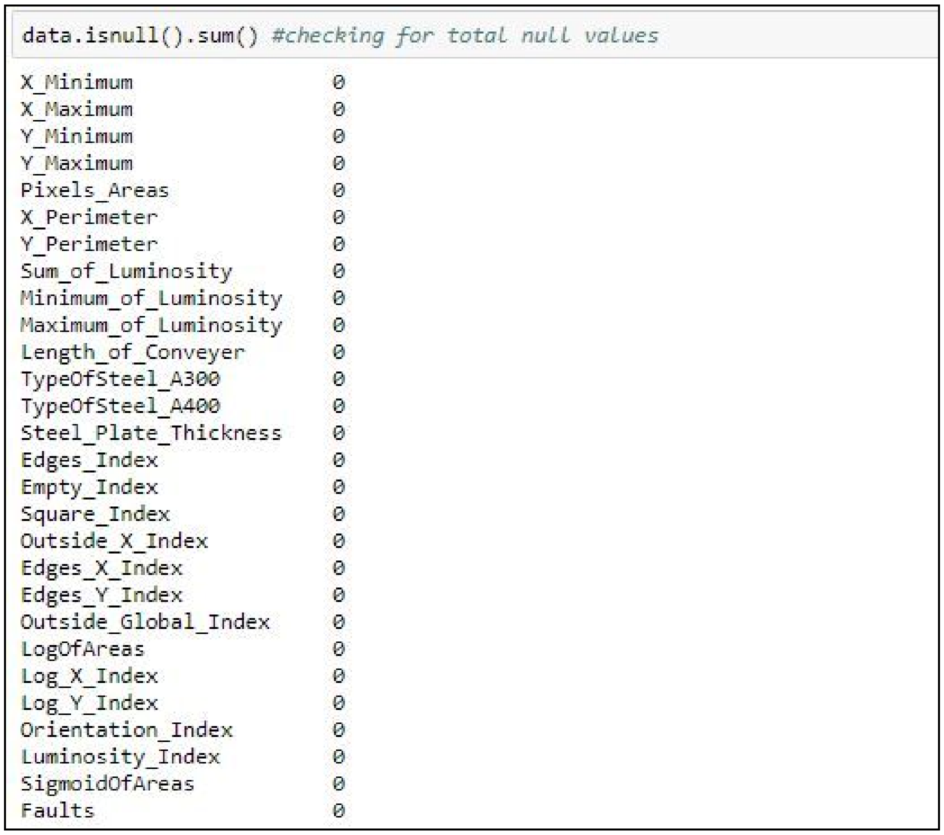
ABSTRACT

Data description

The dataset consists of 27 different features of steel plate samples and the goal is to predict whether a given steel plate is faulty or not. The target variable "Faults" consist of two classes, **'0'** (for not faulty) and **'1'** (for faulty) and thus this turns out to be a binary classification problem.

Data Exploration

- The dataset contains 1358 rows and 28 columns, the last column being the target variable. It does not contain any null values and hence **no** data imputation was needed.



