Universiteit Gent
imec
IDLab Department of Electronics
and Information Systems

For: Prof. Dr. Ir. Sofie Van Hoecke
Dr. Azarakhsh Jalalvand

By: Hima Nikafshan Rad

Confidential Internal Use
Weld break prediction challenge
Sep 2017

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

There is a company that welds the end of steel coils together in their production process. Often their production line succeeds in doing this. However, sometimes after the welding takes place, in a subsequent step of the production line, the weld breaks. This is unfortunate because the company has to stop the production line, get the steel coil out of a machine and fix the broken steel. To prevent welded steel from breaking, the company wants to use machine learning. By using machine learning, they in fact want to be able to predict if the welded steel coil will or will not break in one of the subsequent steps of the production line. To do so, they have provided us with data containing measurements taken during the welding process in addition to the data containing the settings of the welding machine and the properties of the material being welded. Using this data and machine learning, we should be able to predict if a steel coil will break in a next step or not. Hence, if we can predict if a steel coil will break, the company can divert the steel coil to be rewelded instead of bringing the production line to a halt.

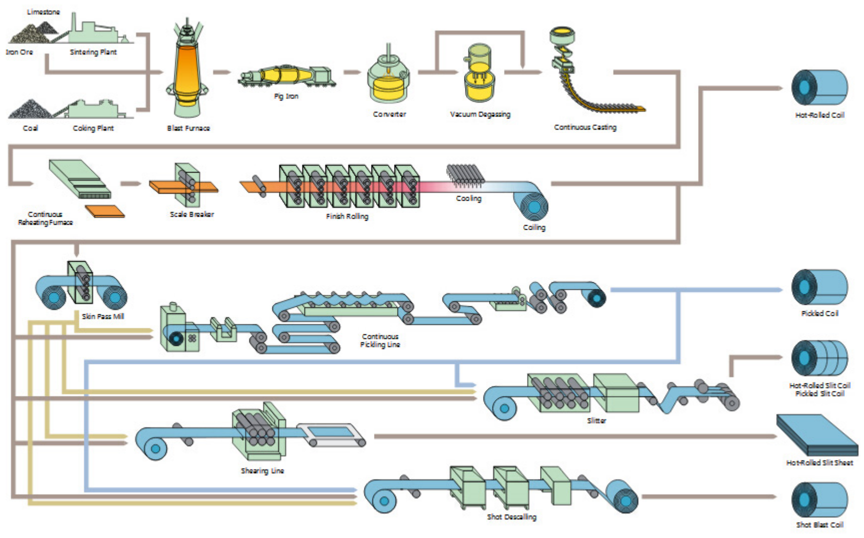

Figure 1.1: Steel Coil manufacturer

The steel coil industry is one of the most technology-evolving and capital-intensive market sectors. Effective prediction of fault detection in products is necessary to prevent abrupt products breaking down and is also benefit to improve productivity, reduce costs and repairing time. Machine Learning (ML) is the engineering of methods that enable computers to adapt their behavior based on empirical data. ML analyzes data and uses the theory of statistics to build mathematical models to predict events in the future. Machine learning methodologies are nowadays applied in many industrial and scientific environments including technology-intensive manufacturing and in general every data-intensive field that might benefit from reliable predictive capabilities, like the steel coil industry.

Machine learning is a scientific discipline that explores the construction and study of algorithms that can learn from data. Such algorithms operate by building a model based on inputs and using that to make predictions or decisions, rather than following only explicitly programmed instructions.

1.2 Outline

Machine learning is about the application of learning algorithm to build model that can best characterize underlying data and accurately predict the class of unlabeled data. In modern industry with numerous automated machines, a large amount of data has been automatically collected. Engineers may potentially use these raw data to identify specific hidden patterns such as the process fault model to assist the timely investigation of the root causes of the defects.

The organization of this report is shown in 1.2. In section 2, static features are considered with machine learning technique. A series of experimentation and results are presented in Section 2. Matrix Similarity is applied and results are shown in section 3. In section 4, time series prediction algorithms are designed and the results are considered. Finally, Section 5 concludes the report with discussion results and choose the best result.

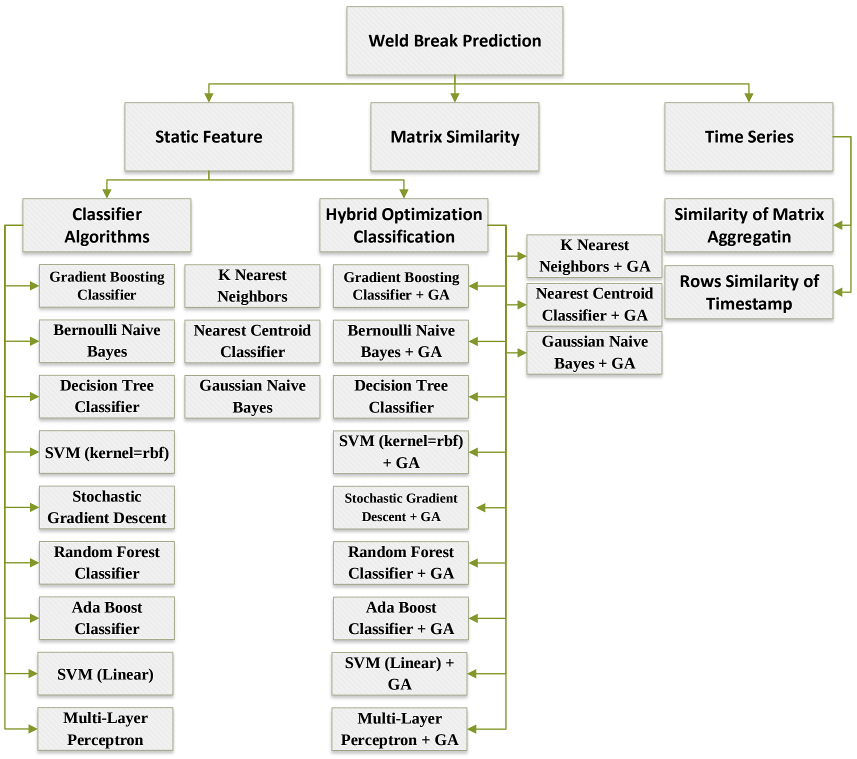
Weld Break Prediction
Static Feature
Matrix Similarity
Time Series
Classifier Algorithms
Hybrid Optimization Classification
Similarity of Matrix Aggregatin
Rows Similarity of Timestamp
Gradient Boosting Classifier
Bernoulli Naive Bayes
Decision Tree Classifier
SVM (kernel=rbf)
Stochastic Gradient Descent
Random Forest Classifier
Ada Boost Classifier
SVM (Linear)
Multi-Layer Perceptron
K Nearest Neighbors
Nearest Centroid Classifier
Gaussian Naive Bayes
Gradient Boosting Classifier + GA
Bernoulli Naive Bayes + GA
Decision Tree Classifier
SVM (kernel=rbf) + GA
Stochastic Gradient Descent + GA
Random Forest Classifier + GA
Ada Boost Classifier + GA
SVM (Linear) + GA
Multi-Layer Perceptron + GA
K Nearest Neighbors + GA
Nearest Centroid Classifier + GA
Gaussian Naive Bayes + GA

Figure 1.2: Weld Breaks Prediction Approaches

2

Static Features

2.1 Introduction

Classification is one of the most important methods and a vital tool in detection. There are many classification methods that are currently available; these methods can be used for categorizing multivariate data patterns. Additionally, many comparisons studies among different models were implemented on the weld break data set. Hence the main purpose of this report is to compare between the different types of classification methods in order to find the best algorithm. All of the classification methods and feature selection method have been used in this report on the weld break data set. This requires the data to be prepared in a particular and similar structure, in order to have consistent and accurate results for each classification approach. The structure of a classification algorithm is shown in 3.1.

