Custom ensemble part 1

In [1]:
```
#import libraries....

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix,f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier,LogisticRegression
from sklearn.tree import DecisionTreeClassifier
import lightgbm as lgb
import xgboost as xgb
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import RandomOverSampler
from sklearn.svm import SVC
import datetime

import warnings
warnings.filterwarnings("ignore")
```

Multi class classification among 1,2,3,4

In [2]:
```
#load the data with all created features
data = pd.read_csv("data_with_advanced_features.csv")
data.drop("Unnamed: 0", inplace=True, axis=1)
```

In [3]:
```
#label encoding of seller_id
label = LabelEncoder()
seller = label.fit_transform(data.seller_id)
data["seller_id"] = seller


#label encoding of product id
label = LabelEncoder()
product = label.fit_transform(data.product_id)
data["product_id"] = product
```

In [4]:
```
#let us drop some of the columns which are not needed
data.drop(["order_id","customer_id","order_status","order_approved_at","order_delivered_carrier_date",
           "order_estimated_delivery_date","order_delivered_customer_date","customer_unique_id","customer_
          "seller_city","shipping_limit_date","seller_state","customer_state",
           "order_purchase_timestamp"],inplace=True,axis=1)
```

In [5]:
```
#shape of the data after dropping unnecessary columns
data.shape
```

Out[5]:
```
(113105, 57)
```

1.1 Stratified Splitting

In [6]:
```
#data only with review_score 1,2,3,4
data = data[data["review_score"]!=5]
Y = data["review_score"]
X = data
```

Train Test Split

In [7]:
```
#train test split with test size 20% and 85% of data as train
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=10)
```

In [8]:
```
print("Dimensions of the splitted data :")
print("Train: ",x_train.shape,y_train.shape)
print("Test: ",x_test.shape,y_test.shape)
```

```
Dimensions of the splitted data :
Train:  (38774, 57) (38774,)
Test:  (9694, 57) (9694,)
```

In [9]:
```
#check the distribution of each class in train,test as well as original data
print("% Distribution of class labels in the total data :")
print(round(data["review_score"].value_counts(normalize=True)*100,2))
print("*"*50)

print("% Distribution of class labels in the train data :")
print(round(x_train["review_score"].value_counts(normalize=True)*100,2))
print("*"*50)

print("% Distribution of class labels in the test data :")
print(round(x_test["review_score"].value_counts(normalize=True)*100,2))
print("*"*50)
```

```
% Distribution of class labels in the total data :
4    44.82
1    27.32
3    19.75
2     8.11
Name: review_score, dtype: float64
**************************************************
% Distribution of class labels in the train data :
4    44.82
1    27.32
3    19.75
2     8.11
Name: review_score, dtype: float64
**************************************************
% Distribution of class labels in the test data :
4    44.82
1    27.32
3    19.75
2     8.11
Name: review_score, dtype: float64
**************************************************
```

- Distribution of each class label is same in train.test and original data.

let us use simple CountVectorizer for categorical data.

1.2 Featurization:

1.2.1 Vectorization of categorical variables:

1. payment_type

In [11]:
```
#payment_type
vec = CountVectorizer()

vec.fit(x_train["payment_type"].values)

x_tr_pay_type = vec.transform(x_train.payment_type.values)
x_te_pay_type = vec.transform(x_test.payment_type.values)

print(x_tr_pay_type.shape)
print(x_te_pay_type.shape)
```

```
(38774, 4)
(9694, 4)
```

2. order_item_id

In [13]:
```
#order_item_id

x_train.order_item_id = x_train.order_item_id.astype(str)
x_test.order_item_id = x_test.order_item_id.astype(str)

vec = CountVectorizer(vocabulary=range(1,22))

vec.fit(x_train["order_item_id"])

x_tr_id = vec.transform(x_train.order_item_id)
x_te_id = vec.transform(x_test.order_item_id)

print(x_tr_id.shape)

print(x_te_id.shape)
```

```
(38774, 21)
(9694, 21)
```

3. product_category_name

In [14]:
```
#product_category_name
vec = CountVectorizer()

vec.fit(x_train["product_category_name"].values)

x_tr_cat = vec.transform(x_train.product_category_name.values)
#x_cv_cat = vec.transform(x_cv.product_category_name.values).toarray()
x_te_cat = vec.transform(x_test.product_category_name.values)

print(x_tr_cat.shape)
#print(x_cv_cat.shape)
print(x_te_cat.shape)
```

```
(38774, 72)
(9694, 72)
```

1.2.2 Binary features

In [15]:
```
x_tr_same_state = x_train.same_state.values.reshape(-1,1)
x_te_same_state = x_test.same_state.values.reshape(-1,1)

x_tr_same_city = x_train.same_city.values.reshape(-1,1)
x_te_same_city = x_test.same_city.values.reshape(-1,1)

x_tr_late_shipping = x_train.late_shipping.values.reshape(-1,1)
x_te_late_shipping = x_test.late_shipping.values.reshape(-1,1)

x_tr_high_freight = x_train.high_freight.values.reshape(-1,1)
x_te_high_freight = x_test.high_freight.values.reshape(-1,1)
```

1.2.3 Numrical features

In [16]:

```
def scaling(train_data,test_data):
    """This function will standardize the numerical data"""
    norm = StandardScaler()

    norm.fit(train_data.values)

    x_tr_num = norm.transform(train_data.values)
    x_te_num = norm.transform(test_data.values)


    return x_tr_num,x_te_num
```

In [17]:

```
data.columns
```

Out[17]:

```
Index(['payment_sequential', 'payment_type', 'payment_installments',
       'payment_value', 'review_score', 'zip_code_prefix_customer',
       'lat_customer', 'lng_customer', 'product_id', 'product_name_lenght',
       'product_description_lenght', 'product_photos_qty', 'product_weight_g',
       'product_length_cm', 'product_height_cm', 'product_width_cm',
       'order_item_id', 'seller_id', 'price', 'freight_value',
       'zip_code_prefix_seller', 'lat_seller', 'lng_seller',
       'product_category_name', 'estimated_time', 'actual_time',
       'diff_actual_estimated', 'diff_purchased_approved',
       'diff_purchased_courrier', 'distance', 'speed', 'same_state',
       'same_city', 'late_shipping', 'high_freight', 'seller_share',
       'bs_share', 'cust_share', 'bu_share', 'similarity',
       'seller_category_share', 'cat_seller_share', 'cust_category_share',
       'cat_cust_share', 'similarity_using_cat', 'size', 'delivery_day',
       'delivery_date', 'delivery_month', 'delivery_hour', 'purchased_day',
       'purchased_date', 'purchased_month', 'purchased_hour',
       'num_of_customers_for_seller', 'num_of_sellers_for_cust',
       'total_order_for_seller'],
      dtype='object')
```

In [18]:

```
#data to be standardized
tr = x_train[["payment_sequential","payment_installments","payment_value","seller_id","product_id","seller
              "bs_share","cust_share",
          "lat_customer","lng_customer","lat_seller","lng_seller","product_name_lenght","product_descripti
           "product_photos_qty","product_weight_g","size","price","delivery_day","delivery_date","delivery
              "delivery_hour","purchased_day","purchased_date","purchased_month","purchased_hour","num_of_
              "num_of_sellers_for_cust","total_order_for_seller",
           "freight_value","estimated_time","actual_time","diff_actual_estimated","diff_purchased_approved
           "diff_purchased_courrier","distance","speed","similarity","similarity_using_cat"]]

te = x_test[["payment_sequential","payment_installments","payment_value","seller_id","product_id","seller_
              "bs_share","cust_share",
          "lat_customer","lng_customer","lat_seller","lng_seller","product_name_lenght","product_descripti
           "product_photos_qty","product_weight_g","size","price","delivery_day","delivery_date","delivery
              "delivery_hour","purchased_day","purchased_date","purchased_month","purchased_hour","num_of_
              "num_of_sellers_for_cust","total_order_for_seller",
           "freight_value","estimated_time","actual_time","diff_actual_estimated","diff_purchased_approved
           "diff_purchased_courrier","distance","speed","similarity","similarity_using_cat"]]
```

In [19]:

```
#standardizing
x_tr_num,x_te_num = scaling(tr,te)
```

In [20]:

```
from scipy.sparse import hstack
#horizontal stacking of all the features
train = hstack((x_tr_pay_type,x_tr_id,x_tr_cat,x_tr_num,x_tr_same_state,
                   x_tr_same_city,x_tr_late_shipping,x_tr_high_freight)).toarray()

test = hstack((x_te_pay_type,x_te_id,x_te_cat,x_te_num,x_te_same_state,
                  x_te_same_city,x_te_late_shipping,x_te_high_freight)).toarray()
```

