Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #48

Artificial Intelligence for Data Science

Week-14:

MACHINE LEARNING (Part XVI)

Exploring Machine Learning Models for Steel Plate Fault Detection

Course Instructor:

Dr. Monidipa Das

Assistant Professor

Department of Computational and Data Sciences

Indian Institute of Science Education and Research Kolkata, India 741246

Steel-Plate-Fault-Detection

- **Steel-Plate-Fault-Detection Dataset** comprising of **samples having 7 different types of faults** (*Pastry, Z_Scratch, K_Scratch, Stains, Dirtiness, Bumps, Other_Faults*)
  0 1 2 3 4 5 6
- Each sample has **27 Input Features**:
  X_Minimum, X_Maximum, Y_Minimum, Y_Maximum, Pixels, Areas_X, Perimeter_Y, Perimeter, Sum_of_Luminosity, Minimum_of_Luminosity, Maximum_of_Luminosity, Length_of_Conveyer, TypeOfSteel_A300, TypeOfSteel_A400, Steel_Plate_Thickness, Edges_Index, Empty_Index, Square_Index, Outside_X_Index, Edges_X_Index, Edges_Y_Index, Outside_Global_Index, LogOfAreas, Log_X_Index, Log_Y_Index, Orientation_Index, Luminosity_Index, SigmoidOfAreas

	0	1	2	3	4	...	23	24	25	26	27
0	42	50	270900	270944	267	...	1.6435	0.8182	-0.2913	0.5822	0
1	645	651	2538079	2538108	108	...	1.4624	0.7931	-0.1756	0.2984	0
2	829	835	1553913	1553931	71	...	1.2553	0.6667	-0.1228	0.2150	0
3	853	860	369370	369415	176	...	1.6532	0.8444	-0.1568	0.5212	0
4	1289	1306	498078	498335	2409	...	2.4099	0.9338	-0.1992	1.0000	0

Target Variable

Steel-Plate-Fault-Detection

3

```python
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import classification_report,
confusion_matrix,accuracy_score

from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
```

Dr. Monidipa Das, Department of CDS, IISER Kolkata

Steel-Plate-Fault-Detection

4

```python
# Number of rows or instances and number of columns features
print("\nTotal number of data points/examples/instances:", dataset.shape[0])
print("Total number of input features:", dataset.shape[1]-1)

# Dataset description
print("\nDescription of the dataset:")
print(dataset.describe())

# Print number of classes
c=dataset[class_col].nunique()
print("\nNumber of classes
(discrete labels):", c)

#number of instances per class
print("\nSample count per class:")
print(dataset[class_col].value_counts())
```

Corresponds to Target Variable

```
Total number of data points/examples/instances: 1941
Total number of input features: 27
```

```
Description of the dataset:
                 0            1  ...           26           27
count  1941.000000  1941.000000  ...  1941.000000  1941.000000
mean    571.136012   617.964451  ...     0.585420     3.841319
std     520.690671   497.627410  ...     0.339452     2.144175
min       0.000000     4.000000  ...     0.119000     0.000000
25%      51.000000   192.000000  ...     0.248200     2.000000
50%     435.000000   467.000000  ...     0.506300     5.000000
75%    1053.000000  1072.000000  ...     0.999800     6.000000
max    1705.000000  1713.000000  ...     1.000000     6.000000
```

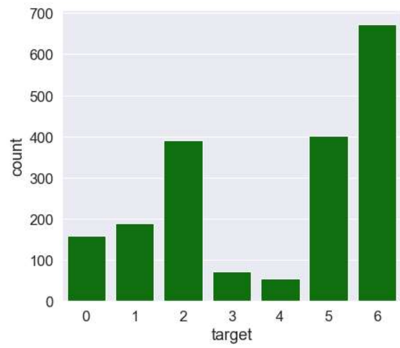
```
Number of classes (discrete labels): 7

Sample count per class:
6    673
5    402
2    391
1    190
0    158
3     72
4     55
```

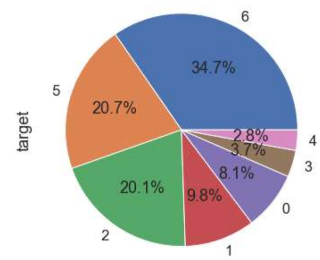
Dr. Monidipa Das, Department of CDS, IISER Kolkata

Steel-Plate-Fault-Detection

5

```
fig, ax = plt.subplots(1,2,figsize=(15,6))
sns.set(font_scale=1.5)
sns.countplot(x='target', data=dataset, color='green', ax=ax[0])
dataset['target'].value_counts().plot.pie(autopct="%1.1f%%", ax=ax[1])
plt.show()
```

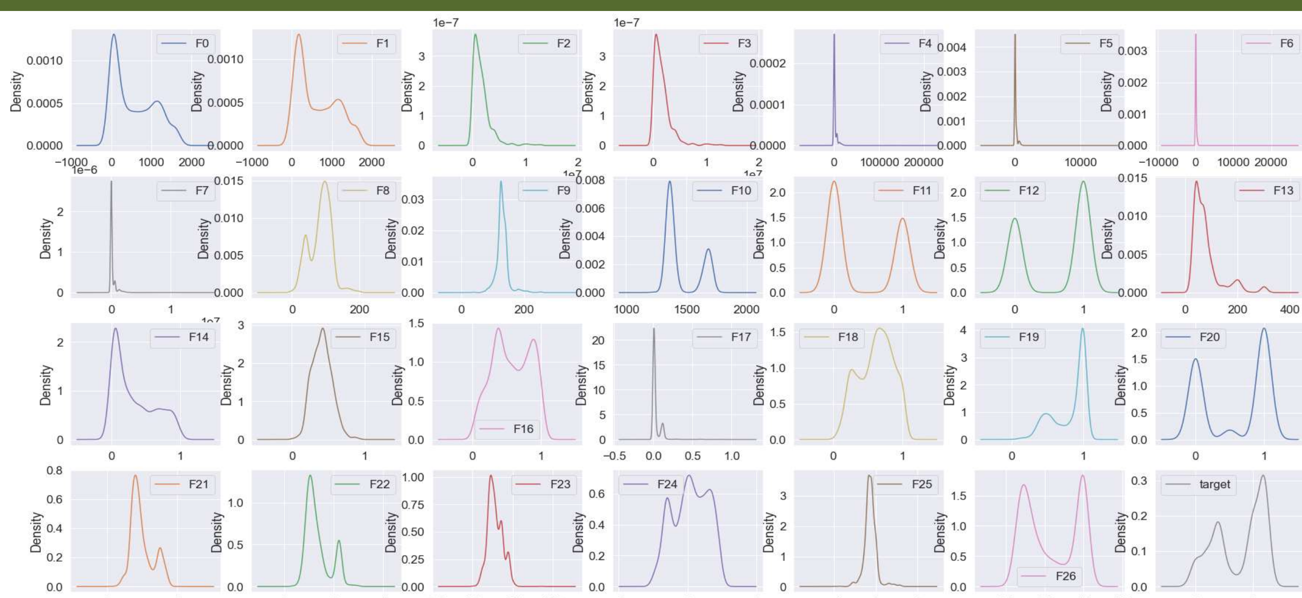





```
dataset.plot(kind="density", layout=(4,7),
             subplots=True,sharex=False, sharey=False, figsize=(30,15))
plt.show()
```


