INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE

Department of Computer Science and Engineering

(2021-2023)

CSN -515 DATA MINING AND WAREHOUSING PROJECT

"STEEL PRODUCTION IN FACTORIES"

GROUP ID - 10

Submitted To:

Dr. Durga Toshniwal
Professor, Computer Engg.
Department of Computer Science & Engineering
And Head - Centre for Transportation Systems (CTRANS),
IIT Roorkee, Uttarakhand, India- 247667

Submitted By:

Abhishek Bagde (21535001)
Prashant Shukla (21535021)
Ritika Khurana (21535025)
Shree Chand Khichar (21535027)
Stuti Kothari (21535031)
Tanishq Anand (21535034)

MTech CSE 1st year

Group Members Contribution

Enrolment Number	Member name	Contribution	Page No.
21535001	Abhishek Bagde	K-means Clustering	11,12,14
21535021	Prashant Shukla	Logistic Regression	9,16
21535025	Ritika Khurana	Data Pre-processing, DBSCAN	7,8
21535027	Shree Chand Khichar	Hierarchical Clustering	13,15 ,19
21535031	Stuti Kothari	Decision tree	10,16,17
21535034	Tanishq Anand	Data Analysis, pre-processing	5,6,7

ACKNOWLEDGEMENT

We are highly indebted to Dr. Durga Toshniwal (Professor) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. We would like to express our gratitude towards members of INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE for their kind cooperation and encouragement which helped in completion of this project.

Our thanks and appreciations also go to colleagues in developing the project and people who have willingly helped us out with their abilities.

CONTENTS

1. Problem Statement

Suggest and implement an appropriate solution using data mining techniques for the problem “Steel Production in Factories”. Choose a real world data based on the problem. The data must contain enough samples.

While collecting the data, we come across various futile attributes. Extraction of useful data is not possible by going through them manually. Data pre-processing was heavily required in order to extract useful attributes. Most of the pre-processing was done during feeding of these records. However, pre-processing specific to certain problems was also required later.

We could identify three problems which can be solved using data mining techniques.

1.1 Steel fault Analysis & Pre-processing

We have collected data of type of steel defects regarding the attributes (Metallurgic) from the research paper published by Semeion, Research Center of Sciences of Communication.
The dataset includes 1941 observation, and 27 features. The data is already labelled and there are 7 types of steel plate faults that are added to the dataset as 7 fields representing the defects.
Later we done the done the pre-processing part in which we gone through feature skewness feature scaling and some sort of outlier analysis. For outlier analysis purpose we utilize the DBSCAN.

1.2 Classification and Clustering of dataset

We have collected the data and tried to work on classification and clustering between the other attribute of the steel and the defects. There are 27 types of of metallurgic attributes pertaining to steel metal and 7 kind of defects which are Pastry, Z_Scatch, K_Scatche, Stains, Dirtiness, Bumps, Other Faults. For classification purpose we have utilize the Logistic Regression and Decision tree. And for clustering Purpose we utilize the k-means and hierarchal approach (down). After classification we also shown the Accuracy graph between them.

1.3 Association Rule Mining regarding defects.

Association rule mining is a crucial part of the data mining. In this one we find out the strong association rule of happening an event in order to a previous event. For association rule mining purpose, we utilized the Apriori algorithm. Here we tried to find the associativity between the defects of the steel defects frequent itemset.

2. Data Pre-processing :-

2.1 Dataset

Our dataset contains 27 attributes of steel , with 7 types of faults , resulting in the number of columns to be 34 . It contains 1941 instances of integer and real type characteristics .

