In [1528]:
```
import csv
import numpy as np
import pandas as pd
import math
import random as rd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import tree
from sklearn import svm
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import BayesianRidge
from sklearn.linear_model import LinearRegression
# all the sklearn library packages are searched in google and related mod
# immitated by the samples before implemented
```

executed in 16ms, finished 15:37:21 2022-04-27

In [231]:
```
totalTrainList=pd.read_csv("/Users/jiazhidai/Downloads/python3/student_pe
totalTestList=pd.read_csv("/Users/jiazhidai/Downloads/python3/student_per
```

executed in 25ms, finished 17:31:14 2022-04-23

In [173]:
```
totalTrainList
```
executed in 29ms, finished 16:19:17 2022-04-23

Out[173]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	free
0	MS	M	17	R	GT3	T	1	1	other	services	...	4	
1	MS	F	17	U	LE3	A	3	2	services	other	...	1	
2	MS	F	18	U	GT3	T	1	1	other	other	...	3	
3	MS	M	16	U	LE3	A	2	2	other	services	...	4	
4	MS	F	17	U	GT3	T	2	2	other	at_home	...	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	
481	GP	M	17	U	LE3	T	4	4	services	other	...	5	
482	MS	F	17	R	GT3	T	1	1	at_home	at_home	...	3	
483	MS	M	18	U	LE3	T	1	2	at_home	services	...	4	
484	MS	M	17	U	GT3	T	1	1	other	other	...	4	
485	GP	F	15	U	GT3	T	1	1	other	services	...	4	

486 rows × 33 columns

In [328]:
```
totalTestList
```
executed in 32ms, finished 18:11:27 2022-04-24

Out[328]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	free
0	GP	F	17	U	GT3	T	1	1	at_home	other	...	4	
1	GP	F	18	U	LE3	T	1	1	other	other	...	4	
2	GP	M	18	U	GT3	T	2	2	other	other	...	5	
3	GP	F	17	U	GT3	T	4	3	health	services	...	4	
4	GP	F	15	U	LE3	A	4	3	other	other	...	5	
...	...	...	...	...	...	...	...	...	...	...	...	...	
158	GP	F	16	U	GT3	T	4	2	health	services	...	4	
159	GP	F	17	U	GT3	T	3	2	other	other	...	4	
160	GP	F	17	U	LE3	T	4	2	teacher	services	...	4	
161	GP	M	15	U	LE3	T	2	2	services	services	...	5	
162	GP	F	17	U	GT3	T	2	3	at_home	other	...	3	

163 rows × 33 columns

1 Classification

1.1 Trivial System

In [374]:
```
#shuffle all the training data and create validate data
def shuffle_data(totalTrainList):
    idx_list=list(range(0,len(totalTrainList)))
    shuffled_idx=rd.sample(idx_list,len(idx_list))
    training_idx=shuffled_idx[0:400]
    validate_idx=shuffled_idx[400:]

    shuffled_trainingList=pd.DataFrame(columns=totalTrainList.columns)
    shuffled_validateList=pd.DataFrame(columns=totalTrainList.columns)
    for idx in training_idx:
        shuffled_trainingList=shuffled_trainingList.append(totalTrainList
    for idx in validate_idx:
        shuffled_validateList=shuffled_validateList.append(totalTrainList
    return shuffled_trainingList,shuffled_validateList
```

executed in 16ms, finished 18:44:35 2022-04-24

In [1380]:
```
# generate trival P
seq=['A','B','C','D','F']
def P_trivial_generate(target_g):
    trainList, validateList=shuffle_data(totalTrainList)
    G_A=trainList[trainList[target_g]>=16]
    G_B=trainList[(trainList[target_g]>=14) & (trainList[target_g]<16)]
    G_C=trainList[(trainList[target_g]>=12) & (trainList[target_g]<14)]
    G_D=trainList[(trainList[target_g]>=10) & (trainList[target_g]<12)]
    G_F=trainList[trainList[target_g]<10]
    P_A_trivial=len(G_A)/len(trainList)
    P_B_trivial=len(G_B)/len(trainList)
    P_C_trivial=len(G_C)/len(trainList)
    P_D_trivial=len(G_D)/len(trainList)
    P_F_trivial=len(G_F)/len(trainList)
#     seq=['A','B','C','D','F']
    P_trivial_G=[P_A_trivial, P_B_trivial, P_C_trivial, P_D_trivial, P_F_
    return P_trivial_G, validateList
```

executed in 14ms, finished 14:49:40 2022-04-27

In [1381]:
```
# a function to generate random classes
def generate_random_class(sequence, probability):
    random_loc = rd.uniform(0, 1)
    total_probability = 0.0
    for class_i, class_i_probability in zip(sequence, probability):
        total_probability += class_i_probability
        if random_loc < total_probability:
            break
    return class_i
# this def is from the internet
#https://blog.csdn.net/liulicuican/article/details/102502540
```

executed in 3ms, finished 14:49:49 2022-04-27

In [1382]:
```
# test trivial
def trivial_accurate(sequence, probability, target_g, pred_g, testlist):
    predict_P_trivial=[]
    testlist_wP=[]
    for i in range(0,len(testlist)):
        random_P=''
        random_P=generate_random_class(sequence, probability)
        predict_P_trivial.append(random_P)

    testlist_wP=testlist.copy()
    testlist_wP[pred_g]=predict_P_trivial
    testlist_wP.reset_index(drop=True, inplace=True)

    num_correct=0
    for i in range(0,len(testlist)):
        row=testlist_wP[i:i+1]
        if row.at[i,target_g]>=16 and row.at[i,pred_g]=='A':
            num_correct+=1
        elif row.at[i, target_g]>=14 and row.at[i,target_g]<16 and row.at
            num_correct+=1
        elif row.at[i,target_g]>=12 and row.at[i,target_g]<14 and row.at[
            num_correct+=1
        elif row.at[i,target_g]>=10 and row.at[i,target_g]<12 and row.at[
            num_correct+=1
        elif row.at[i,target_g]<10 and row.at[i,pred_g]=='F':
            num_correct+=1
    accurate=num_correct/len(testlist)
    return accurate
```

executed in 4ms, finished 14:49:58 2022-04-27

In [1383]:
```
def calculate_accurate(sequence, probability, target_g, pred_g, testlist)
    mean_trivial_accurate=0
    sum_accurate=0
    for i in range(0,10):
        sum_accurate+=trivial_accurate(sequence, probability, target_g, p
    mean_trivial_accurate=sum_accurate/10
    return mean_trivial_accurate
```

executed in 3ms, finished 14:50:07 2022-04-27

mission 1

In [1384]:
```
trivial_best_G1_P=[]
max_P=0
for i in range(0,10):
    P_trivial_G1, validateList=P_trivial_generate('G1')
    temp_P = calculate_accurate(seq,P_trivial_G1,'G1','pred_G1',validateL
    if temp_P > max_P:
        trivial_best_G1_P = P_trivial_G1
        max_P = temp_P
calculate_accurate(seq, trivial_best_G1_P, 'G1', 'pred_G1', totalTestList
```

executed in 5.28s, finished 14:50:22 2022-04-27

Out[1384]: 0.22024539877300614

In [1396]:
```
predict_P_trivial=[]
testlist_wP=[]
for i in range(0,len(totalTestList)):
    random_P=''
    random_P=generate_random_class(seq, trivial_best_G1_P)
    predict_P_trivial.append(random_P)

testlist_wP=totalTestList.copy()
testlist_wP['pred_G1']=predict_P_trivial

cate_G1=[]
for i in testlist_wP['G1']:
    if i>=16:
        cate_G1.append('A')
    elif i<16 and i>=14:
        cate_G1.append('B')
    elif i<14 and i>=12:
        cate_G1.append('C')
    elif i<12 and i>=10:
        cate_G1.append('D')
    else:
        cate_G1.append('F')
```

executed in 20ms, finished 15:00:26 2022-04-27

In [1393]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(cate_G1, predict_P_trivial, average='macro')
```

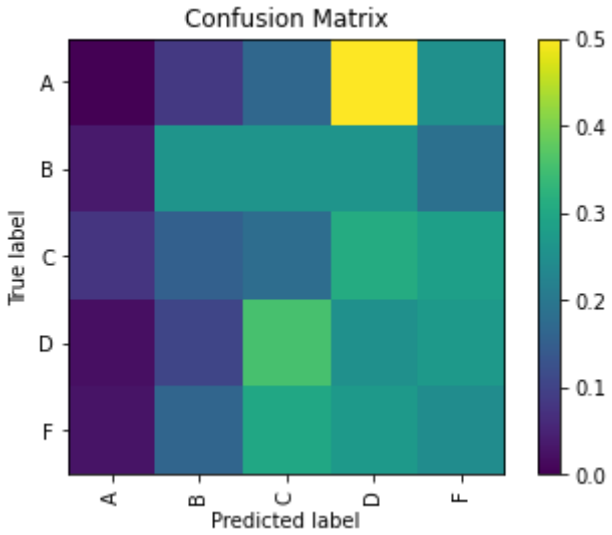
executed in 14ms, finished 14:58:39 2022-04-27

Out[1393]: 0.18426125554850983

In [1394]:
```
cm=confusion_matrix(cate_G1, predict_P_trivial, labels=["A", "B", "C","D"
labels_name=["A", "B", "C", "D", "F"]
cm
```
executed in 18ms, finished 14:59:39 2022-04-27

Out[1394]:
```
array([[ 0,  1,  2,  6,  3],
       [ 1,  7,  7,  7,  5],
       [ 3,  6,  7, 12, 11],
       [ 1,  5, 17, 12, 13],
       [ 1,  6, 11, 10,  9]])
```

In [1395]:
```
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```
executed in 109ms, finished 14:59:59 2022-04-27



mission 2 and 3

In [258]:
```
trivial_best_G3_P=[]
max_P=0
for i in range(0,10):
    P_trivial_G3, validateList=P_trivial_generate('G3')
    temp_P = calculate_accurate(seq, P_trivial_G3,'G3','pred_G3',validatel
    if temp_P > max_P:
        trivial_best_G3_P = P_trivial_G3
        max_P = temp_P
calculate_accurate(seq, trivial_best_G3_P, 'G3', 'pred_G3', totalTestList
```
executed in 5.27s, finished 16:46:10 2022-04-24

Out[258]:
```
0.2202453987730061
```

In [1405]:
```
predict_P_trivial3=[]
testlist_wP=[]
for i in range(0,len(totalTestList)):
    random_P=''
    random_P=generate_random_class(seq, trivial_best_G3_P)
    predict_P_trivial3.append(random_P)

testlist_wP=totalTestList.copy()
testlist_wP['pred_G3']=predict_P_trivial3

cate_G3=[]
for i in testlist_wP['G3']:
    if i>=16:
        cate_G3.append('A')
    elif i<16 and i>=14:
        cate_G3.append('B')
    elif i<14 and i>=12:
        cate_G3.append('C')
    elif i<12 and i>=10:
        cate_G3.append('D')
    else:
        cate_G3.append('F')
```

executed in 16ms, finished 15:04:05 2022-04-27

In [1409]:
```
f1_score(cate_G3, predict_P_trivial3, average='macro')
```

executed in 5ms, finished 15:04:45 2022-04-27

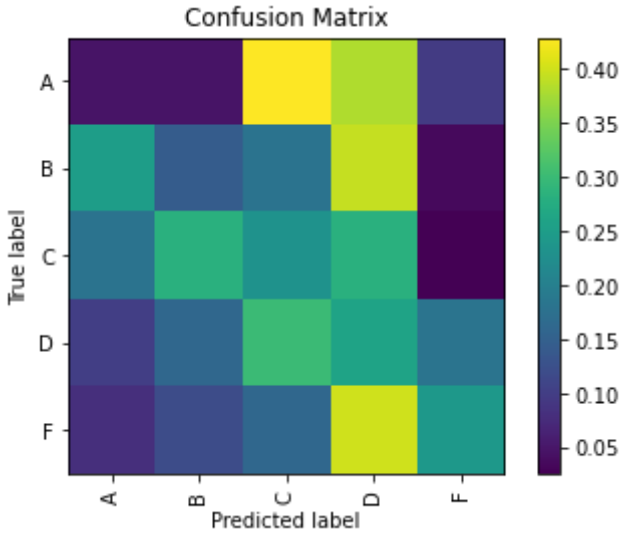
Out[1409]:
```
0.18786857055540732
```

In [1411]:
```
cm=confusion_matrix(cate_G3, predict_P_trivial3, labels=["A", "B", "C","D
labels_name=["A", "B", "C", "D", "F"]
cm
```

executed in 19ms, finished 15:05:01 2022-04-27

Out[1411]:
```
array([[ 1,  1,  9,  8,  2],
       [ 7,  4,  5, 11,  1],
       [ 7, 11,  9, 11,  1],
       [ 5,  8, 15, 13,  9],
       [ 2,  3,  4, 10,  6]])
```