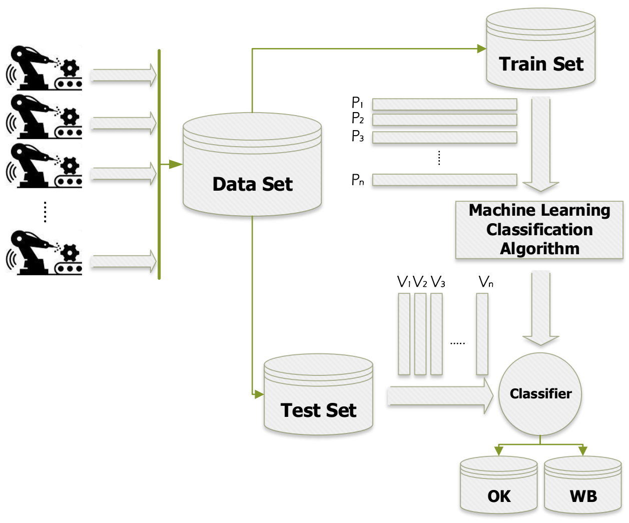

Figure 2.1: Structure of a classification algorithms

In this report, one type of technique was used for feature selection; the genetic algorithm was embedded for this. If we consider each type of signal as a feature, then feature selection may be applied to identify the most relevant signals. These method were used to determine key factors in the prediction process. This will enable an increase in process throughput, decreased time to learning.

2.2 Data Exploration

A steel coil manufacturing process is normally under consistent surveillance via the monitoring of signals/variables collected from sensors and or process measurement points. However, not all of these signals are equally valuable in a specific monitoring system. The measured signals contain a combination of useful information, irrelevant information as well as noise.

Lasparameter.Ingang - Coil ID 0282987S
Lasparameter.Ingang - Dikte (um) 2400
Lasparameter.Ingang - Breedte (mm) 1176
Lasparameter.Ingang - Materiaal 0002
Lasparameter.Uitgang - Coil ID 0282979S
Lasparameter.Uitgang - Dikte (um) 2400
Lasparameter.Uitgang - Breedte (mm) 1157
Lasparameter.Uitgang - Materiaal 0002
Lasparameter.P2 (mm) 40.0
Lasparameter.P3 (mm) 50.0

Lasparameter.P4 (mm) 58.1

Lasparameter.P5 (mm) 61.6

Lasparameter.P6 (mm) 66.0

Lasparameter.P7 (mm) 70.1

Lasparameter.Vonksnelheid (mm/s) 1.20

Lasparameter.Lassnelheid (mm/s) 0.48

Lasparameter.Stroom Referentie (A) 0223

Lasparameter.Stroomtoename (A/s2) 88.0

Lasparameter.Stuikstroomtijd (per) 0006

Lasparameter.Afkoeltijd Stuik (per) 0150

The dataset presented in this case represents a selection of such features where each example represents a single production entity with associated measured features and the labels represent a simple OK/WB yield for in production line testing, figure 2.2 and 2.3, and associated date time stamp. Where 0 corresponds to a pass(OK) and 1 corresponds to a fail(WB). It is clear, there is not significantly different between the behavior of features in fail and correct products.

