Project Report

on

Predicting the Activity Type of Accidents at Work using Ensemble Methods and Optimization Techniques

submitted by

Ayush Kumar	**Ravindra Kumar**	**Sreeram Jagannath. S**
17NA10009	17NA10023	17NA10029

under the guidance of

Prof. Jhareshwar Maiti

Industrial and Systems Engineering Department,

Indian Institute of Technology,

Kharagpur-721302

ACKNOWLEDGEMENT

We would like to show our gratitude to Prof. Jhareswar Maiti for giving us this project and believing in us. We would like to thank our project guide Mr. Krantiraditya Dhalmahapatra who constantly supervised us during the project. Finally, we would like to thank the department of “Industrial and Systems Engineering” for giving us this opportunity.

TABLE OF CONTENTS

Introduction

Now-a-days in any industry, occurrence of accidents during the work period are quite often, so they lead to different type of impact such as equipment property damage, fire, release of toxic gases and injury to workers present at the workplace etc. It has been a challenge to deal with these accidents. In order to overcome these challenges it is important to find out the reasons behind it and have some early preparation to avoid it, so for this we study on data of these incidents but the major issue to deal with this data is that it is found in large amount and unorganised form. So it is very difficult to get the relevant information or pattern from the large data and make some predictions. So, we need tools and techniques to organize, search and understand vast quantities of information.

The proposed methodology is shown in the figure below. We are interested in classifying the activity type of an incident as 'Operations' or 'Maintenance', and then, extracting rules for respective activity type. We describe a generic methodology here. The methodology consists of five steps:

(i) Data pre-processing,
(ii) Classification
(iii) Ensembling
(iv) Optimization
(v) Rule Extraction

The following sections describe the methodology, in details.

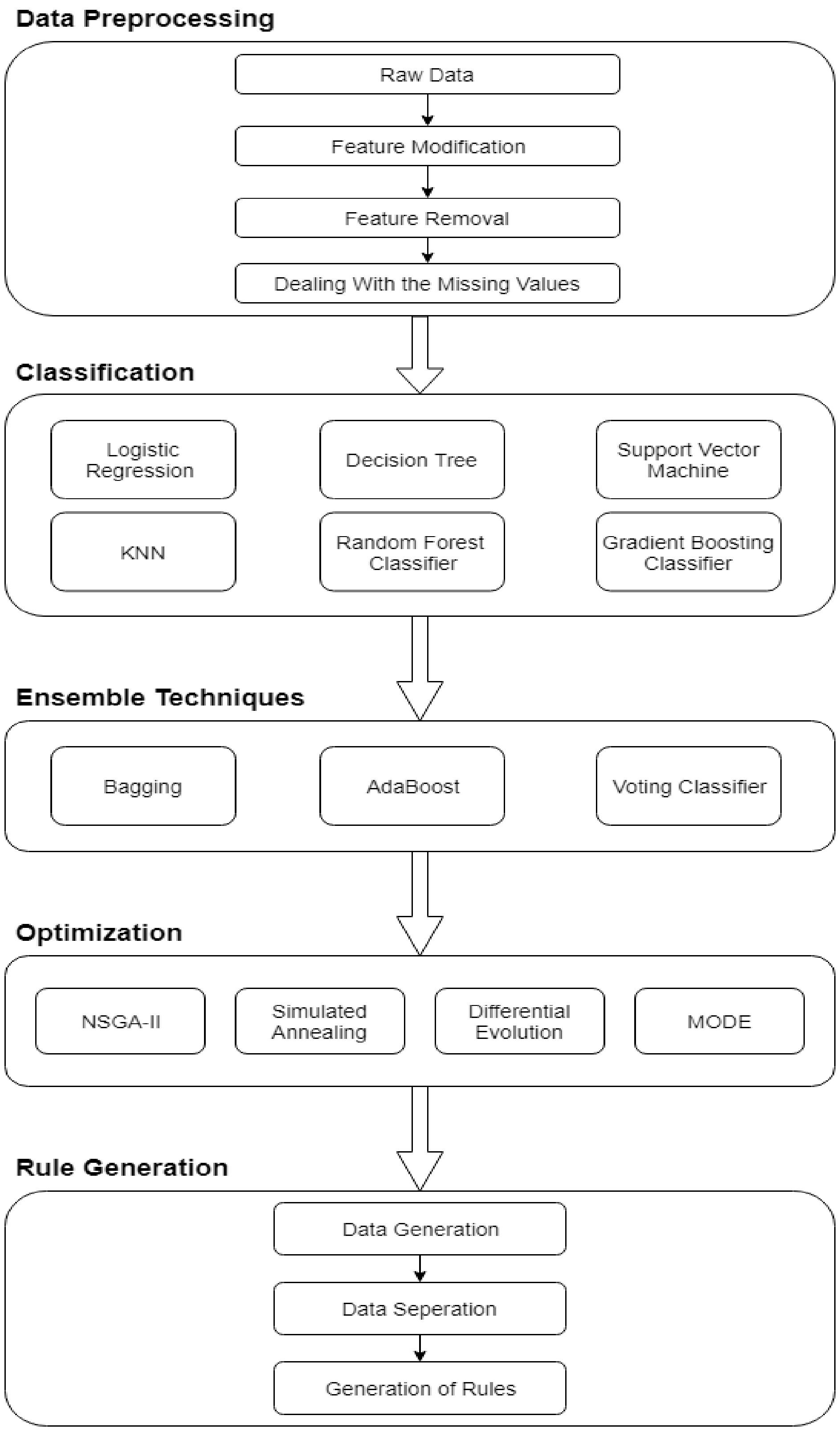


Figure: Flowchart of the proposed research methodology

Data Preprocessing

Data Description

The data we analysed was an industrial data, describing the accidents that happen during work. It provided us with the following information:

1. Date of accident
2. Time of accident
3. Incident Occurrence Division e.g., Wire division, Shared services, Steel manufacturing etc. A total of 16 categories.
4. Incident Category, only one category i.e., Near Miss.
5. Incident Location, (Workplace or mines). 4 categories in total.
6. Impact of accident e.g., First Aid, LTI, Equipment Property Damage etc. A total of 12 categories.
7. Activity Type (During which type of activity that accident occurred) e.g., Operations, Maintenance, Process Related, Construction Site etc. A total of 14 categories.
8. Primary Cause e.g., Fall from Height, fall of object, Slip/trip/fall etc. 49 categories in total.

