Defect type classification in Steel plates

Uttej Kuumar Goud Thandu
CECS Department
University of Michigan
Dearborn, USA
uttej@umich.edu

***Abstract*—This paper proposes a Machine learning-based method for classifying the defect types in stainless steel plates. The approach employs a class of methodologies i.e., K-Nearest Neighbor, Random Forest methods. Validated the results using the accuracy score, confusion matrices, recall, precision, ROC curves. The obtained results are encouraging and demonstrate the approach's effectiveness of about 94% accuracy.**

***Index Terms*—Machine Learning (ML), K-Nearest Neighbors(KNN), Random Forest, Performance Matrix (PM), Accuracy, Precision, Recall.**

I. Introduction

In contemporary industrial processes, fault diagnostic aims to determine the timing, location, and extent of specific defects. It is frequently based on on-the-spot data and systematic fault categorization records. Not only will an efficient system for determining fault types and causes save maintenance costs and waste, but it will also increase production efficiency and product quality. Even specialists with fault diagnostic guides have always had to carefully evaluate operating surroundings in order to determine the reasons of a given failure. However, more sophisticated methods emerging from machine learning research have made significant progress in addressing this problem fast and correctly. [1] Logistic regression (LR), decision trees (DT), and support vector machines (SVMs) are common examples. [2]
The first section of this paper provides a brief overview of the basic concepts of KNN and Random Forest. Then, as an object of fault diagnosis, frequent defects reported in the steel plate faults dataset are chosen and utilized to assess the classification performance of the algorithms in question. Finally, the precisions of various approaches are compared, and conclusions are drawn.

II. Dataset Description

Dataset is collected from a company in Italy on surface quality in stainless steel plates. Images are collected from 1941 defective plates to understand the surface quality and defect type.
27 features are post-extracted from the stainless-steel plate images.
X-Minimum, X-Maximum, Y-Minimum, Y-Maximum, Pixels-Areas, X-Perimeter, Y-Perimeter, SumofLuminosity, MinimumofLuminosity, MaximumofLuminosity, LengthofConveyer, TypeOfSteel-A300, TypeOfSteel-A400, SteelPlateThickness, Edges-Index, Empty-Index, Square-Index, OutsideXIndex, EdgesXIndex, EdgesYIndex, OutsideGlobalIndex, LogOfAreas, LogXIndex, LogYIndex, Orientation-Index, Luminosity-Index, SigmoidOfAreas
And different defect types are given in the data.
Pastry, Z-Scratch, K-Scatch, Stains, Dirtiness, Bumps, Other-Faults

A. Data Prepossessing

The data set consists of 1941 samples with no missing data points. All data is numeric (dotted number and integer), the last seven columns are label columns.

	X_Minimum	X_Maximum	Y_Minimum
count	1941.000000	1941.000000	1.941000e+03
mean	571.136012	617.964451	1.650685e+06
std	520.690671	497.627410	1.774578e+06
min	0.000000	4.000000	6.712000e+03
25%	51.000000	192.000000	4.712530e+05
50%	435.000000	467.000000	1.204128e+06
75%	1053.000000	1072.000000	2.183073e+06
max	1705.000000	1713.000000	1.298766e+07

8 rows × 34 columns

Fig. 1. *Datadescription*

III. Data Analysis and Visualization

1) Kurtosis*:* Kurtosis is a statistical parameter that is used to describe a signal [3]. In essence, it provides a measure of a random signal's "peakedness." Signals with a higher kurtosis value have more peaks that are greater than three-sigma, or peaks that are greater than three times the signal's RMS value. Kurtosis is defined as a normalized value "K" obtained by dividing the fourth statistical moment by the square of the

second statistical moment. The K calculation for N samples is shown in the equation below. The formula for the calculation of Kurtosis is as follows:

$$K = \frac{\frac{1}{N}\sum(x_i^4)}{(\frac{1}{N}\sum(x_i^2))^2} \quad (1)$$

2) Skewness:*:* Skewness indicates the symmetry of the probability density function (PDF) of the amplitude of a time series. A time series with an equal number of large and small amplitude values has a skewness of zero. A time series with many small values and few large values is positively skewed (right tail) and has a positive skewness value. A time series with a large number of large values and a small number of small values is negatively skewed (left tail), and the skewness value is negative. More details about calculating and finding the skewness in a signal is cited in [4]

