**Problem Statement:**

Given the historic student records, we would like you to create a machine learning model which is able to predict the majors of new students.

The 2 pipe-seperated files with student data: **training.psv** and **eval.psv**.

**training.psv** has the historical data of about 10,000 students who have declared their majors while **eval.psv** has the data of about 2,000 students who have yet to declare their majors.

**Solution:**

**Step 1: Converting the .psv input files into .csv files**

I first converted the .psv files into .csv files for ease of use since it is easier to use .csv files in R and Python.

I used the following code in R to convert the .psv files into .csv files

install.packages("dplyr")

library(dplyr)

training <- read.csv("training.psv", sep = "|", header = TRUE, stringsAsFactors = FALSE)

write.csv(training, file = "training.csv")

eval <- read.csv("eval.psv", sep = "|", header = TRUE, stringsAsFactors = FALSE)

write.csv(eval, file = "eval.csv")

So now our two files have become **training.csv** and **eval.csv**

**Step 2: Data Pre-Processing**

The data in **training.csv** is of the form:

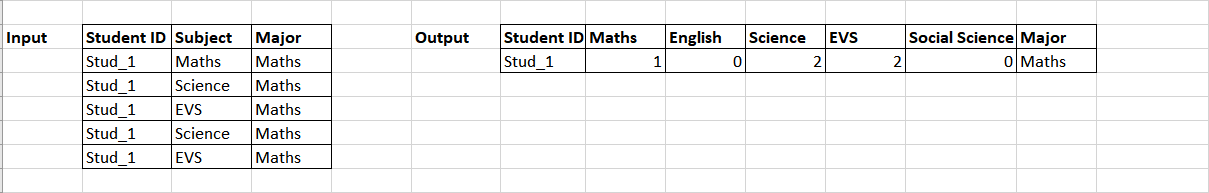
student\_id,level,course,grade,major

**Assumptions:**

1. My assumption is that a student’s major is primarily decided by the most number of courses he has taken in the relevant department. We also get rid of the course numbers and retain only the relevant departments. i.e ECE:489 becomes just ECE
2. The other assumption is that level and grade does not play a significantly deciding role in choosing the major.

Kindly request you to please humor me with these two assumptions for now.

1. Based on the above assumptions, I dropped the columns “levels” and “grade” in both the **training.csv** and **eval.csv** files. I did this in Excel since it was easy to manipulate this in Excel.
2. Since the number of entries for every student varies in number, I considered each row entry for a student as a separate feature and converted the rows into columns (described in the picture below with a toy example)



The reason behind doing this transformation is twofold,

1. A machine learning algorithm usually considers each instance (i.e. each row) as an individual entity. This transformation makes sure that we run the algorithm for 10,000 students in training.csv file and 2,000 students in training.csv file.
2. It is easier to deal with the numbers and make predictions than use Strings.

The Python code for this transformation can be found in the mail as a .py file under the file name **dataPreprocessing.py**

Let’s call the new training data after data preprocessing as training\_new.csv. This training\_new.csv file has 184 columns. First column is the Student ID, the next 182 columns represent the 182 unique departments and the last column is the Major. This training\_new.csv has 10,000 rows, one each for every student with their corresponding columns containing the number of courses he/she took in that department (like how it is described in the toy example above). We do the similar transformation to the eval.csv dataset as well and name it eval\_new.csv.

**Step 3: Classification algorithms:**

I consider this as a classification problem where in we have to classify new student data into one of the 81 Majors based on the number of courses they have taken in each relevant department. The classification algorithm predicts the Major for every student based on the courses he has taken. I used the following classification algorithms for this task,

1. SVM
2. Random Forest
3. K-Nearest Neighbors (KNN)
4. Decision Tree

I split the training.csv file into training and test set. I trained my algorithm in the training set and predicted on the test set to evaluate the accuracy of prediction for each of the algorithms. I took the liberty of performing the above-mentioned algorithms in both Python and R.