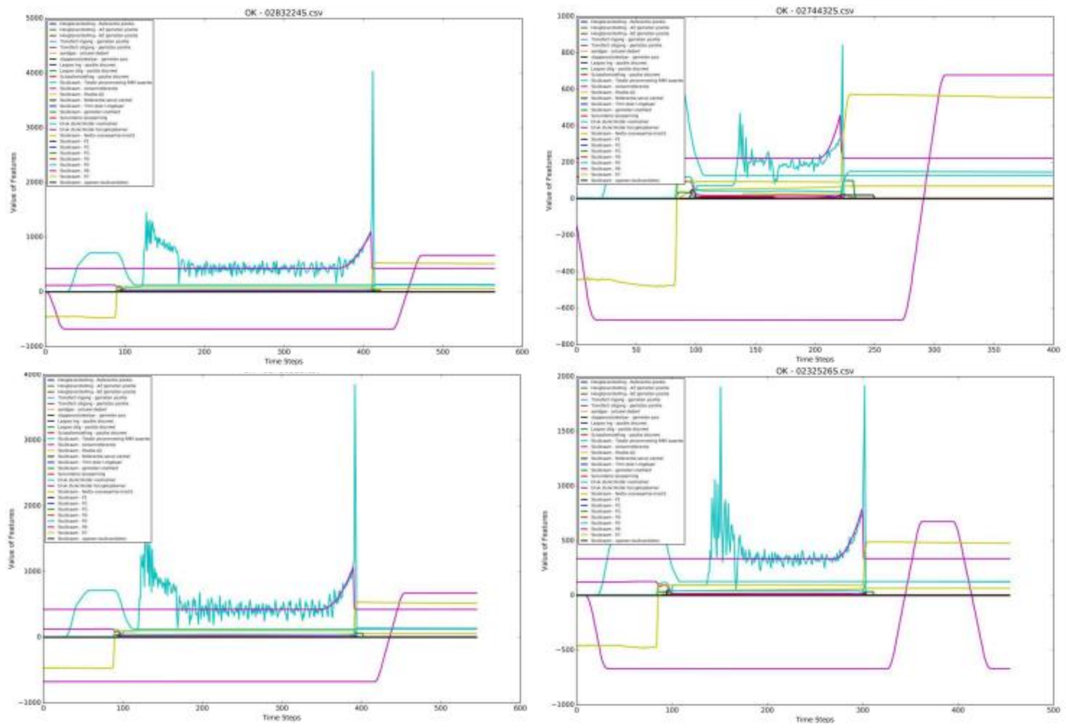

Figure 2.2: Behavioral of Products OK

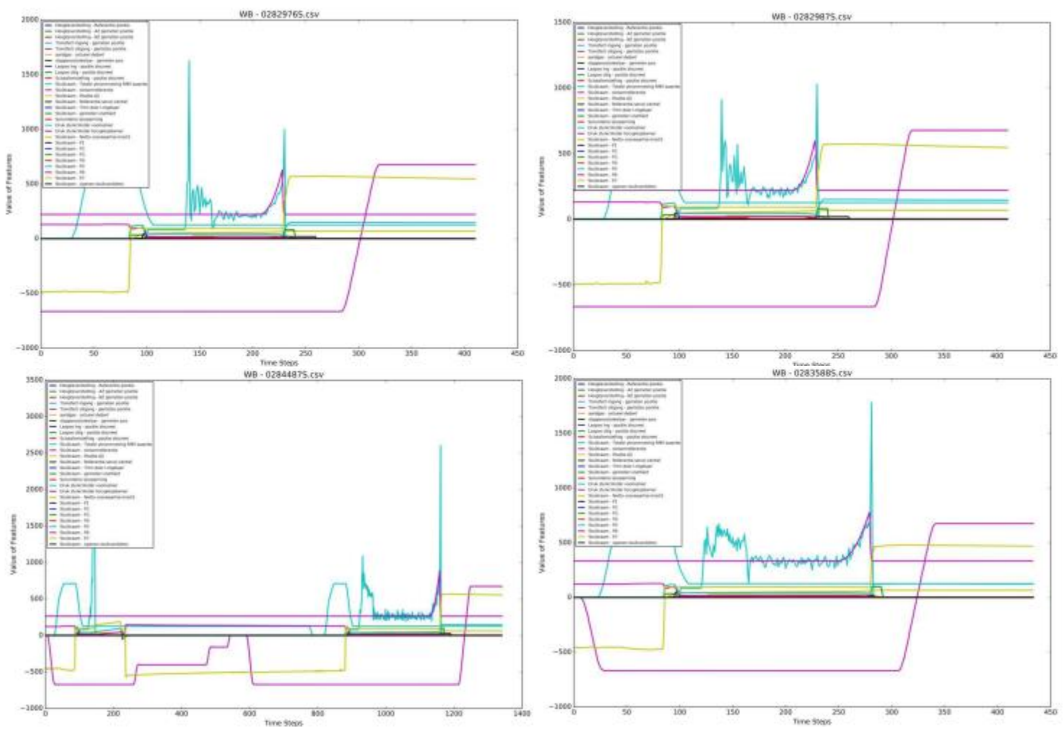

Figure 2.3: Behavioral of Products WB

2.3 Result

First, classifier algorithms in normal way on static features are used. In table 4.1, Random Forest Classifier has a good result for both of precision of WB and ROC.

Source: Code/Models/models.py

Model	Class 0 (WB)			Class 1 (OK)			
	Precision	Recall	F1	Precision	Recall	F1	ROC
Gradient Boosting Classifier	0.28	0.16	0.21	0.95	0.97	0.96	0.52
Bernoulli Naive Bayes	0.14	0.7	0.23	0.97	0.72	0.83	0.66
Decision Tree Classifier	0.22	0.33	0.26	0.96	0.93	0.94	0.59
SVM (kernel=rbf)	0.0	0.0	0.0	0.94	1.0	0.97	0.5
Stochastic Gradient Descent	0.21	0.08	0.12	0.94	0.98	0.96	0.56
Random Forest Classifier	**0.46**	0.18	0.26	0.95	0.99	0.97	**0.58**
Ada Boost Classifier	0.36	0.07	0.11	0.94	0.99	0.97	0.53
SVM (Linear)	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Multi-Layer Perceptron	0.30	0.21	0.25	0.95	0.97	0.96	0.63
K Nearest Neighbors	0.44	0.13	0.20	0.95	0.99	0.97	0.57
Nearest Centroid Classifier	0.14	0.72	0.23	0.98	0.71	0.82	0.66
Gaussian Naive Bayes	0.13	0.46	0.21	0.96	0.81	0.88	0.64

Table 2.1: Using Classifier Algorithms in normal way on static features.

Second, classifier algorithms with genetic algorithm as feature selection method on static features. In table 2.2, Ada Boost Classifier + GA and K Nearest Neighbors + GA have a good results for both of precisions. However the precision has increased, the recall is decrease. Then the ROC value was valuable and acceptable.

Source: Code/Models-GA

Model	Class 0 (WB)			Class 1 (OK)			
	Precision	Recall	F1	Precision	Recall	F1	ROC
Gradient Boosting Classifier + GA	0.50	0.11	0.19	1.0	0.95	0.99	0.60
Bernoulli Naive Bayes + GA	0.37	0.11	0.17	0.95	0.99	0.97	0.52
Decision Tree Classifier	0.35	0.38	0.37	0.96	0.96	0.96	0.66
SVM (kernel=rbf) + GA	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Stochastic Gradient Descent + GA	0.04	0.02	0.02	0.94	0.97	0.96	0.59
Random Forest Classifier + GA	0.55	0.26	0.36	0.95	0.99	0.97	0.59
Ada Boost Classifier + GA	**1.0**	0.02	0.03	0.94	1.00	0.97	**0.52**
SVM (Linear) + GA	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Multi-Layer Perceptron + GA	0.71	0.08	0.15	0.94	1.00	0.97	0.58
K Nearest Neighbors + GA	**1.00**	0.07	0.12	0.94	1.00	0.97	**0.53**
Nearest Centroid Classifier + GA	0.14	0.70	0.23	0.97	0.72	0.83	0.67
Gaussian Naive Bayes + GA	0.24	0.11	0.16	0.95	0.98	0.96	0.58

Table 2.2: Using classifier algorithms with genetic algorithm as feature selection method on static features.

Ada Boost Classifier with genetic algorithm had a good result, then it was run for 10 times . In table 2.3, it's clear that the results were acceptable for all.

Source: Code/Models/Models-GA/AB_GA.py

RUN #	Class 0 (WB)			Class 1 (OK)		
	Precision	Recall	F1	Precision	Recall	F1
1	1.00	0.02	0.03	0.94	1.00	0.97
2	1.00	0.03	0.06	0.94	1.00	0.97
3	1.00	0.03	0.06	0.94	1.00	0.97
4	1.00	0.02	0.03	0.94	1.00	0.97
5	1.00	0.02	0.03	0.94	1.00	0.97
6	1.00	0.03	0.06	0.94	1.00	0.97
7	1.00	0.03	0.06	0.94	1.00	0.97
8	1.00	0.02	0.03	0.94	1.00	0.97
9	1.00	0.03	0.06	0.94	1.00	0.97
10	1.00	0.02	0.03	0.94	1.00	0.97
Average	1.00	0.025	0.045	0.94	1.00	0.97

Table 2.3: 10 times run for Ada Boost Classifier + GA

K Nearest Neighbors with genetic algorithm had a good result, then it was run for 10 times . In table 2.3, it's clear that the results were acceptable for all, but Ada Boost Classifier + GA was a suitable.

Source: Code/Models/Models-GA/KNN_GA.py

RUN #	Class 0 (WB)			Class 1 (OK)		
	Precision	Recall	F1	Precision	Recall	F1
1	0.67	0.13	0.22	0.95	1.00	0.97
2	0.83	0.08	0.15	0.94	1.00	0.97
3	0.75	0.10	0.17	0.95	1.00	0.97
4	0.78	0.11	0.20	0.95	1.00	0.97
5	1.00	0.03	0.06	0.94	1.00	0.97
6	0.55	0.10	0.17	0.95	0.99	0.97
7	1.00	0.02	0.03	0.94	1.00	0.97
8	1.00	0.05	0.09	0.94	1.00	0.97
9	1.00	0.07	0.12	0.94	1.00	0.97
10	0.75	0.05	0.09	0.94	1.00	0.97
Average	0.833	0.074	0.13	0.944	0.999	0.97

Table 2.4: 10 times run for K Nearest Neighbors + GA

In this data, there is an imbalance problem. There are some algorithms to handle the imbalanced data problem same as sampling approaches. Experimental comparison of those methods and this sampling strategy are shown in table 2.5. In this table, Ada Boost Classifier with SamplereMethod = Random Under Sampler have a good result.