In [21]:
```
#shape of final train and test data
print("Shape of train data : ",train.shape)
print("Shape of test data : ",test.shape)
```

```
Shape of train data :  (38774, 141)
Shape of test data :  (9694, 141)
```

In [22]:
```
#reset the index of target variable
y_trains = y_train.reset_index()
y_train = y_trains["review_score"]

y_tests = y_test.reset_index()
y_test = y_tests["review_score"]
```

Plotting Confusion matrix

In [23]:
```
# This function plots the confusion matrices given y_i, y_i_hat.
def plot_confusion_matrix(test_y, predict_y):
    """This function will plot confusion matrix, precision matrix and recall matrix"""
    C = confusion_matrix(test_y, predict_y)
    A =(((C.T)/(C.sum(axis=1))).T)
    B =(C/C.sum(axis=0))

    labels = [1,2,3,4]
    # representing A in heatmap format
    print("-"*20, "Confusion matrix", "-"*20)
    plt.figure(figsize=(16,7))
    sns.heatmap(C, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()

    print("-"*20, "Precision matrix (Columm Sum=1)", "-"*20)
    plt.figure(figsize=(16,7))
    sns.heatmap(B, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()

    # representing B in heatmap format
    print("-"*20, "Recall matrix (Row sum=1)", "-"*20)
    plt.figure(figsize=(16,7))
    sns.heatmap(A, annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
```

Custom Ensemble Model

Train data will be splitted into 50-50 as d1 and d2 sets. From d1 set randomly sample the points with replacement and take say k samples. Train k models with k samples. And predict d2 set by passing d2 to each f the k models. Now we get k predictions.

Now build meta classifier using k predictions as input, that means train the meta classifier using k predictions. While training the target variable should be review_score in d2 set. The output of the meta classifier will be considered as the predicted reveiw_score.

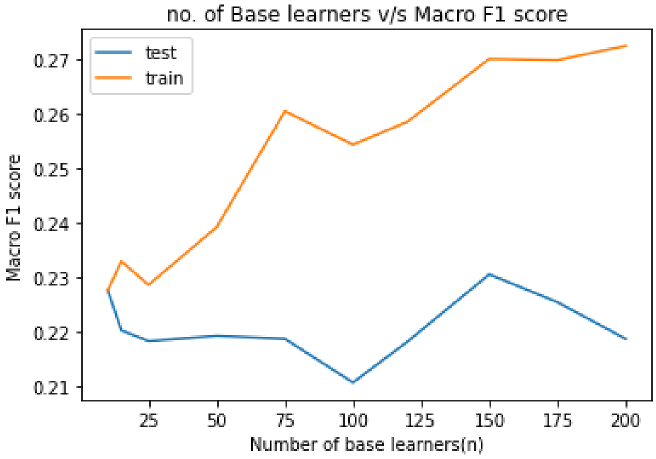
In [24]:
```
def custom_ensemble(x_tr,y_tr,x_te,n_estimators,estimator,meta_clf):
    """This function creates the custom ensemble model and returns predicted  target variable of test set"

    ########### SPlitting train data into 50-50 as d1 and d2 ############
    kf = StratifiedKFold(n_splits=2)

    d1 = x_tr[list(kf.split(x_tr,y_tr))[1][0]]
    d1_y = y_tr[list(kf.split(x_tr,y_tr))[1][0]]

    d2 = x_tr[list(kf.split(x_tr,y_tr))[1][1]]
    d2_y = y_tr[list(kf.split(x_tr,y_tr))[1][1]]
    ######################################################################
    d1_y = np.array(d1_y)
    d2_y = np.array(d2_y)
    ######################################################################
    ### Creating base learners and training them using samples of d1 ####

    models=[]

    for i in range(n_estimators):
        ind = np.random.choice(19387,size=(20000),replace=True)
        sample = d1[ind]
        sample_y = d1_y[ind]

        estimator.fit(sample,sample_y)
        models.append(estimator)

    ########### Predictions from base learners for d2 set ###############
    predictions = []
    for model in models:

        pred = model.predict(d2)
        predictions.append(pred)

    predictions = np.array(predictions).reshape(-1,n_estimators)

    ########## meta classifier on predictions of base learners ##########

    meta_clf.fit(predictions,d2_y)
    train_pred = meta_clf.predict(predictions)

    ######################################################################
    ######################## TEST SET ###################################

    pred_test = []
    for model in models:
        pred_test.append(model.predict(test))

    pred_test = np.array(pred_test).reshape(-1,n_estimators)
    test_y_predicted = meta_clf.predict(pred_test)

#### Return train predictions on d2, test predictions and actual labels of d2 ####

    return train_pred,test_y_predicted,d2_y
```

Logistic Regression

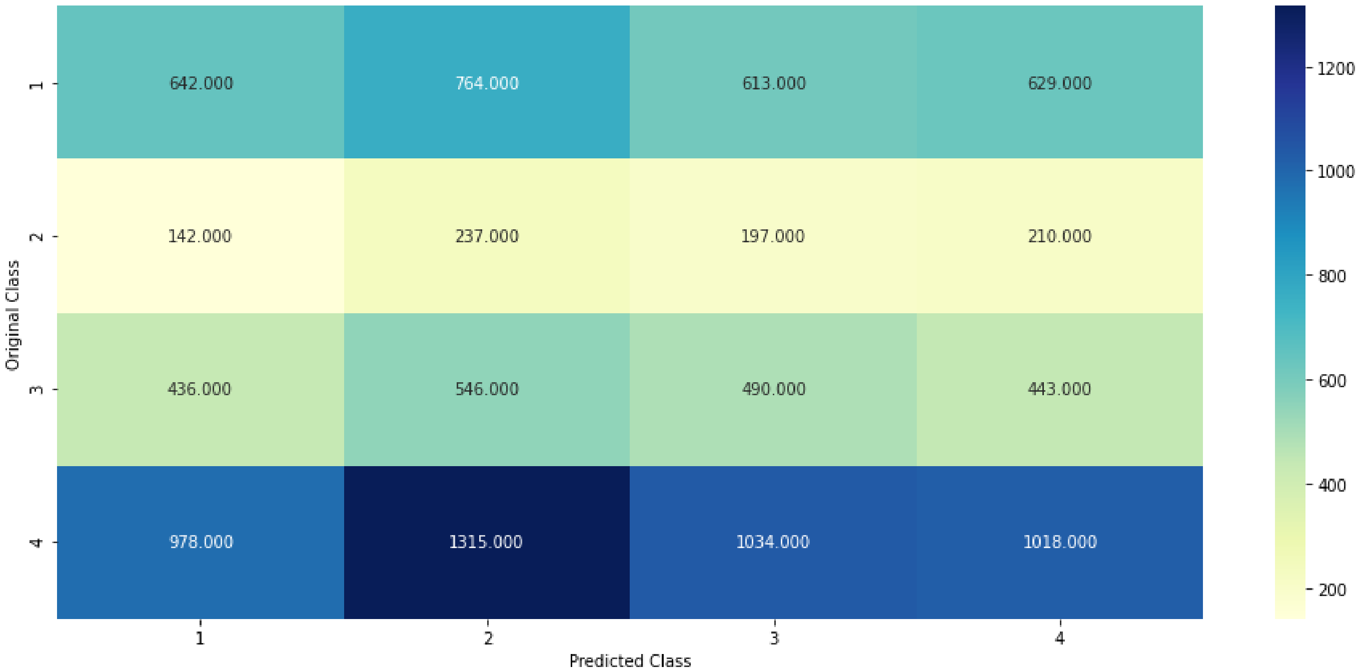
In [25]:
```
n = [10,15,25,50,75,100,120,150,175,200]
test_f1 =[]
train_f1 = []

for i in n:

    train_pred,test_pred,d2_y = custom_ensemble(train,y_train,test,i,LogisticRegression(class_weight="bala
                                               LogisticRegression(class_weight="balanced"))

    train_score = f1_score(d2_y,train_pred,average="macro",labels=[1,2,3,4])
    test_score = f1_score(y_test,test_pred,average="macro",labels=[1,2,3,4])

    train_f1.append(train_score)
    test_f1.append(test_score)

    print("*"*60)
    print("Train Macro F1 score for n_estimator={} is : {}".format(i,train_score))
    print("Test Macro F1 score for n_estimator={} is : {}".format(i,test_score))
    print("*"*60)

plt.plot(n,test_f1,label="test")
plt.plot(n,train_f1,label="train")
plt.legend()
plt.xlabel("Number of base learners(n)")
plt.ylabel("Macro F1 score")
plt.title("no. of Base learners v/s Macro F1 score")
plt.show()

best_n = n[np.argmax(test_f1)]


train_pred,test_pred,d2_y = custom_ensemble(train,y_train,test,best_n,LogisticRegression(class_weight="bal
                                               LogisticRegression(class_weight="balanced"))

train_score = f1_score(d2_y,train_pred,average="macro",labels=[1,2,3,4])
test_score = f1_score(y_test,test_pred,average="macro",labels=[1,2,3,4])

print("*"*60)
print("Train Macro F1 score for n_estimator={} is : {}".format(best_n,train_score))
print("Test Macro F1 score for n_estimator={} is : {}".format(best_n,test_score))
print("*"*60)

plot_confusion_matrix(y_test,test_pred)
```

```
************************************************************
Train Macro F1 score for n_estimator=10 is : 0.2275975085298737
Test Macro F1 score for n_estimator=10 is : 0.22768654176316655
************************************************************
************************************************************
Train Macro F1 score for n_estimator=15 is : 0.23294871450254453
Test Macro F1 score for n_estimator=15 is : 0.2202721123645503
************************************************************
************************************************************
Train Macro F1 score for n_estimator=25 is : 0.22863676451490095
Test Macro F1 score for n_estimator=25 is : 0.21830870779971911
************************************************************
************************************************************
Train Macro F1 score for n_estimator=50 is : 0.2392766892756102
Test Macro F1 score for n_estimator=50 is : 0.2192628608297082
************************************************************
************************************************************
Train Macro F1 score for n_estimator=75 is : 0.2605528481876316
Test Macro F1 score for n_estimator=75 is : 0.21871970743696173
************************************************************
************************************************************
Train Macro F1 score for n_estimator=100 is : 0.25442330900194493
Test Macro F1 score for n_estimator=100 is : 0.2106638150334702
************************************************************
************************************************************
Train Macro F1 score for n_estimator=120 is : 0.25863560405190616
Test Macro F1 score for n_estimator=120 is : 0.21821822503158228
************************************************************
************************************************************
Train Macro F1 score for n_estimator=150 is : 0.2701955250061928
Test Macro F1 score for n_estimator=150 is : 0.23056390517919373
************************************************************
```

