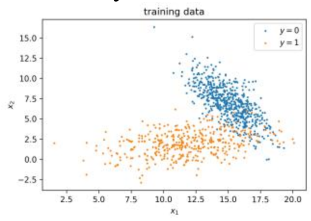
Homework 2

Student ID: 2020321249

Name: 남혜린 Hyelin Nam

(a)

	Hw#1 dataset	Wine dataset	Letter Recognition dataset
Number of samples	1000	178	20000
Number of features	2	13	16
Number of classes (Number of samples per class)	2 (600 / 400)	3 (59 / 71 / 48)	26 (700-800)
Explanation	When plotted on a two-dimensional plane, the samples appear to form clusters by classes 	Each three types of wines have 13 features of chemical analysis	16 features of each 26 capital letters in the English alphabet, composed of black-and-white rectangular pixel displays.

(b)

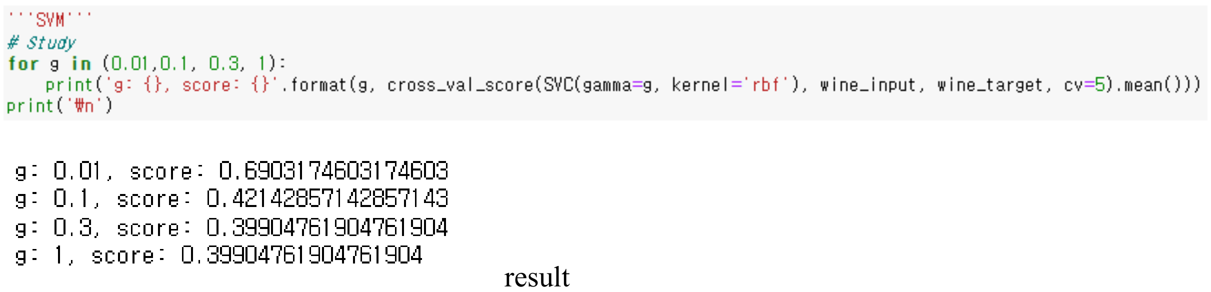
For fair cross validation scoring, test set and train sets of 5-fold trials must have samples of all classes. Given data itself were sorted in classes, so I shuffled dataset first and processed 5-fold cross validation test.

(i) SVM

In SVM algorithm, optimization process operates to maximize margin between linear decision boundary and data samples, and it uses kernel to enable linearity of deicision boundary by converting to higher dimensions. It finds optimal weights of linear boundary with all dataset with radial basis function (rbf), or Gaussian kernel.

$$k(x_1, x_2) = \exp\left(-\gamma||x_1 - x_2||^2\right)$$

Rbf kernel has paramter gamma, which decides distance of each data samples influences in determining the decision boundary. It is inversely proportional to standard deviation of Gaussian distribution, thus the influential distance gets smaller when gamma gets bigger, recurring under-fitting. When studied with few parameters, smaller gamma performed better. The best gamma is 1/number of features.



```
'''SVM'''
# Study
for g in (0.01,0.1, 0.3, 1):
    print('g: {}, score: {}'.format(g, cross_val_score(SVC(gamma=g, kernel='rbf'), wine_input, wine_target, cv=5).mean()))
print('\n')
```

```
g: 0.01, score: 0.6903174603174603
g: 0.1, score: 0.4214285714285714
g: 0.3, score: 0.3990476190476190
g: 1, score: 0.3990476190476190
```

result

(ii) TER-RM

TER performs overwhelmingly when numbers of samples are imbalanced between classes, because TER considers the imbalance by applying weighted summation of mean sqare loss according to the number of data in each class in the object function. When the data are balanced across classes, it operates the same with linear regression. In this experiment, η is ignored. (def: TER)

RM, reduced polynomial model reduces number of weight parameters thus saving computation resource. To form RM, I calculated data correspondingly to form $\mathrm{m} \times \mathrm{K}$ matrix, where m is number of data samples and $\mathrm{K} = 1 + \mathrm{r} + \mathrm{d}(2\mathrm{r} - 1)$, r =model orders, d= number of features. Then applying converted data to linear regression model forms RM model. (def: RM_data)

I performed multiple experiments with different orders, but order=1 equals to normal SVM, so I excluded that experiment.

(iii) Linear Regression

Linear Regression finds linear line to optimally express the data sample. Therefore, the output of linear regression model does not predict the exact class. To get predicted class, I found the closest value among classes of each output. (def: preds2labels)

(iv) Comparison of methods

To make a fair comparison of the methods, first, I compared each models optimized with minimizing MSE loss, as the original method of SVM. In this experiment, RM with MSE optimization is operated, not weighted object function by distribution of classes.

I also compared TER minimizing methods. However, SVM itself is an algorithm using MSE loss, so I just scored with TER function with trained SVM model for comparison.

(c)

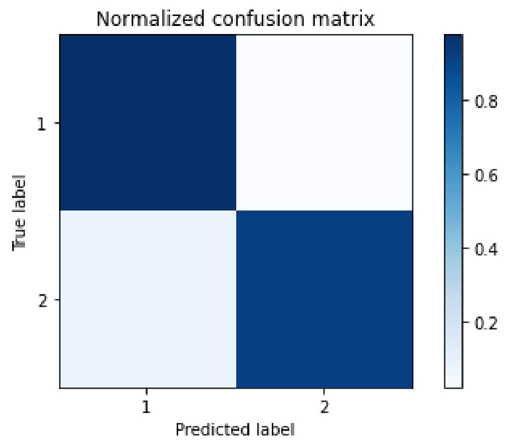
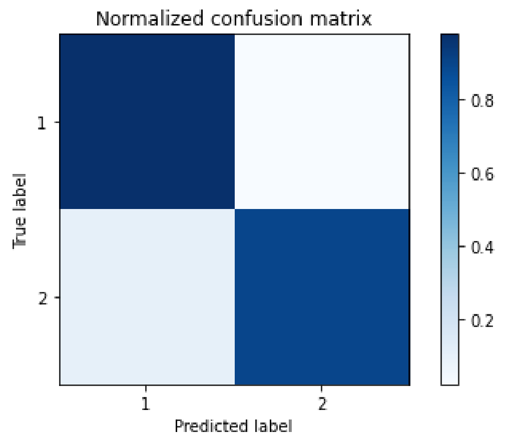
(i) Hw1 dataset

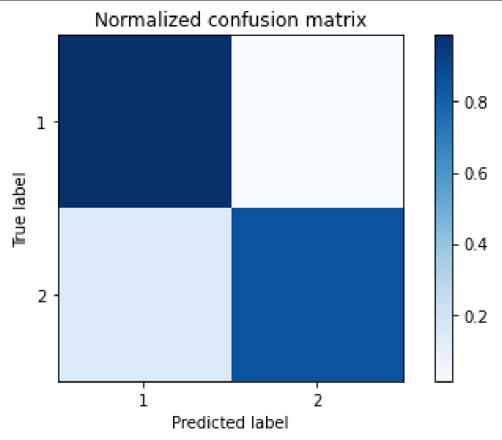
- MSE

Methods	SVM	RM		Linear Regression
Score (Accuracy)	0.9460	order=2	0.7329	0.7004
		order=3	0.7662	
		order=4	0.7771	
		order=5	0.7687	

- TER

Methods	SVM	RM		Linear Regression
Score (loss, value of object function)	0.1241	order=2	0.1471	0.1559
		order=3	0.1553	
		order=4	0.1417	
		order=5	0.1391	

Methods	SVM	TER-RM (order=5)
Confusio n matrix		

Methods	Linear Regression
Confusion matrix	

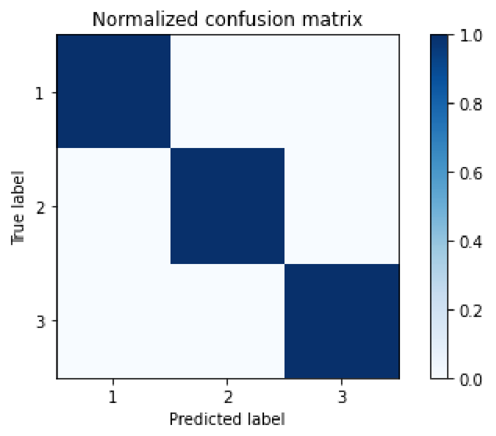
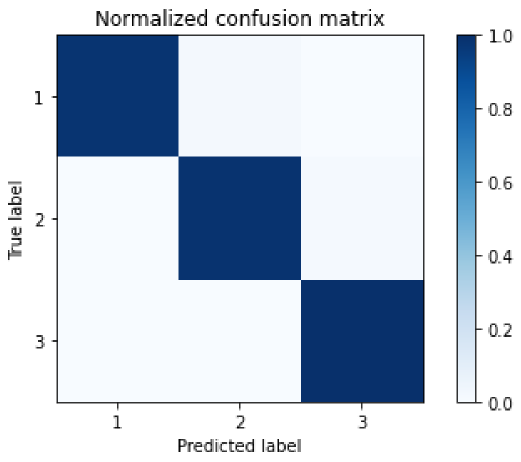
(ii) Wine dataset

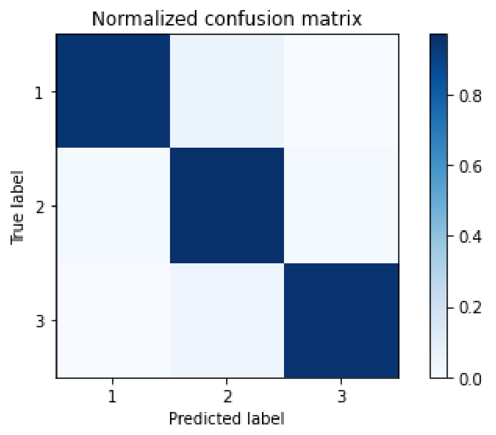
- MSE

Methods	SVM	RM		Linear Regression
Score (Accuracy)	0.4439	order=2	0.6998	0.8815
		order=3	0.7732	
		order=4	0.5658	
		order=5	0.6614	

- TER

Methods	SVM	RM		Linear Regression
Score (loss, value of object function)	1.8584	order=2	0.4159	0.1639
		order=3	0.3021	
		order=4	0.5946	
		order=5	0.7252	

Methods	SVM	TER-RM (order=5)
Confusio n matrix		

Methods	Linear Regression
Confusion matrix	

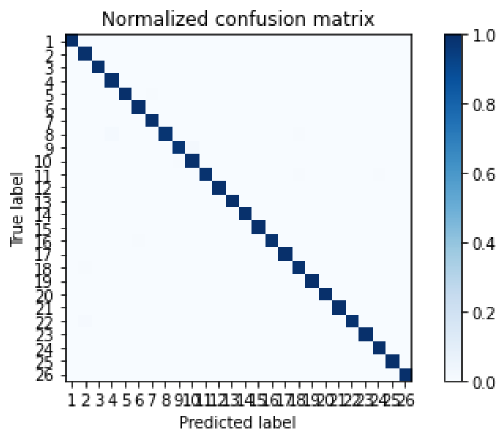
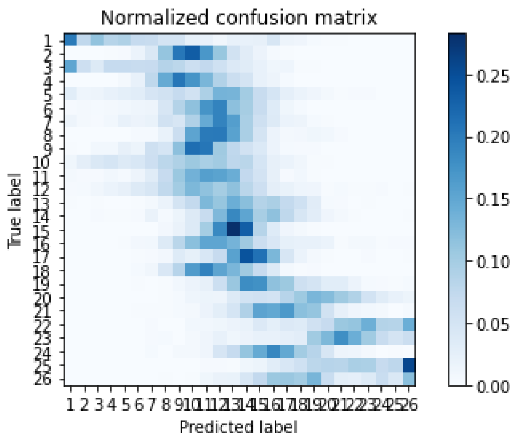
(iii) Letter Recognition dataset

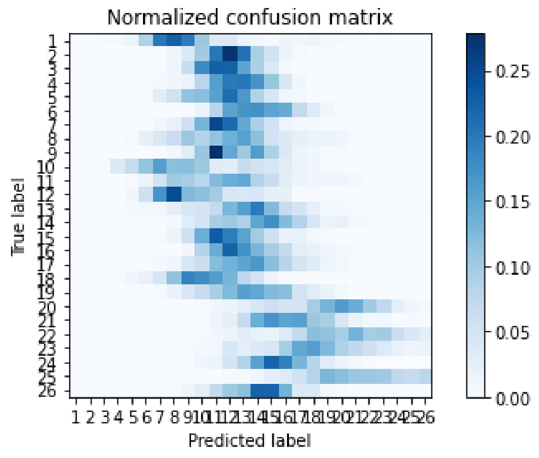
- MSE

Methods	SVM	RM		Linear Regression
Score (Accuracy)	0.9723	order=2	0.4181	0.2847
		order=3	0.4404	
		order=4	0.4821	
		order=5	0.4907	

- TER

Methods	SVM	RM		Linear Regression
Score (loss, value of object function)	0.7250	order=2	24.29	24.75
		order=3	24.22	
		order=4	24.13	
		order=5	24.10	

Methods	SVM	TER-RM (order=5)
Confusio n matrix		

Methods	Linear Regression
Confusion matrix	

In HW1 dataset, TER-RM performs than the other methods as can be seen in overall evaluations. It overwhelms linear regression, which can be seem that RM performs effectively.

When it comes to dataset with more features, wine dataset, TER-RM results as good as SVM in prediction, but it degrades in score. I mainly used pre-made library where exact calculation scheme is not clear, but confusion matrix shows that TER-RM performs perfectly as SVM but with less computation, and even overwhelms linear regression.

Experiment with Letter Recognition dataset, which has large feature and much more classes than wine dataset, the performance of TER-RM decreases. When the dataset is confused, it classifies more exactly when converting the data into infinitely many dimensions through the rbf kernel and then obtaining a decision boundary with a large margin, rather than trying to represent the data into a linear plane.

(d) Appendix

(1) Importing necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels
from sklearn.linear_model import LinearRegression
import tensorflow as tf
import math
```

(2) load datasets

```
'''dataset load'''
wine_data = pd.read_csv('D:/Dropbox/나메헹/coursework/2022 2학기/통계적패턴인식/Hw2/winedata/wine.data', header=None)
wine_name = open('D:/Dropbox/나메헹/coursework/2022 2학기/통계적패턴인식/Hw2/winedata/wine.names', 'r')

letter_data = pd.read_csv('D:/Dropbox/나메헹/coursework/2022 2학기/통계적패턴인식/Hw2/letter-recognitiondata/letter-recogniti:ion.data', header=None)
letter_name = open('D:/Dropbox/나메헹/coursework/2022 2학기/통계적패턴인식/Hw2/letter-recognitiondata/letter-recognition.namees', 'r')

train_data= open('D:/Dropbox/나메헹/coursework/2022 2학기/통계적패턴인식/Hw2/hw1data/train.txt', 'r')
test_data= open('D:/Dropbox/나메헹/coursework/2022 2학기/통계적패턴인식/Hw2/hw1data/test.txt', 'r')
np_traindata=np.loadtxt('D:/Dropbox/나메헹/coursework/2022 2학기/통계적패턴인식/Hw1/train.txt', dtype = 'str')
np_testdata=np.loadtxt('D:/Dropbox/나메헹/coursework/2022 2학기/통계적패턴인식/Hw1/test.txt', dtype = 'str')
```

(3) def: plot_confusion_matrix

```
def plot_confusion_matrix(y_true, y_pred, classes, normalize=False, title=None, cmap=plt.cm.Blues):
    # Set title
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)

    # Normalize
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    # Plot
    fig, ax = plt.subplots()
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]), yticks=np.arange(cm.shape[0]),
           xticklabels=classes, yticklabels=classes,
           title=title, ylabel='True label', xlabel='Predicted label')

    fig.tight_layout()

    return ax
```

(4) def: RM_data

Converting data to form RM

```
def RM_data(x, order):
    [m,d]=x.shape
    # m= num of samples
    # d= feature dim

    out_data=np.ones((m,1))

    for i in range(order):
        for k in range(d):
            out_data=np.concatenate((out_data, np.expand_dims(x[:,k], axis=1)**(i+1)),1)

    for i in range(order):
        out_data=np.concatenate((out_data, np.expand_dims(np.sum(x,1), axis=1)**(i+1)),1)

    for i in range(order-1):
        out_data=np.concatenate((out_data,x*np.expand_dims(np.sum(x,1)**(i+1), axis=1)),1)

    # mxK
    # K=1+r+d*(2*r-1)
    # print('K dim:',out_data.shape)
    return out_data
```

(5) def: preds2labels

Making a prediction with output of linear regression or RM model, by finding the closest label value.

```
def preds2labels(preds, labels):
    search = np.searchsorted(preds, labels)
    a = preds[search - 1]
    b = preds[np.minimum(len(preds) - 1, search)]
    return np.where(np.fabs(labels - a) < np.fabs(labels - b), a, b)
```

(6) def: TER

Weighted sum of MSE loss of FN and FP

```
def TER(y_target, y_pred):
    num_class=np.max(y_target)
    num_samples_class=np.zeros((int(num_class)))
    loss=0
    y_pred=preds2labels(np.arange(1,int(num_class)+1), y_pred)
    for c in range(int(num_class)):
        c_idx=np.where(y_target==c+1)[0]
        num_samples_class[c]=c_idx.shape[0]
        if num_samples_class[c]>0:
            loss+=(y_target[c_idx]!=y_pred[c_idx]).sum()/num_samples_class[c]
    return loss
```

(7) Dataset pre-processing (shuffling)

```
# Hw1 dataset
hw1_input=np_train_data[:,:2]
hw1_target=np_train_data[:,2].astype(int)
hw1_target=hw1_target+np.ones_like(hw1_target)

idx = np.arange(hw1_target.shape[0])
np.random.shuffle(idx)

hw1_target=hw1_target[idx]
hw1_input=hw1_input[idx]

num_class=2
hw1_class= np.arange(1,num_class+1)
```

```
# Wine dataset
wine_target=np.array(wine_data)[:,0]
wine_input=np.array(wine_data)[:,1:]

idx = np.arange(wine_target.shape[0])
np.random.shuffle(idx)

wine_target=wine_target[idx]
wine_input=wine_input[idx]

num_class=3
wine_class= np.arange(1,num_class+1)
```

```
# Letter recognition dataset
letter_target_alp=np.array(letter_data)[:,0]
letter_target=np.array([ord(i)-64 for i in letter_target_alp])
letter_input=np.array(letter_data)[:,1:]

idx = np.arange(letter_target.shape[0])
np.random.shuffle(idx)

letter_target=letter_target[idx]
letter_input=letter_input[idx]

num_class=26
letter_class= np.arange(1,num_class+1)
```

Converted alphabet class to corresponding int value for training.

(8) SVM (with HW1 dataset)

```
'''SVM'''
g=1/hw1_input.shape[1]
svm_model=SVC(gamma=g, kernel='rbf')
svm_model.fit(hw1_input, hw1_target)
print('SVM: ',cross_val_score(svm_model, hw1_input, hw1_target, cv=5).mean())
print('SVM test w/ TER score: ',cross_val_score(svm_model, hw1_input, hw1_target, cv=5, scoring=make_scorer(TER)).mean())
print('₩n')
```

(9) TER-RM (with HW1 dataset)

```
'''TER-RM'''
orders=[2,3,4,5]
for r in orders:
    hw1_rm_input= RM_data(hw1_input, r)
    rm_model=LinearRegression()
    rm_model.fit(hw1_rm_input, hw1_target)
    hw1_pred=rm_model.predict(hw1_rm_input)

    print('order: ',r)
    print('TER-RM: ',cross_val_score(rm_model, hw1_rm_input, hw1_target, cv=5).mean())
    print('TER-RM w/ TER score: ',cross_val_score(rm_model, hw1_rm_input, hw1_target, cv=5, scoring=make_scorer(TER)).mean())
print('₩n')
```

(10) Linear Regression (with HW1 dataset)

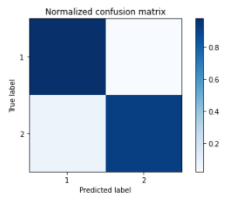
```
'''Linear Regression'''
lr_model=LinearRegression()
lr_model.fit(hw1_input, hw1_target)
print('Linear Regression: ',cross_val_score(lr_model, hw1_input, hw1_target, cv=5).mean())
print('Linear Regression w/ TER score: ',cross_val_score(lr_model, hw1_input, hw1_target, cv=5, scoring=make_scorer(TER)).mean())
print('₩n')
```

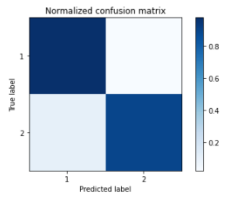
(11) Plot each confusion matrix (with HW1 dataset)

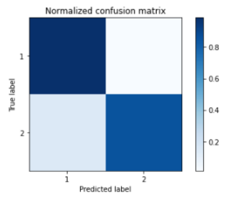
```
# Plot normalized confusion matrix
hw1_pred=svm_model.predict(hw1_input)
plot_confusion_matrix(hw1_target, hw1_pred, classes=hw1_class, normalize=True)
plt.show()

hw1_pred=rm_model.predict(hw1_rm_input)
hw1_pred=preds2labels(hw1_class, hw1_pred)
plot_confusion_matrix(hw1_target, hw1_pred, classes=hw1_class, normalize=True)
plt.show()

hw1_pred=lr_model.predict(hw1_input)
hw1_pred=preds2labels(hw1_class, hw1_pred)
plot_confusion_matrix(hw1_target, hw1_pred, classes=hw1_class, normalize=True)
plt.show()
```

(12) Output (with HW1 dataset)

```
SVM:  0.945
SVM test w/ TER score:  0.12249999999999998


order:  2
TER-RM:  0.7357455724805408
TER-RM w/ TER score:  0.1488235547887657
order:  3
TER-RM:  0.7728969172852749
TER-RM w/ TER score:  0.1490953347606184
order:  4
TER-RM:  0.7736386877322958
TER-RM w/ TER score:  0.13830326715627164
order:  5
TER-RM:  0.7077184182208169
TER-RM w/ TER score:  0.14031086694621347


Linear Regression:  0.70470630682336
Linear Regression w/ TER score:  0.15620314794628165
```







(13) Experiments (with wine dataset)

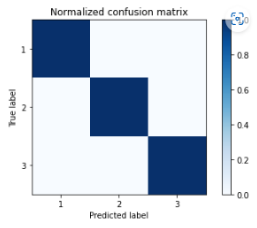
```
'''SVM'''
# Study
for g in (0.01,0.1, 0.3, 1):
    print('g: {}, score: {}'.format(g, cross_val_score(SVC(gamma=g, kernel='rbf'), wine_input, wine_target, cv=5).mean()))
print('₩n')


g=1/wine_input.shape[1]
svm_model=SVC(gamma=g, kernel='rbf')
svm_model.fit(wine_input, wine_target)
print('SVM: ',cross_val_score(svm_model, wine_input, wine_target, cv=5).mean())
print('SVM test w/ TER score: ',cross_val_score(svm_model, wine_input, wine_target, cv=5, scoring=make_scorer(TER)).mean())
print('₩n')



'''TER-RM'''
orders=[2,3,4,5]
for r in orders:
    wine_rm_input= RM_data(wine_input, r)
    rm_model=LinearRegression()
    rm_model.fit(wine_rm_input, wine_target)
    wine_pred=rm_model.predict(wine_rm_input)

    print('order: ',r)
    print('TER-RM: ',cross_val_score(rm_model, wine_rm_input, wine_target, cv=5).mean())
    print('TER-RM w/ TER score: ',cross_val_score(rm_model, wine_rm_input, wine_target, cv=5, scoring=make_scorer(TER)).mean(
print('₩n')



'''Linear Regression'''
lr_model=LinearRegression()
lr_model.fit(wine_input, wine_target)
print('Linear Regression: ',cross_val_score(lr_model, wine_input, wine_target, cv=5).mean())
print('Linear Regression w/ TER score: ',cross_val_score(lr_model, wine_input, wine_target, cv=5, scoring=make_scorer(TER)).m
print('₩n')
```

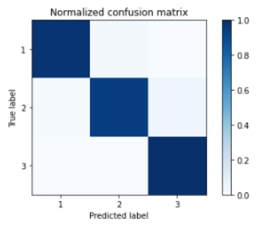
(14) Plot each confusion matrix (with wine dataset)

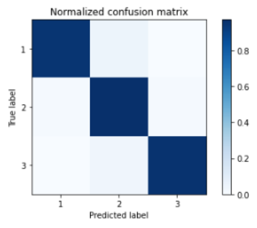
```python
# Plot normalized confusion matrix
wine_pred=svm_model.predict(wine_input)
plot_confusion_matrix(wine_target, wine_pred, classes=wine_class, normalize=True)
plt.show()

wine_pred=rm_model.predict(wine_rm_input)
wine_pred=preds2labels(wine_class, wine_pred)
plot_confusion_matrix(wine_target, wine_pred, classes=wine_class, normalize=True)
plt.show()

wine_pred=lr_model.predict(wine_input)
wine_pred=preds2labels(wine_class, wine_pred)
plot_confusion_matrix(wine_target, wine_pred, classes=wine_class, normalize=True)
plt.show()
```

(15) Output (with wine dataset)

```
SVM:  0.42714285714285716
SVM test w/ TER score:  1.906262626262626


order:  2
TER-RM:  0.6514048116906539
TER-RM w/ TER score:  0.3218355500708442
order:  3
TER-RM:  0.35552212169050834
TER-RM w/ TER score:  0.6061800127976598
order:  4
TER-RM:  0.14672521912329875
TER-RM w/ TER score:  0.6223270030622972
order:  5
TER-RM:  -0.7505600963510162
TER-RM w/ TER score:  0.6994083367612779


Linear Regression:  0.8623356721876723
Linear Regression w/ TER score:  0.17230380730380732
```







(16) Experiments (with letter dataset)

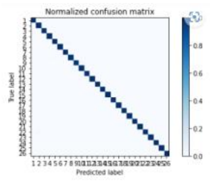
```
'''SVM'''
g=1/letter_input.shape[1]
svm_model=SVC(gamma=g, kernel='rbf')
svm_model.fit(letter_input, letter_target)
print('SVM: ',cross_val_score(svm_model, letter_input, letter_target, cv=5).mean())
print('SVM test w/ TER score: ',cross_val_score(svm_model, letter_input, letter_target, cv=5, scoring=make_scorer(TER)).mean())
print('₩n')



'''TER-RM'''
orders=[2,3,4,5]
for r in orders:
    letter_rm_input= RM_data(letter_input, r)
    rm_model=LinearRegression()
    rm_model.fit(letter_rm_input, letter_target)
    letter_pred=rm_model.predict(letter_rm_input)

    print('order: ',r)
    print('TER-RM: ',cross_val_score(rm_model, letter_rm_input, letter_target, cv=5).mean())
    print('TER-RM w/ TER score: ',cross_val_score(rm_model, letter_rm_input, letter_target, cv=5, scoring=make_scorer(TER)).mean())
print('₩n')



'''Linear Regression'''
lr_model=LinearRegression()
lr_model.fit(letter_input, letter_target)
print('Linear Regression: ',cross_val_score(lr_model, letter_input, letter_target, cv=5).mean())
print('Linear Regression w/ TER score: ',cross_val_score(lr_model, letter_input, letter_target, cv=5, scoring=make_scorer(TER)).mean())
print('₩n')
```

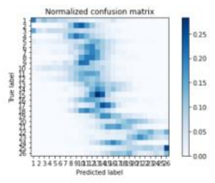
(17) Plot each confusion matrix (with wine dataset)

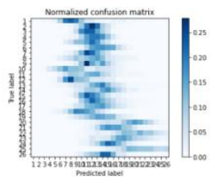
```
# Plot normalized confusion matrix
letter_pred=svm_model.predict(letter_input)
plot_confusion_matrix(letter_target, letter_pred, classes=letter_class, normalize=True)
plt.show()

letter_pred=rm_model.predict(letter_rm_input)
letter_pred=preds2labels(letter_class, letter_pred)
plot_confusion_matrix(letter_target, letter_pred, classes=letter_class, normalize=True)
plt.show()

letter_pred=lr_model.predict(letter_input)
letter_pred=preds2labels(letter_class, letter_pred)
plot_confusion_matrix(letter_target, letter_pred, classes=letter_class, normalize=True)
plt.show()
```

(18) Output (with wine dataset)

```
SVM:  0.97395
SVM test w/ TER score:  0.6828261018631844


order:  2
TER-RM:  0.41789360831271444
TER-RM w/ TER score:  24.276696800575685
order:  3
TER-RM:  0.439479531285872
TER-RM w/ TER score:  24.28304320422092
order:  4
TER-RM:  0.48185795550433394
TER-RM w/ TER score:  24.110067195475505
order:  5
TER-RM:  0.48996220252119266
TER-RM w/ TER score:  24.129043075865802


Linear Regression:  0.28466783012272723
Linear Regression w/ TER score:  24.76372366413116
```


Normalized confusion matrix
True label
Predicted label


Normalized confusion matrix
True label
Predicted label


Normalized confusion matrix
True label
Predicted label