Source: Code/Imbalance-Models/imbalanced_models.py

	Class 0 (WB)			Class 1 (OK)			
Model	Precision	Recall	F1	Precision	Recall	F1	ROC
GradientBoostingClassifier with SamplereMethod = Near Miss Classifier	0.05	0.82	0.10	0.88	0.08	0.15	0.45
GradientBoostingClassifier with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.12	0.39	0.19	0.96	0.82	0.88	0.60
GradientBoostingClassifier with SamplereMethod = Edited Nearest Neighbours	0.31	0.34	0.33	0.96	0.95	0.95	0.64
GradientBoostingClassifier with SamplereMethod = Repeated Edited Nearest Neighbours	0.29	0.39	0.34	0.96	0.94	0.95	0.66
GradientBoostingClassifier with SamplereMethod = All KNN	0.27	0.36	0.31	0.96	0.94	0.95	0.64
GradientBoostingClassifier with SamplereMethod = Instance Hardness Threshold	0.19	0.38	0.25	0.96	0.90	0.93	0.63
GradientBoostingClassifier with SamplereMethod = Neighbour hood Cleaning Rule	0.37	0.21	0.27	0.95	0.98	0.96	0.59
GradientBoostingClassifier with SamplereMethod = OneSidedSelection	0.37	0.21	0.27	0.95	0.98	0.96	0.59
GradientBoostingClassifier with SamplereMethod = Random Under Sampler	0.10	0.52	0.17	0.96	0.69	0.80	0.60
GradientBoostingClassifier with SamplereMethod = TomekLinks(random_state=42)	0.35	0.26	0.30	0.95	0.97	0.96	0.61
Bernoulli Naive Bayes with SamplereMethod = Near Miss Classifier	0.04	0.56	0.08	0.86	0.18	0.30	0.36
Bernoulli Naive Bayes with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.12	0.64	0.21	0.97	0.71	0.82	0.67
Bernoulli Naive Bayes with SamplereMethod = Edited Nearest Neighbours	0.12	0.67	0.21	0.97	0.70	0.81	0.68
Bernoulli Naive Bayes with SamplereMethod = Repeated Edited Nearest Neighbours	0.12	0.67	0.21	0.97	0.70	0.81	0.68
Bernoulli Naive Bayes with SamplereMethod = All KNN	0.12	0.67	0.21	0.97	0.70	0.81	0.68
Bernoulli Naive Bayes with SamplereMethod = Instance Hardness Threshold	0.12	0.67	0.21	0.97	0.70	0.81	0.68
Bernoulli Naive Bayes with SamplereMethod = Neighbour hood Cleaning Rule	0.12	0.67	0.21	0.97	0.70	0.81	0.68
Bernoulli Naive Bayes with SamplereMethod = OneSidedSelection	0.12	0.67	0.21	0.97	0.70	0.81	0.68

Table 2.5: Experimental comparison of sampling strategy

Model	Class 0 (WB) Precision	Class 0 (WB) Recall	Class 0 (WB) F1	Class 1 (OK) Precision	Class 1 (OK) Recall	Class 1 (OK) F1	ROC
Bernoulli Naive Bayes with SamplereMethod = Random Under Sampler	0.13	0.69	0.21	0.97	0.70	0.81	0.69
Bernoulli Naive Bayes with SamplereMethod = TomekLinks(random_state=42)	0.12	0.67	0.21	0.97	0.70	0.81	0.68
DecisionTreeClassifier with SamplereMethod = Near Miss Classifier	0.06	0.87	0.11	0.92	0.09	0.17	0.47
DecisionTreeClassifier with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.14	0.52	0.23	0.96	0.80	0.88	0.66
DecisionTreeClassifier with SamplereMethod = Edited Nearest Neighbours	0.24	0.41	0.30	0.96	0.92	0.94	0.68
DecisionTreeClassifier with SamplereMethod = Repeated Edited Nearest Neighbours	0.22	0.48	0.30	0.96	0.89	0.93	0.69
DecisionTreeClassifier with SamplereMethod = All KNN	0.23	0.51	0.31	0.97	0.89	0.93	0.69
DecisionTreeClassifier with SamplereMethod = Instance Hardness Threshold	0.21	0.51	0.30	0.97	0.88	0.92	0.69
DecisionTreeClassifier with SamplereMethod = Neighbour hood Cleaning Rule	0.26	0.38	0.31	0.96	0.93	0.94	0.65
DecisionTreeClassifier with SamplereMethod = OneSidedSelection	0.29	0.34	0.32	0.96	0.95	0.95	0.64
DecisionTreeClassifier with SamplereMethod = Random Under Sampler	0.12	0.64	0.21	0.97	0.71	0.82	0.67
DecisionTreeClassifier with SamplereMethod = TomekLinks(random_state=42)	0.24	0.31	0.27	0.96	0.94	0.95	0.62
SVM (rbf) with SamplereMethod = Near Miss Classifier	0.05	0.74	0.09	0.87	0.12	0.20	0.42
SVM (rbf) with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (rbf) with SamplereMethod = Edited Nearest Neighbours	0.00	0.00	0.00	0.94	1.00	0.97	0.50
SVM (rbf) with SamplereMethod = Repeated Edited Nearest Neighbours	0.50	0.02	0.03	0.94	1.00	0.97	0.5
SVM (rbf) with SamplereMethod = All KNN	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (rbf) with SamplereMethod = Instance Hardness Threshold	0.00	0.00	0.00	0.94	1.00	0.97	0.49
SVM (rbf) with SamplereMethod = Neighbour hood Cleaning Rule	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (rbf) with SamplereMethod = OneSidedSelection	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (rbf) with SamplereMethod = Random Under Sampler	0.12	0.67	0.21	0.97	0.69	0.81	0.68

Table 2.6: Experimental comparison of sampling strategy(cont.)

	Class 0 (WB)			Class 1 (OK)			
Model	Precision	Recall	F1	Precision	Recall	F1	ROC
SVM (rbf) with SamplereMethod = TomekLinks(random_state=42)	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Stochastic Gradient Descent with SamplereMethod = Near Miss Classifier	0.05	0.56	0.09	0.91	0.30	0.45	0.42
Stochastic Gradient Descent with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.08	0.49	0.14	0.95	0.65	0.78	0.57
Stochastic Gradient Descent with SamplereMethod = Edited Nearest Neighbours	0.14	0.02	0.03	0.94	0.99	0.97	0.50
Stochastic Gradient Descent with SamplereMethod = Repeated Edited Nearest Neighbours	0.23	0.20	0.21	0.95	0.96	0.95	0.57
Stochastic Gradient Descent with SamplereMethod = All KNN	0.21	0.16	0.19	0.95	0.96	0.95	0.56
Stochastic Gradient Descent with SamplereMethod = Instance Hardness Threshold	0.06	0.20	0.09	0.94	0.81	0.87	0.50
Stochastic Gradient Descent with SamplereMethod = Neighbour hood Cleaning Rule	0.07	0.05	0.06	0.94	0.96	0.95	0.50
Stochastic Gradient Descent with SamplereMethod = OneSidedSelection	0.23	0.13	0.17	0.95	0.97	0.96	0.55
Stochastic Gradient Descent with SamplereMethod = Random Under Sampler	0.11	0.69	0.19	0.97	0.64	0.77	0.66
Stochastic Gradient Descent with SamplereMethod = TomekLinks(random_state=42)	0.24	0.30	0.27	0.95	0.94	0.95	0.61
Random Forest Classifier with SamplereMethod = Near Miss Classifier	0.05	0.80	0.10	0.87	0.08	0.15	0.44
Random Forest Classifier with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.21	0.38	0.27	0.96	0.91	0.93	0.64
Random Forest Classifier with SamplereMethod = Edited Nearest Neighbours	0.27	0.25	0.26	0.95	0.96	0.95	0.60
Random Forest Classifier with SamplereMethod = Repeated Edited Nearest Neighbours	0.23	0.30	0.26	0.95	0.94	0.94	0.61
Random Forest Classifier with SamplereMethod = All KNN	0.27	0.31	0.29	0.96	0.95	0.95	0.62
Random Forest Classifier with SamplereMethod = Instance Hardness Threshold	0.26	0.43	0.32	0.96	0.92	0.94	0.67
Random Forest Classifier with SamplereMethod = Neighbour hood Cleaning Rule	0.33	0.21	0.26	0.95	0.97	0.96	0.59
Random Forest Classifier with SamplereMethod = OneSidedSelection	0.50	0.20	0.28	0.95	0.99	0.97	0.59

Table 2.7: Experimental comparison of sampling strategy(cont.)