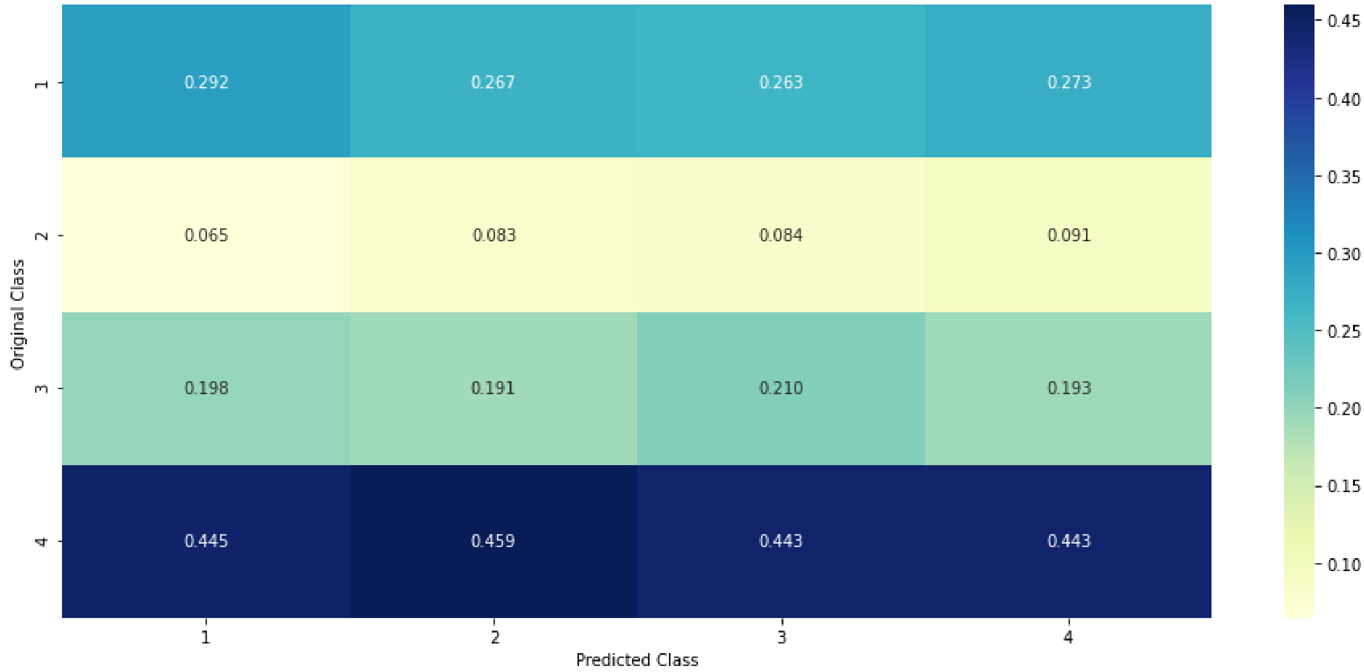
```
************************************************************
Train Macro F1 score for n_estimator=175 is : 0.26995816705602926
Test Macro F1 score for n_estimator=175 is : 0.22545213261796726
************************************************************
************************************************************
Train Macro F1 score for n_estimator=200 is : 0.2725679232221619
Test Macro F1 score for n_estimator=200 is : 0.21870631179230154
************************************************************
```



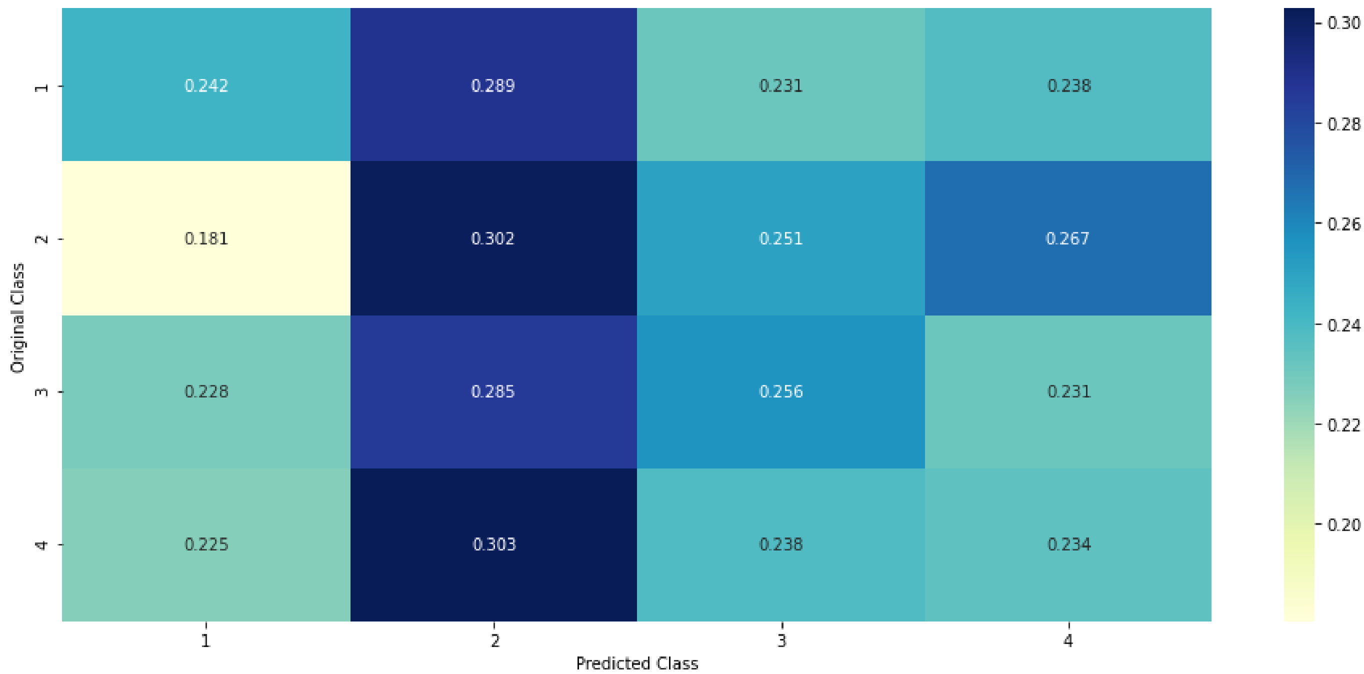

```
************************************************************
Train Macro F1 score for n_estimator=150 is : 0.27057451737250016
Test Macro F1 score for n_estimator=150 is : 0.23298332333913324
************************************************************
-------------------- Confusion matrix --------------------
```




```
-------------------- Precision matrix (Columm Sum=1) --------------------
```




```
-------------------- Recall matrix (Row sum=1) --------------------
```