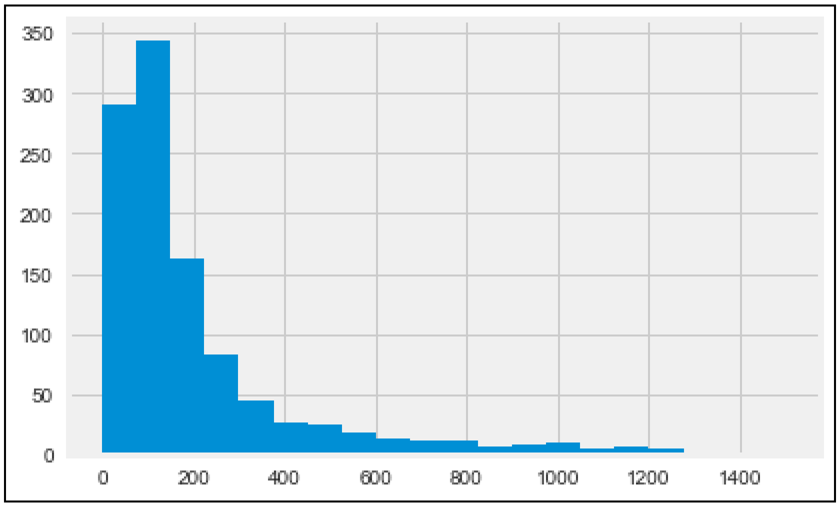
```
data.isnull().sum() #checking for total null values

X_Minimum                0
X_Maximum                0
Y_Minimum                0
Y_Maximum                0
Pixels_Areas             0
X_Perimeter              0
Y_Perimeter              0
Sum_of_Luminosity        0
Minimum_of_Luminosity    0
Maximum_of_Luminosity    0
Length_of_Conveyer       0
TypeOfSteel_A300         0
TypeOfSteel_A400         0
Steel_Plate_Thickness    0
Edges_Index              0
Empty_Index              0
Square_Index             0
Outside_X_Index          0
Edges_X_Index            0
Edges_Y_Index            0
Outside_Global_Index     0
LogOfAreas               0
Log_X_Index              0
Log_Y_Index              0
Orientation_Index        0
Luminosity_Index         0
SigmoidOfAreas           0
Faults                   0
```

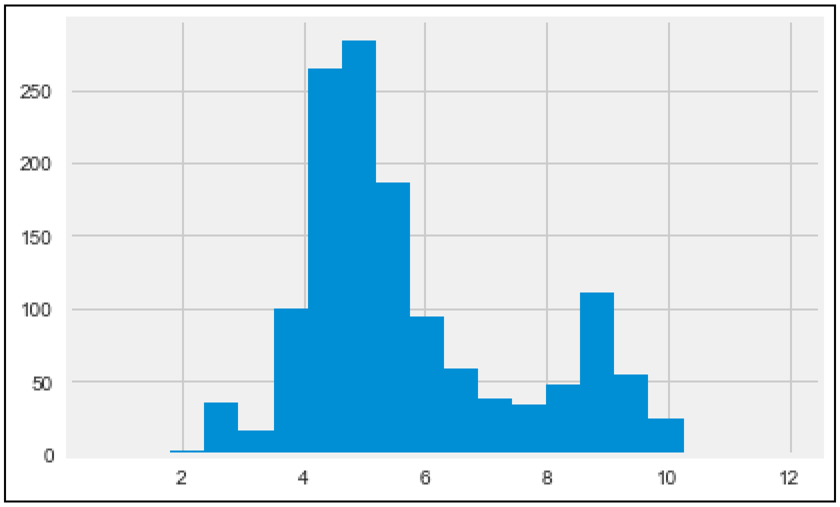
- There are 2 categorical variables, *"TypeOfSteel"* and *"Outside_Global_Index"* and the rest (25) are continuous variables.

- The feature *“Outside_Global_Index”* was one-hot encoded into 3 columns, namely Outside0, Outside1 and Outside0.5, since it consisted of three factors (**0**,**0.5** and **1**).
- Histogram for each of the continuous variables was plotted and if the distribution turned out to be skewed then either **log** or **square-root** was taken appropriately to make the shape close to normal. For example, let us take “Pixel_Areas” ,