Please find the Python codes used for these classification algorithms in the following files,

1. SVM – svm.py
2. Random Forest – randomforest.py
3. K-Nearest Neighbors (KNN) – knn.py
4. Decision Tree – decisiontree.py

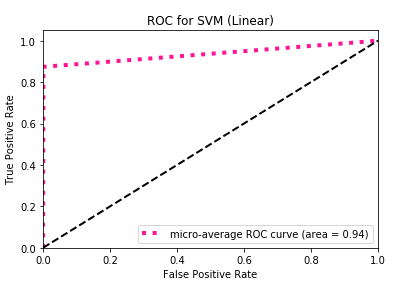
Please find the R codes used for these classification algorithms in the following files,

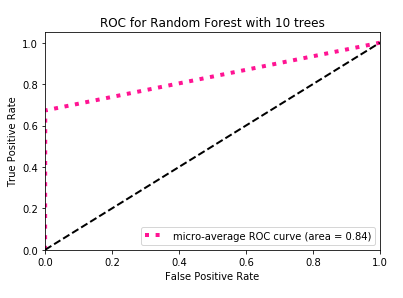
1. SVM – svm.R
2. Random Forest – randomforest.R
3. Decision Tree – decisiontree.R

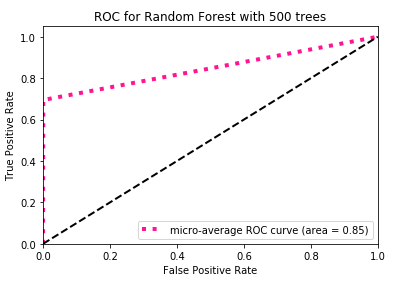
**Step 4:** **Evaluating the model performances:**

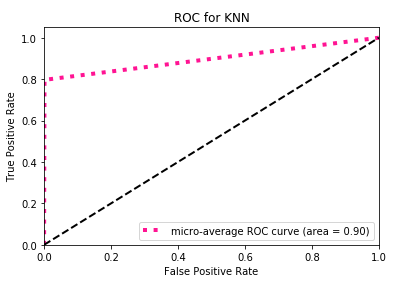
I used the ROC metric to evaluate the model accuracy. Since ours is a multi-class classification problem with each class label not distributed equally, I used micro average ROC curve instead macro average ROC curve. Using macro average in this case could be misleading as the classes with more entries might tend to dominate and screw our results as compared to the classes with the lesser entries.

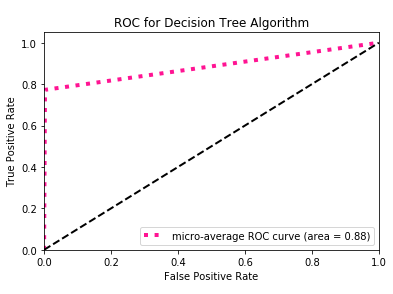
Please find the micro average ROC curves of each of the following classification algorithms,











Based on the micro-average ROC curves and the area under the curves (AUC), we see that SVM has the maximum area under the curve with 0.94. Hence, I went with SVM as my final model and predicted the eval.csv file to find the respective Majors for the students.

Please find in the mail, the **eval.csv** file with one predicted major for every student.

**End notes:**

All the models implemented in this approach utilize only the “department” information i.e. the string part of the “course” column. (E.g. ECE:489 becomes just ECE) to perform the prediction for the Majors of the new students. This is just one among other ways to approach this problem specification. Although, in hindsight this might seem like a much simpler approach, I believe that an algorithm must be simple in order to be flexible with the new data and not over-fit the training data too much. I just got another perspective to approach this problem that can be looked at by combining the “department” information + number of courses taken in that particular department + grades obtained in each course (by assigning a numerical to each grade). By assigning a numerical value to the department that combines all these attributes and not just the count of the number of courses taken in each department (as in my current implementation), we can build a better classifier. I just got this perspective while writing this, so I don’t have the time to implement it now. However, I feel that it should be worth a shot to try and should increase the accuracy of our model. The problem specification asks us to suggest 3 Majors for a particular student, but the algorithms I’ve implemented predicts one Predicted Major for every student based on the training data.

I had much fun in trying to solve this problem using ML algorithms and learned a lot along the way. Thank you very much for giving this opportunity. Cheers!