Steel-Plate-Fault-Detection

6



Dr. Monidipa Das, Department of CDS, IISER Kolkata

Steel-Plate-Fault-Detection

7

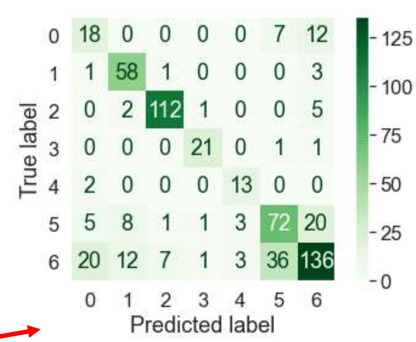
```
X = dataset[input_features]
y = dataset[class_col]

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X=scaler.fit_transform(X)

#split dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.3, random_state = 42)

#check the shape of X_train and X_test
print("Traning set shape:",X_train.shape,"\nTest set
shape:",X_test.shape)
```

```
Traning set shape: (1358, 27)
Test set shape: (583, 27)
```


KNN Model

8

```
# Instantiate learning model (k = 3)
classifier = KNeighborsClassifier(n_neighbors=3)

# Fitting the model
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

#Evaluating predictions
cm = confusion_matrix(y_test, y_pred)
print(cm)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

accuracy = accuracy_score(y_test, y_pred)*100
print('Accuracy of our model is equal ' + str(round(accuracy, 2)) + ' %.')
```

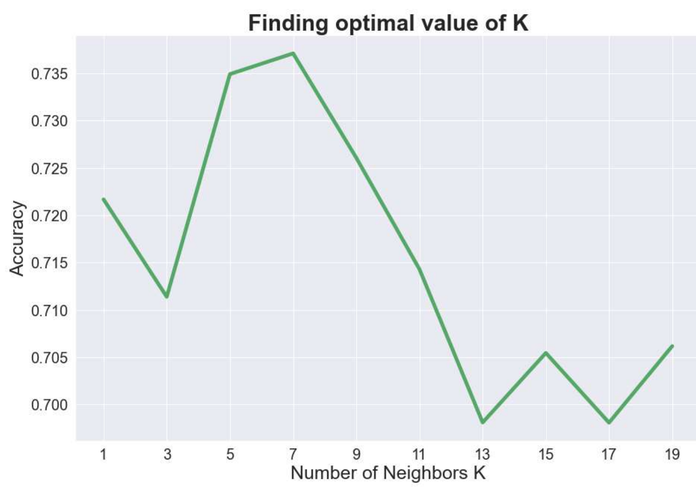



```
Accuracy of our model is equal 73.76 %.
```

KNN Model

```python
from sklearn.model_selection import cross_val_score
# creating list of K for KNN
k_list = list(range(1,20,2))
# creating list of cv scores
cv_scores = []

# perform 10-fold cross validation
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

plt.figure(figsize=(15,10))
plt.title('Finding optimal value of K', fontsize=30, fontweight='bold')
plt.xlabel('Number of Neighbors K', fontsize=25)
plt.ylabel('Accuracy', fontsize=25)
plt.plot(k_list, cv_scores,linewidth=5,color='g')
plt.xticks(k_list,fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```


KNN Model: Finding Optimal K

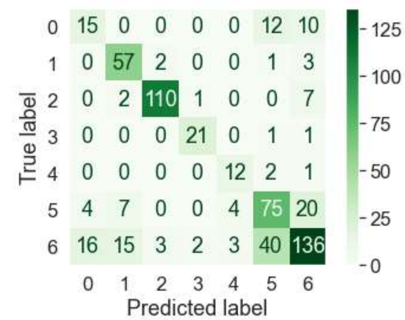



```python
best=k_list[cv_scores.index(max(cv_scores))]
print("Best value of K:",best)
```

```
Best value of K: 7
```

KNN Model

```
# Instantiate learning model (k = best)
classifier = KNeighborsClassifier(n_neighbors=best)

# Fitting the model
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

#Evaluating predictions
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

accuracy = accuracy_score(y_test, y_pred)*100
print('Accuracy of KNN model is ' + str(round(accuracy, 2)) + ' %.')
```




