1. *Outline of assignment*

For this assignment, we were required to build on the work done previously, when we researched machine learning packages, and chose two classification algorithms that we felt would appropriately deal with a given classification task. My choice of machine learning package was *scikit-learn*, and the classification algorithms I selected were the ***k-Nearest Neighbours*** and ***Support Vector Machines*** algorithms.

*scikit-learn* has built-in methods for constructing ROC curves when provided with a given set of probability scores for the ‘positive’ class, which I discuss in detail in section 2. Hence, I was able to obtain a ROC curve for both algorithms, which provided a new means of comparing them. These are included in section 3. I made three observations about the two curves, outlined in section 4.

1. *Methodology and assumptions*

*scikit-learn*’s ability to deal with ROC curves comes in the form of the [*roc\_curve*](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html)function. The parameters that were important for this dataset were *y\_true, y\_score,* and *pos\_label.*

* y\_true: the array of true binary labels. This array was simply a subset of the *autoimmune\_target* array, which was obtained with the help of a *StratifiedKFold* (more below).
* y\_score: the array of probability scores for the ‘positive’ label. This was obtained with the help of the *predict\_proba* method (more below).
* pos\_label: since the class labels in our dataset were not simple labels like {-1,1} or {0,1}, this had to specified so that *scikit-learn* knew what label corresponded to the positive class. In our case, this label was “positive”. The function would then automatically assume that any labels other than “positive” would correspond to the negative class, since the function can only work with binary classification tasks.

The function returns the false positive and true positive rates as separate arrays, as well as a threshold array. As explained in the documentation, the value at *thresholds[i]* is used to calculate false positive and true positive rates, which are stored at *fpr[i]* and *tpr[i]* respectively. Representing the ROC curves visually was then a simple matter of plotting the values of *tpr* on the y-axis, and *fpr* on the x-axis.

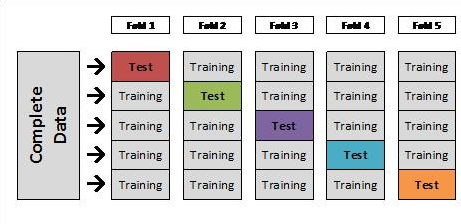
As required, I plotted my curves using a held-out test set which was 1/3 of the overall dataset size. I used a [*StratifiedKFold*](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html)to achieve this. Specifically, I used the [*split*](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#sklearn.model_selection.StratifiedKFold.split) method, the purpose of which is to “generate indices to split data into training and test set”. This works by splitting the data set into n folds and using 1/n of the dataset as a test set, while using the rest as a training set. Each sample gets used once for testing, and (n-1) times for training [See Figure 1].

Figure 1: Visual representation of k fold generation

So, by setting *n\_splits* to 3, I generated 3 folds, each with a test set that was 1/3 of the overall dataset size. In fact, since I had 3 folds, I was able to construct 3 ROC curves, which offered a better idea of how the algorithm performed on average. Also, for each curve, I calculated its AUROC score, again with the help of one of the package functions; the [*auc*](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html) function. Each curve’s AUROC score is reported in the legends of their graphs, included in section 3.

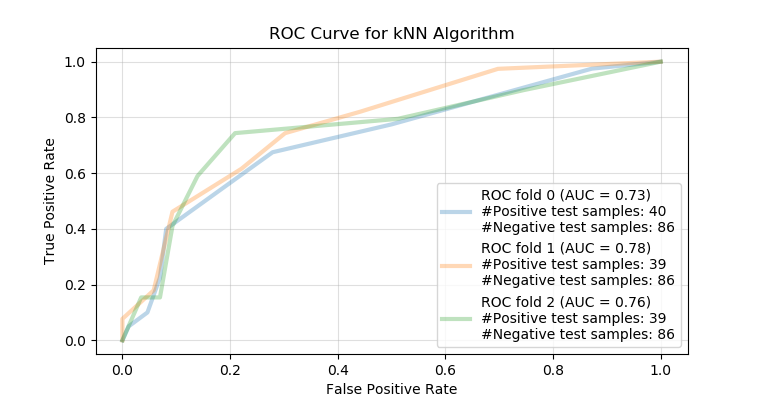
1. *Methodology and assumptions (continued)*