The skewness of a set of numbers, xn, n = 1, ..., N, is given

$$\gamma = \frac{1}{N\sigma^3}\sum_{(n=1)}^{N}(x_n - \mu)^3 \quad (2)$$

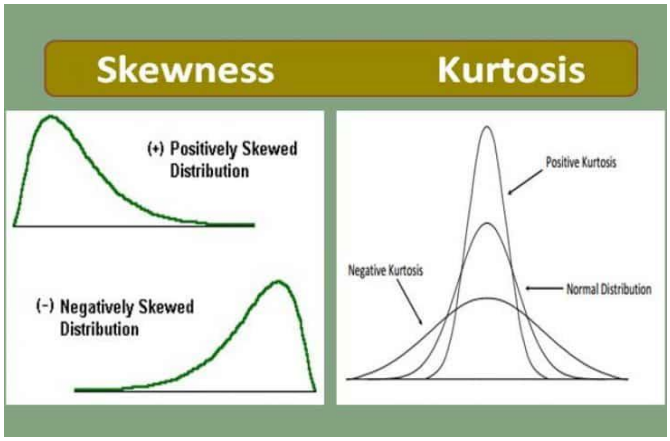

Fig. 2. *Kurtosis and Skewness classification*

The description of the dataset gives the statistical summary of the features in the data set is defined as Descriptive Statistics.

It aims to present the summary and analysis of the descriptive statistics data set with statistical calculations.

With the basic calculations used; mean, standard deviation, mode, min, max, title and curvature values are obtained.

The description after adding the values of Skewness and Kurtosis.Let's look at the distribution of the output(class, target) variable in the dataset. The data consists of Other-Faults 673, Bumps 402, K-Scatch 391, Z-Scratch 190, Pastry 158, Stains 72, dirty 55 type defects these can be seen the graph below.

Examining the data with graphics allows much better

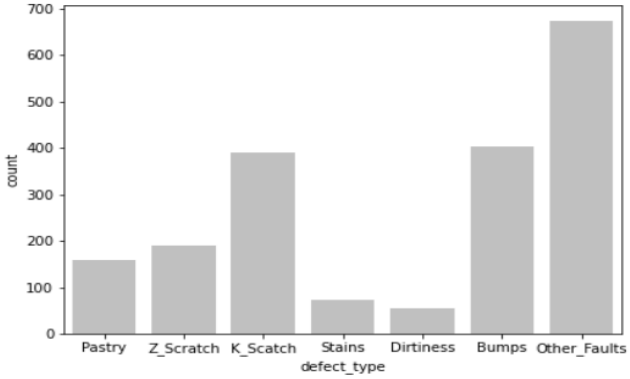

Fig. 3. *Count of Defect types*

understanding for those who are not machine learning experts to interpret the data, as well as allowing experts in machine learning to interpret the data in better way. I have used Histogram for representation of the distribution of data.

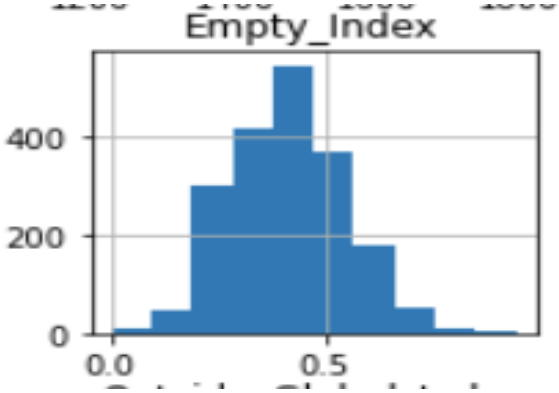

Fig. 4. *Plot of Empty Index*

Fig.4, Fig.5 gives normal distribution with respect to the data distribution and can be the significant features in deciding the output.

