Candace Edwards
ICS 635: Machine Learning
Homework 1
Part 1: Q1- Q3 in C.Edwards_Homework 1 Question 1_to_3.ipynb
Part 2: Q4

Implement a function for (a) accuracy and (b) 10-fold cross validation using any provided dataset. Fill in the calculate_accuracy and run_ten_fold_cross_validation function templates below.

You may not import any Python library apart from the imports which are already provided below. This includes sub-libraries (i.e., you cannot use sklearn beyond the svm module used in the helper functions).

```
###
### Code imports.
###

import numpy as np
from sklearn import svm
from sklearn.datasets import load_iris


def validate_row(X,y):
  '''
  Raise ValueError if array input length(rows) is not equal
  '''
  if X.shape[0] != y.shape[0]:
        raise ValueError("Input arrays X and y must have the same number of rows")


def train_model(X_train, y_train):
  '''
  Helper function which returns a trained model given a training
  set (X_train, y_train).
  '''
  clf = svm.SVC()
  clf.fit(X_train, y_train)
  return clf

def make_predictions(model, X_test):
  '''
  Helper function which returns a NumPy array of predictions given
  a model and a set of data points X_test.
  '''
  return model.predict(X_test)


def calculate_accuracy(y_true, y_pred):
  '''
  Question 4a
  Accuracy = (true values)/(true values + not true values)
  '''
  validate_row(y_true,y_pred)

  true_val, not_true_val = [0] *2

  for y_hat,y in zip(y_pred,y_true):

    #print((y_hat,y))
    if (y_hat == y):
      true_val += 1
      continue

    elif(y_hat != y):
      not_true_val +=1
      continue

    else:
      print('should_not_reach')


  return ((true_val)/(true_val+not_true_val))*100
```

```
def unit_test_calculate_accuracy():
  '''
  Test your solution.
  '''
  y_true = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2])
  y_pred = np.array([0, 0, 1, 1, 2, 1, 0, 1, 2])
  accuracy = calculate_accuracy(y_true, y_pred)

  print('Calculated accuracy: ', accuracy)
  print('Expected accuracy: 55.55555555555556')

unit_test_calculate_accuracy()
```

```
    Calculated accuracy:  55.55555555555556
    Expected accuracy: 55.55555555555556
```

```
def run_ten_fold_cross_validation(X, y,n=10):
  '''
  Question 4b.
  **Extra Credit**
  n = number of folds, k
  '''
  validate_row(X,y)
  accuracy = np.empty(n)
  fold = len(X)/n

  for i in range(n):

    #first itr
    if i == 0:
      #print('first_itr')
      test_start = 0
      test_end = int(fold)

      train_start= int(fold)
      train_end = len(X)

      test_data_x=X[test_start:test_end]
      train_data_x=X[train_start: train_end]

      test_data_y=y[test_start:test_end]
      train_data_y=y[train_start: train_end]

      #train model
      model = train_model(train_data_x, train_data_y)
      #test
      y_pred = make_predictions(model, test_data_x)
      #calc accuracy
      accuracy[i] = calculate_accuracy(test_data_y, y_pred)
      continue


    #last itr
    if i==(n-1):
      #print('last_itr')
      test_start = test_end
      test_end = len(X)

      train_start = 0
      train_end = test_start

      test_data_x=X[test_start:test_end]
      train_data_x=X[train_start: train_end]

      test_data_y=y[test_start:test_end]
      train_data_y=y[train_start: train_end]

      #train model
      model = train_model(train_data_x, train_data_y)

      #test
      y_pred = make_predictions(model, test_data_x)

      #calc accuracy
      accuracy[i] = calculate_accuracy(test_data_y, y_pred)
      continue
```

```
    #normal itr
    #print (f"normal_itr {i}")
    test_start = test_end
    test_end = int(fold)*(i+1)

    train_start= 0
    train_end = test_start

    train_start_2= test_end
    train_end_2 = len(X)

    test_data_x=X[test_start:test_end]
    train_data_x= np.concatenate((X[train_start: train_end],X[train_start_2: train_end_2]))

    test_data_y=y[test_start:test_end]
    train_data_y= np.concatenate((y[train_start: train_end],y[train_start_2: train_end_2]))

    #train model
    model = train_model(train_data_x, train_data_y)

    #test
    y_pred = make_predictions(model, test_data_x)

    #calc accuracy
    accuracy[i] = calculate_accuracy(test_data_y, y_pred)


  return np.mean(accuracy), np.std(accuracy)

def unit_test_cross_validation_example():
  '''
  Test your solution.