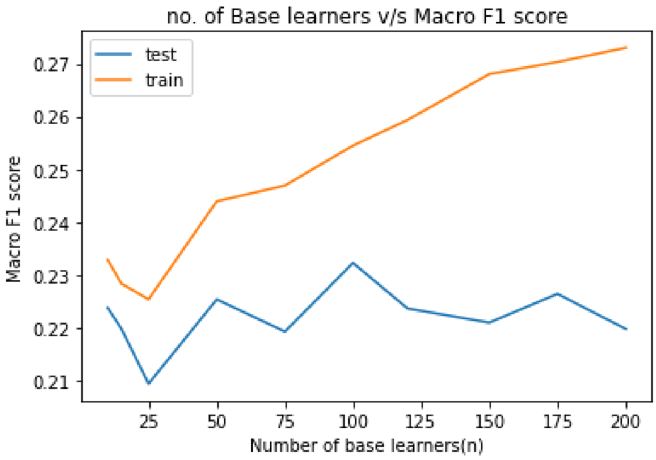


Decision Tree

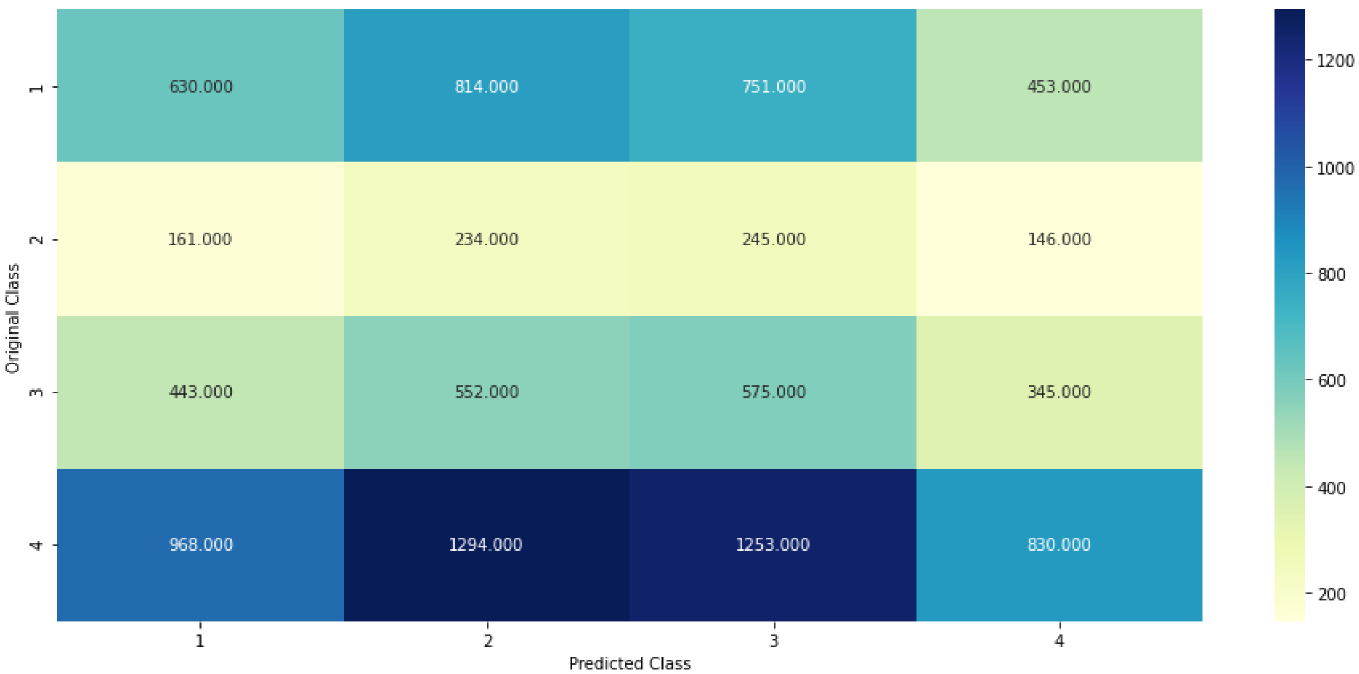
In [26]:
```
n = [10,15,25,50,75,100,120,150,175,200]

test_f1 =[]
train_f1 = []

for i in n:

    train_pred,test_pred,d2_y = custom_ensemble(train,y_train,test,i,DecisionTreeClassifier(class_weight='
                                               LogisticRegression(class_weight="balanced"))

    train_score = f1_score(d2_y,train_pred,average="macro",labels=[1,2,3,4])
    test_score = f1_score(y_test,test_pred,average="macro",labels=[1,2,3,4])

    train_f1.append(train_score)
    test_f1.append(test_score)

    print("*"*60)
    print("Train Macro F1 score for n_estimator={} is : {}".format(i,train_score))
    print("Test Macro F1 score for n_estimator={} is : {}".format(i,test_score))
    print("*"*60)

plt.plot(n,test_f1,label="test")
plt.plot(n,train_f1,label="train")
plt.legend()
plt.xlabel("Number of base learners(n)")
plt.ylabel("Macro F1 score")
plt.title("no. of Base learners v/s Macro F1 score")
plt.show()

best_n = n[np.argmax(test_f1)]


train_pred,test_pred,d2_y = custom_ensemble(train,y_train,test,best_n,DecisionTreeClassifier(class_weight=
                                           LogisticRegression(class_weight="balanced"))

train_score = f1_score(d2_y,train_pred,average="macro",labels=[1,2,3,4])
test_score = f1_score(y_test,test_pred,average="macro",labels=[1,2,3,4])

print("*"*60)
print("Train Macro F1 score for n_estimator={} is : {}".format(best_n,train_score))
print("Test Macro F1 score for n_estimator={} is : {}".format(best_n,test_score))
print("*"*60)

plot_confusion_matrix(y_test,test_pred)
```

```
************************************************************
Train Macro F1 score for n_estimator=10 is : 0.2328363949463364
Test Macro F1 score for n_estimator=10 is : 0.22377371019544653
************************************************************
************************************************************
Train Macro F1 score for n_estimator=15 is : 0.22836614941903877
Test Macro F1 score for n_estimator=15 is : 0.21982933667450555
************************************************************
************************************************************
Train Macro F1 score for n_estimator=25 is : 0.22539016149311641
Test Macro F1 score for n_estimator=25 is : 0.20940018755702305
************************************************************
************************************************************
Train Macro F1 score for n_estimator=50 is : 0.24393772393415414
Test Macro F1 score for n_estimator=50 is : 0.22538102865271586
************************************************************
************************************************************
Train Macro F1 score for n_estimator=75 is : 0.24692644049433876
Test Macro F1 score for n_estimator=75 is : 0.2192767591375876
************************************************************
************************************************************
Train Macro F1 score for n_estimator=100 is : 0.25446914651838204
Test Macro F1 score for n_estimator=100 is : 0.23226096611764402
************************************************************
************************************************************
Train Macro F1 score for n_estimator=120 is : 0.25933394620860856
Test Macro F1 score for n_estimator=120 is : 0.22365362976564235
************************************************************
************************************************************
Train Macro F1 score for n_estimator=150 is : 0.2680157221028927
```