In [1412]:
```
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```
executed in 79ms, finished 15:05:10 2022-04-27



1.2 Baseline System

In [1413]:
```
# preprocessing data : turning all the non-binary and binary categorial d
totalTrainList_wo4=totalTrainList.drop(['Mjob','Fjob','reason','guardian'
totalTrainList_wo4_wog123=totalTrainList_wo4.drop(['G1','G2','G3'],axis=1
totalTrainList_wo4_wog3=totalTrainList_wo4.drop(['G3'],axis=1)

totalTestList_wo4=totalTestList.drop(['Mjob','Fjob','reason','guardian'],
totalTestList_wo4_wog123=totalTestList_wo4.drop(['G1','G2','G3'],axis=1)
totalTestList_wo4_wog3=totalTestList_wo4.drop(['G3'],axis=1)

processed_test=pd.get_dummies(totalTestList_wo4_wog123)
processed_test_wo3=pd.get_dummies(totalTestList_wo4_wog3)
```
executed in 32ms, finished 15:05:40 2022-04-27

In [1414]:
```
true_P=[]
for i in range(0,len(totalTestList_wo4)):
    row=totalTestList_wo4[i:i+1]
    if row.at[i,'G1']>=16:
        true_P.append('A')
    elif row.at[i,'G1']>=14 and row.at[i,'G1']<16:
        true_P.append('B')
    elif row.at[i,'G1']>=12 and row.at[i,'G1']<14:
        true_P.append('C')
    elif row.at[i,'G1']>=10 and row.at[i,'G1']<12:
        true_P.append('D')
    elif row.at[i,'G1']<10:
        true_P.append('F')
```
executed in 15ms, finished 15:05:50 2022-04-27

In [1415]:
```
processed_list_A=pd.get_dummies(totalTrainList_wo4_wog123)[totalTrainList
processed_list_B=pd.get_dummies(totalTrainList_wo4_wog123)[(totalTrainLis
processed_list_C=pd.get_dummies(totalTrainList_wo4_wog123)[(totalTrainLis
processed_list_D=pd.get_dummies(totalTrainList_wo4_wog123)[(totalTrainLis
processed_list_F=pd.get_dummies(totalTrainList_wo4_wog123)[totalTrainList

list_A_mean=processed_list_A.mean()
list_B_mean=processed_list_B.mean()
list_C_mean=processed_list_C.mean()
list_D_mean=processed_list_D.mean()
list_F_mean=processed_list_F.mean()

processed_test=pd.get_dummies(totalTestList_wo4_wog123)
```
executed in 35ms, finished 15:05:59 2022-04-27

In [1416]:
```
pred_P=[]
for i in range(0,len(processed_test)):
    row = processed_test[i:i+1]
    distA = np.linalg.norm(row-list_A_mean)
    distB = np.linalg.norm(row-list_B_mean)
    distC = np.linalg.norm(row-list_C_mean)
    distD = np.linalg.norm(row-list_D_mean)
    distF = np.linalg.norm(row-list_F_mean)
    min_dist=min(distA,distB,distC,distD,distF)
    if min_dist==distA:
        pred_P.append('A')
    elif min_dist==distB:
        pred_P.append('B')
    elif min_dist==distC:
        pred_P.append('C')
    elif min_dist==distD:
        pred_P.append('D')
    else:
        pred_P.append('F')
```
executed in 344ms, finished 15:06:09 2022-04-27

mission 1

In [532]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P[i]==pred_P[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```

executed in 12ms, finished 21:55:57 2022-04-24

Out[532]: 0.25153374233128833

In [1417]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P, pred_P, average='macro')
```

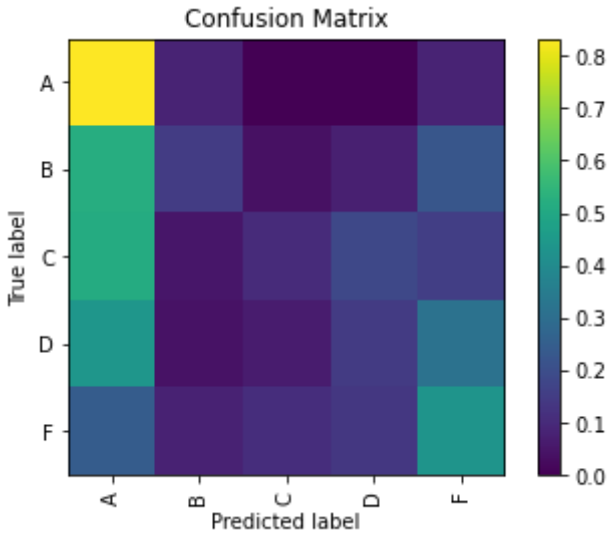
executed in 15ms, finished 15:06:34 2022-04-27

Out[1417]: 0.2385018737761655

In [1418]:
```
cm=confusion_matrix(true_P, pred_P, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 114ms, finished 15:07:12 2022-04-27

```
[[10  1  0  0  1]
 [14  4  1  2  6]
 [20  2  4  7  6]
 [21  2  3  7 15]
 [ 9  3  4  5 16]]
```



In [1420]:
```
true_P3=[]
for i in range(0,len(totalTestList_wo4)):
    row=totalTestList_wo4[i:i+1]
    if row.at[i,'G3']>=16:
        true_P3.append('A')
    elif row.at[i,'G3']>=14 and row.at[i,'G3']<16:
        true_P3.append('B')
    elif row.at[i,'G3']>=12 and row.at[i,'G3']<14:
        true_P3.append('C')
    elif row.at[i,'G3']>=10 and row.at[i,'G3']<12:
        true_P3.append('D')
    elif row.at[i,'G3']<10:
        true_P3.append('F')
```
executed in 15ms, finished 15:07:43 2022-04-27

In [1421]:
```
sed_list_A3=pd.get_dummies(totalTrainList_wo4_wog123)[totalTrainList_wo4['G3
sed_list_B3=pd.get_dummies(totalTrainList_wo4_wog123)[(totalTrainList_wo4['G
sed_list_C3=pd.get_dummies(totalTrainList_wo4_wog123)[(totalTrainList_wo4['G
sed_list_D3=pd.get_dummies(totalTrainList_wo4_wog123)[(totalTrainList_wo4['G
sed_list_F3=pd.get_dummies(totalTrainList_wo4_wog123)[totalTrainList_wo4['G3

_mean3=processed_list_A3.mean()
_mean3=processed_list_B3.mean()
_mean3=processed_list_C3.mean()
_mean3=processed_list_D3.mean()
_mean3=processed_list_F3.mean()

sed_test=pd.get_dummies(totalTestList_wo4_wog123)
```
executed in 39ms, finished 15:07:53 2022-04-27

In [1422]:
```
pred_P3=[]
for i in range(0,len(processed_test)):
    row = processed_test[i:i+1]
    distA = np.linalg.norm(row-list_A_mean3)
    distB = np.linalg.norm(row-list_B_mean3)
    distC = np.linalg.norm(row-list_C_mean3)
    distD = np.linalg.norm(row-list_D_mean3)
    distF = np.linalg.norm(row-list_F_mean3)
    min_dist=min(distA,distB,distC,distD,distF)
    if min_dist==distA:
        pred_P3.append('A')
    elif min_dist==distB:
        pred_P3.append('B')
    elif min_dist==distC:
        pred_P3.append('C')
    elif min_dist==distD:
        pred_P3.append('D')
    else:
        pred_P3.append('F')
```
executed in 344ms, finished 15:08:03 2022-04-27

mission 2

In [1423]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P3[i]==pred_P3[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```

executed in 3ms, finished 15:08:12 2022-04-27

Out[1423]: 0.26993865030674846

In [1424]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P3, pred_P3, average='macro')
```

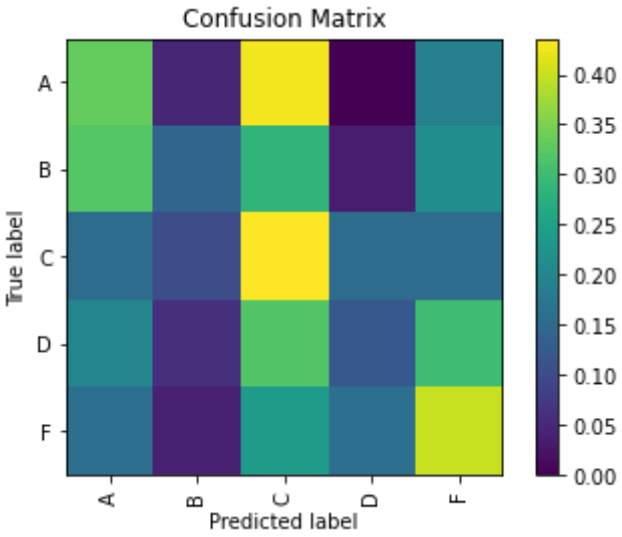
executed in 4ms, finished 15:08:21 2022-04-27

Out[1424]: 0.25615310075831604

In [1425]:
```
cm=confusion_matrix(true_P3, pred_P3, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 90ms, finished 15:08:31 2022-04-27

```
[[ 7  1  9  0  4]
 [ 9  4  8  1  6]
 [ 6  4 17  6  6]
 [10  3 16  6 15]
 [ 4  1  6  4 10]]
```



In [1426]:
```
true_P4=[]
for i in range(0,len(totalTestList_wo4)):
    row=totalTestList_wo4[i:i+1]
    if row.at[i,'G3']>=16:
        true_P4.append('A')
    elif row.at[i, 'G3']>=14 and row.at[i,'G3']<16:
        true_P4.append('B')
    elif row.at[i,'G3']>=12 and row.at[i,'G3']<14:
        true_P4.append('C')
    elif row.at[i,'G3']>=10 and row.at[i,'G3']<12:
        true_P4.append('D')
    elif row.at[i,'G3']<10:
        true_P4.append('F')
```
executed in 13ms, finished 15:08:40 2022-04-27

In [1427]:
```
processed_list_A=pd.get_dummies(totalTrainList_wo4_wog3)[totalTrainList_w
processed_list_B=pd.get_dummies(totalTrainList_wo4_wog3)[(totalTrainList_
processed_list_C=pd.get_dummies(totalTrainList_wo4_wog3)[(totalTrainList_
processed_list_D=pd.get_dummies(totalTrainList_wo4_wog3)[(totalTrainList_
processed_list_F=pd.get_dummies(totalTrainList_wo4_wog3)[totalTrainList_w

list_A_mean4=processed_list_A.mean()
list_B_mean4=processed_list_B.mean()
list_C_mean4=processed_list_C.mean()
list_D_mean4=processed_list_D.mean()
list_F_mean4=processed_list_F.mean()

processed_test=pd.get_dummies(totalTestList_wo4_wog3)
```
executed in 34ms, finished 15:08:50 2022-04-27

In [1428]:
```
pred_P4=[]
for i in range(0,len(processed_test)):
    row = processed_test[i:i+1]
    distA = np.linalg.norm(row-list_A_mean4)
    distB = np.linalg.norm(row-list_B_mean4)
    distC = np.linalg.norm(row-list_C_mean4)
    distD = np.linalg.norm(row-list_D_mean4)
    distF = np.linalg.norm(row-list_F_mean4)mission 1
    min_dist=min(distA,distB,distC,distD,distF)
    if min_dist==distA:
        pred_P4.append('A')
    elif min_dist==distB:
        pred_P4.append('B')
    elif min_dist==distC:
        pred_P4.append('C')
    elif min_dist==distD:
        pred_P4.append('D')
    else:
        pred_P4.append('F')
```
executed in 355ms, finished 15:08:59 2022-04-27

mission 3

In [1429]:
```
num_acc4=0
for i in range(0,len(totalTestList)):
    if true_P4[i]==pred_P4[i]:
        num_acc4+=1
total_acc_rate4=num_acc4/len(totalTestList)
total_acc_rate4
```

executed in 3ms, finished 15:09:09 2022-04-27

Out[1429]: 0.6012269938650306

In [1430]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P4, pred_P4, average='macro')
```