  '''

  X, y = load_iris(return_X_y=True)
  mean_accuracy, std_accuracy = run_ten_fold_cross_validation(X, y)

  #print(X.shape)
  #print(y.shape)

  print('Mean accuracy: ', mean_accuracy, ' +/- ', std_accuracy)
  print('Expected mean and standard deviation: 93.33333333333334  +/-  7.302967433402213')

unit_test_cross_validation_example()
```

```
    Mean accuracy:  93.33333333333334  +/-  7.302967433402213
    Expected mean and standard deviation: 93.33333333333334  +/-  7.302967433402213
```

SANDBOX

```
##sandbox


test_arr = np.array([0,1,2,3,4,5,6,7,8,9])


#split data

fold = len(test_arr)/10
print(fold)

#first itr
test_start = 0
test_end = int(fold) #exclusive

train_start= int(fold)
train_end = len(test_arr)

test_data=test_arr[test_start:test_end]
train_data=test_arr[train_start: train_end]
```

```
print(test_data) #expected 0
print(train_data) #expected 1,2,3,4,5,6,7,8,9
```

```
1.0
[0]
[1 2 3 4 5 6 7 8 9]
```

```
#second itr (two train variables)

test_start = test_end
test_end = int(fold)*2 #i+1

train_start= 0
train_end = test_start

train_start_2= test_end
train_end_2 = len(test_arr)

test_data=test_arr[test_start:test_end]
train_data= np.concatenate((test_arr[train_start: train_end],test_arr[train_start_2: train_end_2]))

print(test_data) #expected 1
print(train_data) #expected 0,2,3,4,5,6,7,8,9
```

```
[1]
[0 2 3 4 5 6 7 8 9]
```

```
#third itr (two train variables)

test_start = test_end
test_end = int(fold)*3 #i+1

train_start= 0 #stays same
train_end = test_start #stays same

train_start_2= test_end
train_end_2 = len(test_arr)

test_data=test_arr[test_start:test_end]
train_data= np.concatenate((test_arr[train_start: train_end],test_arr[train_start_2: train_end_2]))

print(test_data) #expected 2
print(train_data) #expected 0,1,3,4,5,6,7,8,9
```

```
[2]
[0 1 3 4 5 6 7 8 9]
```

```
#last itr (two train variables) -- DNU_fixed in loop

test_start = test_end
test_end = int(fold)*9 #i+1

train_start= 0
train_end = test_start

train_start_2= test_end
train_end_2 = len(test_arr)

test_data=test_arr[test_start:test_end]
train_data= np.concatenate((test_arr[train_start: train_end],test_arr[train_start_2: train_end_2]))

print(test_data) #expected 9
print(train_data) #expected 0,1,2,3,4,5,6,7,8
```

```
[3 4 5 6 7 8]
[0 1 2 9]
```

```
test_arr = np.array([0,1,2,3,4,5,6,7,8,9])
n = 10 #k

fold = len(test_arr)/n
```

```
for i in range(n):

  #first itr
  if i == 0:
    print('first_itr')
    test_start = 0
    test_end = int(fold) #exclusive

    train_start= int(fold)
    train_end = len(test_arr)

    test_data=test_arr[test_start:test_end]
    train_data=test_arr[train_start: train_end]

    print(test_data) #expected 0
    print(train_data) #expected 1,2,3,4,5,6,7,8,9
    continue


  #last itr
  if i==(n-1):
    print('last_itr')
    test_start = test_end
    test_end = len(test_arr)

    train_start = 0
    train_end = test_start

    test_data=test_arr[test_start:test_end] #expected 9
    train_data=test_arr[train_start: train_end] #expected 0,1,2,3,4,5,6,7,8

    print(test_data)
    print(train_data)
    continue


  #normal itr
  print (f"normal_itr {i}")
  test_start = test_end
  test_end = int(fold)*(i+1)

  train_start= 0