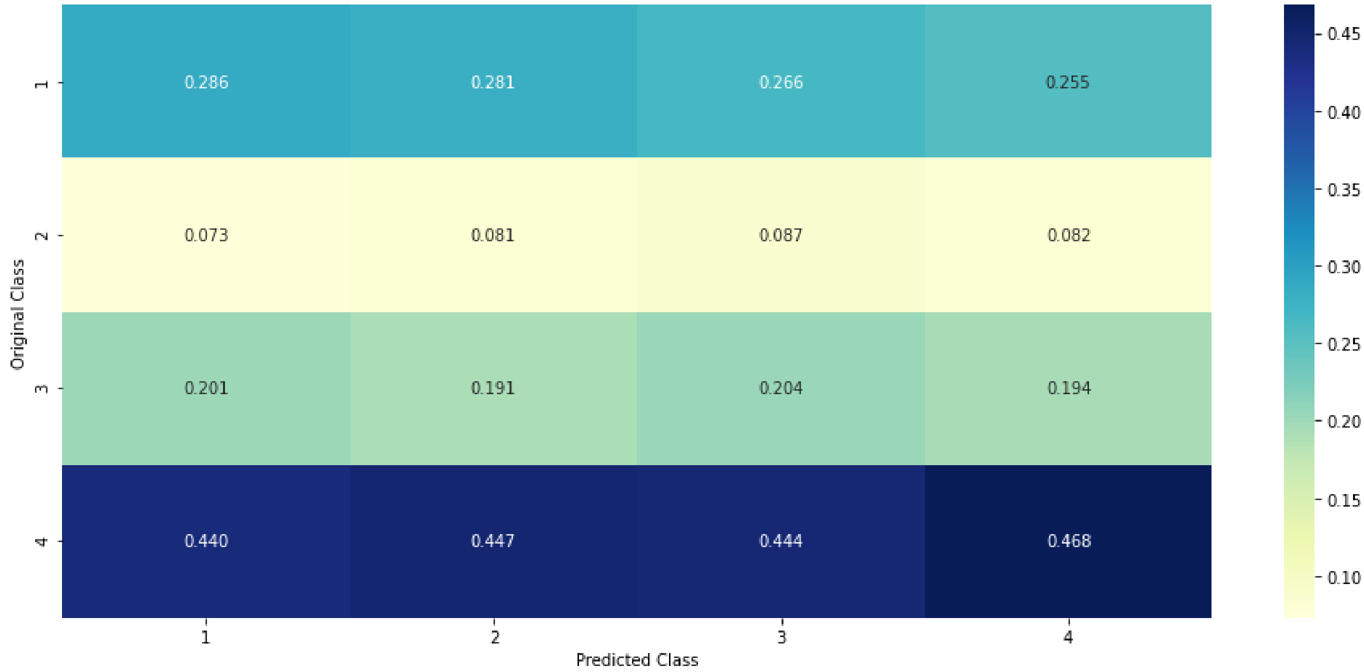
```
Test Macro F1 score for n_estimator=150 is : 0.2209791035025051
************************************************************
************************************************************
Train Macro F1 score for n_estimator=175 is : 0.2703077522768628
Test Macro F1 score for n_estimator=175 is : 0.22642828386923983
************************************************************
************************************************************
Train Macro F1 score for n_estimator=200 is : 0.2729840193776298
Test Macro F1 score for n_estimator=200 is : 0.2197891552924506
************************************************************
```



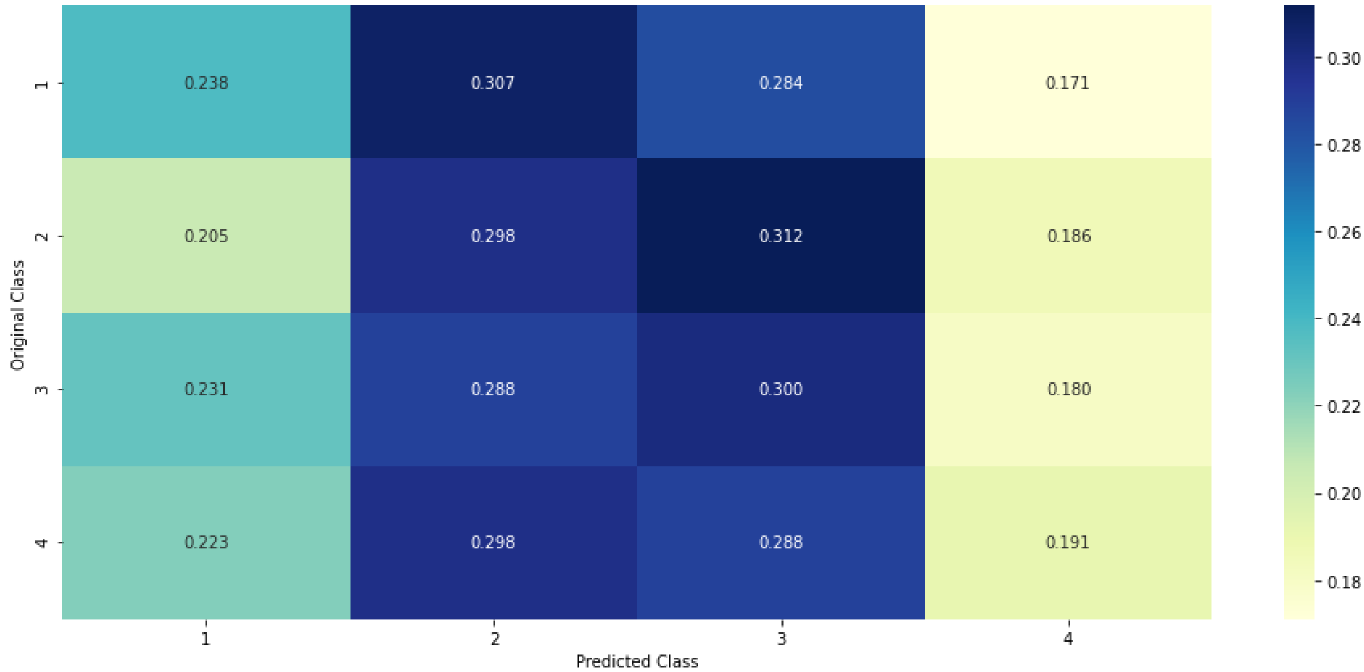

```
************************************************************
Train Macro F1 score for n_estimator=100 is : 0.2554723104438191
Test Macro F1 score for n_estimator=100 is : 0.22523027867967155
************************************************************
-------------------- Confusion matrix --------------------
```




```
-------------------- Precision matrix (Columm Sum=1) --------------------
```




```
-------------------- Recall matrix (Row sum=1) --------------------
```