	Class 0 (WB)			Class 1 (OK)			
Model	Precision	Recall	F1	Precision	Recall	F1	ROC
Random Forest Classifier with SamplereMethod = Random Under Sampler	0.14	0.69	0.23	0.97	0.72	0.83	0.70
Random Forest Classifier with SamplereMethod = Tomek-Links(random_state=42)	0.32	0.10	0.15	0.94	0.99	0.97	0.54
Ada Boost Classifier with SamplereMethod = Near Miss Classifier	0.06	0.92	0.11	0.95	0.09	0.17	0.50
Ada Boost Classifier with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.25	0.28	0.27	0.95	0.95	0.95	0.61
Ada Boost Classifier with SamplereMethod = Edited Nearest Neighbours	0.39	0.18	0.25	0.95	0.98	0.97	0.58
Ada Boost Classifier with SamplereMethod = Repeated Edited Nearest Neighbours	0.35	0.31	0.33	0.96	0.96	0.96	0.63
Ada Boost Classifier with SamplereMethod = All KNN	0.36	0.15	0.21	0.95	0.98	0.97	0.56
Ada Boost Classifier with SamplereMethod = Instance Hardness Threshold	0.27	0.33	0.30	0.96	0.94	0.95	0.63
Ada Boost Classifier with SamplereMethod = Neighbour hood Cleaning Rule	0.43	0.10	0.16	0.95	0.99	0.97	0.54
Ada Boost Classifier with SamplereMethod = OneSidedSelection	0.50	0.08	0.14	0.94	0.99	0.97	0.53
Ada Boost Classifier with SamplereMethod = Random Under Sampler	0.15	0.77	0.26	0.98	0.73	0.84	0.75
Ada Boost Classifier with SamplereMethod = Tomek-Links(random_state=42)	0.45	0.08	0.14	0.94	0.99	0.97	0.53
SVM (linear) with SamplereMethod = Near Miss Classifier	0.06	0.69	0.11	0.93	0.27	0.42	0.48
SVM (linear) with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (linear) with SamplereMethod = Edited Nearest Neighbours	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (linear) with SamplereMethod = Repeated Edited Nearest Neighbours	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (linear) with SamplereMethod = All KNN	0.00	0.00	0.00	0.97	0.94	1.00	0.5
SVM (linear) with SamplereMethod = Instance Hardness Threshold	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (linear) with SamplereMethod = Neighbour hood Cleaning Rule	0.00	0.00	0.00	0.94	1.00	0.97	0.5

Table 2.8: Experimental comparison of sampling strategy(cont.)

Model	Class 0 (WB) Precision	Class 0 (WB) Recall	Class 0 (WB) F1	Class 1 (OK) Precision	Class 1 (OK) Recall	Class 1 (OK) F1	ROC
SVM (linear) with SamplereMethod = OneSidedSelection	0.00	0.00	0.00	0.94	1.00	0.97	0.5
SVM (linear) with SamplereMethod = Random Under Sampler	0.13	0.66	0.22	0.97	0.72	0.83	0.68
SVM (linear) with SamplereMethod = TomekLinks(random_state=42)	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Multi Layer Perceptron with SamplereMethod = Near Miss Classifier	0.06	0.89	0.11	0.92	0.09	0.16	0.48
Multi Layer Perceptron with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.16	0.39	0.23	0.96	0.87	0.91	0.63
Multi Layer Perceptron with SamplereMethod = Edited Nearest Neighbours	0.27	0.48	0.35	0.96	0.92	0.94	0.69
Multi Layer Perceptron with SamplereMethod = Repeated Edited Nearest Neighbours	0.20	0.52	0.29	0.97	0.87	0.91	0.69
Multi Layer Perceptron with SamplereMethod = All KNN	0.25	0.56	0.34	0.97	0.89	0.93	0.72
Multi Layer Perceptron with SamplereMethod = Instance Hardness Threshold	0.26	0.48	0.34	0.96	0.91	0.94	0.69
Multi Layer Perceptron with SamplereMethod = Neighbour hood Cleaning Rule	0.26	0.33	0.29	0.96	0.94	0.95	0.63
Multi Layer Perceptron with SamplereMethod = OneSidedSelection	0.28	0.28	0.28	0.95	0.96	0.95	0.61
Multi Layer Perceptron with SamplereMethod = Random Under Sampler	0.10	0.85	0.18	0.98	0.51	0.67	0.68
Multi Layer Perceptron with SamplereMethod = TomekLinks(random_state=42)	0.29	0.33	0.31	0.96	0.95	0.95	0.63
K Nearest Neighbors with SamplereMethod = Near Miss Classifier	0.05	0.79	0.10	0.89	0.11	0.19	0.44
K Nearest Neighbors with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.24	0.33	0.28	0.96	0.93	0.94	0.63
K Nearest Neighbors with SamplereMethod = Edited Nearest Neighbours	0.41	0.28	0.33	0.95	0.97	0.96	0.62
K Nearest Neighbors with SamplereMethod = Repeated Edited Nearest Neighbours	0.30	0.34	0.32	0.96	0.95	0.95	0.64
K Nearest Neighbors with SamplereMethod = All KNN	0.33	0.34	0.34	0.96	0.96	0.96	0.64
K Nearest Neighbors with SamplereMethod = Instance Hardness Threshold	0.31	0.33	0.32	0.96	0.95	0.95	0.64

Table 2.9: Experimental comparison of sampling strategy(cont.)