Number	Feature Attributes	Number	Feature Attributes	Number	Feature Attributes
Attribute 1	X_Minimum	Attribute 10	Maximum_of_Luminosity	Attribute 19	Edges_X_Index
Attribute 2	X_Maximum	Attribute 11	Length_of_Conveyer	Attribute 20	Edges_Y_Index
Attribute 3	Y_Minimum	Attribute 12	TypeOfSteel_A300	Attribute 21	Outside_Global_Index
Attribute 4	Y_Maximum	Attribute 13	TypeOfSteel_A400	Attribute 22	LogOfAreas
Attribute 5	Pixels_Areas	Attribute 14	Steel_Plate_Thickness	Attribute 23	Log_X_Index
Attribute 6	X_Perimeter	Attribute 15	Edges_Index	Attribute 24	Log_Y_Index
Attribute 7	Y_Perimeter	Attribute 16	Empty_Index	Attribute 25	Orientation_Index
Attribute 8	Sum_of_Luminosity	Attribute 17	Square_Index	Attribute 26	Luminosity_Index
Attribute 9	Minimum_of_Luminosity	Attribute 18	Outside_X_Index	Attribute 27	SigmoidOfAreas

Table 1 . List of Attributes in the Steel dataset

Class	Fault Type	No of Samples
1	Pastry	158
2	Z_Scratch	190
3	K_Scratch	391
4	Stains	72
5	Dirtiness	55
6	Bumps	402
7	Other_Faults	673

Table 2. List of classes and number of samples

2.2 Feature Skewness:-

Skewness refers to a distortion or asymmetry that deviates from the symmetrical bell curve, or normal distribution, in a set of data. If the curve is shifted to the left or to the right, it is said to be skewed.

Skewness, in statistics, is the degree of asymmetry observed in a probability distribution. Distributions can exhibit right (positive) skewness or left (negative) skewness to varying degrees. A normal distribution (bell curve) exhibits zero skewness

Besides positive and negative skew, distributions can also be said to have zero or undefined skew. In the curve of a distribution, the data on the right side of the curve may taper differently from the data on the left side. These tapering's are known as "tails." Negative skew refers to a longer or fatter tail on the left side of the distribution, while positive skew refers to a longer or fatter tail on the right.

2.2.1 Aggregation

There were records of different types of defects (e.g. Pastry, Z_Scatch, K_Scatch, Stains, Dirtiness, Bumps, Other_Faults.) in our data. We were only interested in the total defects in our database so we combined all kind of defects and put them in an single column.

2.2.2 Normalization

The range of values are different for the steel metallurgic attributes for our dataset so those attributes normalized to account for the different range of attributes.

2.3 Feature Scaling

2.3.1 Missing Values

We have collected data of type of steel defects regarding the attributes (Metallurgic) from the research paper published by Semeion, Research Center of Sciences of Communication.
The dataset includes 1941 observation, and 27 features. In our dataset, there were no missing values although we cross checked that.

2.3.2 Outlier Analysis :-

Outlier is a data object that deviates significantly from the rest of the data objects and behaves in a different manner. They can be caused by measurement or execution errors. The analysis of outlier data is referred to as outlier analysis or outlier mining.

An outlier cannot be termed as a noise or error. Instead, they are suspected of not being generated by the same method as the rest of the data objects.

Outliers are of three types, namely –

- Global Outliers
- Collective Outliers
- Contextual Outliers

In our project we have utilized DBSCAN for outlier detection and removal.

2.3.2.1 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a well-known data clustering algorithm that is commonly used in data mining and machine learning.
Based on a set of points (let's think in a bidimensional space as exemplified in the figure), DBSCAN groups together points that are close to each other based on a distance measurement (usually Euclidean distance) and a minimum number of points. It also marks as outliers the points that are in low-density regions.
The DBSCAN algorithm basically requires 2 parameters:

eps: specifies how close points should be to each other to be considered a part of a cluster. It means that if the distance between two points is lower or equal to this value (eps), these points are considered neighbors.

minPoints: the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region.

We start from eps =0.1 and increase it until I arrive to maximum 5% of the dataset to be considered as outliers. _eps_ is the maximum distance between two samples for them to be considered as in the same neighborhood. We also play with the _min_samples_ parameter which defines the minimum number of the samples in a cluster.