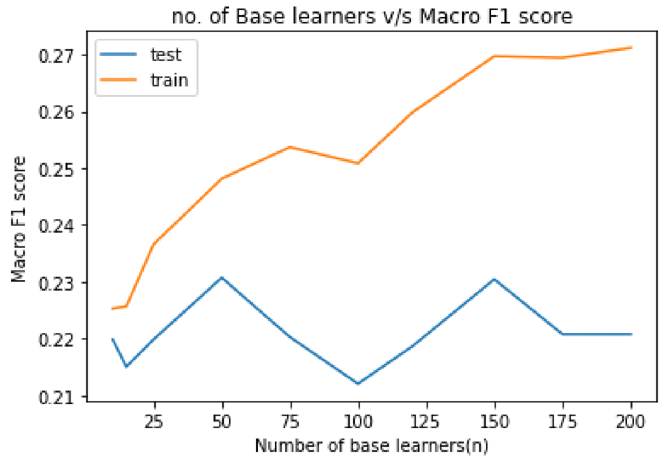


LGBM

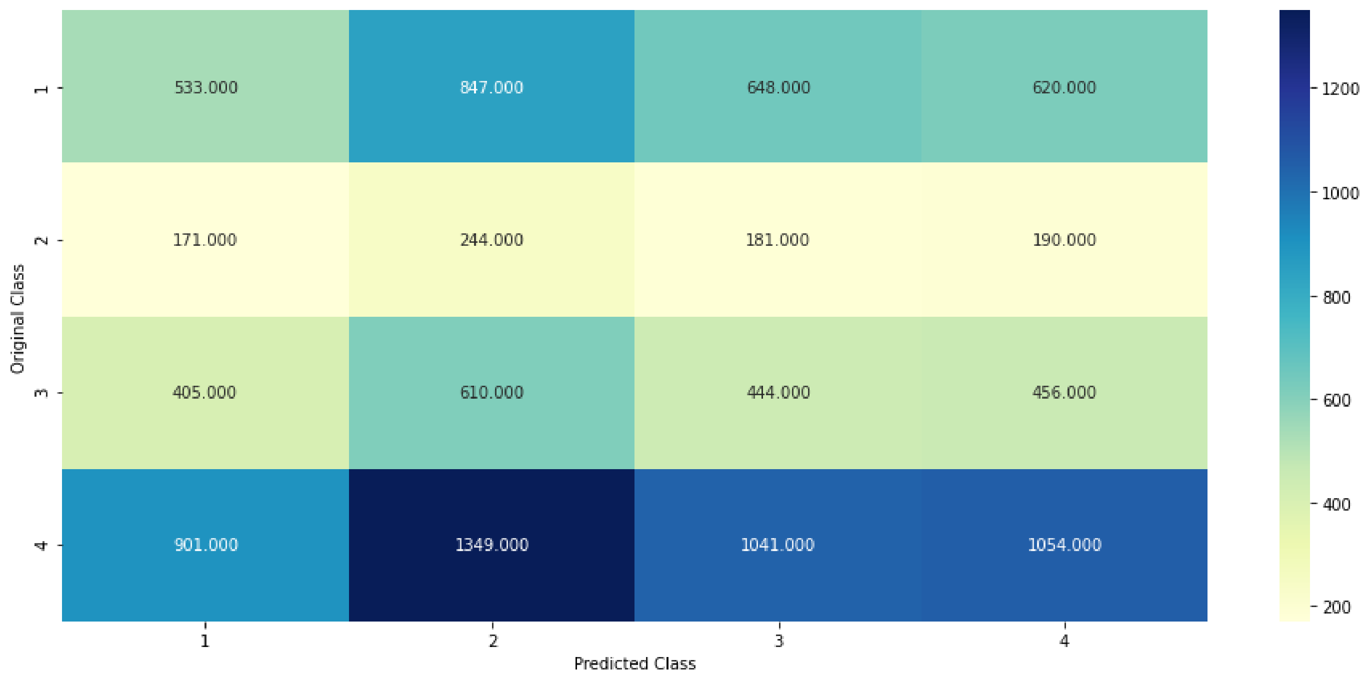
In [27]:
```
n = [10,15,25,50,75,100,120,150,175,200]

test_f1 =[]
train_f1 = []

for i in n:

    train_pred,test_pred,d2_y = custom_ensemble(train,y_train,test,i,lgb.LGBMClassifier(class_weight="bala
                                               LogisticRegression(class_weight="balanced"))

    train_score = f1_score(d2_y,train_pred,average="macro",labels=[1,2,3,4])
    test_score = f1_score(y_test,test_pred,average="macro",labels=[1,2,3,4])

    train_f1.append(train_score)
    test_f1.append(test_score)

    print("*"*60)
    print("Train Macro F1 score for n_estimator={} is : {}".format(i,train_score))
    print("Test Macro F1 score for n_estimator={} is : {}".format(i,test_score))
    print("*"*60)

plt.plot(n,test_f1,label="test")
plt.plot(n,train_f1,label="train")
plt.legend()
plt.xlabel("Number of base learners(n)")
plt.ylabel("Macro F1 score")
plt.title("no. of Base learners v/s Macro F1 score")
plt.show()

best_n = n[np.argmax(test_f1)]


train_pred,test_pred,d2_y = custom_ensemble(train,y_train,test,best_n,lgb.LGBMClassifier(class_weight="bal
                                               LogisticRegression(class_weight="balanced"))

train_score = f1_score(d2_y,train_pred,average="macro",labels=[1,2,3,4])
test_score = f1_score(y_test,test_pred,average="macro",labels=[1,2,3,4])

print("*"*60)
print("Train Macro F1 score for n_estimator={} is : {}".format(best_n,train_score))
print("Test Macro F1 score for n_estimator={} is : {}".format(best_n,test_score))
print("*"*60)

plot_confusion_matrix(y_test,test_pred)
```

```
************************************************************
Train Macro F1 score for n_estimator=10 is : 0.2253071521835535
Test Macro F1 score for n_estimator=10 is : 0.21984390320169342
************************************************************
************************************************************
Train Macro F1 score for n_estimator=15 is : 0.22567625867543098
Test Macro F1 score for n_estimator=15 is : 0.2150542529232191
************************************************************
************************************************************
Train Macro F1 score for n_estimator=25 is : 0.23658027748161664
Test Macro F1 score for n_estimator=25 is : 0.21985662409630674
************************************************************
************************************************************
Train Macro F1 score for n_estimator=50 is : 0.24813776379619443
Test Macro F1 score for n_estimator=50 is : 0.2307245002902875
************************************************************
************************************************************
Train Macro F1 score for n_estimator=75 is : 0.2536725085586656
Test Macro F1 score for n_estimator=75 is : 0.22028088612374408
************************************************************
************************************************************
Train Macro F1 score for n_estimator=100 is : 0.25082928925882575
Test Macro F1 score for n_estimator=100 is : 0.21206333620912082
************************************************************
************************************************************
Train Macro F1 score for n_estimator=120 is : 0.25981120564929827
Test Macro F1 score for n_estimator=120 is : 0.21868384391392087
************************************************************
************************************************************
Train Macro F1 score for n_estimator=150 is : 0.2696702037030754
```

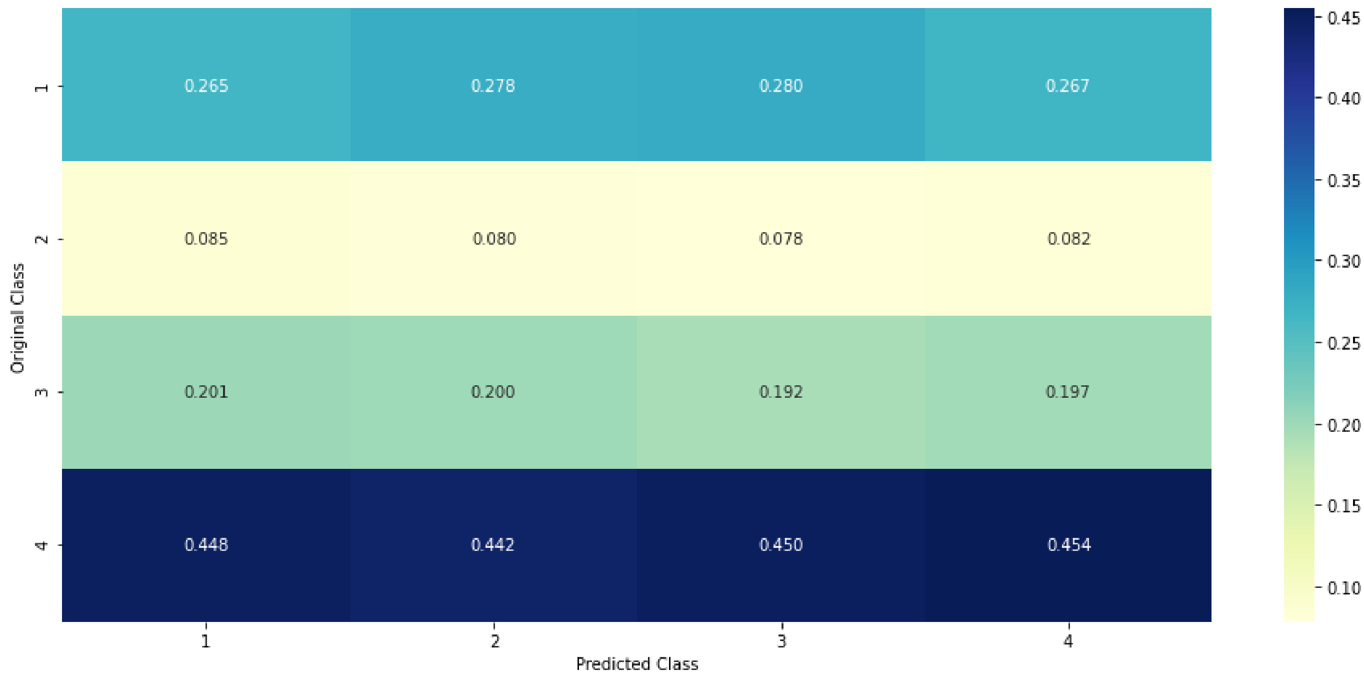
```
Test Macro F1 score for n_estimator=150 is : 0.2304380811274369
************************************************************
************************************************************
Train Macro F1 score for n_estimator=175 is : 0.2694064724336277
Test Macro F1 score for n_estimator=175 is : 0.22077861332664256
************************************************************
************************************************************
Train Macro F1 score for n_estimator=200 is : 0.2711417354247143
Test Macro F1 score for n_estimator=200 is : 0.22078618976391323
************************************************************
```



