4_MSMWD_Training_and_Evaluation

November 28, 2021

1 Training and Evaluation

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import log_loss, confusion_matrix
from sklearn.calibration import CalibratedClassifierCV
import warnings
warnings.filterwarnings("ignore")
from xgboost import XGBClassifier
```

[2]:
```
cv_result = pd.read_csv("CV_Results.csv")
cv_result.sort_values(by='cv_loss').head(10)
```

[2]:

	Data	cv_loss	train_loss	colsample_bynode	learning_rate	max_depth	n_estimators	subsample
0	Set2: XGBClassifier Non Zero Features	0.017920	0.010524	0.6	1.0	3	359	0.7
1	Set2: XGBClassifier Non Zero Features	0.018279	0.009558	0.6	0.5	3	42	0.7
2	Set2: XGBClassifier Non Zero Features	0.018281	0.009455	0.6	0.5	3	175	0.7
3	Set2: XGBClassifier Non Zero Features	0.018281	0.009455	0.6	0.5	3	175	0.7
4	Set2: XGBClassifier Non Zero Features	0.018291	0.009466	0.6	0.5	3	112	0.7
5	Set1: K Best: 800px + 2000 Bigram bytes + othe…	0.018313	0.011606	0.5	1.0	7	337	0.7
6	Set1: K Best: 800px + 2001 Bigram bytes + othe…	0.018338	0.011615	0.5	1.0	7	263	0.7
7	Set2: XGBClassifier Non Zero Features	0.018371	0.009127	0.7	0.5	7	200	0.7
8	Set2: XGBClassifier Non Zero Features	0.018391	0.009128					
9	Set2: XGBClassifier Non Zero Features	0.018391	0.009128					

```
8                0.7             0.5           7            150         0.7
9                0.7             0.5           7            150         0.7

                    model
0  xgboost.XGBClassifier
1  xgboost.XGBClassifier
2  xgboost.XGBClassifier
3  xgboost.XGBClassifier
4  xgboost.XGBClassifier
5  xgboost.XGBClassifier
6  xgboost.XGBClassifier
7  xgboost.XGBClassifier
8  xgboost.XGBClassifier
9  xgboost.XGBClassifier
```

[3]:
```
best_hyperparams = {'colsample_bynode':0.6, 'learning_rate':1,'max_depth':
 ↪3,'n_estimators':359,'subsample':0.7}
x = np.load('feature_set2.npy')
y = np.load('target.npy')
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,␣
 ↪random_state=11)
x_train, x_cv, y_train, y_cv = train_test_split(x_train,y_train,test_size=0.25,␣
 ↪random_state=11)
print('Shape of training data 1', x_train.shape)
```

```
Shape of training data 1 (6520, 620)
```

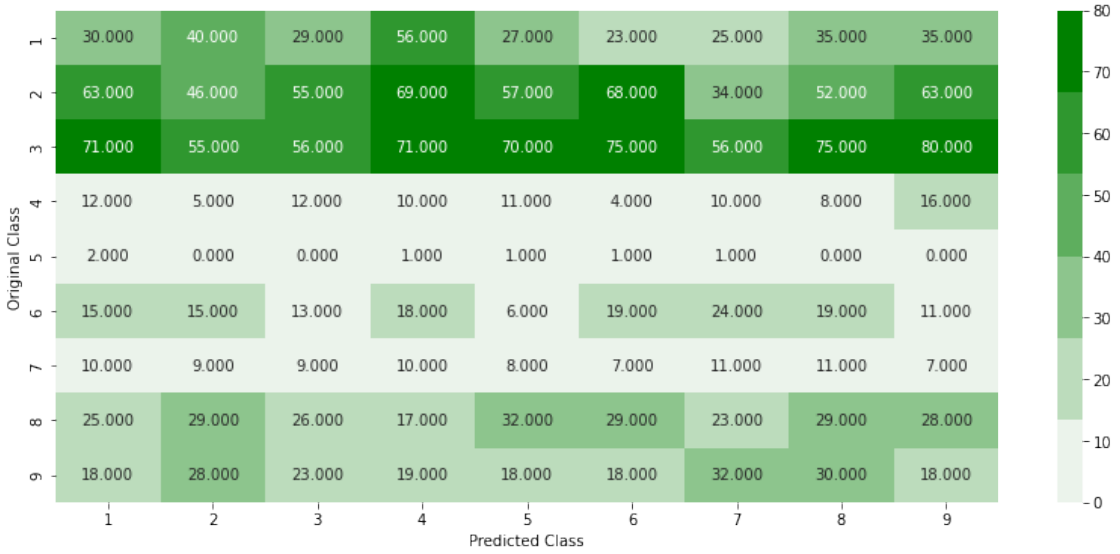
[4]:
```
# credits for plot_confusion_matrix funtion: reference notebook
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    print("Number of misclassified points ",(len(test_y)-np.trace(C))/
 ↪len(test_y)*100)
    A =(((C.T)/(C.sum(axis=1))).T)
    B =(C/C.sum(axis=0))
    labels = [1,2,3,4,5,6,7,8,9]
    cmap=sns.light_palette("green")
    # representing A in heatmap format
    print("-"*25, "Confusion matrix", "-"*25)
    plt.figure(figsize=(14,6))
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels,␣
 ↪yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
    print("-"*25, "Precision matrix", "-"*25)
    plt.figure(figsize=(14,6))
```