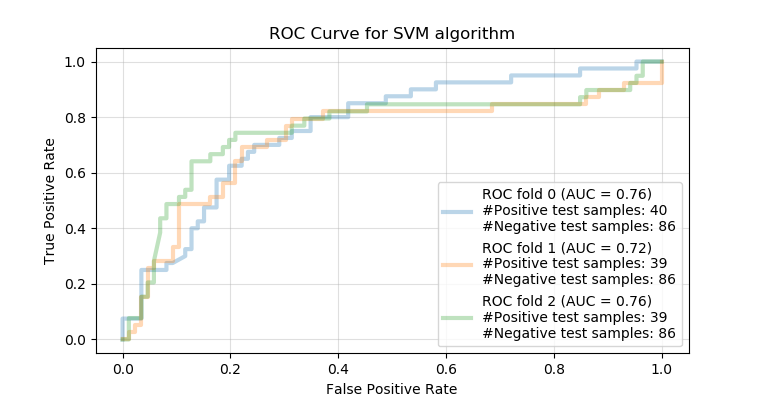
The value of using a StratifiedKFold is the fact that for each fold, *the test set has approximately the same amount of positively labelled samples as every other test set.* For the *autoimmune* dataset, each fold’s test set had either 39 or 40 positively labelled samples, while having 86 negatively labelled samples. This fact is reported in the legends of the graphs.

This is useful because it ensures that each fold is as good a representative of the complete dataset as every other fold, which in theory should yield ROC curves that better reflect the future performance of the algorithm. Of course, this assumes that our current data is representative of any future data that needs to be classified; if it is not, the ROC curve is a somewhat redundant measure of future performance.

[More on the mean curve, mean AUROC]

1. *ROC Curves*

**



1. *Observations*

**Observation 1:** sdhgiuwdghsdg

**Observation 2:** hguiweghiuwegosd

**Observation 3:** hiodfhisdfdsf

*Bibliography*

[1] - [Scikit-learn: Machine Learning in Python](http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html), Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[2] - *[Nearest Neighors](http://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification)*, scikit-learn documentation (v0.20.0)

[3] - [*Support Vector Machines*](http://scikit-learn.org/stable/modules/svm.html#classification), scikit-learn documentation (v0.20.0)

[4] - [documentation](http://scikit-learn.org/stable/documentation.html), scikit-learn documentation (v0.20.0)

[5] - [*genfromtxt*](https://docs.scipy.org/doc/numpy/reference/generated/numpy.genfromtxt.html), NumPy documentation (v1.15)

[6] - [*kNeighborsClassifier*](http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier), scikit-learn documentation (v0.20.0)

[7] - [*SVC*](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC), scikit-learn documentation (v0.20.0)

[8] - [*k-nearest neighbors algorithm*](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm),Wikipedia

[9] - [*Support vector machine*](https://en.wikipedia.org/wiki/Support_vector_machine),Wikipedia

[10] - [*cross\_val\_score*](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html), scikit-learn documentation (v0.20.0)

[11] - [*What is better, k-nearest neighbors algorithm or Support Vector Machine classifier?*](https://www.quora.com/What-is-better-k-nearest-neighbors-algorithm-k-NN-or-Support-Vector-Machine-SVM-classifier-Which-algorithm-is-mostly-used-practically-Which-algorithm-guarantees-reliable-detection-in-unpredictable-situations), quora.org

*Code*

import numpy as np

import os

import matplotlib.pyplot as plt

from sklearn import neighbors, datasets, svm, ensemble

from sklearn.model\_selection import cross\_val\_score

#change into location of dataset, specify file name

os.chdir('D:\\OneDrive - National University of Ireland, Galway\\NUIG\\2018-2019\\Semester 1\\CT475\\Assignment 1')

fname = 'autoimmune\_transpose.txt'

#use np.genfromtxt to read in training data, and target feature data

autoimmune\_data = np.genfromtxt(fname, delimiter='\t', encoding=None, usecols=np.arange(0,9))

autoimmune\_target = np.genfromtxt(fname, delimiter='\t', dtype=None, encoding=None, usecols=9)

#create models with kNN, checking the the 10-fold cross validation scores for various values of k

k\_scores=[]

for i in range(1,20):

kNN = neighbors.KNeighborsClassifier(i)

kNN\_scores = cross\_val\_score(kNN, autoimmune\_data, autoimmune\_target, cv=10)

k\_scores.append(kNN\_scores.mean())

k\_max = max(k\_scores)

print("The highest 10-fold cross validation scores were found for k = ", k\_scores.index(k\_max)+1, ", which had a mean score of %.4f." % k\_max, sep='')

#create models with SVM, checking the the 10-fold cross validation scores when gamma is set to 'scale' or 'auto'

s\_scores=[]

s\_options=['scale','auto']

for i in s\_options:

svc = svm.SVC(gamma=i)

svc\_scores = cross\_val\_score(svc, autoimmune\_data, autoimmune\_target, cv=10)

s\_scores.append(svc\_scores.mean())

s\_max = max(s\_scores)

print("The highest 10-fold cross validation scores were found when gamma was set to '", s\_options[s\_scores.index(s\_max)], "', which had a mean score of %.4f." % s\_max, sep='')

if k\_max >= s\_max:

print("A higher mean cross validation score was obtained by the kNN algorithm.")

else:

print("A higher mean cross validation score was obtained by the SVM algorithm.")