The fig. 6,7,8,9,10, gives the information that the data is distributed exponentially and most of the charts are left skewed

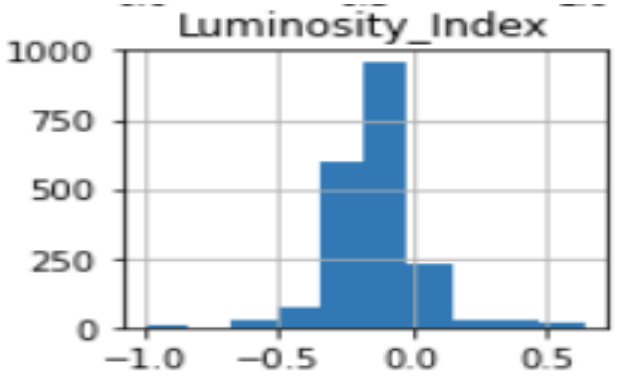

Fig. 5. *Plot of Luminosity Index*

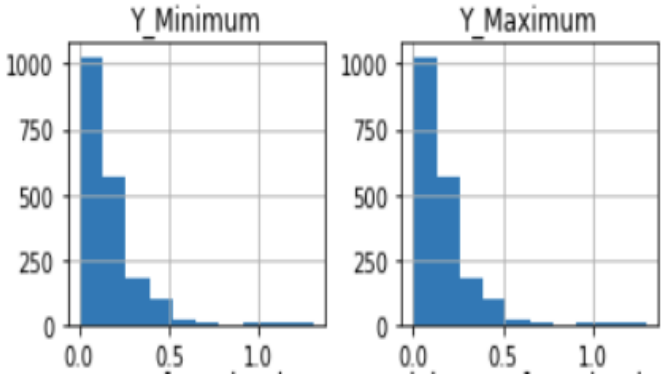

Fig. 7. *Plot of $Y_Mimimum$ $Y_Maximum$*

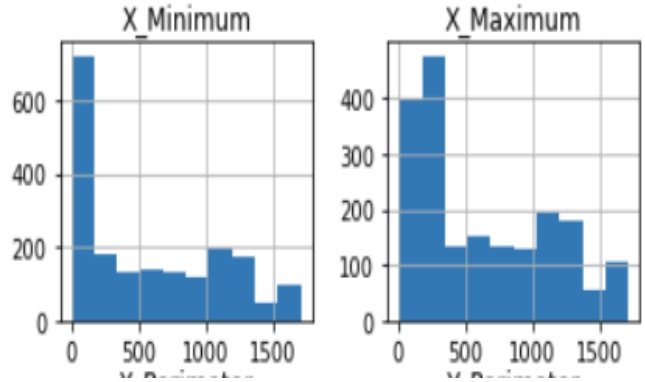

Fig. 6. *Plot of $X_Mimimum$ $X_Maximum$*

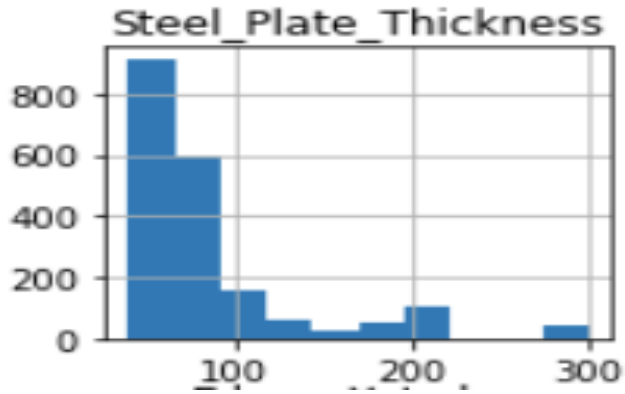

Fig. 8. *Plot of Steel Plate Thickness*

IV. Methodologies Implemented

For the purpose of classification of defects I have made use of 2 types of classifications, they are: K-Nearest Neighbors(KNN), Random Forest(RF).

A. K-Nearest Neighbors(KNN)*:*

The k-nearest neighbors algorithm (kNN) is a non-parametric classification method developed in 1951 by Evelyn Fix and Joseph Hodges and later expanded by Thomas Cover. It is employed in classification and regression. The input in both cases consists of the k closest training examples in the data set. The outcome is determined by whether kNN is used for classification or regression. kNN is a classification method in which the function is only approximated locally and all computation is postponed until the function is evaluated. Because this algorithm relies on distance for classification, normalizing the training data can significantly improve its accuracy if the features represent different physical units or come in vastly different scales. A brief explanation of how KNN works is given [5]

The k-nearest neighbour classifier can be viewed as assigning the k nearest neighbours a weight 1/k and all others 0 weight. This can be generalised to weighted nearest neighbour classifiers. That is, where the ith nearest neighbour is assigned a weight W_{ni}, with $\sum_{(i=1)}^{N} W_{ni} = 1$.