Before converting to normal



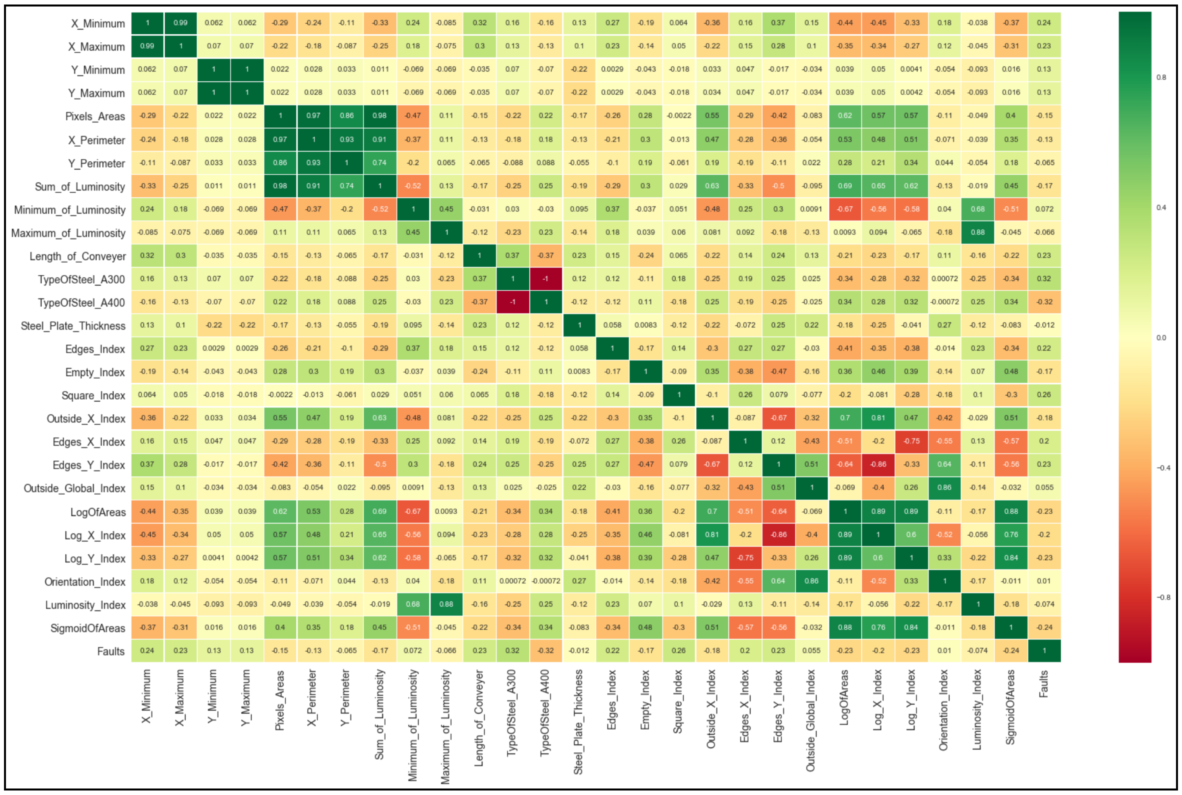
After taking log



In the subsequent steps, all the continuous variables were normalized.

Feature Selection

The selection of the optimal features was carried out by building a correlation matrix.



It was found that -:

- X_minimum and X_maximum
- Y_Minimum and Y_Maximum
- Pixels_Areas, Y_Perimeter and Sum_of_Luminosity
- LogOfAreas, Log_X_Index and Log_Y_Index

were highly correlated. Hence, *X_minimum*, *Y_minimum*, *Y_Perimeter*, *Sum_of_Luminosity*, *LogOfAreas* and *Log_Y_Index* were dropped from the dataframe.

The new features after Data Exploration are :

TypeOfSteel_A300	outx	lumosindex	sqindex	stplthick
TypeOfSteel_A400	xmax	orindex	emptyindex	locon
Outside0	pixarea	logxindex	edgesindex	ymax
Outside1	xper	edgesx	maxlumos	
Outside0.5	sigarea	edgesy	minlumos	