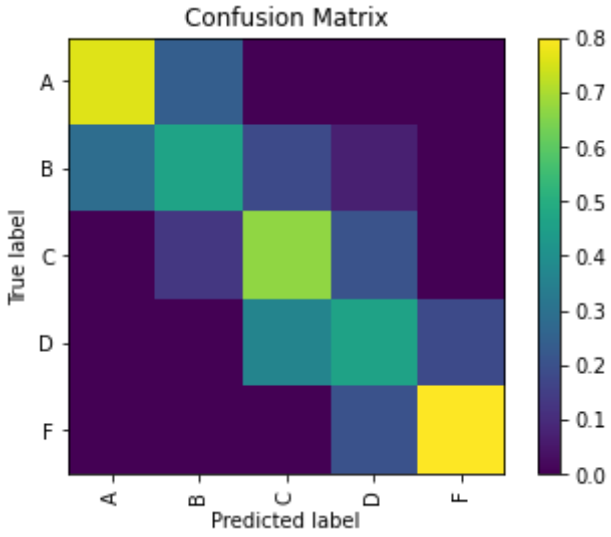
executed in 4ms, finished 15:09:18 2022-04-27

Out[1430]: 0.6150584274113686

In [1431]:
```
cm=confusion_matrix(true_P4, pred_P4, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 89ms, finished 15:09:28 2022-04-27

```
[[16  5  0  0  0]
 [ 8 13  5  2  0]
 [ 0  5 26  8  0]
 [ 0  0 18 23  9]
 [ 0  0  0  5 20]]
```



preprocessing

In [1432]:
```
def normalize(df):
    result = df.copy()
    for feature_name in df.columns:
        max_value = df[feature_name].max()
        min_value = df[feature_name].min()
        if feature_name == 'G1' or feature_name == 'G2' or feature_name ==
            result[feature_name] = df[feature_name]
#         else:
        elif max_value > 0 and max_value != min_value:
#             max_value = df[feature_name].max()
#             min_value = df[feature_name].min()
            result[feature_name] = (df[feature_name] - min_value) / (max_
        elif max_value == min_value:
            result[feature_name] = max_value
    return result
```
executed in 15ms, finished 15:10:34 2022-04-27

In [1433]:
```
totalTrainList_wog123=totalTrainList.drop(['G1','G2','G3'],axis=1)
totalTrainList_wog123_processed=pd.get_dummies(totalTrainList_wog123)
x = totalTrainList_wog123_processed.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
totalTrainList_wog123_processed = pd.DataFrame(x_scaled,columns=totalTrai

totalTrainList_wclass=totalTrainList.copy()

totalTestList_wog123=totalTestList.drop(['G1','G2','G3'],axis=1)
totalTestList_wog123_processed=pd.get_dummies(totalTestList_wog123)
x = totalTestList_wog123_processed.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
totalTestList_wog123_processed = pd.DataFrame(x_scaled,columns=totalTestL


totalTestList_wog3=totalTestList.drop(['G3'],axis=1)
totalTestList_wog3_processed=pd.get_dummies(totalTestList_wog3)
totalTestList_wog3_processed=normalize(totalTestList_wog3_processed)


totalTrainList_processed=pd.get_dummies(totalTrainList)
totalTrainList_processed=normalize(totalTrainList_processed)
```
executed in 55ms, finished 15:10:44 2022-04-27

1.3 SVM

In [1674]:
```
def create_shuffle(target_g):
    shuffle_train,shuffle_validate=shuffle_data(totalTrainList_processed)
    shuffle_train.reset_index(drop=True)
    train_class=[]
    for i in range(0,len(shuffle_train)):

        if shuffle_train.iloc[i][target_g]>=16:
            train_class.append('A')
        elif shuffle_train.iloc[i][target_g]>=14 and shuffle_train.iloc[i
            train_class.append('B')
        elif shuffle_train.iloc[i][target_g]>=12 and shuffle_train.iloc[i
            train_class.append('C')
        elif shuffle_train.iloc[i][target_g]>=10 and shuffle_train.iloc[i
            train_class.append('D')
        elif shuffle_train.iloc[i][target_g]<10:
            train_class.append('F')
    shuffle_train=shuffle_train.drop(['G1','G2','G3'],axis=1)
    shuffle_train_wclass=shuffle_train.copy()
    shuffle_train_wclass['class_g']=train_class

    shuffle_validate.reset_index(drop=True)
    validate_class=[]
    for i in range(0,len(shuffle_validate)):

        if shuffle_validate.iloc[i][target_g]>=16:
            validate_class.append('A')
        elif shuffle_validate.iloc[i][target_g]>=14 and shuffle_validate.
            validate_class.append('B')
        elif shuffle_validate.iloc[i][target_g]>=12 and shuffle_validate.
            validate_class.append('C')
        elif shuffle_validate.iloc[i][target_g]>=10 and shuffle_validate.
            validate_class.append('D')
        elif shuffle_validate.iloc[i][target_g]<10:
            validate_class.append('F')
    shuffle_validate=shuffle_validate.drop(['G1','G2','G3'],axis=1)
    shuffle_validate_wclass=shuffle_validate.copy()
    shuffle_validate_wclass['class_g']=validate_class
    return shuffle_train,shuffle_train_wclass,shuffle_validate,validate_c
```

executed in 18ms, finished 17:07:14 2022-04-27

In [1675]:
```
# sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto',
# coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200
# verbose=False, max_iter=-1, decision_function_shape=None, random_state=

def SVM(train,train_wclass,test,class_g):
    svm_X=train
    svm_Y=train_wclass[class_g]
    clf = svm.SVC()
    clf.fit(svm_X, svm_Y)
    result = clf.predict(test)
    return result
```

executed in 2ms, finished 17:07:24 2022-04-27

In [1676]:
```
classes_g1=[]
for i in range(0,len(totalTrainList_wo4)):
    row=totalTrainList_wo4[i:i+1]
    if row.at[i,'G1']>=16:
        classes_g1.append('A')
    elif row.at[i,'G1']>=14 and row.at[i,'G1']<16:
        classes_g1.append('B')
    elif row.at[i,'G1']>=12 and row.at[i,'G1']<14:
        classes_g1.append('C')
    elif row.at[i,'G1']>=10 and row.at[i,'G1']<12:
        classes_g1.append('D')
    elif row.at[i,'G1']<10:
        classes_g1.append('F')
```
executed in 33ms, finished 17:07:33 2022-04-27

In [1677]:
```
classes_g3=[]
for i in range(0,len(totalTrainList_wo4)):
    row=totalTrainList_wo4[i:i+1]
    if row.at[i,'G3']>=16:
        classes_g3.append('A')
    elif row.at[i,'G3']>=14 and row.at[i,'G3']<16:
        classes_g3.append('B')
    elif row.at[i,'G3']>=12 and row.at[i,'G3']<14:
        classes_g3.append('C')
    elif row.at[i,'G3']>=10 and row.at[i,'G3']<12:
        classes_g3.append('D')
    elif row.at[i,'G3']<10:
        classes_g3.append('F')
```
executed in 33ms, finished 17:07:43 2022-04-27

In [1678]:
```
max_acc_rate=0

for i in range(10):
    shuffle_train,shuffle_train_wclass,shuffle_validate,validate_class=cr
    result=SVM(shuffle_train,shuffle_train_wclass,shuffle_validate,'class
    num_acc=0
    for i in range(0,len(validate_class)):
        if validate_class[i]==result[i]:
            num_acc+=1
    total_acc_rate=num_acc/len(validate_class)
    if total_acc_rate>max_acc_rate:
        max_acc_rate=total_acc_rate
        final_train=shuffle_train
        final_train_wclass=shuffle_train_wclass
```
executed in 11.7s, finished 17:08:04 2022-04-27

In [1682]:
```
svm_X=final_train
svm_Y=final_train_wclass['class_g']
clf = svm.SVC()
clf.fit(svm_X, svm_Y)
result = clf.predict(totalTestList_wog123_processed)
```
executed in 45ms, finished 17:12:59 2022-04-27

mission 1

In [1683]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P[i]==result[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```

executed in 3ms, finished 17:13:09 2022-04-27

Out[1683]: 0.34355828220858897

In [1684]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P, result, average='macro')
```

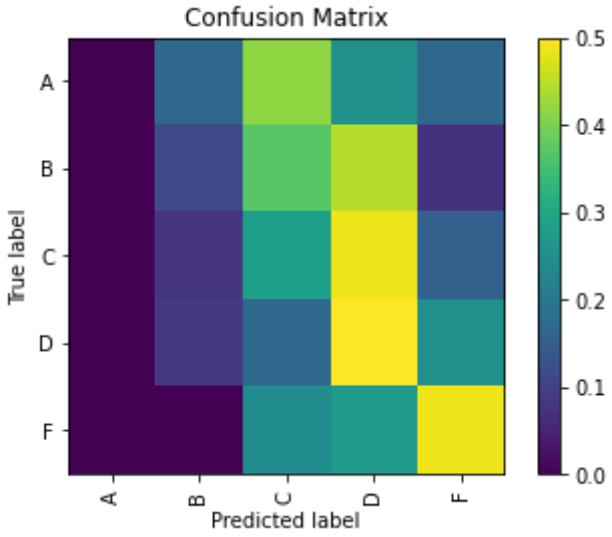
executed in 4ms, finished 17:13:19 2022-04-27

Out[1684]: 0.2606928815507453

In [1685]:
```
cm=confusion_matrix(true_P, result, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 79ms, finished 17:13:28 2022-04-27

```
[[ 0  2  5  3  2]
 [ 0  3 10 12  2]
 [ 0  3 11 19  6]
 [ 0  4  8 24 12]
 [ 0  0  9 10 18]]
```



In [1686]:
```
max_acc_rate=0

for i in range(10):
    shuffle_train,shuffle_train_wclass,shuffle_validate,validate_class=cr
    result=SVM(shuffle_train,shuffle_train_wclass,shuffle_validate,'class
    num_acc=0
    for i in range(0,len(validate_class)):
        if validate_class[i]==result[i]:
            num_acc+=1
    total_acc_rate=num_acc/len(validate_class)
    if total_acc_rate>max_acc_rate:
        max_acc_rate=total_acc_rate
        final_train=shuffle_train
        final_train_wclass=shuffle_train_wclass
```
executed in 11.7s, finished 17:13:50 2022-04-27

In [1444]:
```
svm_X=final_train
svm_Y=final_train_wclass['class_g']
clf = svm.SVC()
clf.fit(svm_X, svm_Y)
result = clf.predict(totalTestList_wog123_processed)
```
executed in 32ms, finished 15:12:53 2022-04-27

mission 2

In [1445]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P3[i]==result[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 4ms, finished 15:13:02 2022-04-27

Out[1445]: 0.34355828220858897

In [1446]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P, result, average='macro')
```
executed in 4ms, finished 15:13:12 2022-04-27

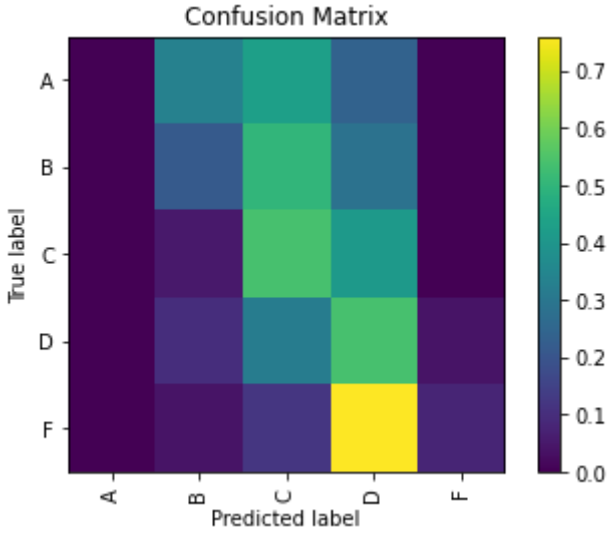
Out[1446]: 0.20857245337159253

In [1447]:

```
cm=confusion_matrix(true_P3, result, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 85ms, finished 15:13:22 2022-04-27