Here are 5 samples from the given dataset.

Date	Time	Incident Occurrence Division	Incident Category	Incident Location	Impact	Activity Type	Primary Cause
10-May-18	6:40	Wire Division	Near Miss	Workplace	First Aid	Operations	Fall from height
26-Apr-18	10:30	Wire Division	Near Miss	Workplace	First Aid	Operations	Fall of object
14-Apr-18	9:10	Engineering and project	Near Miss	Workplace	First Aid	Construction Site	Slip/ trip/ fall
29-Jan-17	10:45	FAM	Near Miss	Workplace	First Aid	Maintenance	Slip/ trip/ fall
27-Nov-17	17:00	Shared Services	Near Miss	Workplace	LTI	Maintenance	Electrical fault

Data Cleaning

We perform data cleaning to have a data free of errors, to make it understandable and to have a higher accuracy. We had 716 records in our study. The cleaning was done in the following manner.

Feature Removal

We removed the features which had no significant effect on data processing while maintaining the integrity of the data in original form. We removed 'Incident Category' column because it had only one category 'Near Miss'. We also removed 'Incident Location' although it had 4 categories but categories other than 'Workplace' were in insignificant amount and so it would have no effect on data processing.

Feature modification

We changed the 'date' column to three separate columns as 'year', 'month' and 'days' and removed the 'date' column. We also changed the 'time' column to shift column as 'shift A' (6am to 2pm), 'shift B' (2pm to 10pm), 'shift C' (10pm to 6am).

We calculated the frequency of all the attributes and then set a threshold frequency for each of the features. Then for those attributes which were below the threshold frequency, we either deleted those rows containing attributes (for e.g., we deleted 'Toxic Release', 'Loose Soil', 'Roads') or renamed them to the attribute having the same meaning and having frequency above threshold frequency (for e.g., we renamed 'Open-Cast Maintenance' and 'Underground Maintenance' to 'maintenance' in 'Activity Type').

Dealing with missing values

There were 4 observations with missing dates and 4 observations which had only date and time. So we removed these 8 observations as they are not in significant amount.

Data Description after Data Cleaning

After the cleaning was done had 654 records for data processing. We were left with following features after cleaning the data:

Features	No. of Categories
Year	2
Month	12
Day	7
Shift	3
Incident Occurrence Division	11
Impact	7
Primary Cause	10
Activity Type	2

Here are 5 samples from our cleaned data:

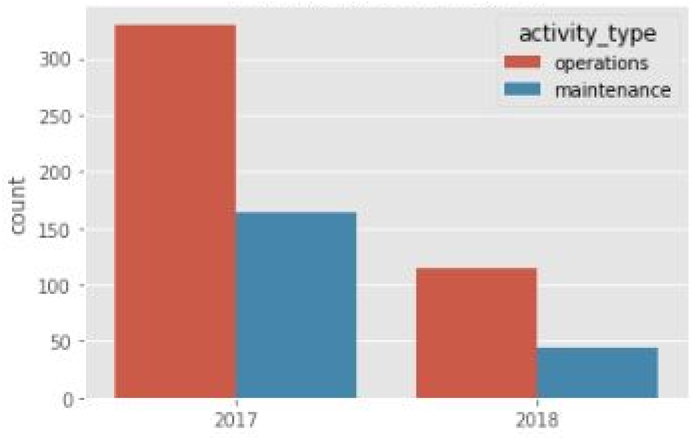
Year	Month	Day	Shift	Incident Occurrence Division	Impact	Activity Type	Primary Cause
2018	May	3	A	Wire division	First aid	Operation	Fall from height
2018	Jan	4	A	Fam	First aid	Maintenance	Slip/ trip/ fall
2017	Mar	3	B	West Bokaro	Derailment	Operation	Unsafe loading
2017	Nov	0	B	Shared services	LTI	Maintenance	Electrical fault
2018	Mar	6	A	Steel manufacturing	Equipment Property Damage	Operation	Fall of object

Exploratory Analysis of Data

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations.

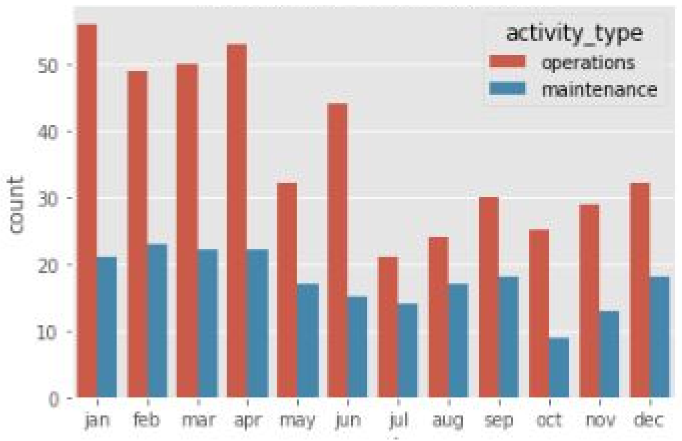
We take a look at each feature in the data and see how 'Activity Type' varies with each category in a feature.

