Fault Diagnosis in Steel Plates using Machine Learning

Submitted by

Void Programmers

Anirud Vem	A20199565
Maruthi Sankar Nanduri	A20224542
Monish Vallamkonda	A20213053
Nikhil Gunti	A20220906

Oklahoma State University

MSIS5600- Programming for Data Science (R and Python)

April 19th, 2020

Table of Contents

Executive Summary **1**
Statement of Scope **2**
Project Schedule **2**
GANTT Chart 3
Data Preparation **4**
Data Access 4
Data Consolidation 4
Data Cleaning 4
Data Transformation 4
Data Reduction 6
Data Dictionary 7
Exploratory Data Analysis **10**
Modeling Techniques **12**
Decision Tree 12
Random Forest 13
Neural Network 14
Data Splitting and Subsampling **15**
Training Data Set Statistics 16
Testing dataset statistics 17
Model Building **18**
Decision Tree 18
Random Forest 21
Neural Networks 23
Model Assessment: **26**
Strengths and Weakness of Models 27
Final Model and Conclusion 28
References **29**
Appendix **30**

Executive Summary

Quality Control and Fault Diagnosis is an important aspect to enhance the quality of production and optimize the cost of product testing. Timely identification of the fault saves input cost. An analysis into the factors responsible for faulty steel plates during the manufacturing process will enable the manufacturer to identify the issue in the plate before packaging.

The core of this analysis will involve pattern recognition and classification of faults in steel plates using Random Forest, Neural Networks and Multinomial Logistic Regression. The data for the analysis is collected from the images of the steel plates by Semeion, Research Centre of Sciences of Communication, Rome, Italy. The data is sourced from UCI Machine Learning repository.

The Classification of faults in plates starts with identifying the important factors for the prediction. This is achieved through Principal Component Analysis where important predictors are identified. Machine Learning Algorithms like Multinomial Logistic Regression, Neural nets and Random Forests are used to classify the issues and the best model is assessed by the comparison of model diagnostics. The Classification models are evaluated based on accuracy and precision and noticed that Random Forest Model has the best accuracy and precision and also the input variables are reduced using Principal Component Analysis to reduce the complexity.

Statement of Scope

This objective is achieved through accomplishing the below tasks:

- Perform Principal Component Analysis and identify the important predictor variables in identifying the faulty steel plates
- Build Neural Network and Multinomial Logistic Regression models to predict the fault in the steel plates.
- Perform a Comparative Analysis of different models and summarize the factors responsible for fault diagnosis in steel plates.

The scope of the analysis includes:

1. Exploration of Steel plate dimensions and specifications
2. Identification of skewed predictors and perform necessary transformations
3. Factor Analysis of the predictors
4. Evaluation of ML models to perform the fault diagnosis on the steel plates

During the project, data corresponding to 2000 steel plates, which included the specifications and dimensions are referenced from UCI Machine Learning repository for the analysis.

Project Schedule

The project was scheduled to be completed by of April 19th, within 14 weeks duration. The initial 3 weeks were spent on the onboarding, idea of the project and project proposal. The data is collected from the UCI ML repository. After performing the comprehensive exploratory analysis of all the features, necessary transformation is done on the data. This take a significant portion of time in the first half of the project. The second half of the project dealt with Model Selection, Model building and Model evaluation which went as per the below mentioned schedule.

GANTT Chart

Research Task	February					March				April				
	Jan 30th	6th	13th	20th	27th	5th	12th	15th	26th	2nd	9th	16th	19th	24th
1. Project Discussion														
2. Project Proposal														
3. Data Preparation														
4. Exploratory Analysis														
5. Factor Analysis														
6.Deliverable 1 Report														
7. Data Sampling														
8. Model Building														
9. Model Assessment														
10. Deliverable 2 Report														
11. Final Presentation														

Data Preparation

Data Access

The primary source of the data is UCI Machine Learning Repository. The data comes from research by Semeion, Research Centre of Sciences of Communication, Via Sersale 117, 00128, Rome, Italy. (www.semeion.it). The data consists of 1941 records with 27 attributes starting from the perimeter, Luminosity, Scratches, Stains, Dirt and other faults related to a steel plate.

Data Consolidation

The data has around 2000 rows which is already gathered from multiple images and sources and been consolidated into a single source, which has no scope for further data consolidation in this project.

Data Cleaning

The data is checked for missing values and due diligence on the data is performed where it is found that it does not have any missing value which makes the process void of data cleaning.

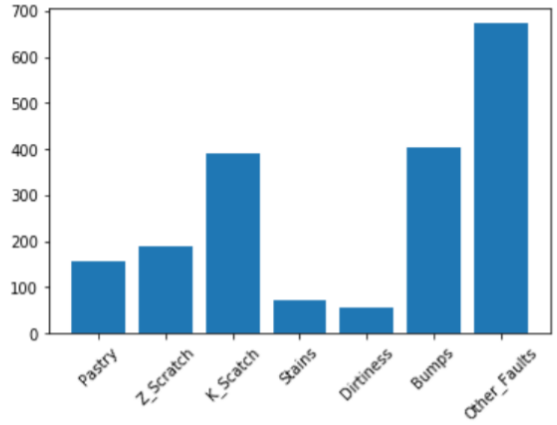
Data Transformation

In this step of data transformation, data is checked for normality as the target is to detect fault in steel plate. Univariate analysis is performed, and the distribution of each variable is examined. All the variables look good except for the 4 variables - Pixels_Areas, Sum_of_Luminosity, X_Perimeter & Y_Perimeter for which the distribution is highly skewed. Log transformation is applied to normalize these variables. Collected dataset has 7 classes of target(fault) such as Pastry, Z_Scratch, K_Scratch, Stains, Dirtness, Bumps and other_Faults which are one hot encoded value.

In order to reduce the dimensionality of these 7 variables, we merged these 7 attributes into one target variable and labelled all those 7 as (1,2,3…7).

```
In [119]: #distribution of the target variable
          plt.bar(x=target_index, height= target_values)
          plt.xticks(rotation=45)

Out[119]: ([0, 1, 2, 3, 4, 5, 6], <a list of 7 Text xticklabel objects>)
```

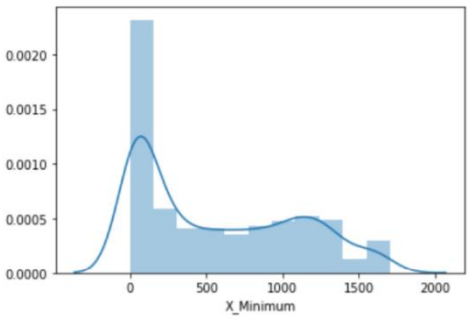



```
In [135]: df.Class.value_counts()

Out[135]: 7    673
          6    402
          3    391
          2    190
          1    158
          4     72
          5     55
          Name: Class, dtype: int64
```

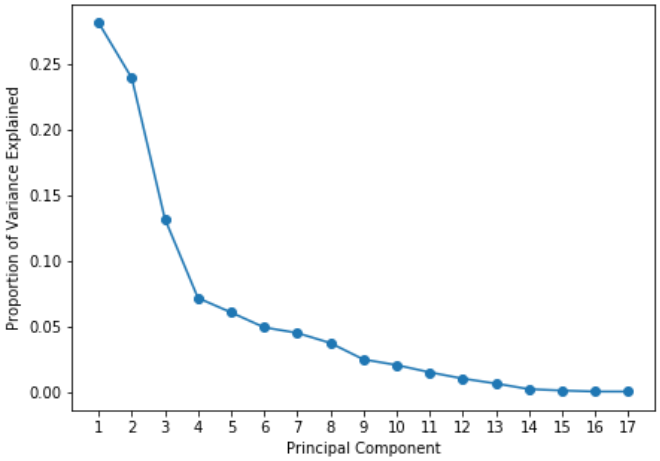
```
In [115]: sns.distplot(df_steel.X_Minimum )

Out[115]: <matplotlib.axes._subplots.AxesSubplot at 0x1e5fc69a188>
```



Data Reduction

We have 27 predictor variables in determining the target faulty class of steel plates. Variables with a high correlation are removed as they share common information among them and does not provide any addition information. Variables with a correlation greater than 0.8 are removed from the analysis. In order to determine the most important variables influencing the target, Principal Component Analysis (PCA) and Factor Analysis is performed. By looking at Scree Plot, elbow point is observed at 9th component which implies that number of components =9. This evidence is corroborated by factor analysis and eigen values variances. The most important variables obtained for modelling are 'LogOfAreas', 'Luminosity_Index', 'Edges_X_Index', 'TypeOfSteel_A300', 'Empty_Index', 'X_Minimum', 'Square_Index', 'Steel_Plate_Thickness', 'Edges_Y_Index'. We did not remove unnecessary columns from the original dataset but used only above-mentioned variables. All the observations were retained as there is no abnormalities in the rows of dataset.



