**Single Programmer Affidavit**

I the undersigned promise that the attached assignment is my own work. While I was free to discuss ideas with others, the work contained is my own. I recognize that should this not be the case, I will be subject to penalties as outlined in the course syllabus. 

Programmer (print & sign your name, then date it)

*'''*

*file: std\_cv.py*

*author: Alexander Giang*

*'''*

from statistics import (stdev, mean)

from ml\_lib.learning import (err\_ratio, train\_and\_test)

def **cross\_validation**(learner, dataset, k=10):

*"""Perform k-fold cross\_validation*

*Run k trials where each trial has a different (k-1)/k percentage*

*of the data as training data and 1/k as test data.*

*Returns tuple (mean\_err, std\_err, fold\_errors, models)*

*"""*

if k is None:

k = len(dataset.examples)

fold\_errT = 0 # fold error on training data

fold\_errV = 0 # fold error on validation data

fold\_err\_list = [] # holds all the fold error rates

train\_list = [] # holds all the trained data

n = len(dataset.examples)

examples = dataset.examples

for fold in range(k): # for each fold

# Split into train and test

# Note that this is not a canonical cross validation where

# every pieces of data is used for training and testing

# due to the shuffling above.

train\_data, val\_data = train\_and\_test(dataset, fold \* (n / k),

(fold + 1) \* (n / k))

train\_list.append(train\_data)

dataset.examples = train\_data

h = learner(dataset)

# predict and accumulate the error rate on

# the training and validation data

fold\_ratio\_errT = err\_ratio(h,dataset,train\_data)

fold\_errT += fold\_ratio\_errT # accumulating

fold\_ratio\_errV = err\_ratio(h,dataset,val\_data)

fold\_errV += err\_ratio(h, dataset, val\_data)

fold\_err\_list.append(fold\_ratio\_errV)

# Reverting back to original once test is completed

dataset.examples = examples

# Return average per fold rates

mean\_folds = mean(fold\_err\_list)

stdev\_folds = stdev(fold\_err\_list,mean\_folds)

# Return values are the mean error rate, the standard deviation of

# the error rate, a list containing the error rate for each fold,

# and a list of models trained.

return (mean\_folds,stdev\_folds,fold\_err\_list, dataset.name)

*'''*

*Created on Apr 15, 2018*

*file: driver.py*

**@author:** *mroch, Alexander Giang*

*'''*

from ml\_lib.learning import (DataSet,

DecisionTreeLearner, NeuralNetLearner)

from std\_cv import cross\_validation

from random import shuffle

from copy import deepcopy

def **learn**(dataset):

*'''Given a dataset, create a shuffled copy of the data. Run a 10 fold*

*cross validation (std\_cv.cross\_validation) using decision trees and neural*

*nets (DecisionTreeLearner and NeuralNetLearner). The data are only*

*shuffled once, so that std\_cv.cross\_validation will use the same exact*

*training and test data on each of the folds for both learners.'''*

shuffle(dataset.examples)

cv\_tree = cross\_validation(DecisionTreeLearner,dataset)

dataset.attributes\_to\_numbers()

cv\_net = cross\_validation(NeuralNetLearner,dataset)

# formatting the output for Learner column

if dataset.name is *'restaurant'*:

formatlearner = (*'\tDecisionTreeLearner'*,*'\tNeuralNetLearner'*)

else:

formatlearner = (*'\t\tDecisionTreeLearner'*,*'\t\tNeuralNetLearner'*)

# formatting the rest of output

output1 = *" "*.join([format(x,*'.3f'*) for x in cv\_tree[2]])

print(format(cv\_tree[0],*'.3f'*) + *'\t'* + format(cv\_tree[1],*'.3f'*) + *' '*

+ output1 + *'\t'* + cv\_tree[3] + formatlearner[0])

output2 = *" "*.join([format(x,*'.3f'*) for x in cv\_net[2]])

print(format(cv\_net[0],*'.3f'*) + *'\t'* + format(cv\_net[1],*'.3f'*) + *' '*

+ output2 + *'\t'* + cv\_net[3] + formatlearner[1])

def **main**():

print(*"Calculating datasets..."*)

print(*"Mean\tStdDev Errors for each fold \t\t\t\t\t\tCorpus\t\tLearner"*)

for dataset in [*'zoo'*,*'iris'*,*'orings'*,*'restaurant'*]:

data = DataSet(name=dataset)

learn(data)

if \_\_name\_\_ == *'\_\_main\_\_'*:

main()

**Output**: 