2.3.3 Feature Selection

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction Feature selection techniques are used for four reasons:

- Simplification of models to make them easier to Interpret by researchers.
- Shorter training times.
- To avoid the curse of dimensionality.
- Enhanced generalization by reducing overfitting.

3. Classification

We have used 2 Algorithms for classification:-

1) Logistic Regression

2) Decision Tree

3.1 Logistic Regression

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. Multinomial logistic regression can model scenarios where there are more than two possible discrete outcomes. Logistic regression is a useful analysis method for classification problems, where one is trying to determine if a new sample fits best into a category.

```
              precision    recall  f1-score   support

       Bumps       0.62      0.59      0.60        44
   Dirtiness       1.00      1.00      1.00         3
    K_Scatch       0.98      1.00      0.99        40
Other_Faults       0.62      0.77      0.69        47
      Pastry       0.71      0.42      0.53        12
      Stains       1.00      0.77      0.87        13
   Z_Scratch       0.74      0.67      0.70        21

    accuracy                           0.74       180
   macro avg       0.81      0.74      0.77       180
weighted avg       0.75      0.74      0.74       180

[[26  0  0 14  1  0  3]
 [ 0  3  0  0  0  0  0]
 [ 0  0 40  0  0  0  0]
 [ 9  0  0 36  1  0  1]
 [ 2  0  0  4  5  0  1]
 [ 1  0  0  2  0 10  0]
 [ 4  0  1  2  0  0 14]]
accuracy is 0.7444444444444445
```

3.2 Decision Tree

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.

It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

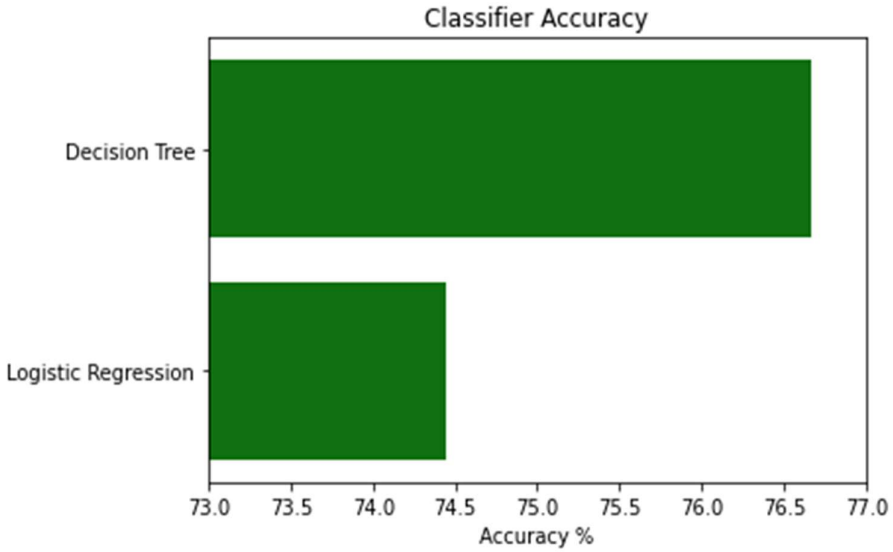
In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

3.3 Comparing the accuracy of both the classifiers used



4. Clustering

4.1 K-means Clustering:

The K-Means clustering algorithm is a simple unsupervised algorithm that's used for quickly predicting groupings. It groups the unlabelled dataset into different clusters.

It is an iterative algorithm that divides the unlabelled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabelled dataset on its own without the need for any training.

An important step in k-means algorithm is the initialisation of K and to choose an centroid which will result into optimal clusters . As we already know that we have 7 types of fault. Therefore, we can consider the number of clusters 'K' to be 7 .

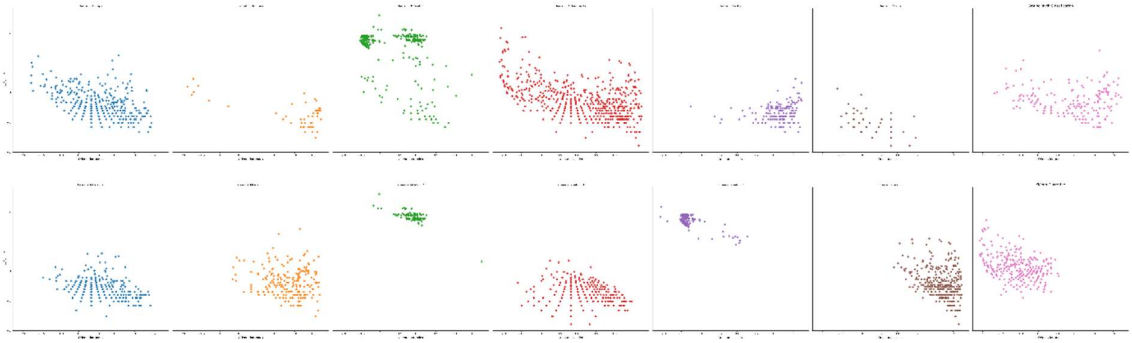
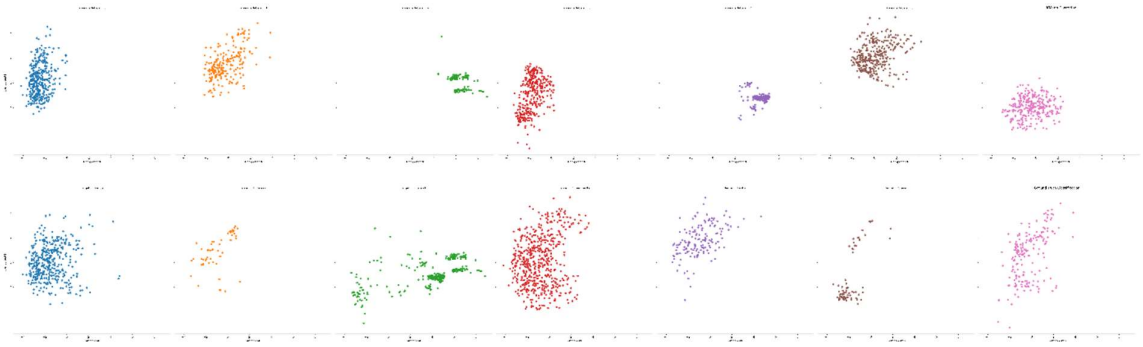


Fig . Plot of ground truth clusters versus k-means cluster obtained

The classification report obtained for k-means (K=7) is as follows :

	precision	recall	f1-score	support
0	0.53	0.47	0.50	396
1	0.00	0.02	0.01	54
2	0.99	0.29	0.45	373
3	0.37	0.20	0.26	627
4	0.00	0.00	0.00	153
5	0.00	0.00	0.00	72
6	0.16	0.26	0.20	188
accuracy			0.25	1863
macro avg	0.29	0.18	0.20	1863
weighted avg	0.45	0.25	0.30	1863

As the results show the precision and recall suffer, and K-Means is not able to cluster the data accurately. So we tried the same approach using the 15 PCA components that we selected:



Classification report with PCA-15 is as follows :

	precision	recall	f1-score	support
-1	0.00	0.00	0.00	76
0	0.47	0.53	0.50	396
1	0.01	0.06	0.02	54
2	0.93	0.28	0.43	373
3	0.32	0.15	0.21	551
4	0.00	0.00	0.00	153
5	0.03	0.10	0.04	72
6	0.57	0.81	0.67	188
accuracy			0.30	1863
macro avg	0.29	0.24	0.23	1863
weighted avg	0.44	0.30	0.32	1863

We can see that by using the 5 PCA components we arrive at the same results. Therefore, it can be considered to use less components would not impact the performance of the model negatively.