```
[[ 0  7  9  5  0]
 [ 0  6 14  8  0]
 [ 0  2 21 16  0]
 [ 0  5 16 27  2]
 [ 0  1  3 19  2]]
```



In [883]:

```
def create_shuffle2(target_g):
    shuffle_train,shuffle_validate=shuffle_data(totalTrainList_processed)
    shuffle_train.reset_index(drop=True)
    train_class=[]
    for i in range(0,len(shuffle_train)):

        if shuffle_train.iloc[i][target_g]>=16:
            train_class.append('A')
        elif shuffle_train.iloc[i][target_g]>=14 and shuffle_train.iloc[i
            train_class.append('B')
        elif shuffle_train.iloc[i][target_g]>=12 and shuffle_train.iloc[i
            train_class.append('C')
        elif shuffle_train.iloc[i][target_g]>=10 and shuffle_train.iloc[i
            train_class.append('D')
        elif shuffle_train.iloc[i][target_g]<10:
            train_class.append('F')
    shuffle_train=shuffle_train.drop(['G3'],axis=1)
    shuffle_train_wclass=shuffle_train.copy()
    shuffle_train_wclass['class_g']=train_class
    shuffle_train=normalize(shuffle_train)


    shuffle_validate.reset_index(drop=True)
    validate_class=[]
    for i in range(0,len(shuffle_validate)):

        if shuffle_validate.iloc[i][target_g]>=16:
            validate_class.append('A')
        elif shuffle_validate.iloc[i][target_g]>=14 and shuffle_validate.
            validate_class.append('B')
        elif shuffle_validate.iloc[i][target_g]>=12 and shuffle_validate.
            validate_class.append('C')
        elif shuffle_validate.iloc[i][target_g]>=10 and shuffle_validate.
            validate_class.append('D')
        elif shuffle_validate.iloc[i][target_g]<10:
            validate_class.append('F')
    shuffle_validate=shuffle_validate.drop(['G3'],axis=1)
    shuffle_validate_wclass=shuffle_validate.copy()
    shuffle_validate_wclass['class_g']=validate_class
    shuffle_validate=normalize(shuffle_validate)
    return shuffle_train,shuffle_train_wclass,shuffle_validate,validate_c
```

executed in 17ms, finished 13:59:37 2022-04-25

In [884]:
```
max_acc_rate=0

for i in range(10):
    shuffle_train,shuffle_train_wclass,shuffle_validate,validate_class=cr

    result=SVM(shuffle_train,shuffle_train_wclass,shuffle_validate,'class
    num_acc=0
    for i in range(0,len(validate_class)):
        if validate_class[i]==result[i]:
            num_acc+=1
    total_acc_rate=num_acc/len(validate_class)
    if total_acc_rate>max_acc_rate:
        max_acc_rate=total_acc_rate
        final_train=shuffle_train
        final_train_wclass=shuffle_train_wclass
```
executed in 9.01s, finished 13:59:49 2022-04-25

In [885]:
```
svm_X=final_train
svm_Y=final_train_wclass['class_g']
clf = svm.SVC()
clf.fit(svm_X, svm_Y)
result = clf.predict(totalTestList_wog3_processed)
```
executed in 22ms, finished 13:59:52 2022-04-25

mission 3

In [886]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P4[i]==result[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 13:59:55 2022-04-25

Out[886]:
```
0.7177914110429447
```

In [887]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P4, result, average='macro')
```
executed in 3ms, finished 13:59:58 2022-04-25

Out[887]:
```
0.7157410260800091
```

In [888]:
```
# confusion_matrix(y_true, y_pred)
cm=confusion_matrix(true_P4, result, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
cm
```
executed in 4ms, finished 14:00:01 2022-04-25

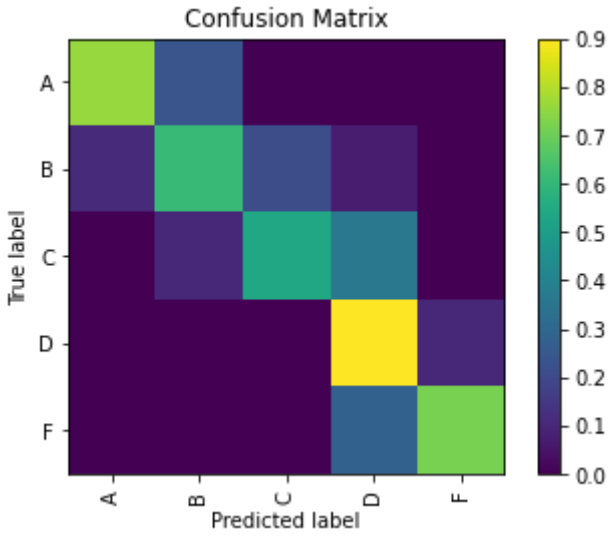
Out[888]:
```
array([[16,  5,  0,  0,  0],
       [ 3, 17,  6,  2,  0],
       [ 0,  4, 21, 14,  0],
       [ 0,  0,  0, 45,  5],
       [ 0,  0,  0,  7, 18]])
```

In [889]:
```
def plot_confusion_matrix(cm, labels_name, title):
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    plt.imshow(cm, interpolation='nearest')
    plt.title(title)
    plt.colorbar()
    num_local = np.array(range(len(labels_name)))
    plt.xticks(num_local, labels_name, rotation=90)
    plt.yticks(num_local, labels_name)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

plot_confusion_matrix(cm, labels_name, "Confusion Matrix")

plt.show()
```

executed in 83ms, finished 14:00:04 2022-04-25



1.4 kNN

In [1454]:
```
def kNN_Model_1(target_g,neighbors):
    max_acc_rate=0
    for i in range(10):
        shuffle_train,shuffle_train_wclass,shuffle_validate,validate_clas

        knn_model = KNeighborsClassifier(n_neighbors=neighbors)
        knn_model.fit(shuffle_train, shuffle_train_wclass['class_g'])
        test_preds = knn_model.predict(shuffle_validate)
        num_acc=0
        for i in range(0,len(validate_class)):
            if validate_class[i]==test_preds[i]:
                num_acc+=1
        total_acc_rate=num_acc/len(validate_class)
        if total_acc_rate>max_acc_rate:
            max_acc_rate=total_acc_rate
            final_train=shuffle_train
            final_train_wclass=shuffle_train_wclass
    return final_train,final_train_wclass
```
executed in 10ms, finished 15:15:11 2022-04-27

In [1455]:
```
final_train,final_train_wclass=kNN_Model_1('G1',5)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(shuffle_train, shuffle_train_wclass['class_g'])
test_preds = knn_model.predict(totalTestList_wog123_processed)
```
executed in 11.6s, finished 15:15:32 2022-04-27

mission 1

In [1456]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P[i]==test_preds[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:15:42 2022-04-27

Out[1456]: 0.25153374233128833

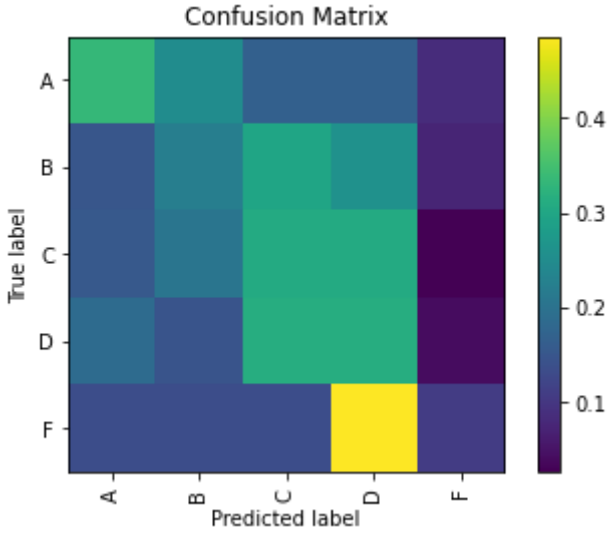
In [1457]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P, test_preds, average='macro')
```
executed in 4ms, finished 15:15:52 2022-04-27

Out[1457]: 0.23498248471965621

In [1458]:
```
cm=confusion_matrix(true_P, test_preds, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 76ms, finished 15:16:02 2022-04-27

```
[[ 4  3  2  2  1]
 [ 4  6  8  7  2]
 [ 6  8 12 12  1]
 [ 9  7 15 15  2]
 [ 5  5  5 18  4]]
```



In [1459]:
```
final_train,final_train_wclass=kNN_Model_1('G3',5)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(final_train, final_train_wclass['class_g'])
test_preds = knn_model.predict(totalTestList_wog123_processed)
```

executed in 11.4s, finished 15:16:23 2022-04-27

mission 2

In [1460]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P3[i]==test_preds[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```

executed in 3ms, finished 15:16:33 2022-04-27

Out[1460]: 0.3006134969325153

In [1461]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P3, test_preds, average='macro')
```

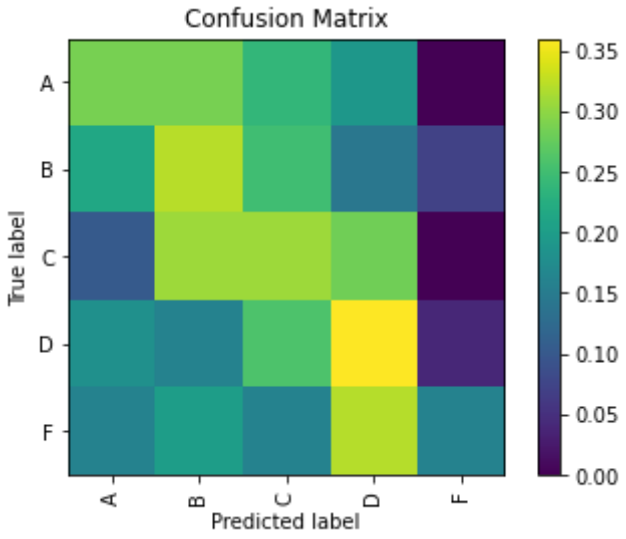
executed in 4ms, finished 15:16:42 2022-04-27

Out[1461]: 0.28521549863964724

In [1463]:
```
cm=confusion_matrix(true_P3, test_preds, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 91ms, finished 15:17:02 2022-04-27

```
[[ 6  6  5  4  0]
 [ 6  9  7  4  2]
 [ 4 12 12 11  0]
 [ 9  8 13 18  2]
 [ 4  5  4  8  4]]
```



In [1464]:
```
def kNN_Model_2(target_g,neighbors):
    max_acc_rate=0
    for i in range(10):
        shuffle_train,shuffle_train_wclass,shuffle_validate,validate_clas

        knn_model = KNeighborsClassifier(n_neighbors=neighbors)
        knn_model.fit(shuffle_train, shuffle_train_wclass['class_g'])
        test_preds = knn_model.predict(shuffle_validate)
        num_acc=0
        for i in range(0,len(validate_class)):
            if validate_class[i]==test_preds[i]:
                num_acc+=1
        total_acc_rate=num_acc/len(validate_class)
        if total_acc_rate>max_acc_rate:
            max_acc_rate=total_acc_rate
            final_train=shuffle_train
            final_train_wclass=shuffle_train_wclass
    return final_train,final_train_wclass
```
executed in 3ms, finished 15:17:11 2022-04-27

In [1465]:
```
final_train,final_train_wclass=kNN_Model_2('G3',5)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(final_train, final_train_wclass['class_g'])
test_preds = knn_model.predict(totalTestList_wog3_processed)
```
executed in 11.5s, finished 15:17:32 2022-04-27

mission 3

In [1466]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P4[i]==test_preds[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:17:42 2022-04-27

