Quality Control Prediction of Steel Plates

Quality Control is an important step in every production system. A lot of business investments aim to reinforce this process in order to grant higher performance products. In last years Machine Learning solutions play a key role in this program of investments for their ability to easily adapt in every contest and for the great results achieved.

In this article, I present an AI solution for Quality Control in a standard production unit, in the form of a classification problem. Following a very interesting approach, I try to achieve the best possible performance, giving a visual explanation of the results and taking into account the **useful human insights**.

I want to underline this latest topic because **human insights** are often underestimated in Machine Learning! It's not a surprise that they permit us to achieve the best performances and to adopt the smartest solutions.

THE DATASET

I took the dataset for the analysis from the faithful UCI repository (Steel Plates Faults Data Set). The data description is very poor but it doesn't matter because it's easy to understand… We have a dataset containing the meta information of steel plates like luminosity, perimeter, edge, thickness, area, type of steel and so on (27 independent variables in total).

We can imagine to manage a factory that works steel and produces steel plates in the final step of the production system to sell them in the wholesale market. Our aim is to maximize the efficiency of the production system, trying to identify the possible types of steel plate faults (7 in total) only considering the metadata of the products. In this way, we will be able to identify the fallacies of the production system and accordingly react.

Steps followed:

Loading the dataset:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import itertools
import os
import seaborn as sns
from sklearn.metrics import  r2_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.manifold import TSNE
```

In [2]:
```
df1= pd.read_csv('Steel_Plates_Faults.csv')
```

In [3]:
```
df1.head()
```

Out[3]:

	X_Minimum	X_Maximum	Y_Minimum	Y_Maximum	Pixels_Areas	X_Perimeter	Y_Perimeter	Sum_of_Luminosity	Minimum_of_Luminosity	Maximum_of_Lumi
0	42	50	270900	270944	267	17	44	24220	76	
1	645	651	2538079	2538108	108	10	30	11397	84	
2	829	835	1553913	1553931	71	8	19	7972	99	
3	853	860	369370	369415	176	13	45	18996	99	
4	1289	1306	498078	498335	2409	60	260	246930	37	

Null Value Check : There is no any null value

In [5]:
```
df1.isna().sum()
```

Out[5]:
```
X_Minimum                0
X_Maximum                0
Y_Minimum                0
Y_Maximum                0
Pixels_Areas             0
X_Perimeter              0
Y_Perimeter              0
Sum_of_Luminosity        0
Minimum_of_Luminosity    0
Maximum_of_Luminosity    0
Length_of_Conveyer       0
TypeOfSteel_A300         0
TypeOfSteel_A400         0
Steel_Plate_Thickness    0
Edges_Index              0
Empty_Index              0
Square_Index             0
Outside_X_Index          0
Edges_X_Index            0
Edges_Y_Index            0
Outside_Global_Index     0
LogOfAreas               0
Log_X_Index              0
Log_Y_Index              0
Orientation_Index        0
Luminosity_Index         0
SigmoidOfAreas           0
Fault                    0
dtype: int64
```

Dataset shows little bit outlier, but it won't affect much.

In [7]:
```
df1.describe()
```

Out[7]:

	X_Minimum	X_Maximum	Y_Minimum	Y_Maximum	Pixels_Areas	X_Perimeter	Y_Perimeter	Sum_of_Luminosity	Minimum_of_Luminosity	Maximu
count	1941.000000	1941.000000	1.941000e+03	1.941000e+03	1941.000000	1941.000000	1941.000000	1.941000e+03	1941.000000	
mean	571.136012	617.964451	1.650685e+06	1.650739e+06	1893.878413	111.855229	82.965997	2.063121e+05	84.548686	
std	520.690671	497.627410	1.774578e+06	1.774590e+06	5168.459560	301.209187	426.482879	5.122936e+05	32.134276	
min	0.000000	4.000000	6.712000e+03	6.724000e+03	2.000000	2.000000	1.000000	2.500000e+02	0.000000	
25%	51.000000	192.000000	4.712530e+05	4.712810e+05	84.000000	15.000000	13.000000	9.522000e+03	63.000000	
50%	435.000000	467.000000	1.204128e+06	1.204136e+06	174.000000	26.000000	25.000000	1.920200e+04	90.000000	
75%	1053.000000	1072.000000	2.183073e+06	2.183084e+06	822.000000	84.000000	83.000000	8.301100e+04	106.000000	
max	1705.000000	1713.000000	1.298766e+07	1.298769e+07	152655.000000	10449.000000	18152.000000	1.159141e+07	203.000000	

8 rows × 25 columns

Dataset shows all independent variables are float and integer.