Year



We have the data from year 2017 and 2018. The point to be noted is that observations in 2018 is collected only till the month of May and not the complete year. We can see that in 2017 accidents in 'Maintenance' is more than half the accidents in 'Operation' so we cannot draw any conclusions confidently while in 2018 although ratio of accident count is more than half but since no. of accidents in 2018 is low, we cannot draw any conclusions in 2018 too.

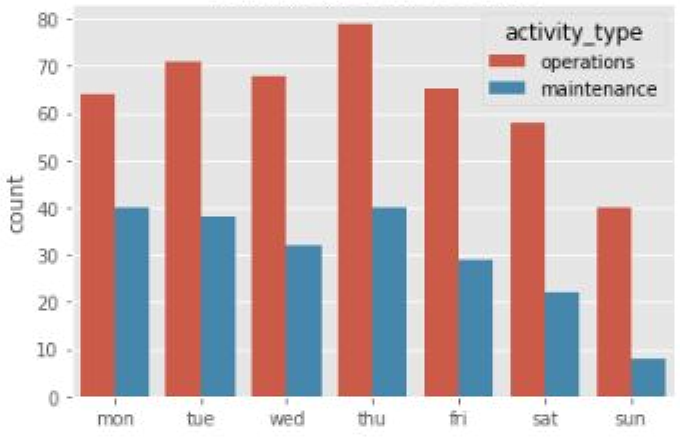
Month



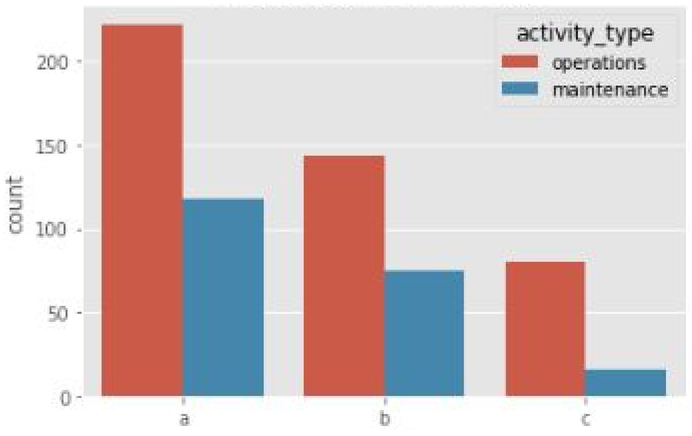
Here we can see the counts of 'Activity Type' across all the months. The trend observed is that incidents have occurred more during 'Operations' than 'Maintenance' throughout the year. Specifically in June and October, the ratio of accidents is quite high in favour of 'Operations'.

Day

The chart tells us that everyday, we see far more accidents in 'Operations' than 'Maintenance'. We notice the number of incidents reduce during the weekends. Thursday records for maximum number of incidents reported and Sunday for minimum.



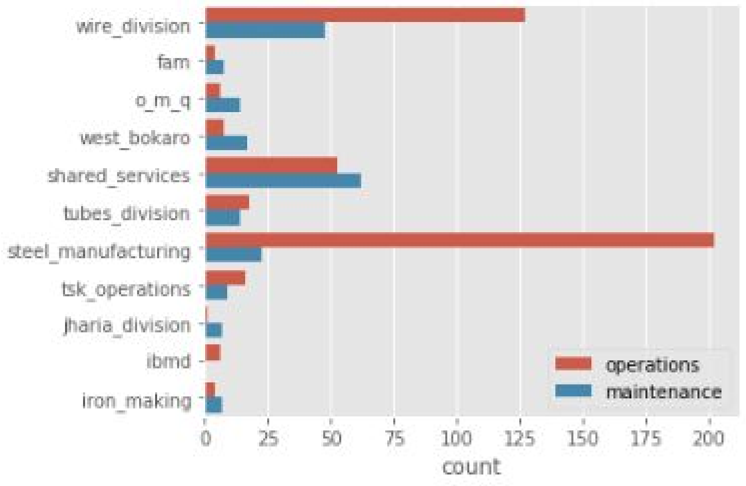
Shift



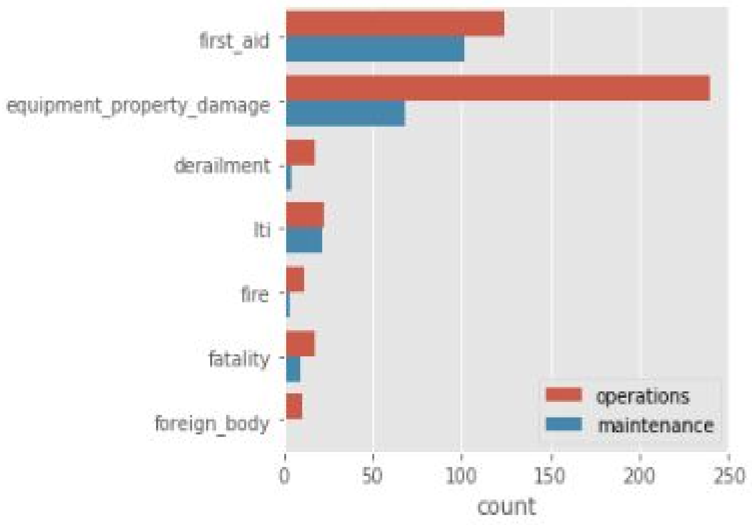
The incidents reported keeps on decreasing as the day progresses is what the data is telling us. This is probably because more accidents are likely to occur during the peak working hours. The counts of incidents in 'Operations' is significantly higher in Shift C.