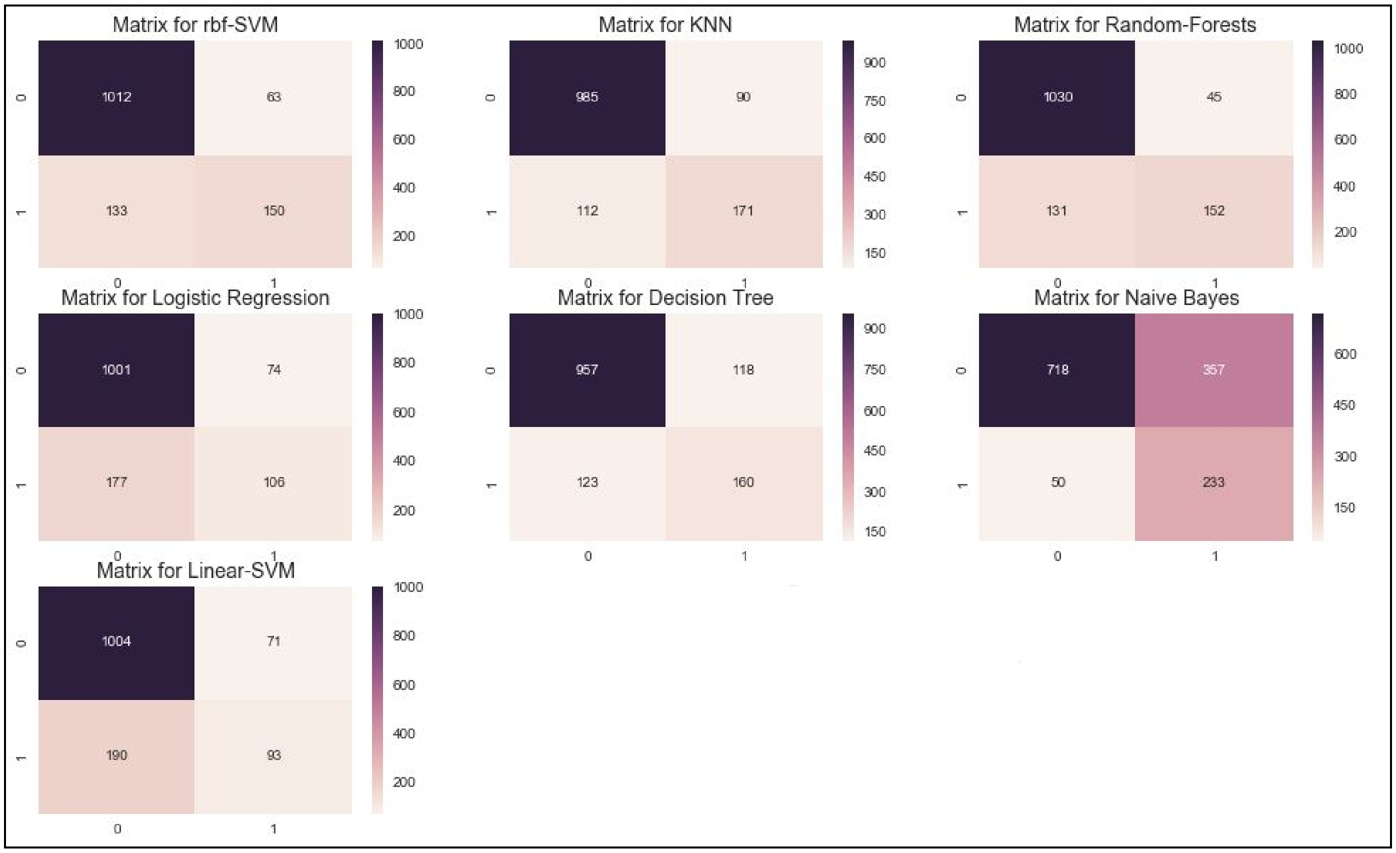
Next, the training dataset was split into two parts :- 70 % for train and 30 % for test.

Predictive Modelling

Different supervised algorithms were applied to predict whether the plate was faulty or not.
The best results after hyperparameter tuning are -:

Radial Support Vector Machine -: 86.03 %
Linear Support Vector Machine -: 81.86 %
Logistic Regression -: 81.73 %
Decision Tree -: 83.33 %
KNN -: 84.17 %
Gaussian Naive-Bayes -: 71.56 %
Random Forests -: 87.25 %

In order to get a better idea of the True positives, True negatives, False positives and False negatives, we made confusion matrices for each of the applied algorithms:



The left diagonal shows the number of correct predictions made for each class while the right diagonal shows the number of wrong predictions made.

Cross Validation :-

Many a times, the data is imbalanced. Thus we should train and test our algorithm on each and every instance of the dataset. Then we can take an average of all the noted accuracies over the dataset.