Out[1466]: 0.6809815950920245

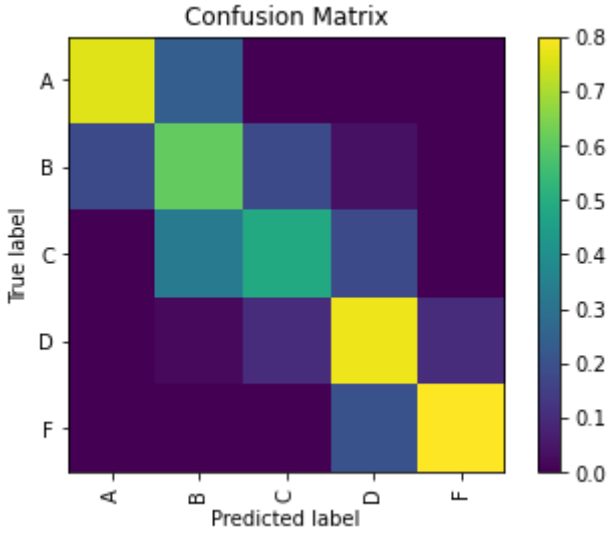
In [1467]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P4, test_preds, average='macro')
```
executed in 4ms, finished 15:17:52 2022-04-27

Out[1467]: 0.6833368347338935

In [1468]:
```
cm=confusion_matrix(true_P4, test_preds, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 81ms, finished 15:18:01 2022-04-27

```
[[16  5  0  0  0]
 [ 5 17  5  1  0]
 [ 0 13 19  7  0]
 [ 0  1  5 39  5]
 [ 0  0  0  5 20]]
```



1.5 Naive Bayes

In [1469]:
```
def Bayes_Model_1(target_g):
    max_acc_rate=0
    for i in range(10):
        shuffle_train,shuffle_train_wclass,shuffle_validate,validate_clas
        mnb = MultinomialNB()
        mnb.fit(shuffle_train,shuffle_train_wclass['class_g'])
        y_pre = mnb.predict(shuffle_validate)
        num_acc=0
        for i in range(0,len(validate_class)):
            if validate_class[i]==y_pre[i]:
                num_acc+=1
        total_acc_rate=num_acc/len(validate_class)
        if total_acc_rate>max_acc_rate:
            max_acc_rate=total_acc_rate
            final_train=shuffle_train
            final_train_wclass=shuffle_train_wclass
    return final_train,final_train_wclass
```
executed in 16ms, finished 15:18:38 2022-04-27

In [1470]:
```
final_train,final_train_wclass=Bayes_Model_1('G1')
mnb = MultinomialNB()
mnb.fit(shuffle_train,shuffle_train_wclass['class_g'])
y_pre = mnb.predict(totalTestList_wog123_processed)
```
executed in 11.4s, finished 15:18:59 2022-04-27

mission 1

In [1471]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P[i]==y_pre[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 4ms, finished 15:19:08 2022-04-27

Out[1471]: 0.3067484662576687

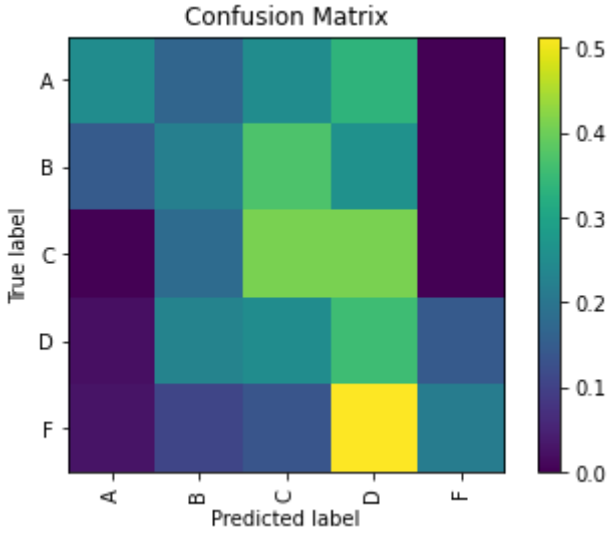
In [1472]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P, y_pre, average='macro')
```
executed in 4ms, finished 15:19:18 2022-04-27

Out[1472]: 0.2973419607475335

In [1473]:
```
cm=confusion_matrix(true_P, y_pre, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 81ms, finished 15:19:28 2022-04-27

```
[[ 3  2  3  4  0]
 [ 4  6 10  7  0]
 [ 0  7 16 16  0]
 [ 1 11 12 17  7]
 [ 1  4  5 19  8]]
```



In [1474]:
```
final_train,final_train_wclass=Bayes_Model_1('G3')
mnb = MultinomialNB()
mnb.fit(final_train,final_train_wclass['class_g'])
y_pre = mnb.predict(totalTestList_wog123_processed)
```

executed in 11.8s, finished 15:19:49 2022-04-27

mission 2

In [1475]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P3[i]==y_pre[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:19:59 2022-04-27

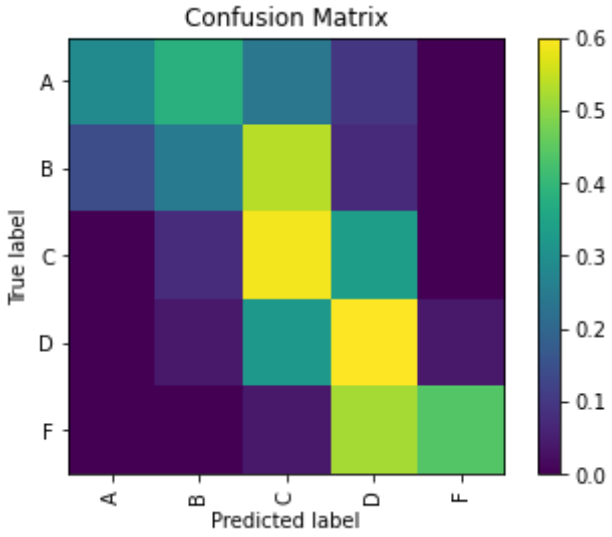
Out[1475]: 0.3067484662576687

In [1476]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P3, y_pre, average='macro')
```
executed in 4ms, finished 15:20:08 2022-04-27

Out[1476]: 0.27925483684397095

In [1482]:
```
cm=confusion_matrix(true_P3, y_pre, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```
executed in 80ms, finished 15:21:17 2022-04-27

```
[[ 6  8  5  2  0]
 [ 4  7 15  2  0]
 [ 0  3 23 13  0]
 [ 0  2 16 30  2]
 [ 0  0  1 13 11]]
```



In [1483]:
```
def Bayes_Model_2(target_g):
    max_acc_rate=0
    for i in range(10):
        shuffle_train,shuffle_train_wclass,shuffle_validate,validate_clas
        mnb = MultinomialNB()
        mnb.fit(shuffle_train,shuffle_train_wclass['class_g'])
        y_pre = mnb.predict(shuffle_validate)
        num_acc=0
        for i in range(0,len(validate_class)):
            if validate_class[i]==y_pre[i]:
                num_acc+=1
        total_acc_rate=num_acc/len(validate_class)
        if total_acc_rate>max_acc_rate:
            max_acc_rate=total_acc_rate
            final_train=shuffle_train
            final_train_wclass=shuffle_train_wclass
    return final_train,final_train_wclass
```
executed in 3ms, finished 15:21:27 2022-04-27

In [1484]:
```
final_train,final_train_wclass=Bayes_Model_2('G3')
mnb = MultinomialNB()
mnb.fit(final_train,final_train_wclass['class_g'])
y_pre = mnb.predict(totalTestList_wog3_processed)
```
executed in 11.5s, finished 15:21:48 2022-04-27

mission 3

In [1485]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P4[i]==y_pre[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:21:57 2022-04-27

Out[1485]: 0.43558282208588955

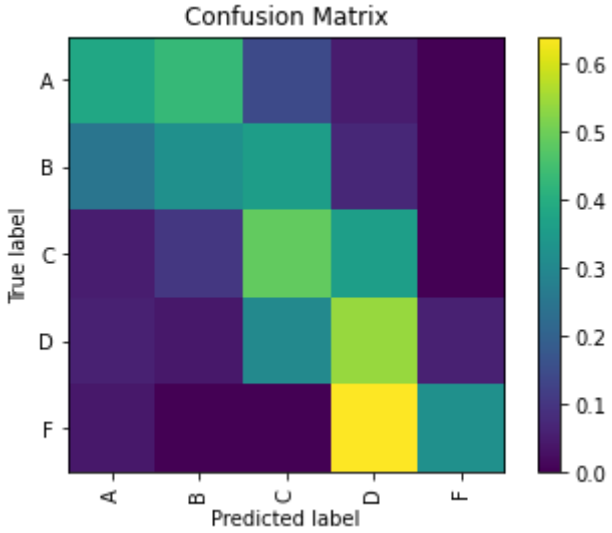
In [1486]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P4, y_pre, average='macro')
```
executed in 4ms, finished 15:22:07 2022-04-27

Out[1486]: 0.4208640455152083

In [1487]:
```
cm=confusion_matrix(true_P4, y_pre, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 86ms, finished 15:22:17 2022-04-27

```
[[ 8  9  3  1  0]
 [ 7  9 10  2  0]
 [ 2  4 19 14  0]
 [ 3  2 15 27  3]
 [ 1  0  0 16  8]]
```



1.6 decision tree

In [1035]:
```
def dtree_Model_1(target_g):
    max_acc_rate=0
    for i in range(10):
        shuffle_train,shuffle_train_wclass,shuffle_validate,validate_clas
        clf = tree.DecisionTreeClassifier()
        clf = clf.fit(shuffle_train, shuffle_train_wclass['class_g'])
        decision_tree=clf.predict(shuffle_validate)
#         mnb = MultinomialNB()
#         mnb.fit(shuffle_train,shuffle_train_wclass['class_g'])
#         y_pre = mnb.predict(shuffle_validate)
        num_acc=0
        for i in range(0,len(validate_class)):
            if validate_class[i]==y_pre[i]:
                num_acc+=1
        total_acc_rate=num_acc/len(validate_class)
        if total_acc_rate>max_acc_rate:
            max_acc_rate=total_acc_rate
            final_train=shuffle_train
            final_train_wclass=shuffle_train_wclass
    return final_train,final_train_wclass
```
executed in 15ms, finished 16:39:30 2022-04-25

In [1488]:
```
final_train,final_train_wclass=dtree_Model_1('G1')
clf = tree.DecisionTreeClassifier()
clf = clf.fit(final_train, final_train_wclass['class_g'])
decision_tree=clf.predict(totalTestList_wog123_processed)
```
executed in 11.5s, finished 15:22:38 2022-04-27

mission 1

In [1489]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P[i]==decision_tree[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:22:47 2022-04-27

Out[1489]: 0.32515337423312884

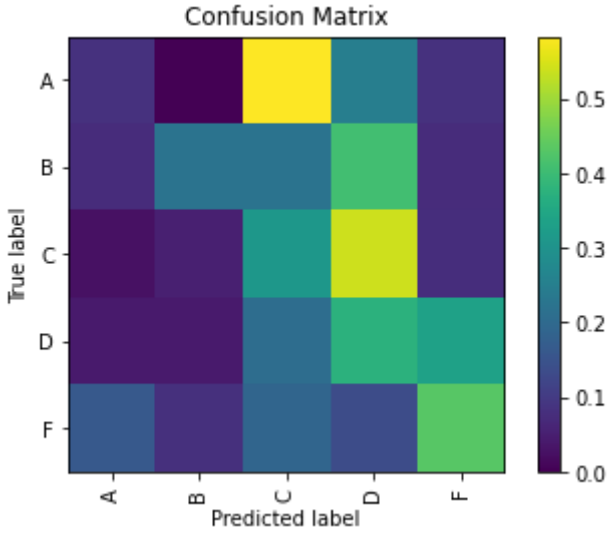
In [1490]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P, decision_tree, average='macro')
```
executed in 5ms, finished 15:22:57 2022-04-27

Out[1490]: 0.28918378756114604

In [1491]:
```
cm=confusion_matrix(true_P, decision_tree, labels=["A", "B", "C","D","F"]
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 83ms, finished 15:23:07 2022-04-27

```
[[ 1  0  7  3  1]
 [ 2  6  6 11  2]
 [ 1  2 12 21  3]
 [ 2  2 10 18 16]
 [ 6  3  7  5 16]]
```