  train_end = test_start

  train_start_2= test_end
  train_end_2 = len(test_arr)

  test_data=test_arr[test_start:test_end]
  train_data= np.concatenate((test_arr[train_start: train_end],test_arr[train_start_2: train_end_2]))

  print(test_data)
  print(train_data)
```

```
first_itr
[0]
[1 2 3 4 5 6 7 8 9]
normal_itr 1
[1]
[0 2 3 4 5 6 7 8 9]
normal_itr 2
[2]
[0 1 3 4 5 6 7 8 9]
normal_itr 3
[3]
[0 1 2 4 5 6 7 8 9]
normal_itr 4
[4]
[0 1 2 3 5 6 7 8 9]
normal_itr 5
[5]
[0 1 2 3 4 6 7 8 9]
normal_itr 6
[6]
[0 1 2 3 4 5 7 8 9]
normal_itr 7
[7]
```

```
[0 1 2 3 4 5 6 8 9]
normal_itr 8
[8]
[0 1 2 3 4 5 6 7 9]
last_itr
[9]
[0 1 2 3 4 5 6 7 8]
```

```
test_x =  np.random.rand(10,4)
test_y=   np.random.rand(10)
```

```
test_x
```

```
array([[0.56394974, 0.4486378 , 0.96050791, 0.37737529],
       [0.60853602, 0.99754772, 0.71637545, 0.65864016],
       [0.30326468, 0.41438253, 0.74578926, 0.6053464 ],
       [0.41291428, 0.38097108, 0.27554814, 0.33320661],
       [0.16100696, 0.61029589, 0.66300431, 0.15373862],
       [0.76763114, 0.70987423, 0.27570109, 0.38466996],
       [0.70388512, 0.21540596, 0.17672332, 0.03015071],
       [0.68575693, 0.02159981, 0.9029085 , 0.04814467],
       [0.86012752, 0.84736786, 0.92304548, 0.19965203],
       [0.78967181, 0.4360335 , 0.87410469, 0.56536933]])
```

```
#test with two input arrays
n = 10 #k
fold = len(test_arr)/n
```

```
for i in range(n):

  #first itr
  if i == 0:
    print('first_itr')
    test_start = 0
    test_end = int(fold) #exclusive

    train_start= int(fold)
    train_end = len(test_arr)

    test_data_x=test_x[test_start:test_end]
    train_data_x=test_x[train_start: train_end]

    test_data_y=test_y[test_start:test_end]
    train_data_y=test_y[train_start: train_end]

    print(test_data_x)
    print(train_data_x)

    print(test_data_y)
    print(train_data_y)
    continue


  #last itr
  if i==(n-1):
    print('last_itr')
    test_start = test_end
    test_end = len(test_arr)

    train_start = 0
    train_end = test_start

    test_data_x=test_x[test_start:test_end]
    train_data_x=test_x[train_start: train_end]

    test_data_y=test_y[test_start:test_end]
    train_data_y=test_y[train_start: train_end]

    print(test_data_x)
    print(train_data_x)

    print(test_data_y)
```

```
        print(train_data_y)
        continue


    #normal itr
    print (f"normal_itr {i}")
    test_start = test_end
    test_end = int(fold)*(i+1)

    train_start= 0
    train_end = test_start

    train_start_2= test_end
    train_end_2 = len(test_arr)

    test_data_x=test_x[test_start:test_end]
    train_data_x= np.concatenate((test_x[train_start: train_end],test_x[train_start_2: train_end_2]))

    test_data_y=test_y[test_start:test_end]
    train_data_y= np.concatenate((test_y[train_start: train_end],test_y[train_start_2: train_end_2]))

    print(test_data_x)
    print(train_data_x)

    print(test_data_y)
    print(train_data_y)
```

```
 [0.70388512 0.21540596 0.17672332 0.03015071]
 [0.68575693 0.02159981 0.9029085  0.04814467]
 [0.86012752 0.84736786 0.92304548 0.19965203]]
[0.63075495]
[0.6540536  0.54415815 0.64101391 0.98459195 0.25863934 0.16821862
 0.58904569 0.16040741 0.43357099]
```


✓ 0s completed at 9:51 PM