random_classifier

October 7, 2022

0.0.1 Import

```
[ ]: import numpy as np
     import random
     from collections import Counter
     from sklearn.metrics import accuracy_score
     from sklearn.metrics import precision_recall_fscore_support
```

0.0.2 Generate data

```
[ ]: def generate_label_data(class_labels, weights,N=10000):
         #e.g. class_labels=[0,1]  weights=[0.2,0.8] (should sum to one)

         y=random.choices(class_labels, weights = weights, k = N)
         print("-----GENERATING DATA-----")
         print("unique entries:",Counter(y).keys())
         print("count of labels:",Counter(y).values()) # counts the elements'␣
      ↪frequency
         print("probability of labels:",np.fromiter(Counter(y).values(),␣
      ↪dtype=float)/len(y)) # counts the elements' frequency
         return y

     #TEST
     y=generate_label_data([0,1],[0.7,0.3],10000)
```

```
-----GENERATING DATA-----
unique entries: dict_keys([0, 1])
count of labels: dict_values([7015, 2985])
probability of labels: [0.7015 0.2985]
```

```
[ ]: ## RANDOM CLASSIFIER
     def random_classifier(y_data):
         ypred=[];
         max_label=np.max(y_data); #print(max_label)
         for i in range(0,len(y_data)):
             ypred.append(int(np.floor((max_label+1)*np.random.uniform(0,1))))

         print("-----RANDOM CLASSIFIER-----")
```

```
    print("count of prediction:",Counter(ypred).values()) # counts the␣
↪elements' frequency
    print("probability of prediction:",np.fromiter(Counter(ypred).values(),␣
↪dtype=float)/len(y_data)) # counts the elements' frequency
    print("accuracy",accuracy_score(y_data, ypred))
    print("percision, recall, fscore,",precision_recall_fscore_support(y_data,␣
↪ypred))

print("\nBINARY CLASS: UNIFORM LOAD")
y=generate_label_data([0,1],[0.5,0.5],10000)
random_classifier(y)

print("\nBINARY CLASS: NON UNIFORM LOAD")
y=generate_label_data([0,1],[0.1,0.9],10000)
random_classifier(y)

print("\nMULTI-CLASS:  UNIFORM LOAD")
y=generate_label_data([0,1,2,3,4],[0.2,0.2,0.2,0.2,0.2],10000)
random_classifier(y)

print("\nMULTI-CLASS:  NON-UNIFORM LOAD")
y=generate_label_data([0,1,2,3,4],[0.5,0.1,0.2,0.1,0.1],10000)
random_classifier(y)
```

```
BINARY CLASS: UNIFORM LOAD
-----GENERATING DATA-----
unique entries: dict_keys([0, 1])
count of labels: dict_values([4992, 5008])
probability of labels: [0.4992 0.5008]
-----RANDOM CLASSIFIER-----
count of prediction: dict_values([5019, 4981])
probability of prediction: [0.5019 0.4981]
accuracy 0.4969
percision, recall, fscore, (array([0.49611476, 0.49769123]), array([0.49879808,
0.49500799]), array([0.4974528 , 0.49634598]), array([4992, 5008]))

BINARY CLASS: NON UNIFORM LOAD
-----GENERATING DATA-----
unique entries: dict_keys([1, 0])
count of labels: dict_values([9013, 987])
probability of labels: [0.9013 0.0987]
-----RANDOM CLASSIFIER-----
count of prediction: dict_values([5039, 4961])
probability of prediction: [0.5039 0.4961]
accuracy 0.5062
percision, recall, fscore, (array([0.10179399, 0.9043461 ]), array([0.51165147,
0.50560302]), array([0.16980498, 0.64859095]), array([ 987, 9013]))

MULTI-CLASS:  UNIFORM LOAD
-----GENERATING DATA-----
unique entries: dict_keys([0, 1, 4, 3, 2])
count of labels: dict_values([2007, 2041, 1945, 2070, 1937])
probability of labels: [0.2007 0.2041 0.1945 0.207  0.1937]
-----RANDOM CLASSIFIER-----
count of prediction: dict_values([2115, 1958, 1954, 1980, 1993])
probability of prediction: [0.2115 0.1958 0.1954 0.198  0.1993]
accuracy 0.2008
percision, recall, fscore, (array([0.19765066, 0.20567376, 0.19393939, 0.2082288
, 0.19805527]), array([0.19282511, 0.21313082, 0.19824471, 0.20048309,
0.19897172]), array([0.19520807, 0.2093359 , 0.19606842, 0.20428255,
0.19851244]), array([2007, 2041, 1937, 2070, 1945]))

MULTI-CLASS:  NON-UNIFORM LOAD
-----GENERATING DATA-----
unique entries: dict_keys([3, 2, 0, 4, 1])
count of labels: dict_values([982, 1973, 5025, 1028, 992])
probability of labels: [0.0982 0.1973 0.5025 0.1028 0.0992]
-----RANDOM CLASSIFIER-----
count of prediction: dict_values([1976, 2022, 1969, 2134, 1899])
probability of prediction: [0.1976 0.2022 0.1969 0.2134 0.1899]
accuracy 0.2003
percision, recall, fscore, (array([0.4901088 , 0.09465792, 0.19905213,
0.10779352, 0.11122397]), array([0.19721393, 0.20362903, 0.19158642, 0.21690428,
0.21303502]), array([0.28125443, 0.12923864, 0.19524793, 0.14401623,
0.14614615]), array([5025,  992, 1973,  982, 1028]))
```

0.0.3 Extra code

[ ]:
```
# from sklearn.metrics import accuracy_score
# y_pred = [0, 2, 1, 0, 1, 3]
# y_true = [0, 1, 2, 0, 1, 3]
# print("accuracy",accuracy_score(y_true, y_pred))
# print("number correct:",accuracy_score(y_true, y_pred, normalize=False))
# # print(accuracy_score(y_true, y_pred, normalize=False)/len(y_pred))
```

[ ]:
```
# ### SANITY CHECK (UNIFORM RANDOM INTEGER GENERATOR)
# class_labels = np.array([0, 1,2,3])
# tmp=[]
# for i in range(0,10000):
#     tmp.append(int(np.floor((np.max(class_labels)+1)*np.random.uniform(0,1))))
# print("count of labels:",Counter(tmp).values()) # counts the elements'␣
  ↪frequency
```

[ ]:

```
# traversing the array
# count=0; unique=[]
# for item in Y:
#     if item not in l1:
#         count += 1
#         unique.append(item)
# print(count,l1)
```