In [1492]:
```
final_train,final_train_wclass=dtree_Model_1('G3')
clf = tree.DecisionTreeClassifier()
clf = clf.fit(final_train, final_train_wclass['class_g'])
decision_tree=clf.predict(totalTestList_wog123_processed)
```

executed in 11.0s, finished 15:23:27 2022-04-27

mission 2

In [1493]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P3[i]==decision_tree[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```

executed in 3ms, finished 15:23:37 2022-04-27

Out[1493]: 0.27607361963190186

In [1494]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P3, decision_tree, average='macro')
```

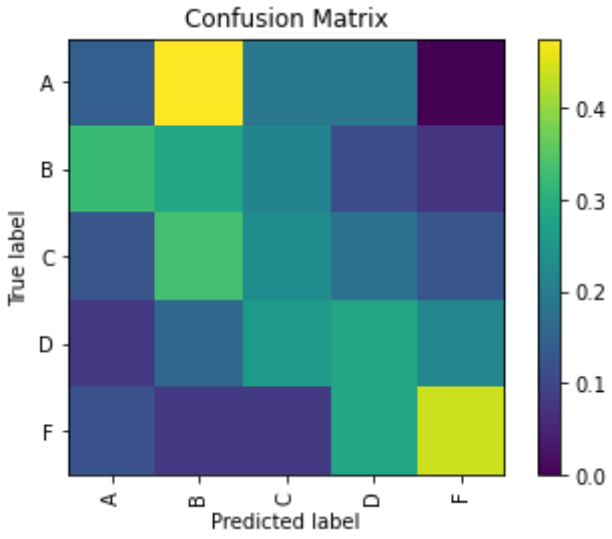
executed in 5ms, finished 15:23:46 2022-04-27

Out[1494]: 0.2697223811766782

In [1495]:
```
cm=confusion_matrix(true_P3, decision_tree, labels=["A", "B", "C","D","F"
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 83ms, finished 15:23:56 2022-04-27

```
[[ 3 10  4  4  0]
 [ 9  8  6  3  2]
 [ 5 13  9  7  5]
 [ 4  8 13 14 11]
 [ 3  2  2  7 11]]
```



In [1042]:
```
def dtree_Model_2(target_g):
    max_acc_rate=0
    for i in range(10):
        shuffle_train,shuffle_train_wclass,shuffle_validate,validate_clas
        clf = tree.DecisionTreeClassifier()
        clf = clf.fit(shuffle_train, shuffle_train_wclass['class_g'])
        decision_tree=clf.predict(shuffle_validate)
#         mnb = MultinomialNB()
#         mnb.fit(shuffle_train,shuffle_train_wclass['class_g'])
#         y_pre = mnb.predict(shuffle_validate)
        num_acc=0
        for i in range(0,len(validate_class)):
            if validate_class[i]==y_pre[i]:
                num_acc+=1
        total_acc_rate=num_acc/len(validate_class)
        if total_acc_rate>max_acc_rate:
            max_acc_rate=total_acc_rate
            final_train=shuffle_train
            final_train_wclass=shuffle_train_wclass
    return final_train,final_train_wclass
```
executed in 20ms, finished 16:41:33 2022-04-25

In [1496]:
```
final_train,final_train_wclass=dtree_Model_2('G3')
clf = tree.DecisionTreeClassifier()
clf = clf.fit(final_train, final_train_wclass['class_g'])
decision_tree=clf.predict(totalTestList_wog3_processed)
```
executed in 11.9s, finished 15:24:18 2022-04-27

mission 3

In [1497]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P4[i]==decision_tree[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:24:27 2022-04-27

Out[1497]: 0.6380368098159509

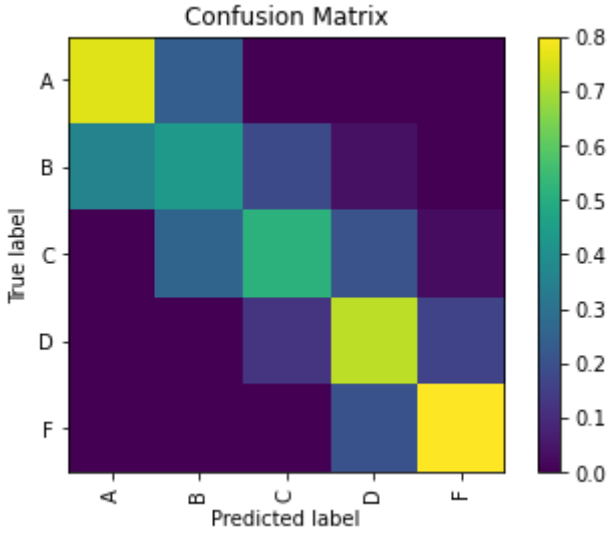
In [1498]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P, decision_tree, average='macro')
```
executed in 4ms, finished 15:24:37 2022-04-27

Out[1498]: 0.5585057882802242

In [1499]:
```
cm=confusion_matrix(true_P4, decision_tree, labels=["A", "B", "C","D","F"
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 91ms, finished 15:24:47 2022-04-27

```
[[16  5  0  0  0]
 [10 12  5  1  0]
 [ 0 10 20  8  1]
 [ 0  0  6 36  8]
 [ 0  0  0  5 20]]
```



1.7 PCA+Random Forest

In [1500]:
```
def RF_Model_1(target_g):
    max_acc_rate=0
    for i in range(10):
        shuffle_train,shuffle_train_wclass,shuffle_validate,validate_clas

        pca = PCA(n_components=20)
        shuffle_train = pca.fit_transform(shuffle_train)
        shuffle_validate = pca.transform(shuffle_validate)

        classifier = RandomForestClassifier(random_state=0)
        classifier.fit(shuffle_train, shuffle_train_wclass['class_g'])
        # Predicting the Test set results
        y_pred = classifier.predict(shuffle_validate)

        num_acc=0
        for i in range(0,len(validate_class)):
            if validate_class[i]==y_pred[i]:
                num_acc+=1
        total_acc_rate=num_acc/len(validate_class)
        if total_acc_rate>max_acc_rate:
            max_acc_rate=total_acc_rate
            final_train=shuffle_train
            final_train_wclass=shuffle_train_wclass
    return final_train,final_train_wclass
```
executed in 14ms, finished 15:24:58 2022-04-27

In [1501]:
```
totalTestList_wog123=totalTestList.drop(['G1','G2','G3'],axis=1)
totalTestList_wog123_processed=pd.get_dummies(totalTestList_wog123)
x = totalTestList_wog123_processed.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
totalTestList_wog123_processed = pd.DataFrame(x_scaled,columns=totalTestL
```
executed in 12ms, finished 15:25:08 2022-04-27

In [1502]:
```
final_train,final_train_wclass=dtree_Model_1('G1')

pca = PCA(n_components=25)
final_train = pca.fit_transform(final_train)
totalTestList_wog123_processed = pca.transform(totalTestList_wog123_proce

classifier = RandomForestClassifier(random_state=0)
classifier.fit(final_train, final_train_wclass['class_g'])
# Predicting the Test set results
y_pred = classifier.predict(totalTestList_wog123_processed)
```
executed in 11.5s, finished 15:25:29 2022-04-27

mission 1

In [1503]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P[i]==y_pred[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:25:39 2022-04-27

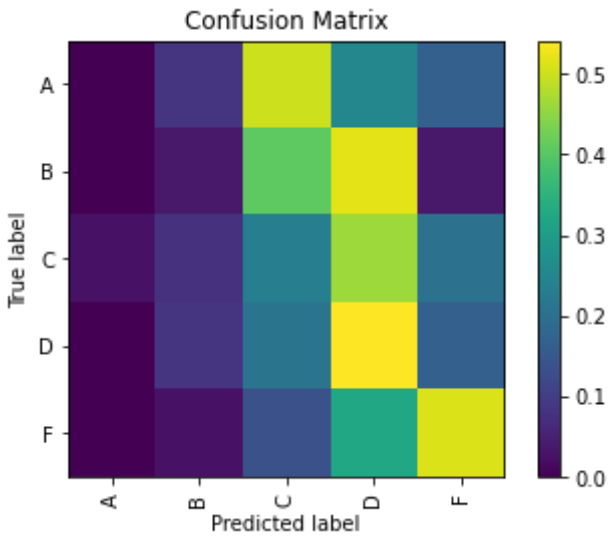
Out[1503]: 0.3374233128834356

In [1505]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P, y_pred, average='macro')
```
executed in 4ms, finished 15:25:57 2022-04-27

Out[1505]: 0.2430945573672846

In [1506]:
```
cm=confusion_matrix(true_P, y_pred, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```
executed in 81ms, finished 15:26:06 2022-04-27

```
[[ 0  1  6  3  2]
 [ 0  1 11 14  1]
 [ 1  3  9 18  8]
 [ 0  4 10 26  8]
 [ 0  1  5 12 19]]
```



In [1507]:
```
totalTestList_wog123=totalTestList.drop(['G1','G2','G3'],axis=1)
totalTestList_wog123_processed=pd.get_dummies(totalTestList_wog123)
x = totalTestList_wog123_processed.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
totalTestList_wog123_processed = pd.DataFrame(x_scaled,columns=totalTestL
```
executed in 11ms, finished 15:26:16 2022-04-27

In [1508]:
```
final_train,final_train_wclass=dtree_Model_1('G3')

pca = PCA(n_components=25)
final_train = pca.fit_transform(final_train)
totalTestList_wog123_processed = pca.transform(totalTestList_wog123_proce

classifier = RandomForestClassifier(random_state=0)
classifier.fit(final_train, final_train_wclass['class_g'])
# Predicting the Test set results
y_pred = classifier.predict(totalTestList_wog123_processed)
```
executed in 11.3s, finished 15:26:36 2022-04-27

mission 2

In [1509]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P3[i]==y_pred[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:26:46 2022-04-27

Out[1509]: 0.38650306748466257

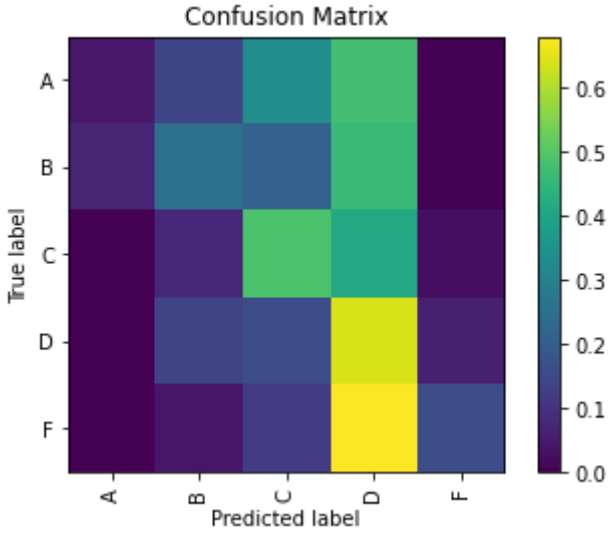
In [1510]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P3, y_pred, average='macro')
```
executed in 3ms, finished 15:26:55 2022-04-27

Out[1510]: 0.3077309223120464

In [1511]:
```
cm=confusion_matrix(true_P3, y_pred, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 88ms, finished 15:27:05 2022-04-27

```
[[ 1  3  7 10  0]
 [ 2  7  6 13  0]
 [ 0  3 19 16  1]
 [ 0  7  8 32  3]
 [ 0  1  3 17  4]]
```