1) The K-Fold Cross Validation works by first dividing the dataset into k-subsets.

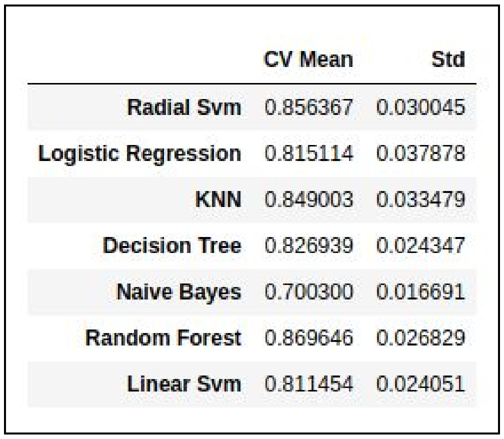
2)We divided the dataset into (k=10) parts. We reserve 1 part for testing and train the algorithm over the 9 parts.

3)We continue the process by changing the testing part in each iteration and training the algorithm over the other parts. The accuracies and errors are then averaged to get a average accuracy of the algorithm.

This is called K-Fold Cross Validation.

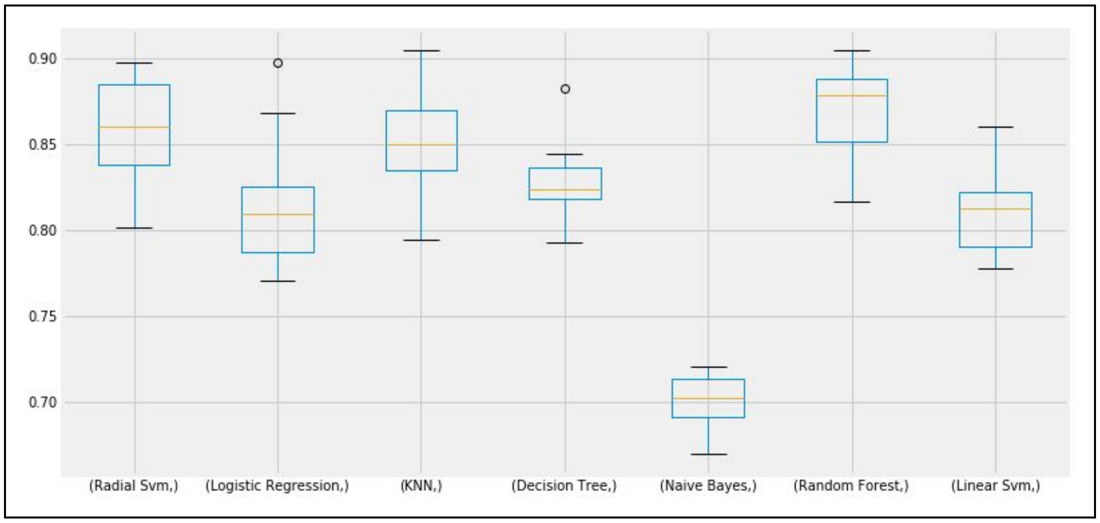
4)An algorithm may underfit over a dataset for some training data and sometimes also overfit the data for other training set. Thus with cross-validation, we can achieve a generalised model.

The results after 10-fold cross validation performing is -:



	CV Mean	Std
Radial Svm	0.856367	0.030045
Logistic Regression	0.815114	0.037878
KNN	0.849003	0.033479
Decision Tree	0.826939	0.024347
Naive Bayes	0.700300	0.016691
Random Forest	0.869646	0.026829
Linear Svm	0.811454	0.024051

Box Plot



Finally Ensemble methods were applied to get the best out of all the algorithms. Ensembling is a good way to increase the accuracy or performance of a model. In simple words, it is the combination of various simple models to create a single powerful model.

Ensembling Methods -

Voting Classifier

It is the simplest way of combining predictions from many different simple machine learning models. It gives an average prediction result based on the prediction of all the submodels. The submodels or the base models are all of different types.