Classification report with PCA-5 is as follows :

	precision	recall	f1-score	support
-1	0.00	0.00	0.00	76
0	0.47	0.53	0.50	396
1	0.01	0.06	0.02	54
2	0.93	0.28	0.43	373
3	0.32	0.15	0.21	551
4	0.00	0.00	0.00	153
5	0.03	0.10	0.04	72
6	0.57	0.81	0.67	188
accuracy			0.30	1863
macro avg	0.29	0.24	0.23	1863
weighted avg	0.44	0.30	0.32	1863

4.2 Hierarchical Clustering:

Hierarchical clustering is an algorithm that groups similar objects into groups called *clusters*. Hierarchical clustering methods predict subgroups within data by finding the distance between each data point and its nearest neighbours.

The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

As we have irregular classes that are not normally distributed, this method might have an advantage to K-Means in producing more accurate clusters.

However, as the original shape of the distribution is important here and this model doesn't perform well on normal distributions, We used the original dataset without the transformations and scaling for this model.

We applied 2 approaches here for agglomerative approach.

1. Euclidean Distance
2. Manhattan Distance

Euclidean Distance:- the Euclidean distance between two points in Euclidean space is the length of a line segment between the two points. It can be calculated from the Cartesian coordinates of the points using the Pythagorean theorem, therefore occasionally being called the Pythagorean distance. The result from hierarchical clustering using Euclidean distance is shown below.

```
x=accuracy_score(origina_data['Target_Code'], h_clustering.labels_)
print("accuracy score with Euclidean distance is," ,x*100 )
```

```
accuracy score with Euclidean distance is, 22.20504894384338
```

Manhattan Distance:- Manhattan distance is a distance metric between two points in a N dimensional vector space. It is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. In simple terms, it is the sum of absolute difference between the measures in all dimensions of two points. The result from hierarchical clustering using Manhattan distance is shown below.

```
y=accuracy_score(origina_data['Target_Code'], h_clustering.labels_)
print("accuracy score with Manhattan distance is," ,y*100)
```

```
accuracy score with Manhattan distance is, 18.341061308603813
```

5. Code Snippets

- K-means Clustering :

[55]:
```
kmeans_model = KMeans(n_clusters=7, random_state=54)
kmeans_model.fit(features_scaled)
```

[55…
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=7, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=54, tol=0.0001, verbose=0)
```

[56]:
```
kmeans_labels = np.choose(kmeans_model.labels_, [0,1,2,3,4,5,6]).astype(np.int64)
data_scaled['kmeans_labels'] = kmeans_labels
```

[57]:
```
color_themes = {0:'#8d99ae',1:'#ffe066', 2:'#f77f00',3:'#348aa7',4:'#bce784',5:'#ffcc99',  6:'#f25f5c'}


sns.lmplot(x='Orientation_Index', y='Log_X_Index', data=data_scaled, fit_reg=False, hue='Target', col='Target', size=8)
plt.title("Ground Truth Classification")

sns.lmplot(x='Orientation_Index', y='Log_X_Index', data=data_scaled,  fit_reg=False, hue='kmeans_labels', col='kmeans_labels',size=8)
plt.title("KMean Clustering")
plt.savefig('save_as_a_png.png')
```

[58]:
```
print(classification_report(data_scaled['Target_Code'], kmeans_labels))
```

[60]:
```
kmeans_labels_pca15 = np.choose(kmeans_model.labels_, [0,1,2,3,4,5,6]).astype(np.int64)
data_pca15['kmeans_labels'] = kmeans_labels_pca15
```

[60]:
```
kmeans_labels_pca15 = np.choose(kmeans_model.labels_, [0,1,2,3,4,5,6]).astype(np.int64)
data_pca15['kmeans_labels'] = kmeans_labels_pca15
```

[61]:
```
sns.lmplot(x='Component0', y='Component1', data=data_pca15, fit_reg=False, hue='Target', col='Target', size=8)
plt.title("Ground Truth Classification")

sns.lmplot(x='Component0', y='Component1', data=data_pca15,  fit_reg=False, hue='kmeans_labels', col='kmeans_labels',size=8)
plt.title("KMean Clustering")
```

[63]:
```
kmeans_model_pca5 = KMeans(n_clusters=7, random_state=54)
kmeans_model_pca5.fit(data_pca5.drop(['Target','Target_Code'], axis=1))
```

[63…
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=7, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=54, tol=0.0001, verbose=0)
```