In [1512]:
```
def RF_Model_2(target_g):
    max_acc_rate=0
    for i in range(10):
        shuffle_train,shuffle_train_wclass,shuffle_validate,validate_clas

        pca = PCA(n_components=20)
        shuffle_train = pca.fit_transform(shuffle_train)
        shuffle_validate = pca.transform(shuffle_validate)

        classifier = RandomForestClassifier(random_state=0)
        classifier.fit(shuffle_train, shuffle_train_wclass['class_g'])
        # Predicting the Test set results
        y_pred = classifier.predict(shuffle_validate)

        num_acc=0
        for i in range(0,len(validate_class)):
            if validate_class[i]==y_pred[i]:
                num_acc+=1
        total_acc_rate=num_acc/len(validate_class)
        if total_acc_rate>max_acc_rate:
            max_acc_rate=total_acc_rate
            final_train=shuffle_train
            final_train_wclass=shuffle_train_wclass
    return final_train,final_train_wclass
```

executed in 4ms, finished 15:27:14 2022-04-27

In [1513]:
```
totalTestList_wog3=totalTestList.drop(['G3'],axis=1)
totalTestList_wog3_processed=pd.get_dummies(totalTestList_wog3)
totalTestList_wog3_processed=normalize(totalTestList_wog3_processed)
```
executed in 26ms, finished 15:27:23 2022-04-27

In [1514]:
```
final_train,final_train_wclass=dtree_Model_2('G3')
pca = PCA(n_components=25)
final_train = pca.fit_transform(final_train)
totalTestList_wog3_processed = pca.transform(totalTestList_wog3_processed
classifier = RandomForestClassifier(random_state=0)
classifier.fit(final_train, final_train_wclass['class_g'])
# Predicting the Test set results
y_pred = classifier.predict(totalTestList_wog3_processed)
```
executed in 11.6s, finished 15:27:44 2022-04-27

mission 3

In [1515]:
```
num_acc=0
for i in range(0,len(totalTestList)):
    if true_P3[i]==y_pred[i]:
        num_acc+=1
total_acc_rate=num_acc/len(totalTestList)
total_acc_rate
```
executed in 3ms, finished 15:27:54 2022-04-27

Out[1515]: 0.7055214723926381

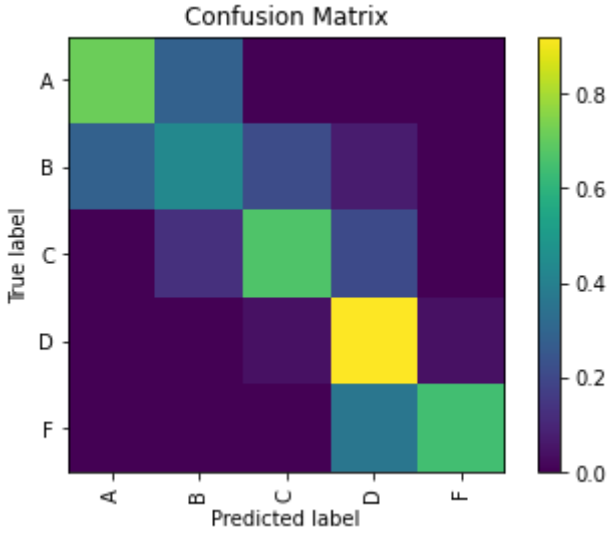
In [1516]:
```
# f1_score(y_true, y_pred, average='macro')
f1_score(true_P4, y_pred, average='macro')
```
executed in 5ms, finished 15:28:03 2022-04-27

Out[1516]: 0.681784246149443

In [1517]:
```
cm=confusion_matrix(true_P4, y_pred, labels=["A", "B", "C","D","F"])
labels_name=["A", "B", "C", "D", "F"]
print(cm)
plot_confusion_matrix(cm, labels_name, "Confusion Matrix")
plt.show()
```

executed in 78ms, finished 15:28:13 2022-04-27

```
[[15  6  0  0  0]
 [ 8 12  6  2  0]
 [ 0  5 26  8  0]
 [ 0  0  2 46  2]
 [ 0  0  0  9 16]]
```



2 Regression

In [1518]:
```
totalTrainList=pd.read_csv("/Users/jiazhidai/Downloads/python3/student_pe
totalTestList=pd.read_csv("/Users/jiazhidai/Downloads/python3/student_per
```

executed in 20ms, finished 15:28:45 2022-04-27

In [1519]:
```
totalTrainList_wog123=totalTrainList.drop(['G1','G2','G3'],axis=1)
totalTrainList_wog123_processed=pd.get_dummies(totalTrainList_wog123)
x = totalTrainList_wog123_processed.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
totalTrainList_wog123_processed = pd.DataFrame(x_scaled,columns=totalTrai

totalTestList_wog123=totalTestList.drop(['G1','G2','G3'],axis=1)
totalTestList_wog123_processed=pd.get_dummies(totalTestList_wog123)
x = totalTestList_wog123_processed.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
totalTestList_wog123_processed = pd.DataFrame(x_scaled,columns=totalTestL


totalTestList_wog3=totalTestList.drop(['G3'],axis=1)
totalTestList_wog3_processed=pd.get_dummies(totalTestList_wog3)
totalTestList_wog3_processed=normalize(totalTestList_wog3_processed)


totalTrainList_processed=pd.get_dummies(totalTrainList)
totalTrainList_processed=normalize(totalTrainList_processed)

totalTestList_processed=pd.get_dummies(totalTestList)
totalTestList_processed=normalize(totalTestList_processed)
```

executed in 73ms, finished 15:28:54 2022-04-27

2.1 Trivial System

In [1520]:
```
def trivial_regressor(target_g):
#     final_out=0
    for i in range(0,10):
        MSE=0
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList)

        train_mean=shuffle_train[target_g].mean()
        validate_output=shuffle_validate[target_g]
        for j in validate_output:
            MSE+=(j-train_mean)**2
        if i==0:
            min_MSE=400*400
            result=train_mean
        if min_MSE>MSE:
            min_MSE=MSE
            result=train_mean
    return result
```

executed in 12ms, finished 15:33:30 2022-04-27

In [1522]:
```
final_mean_pred=[]
final_mean=trivial_regressor('G1')
for i in range(len(totalTestList)):
    final_mean_pred.append(final_mean)
```
executed in 4.58s, finished 15:33:47 2022-04-27

In [1524]:
```
final_MSE=mean_squared_error(totalTestList_processed['G1'], final_mean_pr
```
executed in 12ms, finished 15:35:07 2022-04-27

mission 1

In [1527]:
```
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 15:35:49 2022-04-27

Out[1527]: 2.8404314672678512

In [1529]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(totalTestList_processed['G1'], final_mean_pred)
```
executed in 12ms, finished 15:38:25 2022-04-27

Out[1529]: 2.2437423312883435

In [1530]:
```
# r2_score(y_true, y_pred)
r2_score(totalTestList_processed['G1'], final_mean_pred)
```
executed in 11ms, finished 15:38:52 2022-04-27

Out[1530]: -9.53964862526746e-06

In [1531]:
```
final_mean_pred=[]
final_mean=trivial_regressor('G3')
for i in range(len(totalTestList)):
    final_mean_pred.append(final_mean)
```
executed in 4.63s, finished 15:43:15 2022-04-27

In [1532]:
```
final_MSE=mean_squared_error(totalTestList_processed['G3'], final_mean_pr
```
executed in 2ms, finished 15:43:24 2022-04-27

mission 2 and 3

In [1533]:
```
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 15:43:33 2022-04-27

Out[1533]: 3.1755148367670407

In [1534]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(totalTestList_processed['G3'], final_mean_pred)
```
executed in 3ms, finished 15:43:42 2022-04-27

Out[1534]: 2.4712883435582818

In [1535]:
```
# r2_score(y_true, y_pred)
r2_score(totalTestList_processed['G3'], final_mean_pred)
```
executed in 2ms, finished 15:43:52 2022-04-27

Out[1535]: -0.0007133822386899968

2.2 Baseline model

2.2.1 1NN

In [1598]:
```
def pred_y_1NN(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog123=shuffle_train.drop(['G1','G2','G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog123=shuffle_validate.drop(['G1','G2','G3'],ax
        shuffle_validate_gclass=shuffle_validate[target_g]

        neigh = KNeighborsRegressor(n_neighbors=1)
        neigh.fit(shuffle_train_wog123, shuffle_train_gclass)
        pred_y=neigh.predict(shuffle_validate_wog123)
        test_y=shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_neigh=neigh
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_neigh=neigh


    final_pred_y=final_neigh.predict(totalTestList_wog123_processed)

    return final_pred_y
```
executed in 14ms, finished 16:37:36 2022-04-27

In [1599]:
```
pred_y=pred_y_1NN('G1')
test_y=totalTestList['G1']
```
executed in 9.41s, finished 16:37:55 2022-04-27

mission 1

In [1600]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:38:04 2022-04-27

Out[1600]: 3.1748629263681596

In [1601]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:38:13 2022-04-27

Out[1601]: 2.3865030674846626

In [1602]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:38:23 2022-04-27

Out[1602]: -0.2493538846229204

In [1603]:
```
pred_y=pred_y_1NN('G3')
test_y=totalTestList['G3']
```
executed in 9.41s, finished 16:38:41 2022-04-27

mission 2

In [1604]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:38:50 2022-04-27

Out[1604]: 3.2700537508949807

In [1605]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:39:00 2022-04-27

Out[1605]: 2.496932515337423

In [1606]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:39:09 2022-04-27

Out[1606]: -0.06118523277356114

In [1577]:
```
def pred_y_1NN2(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog3=shuffle_train.drop(['G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog3=shuffle_validate.drop(['G3'],axis=1)
        shuffle_validate_gclass=shuffle_validate[target_g]

        neigh = KNeighborsRegressor(n_neighbors=1)
        neigh.fit(shuffle_train_wog3, shuffle_train_gclass)
        pred_y=neigh.predict(shuffle_validate_wog3)
        test_y=shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_neigh=neigh
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_neigh=neigh


    final_pred_y=final_neigh.predict(totalTestList_wog3_processed)

    return final_pred_y
```
executed in 15ms, finished 16:24:04 2022-04-27

In [1578]:
```
pred_y=pred_y_1NN2('G3')
test_y=totalTestList['G3']
```
executed in 9.54s, finished 16:24:23 2022-04-27

mission 3

In [1579]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:24:32 2022-04-27

Out[1579]: 1.3566468949384038

In [1580]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:24:41 2022-04-27

Out[1580]: 0.9447852760736196

In [1581]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 2ms, finished 16:24:51 2022-04-27

Out[1581]: 0.8173519392816591

2.2.2 Linear Regression

In [1582]:
```
def pred_y_LR(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog123=shuffle_train.drop(['G1','G2','G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog123=shuffle_validate.drop(['G1','G2','G3'],ax
        shuffle_validate_gclass=shuffle_validate[target_g]

        reg = LinearRegression().fit(shuffle_train_wog123, shuffle_train_
        pred_y=reg.predict(shuffle_validate_wog123)
        test_y=shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_reg=reg
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_reg=reg


    final_pred_y=final_reg.predict(totalTestList_wog123_processed)

    return final_pred_y
```
executed in 11ms, finished 16:29:27 2022-04-27

In [1583]:
```
pred_y=pred_y_LR('G1')
test_y=totalTestList['G1']
```
executed in 9.49s, finished 16:29:46 2022-04-27

mission 1

In [1584]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:29:55 2022-04-27

Out[1584]: 2.416933844796481

In [1585]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 2ms, finished 16:30:05 2022-04-27

Out[1585]: 1.7657208588957056

In [1586]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 2ms, finished 16:30:14 2022-04-27

Out[1586]: 0.2759558666168279

mission 2

In [1587]:
```
pred_y=pred_y_LR('G3')
test_y=totalTestList['G3']
```
executed in 9.74s, finished 16:31:13 2022-04-27

In [1588]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:31:22 2022-04-27

Out[1588]: 2.7177813220377347

In [1589]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:31:32 2022-04-27

Out[1589]: 1.9497699386503067

In [1590]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:31:41 2022-04-27

Out[1590]: 0.26698909726009235

In [1591]:
```
def pred_y_LR2(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog3=shuffle_train.drop(['G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog3=shuffle_validate.drop(['G3'],axis=1)
        shuffle_validate_gclass=shuffle_validate[target_g]

        reg = LinearRegression().fit(shuffle_train_wog3, shuffle_train_gc
        pred_y=reg.predict(shuffle_validate_wog3)
        test_y=shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_reg=reg
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_reg=reg


    final_pred_y=final_reg.predict(totalTestList_wog3_processed)

    return final_pred_y
```
executed in 14ms, finished 16:32:37 2022-04-27