Bagging

Bagging is a general ensemble method. It works by applying similar classifiers on small partitions of the dataset and then taking the average of all the predictions. Due to the averaging,there is reduction in variance. Unlike Voting Classifier, Bagging makes use of similar classifiers.

Bagged KNN & Bagged Decision Tree

Bagging works best with models with high variance. An example for this can be Decision Tree or Random Forests. We can use KNN with small value of **n_neighbours**, as small value of n_neighbours.

Boosting

Boosting is an ensembling technique which uses sequential learning of classifiers. It is a step by step enhancement of a weak model. Boosting works as follows:

A model is first trained on the complete dataset. Now the model will get some instances right while some wrong. Now in the next iteration, the learner will focus more on the wrongly predicted instances or give more weight to it. Thus it will try to predict the wrong instance correctly. Now this iterative process continuous, and new classifiers are added to the model until the limit is reached on the accuracy.

AdaBoost (Adaptive Boosting)

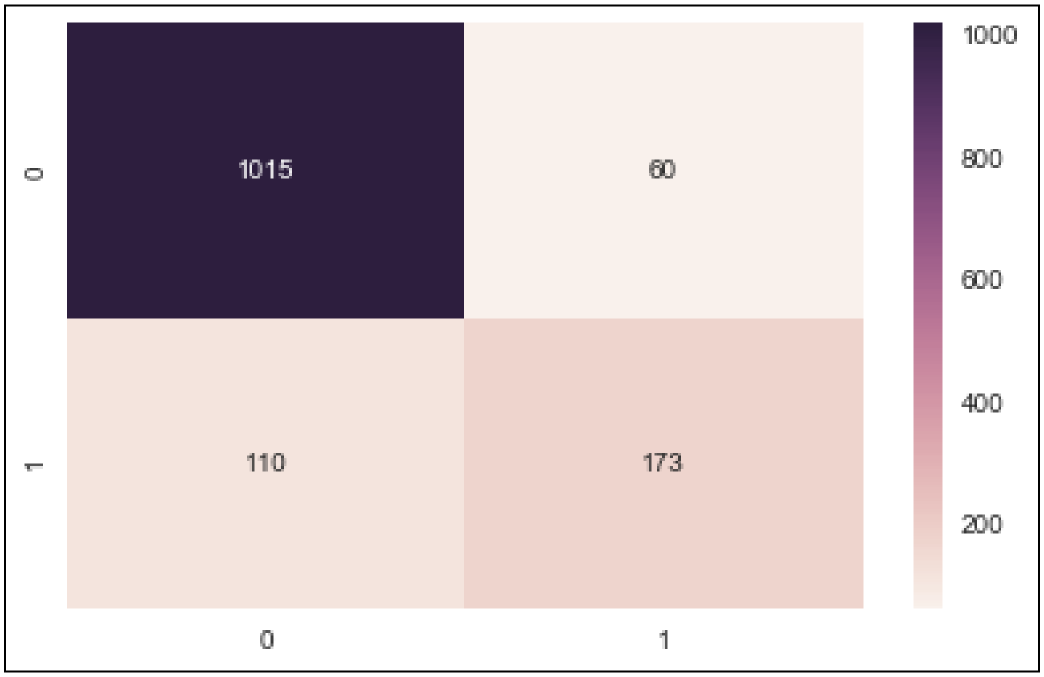
The weak learner or estimator in this case is a Decision Tree. But we can change the dafault base_estimator to any algorithm of our choice.

We have also applied **Stochastic Gradient Boosting** and **XgBoost**.

The various ensembling methods along with their accuracies are :

Voting Classifier	**-: 86.22 %**
Bagged KNN	**-: 85.05 %**
Bagged Decision Tree	**-: 87.19 %**
AdaBoost	**-: 84.46 %**
Stochastic Gradient Boosting	**-: 87.47 %**
XGBoost	**-: 86.96 %**

The confusion matrix for our best model (Stochastic Gradient Boosting) is as follows:



1015
60
110
173
0
1
0
1
1000
800
600
400
200