Data Dictionary

S.No	Attribute Name	Data Type	Description	Source
1	X_Minimum	Integer	Minimum horizontal length of the plate from centre	https://www.kaggle.com/uciml/faulty-steel-plates
2	X_Maximum	Integer	Maximum horizontal length of the plate from centre	https://www.kaggle.com/uciml/faulty-steel-plates
3	Y_Minimum	Integer	Minimum vertical length of the plate from centre	https://www.kaggle.com/uciml/faulty-steel-plates
4	Y_Maximum	Integer	Maximum vertical length of the plate from center	https://www.kaggle.com/uciml/faulty-steel-plates
5	Pixels_Areas	Integer	Overall Pixel Area	https://www.kaggle.com/uciml/faulty-steel-plates
6	X_Perimeter	Integer	Horizontal Perimeter length of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
7	Y_Perimeter	Integer	Vertical Perimeter length of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
8	Sum_of_Luminosity	Integer	Sum of all rate at which the energy is emitted	https://www.kaggle.com/uciml/faulty-steel-plates
9	Minimum_of_Luminosity	Integer	Minimum rate at which the energy is emitted	https://www.kaggle.com/uciml/faulty-steel-plates
10	Maximum_of_Luminosity	Integer	Maximum rate at which the energy is emitted	https://www.kaggle.com/uciml/faulty-steel-plates
11	Length_of_Conveyer	Integer	The length of the conveyer belt	https://www.kaggle.com/uciml/faulty-steel-plates

12	TypeOfSteel_A300	Categorical	‘1’ if the type of steel is A300 else ‘0’	https://www.kaggle.com/uciml/faulty-steel-plates
13	TypeOfSteel_A400	Categorical	‘1’ if the type of steel is A400 else ‘0’	https://www.kaggle.com/uciml/faulty-steel-plates
14	Steel_Plate_Thickness	Integer	Thickness of the steel plate	https://www.kaggle.com/uciml/faulty-steel-plates
15	Edges_Index	Float	Measure of the index edge of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
16	Empty_Index	Float	Measure of the hollowness or emptiness of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
17	Square_Index	Float	Measure of Squareness of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
18	Outside_X_Index	Float	Measure of Outside horizontal Index	https://www.kaggle.com/uciml/faulty-steel-plates
19	Edges_X_Index	Float	Measure of horizonal edges of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
20	Edges_Y_Index	Float	Measure of vertical edges of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
21	Outside_Global_Index	Float	Measure of how much is the global index number	https://www.kaggle.com/uciml/faulty-steel-plates
22	LogOfAreas	Float	Measure of the logarithmic value of Area of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
23	Log_X_Index	Float	Logarithmic values of horizontal index of the plate	https://www.kaggle.com/uciml/faulty-steel-plates

24	Log_Y_Index	Float	Logarithmic values of vertical index of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
25	Orientation_Index	Float	Overall orientation measure of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
26	Luminosity_Index	Float	Measure of rate at which the energy is emitted	https://www.kaggle.com/uciml/faulty-steel-plates
27	SigmoidOfAreas	Float	The measure of Sigmoid function of the area of the plate	https://www.kaggle.com/uciml/faulty-steel-plates
28	Pastry	Categorical	'1' if the defect is Pastry else '0'	https://www.kaggle.com/uciml/faulty-steel-plates
29	Z_Scratch	Categorical	'1' if the defect is Z_Scratch else '0'	https://www.kaggle.com/uciml/faulty-steel-plates
30	K_Scatch	Categorical	'1' if the defect is K_Scatch else '0'	https://www.kaggle.com/uciml/faulty-steel-plates
31	Stains	Categorical	'1' if the defect is Stains else '0'	https://www.kaggle.com/uciml/faulty-steel-plates
32	Dirtiness	Categorical	'1' if the defect is Dirtiness else '0'	https://www.kaggle.com/uciml/faulty-steel-plates
33	Bumps	Categorical	'1' if the defect is Bumps else '0'	https://www.kaggle.com/uciml/faulty-steel-plates
34	Other_Faults	Categorical	'1' if the defect is Other_Faults else '0'	https://www.kaggle.com/uciml/faulty-steel-plates

Exploratory Data Analysis

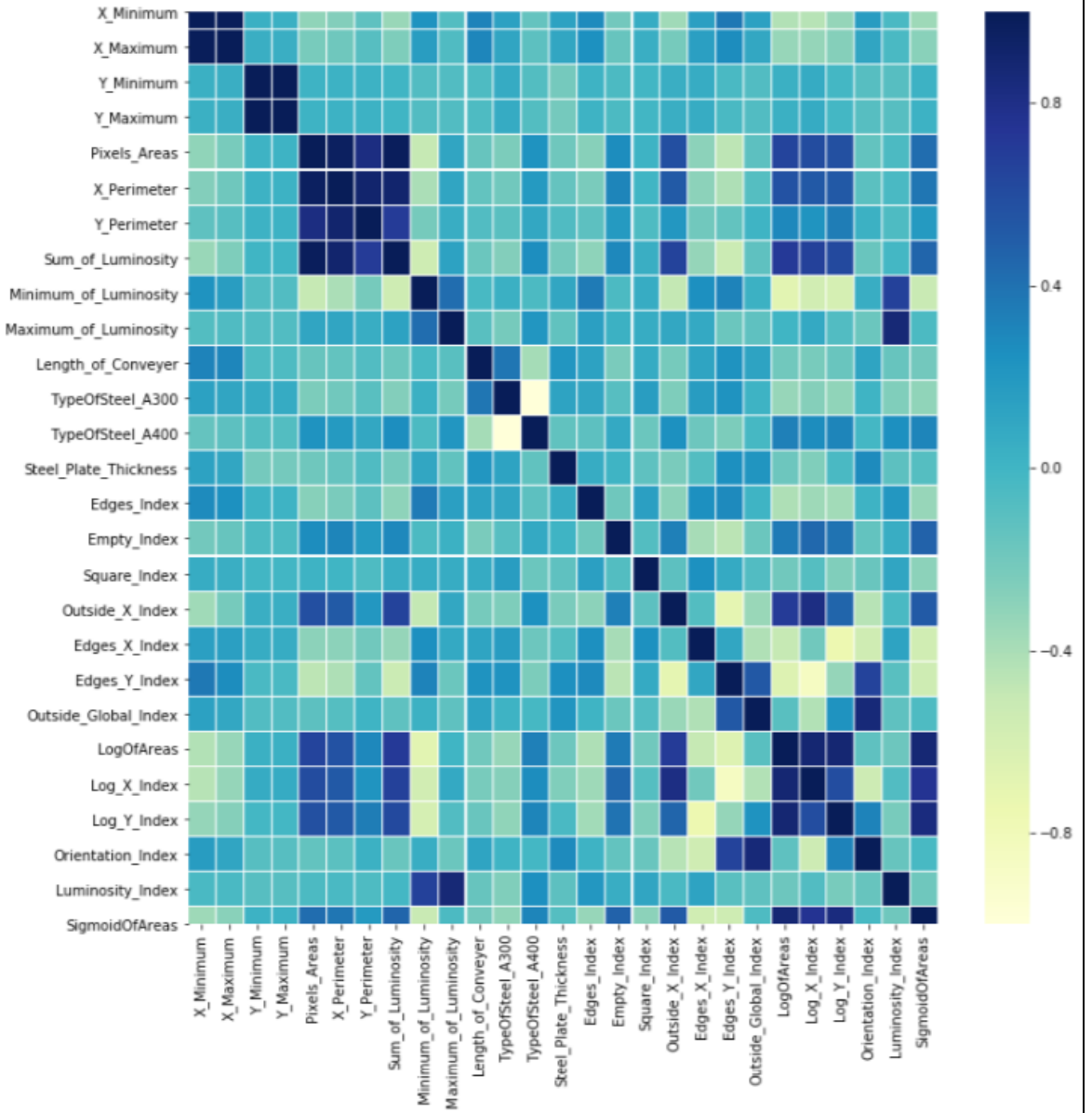
Univariate Analysis

The variables are mix of continuous and categorical data types. Below is the chart of the basic statistics of all the variables in the data including the target variables. Statistics are the total count, mean of the variable, standard deviation, minimum and maximum values of the variables, median and 1st and 3rd quartiles of the all the variables in the data.