```
Accuracy of KNN model is 73.07 %.
```


KNN Model

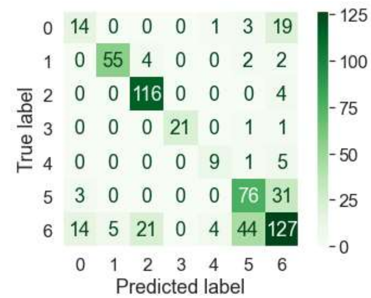
```
# classification report
report = classification_report(y_test, y_pred)
print("\nClassification Report:\n", report)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.43      0.41      0.42        37
           1       0.70      0.90      0.79        63
           2       0.96      0.92      0.94       120
           3       0.88      0.91      0.89        23
           4       0.63      0.80      0.71        15
           5       0.57      0.68      0.62       110
           6       0.76      0.63      0.69       215

    accuracy                           0.73       583
   macro avg       0.70      0.75      0.72       583
weighted avg       0.74      0.73      0.73       583
```

Decision Tree Model

13

```
#Instantiate the DecisionTreeClassifier model with criterion entropy
clf_ent = DecisionTreeClassifier(criterion='entropy', max_depth=6, random_state=0)

#Fit the model
clf_ent.fit(X_train, y_train)
y_pred_ent = clf_ent.predict(X_test)

#Evaluating predictions
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred_ent)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

#Print scores on training and test set
print('DT Training set score: {:.4f}'.format(clf_ent.score(X_train, y_train)))
print('DT Test set score: {:.4f}'.format(clf_ent.score(X_test, y_test)))
```




```
DT Training set score: 0.7290
DT Test set score: 0.7170
```


Decision Tree Model

14

```
# A comprehensive classification report
report = classification_report(y_test, y_pred_ent)
print("\nDecision Tree Classification Report:\n", report)
```

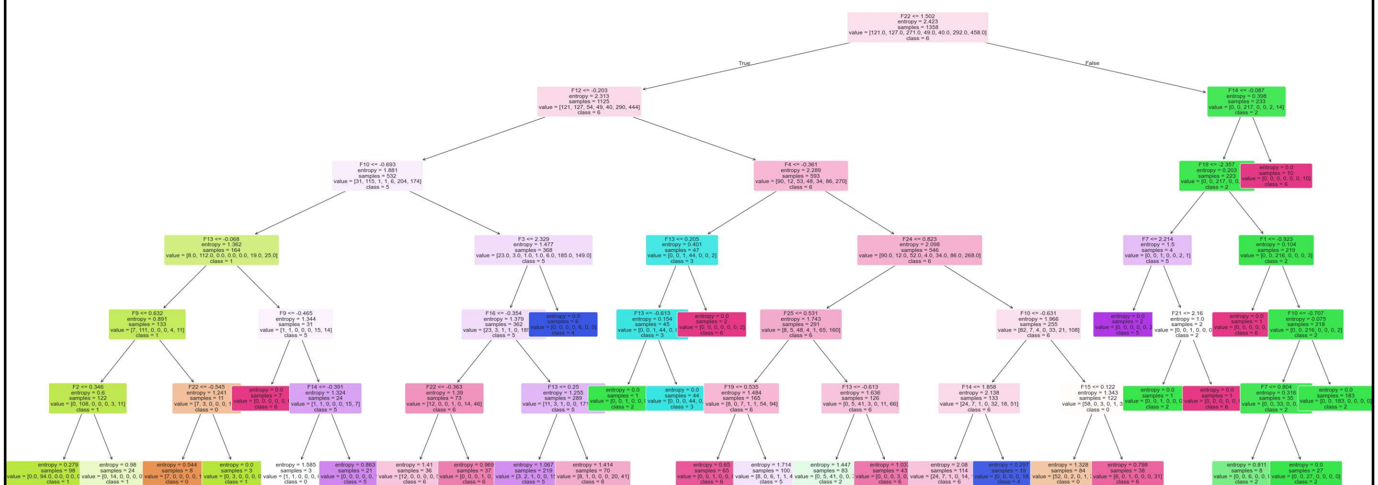
With appropriate parameter settings of DecisioTreeClassifier() you may please try to attain better generalization

Homework

```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.45      0.38      0.41        37
           1       0.92      0.87      0.89        63
           2       0.82      0.97      0.89       120
           3       1.00      0.91      0.95        23
           4       0.64      0.60      0.62        15
           5       0.60      0.69      0.64       110
           6       0.67      0.59      0.63       215

    accuracy                           0.72       583
   macro avg       0.73      0.72      0.72       583
weighted avg       0.71      0.72      0.71       583
```

Decision Tree Plot

15



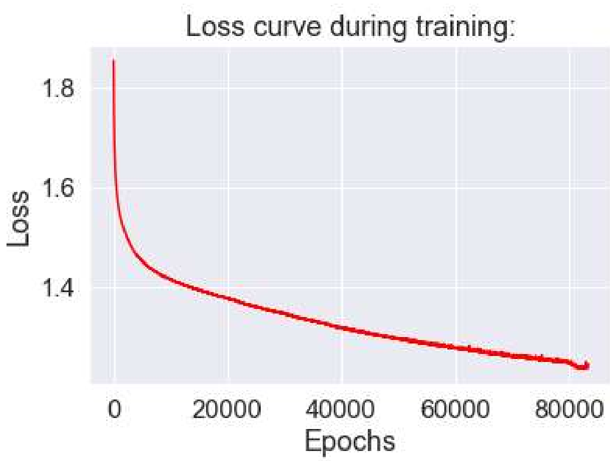
NN Model for Steel Plate Fault Detection

16

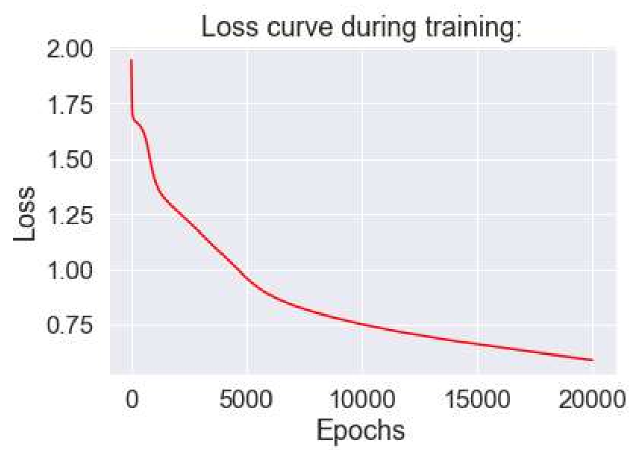
```
mlp =MLPClassifier(hidden_layer_sizes = (50,20),
activation = 'logistic', solver = 'sgd',alpha = 0,learning_rate = 'constant',
learning_rate_init=0.0005,max_iter = 20000,random_state = 42,tol = 0.0000001,
n_iter_no_change = 1000,verbose=True)

#Fit the training data
mlp.fit(X_train, y_train)
#Print("Loss curve during training:")
plt.figure()
plt.plot(range(1,len(mlp.loss_curve_)+1),mlp.loss_curve_,color='red')
plt.grid(True)
plt.title("Loss curve during training:")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.show()

#Prediction
predictions_test = mlp.predict(X_test)
predictions_train = mlp.predict(X_train)
```

NN Model: Loss Curve

17



Without Standardization



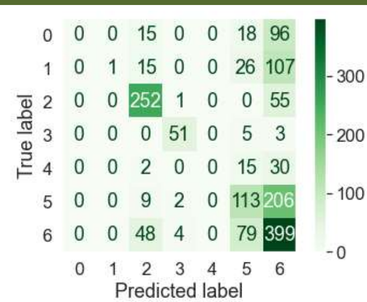
With Standardization

NN Model: Confusion Matrix

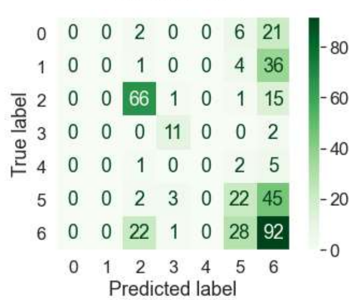
18

Without Standardization

Training Data

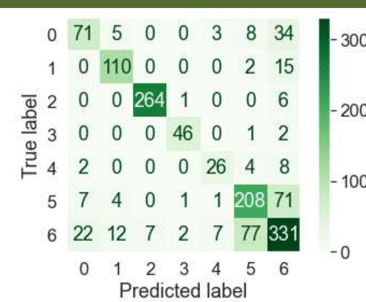


Test Data

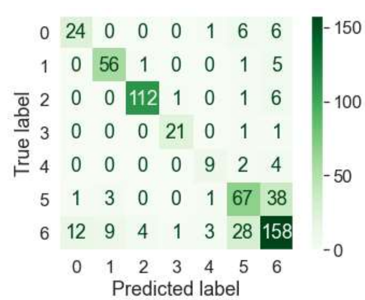


With Standardization

Training Data



Test Data



NN Model: Classification Report

19

Homework

With appropriate parameter settings of MLPClassifier() you may please try to attain better generalization

For Training Data

```
Classification Report for Training:
              precision    recall  f1-score   support

           0       0.70      0.59      0.64       121
           1       0.84      0.87      0.85       127
           2       0.97      0.97      0.97       271
           3       0.92      0.94      0.93        49
           4       0.70      0.65      0.68        40
           5       0.69      0.71      0.70       292
           6       0.71      0.72      0.72       458

    accuracy                           0.78      1358
   macro avg       0.79      0.78      0.78      1358
weighted avg       0.78      0.78      0.78      1358
```

For Test Data

```
Classification Report for Testing:
              precision    recall  f1-score   support

           0       0.65      0.65      0.65        37
           1       0.82      0.89      0.85        63
           2       0.96      0.93      0.95       120
           3       0.91      0.91      0.91        23
           4       0.64      0.60      0.62        15
           5       0.63      0.61      0.62       110
           6       0.72      0.73      0.73       215

    accuracy                           0.77       583
   macro avg       0.76      0.76      0.76       583
weighted avg       0.77      0.77      0.77       583
```

With Standardization

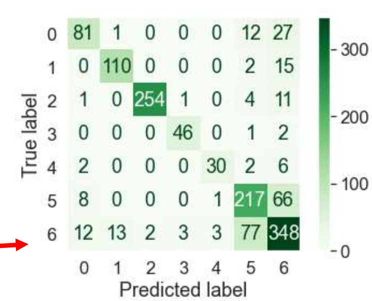
Dr. Monidipa Das, Department of CDS, IISER Kolkata

SVM Model for Steel Plate Fault Detection

20

```
classifier = SVC(C=1,kernel='rbf',gamma='scale')
classifier.fit(X_train, y_train)

# Now predict the value of the test
predictions_test = classifier.predict(X_test)
predictions_train = classifier.predict(X_train)
print('Training set score:')
print(accuracy_score(y_train,predictions_train))
print('Test set score:')
print(accuracy_score(y_test,predictions_test))

cm_train=confusion_matrix(y_train,predictions_train)
cm_pred=confusion_matrix(y_test,predictions_test)
plt.figure()
disp = ConfusionMatrixDisplay(confusion_matrix=cm_train,
display_labels=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()
```

```
Training set score:
0.7997054491899853
Test set score:
0.7615780445969125
```

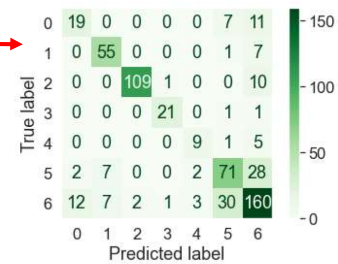


Dr. Monidipa Das, Department of CDS, IISER Kolkata

SVM Model for Steel Plate Fault Detection

21

```
plt.figure()
disp =
ConfusionMatrixDisplay(confusion_matrix=cm_pred,display_lab
els=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

from sklearn.metrics import classification_report
print("Classification Report for Training:")
print(classification_report(y_train, predictions_train))


from sklearn.metrics import classification_report
print("Classification Report for Testing:")
print(classification_report(y_test, predictions_test))
```



SVM Model for Steel Plate Fault Detection

22

For Training Data

```
Classification Report for Training:
              precision    recall  f1-score   support

           0       0.78      0.67      0.72       121
           1       0.89      0.87      0.88       127
           2       0.99      0.94      0.96       271
           3       0.92      0.94      0.93        49
           4       0.88      0.75      0.81        40
           5       0.69      0.74      0.71       292
           6       0.73      0.76      0.75       458

    accuracy                           0.80      1358
   macro avg       0.84      0.81      0.82      1358
weighted avg       0.80      0.80      0.80      1358
```

For Test Data

```
Classification Report for Testing:
              precision    recall  f1-score   support

           0       0.58      0.51      0.54        37
           1       0.80      0.87      0.83        63
           2       0.98      0.91      0.94       120
           3       0.91      0.91      0.91        23
           4       0.64      0.60      0.62        15
           5       0.64      0.65      0.64       110
           6       0.72      0.74      0.73       215

    accuracy                           0.76       583
   macro avg       0.75      0.74      0.75       583
weighted avg       0.76      0.76      0.76       583
```

With appropriate parameter settings of SVC() you may please try to attain better generalization

Homework

NB Model for Steel Plate Fault Detection

23

```
NB=GaussianNB()

#train usinge train data
NB.fit(X_train,y_train)

from sklearn.metrics import
confusion_matrix,ConfusionMatrixDisplay,classification_report,accuracy_score
import numpy as np

#predict target data for test feature data and save into pred variable
predicted=NB.predict(X_test)
print("Test Accuracy={:.4f}".format(accuracy_score(y_test, predicted)))
predicted_train=NB.predict(X_train)
print("Train Accuracy={:.4f}".format(accuracy_score(y_train, predicted_train)))
```

```
Test Accuracy=0.6123
Train Accuracy=0.6097
```


NB Model for Steel Plate Fault Detection

24

```
print("Classification report Training:\n",classification_report(y_train,
predicted_train))
cm_pred = confusion_matrix(y_train, predicted_train)
plt.figure()
disp = ConfusionMatrixDisplay(confusion_matrix=cm_pred,
display_labels=np.unique(y_train))
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

print("Classification report Test:\n",classification_report(y_test, predicted))
cm_pred = confusion_matrix(y_test, predicted)
plt.figure()
disp = ConfusionMatrixDisplay(confusion_matrix=cm_pred,
display_labels=np.unique(y_test))
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()
```

NB Model for Steel Plate Fault Detection

25

For Training Data

```
Classification report Training:
              precision    recall  f1-score   support

           0       0.38      0.69      0.49       121
           1       0.81      0.81      0.81       127
           2       0.93      0.87      0.90       271
           3       0.85      0.96      0.90        49
           4       0.31      0.78      0.45        40
           5       0.49      0.74      0.59       292
           6       0.67      0.24      0.35       458

    accuracy                           0.61      1358
   macro avg       0.64      0.73      0.64      1358
weighted avg       0.67      0.61      0.59      1358
```

For Test Data

```
Classification report Test:
              precision    recall  f1-score   support

           0       0.33      0.70      0.45        37
           1       0.80      0.84      0.82        63
           2       0.90      0.86      0.88       120
           3       0.91      0.91      0.91        23
           4       0.30      0.60      0.40        15
           5       0.43      0.72      0.54       110
           6       0.75      0.31      0.44       215

    accuracy                           0.61       583
   macro avg       0.63      0.71      0.63       583
weighted avg       0.70      0.61      0.61       583
```

With appropriate parameter settings of GaussianNB() you may please try to attain better generalization

Homework

26

Questions?