	Class 0 (WB)			Class 1 (OK)			
Model	Precision	Recall	F1	Precision	Recall	F1	ROC
K Nearest Neighbors with SamplereMethod = Neighbour hood Cleaning Rule	0.45	0.23	0.30	0.95	0.98	0.97	0.60
K Nearest Neighbors with SamplereMethod = OneSidedSelection	0.54	0.21	0.31	0.95	0.99	0.97	0.60
K Nearest Neighbors with SamplereMethod = Random Under Sampler	0.10	0.64	0.18	0.97	0.65	0.78	0.64
K Nearest Neighbors with SamplereMethod = TomekLinks(random_state=42)	0.54	0.21	0.31	0.95	0.99	0.97	0.60
Nearest Centroid Classifier with SamplereMethod = Near Miss Classifier	0.04	0.57	0.08	0.85	0.15	0.25	0.36
Nearest Centroid Classifier with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.12	0.49	0.19	0.96	0.77	0.85	0.62
Nearest Centroid Classifier with SamplereMethod = Edited Nearest Neighbours	0.12	0.66	0.20	0.97	0.70	0.81	0.67
Nearest Centroid Classifier with SamplereMethod = Repeated Edited Nearest Neighbours	0.13	0.69	0.21	0.97	0.70	0.81	0.69
Nearest Centroid Classifier with SamplereMethod = All KNN	0.12	0.67	0.21	0.97	0.70	0.81	0.67
Nearest Centroid Classifier with SamplereMethod = Instance Hardness Threshold	0.13	0.69	0.21	0.97	0.69	0.81	0.69
Nearest Centroid Classifier with SamplereMethod = Neighbour hood Cleaning Rule	0.12	0.66	0.20	0.97	0.69	0.81	0.67
Nearest Centroid Classifier with SamplereMethod = OneSidedSelection	0.11	0.57	0.18	0.96	0.70	0.81	0.63
Nearest Centroid Classifier with SamplereMethod = Random Under Sampler	0.11	0.57	0.18	0.96	0.70	0.81	0.63
Nearest Centroid Classifier with SamplereMethod = TomekLinks(random_state=42)	0.12	0.64	0.20	0.97	0.70	0.81	0.66
Guassian Naive Bayes with SamplereMethod = Near Miss Classifier	0.05	0.61	0.09	0.90	0.21	0.35	0.41
Guassian Naive Bayes with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.14	0.49	0.22	0.96	0.81	0.88	0.65
Guassian Naive Bayes with SamplereMethod = Edited Nearest Neighbours	0.13	0.52	0.20	0.96	0.77	0.85	0.64
Guassian Naive Bayes with SamplereMethod = Repeated Edited Nearest Neighbours	0.13	0.54	0.21	0.96	0.76	0.85	0.65

Table 2.10: Experimental comparison of sampling strategy(cont.)

Model	Class 0 (WB) Precision	Class 0 (WB) Recall	Class 0 (WB) F1	Class 1 (OK) Precision	Class 1 (OK) Recall	Class 1 (OK) F1	ROC
Guassian Naive Bayes with SamplereMethod = All KNN	0.12	0.54	0.20	0.96	0.76	0.85	0.64
Guassian Naive Bayes with SamplereMethod = Instance Hardness Threshold	0.12	0.61	0.20	0.97	0.72	0.83	0.66
Guassian Naive Bayes with SamplereMethod = Neighbour hood Cleaning Rule	0.12	0.49	0.20	0.96	0.78	0.86	0.63
Guassian Naive Bayes with SamplereMethod = OneSidedSelection	0.14	0.48	0.21	0.96	0.81	0.88	0.64
Guassian Naive Bayes with SamplereMethod = Random Under Sampler	0.12	0.66	0.20	0.97	0.69	0.81	0.50
Guassian Naive Bayes with SamplereMethod = TomekLinks(random_state=42)	0.13	0.49	0.20	0.13	0.49	0.20	0.52
Linear Discriminant Analysis with SamplereMethod = Near Miss Classifier	0.06	0.72	0.12	0.95	0.31	0.47	0.51
Linear Discriminant Analysis with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.24	0.30	0.26	0.95	0.94	0.95	0.61
Linear Discriminant Analysis with SamplereMethod = Edited Nearest Neighbours	0.19	0.05	0.08	0.94	0.99	0.96	0. 51
Linear Discriminant Analysis with SamplereMethod = Repeated Edited Nearest Neighbours	0.25	0.13	0.17	0.95	0.97	0.96	0.55
Linear Discriminant Analysis with SamplereMethod = All KNN	0.32	0.13	0.19	0.95	0.98	0.96	0.55
Linear Discriminant Analysis with SamplereMethod = Instance Hardness Threshold	0.24	0.18	0.21	0.95	0.96	0.96	0.55
Linear Discriminant Analysis with SamplereMethod = Neighbour hood Cleaning Rule	0.25	0.05	0.08	0.94	0.99	0.97	0.51
Linear Discriminant Analysis with SamplereMethod = OneSidedSelection	0.25	0.05	0.08	0.94	0.99	0.97	0.51
Linear Discriminant Analysis with SamplereMethod = Random Under Sampler	0.14	0.67	0.24	0.97	0.74	0.84	0.70
Linear Discriminant Analysis with SamplereMethod = TomekLinks(random_state=42)	0.30	0.05	0.08	0.94	0.99	0.97	0.52
Quadratic Discriminant Analysis with SamplereMethod = Near Miss Classifier	0.05	0.80	0.10	0.86	0.08	0.14	0.43
Quadratic Discriminant Analysis with SamplereMethod = Condensed Nearest Neighbour Naive Bayes	0.10	0.03	0.05	0.94	0.98	0.96	0.50

Table 2.11: Experimental comparison of sampling strategy(cont.)

	Class 0 (WB)			Class 1 (OK)			
Model	Precision	Recall	F1	Precision	Recall	F1	ROC
Quadratic Discriminant Analysis with SamplereMethod = Edited Nearest Neighbours	0.14	0.05	0.07	0.94	0.98	0.96	0.51
Quadratic Discriminant Analysis with SamplereMethod = Repeated Edited Nearest Neighbours	0.11	0.10	0.10	0.94	0.95	0.95	0.52
Quadratic Discriminant Analysis with SamplereMethod = All KNN	0.13	0.15	0.14	0.95	0.94	0.94	0.54
Quadratic Discriminant Analysis with SamplereMethod = Instance Hardness Threshold	0.22	0.03	0.06	0.94	0.99	0.97	0.51
Quadratic Discriminant Analysis with SamplereMethod = Neighbour hood Cleaning Rule	0.13	0.08	0.10	0.94	0.97	0.95	0.52
Quadratic Discriminant Analysis with SamplereMethod = OneSided-Selection	0.06	0.43	0.10	0.94	0.54	0.69	0.48
Quadratic Discriminant Analysis with SamplereMethod = Random Under Sampler	0.09	0.05	0.06	0.94	0.97	0.95	0.50
Quadratic Discriminant Analysis with SamplereMethod = Tomek-Links(random_state=42)	0.30	0.05	0.08	0.94	0.99	0.97	0.52

Table 2.12: Experimental comparison of sampling strategy.

In this regard, another algorithm is bagging . Experimental comparison of those methods and this strategy are shown in table 2.13. In this table, Bagging Classifier on Random Forest Classifier have a good result for both of precision of WB and ROC.

Source: Code/Ensemble-Learning/Bagging/ensemble-bagging.py

	Class 0 (WB)			Class 1 (OK)			
Model	Precision	Recall	F1	Precision	Recall	F1	ROC
Bagging Classifier on Bernoulli Naive Bayes	0.13	0.61	0.21	0.97	0.74	0.84	0.67
Bagging Classifier on DecisionTreeClassifier	0.52	0.23	0.32	0.95	0.99	0.97	0.60
Bagging Classifier on SVM (rbf)	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Bagging Classifier on Stochastic Gradient Descent	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Bagging Classifier on Random Forest Classifier	**0.75**	0.05	0.09	0.94	1.00	0.97	**0.52**
Bagging Classifier on SVM (linear)	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Bagging Classifier on Multi Layer Perceptron	0.00	0.00	0.00	0.94	1.00	0.97	0.5
Bagging Classifier on K Nearest Neighbors	0.80	0.07	0.12	0.94	1.00	0.97	0.53
Bagging Classifier on Nearest Centroid Classifier	0.13	0.66	0.21	0.97	0.71	0.82	0.68
Bagging Classifier on Guassian Naive Bayes	0.15	0.36	0.21	0.96	0.87	0.91	0.61
Bagging Classifier on Linear Discriminant Analysis	0.00	0.00	0.00	0.94	1.00	0.97	0.49
Bagging Classifier on Quadratic Discriminant Analysis	0.06	1.00	0.11	0.00	0.00	0.00	0.5

Table 2.13: Bagging Classifier Algorithm

Another approach for improving this result is ensemble learning . The result of this strategy is shown in table 2.14. This algorithm is also have a good result for both of precision of WB and ROC.