	count	mean	std	min	25%	50%	75%	max
X_Minimum	1941.0	5.711360e+02	5.206907e+02	0.0000	51.0000	4.350000e+02	1.053000e+03	1.705000e+03
X_Maximum	1941.0	6.179645e+02	4.976274e+02	4.0000	192.0000	4.670000e+02	1.072000e+03	1.713000e+03
Y_Minimum	1941.0	1.650685e+06	1.774578e+06	6712.0000	471253.0000	1.204128e+06	2.183073e+06	1.298766e+07
Y_Maximum	1941.0	1.650739e+06	1.774590e+06	6724.0000	471281.0000	1.204136e+06	2.183084e+06	1.298769e+07
Pixels_Areas	1941.0	1.893878e+03	5.168460e+03	2.0000	84.0000	1.740000e+02	8.220000e+02	1.526550e+05
X_Perimeter	1941.0	1.118552e+02	3.012092e+02	2.0000	15.0000	2.600000e+01	8.400000e+01	1.044900e+04
Y_Perimeter	1941.0	8.296600e+01	4.264829e+02	1.0000	13.0000	2.500000e+01	8.300000e+01	1.815200e+04
Sum_of_Luminosity	1941.0	2.063121e+05	5.122936e+05	250.0000	9522.0000	1.920200e+04	8.301100e+04	1.159141e+07
Minimum_of_Luminosity	1941.0	8.454869e+01	3.213428e+01	0.0000	63.0000	9.000000e+01	1.060000e+02	2.030000e+02
Maximum_of_Luminosity	1941.0	1.301937e+02	1.869099e+01	37.0000	124.0000	1.270000e+02	1.400000e+02	2.530000e+02
Length_of_Conveyer	1941.0	1.459160e+03	1.445778e+02	1227.0000	1358.0000	1.364000e+03	1.650000e+03	1.794000e+03
TypeOfSteel_A300	1941.0	4.003091e-01	4.900872e-01	0.0000	0.0000	0.000000e+00	1.000000e+00	1.000000e+00
TypeOfSteel_A400	1941.0	5.996909e-01	4.900872e-01	0.0000	0.0000	1.000000e+00	1.000000e+00	1.000000e+00
Steel_Plate_Thickness	1941.0	7.873776e+01	5.508603e+01	40.0000	40.0000	7.000000e+01	8.000000e+01	3.000000e+02
Edges_Index	1941.0	3.317152e-01	2.997117e-01	0.0000	0.0604	2.273000e-01	5.738000e-01	9.952000e-01
Empty_Index	1941.0	4.142033e-01	1.372615e-01	0.0000	0.3158	4.121000e-01	5.016000e-01	9.439000e-01
Square_Index	1941.0	5.707671e-01	2.710584e-01	0.0083	0.3613	5.556000e-01	8.182000e-01	1.000000e+00
Outside_X_Index	1941.0	3.336110e-02	5.896117e-02	0.0015	0.0066	1.010000e-02	2.350000e-02	8.759000e-01
Edges_X_Index	1941.0	6.105286e-01	2.432769e-01	0.0144	0.4118	6.364000e-01	8.000000e-01	1.000000e+00
Edges_Y_Index	1941.0	8.134722e-01	2.342736e-01	0.0484	0.5968	9.474000e-01	1.000000e+00	1.000000e+00
Outside_Global_Index	1941.0	5.757342e-01	4.823520e-01	0.0000	0.0000	1.000000e+00	1.000000e+00	1.000000e+00
LogOfAreas	1941.0	2.492388e+00	7.889299e-01	0.3010	1.9243	2.240600e+00	2.914900e+00	5.183700e+00
Log_X_Index	1941.0	1.335686e+00	4.816116e-01	0.3010	1.0000	1.176100e+00	1.518500e+00	3.074100e+00
Log_Y_Index	1941.0	1.403271e+00	4.543452e-01	0.0000	1.0792	1.322200e+00	1.732400e+00	4.258700e+00
Orientation_Index	1941.0	8.328764e-02	5.008680e-01	-0.9910	-0.3333	9.520000e-02	5.116000e-01	9.917000e-01
Luminosity_Index	1941.0	-1.313050e-01	1.487668e-01	-0.9989	-0.1950	-1.330000e-01	-6.660000e-02	6.421000e-01
SigmoidOfAreas	1941.0	5.854205e-01	3.394518e-01	0.1190	0.2482	5.063000e-01	9.998000e-01	1.000000e+00
Pastry	1941.0	8.140134e-02	2.735209e-01	0.0000	0.0000	0.000000e+00	0.000000e+00	1.000000e+00
Z_Scratch	1941.0	9.788769e-02	2.972393e-01	0.0000	0.0000	0.000000e+00	0.000000e+00	1.000000e+00
K_Scatch	1941.0	2.014426e-01	4.011812e-01	0.0000	0.0000	0.000000e+00	0.000000e+00	1.000000e+00
Stains	1941.0	3.709428e-02	1.890415e-01	0.0000	0.0000	0.000000e+00	0.000000e+00	1.000000e+00
Dirtiness	1941.0	2.833591e-02	1.659734e-01	0.0000	0.0000	0.000000e+00	0.000000e+00	1.000000e+00
Bumps	1941.0	2.071097e-01	4.053393e-01	0.0000	0.0000	0.000000e+00	0.000000e+00	1.000000e+00
Other_Faults	1941.0	3.467285e-01	4.760510e-01	0.0000	0.0000	0.000000e+00	1.000000e+00	1.000000e+00

Correlation Plot:

The below plot describes the correlation between all the predictors with the target variable and also among the predictor variables.



Modeling Techniques

Decision Tree

The goal of this model is to classify the defect of the steel plate out of all the 7 defects using the independent variables in the data. The defect can be either Pastry, Z_Scratch, K_Scratch, Stains, Dirtness, Bumps and Other_Faults. Predicting this defect is the main objective and is the target variable. And one more goal using Decision tree is to figure out the important variables that have high impact on predicting the target variable that is type of defect.

Initially, training data set is fed into decision tree classifier algorithm as it uses estimates and probabilities to calculate likely outcomes. Decision tree uses a tree structure with a set of "if-then" rules to classify data points. The rules are learned sequentially from the training data. The tree is constructed top-down; attributes at the top of the tree have a larger impact on the classification decision. The training process continues until it meets a termination condition. It can model complex decision processes, very intuitive interpretation of results.

Assumptions

- At the beginning, the whole training set is considered as the **root.**
- Feature values are preferred to be categorical. Our target variable is a categorical which is favorable in choosing this algorithm.
- Records are **distributed recursively** based on attribute values.
- Order to placing attributes as root or internal node of the tree is done by using some statistical approach.
- Information gain in a decision tree with categorical variables gives a biased response for attributes with greater no. of categories.

Generally, decision trees are extremely sensitive to small perturbations in the data: a slight change can result in a drastically different tree. They can easily overfit. This can be negated by validation methods and pruning, but this is a grey area. Some of these are related to the problem of multicollinearity: when two variables both explain the same thing, a decision tree will greedily choose the best one, whereas many other methods will use them both. Ensemble methods such as random forests can negate this to a certain extent, but you lose the ease of understanding.

Random Forest

The goal of this model is to classify the defect of the steel plate out of all the 7 defects using the independent variables in the data. The defect can be either Pastry, Z_Scratch, K_Scratch, Stains, Dirtness, Bumps and Other_Faults. Predicting this defect is the main objective which is the target variable for this model. A more advanced version of the decision tree, which addresses overfitting by growing many trees with random variations, then selecting and aggregating the best-performing decision trees. The "forest" is an ensemble of decision trees, typically done using a technique called "bagging". This this model uses two key concepts that gives it the name random:

1. Random sampling of training data points when building trees
2. Random subsets of features considered when splitting nodes

By training each tree on different samples, although each tree might have high variance with respect to a set of the training data, overall, the entire forest will have lower variance but not at the cost of increasing the bias. And one more goal is to figure out the important variables that have high impact on predicting the target variable that is type of defect.

Assumptions

- At each step of building individual tree we find the **best split** of data

- While building a tree we use not the whole dataset, but **bootstrap sample**
- We aggregate the individual tree outputs by **averaging** (actually, 2 and 3 means together more general bagging procedure).
- There is no need of having a relation between target variable and independent variables.

Neural Network

The main goal of this model is to classify the defect of the steel plate out of all the 7 defects using the independent variables in the data. The defect can be either Pastry, Z_Scratch, K_Scratch, Stains, Dirtness, Bumps and Other_Faults. Predicting this defect is the main objective which is the target variable for this model. Artificial neural networks are built of simple elements called neurons, which take in a real value, multiply it by a weight, and run it through a non-linear activation function. By constructing multiple layers of neurons, each of which receives part of the input variables, and then passes on its results to the next layers, the network can learn very complex functions. Since we have 26 variables which induces complexity among them, Neural Networks works well in this case as it is very effective for high dimensionality problems, able to deal with complex relations between variables, non-exhaustive category sets and complex functions relating input to output variables. Powerful tuning options to prevent over- and under-fitting.

Assumptions

Data preprocessing needs to be done before execution of this model. Like missing values needs to be imputed, data transformation needs to be done. All necessary data preprocessing is done already. Neural Network is a black box model and interpretation of the results of Neural Network is difficult. We assume the output of the target variable is true.

Data Splitting and Subsampling

For cross validation purpose and to prevent data from over fitting data splitting is done which is dividing the data into two parts which is into training and testing data sets. Training data is used to build the model and determine the important variables in the data and predict the target variable. Testing data is used to validate the results across different models. Initial dataset contains 1941 observations and 34 columns.

Data has been split into 70% training data and 30% testing data on the target variable. Training data is used for the model to learn the patterns from the data and develop the model rules. Testing data is used to test the model on unseen data. Training data has 1358 observations and testing dataset has 583 observations with 27 attributes. The idea is that more training data is a good thing because it makes the classification model better while more test data makes the error estimate more accurate. Reason for choosing 70-30 split instead of 50-50 is because, higher percent for training data will ensure more trends and patterns can be identified and will help in predicting the target variable with less error. The below is the distribution of target variables for both the training data set and the testing data set.

```
y_train.value_counts()
```

```
Other_Faults    462
Bumps           287
K_Scatch        268
Z_Scratch       132
Pastry          120
Stains           49
Dirtiness        40
Name: DefType, dtype: int64
```

```
y_test.value_counts()
```

```
Other_Faults    211
K_Scatch        123
Bumps           115
Z_Scratch        58
Pastry           38
Stains           23
Dirtiness        15
Name: DefType, dtype: int64
```

Fig: Distribution of Target variable for training and testing data

Descriptive Statistics of the training dataset and testing dataset are in line with the actual dataset.

The below is the statistics, of the training data and the testing data.

Training Data Set Statistics

	count	mean	std	min	25%
X_Minimum	1358.0	5.726635e+02	5.244790e+02	0.0000	48.000000
X_Maximum	1358.0	6.183660e+02	5.016641e+02	4.0000	191.250000
Y_Minimum	1358.0	1.678222e+06	1.736243e+06	6712.0000	492998.250000
Y_Maximum	1358.0	1.678265e+06	1.736242e+06	6724.0000	493105.750000
Pixels_Areas	1358.0	1.781730e+03	3.851907e+03	2.0000	82.000000
X_Perimeter	1358.0	1.036701e+02	1.836486e+02	2.0000	14.000000
Y_Perimeter	1358.0	7.181885e+01	1.142033e+02	1.0000	13.000000
Sum_of_Luminosity	1358.0	1.963177e+05	4.395054e+05	250.0000	9210.250000
Minimum_of_Luminosity	1358.0	8.456554e+01	3.216052e+01	0.0000	63.000000
Maximum_of_Luminosity	1358.0	1.299514e+02	1.837353e+01	37.0000	124.000000
Length_of_Conveyer	1358.0	1.459236e+03	1.446490e+02	1227.0000	1358.000000
TypeOfSteel_A300	1358.0	4.057437e-01	4.912163e-01	0.0000	0.000000
TypeOfSteel_A400	1358.0	5.942563e-01	4.912163e-01	0.0000	0.000000
Steel_Plate_Thickness	1358.0	7.848012e+01	5.430125e+01	40.0000	40.000000
Edges_Index	1358.0	3.287971e-01	3.003263e-01	0.0000	0.060400
Empty_Index	1358.0	4.131761e-01	1.333801e-01	0.0000	0.321400
Square_Index	1358.0	5.703882e-01	2.707642e-01	0.0090	0.360875
Outside_X_Index	1358.0	3.256870e-02	5.522742e-02	0.0015	0.006600
Edges_X_Index	1358.0	6.157730e-01	2.418829e-01	0.0657	0.428600
Edges_Y_Index	1358.0	8.142315e-01	2.341229e-01	0.0484	0.597175
Outside_Global_Index	1358.0	5.636966e-01	4.842701e-01	0.0000	0.000000
LogOfAreas	1358.0	2.476850e+00	7.853281e-01	0.3010	1.913800
Log_X_Index	1358.0	1.328766e+00	4.798089e-01	0.3010	1.000000
Log_Y_Index	1358.0	1.392601e+00	4.474672e-01	0.0000	1.079200

Testing dataset statistics

	count	mean	std	min	25%
X_Minimum	583.0	5.675780e+02	5.121860e+02	0.0000	67.50000
X_Maximum	583.0	6.170292e+02	4.885201e+02	8.0000	194.50000
Y_Minimum	583.0	1.586543e+06	1.860767e+06	7003.0000	410706.00000
Y_Maximum	583.0	1.586621e+06	1.860808e+06	7020.0000	410830.00000
Pixels_Areas	583.0	2.155108e+03	7.372309e+03	6.0000	90.50000
X_Perimeter	583.0	1.309211e+02	4.725091e+02	4.0000	16.00000
Y_Perimeter	583.0	1.089314e+02	7.582328e+02	2.0000	14.00000
Sum_of_Luminosity	583.0	2.295924e+05	6.508866e+05	775.0000	10691.00000
Minimum_of_Luminosity	583.0	8.450943e+01	3.210062e+01	0.0000	63.50000
Maximum_of_Luminosity	583.0	1.307581e+02	1.941476e+01	84.0000	124.00000
Length_of_Conveyer	583.0	1.458985e+03	1.445360e+02	1227.0000	1358.00000
TypeOfSteel_A300	583.0	3.876501e-01	4.876324e-01	0.0000	0.00000
TypeOfSteel_A400	583.0	6.123499e-01	4.876324e-01	0.0000	0.00000
Steel_Plate_Thickness	583.0	7.933791e+01	5.691509e+01	40.0000	40.00000
Edges_Index	583.0	3.385123e-01	2.984216e-01	0.0000	0.06040
Empty_Index	583.0	4.165962e-01	1.459938e-01	0.0278	0.30270
Square_Index	583.0	5.716497e-01	2.719732e-01	0.0083	0.36335
Outside_X_Index	583.0	3.520686e-02	6.687004e-02	0.0022	0.00660
Edges_X_Index	583.0	5.983127e-01	2.462695e-01	0.0144	0.39340
Edges_Y_Index	583.0	8.117036e-01	2.348160e-01	0.1123	0.59355
Outside_Global_Index	583.0	6.037736e-01	4.770897e-01	0.0000	0.00000
LogOfAreas	583.0	2.528583e+00	7.967589e-01	0.7782	1.95660
Log_X_Index	583.0	1.351806e+00	4.858160e-01	0.4771	1.00000
Log_Y_Index	583.0	1.428127e+00	4.694271e-01	0.3010	1.07920

For both training and testing data sets the basic statistics like mean, standard deviation, median is almost same.

Model Building

Decision Tree

The data set have 27 variables which was eventually reduced to 9 variables by performing Principal Component Analysis. It is observed that 9 predictor variables significantly contribute to the classification of defects in the steel plates. Two decision tree models are built – One with all the predictor variables while the other with 9 predictor variables which are obtained after PCA. The model which utilizes all the predictor variables will be referred to as Model 1 while the model which uses only 9 variables out of the PCA for classifying the defects will be referred to as Model 2. The DT models split the data based on rules that minimize the entropy to the subsequent levels and efficiently classify the defects - Pastry, Z_Scratch, K_Scratch, Stains, Dirtiness, Bumps, Other_Faults. Root node is where all the observations are present, and tree is subdivided into pure subsets of respective defective classes which are called as sub-nodes and leaves.

Results and Interpretation of Decision Tree

In this case Decision tree has used Gini Index Metrix as classification criteria to split the node. At each node Gini Metrix is calculated for each variable at each observation of the variable. Value of the variable at which data point has the highest Gini is used to split the node. This procedure is followed at each node is developing the decision trees to form pure samples of defective class. For instance, the below snippet explains the splitting rule for the root node of the model with all the predictor variables. The parent node is split on the Log_X_Index variable with a gini of 0.781.

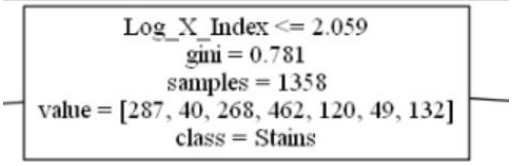


Fig: First Root Node in the Decision Tree

Important features that help in classifying the defects into the respective classes in both the models are listed below. Weight provides the importance of each feature. Type of steel that gives the information about the material of the product, Steel plate thickness, length of conveyer, pixel areas which gives key information about the steel plate are identified as important predictors in Model 1 by the DT algorithm. In the second model Areas, Luminosity_Index, edges index and Type of Steel are identified as the key features in Model 2.

Model 1

Weight	Feature
0.1883	Log_X_Index
0.0763	Length_of_Conveyer
0.0675	Steel_Plate_Thickness
0.0578	Maximum_of_Luminosity
0.0542	TypeOfSteel_A400
0.0532	Pixels_Areas
0.0501	Square_Index
0.0422	Y_Minimum
0.0402	Edges_Index
0.0389	Edges_Y_Index
0.0327	Edges_X_Index
0.0313	Y_Maximum
0.0313	X_Maximum
0.0299	Log_Y_Index
0.0298	X_Minimum
0.0265	Minimum_of_Luminosity
0.0263	TypeOfSteel_A300
0.0231	Sum_of_Luminosity
0.0207	Empty_Index
0.0202	X_Perimeter
0.0142	Orientation_Index
0.0101	Y_Perimeter
0.0090	Outside_X_Index

Model 2

Weight	Feature
0.2995	LogOfAreas
0.1495	X_Minimum
0.1192	Steel_Plate_Thickness
0.0910	Edges_X_Index
0.0883	Square_Index
0.0841	Luminosity_Index
0.0796	Empty_Index
0.0541	Edges_Y_Index
0.0348	TypeOfSteel_A300

A confusion matrix is plotted on the predicted defects given by the model vs the actual defect type from the data for either models. Model has performed well in categorizing all observations into their respective defective classes. In the confusion matrix the number on the diagonal row represent

the number of correctly classified categories and the off-diagonal numbers represents the misclassified classes.

Model1

```
[[ 59   0   2  39  12   0   3]
 [  1  10   0   3   1   0   0]
 [  0   0 111  11   0   0   1]
 [ 26   5  15 130  25   1   9]
 [  6   0   0  12  20   0   0]
 [  0   0   0   1   0  22   0]
 [  1   0   0   6   1   0  50]]
```

Model 2

```
[[ 62   0   0  39   7   0   2]
 [  0   9   0   5   1   0   0]
 [  2   0 105  13   0   0   0]
 [ 46   3   3 137  13   2  11]
 [ 12   2   0   8  15   0   0]
 [  1   0   0   1   0  21   0]
 [  3   1   1  10   1   0  47]]
```

Fig: Confusion Matrix for the Target Variables

Decision tree models with all predictors and limited predictors classified the defects on the test data sample and the model assessment metrics are compared. In this case the True positives are important to analyze as the False Positives are costly if the defects are wrongly classified or not classified. So, the model with full variables has a 70% precision while the one with limited independent variables has a precision of 69%. The below snippet also consists of precision, recall, f1-score, and accuracy for all the individual classes in the target variable - defect. Stains has the highest precision, recall and the f1-score while Pastry has the least accuracy, precision, and f1-score for the first model. K_Scatch has the highest precision, recall and the f1-score while Pastry has the least accuracy, precision, and f1-score for the second model. The first model which has all the input variables has superior assessment metric numbers when compared to that of the model with reduced variables.

Model with All Predictors

	precision	recall	f1-score	support
Bumps	0.63	0.53	0.58	115
Dirtiness	0.71	0.67	0.69	15
K_Scatch	0.88	0.90	0.89	123
Other_Faults	0.65	0.62	0.63	211
Pastry	0.33	0.50	0.40	38
Stains	0.96	0.96	0.96	23
Z_Scratch	0.80	0.90	0.85	58
avg / total	0.70	0.69	0.70	583

Model with 9 Predictors

	precision	recall	f1-score	support
Bumps	0.49	0.56	0.53	110
Dirtiness	0.60	0.60	0.60	15
K_Scatch	0.96	0.88	0.92	120
Other_Faults	0.64	0.64	0.64	215
Pastry	0.41	0.41	0.41	37
Stains	0.91	0.91	0.91	23
Z_Scratch	0.78	0.75	0.76	63
accuracy			0.68	583
macro avg	0.69	0.68	0.68	583
weighted avg	0.69	0.68	0.68	583

Fig: Model Assessment Metrics

Random Forest

Random forest model is an ensemble of the decision tree model that is used to improve the accuracy reduce the overfitting. A Random forest builds multiple models with the same target variable and different predictor variables and an average or mode of the target prediction is considered as final value. The Random Forest Model is also built in two scenarios mentioned above and both the models are compared.

The below figure enlists the important features that help in classifying the defects into the respective classes in both the models. Weight gives the importance of each feature. Pixel_Areas, Areas, Orientation_Index, Length of the conveyer are some of the important features in model 1 while Areas, X_Minimum, Steel Plate Thickness and Luminosity_Index are some of the important features in model 2 which has the reduced variables.

Model 1

Weight	Feature
0.0846 ± 0.1850	Pixels_Areas
0.0698 ± 0.0473	Length_of_Conveyer
0.0651 ± 0.1234	X_Perimeter
0.0575 ± 0.1014	Steel_Plate_Thickness
0.0540 ± 0.1499	Log_X_Index
0.0487 ± 0.0827	Minimum_of_Luminosity
0.0481 ± 0.0589	X_Minimum
0.0433 ± 0.0295	X_Maximum
0.0412 ± 0.0726	Edges_Y_Index
0.0400 ± 0.0284	Square_Index
0.0398 ± 0.0147	Y_Minimum
0.0373 ± 0.0412	Sum_of_Luminosity
0.0370 ± 0.0200	Luminosity_Index
0.0358 ± 0.0276	Orientation_Index
0.0339 ± 0.0261	Y_Maximum
0.0320 ± 0.0242	Edges_Index
0.0298 ± 0.0157	Empty_Index
0.0257 ± 0.0264	SigmoidOfAreas
0.0250 ± 0.0149	Maximum_of_Luminosity
0.0249 ± 0.0386	LogOfAreas
0.0243 ± 0.0250	Outside_X_Index
0.0219 ± 0.0321	TypeOfSteel_A300
0.0218 ± 0.0247	Edges_X_Index
0.0194 ± 0.0190	Y_Perimeter
0.0178 ± 0.0083	Log_Y_Index
0.0175 ± 0.0262	TypeOfSteel_A400
0.0037 ± 0.0042	Outside_Global_Index

Model 2

Weight	Feature
0.1944 ± 0.1533	LogOfAreas
0.1611 ± 0.1101	X_Minimum
0.1273 ± 0.1000	Steel_Plate_Thickness
0.1103 ± 0.0386	Luminosity_Index
0.1071 ± 0.0313	Square_Index
0.0933 ± 0.0390	Empty_Index
0.0855 ± 0.0840	Edges_Y_Index
0.0819 ± 0.0243	Edges_X_Index
0.0392 ± 0.0203	TypeOfSteel_A300

Fig: Features with their variable weights

A confusion matrix is obtained with the predicted defects by the model vs the actual defects type from the data in the. Model has performed well in categorizing all observations into their respective defective classes. In the confusion matrix the number on the diagonal row represent the number of correctly classified categories and the off-diagonal numbers represents the misclassified classes.

Model 1

```
[[ 75   0   1  33   6   0   0]
 [  0  13   0   1   1   0   0]
 [  1   0 113   9   0   0   0]
 [ 45   3   3 144  12   0   4]
 [  8   0   0  11  19   0   0]
 [  0   0   0   2   0  21   0]
 [  0   0   1  10   0   0  47]]
```

Model 2

```
[[ 68   1   1  35   4   0   1]
 [  0   8   0   5   2   0   0]
 [  0   0 115   5   0   0   0]
 [ 43   3   6 153   5   0   5]
 [ 10   0   0  13  14   0   0]
 [  1   0   0   1   0  21   0]
 [  5   1   1   9   2   0  45]]
```

Fig: Confusion Matrix for the Target Variables

In the current problem precision is the key metric to compare the models. Model 1 has 75% precision while model 2 has a precision of 73%. There are other assessment metrics like recall, f1-score, and support for all the individual classes in the target variable defect on the random forest model.

Stains has the highest precision, recall and the f1-score while Bumps has the least accuracy, precision, and f1-score for model 1 and model 2. Model 1 has better numbers in the metrics when compared to those of the model2.

Model 1

	precision	recall	f1-score	support
Bumps	0.58	0.65	0.61	115
Dirtiness	0.81	0.87	0.84	15
K_Scatch	0.96	0.92	0.94	123
Other_Faults	0.69	0.68	0.68	211
Pastry	0.50	0.50	0.50	38
Stains	1.00	0.91	0.95	23
Z_Scratch	0.92	0.81	0.86	58
accuracy			0.74	583
macro avg	0.78	0.76	0.77	583
weighted avg	0.75	0.74	0.74	583

Model 2

	precision	recall	f1-score	support
Bumps	0.54	0.62	0.57	110
Dirtiness	0.62	0.53	0.57	15
K_Scatch	0.93	0.96	0.95	120
Other_Faults	0.69	0.71	0.70	215
Pastry	0.52	0.38	0.44	37
Stains	1.00	0.91	0.95	23
Z_Scratch	0.88	0.71	0.79	63
accuracy			0.73	583
macro avg	0.74	0.69	0.71	583
weighted avg	0.73	0.73	0.73	583

Fig: Model Assessment Metrics

Neural Networks

Neural Network consists of the layers of neurons that process the input variables by applying appropriate activation functions and attach weights to the inputs to classify the target variable.

Model accuracy and complexity depends on the hyperparameters, number of hidden layers, number of neurons and the activation function used in each neuron. Two neural network models were developed, One has three hidden layers and 50 neurons in each, and another one is an auto neural network model in which the model determines the optimum number of layers and neurons. Activation function 'RELU' is used to activate the neurons. Neural network provides better accuracy than the other models, but the model is prone to very less interpretability due to the complex design. When the business importance is only to classify the defect class and not in identifying the feature importance and contribution of each feature, neural networks provides the best results. But when the business needs to find out the feature importance and process at the cost of small accuracy, decision trees or random forest would be a better approach. Two types of Neural Network models are built on Model 1 and Model 2.

The confusion matrix is developed on the predicted values by the model vs the actual defect type from the data. Model has performed well in categorizing all observations into their respective defective classes. In the confusion matrix the number on the diagonal row represent the number of correctly classified categories and the off-diagonal numbers represents the misclassified classes.

Neural Network

Model 1

```
[[ 67   0   1  40   4   0   3]
 [  1   3   0   5   6   0   0]
 [  0   0 114   9   0   0   0]
 [ 44   0   5 140  16   2   4]
 [  6   0   0  10  20   0   2]
 [  1   0   0   3   0  19   0]
 [  2   0   0   5   0   0  51]]
```

Model 2

```
[[ 51   0   0  43   3   0  13]
 [  0   0   0  10   5   0   0]
 [  0   0 111   9   0   0   0]
 [ 33   0   4 154  11   1  12]
 [  4   0   0  16  14   0   3]
 [  1   0   0   7   0  15   0]
 [  7   0   1   5   0   0  50]]
```

Auto Neural Network:

Model 1

```
[[ 67   0   1  40   4   0   3]
 [  1   3   0   5   6   0   0]
 [  0   0 114   9   0   0   0]
 [ 44   0   5 140  16   2   4]
 [  6   0   0  10  20   0   2]
 [  1   0   0   3   0  19   0]
 [  2   0   0   5   0   0  51]]
```

Model 2

```
[[ 54   0   0  42   2   0  12]
 [  0   0   0   8   7   0   0]
 [  0   0 114   6   0   0   0]
 [ 31   0   8 154  13   0   9]
 [  4   0   0  13  18   0   2]
 [  4   0   0  10   0   9   0]
 [  6   0   1   9   0   0  47]]
```

The business problem encourages to observe the precision as the False Positives are costlier in this scenario. So, both the Neural Network with custom parameters and Auto Neural Network with the system defined parameters are built and compared on the precision and other assessment metrics. The neural network with customised hyperparameters has a precision of 74% and 69% respectively for model 1 and model 2. As defined earlier model 1 represents the classification model with all the independent variables while model 2 represents the model with reduced variables after PCA. Stains are classified with highest precision, recall, f1-score, and support while Bumps are classified with the least for model 1. K_Scratch has the highest numbers in the assessment metrics while Dirtiness has the least for model 2.

Neural Network

Model 1

	precision	recall	f1-score	support
Bumps	0.55	0.58	0.57	115
Dirtiness	1.00	0.20	0.33	15
K_Scatch	0.95	0.93	0.94	123
Other_Faults	0.66	0.66	0.66	211
Pastry	0.43	0.53	0.48	38
Stains	0.90	0.83	0.86	23
Z_Scratch	0.85	0.88	0.86	58
accuracy			0.71	583
macro avg	0.76	0.66	0.67	583
weighted avg	0.72	0.71	0.71	583

Model 2

	precision	recall	f1-score	support
Bumps	0.53	0.46	0.50	110
Dirtiness	0.00	0.00	0.00	15
K_Scatch	0.96	0.93	0.94	120
Other_Faults	0.63	0.72	0.67	215
Pastry	0.42	0.38	0.40	37
Stains	0.94	0.65	0.77	23
Z_Scratch	0.64	0.79	0.71	63
accuracy			0.68	583
macro avg	0.59	0.56	0.57	583
weighted avg	0.66	0.68	0.67	583

Fig: Model Assessment Metrics

The auto neural network model with system defined parameters has a precision of 72% and 66% for model 1 and model 2, respectively. The below is the output of precision, recall, f1-score and support for all the individual classes in the target variable defect for auto neural network model. Stains and Pastry has the highest and least assessment metric numbers for model1 while K_Scratch and Dirtiness take those spots for Model 2.

Auto Neural Network

Model 1

	precision	recall	f1-score	support
Bumps	0.54	0.60	0.57	115
Dirtiness	0.91	0.67	0.77	15
K_Scatch	0.97	0.92	0.94	123
Other_Faults	0.68	0.65	0.67	211
Pastry	0.41	0.50	0.45	38
Stains	0.95	0.91	0.93	23
Z_Scratch	0.86	0.83	0.84	58
accuracy			0.72	583
macro avg	0.76	0.73	0.74	583
weighted avg	0.73	0.72	0.72	583

Model 2

	precision	recall	f1-score	support
Bumps	0.55	0.49	0.52	110
Dirtiness	0.00	0.00	0.00	15
K_Scatch	0.93	0.95	0.94	120
Other_Faults	0.64	0.72	0.67	215
Pastry	0.45	0.49	0.47	37
Stains	1.00	0.39	0.56	23
Z_Scratch	0.67	0.75	0.71	63
accuracy			0.68	583
macro avg	0.60	0.54	0.55	583
weighted avg	0.67	0.68	0.67	583

Fig: Model Assessment Metrics

In this case it is observed that Neural network which has customized parameters perform better than the Auto neural network. The Auto neural network cannot be controlled, and it is difficult to interpret the reasons for choosing the specified parameters. Also, Neural Networks are classifying the targets which has more input parameters as the precision for model 1 is greater in both the neural nets.

Model Assessment:

The target variable in this problem is the defect category that a steel plate can probably have with the given properties as input variables. There are 7 defect categories of defects in the target. Since this is a classification problem, the models are assessed based on Accuracy, Precision, Recall, F1 Score. These metrics explain how well a defect is predicted or how bad if a prediction has missed the mark. The below tables provide a comprehensive summary of all assessment metrics from different types of models built (Decision Tree, Random Forest and Neural Network) on all the predictors and also reduced predictors post PCA.

Before performing PCA (Model 1 - 17 independent variables):

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	68%	69%	71%	0.68
Random Forest	74%	74%	74%	0.74
Neural Networks	71%	72%	71%	0.71
Auto Neural Network	72%	73%	72%	0.72

After performing PCA (Model 2 - 9 independent variables):

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	66%	68%	66%	0.67
Random Forest	74%	75%	74%	0.75
Neural Networks	69%	67%	69%	0.68
Auto Neural Network	68%	67%	68%	0.67

Note: Accuracy changes on changing the random seed value.

Since the fault diagnosis is sensitive to False Positives, it is paramount to reduce the FP that minimizes the cost. False positives are represented as part of precision score. Therefore, the models are judged based on precision and the model with highest precision would be chosen as the final model. The project also analyzed two different types of models with different number of input variables. It is noticed that the models have relatively less precision and accuracy for reduced variables. It is also important to weigh the complexity vs Precision tradeoff in such scenario where including more variables might improve the accuracy, but the model becomes more complex.

Strengths and Weakness of Models

Decision Tree

Decision trees work on a simple rule-based prediction algorithm. The predictions are retractable to the source and can be interpreted easily. The tree in this case will also help to identify the important features that are used in predicting the Fault in a steel plate based on the Gini and Entropy. Decision trees have a con of overfitting where the model replicates every scenario of the target variable and can sometimes be risky on unseen data.

Random Forest

Random forests are part of the same family of tree-based classification algorithms. The data where there are large number input variables can be modelled using a Random Forest model. Random forests typically build multiple decision tree models with the same target variable and different input variables and the mode or average of the output value is considered in the final model. This will help overcome the overfitting to some extent that was seen in the decision tree models. In this scenario the random forest might be weak in predicting the faults for the inputs beyond a certain range and on new data.

Neural Network

The neural networks are designed to model large number of input variables with multiple layers of neurons with activation and pre-activation functions that require a lot of computations. The input variables are attached with weights and biases in the neural networks which will make the model complicated and difficult to interpret. The fault detection in this case might have a greater precision but the explain ability of the model is low and is difficult to understand which features made a significant contribution in the prediction.

Final Model and Conclusion

Being a classification model, considering accuracy and recall, Random Forest classifier algorithm with 9 input variables is chosen as final model. This model has highest accuracy (74%) in identifying the faulty steel types when fed into it. Also, number of parameters it deals is less when compared to other models. When implemented in practice, time and memory would be optimized with less input resources.

Finally, from the above models it can be concluded that the faults in a steel plate occur majorly due to the differences in Steel plate thickness, Luminosity Index, Type of Steel, Edges and Area dimensions of the steel plate.

References

1. http://archive.ics.uci.edu/ml/datasets/Steel+Plates+Faults
2. www.semeion.it
3. https://www.researchgate.net/publication/13731626_MetaNet_The_Theory_of_Independent_Judges
4. https://www.kaggle.com/uciml/faulty-steel-plates

Appendix

Python Code

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.decomposition import PCA as pca
from sklearn.decomposition import FactorAnalysis as fact
from sklearn import preprocessing
from factor_analyzer import FactorAnalyzer
from varclushi import VarClusHi
#For the tree
from sklearn.feature_extraction.image import grid_to_graph
from sklearn import tree
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
#For displaying the tree
from sklearn.externals.six import StringIO
from IPython.display import Image, display
#Neural Network
from sklearn.neural_network import MLPRegressor
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
#Multiple Regression
from sklearn.linear_model import LinearRegression
import statsmodels.formula.api as smf
#importing interpretable machine learning
import eli5
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, roc_auc_score
from matplotlib import pyplot as plt
#changing jupyter notebook display size
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)

# #### Loading the data file
#changing the working directory
os.chdir(r'C:\Users\Anirud Vem\Desktop\Faulty steel plates')
#loading the data file
df_steel = pd.read_csv('faults.CSV')
df_steel.head()
df_steel.columns
#shape of the data file
```

```
df_steel.shape
# ### Describing the data
df_steel.describe().T
df_steel.info()
# ##### Missing value detection
#missing value detection
df_steel.isnull().sum()
sns.distplot(df_steel.X_Minimum )
# ##### Plotting the histograms for all the variables to detect outliers
and data distributions
for x in range(0, len(df_steel.columns)-8):
    print("Variable Name:" ,df_steel.columns[x])
    sns.distplot(df_steel.iloc[:,x])
    plt.show()

# ##### Boxplot for all the independent variables
for x in range(0, len(df_steel.columns)-8):
    print("Variable Name:" ,df_steel.columns[x])
    sns.boxplot(df_steel.iloc[:,x])
    plt.show()


# ##### Log transforming the Pixel Areas variable
# ###### Observing a normal distribution after transforming the variables
sns.distplot(np.log(df_steel.Pixels_Areas))
# ###### Observing a normal distribution after transforming the variables
sns.distplot(np.log(df_steel.X_Perimeter))
sns.distplot(np.log(df_steel.Y_Perimeter))
sns.distplot(np.log(df_steel.Sum_of_Luminosity))

# ##### Target variable - Defects distribution
df_steel[['Pastry', 'Z_Scratch', 'K_Scatch', 'Stains', 'Dirtiness',
'Bumps', 'Other_Faults']].sum()

target_index = df_steel[['Pastry', 'Z_Scratch', 'K_Scatch', 'Stains',
'Dirtiness', 'Bumps', 'Other_Faults']].sum().index
target_values = df_steel[['Pastry', 'Z_Scratch', 'K_Scatch', 'Stains',
'Dirtiness', 'Bumps', 'Other_Faults']].sum().values

# ###### distribution of the target variable
#distribution of the target variable
plt.bar(x=target_index, height= target_values)
plt.xticks(rotation=45)

# ##### Clubbing the dummy coded target variable in 7 columns to one
column
df_steel['Class']=0
df_steel['DefType']=''
#data Consolidation - Consolidating the target dummy variables into single
variable
df_steel.loc[df_steel.Pastry==1,'Class'] = 1
df_steel.loc[df_steel.Z_Scratch==1,'Class'] = 2
df_steel.loc[df_steel.K_Scatch==1,'Class'] = 3
df_steel.loc[df_steel.Stains==1,'Class'] = 4
```

```
df_steel.loc[df_steel.Dirtiness==1,'Class'] = 5
df_steel.loc[df_steel.Bumps==1,'Class'] = 6
df_steel.loc[df_steel.Other_Faults==1,'Class'] = 7
#data Consolidation - Consolidating the target dummy variables into single
variable
df_steel.loc[df_steel.Pastry==1,'DefType'] = 'Pastry'
df_steel.loc[df_steel.Z_Scratch==1,'DefType'] = 'Z_Scratch'
df_steel.loc[df_steel.K_Scatch==1,'DefType'] = 'K_Scatch'
df_steel.loc[df_steel.Stains==1,'DefType'] = 'Stains'
df_steel.loc[df_steel.Dirtiness==1,'DefType'] = 'Dirtiness'
df_steel.loc[df_steel.Bumps==1,'DefType'] = 'Bumps'
df_steel.loc[df_steel.Other_Faults==1,'DefType'] = 'Other_Faults'
df_steel.Class.value_counts()
df_steel[['Pastry', 'Z_Scratch', 'K_Scatch', 'Stains', 'Dirtiness',
'Bumps', 'Other_Faults']].sum()
df_steel[['Class','DefType']].info()
df_steel['Class'] = df_steel['Class'].astype('int64')
df_steel[['Class','DefType']].info()

# ###### Removing the dummy coded target variables
df =df_steel.loc[:, ~df_steel.columns.isin(['Pastry', 'Z_Scratch',
'K_Scatch', 'Stains', 'Dirtiness', 'Bumps', 'Other_Faults'])].copy()
df.info()
df.DefType.value_counts()
df.Class.value_counts()
#df.to_csv("Faulty_Steel_Plates.CSV",index=False)
df.describe().T
# #### Corelation
df.corr()
# #### Corelation Plot
corrmat = df.corr()
f, ax = plt.subplots(figsize=(12,12))
sns.heatmap(corrmat,ax=ax,cmap="YlGnBu",linewidths =0.1)
# ### Predictive Modelling with all Unchanged independent variables
# #### Splitting the data into training and testing
#shuffling the data
#df = df.sample(frac=1).reset_index(drop=True)
print(df.shape)

X_parts = df.loc[ :, ~df.columns.isin(['Class','DefType'])]
X_parts.head()
print(X_parts.shape)
Y_cat = df.loc[:, 'DefType']
print(len(Y_cat.unique()))
Y_cat.head()
X_train, X_test, y_train, y_test = train_test_split(X_parts, Y_cat,
test_size=0.3, random_state=12342)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

X_train.describe().T
X_test.describe().T
```

```
y_train.value_counts()
y_test.value_counts()
# #### Decision Trees
col_names = list(df.loc[ :,
~df.columns.isin(['Class','DefType'])].columns.values)
classnames = list(df.DefType.unique())
##Performing Descision trees using all categories
tre2 = tree.DecisionTreeClassifier().fit(X_train,y_train)
predicted = tre2.predict(X_test)
print(metrics.classification_report(y_test, predicted))
# ###### Confusion Matrix of Predicted vs Actual - Test Data
cm = metrics.confusion_matrix(y_test, predicted)
print(cm)
#cm chart
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.xlabel('Actual Value')
plt.ylabel('Predicted Value')
plt.xticks([0,1,2,3,4,5,6], ['I','II','III','IV','V','VI','VII'])

# #### Important Features in the decision tree model
eli5.show_weights(tre2,feature_names = list(X_test.columns),top=None)

# ##### Interpreting which variables are impacting in predicting an
observation
print("Actual defect value is:",y_test.iloc[150])
test_row = pd.DataFrame(X_test.iloc[150,:]).T

# ##### Contribution of feature in predicting the category
#eli5.show_prediction(tre2,
test_row.values[0],feature_names=list(X_test.columns),top=None)
lid = [150]
for i in lid:
    print("Actual test value:",y_test.iloc[i])
    print("------------")
    print("Pedidcted value is ")

eli5.show_prediction(tre2, X_test.iloc[i], feature_names =
list(X_train.columns))

# ##### Decision Tree rules
from sklearn.tree import _tree
def tree_to_pseudo(tree, feature_names):
      left = tree.tree_.children_left
      right = tree.tree_.children_right
      threshold = tree.tree_.threshold
      features = [feature_names[i] for i in tree.tree_.feature]
      value = tree.tree_.value

      def recurse(left, right, threshold, features, node, depth=0):
            indent = "  " * depth
            if (threshold[node] != -2):
                  print (indent,"if ( " + features[node] + " <= " +
str(threshold[node]) + " ) {")
```

```
                if left[node] != -1:
                      recurse (left, right, threshold, features,
left[node], depth+1)
                      print (indent,"} else {")
                      if right[node] != -1:
                            recurse (left, right, threshold, features,
right[node], depth+1)
                      print (indent,"}")
           else:
                print (indent,"return " + str(value[node]))

     recurse(left, right, threshold, features, 0)
tree_to_pseudo(tre2, list(X_train.columns))


# ##### node rules- in text format
dotfile = open(r"C:\Users\Anirud Vem\Desktop\Faulty steel
plates\dt_rules.dot", 'w')
tree.export_graphviz(tre2, out_file=dotfile,
feature_names=list(X_train.columns))
dotfile.close()

# Load libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
from IPython.display import Image
from sklearn import tree
import pydotplus
col_names = list(df.loc[ :,
~df.columns.isin(['Class','DefType'])].columns.values)
classnames = list(df.DefType.unique())

# Create DOT data
dot_data = tree.export_graphviz(tre2, out_file=None,
                                feature_names=col_names,
                                class_names=classnames)

# Draw graph
graph = pydotplus.graph_from_dot_data(dot_data)

# Show graph
Image(graph.create_png())

# Create PDF
#graph.write_pdf("faultyplates.pdf")

# Create PNG
graph.write_png("faultyplates.png")

# #### Building Random Forest Model
##Performing random forest using all categories
from sklearn.ensemble import RandomForestClassifier
rand1 = RandomForestClassifier().fit(X_train,y_train)
rand1_pred = rand1.predict(X_test)
```

```
print(metrics.classification_report(y_test, rand1_pred))

# ##### Confusion matrix on random forest
cm = metrics.confusion_matrix(y_test, rand1_pred)
print(cm)
#cm chart
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.xlabel('Actual Value')
plt.ylabel('Predicted Value')
plt.xticks([0,1,2,3,4,5,6], ['I','II','III','IV','V','VI','VII'])

# #### Feature Importance
eli5.show_weights(rand1,feature_names=list(X_test.columns),top=None)

# ### Neural Network
# Standardize the scaling of the variables by
# computing the mean and std to be used for later scaling.
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)

# Perform the standardization process
steel_data_train_std = scaler.transform(X_train)
steel_data_test_std = scaler.transform(X_test)

nnclass2 = MLPClassifier(activation='relu', solver='sgd',
hidden_layer_sizes=(50,50,50))
nnclass2.fit(steel_data_train_std, y_train)
nnclass2_pred = nnclass2.predict(steel_data_test_std)
print(metrics.classification_report(y_test, nnclass2_pred))
cm = metrics.confusion_matrix(y_test, nnclass2_pred)
print(cm)

plt.matshow(cm)
plt.title('Confusion Matrix')
plt.xlabel('Actual Value')
plt.ylabel('Predicted Value')
plt.xticks([0,1,2,3,4,5,6], ['I','II','III','IV','V','VI','VII'])

# ##### Auto Neural Network
nnclass3 = MLPClassifier(activation='relu', solver='sgd')
nnclass3.fit(steel_data_train_std, y_train)
nnclass3_pred = nnclass3.predict(steel_data_test_std)
print(metrics.classification_report(y_test, nnclass3_pred))
nnclass3.hidden_layer_sizes
cm = metrics.confusion_matrix(y_test, nnclass2_pred)
print(cm)

# ### Modifying the independent variables
# #### Normalizing the data
##Normalizing the data - range transformation
reduc_data = df.loc[:, ~df.columns.isin(['Class','DefType'])]
col_names = reduc_data.columns
reduc_data_values = reduc_data.values
```

```
scaler = preprocessing.MinMaxScaler()
reduc_data_scaled = scaler.fit_transform(reduc_data_values)
reduc_norm = pd.DataFrame(reduc_data_scaled,columns = col_names)
reduc_norm

# ##### Checking for corelation among independent variables
reduc_norm.corr()
# ##### Removing the highly correlated independent variables
reduc_filt = reduc_norm.loc[:,
~reduc_norm.columns.isin(['X_Maximum','Y_Maximum','X_Perimeter','Y_Perimet
er','Sum_of_Luminosity','TypeOfSteel_A400','Log_X_Index','Log_Y_Index','Or
ientation_Index','SigmoidOfAreas'])].copy()
print(reduc_filt.shape)
# #### Principal Component Analysis - Normalized Data
pca_result = pca(n_components=17).fit(reduc_filt)
#Obtain eigenvalues
pca_result.explained_variance_
#plotting PCA
plt.figure(figsize=(7,5))
plt.plot([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16, 17],
pca_result.explained_variance_ratio_, '-o')
plt.ylabel('Proportion of Variance Explained')
plt.xlabel('Principal Component')
#plt.xlim(0.75,4.25)
#plt.ylim(0,1.05)
plt.xticks([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16, 17])

# #### Factor Analysis
fa = FactorAnalyzer(9,rotation='varimax')
fa.fit(reduc_filt)
fa.loadings_

# ##### Converting the factor analysis results into dataframe
fa_load_df = pd. DataFrame([[ 0.33844407, -0.04467156,  0.05897517,
0.00892601, -0.04800935,
         0.64851618,  0.02186099, -0.04142992,  0.05163826],
       [-0.01328759, -0.06920228, -0.03499891,  0.04844939, -0.01424139,
         0.04533559, -0.03528067, -0.41879793,  0.03125152],
       [-0.72074514,  0.05670127,  0.0417465 , -0.089478  ,  0.13270463,
        -0.10070663,  0.1011614 , -0.07000087,  0.0541188 ],
       [ 0.74199926,  0.54179381, -0.04402201, -0.01837413,  0.14797361,
         0.03733792, -0.01443793,  0.06520432,  0.06253286],
       [-0.0680706 ,  0.89982635, -0.06134579, -0.06172601, -0.03008206,
        -0.0147013 ,  0.0262933 ,  0.00520856, -0.00274686],
       [ 0.0422524 , -0.07748696,  0.03156894,  0.33985495, -0.23333652,
         0.45154258,  0.03897444,  0.20986452, -0.0821435 ],
       [ 0.20058351, -0.1649933 , -0.02692938,  0.95248404, -0.00339229,
         0.08776957,  0.10045671, -0.04067847,  0.03059028],
       [ 0.15662106, -0.11157481,  0.12417299,  0.10497948,  0.04047146,
         0.15178128, -0.14306714,  0.5444327 ,  0.08100534],
       [ 0.35419702,  0.17752288, -0.05143901,  0.03698364, -0.08560933,
         0.24708459,  0.11268649, -0.01915752,  0.07701451],not
       [-0.23587972,  0.03629773, -0.02064437, -0.02012922,  0.73473651,
        -0.11963355, -0.03487342,  0.073519  , -0.01082887],
```

```
       [ 0.06421641,  0.05923367, -0.07702312,  0.07855792, -0.03956167,
         0.04517189,  0.77518524, -0.03880839,  0.00952248],
       [-0.75356941,  0.04681758, -0.29746262, -0.08637265,  0.19271434,
        -0.13349901, -0.0967648 , -0.11817857,  0.08639293],
       [ 0.30872356,  0.06221269, -0.77756023,  0.07095359, -0.42040798,
         0.090543  ,  0.15960372, -0.05478872,  0.27370287],
       [ 0.58744241, -0.11621573,  0.47466621,  0.08991385, -0.39992519,
         0.1555088 ,  0.07998125,  0.102352  ,  0.38042246],
       [ 0.12740829, -0.07893841,  0.69026662,  0.00282869, -0.15139386,
         0.07819046, -0.04281443,  0.1463228 ,  0.10609889],
       [-0.87309107, -0.05623831,  0.09138196, -0.13068968,  0.16868582,
        -0.17634411, -0.13516937, -0.07281583, -0.14336892],
       [ 0.17279672,  0.96393867, -0.07136082, -0.136412  ,  0.07667582,
        -0.06051132,  0.06990843,  0.01728517, -0.01983381]],columns =
["f1", "f2","f3","f4","f5","f6","f7","f8","f9"],index=reduc_filt.columns)
fa_load_df

#selecting the highest variance shared variable from each factor
'LogOfAreas','Luminosity_Index','Edges_X_Index','TypeOfSteel_A300','Empty_
Index','X_Minimum','Square_Index','Steel_Plate_Thickness','Edges_Y_Index'

# ### Variable Clustering Algorithm to Cluster the Independent variables
demo1_vc = VarClusHi(reduc_filt,maxeigval2=1,maxclus=9)
demo1_vc.varclus()
demo1_vc.info

# ##### Variable clustering algorithm suggested 4 clusters
demo1_vc.rsquare

# ## Predictive Modelling - Variables suggested by factor analysis
#variables are
#'LogOfAreas','Luminosity_Index','Edges_X_Index','TypeOfSteel_A300','Empty
_Index','X_Minimum','Square_Index','Steel_Plate_Thickness','Edges_Y_Index'

reduc_factor = reduc_filt.loc[:,
reduc_filt.columns.isin(['LogOfAreas','Luminosity_Index','Edges_X_Index','
TypeOfSteel_A300','Empty_Index','X_Minimum','Square_Index','Steel_Plate_Th
ickness','Edges_Y_Index'])].copy()

# #### Splitting the data into training and testing
Y_cat = df.loc[:, 'DefType']
print(len(Y_cat.unique()))
Y_cat.head()
X_train, X_test, y_train, y_test = train_test_split(reduc_factor, Y_cat,
test_size=0.3, random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

# #### Decision Trees
col_names = list(reduc_factor.columns.values)
classnames = list(df.DefType.unique())
col_names
```

```
##Performing Descision trees using all categories
tre2 = tree.DecisionTreeClassifier().fit(X_train,y_train)
predicted = tre2.predict(X_test)
print(metrics.classification_report(y_test, predicted))

##Performing Descision trees using all categories
cm = metrics.confusion_matrix(y_test, predicted)
print(cm)

#cm chart
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.xlabel('Actual Value')
plt.ylabel('Predicted Value')
plt.xticks([0,1,2,3,4,5,6], ['I','II','III','IV','V','VI','VII'])

# #### Important Features in the decision tree model
eli5.show_weights(tre2,feature_names = list(X_test.columns),top=None)
print("Actual defect value is:",y_test.iloc[101])
test_row = pd.DataFrame(X_test.iloc[101,:]).T
test_row

# ##### Contribution of feature in predicting the category
#eli5.show_prediction(tre2,
test_row.values[0],feature_names=list(X_test.columns),top=None)
lid = [101]
for i in lid:
    print("Actual test value:",y_test.iloc[i])
    print("------------")
    print("Pedidcted value is ")

eli5.show_prediction(tre2, X_test.iloc[i], feature_names =
list(X_train.columns))

# #### Building Random Forest Model
##Performing random forest using all categories
from sklearn.ensemble import RandomForestClassifier
rand1 = RandomForestClassifier().fit(X_train,y_train)
rand1_pred = rand1.predict(X_test)
print(metrics.classification_report(y_test, rand1_pred))

##Performing random forest using all categories
cm = metrics.confusion_matrix(y_test, rand1_pred)
print(cm)

# #### Feature Importance
eli5.show_weights(rand1,feature_names=list(X_test.columns),top=None)

# #### Neural Network
# Standardize the scaling of the variables by
# computing the mean and std to be used for later scaling.
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)
```

```
# Perform the standardization process
steel_data_train_std = scaler.transform(X_train)
steel_data_test_std = scaler.transform(X_test)
nnclass2 = MLPClassifier(activation='relu', solver='sgd',
                         hidden_layer_sizes=(50,50,50))
nnclass2.fit(steel_data_train_std, y_train)
nnclass2_pred = nnclass2.predict(steel_data_test_std)
print(metrics.classification_report(y_test, nnclass2_pred))
cm = metrics.confusion_matrix(y_test, nnclass2_pred)
print(cm)
plt.matshow(cm)
plt.title('Confusion Matrix')
plt.xlabel('Actual Value')
plt.ylabel('Predicted Value')
plt.xticks([0,1,2,3,4,5,6], ['I','II','III','IV','V','VI','VII'])

# ##### Auto Neural Network
nnclass3 = MLPClassifier(activation='relu', solver='sgd')
nnclass3.fit(steel_data_train_std, y_train)
nnclass3_pred = nnclass3.predict(steel_data_test_std)
print(metrics.classification_report(y_test, nnclass3_pred))
nnclass3.hidden_layer_sizes
cm = metrics.confusion_matrix(y_test, nnclass3_pred)
print(cm)
```