Incident Occurrence Division

Here we observe that in some divisions, more accidents are reported during operations and in some other divisions, more accidents occur in maintenance. For e.g. in 'Steel Manufacturing' we can see that incidents during 'Operations' is way greater and



we can say for a particular data if an accident has occurred in 'Steel Manufacturing', it must have been during 'Operations' very confidently. We will see this when we generate the rules.

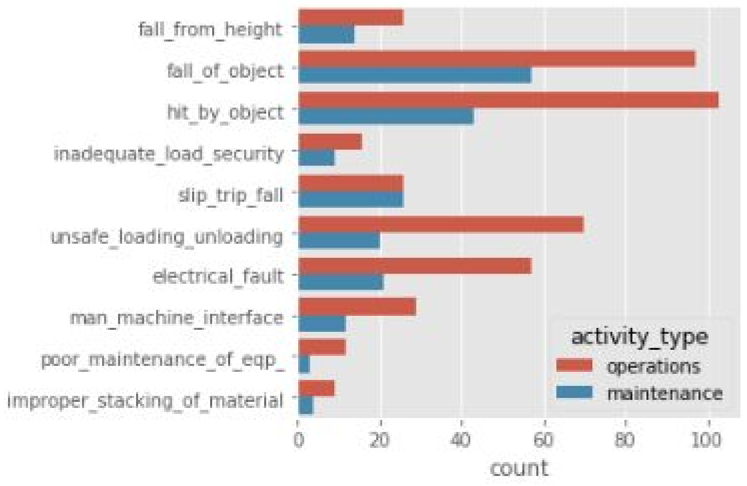


Impact

When impact type is 'Foreign Body', we see that generally accident took place during 'Operations'. Whenever an incident has occurred in 'Maintenance', the impact generally has been 'First Aid'.

Primary Cause

Here we plot the various causes of incidents across activity types. Most incidents in industry are caused due to falling of objects or getting hit by an object. When we generate rules we specifically find the main causes of incidents during 'Operations' and 'Maintenance' .



After the analysis of each feature of our data, we can move forward to training the model.

Classification

A classification problem is when the output variable is a category, such as ‘red’ or ‘blue’ or ‘disease’ and ‘no disease’. A classification model attempts to draw some conclusions from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. Here, using different machine learning algorithms, we try to predict whether the activity type has occurred during ‘Operations’ or ‘Maintenance’. We list out the accuracies obtained from each model at the end.

Logistic Regression

Logistic Regression is classification algorithm used to assign observation to a discrete set of class. Logistic Regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

Sigmoid function we use to map predicted values to probabilities.

$$S(z) = \frac{1}{(1+e^{-z})}$$

$$S(z) = \textit{output lies in } (0,1)$$
$$z = \textit{input to function}$$
$$e = \textit{base of natural log}$$

In order to map it to discrete class we set a threshold value. In this case we are taking it as binary logistic regression thus value above threshold we will take as class 1 and below threshold as class 2. We take 0.5 as threshold value.

$$p \geq 0.5,\ class = 1$$
$$p < 0.5,\ class = 0$$

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots\ldots.\ \ where,$$
$$\theta_1,\ \theta_2, \ldots..\ \textit{are the weights for the different features}$$
$$x_0,\ x_1,\ x_2, \ldots..\ \textit{are the different features}$$

We use the following cost function here called Cross Entropy. Cross-entropy loss can be divided into two separate cost functions: one for y=1 and one for y=0.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

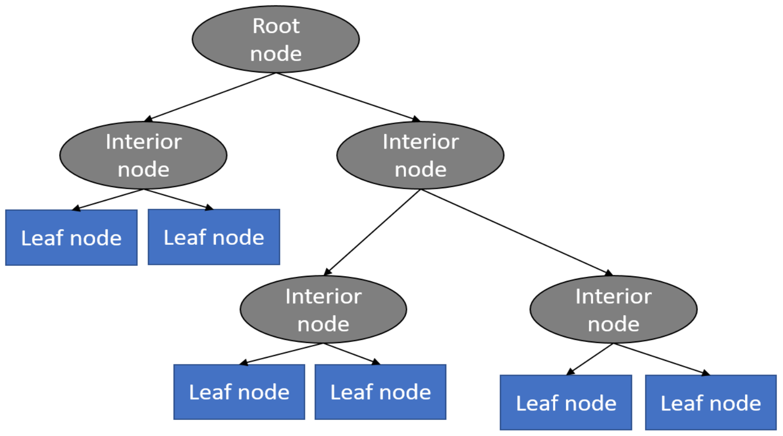
$$Cost(h_\theta(x), y) = -log(h_\theta(x)) \quad if\ y = 1$$

$$Cost(h_\theta(x), y) = -log(1 - h_\theta(x)) \quad if\ y = 0$$

In order to minimize the cost function we use gradient descent method.

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \qquad where\ \alpha\ is\ the\ learning\ rate.$$

Decision Tree Classifier



Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. So basically Decision Tree is a tree where each node represents a feature(attribute), each link represents a decision(rule) and each leaf represents an outcome.
To build Decision Tree one of the major challenges is to identify the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures.

Information Gain	→	ID3 (Iterative Dichotomiser 3)
Gini Index	→	CART (Classification and Regression Trees)

Gini Index

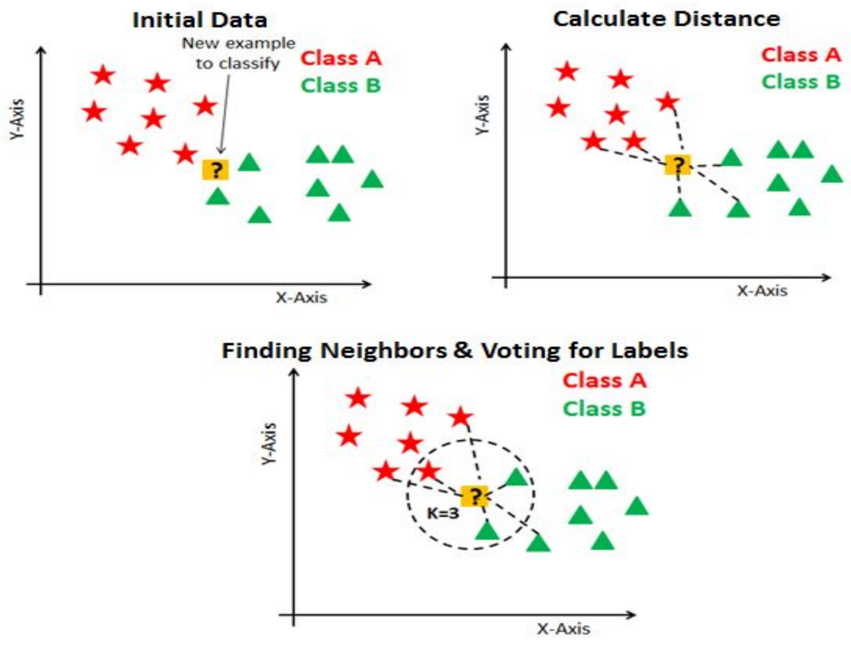
In CART we use Gini index as a metric. Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. We use the Gini Index as our cost function used to evaluate splits in the dataset. Our target variable is Binary variable which means it takes two values ('Operations' and 'Maintenance'). There can be 4 combinations. The attribute with lower Gini index should be preferred. The Formula for the calculation of the of the Gini Index is given below.

$$1 - \sum_{t=0}^{t=k} P_t^{\,2}$$

Maximum value of Gini Index could be when all target values are equally distributed. Minimum value of Gini Index will be 0 when all observations belong to one label.

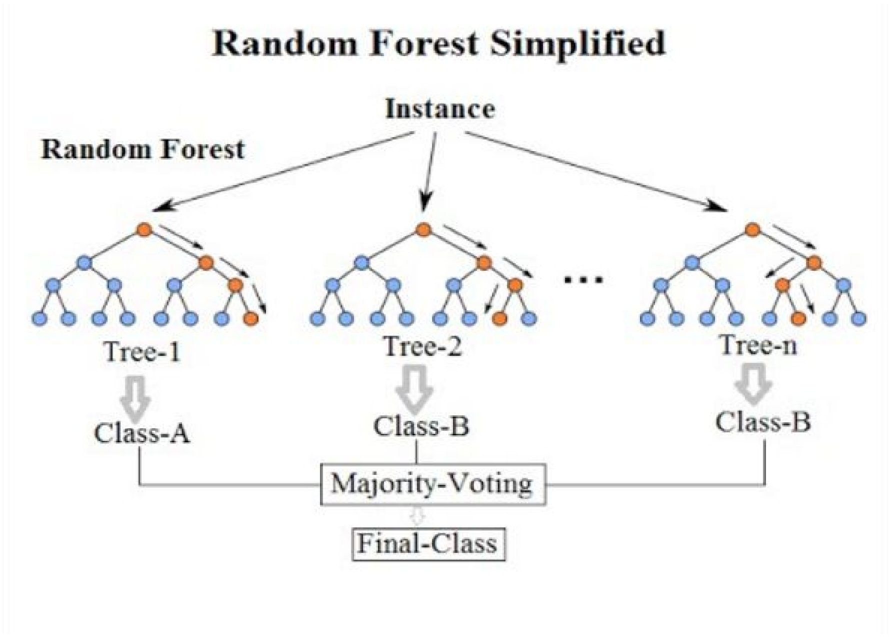
K-Nearest Neighbors

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data. In our case, we have taken the value of k to be 5.



Random Forest Classifier

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.



Approach:

1. Pick at random K data points from the training set.
2. Build the decision tree associated with those K data points.
3. Choose the number, N, of trees you want to build and repeat step 1 & 2.
4. For a new data point, make each one of your N trees predict the value of Y for the data point, and assign the new data point the average across all of the predicted Y values

Gradient Boosting Classifier

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The objective of any supervised learning algorithm is to define a loss function and minimize it.

Say we have mean squared error (MSE) as loss defined as:

$$Loss = MSE = \sum(y_i - y_i^p)^2$$

$$where,\ y_i = ith\ target\ value,\ y_i^p = ith\ prediction,\ L(y_i,\ y_i^p)\ \ is\ Loss\ function$$

We want our predictions, such that our loss function (MSE) is minimum. By using gradient descent and updating our predictions based on a learning rate, we can find the values where MSE is minimum.

$$y_i^p = y_i^P + \alpha * \delta \sum(y_i^p - y_i^p)^2 / \delta y_i^p$$

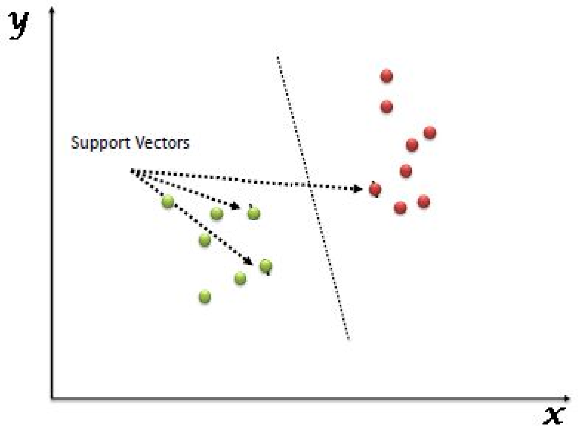
$$which\ becomes,\ y_i = y_i^p - \alpha * 2 * \sum(y_i - y_i^p)$$

$$where,\ \alpha\ is\ learning\ rate\ and\ \sum(y_i - y_i^p)\ \ is\ sum\ of\ residuals$$

So, we are basically updating the predictions such that the sum of our residuals is close to 0 (or minimum) and predicted values are sufficiently close to actual values.

Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).



To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier.

In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

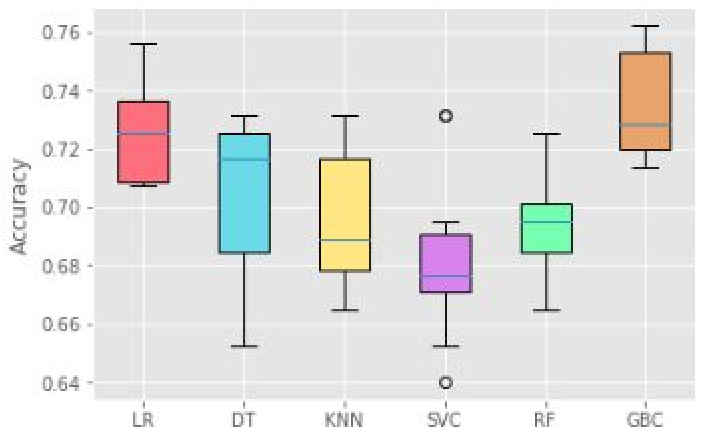
$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

Process:

We have a total of 654 observations. The first step is to split the total dataset into training set and testing set. Our goal is to create a model that generalizes well to new data. Our test set serves as a proxy for new data. We do a 75/25 split of the data resulting in 490 observations in our training set and 164 observations in test set. Then we train the model on the training set and get its accuracy on test set.

We do the 75/25 splitting of the data 10 times so that we get a general accuracy of the model. The table below gives us the mean accuracy across the 10 splits and the plot shows us the distribution of accuracy of each classifier.

Classifier	Accuracy
Logistic Regression	**0.726**
K-Nearest Neighbors	0.695
Support Vector Classifier	0.682
Decision Tree Classifier	**0.706**
Random Forest Classifier	0.694
Gradient Boosting Classifier	**0.735**



The 3 best classifiers that we get are *Gradient Boosting Classifier*, *Logistic Regression* and *Decision Tree Classifier*. We take these classifiers and move forward to the next step which is Ensembling.

Ensemble Models

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model.

Bagging

Bootstrap Aggregation (or Bagging for short), is a simple and very powerful ensemble method. Bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees.

Suppose there are N observations and M features. A sample from observation is selected randomly with replacement (Bootstrapping). A subset of features are selected to create a model with sample of observations and subset of features. Feature from the subset is selected which gives the best split on the training data. This is repeated to create many models and every model is trained in parallel. Prediction is given based on the aggregation of predictions from all the models.

AdaBoost

AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. The classifier should be trained interactively on various weighed training examples. In each iteration, it tries to provide an excellent fit for these examples by minimizing training error. Let’s look first at the equation for the final classifier.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

The final classifier consists of 'T' weak classifiers. $h_t(x)$ is the output of weak classifier 't'. α_t is the weight applied to classifier 't' as determined by AdaBoost.

The first classifier (t=1) is trained with equal probability given to all training examples. After it's trained, we compute the output weight (alpha) for that classifier.

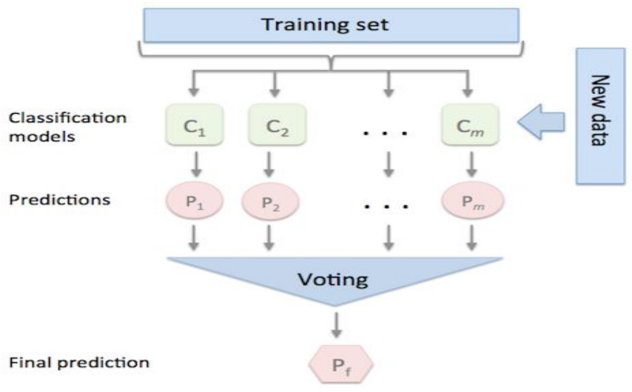
$$\alpha_t = \frac{1}{2} ln\left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$$

After computing the alpha for the first classifier, we update the training example weights using the following formula.

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x))}{Z_t}$$

Simple Voting Classifier

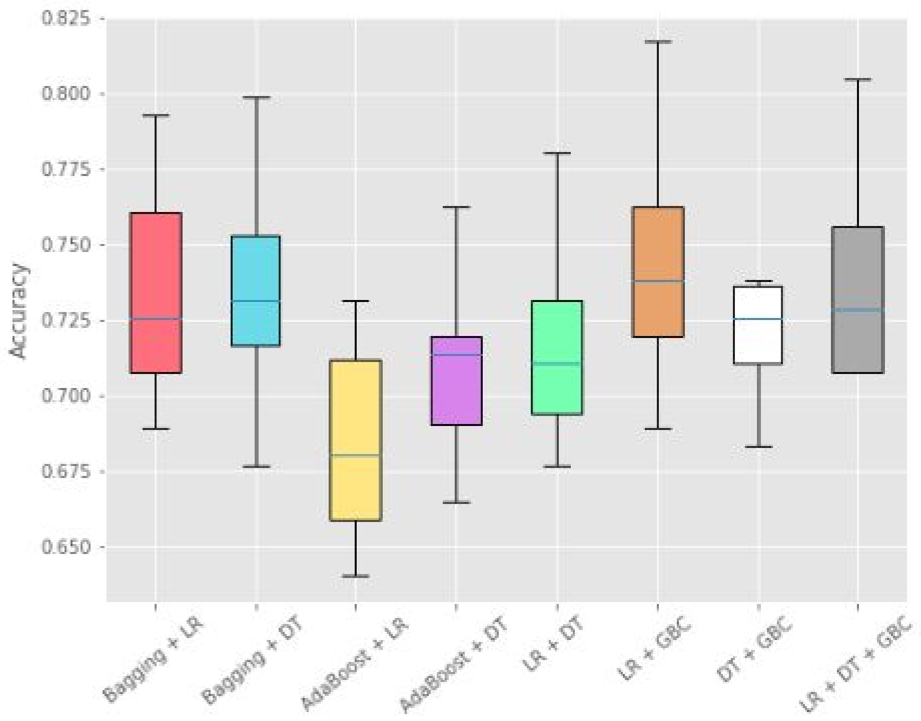
A Voting classifier model combines multiple different models (i.e., sub-estimators) into a single model, which is (ideally) stronger than any of the individual models alone. The predictions of the sub-model are weighted uniformly, i.e., the probabilities of the sub-model are averaged to find the final prediction