Source: Code/Ensemble-Learning/Voting/ensemble-voting.py

	Class 0 (WB)			Class 1 (OK)			Total			
Model	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	ROC
WAPEL	0.56	0.08	0.14	0.94	1.00	0.97	0.92	0.94	0.92	0.54

Table 2.14: Weighted Average Probabilities (Soft Voting) Ensemble Learning

In this regard, the best algorithm for the ROC result was Linear Discriminant Analysis with sampler ”Random Under Sampler”+GA. The result is shown in table 2.15. In this table, Linear Discriminant Analysis with sampler ”Random Under Sampler”+GA have a good result ROC.

Source: Code/Imbalance-Models/imbalanced_models.py

	Class 0 (WB)			Class 1 (OK)			Total			
Model	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	ROC
Sampler GA	0.14	0.84	0.24	0.98	0.67	0.80	0.93	0.68	0.76	0.75

Table 2.15: Linear Discriminant Analysis with sampler ”Random Under Sampler”+GA

3
Matrix Similarity

3.1 Introduction

In statistics and related fields, a similarity measure or similarity function is a real-valued function that quantifies the similarity between two objects. Although no single definition of a similarity measure exists, usually such measures are in some sense the inverse of distance metrics: they take on large values for similar objects and either zero or a negative value for very dissimilar objects. In this part, one kind of approaches is presented.

3.2 Proposed Algorithm

For weld break prediction, all 3170 datasets were randomly divided into training and testing phases for selecting the most precise model using 70% and 30% . In other words, 2219 and 951 datasets of the whole datasets were used as training and testing datasets, in order. It is a well-established fact that a predictor with high precision of WB.

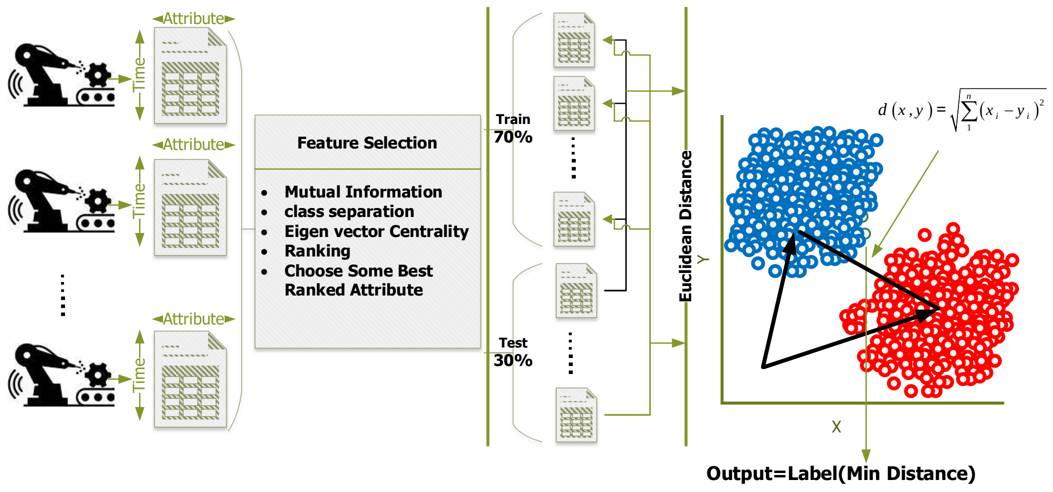

Figure 3.1: Matrix Similarity Algorithm

As given in Figure 3.1, this algorithm was implemented. In this work, feature selection via eigenvector centrality was used. This method is a filter method which maps the feature selection problem on an affinity graph where features are the nodes. the solution is given by assessing the importance of nodes through some indicators of centrality, in particular, the eigenvector centrality.

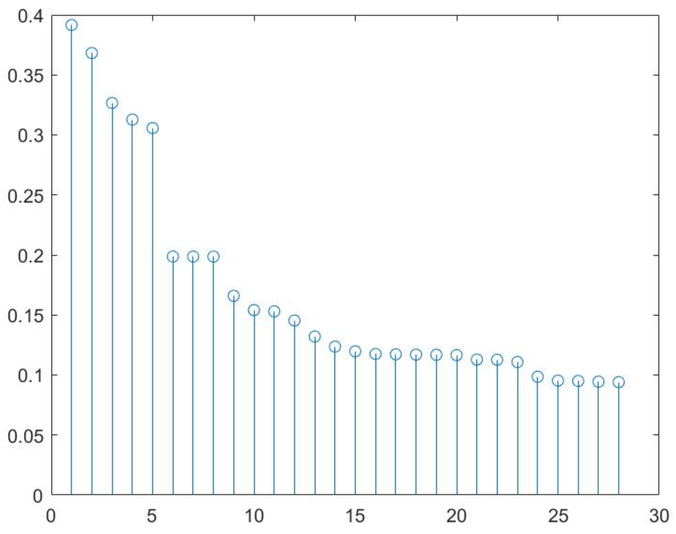

Figure 3.2: The rank of feasters

The gist of eigenvector centrality is to estimate the importance of a feature as a function of the importance of its neighbors. Ranking central nodes individuates candidate features, which turn out to be effective from a classification. By defining the threshold, the number of features selected 3.2.

The distance between each testing dataset and training dataset calculated according to Euclidean Distance. The minimum distance for testing dataset is determined by the label of the training dataset. In table 4.1, the result of the time series algorithm I are shown.

Source: Code/Matlab/MatrixPredict/MatrixPredict.m

	Class 0 (WB)			Class 1 (OK)			
Model	Precision	Recall	F1	Precision	Recall	F1	ROC
Similarity Algorithm	0.08	0.13	0.10	0.94	0.89	0.92	0.51

Table 3.1: Matrix Similarity Approach

4
Time Series

4.1 Introduction

It is a fundamental observation in science that a complex problem can be solved by partitioning the problem space into smaller components and analyzing these components individually. Solutions to these individual components are easier to find, and these solutions can be combined to provide a better solution or provide (in this case) a more precise estimate than can be obtained by a single model. Conventionally, single models were used for the prediction of time series, and although theoretically a single model can emulate any function, it is often very difficult to extract such an accurate single model from the data.

4.2 Proposed Algorithm

For weld break prediction, all 3170 datasets were randomly divided into training and testing phases for selecting the most precise model. Using 70% and 30% . In other words, 2219 and 951 datasets of the whole datasets were used as training and testing datasets, in order. It is a well-established fact that a predictor with high precision of WB. Also in this approach, feature selection via eigenvector centrality was used.

4.2.1 Algorithm I

For prediction, preprocessing of time series is required. As a first step, the all matrixes are converted into 1-D space. The time series are generated by using the same time delay parameter. The distance between each testing time series and training time series calculated according to Mahalanobis Distance. The minimum distance for testing time series is determined by the label of the training time series 4.1.