```
dtype: int64
```

In [6]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1941 entries, 0 to 1940
Data columns (total 28 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   X_Minimum              1941 non-null   int64  
 1   X_Maximum              1941 non-null   int64  
 2   Y_Minimum              1941 non-null   int64  
 3   Y_Maximum              1941 non-null   int64  
 4   Pixels_Areas           1941 non-null   int64  
 5   X_Perimeter            1941 non-null   int64  
 6   Y_Perimeter            1941 non-null   int64  
 7   Sum_of_Luminosity      1941 non-null   int64  
 8   Minimum_of_Luminosity  1941 non-null   int64  
 9   Maximum_of_Luminosity  1941 non-null   int64  
 10  Length_of_Conveyer     1941 non-null   int64  
 11  TypeOfSteel_A300       1941 non-null   bool   
 12  TypeOfSteel_A400       1941 non-null   bool   
 13  Steel_Plate_Thickness  1941 non-null   int64  
 14  Edges_Index            1941 non-null   float64
 15  Empty_Index            1941 non-null   float64
 16  Square_Index           1941 non-null   float64
 17  Outside_X_Index        1941 non-null   float64
 18  Edges_X_Index          1941 non-null   float64
 19  Edges_Y_Index          1941 non-null   float64
 20  Outside_Global_Index   1941 non-null   float64
 21  LogOfAreas             1941 non-null   float64
 22  Log_X_Index            1941 non-null   float64
 23  Log_Y_Index            1941 non-null   float64
 24  Orientation_Index      1941 non-null   float64
 25  Luminosity_Index       1941 non-null   float64
 26  SigmoidOfAreas         1941 non-null   float64
 27  Fault                  1941 non-null   object 
dtypes: bool(2), float64(13), int64(12), object(1)
memory usage: 398.2+ KB
```

Declaration of Independent (features) and dependent (target) variables.

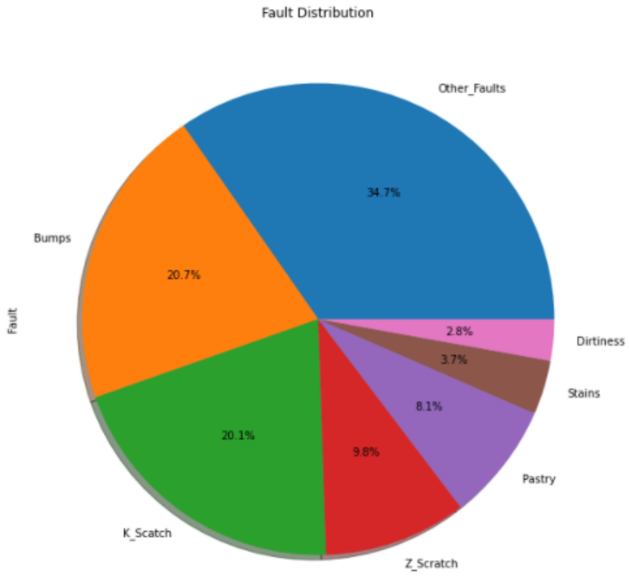
In [8]:
```
y = df1.Fault
x = df1.drop('Fault',axis=1)
```

Counting target variables based on features.
In dataset other_faults, k_scratch and Bumps shows the maximum contribution.

In [9]:
```
y.value_counts()
```

Out[9]:
```
Other_Faults    673
Bumps           402
K_Scatch        391
Z_Scratch       190
Pastry          158
Stains           72
Dirtiness        55
Name: Fault, dtype: int64
```

In [10]:
```
y.value_counts().plot.pie(figsize=(10,10),startangle = 0,shadow = True,autopct = '%1.1f%%')
plt.title ('Fault Distribution')
plt.show()
```



Declaring the Train and Test data set.

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y,test_size=0.2)
print("train data:", X_train.shape, "test data:", X_test.shape)
```

```
train data: (1552, 27) test data: (389, 27)
```

Normalizing the data set, and checking the how dataset is scattered.

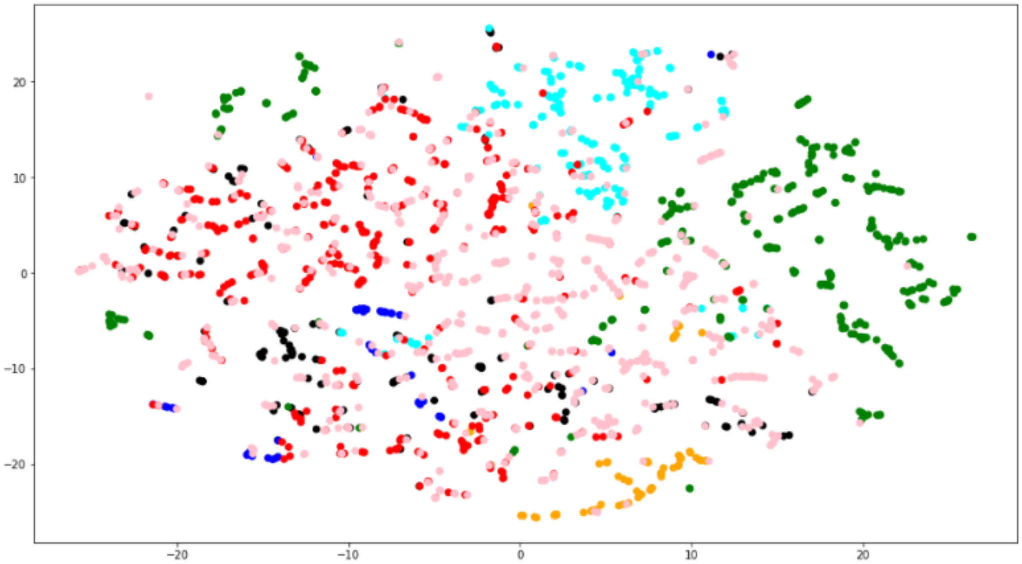
In [12]:
```
scaler = StandardScaler()
scaler.fit(X_train)

tsne = TSNE(n_components=2, random_state=42, n_iter=300, perplexity=5)
T = tsne.fit_transform(scaler.transform(x))
```

In [13]:
```
plt.figure(figsize=(16,9))

colors = {0:'red', 1:'blue', 2:'green', 3:'pink', 4:'black', 5:'orange', 6:'cyan'}
plt.scatter(T.T[0], T.T[1], c=[colors[i] for i in LabelEncoder().fit_transform(y)])
```

Out[13]: <matplotlib.collections.PathCollection at 0x16f867c2370>



MODEL CREATION:

Here I tried with Random forest classifier which shows around 96% of Accuracy.

In [21]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=26, criterion='entropy')
classifier.fit (X_train,y_train)
```

Out[21]: RandomForestClassifier(criterion='entropy', n_estimators=26)

In [22]:
```
y_pred= classifier.predict(x)
y_pred
```

Out[22]:
```
array(['Pastry', 'Pastry', 'Pastry', ..., 'Other_Faults', 'Other_Faults',
       'Other_Faults'], dtype=object)
```

In [23]:
```
from sklearn.metrics import accuracy_score
accuracy_score (y,y_pred)
```

Out[23]: 0.9546625450798557

Classification Report:

In [24]:
```
from sklearn.metrics import classification_report
print(classification_report (y,y_pred))
```

```
              precision    recall  f1-score   support

       Bumps       0.94      0.94      0.94       402
   Dirtiness       0.98      0.98      0.98        55
    K_Scatch       0.99      0.99      0.99       391
Other_Faults       0.93      0.95      0.94       673
      Pastry       0.95      0.89      0.92       158
      Stains       0.99      0.99      0.99        72
   Z_Scratch       0.96      0.96      0.96       190

    accuracy                           0.95      1941
   macro avg       0.96      0.96      0.96      1941
weighted avg       0.95      0.95      0.95      1941
```

Confusion Matrix:

In [25]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix (y,y_pred)
print (cm)
```

```
[[378   0   0  22   2   0   0]
 [  0  54   0   1   0   0   0]
 [  0   0 389   2   0   0   0]
 [ 20   1   1 638   5   1   7]
 [  4   0   0  14 140   0   0]
 [  0   0   0   1   0  71   0]
 [  1   0   1   5   0   0 183]]
```

In [27]:
```
plt.figure(figsize=(7,7))

cnf_matrix = confusion_matrix(y_test, classifier.predict(X_test))
plot_confusion_matrix(cnf_matrix, classes=np.unique(y), title="Confusion matrix")
```