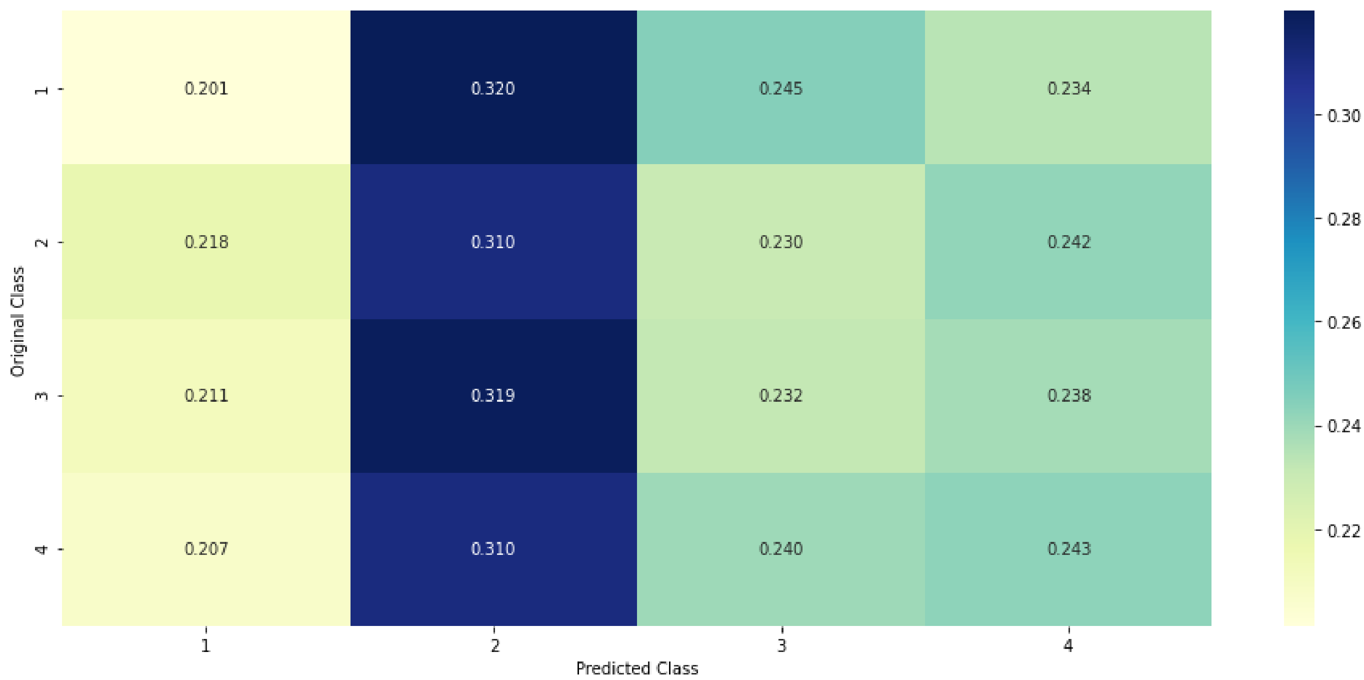

```
************************************************************
Train Macro F1 score for n_estimator=50 is : 0.24091801912897187
Test Macro F1 score for n_estimator=50 is : 0.22058180580346975
************************************************************
-------------------- Confusion matrix --------------------
```



------------------- Precision matrix (Columm Sum=1) -------------------



------------------- Recall matrix (Row sum=1) -------------------



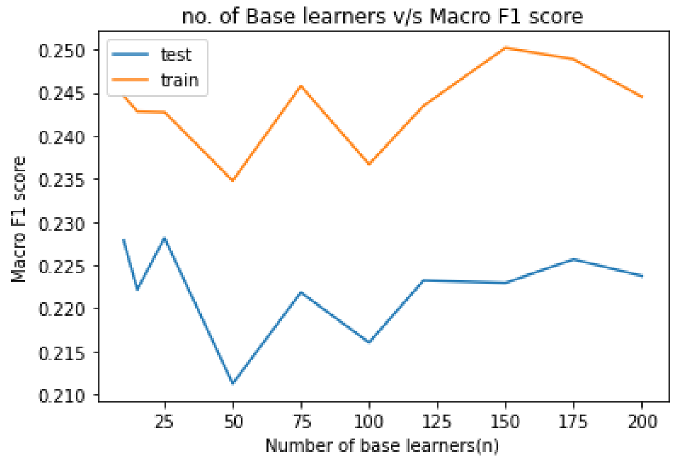
LGBM with GOSS

In [28]:

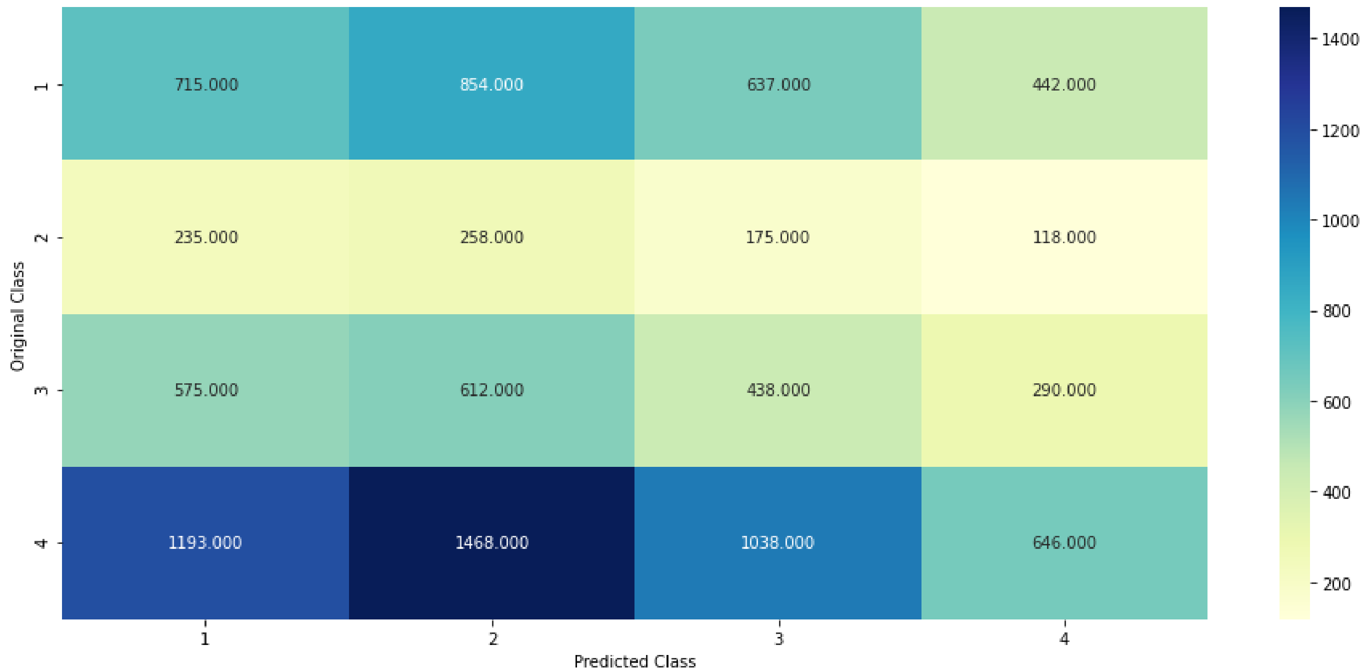
```
n = [10,15,25,50,75,100,120,150,175,200]

test_f1 =[]
train_f1 = []

for i in n:

    train_pred,test_pred,d2_y = custom_ensemble(train,y_train,test,best_n,lgb.LGBMClassifier(class_weight=
                                           LogisticRegression(class_weight="balanced"))

    train_score = f1_score(d2_y,train_pred,average="macro",labels=[1,2,3,4])
    test_score = f1_score(y_test,test_pred,average="macro",labels=[1,2,3,4])

    train_f1.append(train_score)
    test_f1.append(test_score)

    print("*"*60)
    print("Train Macro F1 score for n_estimator={} is : {}".format(i,train_score))
    print("Test Macro F1 score for n_estimator={} is : {}".format(i,test_score))
    print("*"*60)

plt.plot(n,test_f1,label="test")
plt.plot(n,train_f1,label="train")
plt.legend()
plt.xlabel("Number of base learners(n)")
plt.ylabel("Macro F1 score")
plt.title("no. of Base learners v/s Macro F1 score")
plt.show()

best_n = n[np.argmax(test_f1)]


train_pred,test_pred,d2_y = custom_ensemble(train,y_train,test,best_n,lgb.LGBMClassifier(class_weight="bal
                                       LogisticRegression(class_weight="balanced"))

train_score = f1_score(d2_y,train_pred,average="macro",labels=[1,2,3,4])
test_score = f1_score(y_test,test_pred,average="macro",labels=[1,2,3,4])

print("*"*60)
print("Train Macro F1 score for n_estimator={} is : {}".format(best_n,train_score))
print("Test Macro F1 score for n_estimator={} is : {}".format(best_n,test_score))
print("*"*60)

plot_confusion_matrix(y_test,test_pred)
```

```
************************************************************
Train Macro F1 score for n_estimator=10 is : 0.24461584646500356
Test Macro F1 score for n_estimator=10 is : 0.2278391511731786
************************************************************
************************************************************
Train Macro F1 score for n_estimator=15 is : 0.24281473288274646
Test Macro F1 score for n_estimator=15 is : 0.22217340756765924
************************************************************
************************************************************
Train Macro F1 score for n_estimator=25 is : 0.24274010125380896
Test Macro F1 score for n_estimator=25 is : 0.22814660800642522
************************************************************
************************************************************
Train Macro F1 score for n_estimator=50 is : 0.23477486049385524
Test Macro F1 score for n_estimator=50 is : 0.21124697972358808
************************************************************
************************************************************
Train Macro F1 score for n_estimator=75 is : 0.24576844762714012
Test Macro F1 score for n_estimator=75 is : 0.22184567970574198
************************************************************
************************************************************
Train Macro F1 score for n_estimator=100 is : 0.23667949537471752
Test Macro F1 score for n_estimator=100 is : 0.2160207798623763
************************************************************
************************************************************
Train Macro F1 score for n_estimator=120 is : 0.24346680688151065
Test Macro F1 score for n_estimator=120 is : 0.2232349049367109
************************************************************
************************************************************
Train Macro F1 score for n_estimator=150 is : 0.25018457137237404
```