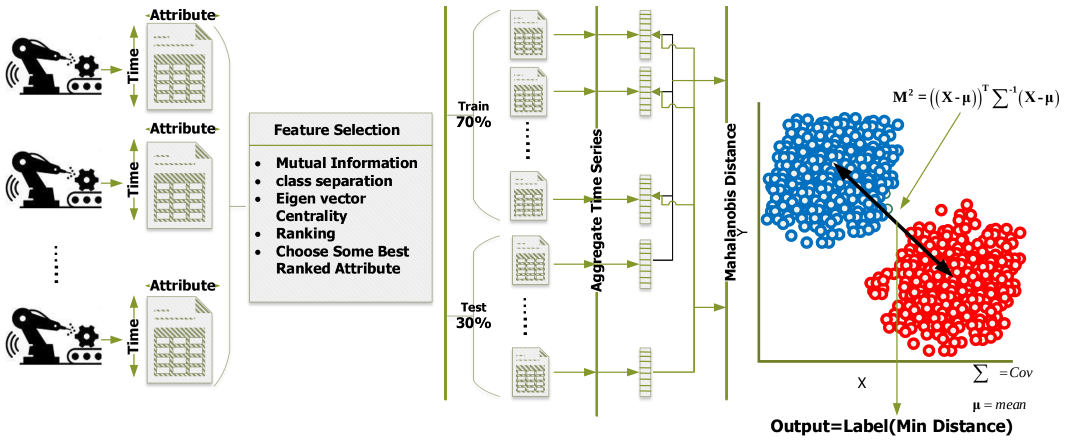

Figure 4.1: Time Series Agorithm I

In table 4.1, the result of the time series algorithm I are shown and have a good result for ROC.

Source: Code/Matlab/SeriesAgrregatePredict/SeriesAgrregatePredict.m

	Class 0 (WB)			Class 1 (OK)			
Model	Precision	Recall	F1	Precision	Recall	F1	ROC
Similarity Algorithm	0.19	0.39	0.26	0.95	0.88	0.91	0.63

Table 4.1: Time Series Agorithm I

4.2.2 Algorithm II

First, the each rows of each matrix of dataset are transformed to another matrix. In fact, dimensional changes are given. The number of new matrices is the number of time steps. Then, the distance between each rows of testing matrixes and each rows of matrix with the same number training matrix calculated according to Mahalanobis Distance. The minimum distance for testing rows is determined by the label of the training matrix number that matches the row according to 4.2.

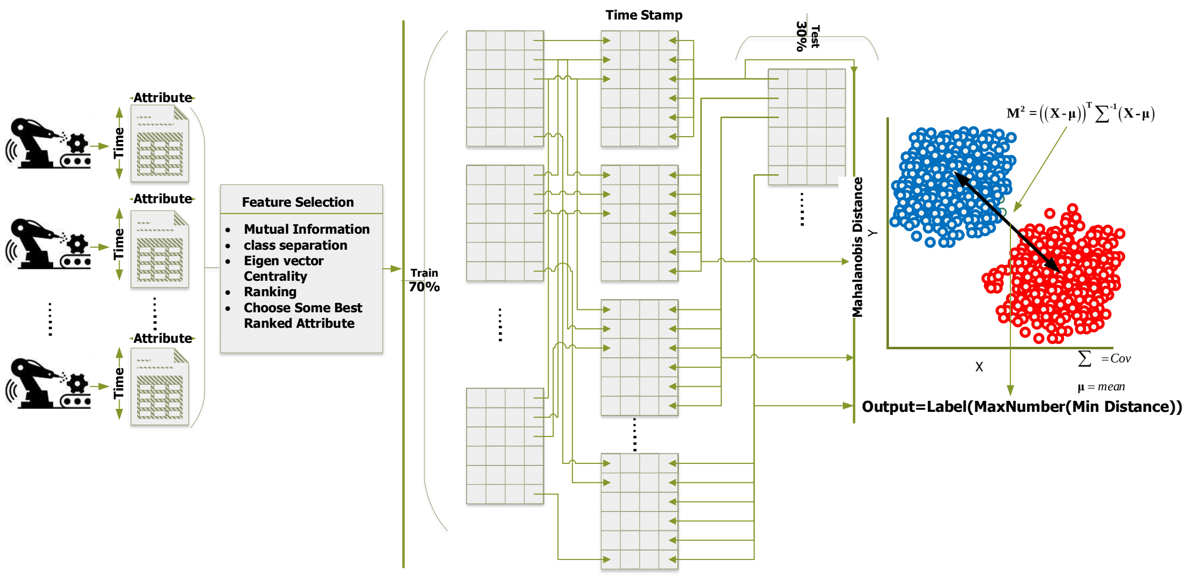

Figure 4.2: Time Series Agorithm II

The result of this algorithm is not desirable due to the low number of WB data. Source: Code/Matlab/SeriesrowsPredict/SeriesrowsPredict.m

5
Overall Conclusion

5.1 Introduction

This report was provided to answer the questions of the weld break prediction challenge.

5.2 Considered Questions

- Data exploration. E.g.: What did you learn/notice during exploring the data?

  According to data behavior in the figures of section2, there is not significantly different between the behavior of features in fail and correct products.

- Features. E.g.: What did you provide as input to your machine learning algorithm? Which features? How did you choose those features?

  Both of dynamic feature and static feature were considered and were described. In the first step, only-one feature classifier is used (trying one by one, with all the features). The feature that leads to a better classification performance is selected. The features are determined in each separate code. In each time you run, different features may be selected. Also in time series model, feature selection via eigenvector centrality was used. Because of the random selection of the training and testing data, the centrality of data was changing. After normalization, different features were selected.

- Learning algorithms. E.g.: Which machine learning algorithm(s) did you use and why? Did you have to do any optimization?

  All of the classification methods have been used in this report on the weld break data set. They needed to be optimized and one of the good one is Genetic Algorithm.

- Evaluation metric. E.g.: What evaluation metric(s) did you use to evaluate your machine learning model? Why did you use that metric?

  According to the problem, the precision for zero class is more important than others. In some methods same as Ada Boost Classifier + GA and K Nearest Neighbors + GA,the precision for zero class was 1. These methods showed the best performance, but the recall of these methods were low. The ROC was considered because those problem was not logical. ROC is a weighted average for precision and recall.

- Evaluation procedure. E.g.: Did you create a training and testing set? Did you do it randomly? According to time?

  Yes, for weld break prediction, all 3170 datasets were randomly divided into training and testing phases for selecting the most precise model using 70% and 30% . Both methods based on time and feature were evaluated.

- Results. Clearly display your results and provide an interpretation of the results.

  The best algorithm for the ROC result was Linear Discriminant Analysis with sampler ”Random Under Sampler”+GA. The result is shown in section2. The results indicated the superiority of the Linear Discriminant Analysis with sampler ”Random Under Sampler” models in predicting weld break in comparison to others. The ROC equal to 0.75 was obtained from this model using testing datasets. This shows a higher accuracy of the sampler-GA technique in predicting the weld break, while these values were obtained as 0.54 and 0.67 for Weighted Average Probabilities (Soft Voting) Ensemble Learning and Nearest Centroid Classifier Samplers Method, respectively. Eventually, it can be concluded that sampler GA, is a reliable and enforceable model in the field of fault detection issues in manufactures.