[64]:
```
print(classification_report(data_pca5['Target_Code'], kmeans_model_pca15.labels_))
```

- Hierarchical Clustering :

[65]:
```
original_features = origina_data.drop(['Target'], axis=1).copy()
origina_data['Target'] = pd.Categorical(origina_data['Target'])
origina_data['Target_Code'] = origina_data.Target.cat.codes
```

[66]:
```
linkage_model = linkage(original_features, method='ward')
dendrogram(linkage_model, truncate_mode='lastp', p=12, leaf_rotation=45, leaf_font_size=12, show_contracted=True)
plt.title('Truncated Hierarchical Clustering Dendrogram')
plt.xlabel('Cluster Size')
plt.ylabel('Distance')

plt.axhline(y=0.4*10**(8))
plt.axhline(y=0.2*10**(8))
```

[67]:
```
k = 7
h_clustering = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage='ward')
h_clustering.fit(original_features)

accuracy_score(origina_data['Target_Code'], h_clustering.labels_)
```

[67… 0.2220504894384338

[68]:
```
h_clustering = AgglomerativeClustering(n_clusters=k, affinity='manhattan', linkage='complete' )
h_clustering.fit(original_features)

accuracy_score(origina_data['Target_Code'], h_clustering.labels_)
```

[68… 0.09325090159711488

[69]:
```
h_clustering = AgglomerativeClustering(n_clusters=k, affinity='manhattan', linkage='average')
h_clustering.fit(original_features)

accuracy_score(origina_data['Target_Code'], h_clustering.labels_)
```

[69… 0.18341061308603812

[70]:
```
k = 7
h_clustering_pca5 = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage='ward' )
h_clustering_pca5.fit(data_pca5.drop(['Target','Target_Code'], axis=1))

accuracy_score(data_pca5['Target_Code'], h_clustering_pca5.labels_)
```

[70… 0.09750812567713976

- Logistic Regression

```
# LogisticRegression
from sklearn.linear_model import LogisticRegression
LogReg = LogisticRegression(max_iter=1000)
LogReg.fit(X_train, y_train)

y_pred_LogReg = LogReg.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred_LogReg))
print(confusion_matrix(y_test, y_pred_LogReg))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred_LogReg,y_test))
```

```
              precision    recall  f1-score   support

       Bumps       0.62      0.59      0.60        44
   Dirtiness       1.00      1.00      1.00         3
     K_Scatch      0.98      1.00      0.99        40
Other_Faults       0.62      0.77      0.69        47
      Pastry       0.71      0.42      0.53        12
      Stains       1.00      0.77      0.87        13
   Z_Scratch       0.74      0.67      0.70        21

    accuracy                           0.74       180
   macro avg       0.81      0.74      0.77       180
weighted avg       0.75      0.74      0.74       180

[[26  0  0 14  1  0  3]
 [ 0  3  0  0  0  0  0]
 [ 0  0 40  0  0  0  0]
 [ 9  0  0 36  1  0  1]
 [ 2  0  0  4  5  0  1]
 [ 1  0  0  2  0 10  0]
 [ 4  0  1  2  0  0 14]]
accuracy is 0.7444444444444445
```


```
# LogisticRegression
from sklearn.linear_model import LogisticRegression
LogReg = LogisticRegression(max_iter=1000)
LogReg.fit(X_train, y_train)

y_pred_LogReg = LogReg.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred_LogReg))
print(confusion_matrix(y_test, y_pred_LogReg))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred_LogReg,y_test))
```


```
              precision    recall  f1-score   support

       Bumps       0.62      0.59      0.60        44
   Dirtiness       1.00      1.00      1.00         3
     K_Scatch      0.98      1.00      0.99        40
Other_Faults       0.62      0.77      0.69        47
```