Let C_n^{wnn} denote the weighted nearest classifier with weights $\{w_{ni}\}_{i=1}^{n}\{w_{ni}\}_{i=1}^{n}$. Subject to regularity conditions[further explanation needed] on the class distributions the excess risk has the following asymptotic expansion

B. Random Forest(RF)*:*

The random forest classifier consists of a combination of tree classifiers where each classifier is generated using a random vector sampled independently from the input vector, and each tree casts a unit vote for the most popular class to classify an input vector (Breiman). The random forest classifier used for this study consists of using randomly selected features or a combination of features at each node to grow a tree. Bagging, a method to generate a training dataset by randomly drawing with replacement N examples, where N is the size of the original training set (Breiman), was used for each feature/feature combination selected. Any examples (pixels) are classified by taking the most popular voted class from all the tree predictors in the forest (Breiman). Design of a decision tree required the choice of an attribute selection measure and a pruning method. There are many approaches to the selection of attributes used for decision tree induction and most approaches assign a quality measure directly to the attribute. The most frequently used attribute selection measures in decision tree induction are the Information Gain Ratio criterion (Quinlan) and the Gini Index (Breiman et al.).

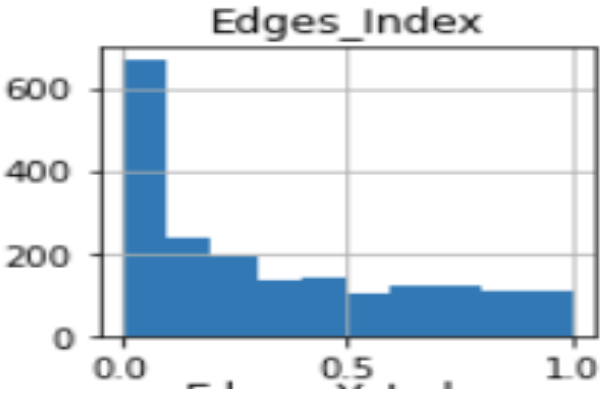

Fig. 9. *Plot of Edges Index*

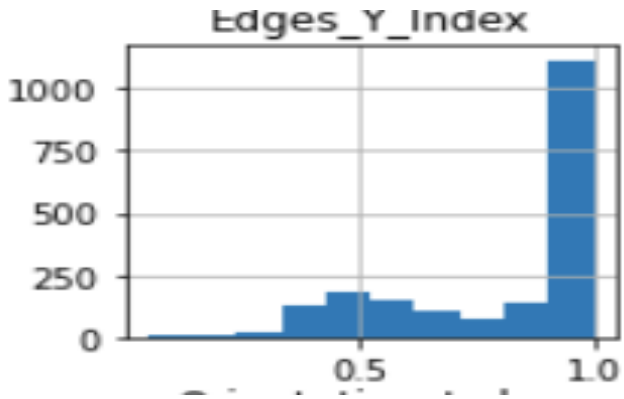

Fig. 10. *Plot of Edges – Y Index*

The random forest classifier uses the Gini Index as an attribute selection measure, which measures the impurity of an attribute with respect to the classes. [6]

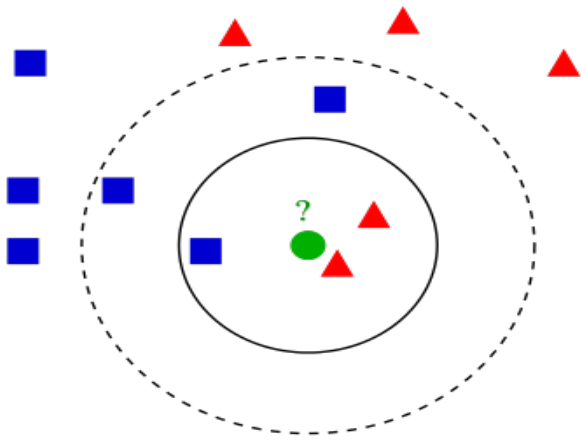

Fig. 11. *K – NearestNeighbour*

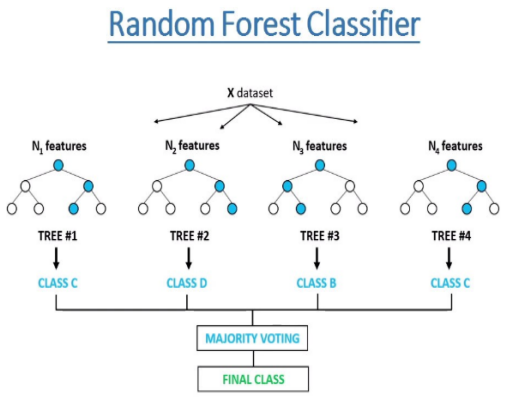

Fig. 12. *K – NearestNeighbour*

C. Feature selection*:*

Significant features are selected on the basis of their P-Scores, considering the value of P¡0.05, after fitting into the OLS(Ordinary Least Square) model, below are the features that are found insignificant: 'X-Maximum','X-Perimeter','Luminosity-Index','Log-Y-Index','Log-X-Index','LogOfAreas','Edges-Y-Index','Edges-X-Index','Empty-Index' ,'Steel-Plate-Thickness','Minimum-of-Luminosity','Sum-of-Luminosity','Pixels-Areas'. The features that are found to be significant are as follows: X-Minimum', 'Y-Minimum', 'Y-Maximum', 'Y-Perimeter', 'Maximum-of-Luminosity', 'Length-of-Conveyer', 'TypeOfSteel-A300', 'TypeOfSteel-A400', 'Edges-Index', 'Square-Index', 'Outside-X-Index', 'Outside-Global-Index', 'Orientation-Index', 'SigmoidOfAreas'

V. Results

A. Performance Evaluation Measures*:*

Since we split the dataset into train and test sets with equal proportions from each class, Accuracy can be used as a metric to evaluate the methods proposed. In addition to Accuracy, we have also used Precision, Recall, F_1Score for performance

evaluation. So, the efficacy of a method is determined by percentage of correct identification of a defect.
We calculated Accuracy, precision, recall, F_1Score as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F_1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

B. Experimental Results and Analysis:

For this experiment, we trained a K-Nearest Neighbours, Gaussian Naive Bayes and Support Vector Machine classifiers on the feature set after removing the outliers. We split the data and used 80 percent of the data for training and remaining 20 percent for testing.
We performed hyperparameter tuning on KNN for the optimal value of K and the weight function. we fitted and testing model for various combinations values of K ranging from 1 to 20 and weight functions and settled on K = 11

$$Weightfunction = \frac{1}{1 + distance} \tag{7}$$

We obtained an average accuracy of 46.7%, Precision, Recall and standard deviation of 0.072% with a stratified 5-fold Cross validation on the KNN.
From the above table it can be said that the precision for the

Accuracy: 0.46786632390745503

classification_report

	precision	recall	f1-score	support
0	0.32	0.46	0.38	79
1	0.33	0.40	0.36	10
2	0.86	0.79	0.82	80
3	0.47	0.55	0.50	138
4	0.11	0.03	0.05	29
5	0.00	0.00	0.00	13
6	0.12	0.05	0.07	40
accuracy			0.47	389
macro avg	0.32	0.33	0.31	389
weighted avg	0.44	0.47	0.45	389

Fig. 13. *Precission Recall $f1 - Score$ Support report*

Y feature at index 4 has about 47% and everthing else has less than that.

For fig.14.,after finding the accuracy with several combinations of k values, the accuracy is found to be at saturated level at k=17 and later it is same and continues to be the same.

```
k: 1 : mean accuracy: 0.32254559873116573 :std: 0.06480366069495755
k: 3 : mean accuracy: 0.3792201956119482 :std: 0.0688944310464113
k: 5 : mean accuracy: 0.3998361089082739 :std: 0.06397592489791809
k: 7 : mean accuracy: 0.42609833465503566 :std: 0.06099321933182027
k: 9 : mean accuracy: 0.43280993920169175 :std: 0.06731233321109982
k: 11 : mean accuracy: 0.44569653713983604 :std: 0.07741423773564905
k: 13 : mean accuracy: 0.4410573618821041 :std: 0.07203198936385827
k: 15 : mean accuracy: 0.4374464710547185 :std: 0.06520700040407791
k: 17 : mean accuracy: 0.4379645783769496 :std: 0.06321458019498229
k: 19 : mean accuracy: 0.438995506212001 :std: 0.07543590934990105
```

Fig. 14. *KNN with different folds*

```
array([[36,  3,  0, 36,  2,  0,  4],
       [ 1,  3,  0,  5,  0,  1,  0],
       [ 4,  0, 64,  9,  0,  0,  1],
       [35,  1,  8, 75,  4,  0,  6],
       [12,  0,  3, 22,  1,  0,  2],
       [ 7,  0,  0,  4,  0,  0,  1],
       [16,  1,  3, 17,  1,  1,  0]], dtype=int64)
```

Fig. 15. *Confusion Matrix for KNN*

Also, tested the model for the results from the Random Forest classification, the results are much better compared to the KNN classification. The accuracy obtained out of this model is about 93% with out the fit from significant features, the testing accuracy is found to be 75% from the model. The results for RFC is as follows.

	precision	recall	f1-score	support
0	0.65	0.63	0.64	81
1	1.00	0.80	0.89	10
2	1.00	0.88	0.94	78
3	0.65	0.82	0.73	129
4	0.70	0.35	0.47	40
5	0.79	0.92	0.85	12
6	0.92	0.85	0.88	39
accuracy			0.75	389
macro avg	0.81	0.75	0.77	389
weighted avg	0.76	0.75	0.75	389

Fig. 16. *Precission Recall $f1 - Score$ Support report*

The RFC model gives the precision about 100% for the output features that are at the index 1,2 and every other feature is at good level comapred to the KNN fit.

```
array([[ 51,   0,   0,  28,   2,   0,   0],
       [  0,   8,   0,   2,   0,   0,   0],
       [  1,   0,  69,   6,   0,   2,   0],
       [ 17,   0,   0, 106,   4,   1,   1],
       [  7,   0,   0,  17,  14,   0,   2],
       [  0,   0,   0,   1,   0,  11,   0],
       [  3,   0,   0,   3,   0,   0,  33]], dtype=int64)
```

Fig. 17. *Confusion Matrix for RFC*

C. Results after fitting significant features

After fitting the features X-Minimum', 'Y-Minimum', 'Y-Maximum', 'Y-Perimeter', 'Maximum-of-Luminosity', 'Length-of-Conveyer', 'TypeOfSteel-A300', 'TypeOfSteel-A400', 'Edges-Index', 'Square-Index', 'Outside-X-Index', 'Outside-Global-Index', 'Orientation-Index', 'SigmoidOfAreas'. the accuracy for the model Random Forest increased.

```
The training accuracy of the model is 0.9445876288659794
The testing accuracy of the model is 0.7686375321336761
```

Fig. 18. *Accuracy after Significant feature*

VI. Conclusion

From the above results it can be said that the features that has been used for the classification are good enough, however the data volume is not greatly affecting the results, which concludes that more data volume increases the training and testing accuracy of the model. The significant features are most of them show biased distribution. But using the Random Forest classifier the results are significantly better compared to the KNN fit. The accuracy of the RFC is no where near to the KNN and adding more volume of data increases the model accuracy. Using the RFC the data can be classified with great precision.

VII. Appendix

The code goes here:

——— Imported necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn import feature_selection
from sklearn.ensemble import RandomForestClassifier
----#Data reading
data=pd.read_csv('S:/Sem 3/Multivariate
Statistics/Assignments/Final project/surface_faults.csv')
-----#defining X and Y variables
data_new=data.drop(data[['Pastry','Z_Scratch','K_Scatch'
,'Stains','Dirtiness','Bumps','Other_Faults']], axis=1)
le=LabelEncoder()
data_new['defect_type']=le.fit_transform(y)
data_new
-----#Heat Map
plt.figure(figsize=(30,50))
sns.heatmap(data_new.corr(), cmap="YlGnBu", annot = True)
plt.show()
------#Skewnesss and Kurtosis
describe=data_new.describe().T
describe['Skew']= data_new.skew().values
describe['Kurtosis']= data_new.kurt().values
print(data_new['defect_type'].describe())
describe
--------#Countplot
fig, ax=plt.subplots(1,2,figsize=(15,5))
sns.countplot(x='defect_type',
data=data_new, ax=ax[0],color='silver')
---------#Box Plot
data_new.hist(figsize=(15,15))
plt.show()
-------#Label Encoder
le=LabelEncoder()
X=data_new.drop('defect_type',axis=1)
Y=le.fit_transform(data_new['defect_type'])
data_new
------#train test split
x_train, x_test, y_train, y_test = train_test_split
(X, Y, train_size = 0.8,random_state=1)
------#Logistic Regression
lr = sm.OLS(y_train,x_train).fit()
lr.summary()
-------#creating new dataset with significant features
data_new_droped=data_new.drop(['defect_type','X_Maximum',
'X_Perimeter','Luminosity_Index','Log_Y_Index',
'Log_X_Index','LogOfAreas','Edges_Y_Index',
'Edges_X_Index','Empty_Index'
,'Steel_Plate_Thickness','Minimum_of_Luminosity',
'Sum_of_Luminosity','Pixels_Areas'],axis=1)
X1=data_new_droped
Y1=data_new['defect_type']
-------#KNN fit
knn = KNeighborsClassifier(n_neighbors=5, p=2 )
clf_fit=knn.fit(x_train, y_train)
# kn_scores=clf_fit.score(x_test, y_test)
y_pred=knn.predict(x_test)
-------#KNN predict
knn.predict(x_test)
--------#Accuracy and classification report
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# print("Precision:",metrics.precision_score(y_test, y_pred))
# print("Recall:",metrics.recall_score(y_test, y_pred))
print('\nclassification_report\n',
classification_report(y_test, y_pred))
-------#Cross_val_score
cross_val_score(knn, X, y, cv=10).mean()

for k in range(1,20,2):
```

```
    knn= KNeighborsClassifier(n_neighbors=k)
    score= cross_val_score(knn, X,Y, cv=10)

    print('k:',k, ': mean accuracy:',
    score.mean(),':std:',score.std() )
-------#Confusion matrix
confusion_matrix(y_test,y_pred)
-------#Random Forest
rfc = RandomForestClassifier(n_estimators=30, random_state=68, max_depth = 10)
rfc.fit(x_train, y_train)
y_pred_rfc = rfc.predict(x_test)
y_pred_train = rfc.predict(x_train)
print('The training accuracy of the model is
{}'.format(accuracy_score(y_train,y_pred_train)))
print('The testing accuracy of the model is
{}'.format(accuracy_score(y_test,y_pred_rfc)))
------#Metrics
print(metrics.classification_report(y_test,y_pred_rfc ))
-----#Confusion matrix for RFC
confusion_matrix(y_test,y_pred_rfc)
------# train test split for new data
x_train, x_test, y_train, y_test = train_test_split(X1, Y1
, train_size = 0.8,random_state=1)
-----##KNN for selected features
knn = KNeighborsClassifier(n_neighbors=5, p=2 )
clf_fit=knn.fit(x_train, y_train)
# kn_scores=clf_fit.score(x_test, y_test)
y_pred=knn.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print('\nclassification_report\n',classification_report(
y_test, y_pred))
------## random forest for selected features
rfc = RandomForestClassifier(n_estimators=30, random_state=68, max_depth = 10)
rfc.fit(x_train, y_train)
y_pred_rfc = rfc.predict(x_test)
y_pred_train = rfc.predict(x_train)
print('The training accuracy of the model is
{}'.format(accuracy_score(y_train,y_pred_train)))
print('The testing accuracy of the model is
{}'.format(accuracy_score(y_test,y_pred_rfc)))
```

REFERENCES

[1] Y. Tian, M. Fu, and F. Wu, "Steel plates fault diagnosis on the basis of support vector machines," *Neurocomputing*, vol. 151, pp. 296–303, 2015.

[2] ——, "Steel plates fault diagnosis on the basis of support vector machines," *Neurocomputing*, vol. 151, pp. 296–303, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231214012193

[3] J. Longbottom, A. Walden, and R. White, "Principles and application of maximum kurtosis phase estimation1," *Geophysical Prospecting*, vol. 36, no. 2, pp. 115–138, 1988.

[4] M. Källersjö, J. S. Farris, A. G. Kluge, and C. Bult, "Skewness and permutation," *Cladistics*, vol. 8, no. 3, pp. 275–287, 1992.

[5] G. E. Batista, M. C. Monard *et al.*, "A study of k-nearest neighbour as an imputation method." *His*, vol. 87, no. 251-260, p. 48, 2002.

[6] M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.