In [1592]:
```
pred_y=pred_y_LR2('G3')
test_y=totalTestList['G3']
```
executed in 9.74s, finished 16:32:56 2022-04-27

mission 3

In [1593]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 2ms, finished 16:33:05 2022-04-27

Out[1593]: 1.0827802572551912

In [1594]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:33:14 2022-04-27

Out[1594]: 0.7977257476993865

In [1595]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:33:24 2022-04-27

Out[1595]: 0.8836511561409621

2.3 kNN

In [1607]:
```
def pred_y_kNN(target_g,neighbor):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog123=shuffle_train.drop(['G1','G2','G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog123=shuffle_validate.drop(['G1','G2','G3'],ax
        shuffle_validate_gclass=shuffle_validate[target_g]

        neigh = KNeighborsRegressor(n_neighbors=neighbor)
        neigh.fit(shuffle_train_wog123, shuffle_train_gclass)
        pred_y=neigh.predict(shuffle_validate_wog123)
        test_y=shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_neigh=neigh
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_neigh=neigh


    final_pred_y=final_neigh.predict(totalTestList_wog123_processed)

    return final_pred_y
```
executed in 4ms, finished 16:39:18 2022-04-27

In [1608]:
```
pred_y=pred_y_kNN('G1',15)
test_y=totalTestList['G1']
```
executed in 9.47s, finished 16:39:37 2022-04-27

mission 1

In [1609]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:39:46 2022-04-27

Out[1609]: 2.6475490588630404

In [1610]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:39:55 2022-04-27

Out[1610]: 2.0482617586912064

In [1611]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 2ms, finished 16:40:04 2022-04-27

Out[1611]: 0.13119253253384022

mission 2

In [1612]:
```
pred_y=pred_y_kNN('G3',15)
test_y=totalTestList['G3']
```
executed in 9.83s, finished 16:40:43 2022-04-27

In [1613]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:40:52 2022-04-27

Out[1613]: 2.9182966523396106

In [1614]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:41:01 2022-04-27

Out[1614]: 2.2482617586912066

In [1615]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:41:11 2022-04-27

Out[1615]: 0.15483736395811332

In [1618]:
```
def pred_y_kNN2(target_g,neighbor):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog3=shuffle_train.drop(['G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog3=shuffle_validate.drop(['G3'],axis=1)
        shuffle_validate_gclass=shuffle_validate[target_g]

        neigh = KNeighborsRegressor(n_neighbors=neighbor)
        neigh.fit(shuffle_train_wog3, shuffle_train_gclass)
        pred_y=neigh.predict(shuffle_validate_wog3)
        test_y=shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_neigh=neigh
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_neigh=neigh


    final_pred_y=final_neigh.predict(totalTestList_wog3_processed)

    return final_pred_y
```
executed in 17ms, finished 16:42:52 2022-04-27

In [1619]:
```
pred_y=pred_y_kNN2('G3',15)
test_y=totalTestList['G3']
```
executed in 9.61s, finished 16:43:11 2022-04-27

mission 3

In [1620]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 2ms, finished 16:43:20 2022-04-27

Out[1620]: 1.1161820210081388

In [1621]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:43:30 2022-04-27

Out[1621]: 0.7918200408997955

In [1622]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```

executed in 3ms, finished 16:43:39 2022-04-27

Out[1622]: 0.8763621453282602

2.4 SVR

In [1623]:
```
def pred_y_SVR(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog123=shuffle_train.drop(['G1','G2','G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog123=shuffle_validate.drop(['G1','G2','G3'],ax
        shuffle_validate_gclass=shuffle_validate[target_g]

        clf = SVR(C=1.0, epsilon=0.2)
        clf.fit(shuffle_train_wog123, shuffle_train_gclass)
        pred_y = clf.predict(shuffle_validate_wog123)
        test_y = shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_clf=clf
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_clf=clf


    final_pred_y=final_clf.predict(totalTestList_wog123_processed)

    return final_pred_y
```

executed in 16ms, finished 16:45:47 2022-04-27

In [1624]:
```
pred_y=pred_y_SVR('G1')
test_y=totalTestList['G1']
```

executed in 9.90s, finished 16:46:19 2022-04-27

mission 1

In [1625]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:46:29 2022-04-27

Out[1625]: 2.4635431673726536

In [1626]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:46:38 2022-04-27

Out[1626]: 1.866445700093324

In [1627]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:46:48 2022-04-27

Out[1627]: 0.24776096555919092

In [1628]:
```
pred_y=pred_y_SVR('G3')
test_y=totalTestList['G3']
```
executed in 9.61s, finished 16:47:35 2022-04-27

mission 2

In [1629]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:47:44 2022-04-27

Out[1629]: 2.781410856820877

In [1630]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 2ms, finished 16:47:54 2022-04-27

Out[1630]: 2.0635657789287816

In [1631]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 2ms, finished 16:48:03 2022-04-27

Out[1631]: 0.2322643526596242

In [1632]:
```
def pred_y_SVR2(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog3=shuffle_train.drop(['G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog3=shuffle_validate.drop(['G3'],axis=1)
        shuffle_validate_gclass=shuffle_validate[target_g]

        clf = SVR(C=1.0, epsilon=0.2)
        clf.fit(shuffle_train_wog3, shuffle_train_gclass)
        pred_y = clf.predict(shuffle_validate_wog3)
        test_y = shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_clf=clf
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_clf=clf


    final_pred_y=final_clf.predict(totalTestList_wog3_processed)

    return final_pred_y
```
executed in 4ms, finished 16:48:12 2022-04-27

In [1637]:
```
pred_y=pred_y_SVR2('G3')
test_y=totalTestList['G3']
```
executed in 9.91s, finished 16:49:19 2022-04-27

mission 3

In [1638]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:49:29 2022-04-27

Out[1638]: 1.1327874635928095

In [1639]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 2ms, finished 16:49:38 2022-04-27

Out[1639]: 0.7317108847387506

In [1640]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```

executed in 3ms, finished 16:49:48 2022-04-27

Out[1640]: 0.8726560598385871

2.5 Ridge Regression

In [1658]:
```
def pred_y_RR(target_g):
#     min_MSE=float('inf')
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog123=shuffle_train.drop(['G1','G2','G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog123=shuffle_validate.drop(['G1','G2','G3'],ax
        shuffle_validate_gclass=shuffle_validate[target_g]

        clf = Ridge(alpha=1.0)
        clf.fit(shuffle_train_wog123, shuffle_train_gclass)
        pred_y = clf.predict(shuffle_validate_wog123)
        test_y = shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_clf=clf
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_clf=clf


    final_pred_y=final_clf.predict(totalTestList_wog123_processed)

    return final_pred_y
```

executed in 3ms, finished 16:53:43 2022-04-27

In [1643]:
```
pred_y=pred_y_RR('G1')
test_y=totalTestList['G1']
```

executed in 9.51s, finished 16:51:01 2022-04-27

mission 1

In [1644]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 2ms, finished 16:51:11 2022-04-27

Out[1644]: 2.4337974822889024

In [1645]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:51:21 2022-04-27

Out[1645]: 1.8318525944561834

In [1646]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:51:30 2022-04-27

Out[1646]: 0.26581689304882683

mission 2

In [1647]:
```
pred_y=pred_y_RR('G3')
test_y=totalTestList['G3']
```
executed in 9.65s, finished 16:51:49 2022-04-27

In [1648]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:51:58 2022-04-27

Out[1648]: 2.771181679570248

In [1649]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:52:08 2022-04-27

Out[1649]: 2.0378447488164904

In [1650]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:52:18 2022-04-27

Out[1650]: 0.23790096184707987

In [1653]:
```
def pred_y_RR2(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog3=shuffle_train.drop(['G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog3=shuffle_validate.drop(['G3'],axis=1)
        shuffle_validate_gclass=shuffle_validate[target_g]

        clf = Ridge(alpha=1.0)
        clf.fit(shuffle_train_wog3, shuffle_train_gclass)
        pred_y = clf.predict(shuffle_validate_wog3)
        test_y = shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_clf=clf
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_clf=clf


    final_pred_y=final_clf.predict(totalTestList_wog3_processed)

    return final_pred_y
```
executed in 15ms, finished 16:52:45 2022-04-27

In [1654]:
```
pred_y=pred_y_RR2('G3')
test_y=totalTestList['G3']
```
executed in 9.48s, finished 16:53:04 2022-04-27

mission 3

In [1655]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:53:14 2022-04-27

Out[1655]: 1.071758998540313

In [1656]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:53:24 2022-04-27

Out[1656]: 0.7983375283276066

In [1657]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```

executed in 3ms, finished 16:53:33 2022-04-27

Out[1657]: 0.886007653859711

2.6 Lasso Regression

In [1659]:
```
def pred_y_LaR(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog123=shuffle_train.drop(['G1','G2','G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog123=shuffle_validate.drop(['G1','G2','G3'],ax
        shuffle_validate_gclass=shuffle_validate[target_g]

        clf = Lasso(alpha=0.1)
        clf.fit(shuffle_train_wog123, shuffle_train_gclass)
        pred_y = clf.predict(shuffle_validate_wog123)
        test_y = shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_clf=clf
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_clf=clf


    final_pred_y=final_clf.predict(totalTestList_wog123_processed)

    return final_pred_y
```

executed in 15ms, finished 16:55:01 2022-04-27

mission 1

In [1660]:
```
pred_y=pred_y_LaR('G1')
test_y=totalTestList['G1']
```

executed in 9.79s, finished 16:55:41 2022-04-27

In [1661]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:55:51 2022-04-27

Out[1661]: 2.538547329405208

In [1662]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 4ms, finished 16:56:00 2022-04-27

Out[1662]: 1.9355713615008638

In [1663]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:56:10 2022-04-27

Out[1663]: 0.2012588811161058

In [1664]:
```
pred_y=pred_y_LaR('G3')
test_y=totalTestList['G3']
```
executed in 9.72s, finished 16:56:29 2022-04-27

mission 2

In [1665]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:56:39 2022-04-27

Out[1665]: 2.8071550955483024

In [1666]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 3ms, finished 16:56:48 2022-04-27

Out[1666]: 2.142118310575255

In [1667]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```
executed in 3ms, finished 16:56:58 2022-04-27

Out[1667]: 0.21798653439809756

In [1668]:
```
def pred_y_LaR2(target_g):
    min_MSE=400*400
    for i in range(10):
        shuffle_train, shuffle_validate=shuffle_data(totalTrainList_proce

        shuffle_train_wog3=shuffle_train.drop(['G3'],axis=1)
        shuffle_train_gclass=shuffle_train[target_g]
        shuffle_validate_wog3=shuffle_validate.drop(['G3'],axis=1)
        shuffle_validate_gclass=shuffle_validate[target_g]

        clf = Lasso(alpha=1.0)
        clf.fit(shuffle_train_wog3, shuffle_train_gclass)
        pred_y = clf.predict(shuffle_validate_wog3)
        test_y = shuffle_validate_gclass

        final_MSE=mean_squared_error(test_y, pred_y)
        if i==0:
            min_MSE=final_MSE

            final_clf=clf
        if final_MSE<min_MSE:
            min_MSE=final_MSE

            final_clf=clf


    final_pred_y=final_clf.predict(totalTestList_wog3_processed)

    return final_pred_y
```
executed in 4ms, finished 16:57:07 2022-04-27

In [1669]:
```
pred_y=pred_y_LaR2('G3')
test_y=totalTestList['G3']
```
executed in 9.66s, finished 16:57:27 2022-04-27

mission 3

In [1670]:
```
final_MSE=mean_squared_error(test_y, pred_y)
final_RMSE=np.sqrt(final_MSE)
final_RMSE
```
executed in 3ms, finished 16:57:36 2022-04-27

Out[1670]: 0.9564009420707236

In [1671]:
```
# mean_absolute_error(y_true, y_pred)
mean_absolute_error(test_y, pred_y)
```
executed in 2ms, finished 16:57:45 2022-04-27

Out[1671]: 0.6936270526459382

In [1672]:
```
# r2_score(y_true, y_pred)
r2_score(test_y, pred_y)
```

executed in 3ms, finished 16:57:54 2022-04-27

Out[1672]: 0.909226014150883