- Decision Tree :

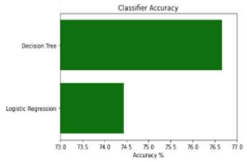
Predicted

```
from sklearn.tree import DecisionTreeClassifier

DTC = DecisionTreeClassifier()

DTC.fit(X_train, y_train)

y_pred_DTC = DTC.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred_DTC))
print(confusion_matrix(y_test, y_pred_DTC))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred_DTC,y_test))
```

```
              precision    recall  f1-score   support

       Bumps       0.67      0.66      0.67        44
   Dirtiness       1.00      0.67      0.80         3
     K_Scatch      0.93      1.00      0.96        40
Other_Faults       0.68      0.64      0.66        47
      Pastry       0.50      0.58      0.54        12
      Stains       0.85      0.85      0.85        13
   Z_Scratch       0.90      0.90      0.90        21

    accuracy                           0.77       180
   macro avg       0.79      0.76      0.77       180
weighted avg       0.77      0.77      0.77       180

[[29  0  0  9  3  1  2]
 [ 0  2  0  0  1  0  0]
 [ 0  0 40  0  0  0  0]
 [11  0  2 30  3  1  0]
 [ 1  0  0  4  7  0  0]
 [ 1  0  0  1  0 11  0]
 [ 1  0  1  0  0  0 19]]
accuracy is 0.7666666666666667
```

```
acc=[accuracy_score(y_pred_DTC,y_test),accuracy_score(y_pred_LogReg,y_test)]
models = ['Decision Tree','Logistic Regression']
acc_ = np.array(acc)*100
sns.barplot(acc_, models, color="g")
plt.xlim([73,77])
plt.xlabel('Accuracy %')
plt.title('Classifier Accuracy')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other
  FutureWarning
```




```
from sklearn.tree import DecisionTreeClassifier

DTC = DecisionTreeClassifier()

DTC.fit(X_train, y_train)

y_pred_DTC = DTC.predict(X_test)

# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred_DTC))
print(confusion_matrix(y_test, y_pred_DTC))
# Accuracy score
from sklearn.metrics import accuracy_score
print('accuracy is',accuracy_score(y_pred_DTC,y_test))
```

6. Conclusion

Data Mining techniques can be used and applied to steel manufacturing process which relies on monitoring strategies such as fault detection to reduce number of errors which can lead to huge losses. Investment in understanding how to apply data mining algorithms can be applied in order to help in fault diagnosis which can assist in accurate decision-making. We have applied several techniques like Data Preprocessing , DBSCAN for outlier detection , Logistic regression , K-means , Decision tree , Hierarchical Clustering , Apriori algorithm for frequent itemset generation etc. to mine knowledge from the dataset.

Future research can be to evaluate techniques for fault diagnostics in real time using predictive maintenance , such research can prove to be very essential as sensors record huge amounts of data and proper analysis can help us in better decision making.

7. References:

- Dataset : http://archive.ics.uci.edu/ml/datasets/steel+plates+faults by Semeion, Research Center of Sciences of Communication , Rome, Italy
- M Buscema, S Terzi, W Tastle, A New Meta-Classifier,in NAFIPS 2010, Toronto (CANADA),26-28 July 2010, IEEE
- M Buscema, MetaNet: The Theory of Independent Judges, in Substance Use & Misuse, 33(2), 439-461,1998
- https://en.wikipedia.org/wiki/Classification
- https://en.wikipedia.org/wiki/Clustering
- The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011