```
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels,␣
↪yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
    print("Sum of columns in precision matrix",B.sum(axis=0))
    # representing B in heatmap format
    print("-"*25, "Recall matrix"    , "-"*25)
    plt.figure(figsize=(14,6))
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels,␣
↪yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
    print("Sum of rows in precision matrix",A.sum(axis=1))
```

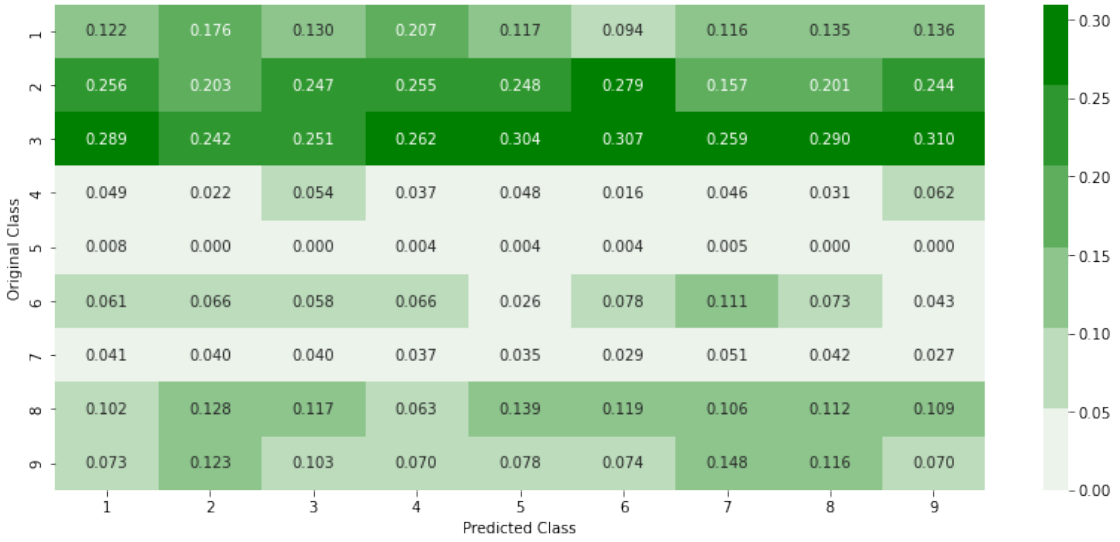
2 Dummy Classifier

[5]:
```
from sklearn.dummy import DummyClassifier
dummy_clf = DummyClassifier(strategy="uniform")
dummy_clf.fit(x_train, y_train)
dummy_train_loss = log_loss(y_train,dummy_clf.predict_proba(x_train))
dummy_cv_loss = log_loss(y_cv,dummy_clf.predict_proba(x_cv))
dummy_test_loss = log_loss(y_test,dummy_clf.predict_proba(x_test))
plot_confusion_matrix(y_test, dummy_clf.predict(x_test))
print('Train loss = ', dummy_train_loss)
print('cv loss = ', dummy_cv_loss)
print('test loss = ', dummy_test_loss)
```

```
Number of misclassified points  89.88040478380864
------------------------- Confusion matrix --------------------------
```

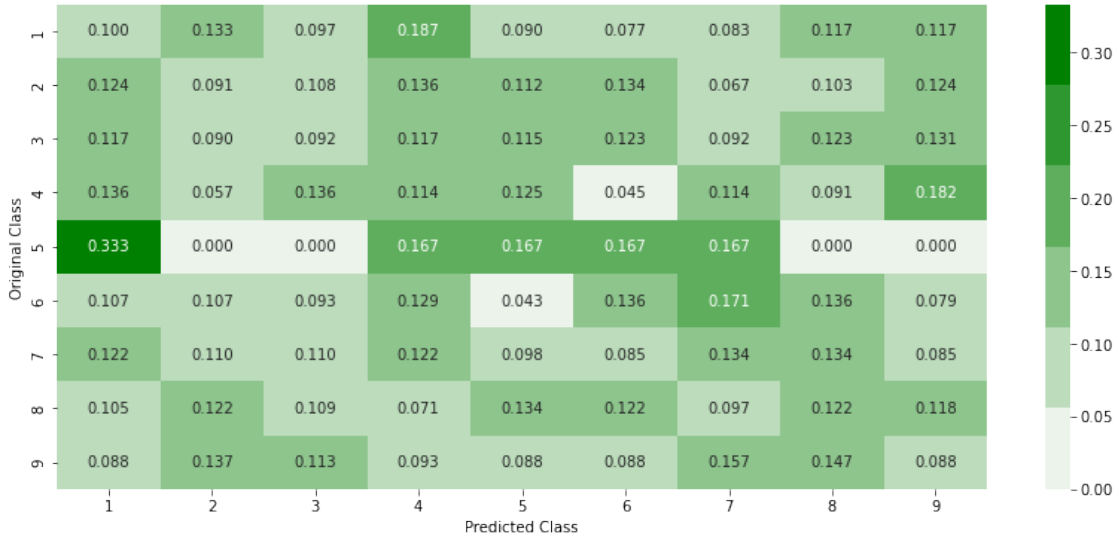


---------- Precision matrix ----------



Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]

---------- Recall matrix ----------



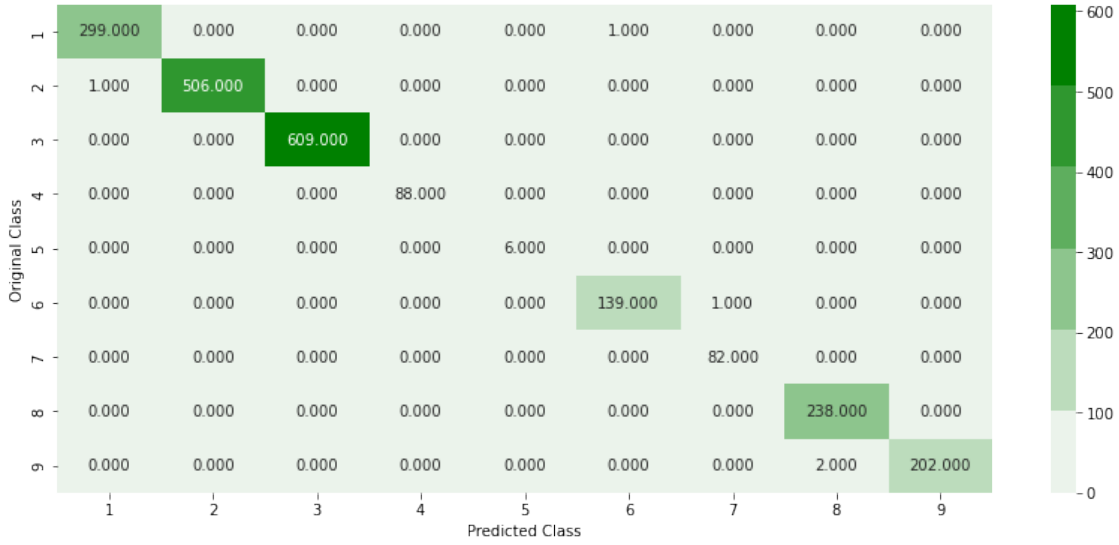

```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
Train loss =  2.197224577336219
cv loss =  2.1972245773362196
test loss =  2.1972245773362196
```

3 XGBClassifier on Set2 - Non zero feature importances

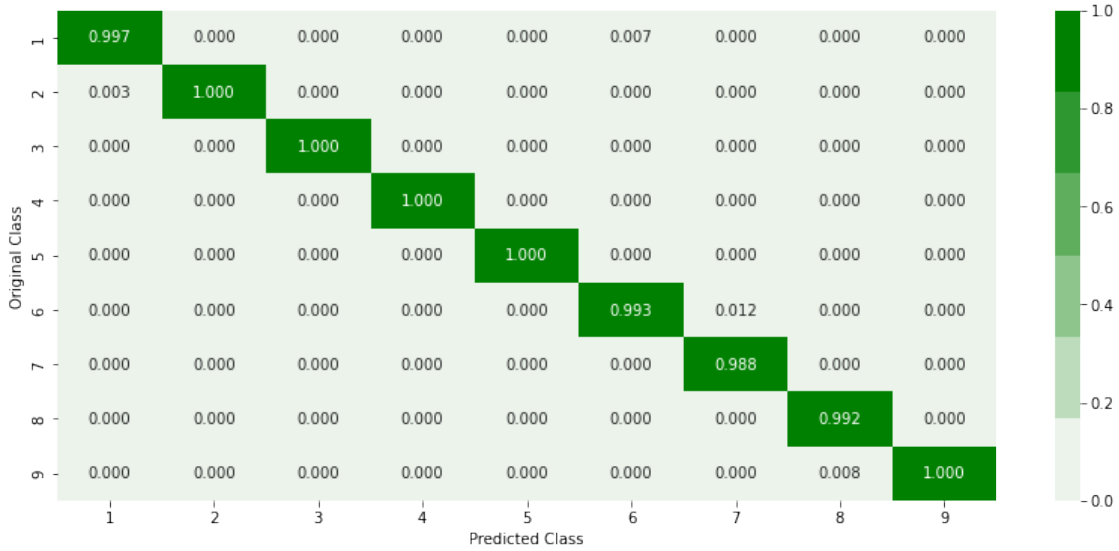
[6]:
```
print('XGBClassifier using following Hyperparameters..','\n','-'*50)
for k,v in best_hyperparams.items(): print(k,' = ',v)
print('-'*50,'\n\n')
clf = XGBClassifier(eval_metric='mlogloss',**best_hyperparams)
clf.fit(x_train,y_train.ravel())
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(x_train, y_train.ravel())
train_loss = log_loss(y_train,sig_clf.predict_proba(x_train))
cv_loss = log_loss(y_cv,sig_clf.predict_proba(x_cv))
test_loss = log_loss(y_test,sig_clf.predict_proba(x_test))
plot_confusion_matrix(y_test, sig_clf.predict(x_test))
print('Train loss = ', train_loss)
print('cv loss = ', cv_loss)
print('test loss = ', test_loss)
```

```
XGBClassifier using following Hyperparameters..
 ----------------------------------------------------
colsample_bynode  =  0.6
learning_rate  =  1
max_depth  =  3
n_estimators  =  359
```

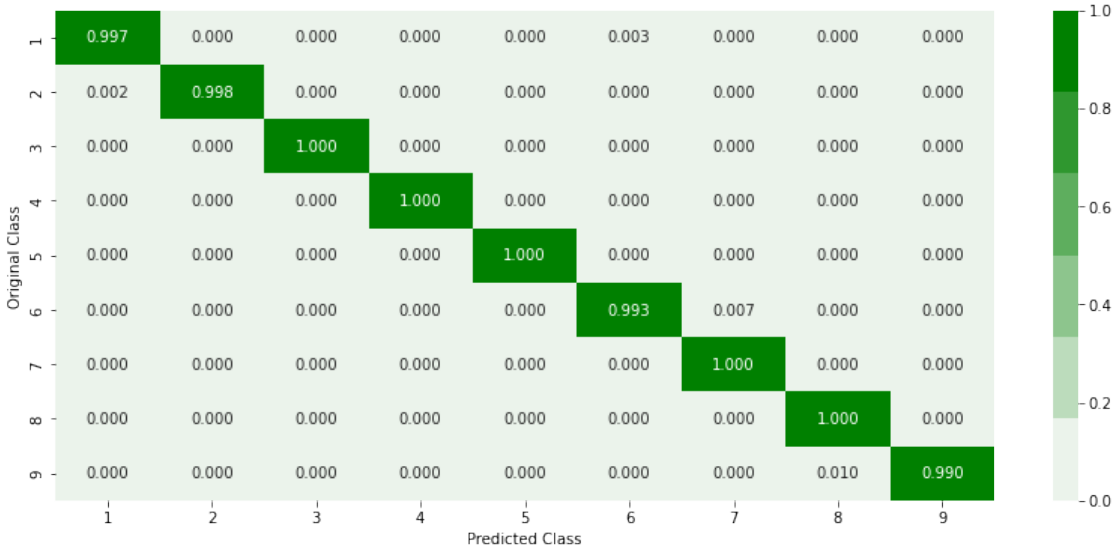
```
subsample   =  0.7
-----------------------------------------------------


Number of misclassified points  0.22999080036798528
------------------------- Confusion matrix --------------------------
```




```
------------------------- Precision matrix --------------------------
```




```
Sum of columns in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
------------------------- Recall matrix --------------------------
```




```
Sum of rows in precision matrix [1. 1. 1. 1. 1. 1. 1. 1. 1.]
Train loss =  0.010523712028967356
cv loss =  0.017919715350828213
test loss =  0.019941851319342556
```