Process

We follow a similar process that we did before, we split the dataset into training and test sets 10 times randomly, train our ensemble models on training sets, get the accuracy from test set. The table below gives us the mean accuracy of different ensemble models and the plot shows us the distribution of accuracy of each ensemble model.

Ensembles	Accuracy
Bagging + LR	0.735
Bagging + DT	0.735
AdaBoost + LR	0.684
AdaBoost + DT	0.710
LR + DT	0.718
LR + GBC	**0.741**
DT + GBC	0.727
LR + DT + GBC	0.738



We can see that a *Simple Voting Classifier* of *Logistic Regression* and *Gradient Boosting Classifier* performs the best among all other ensembles. We had used uniform weights to get the final prediction of the ensemble, so now we need to think, can we assign optimal weights to each classifier such that the final accuracy is optimal. This thought brings us to the next step which is optimization of weights.

Optimization

Optimization is at the heart of almost all machine learning and statistical techniques used in data science.

Basic elements of optimization

There are three basic elements of any optimization problem -

- **Variables:** These are the free parameters which the algorithm can tune. In our case they are the weights of the classifier.
- **Constraints:** These are the boundaries within which the parameters must fall. In our case, we constrain the weights in the range (-1, 1).
- **Objective function:** This is the set of goal towards which the algorithm drives the solution. In case of multi-objective optimizations, we use the total *precision* and *recall* as our objectives. In case of single-objective optimization, we use the *accuracy* as our objective.

We have used four optimization techniques, two are single-objective and two are multi-objective techniques.

NSGA-II

The Non-dominated Sorting Genetic Algorithm or NSGA-II is one of the most popular multi-objective optimization algorithms with three special characteristics, fast non-dominated sorting approach, fast crowded distance estimation procedure and simple crowded comparison operator. Generally, NSGA-II can be roughly detailed as following steps.

Step 1: Population initialization

Initialize the population based on the problem range and constraint.

Step 2: Non-dominated sort

Sorting process based on non domination criteria of the population that has been initialized.

Step 3: Crowding distance

Once the sorting is complete, the crowding distance value is assigned front wise. The individuals in population are selected based on rank and crowding distance.

Step 4: Selection

The selection of individuals is carried out using a binary tournament selection with crowded-comparison operator .

Step 5: Genetic Operators

Real coded GA using simulated binary crossover and polynomial mutation.

Step 6: Recombination and selection

Offspring population and current generation population are combined and the individuals of the next generation are set by selection. The new generation is filled by each front subsequently until the population size exceeds the current population size.

Simulated Annealing

Simulated Annealing (SA) is an effective and general form of optimization. It is a single-objective optimization technique. It is useful in finding global optima in the presence of large numbers of local optima. 'Annealing' refers to an analogy with thermodynamics, specifically with the way that metals cool and anneal. Simulated annealing uses the objective function of an optimization problem instead of the energy of a material. This algorithm states that if a selected weight improves the accuracy, then it is always accepted. Otherwise, the algorithm selects that weight anyway with some probability less than 1. The probability decreases exponentially with the "badness" of the weight, which is the amount δE by which the solution is worsened (i.e., energy is increased.)

$$prob\ of\ accepting\ uphill\ move = 1 - e^{\delta E/kT}$$

A parameter T is also used to determine this probability. It is analogous to temperature in an annealing system. At higher values of T, uphill moves are more likely to occur. As T tends to zero, they become more and more unlikely, until the algorithm behaves more or less like hill-climbing. In a typical SA optimization, T starts high and is gradually decreased according to an “annealing schedule”. The parameter k is some constant that relates temperature to energy (in nature it is Boltzmann’s constant.)

Differential Evolution

Differential evolution is a stochastic population based method that is useful for global optimization problems. At each pass through the population the algorithm mutates each candidate solution by mixing with other candidate solutions to create a trial candidate. There are several strategies for creating trial candidates, which suit some problems more than others, but here we used ‘best1bin’ strategy. In this strategy two members of the population are randomly chosen. Their difference is used to mutate the best member (the best in ‘best1bin’), b_0, so far:

$$b' = b_0 + mutation * (pop[rand_0] - pop[rand_1])$$

A trial vector is then constructed. Starting with a randomly chosen ith parameter the trial is sequentially filled (in modulo) with parameters from b' or the original candidate. The choice of whether to use b' or the original candidate is made with a binomial distribution (the ‘bin’ in ‘best1bin’) - a random number in [0, 1) is generated. If this number is less than the recombination constant then the parameter is loaded from b', otherwise it is loaded from the original candidate. The final parameter is always loaded from b'. Once the trial candidate is built its fitness is assessed. If the trial is better than the original candidate then it takes its place. If it is also better than the best overall candidate it also replaces that.

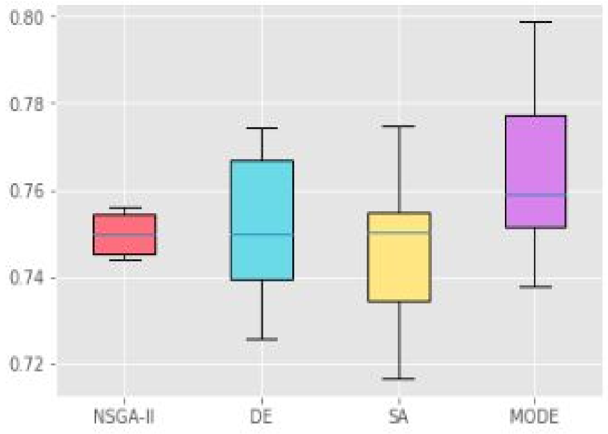
MODE

Multi-Objective Differential Evolution or MODE is a multi-objective variant of Differential Evolution. To design a multiobjective variant of differential evolution, the algorithm should be modified so that it can obtain a set of non-dominated solutions. This algorithm uses Pareto-based ranking assignment and crowding distance metric to find the optimal solution. A. In MODE, the fitness of a solution is first computed using Pareto-based ranking and then reduced with respect to the solution’s crowding distance value. This fitness values are then used to select the best solutions for the next generation.

Process

We follow the same process as before. We split the data into training and test sets with the training set containing 75 percent of the total data. Then we optimize the weights of the trained ensemble to get the optimal accuracy. This process has been performed 10 times. The mean of the accuracies has been provided in the table and the distributions of the accuracy has been provided in the box-plot besides the table.

Optimization Methods	Accuracy
NSGA-II	0.748
Differential Evolution	0.751
Simulated Annealing	0.753
MODE	**0.763**



We can see from the results that the Multi-Objective Differential Evolution works best for our data among other optimization techniques. So we move forward to the rule generation process using this trained ensemble and optimized weights.

Rule Extraction

Rule learning is always an important research area of machine learning. With the rules, we can easily see which features causes accidents during ‘Operations’ and which feature causes ‘Maintenance’, so that more precaution could be taken in future. The method we used for rule generation is described below.

Data Generation

Ensemble models are said to have strong generalization ability. Since we have a trained ensemble model already, we use the trained ensemble to generate a data set, then the new data set may contain more information helpful for prediction than the original data set does. So we generate a dataset which has 3,88,080 samples and we get the predictions from the trained ensemble. So now we have the required data to generate the rules.

Data Separation

We separate the generated dataset into two parts, one which has samples of ‘Operations’ and the other which has ‘Maintenance’. This data has been separated using a threshold. The ensemble outputs a probability which is between 0 and 1. Zero stands for ‘Operations’ and One stands for ‘Maintenance’. We have set the threshold for separating the ‘Operations’ class to be 0.2 and the threshold of 0.6 to get the ‘Maintenance’ data. We now generate two sets of rules, one for ‘Operations’ and one for ‘Maintenance’ using this separated data.

Generation of Rules

Once the data set D is ready, a rule set R can be generated. The rule generation process identifies a subset A of categorical attributes and then generates a rule based on it. At first, A comprises only one categorical attribute a_i randomly picked from available categorical attributes. If the number of instances possessing a particular value at a_i fall into class C, pass a certain threshold, then a rule is generated via regarding a_i as the antecedent and C as the consequent. Otherwise a_i is replaced by another categorical attribute a_j and a similar process occurs.

Rules

The rules shown below are in order of priority, the first being the one with highest priority.

Rules for activity type ‘Operations’:

RN	Cases (if)
R1	IOD => {Steel Manufacturing, IBMD}
R2	Impact => {Foreign Body}
R3	Month => {November, March, July} & PC => {Unsafe Loading}
R4	IOD => {Wire Division} & Impact => {Derailment}
R5	Month => {January} & Day => {Sunday} & PC => {Inadequate load security, Man machine interface}
R6	Day => {Sunday} & IOD => {Tubes Division} & Impact => {Eq. Property Damage}

Rules for activity type ‘Maintenance’:

RN	Cases (if)
R1	IOD => {Jharia Division}
R2	IOD => {Iron making} & Impact => {First aid, Fatality, LTI, Fire}
R3	IOD => {OMQ, Iron making, FAM} & Impact => {First aid}
R4	Shift => {A} & Impact => {First aid}
R5	Shift => {A} & IOD => {West Bokaro}
R6	IOD => {Shared services} & PC => {STF, Hit by object}
R7	Day => {Tuesday} & Impact => {LTI, Fatality} & PC => {Poor maintenance of Eqp.}
R8	Month => {December} & Impact => {Fire} & PC => {Electrical fault}

*RN = Rule Number

Conclusion

To perform the classification and optimization algorithms, Python 3.6 has been used. The libraries that have been used are *Scikit-Learn, Numpy, Pandas, Matplotlib, Scipy,* etc. Our objective in this project was to analyze the given data and find rules that can prevent further accidents. In this study we used some powerful classification and optimization techniques to predict whether the accident has occurred in 'Operation' or in 'Maintenance' and generate rules to prevent future accidents.

We started with six popular classifiers, eliminated three of them on the basis of accuracy and moved forward to ensembling. Three methods of ensembling were used out of which the best one was selected based on the accuracy. Then four different optimization methods were tried to optimize the weights of the ensemble out of which *MODE* provided the most optimal accuracy. So using the ensemble of *Logistic Regression* and *Gradient Boosting Classifier* with the weights optimized by *MODE*, we generated new data to generate rules.

14 safety rules have been extracted in this study. We found out the divisions where accidents occurred during 'Operations' and 'Maintenance'. Also 'Shift' and 'Primary Cause' which resulted in accidents in a particular activity type were also found. The 'Impact' due to accidents during 'Maintenance' were found to be more dangerous and hence workers can be more careful.