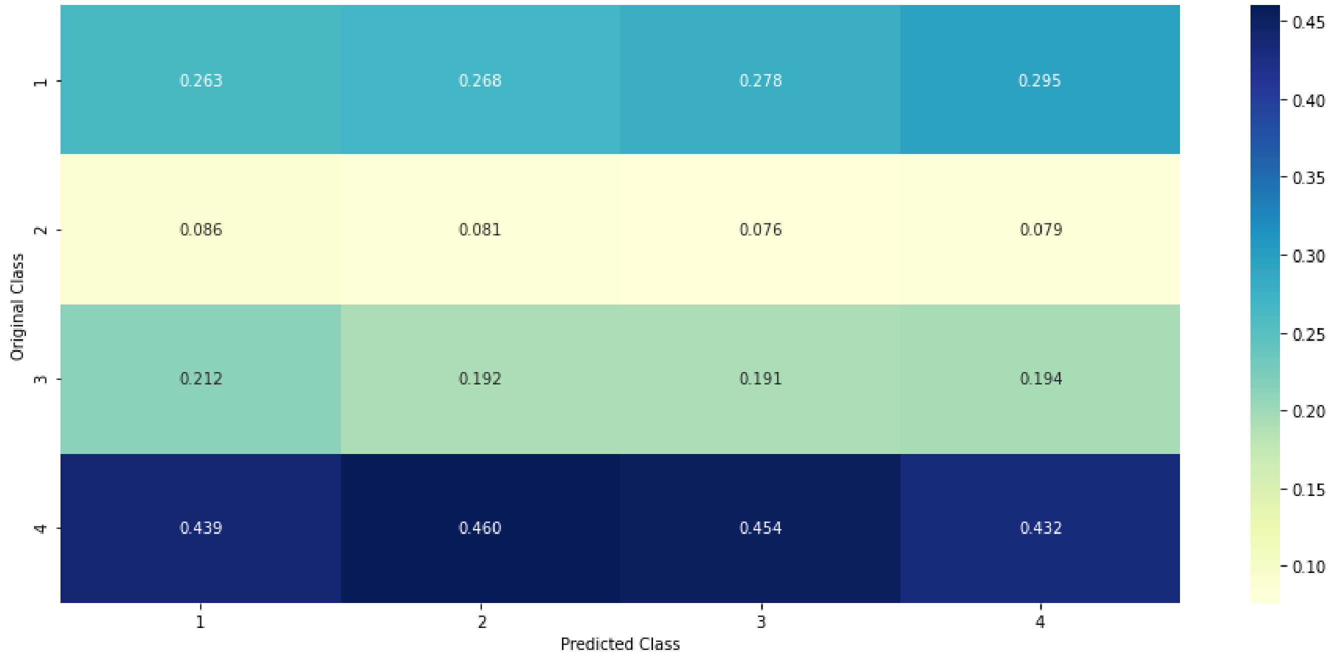
```
Test Macro F1 score for n_estimator=150 is : 0.22292897145043802
************************************************************
************************************************************
Train Macro F1 score for n_estimator=175 is : 0.24886049262484256
Test Macro F1 score for n_estimator=175 is : 0.22567515606581784
************************************************************
************************************************************
Train Macro F1 score for n_estimator=200 is : 0.244518012485107
Test Macro F1 score for n_estimator=200 is : 0.22373713642144094
************************************************************
```



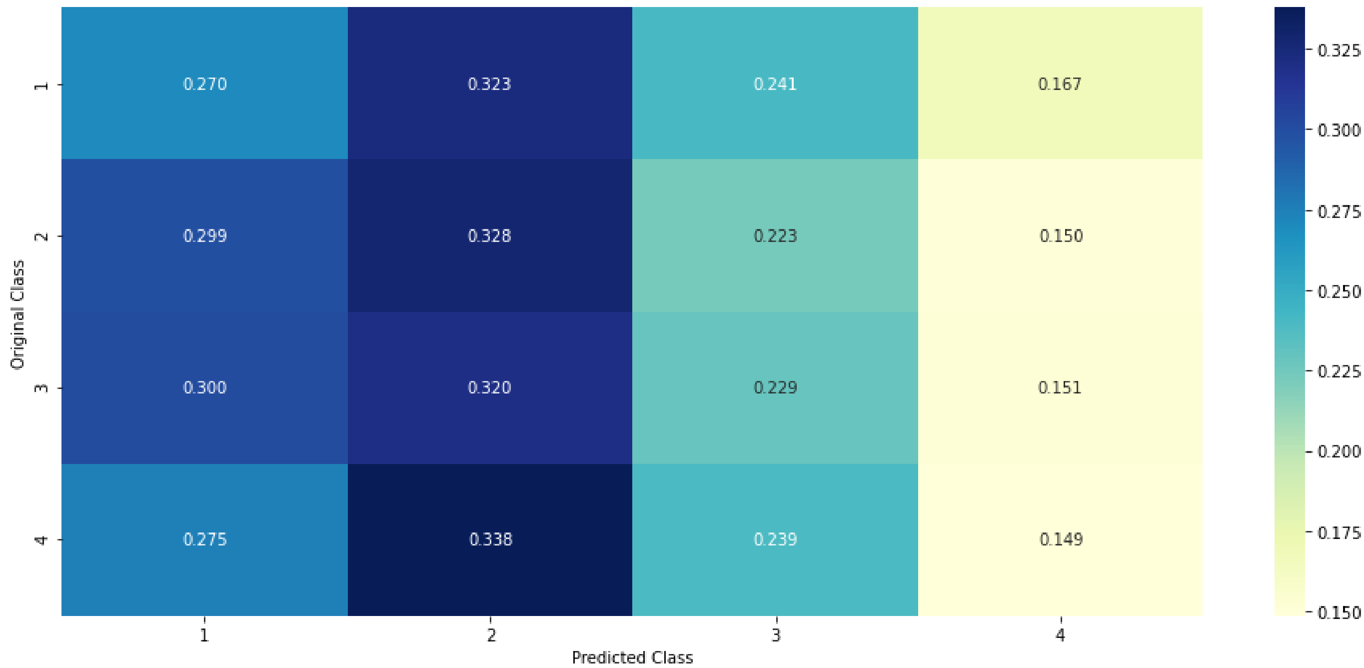

```
************************************************************
Train Macro F1 score for n_estimator=25 is : 0.23258597383182883
Test Macro F1 score for n_estimator=25 is : 0.20645592800527962
************************************************************
-------------------- Confusion matrix --------------------
```




```
-------------------- Precision matrix (Columm Sum=1) --------------------
```




```
------------------- Recall matrix (Row sum=1) --------------------
```



Here from these models we got LogisticRegression as best model with base estimator 150.

In [ ]: