Student Intervention

This is a classification problem because the output we are aiming for is a discreet output Classification usually is about making prediction on unseen examples and deciding which category new instants belongs_, if the required output is continues output (number), then we'll identify the problem as regression problem,

Exploring the Data

```
Total number of students: 395
Number of students who passed: 265
Number of students who failed: 130
Number of features: 30
Graduation rate of the class: 67.09%
```

As we identified that it's a classification problem, I've selected 3 classification models SVM, decision tree & KNN,

SVM discriminates between two classes by generating a hyperplane that optimally separates classes, it has demonstrated a high performance in solving classification problems in many biomedical field, *why SVM?* Because the main goal of SVM is to separate training example into of two categories (student pass/not pass for this example), separated by decision surface that attempt to maximize the distance between the two categories

Decision Tree Classifier: are commonly used in operations research, where experts seeks reasons behind decision which Decision Tree can provide since it's break down the data into tree like model along with their possible consequence, *why decision tree?* Because it can handle both numerical and categorical data, and our dataset has the both data types, plus it decision tree is easier to interpret.

KNN: is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure. It main approach is remembering, and making analogy, *why KNN?* Because it perform classification by computing the majority votes of nearest neighbors for each points, since our dataset is small, KNN would be an efficient way to classify the data

Below is a table of advantages and disadvantages for each classification algorithm selected

	Advantages	Disadvantages
SVC	the use of kernels, and the absence of local minima	inefficient to train, not recommend for very large scale applications
Decision Tree Classifier	Easy to interpret, complexity is the down side & the tree might get too large even after some pruning	May over fit data
KNN	remember old data, it fast, and simple	no generalization, sensitive to noise which means Overfitting, Take up a lot of memory to run

After multiple iteration of fitting the data under the three selected classifiers and using different training sizes, we came up with below results

	SVC			**DecisionTreeClassifier**			**KNeighborsClassifier**		
Training Size	300	200	100	300	200	100	300	**200**	100
Testing Size	95	95	95	95	95	95	95	**95**	95
Training time (secs)	0.015	0	0	0	0	0	0	**0**	0
Prediction time (secs)	0	0	0.015	0	0	0	0	**0.003**	0
F1 score for training set	0.877	0.859	0.847	1	1	1	0.86859	**0.8581**	0.8243
F1 score for test set (un-tuned)	0.789	0.789	0.777	0.6608	0.61904	0.73529	0.76056	**0.8**	0.7894
F1 score for test set (Tuned)							0.826	**0.84507**	0.8235

From the classifier comparison table we can notice that running time is insignificant, so we’ll not take it into consideration in the evaluation, the main factor for evaluation will be the F1 score, again from table we can see that KNN Classifier gives us the best F1 score for both training and testing sets, if no parameters been tuned

How it works? During learning, KNN Classifier simply stores all available points in a database, in the predicting stage, it classifies any new points by calculating the K mean value for number of neighbors (Y), it takes into consideration the similarity and Distance (e.g., distance functions).

In this example KNN classifier gives us the best F1 score (0.8) among the three classifiers, now using the gridsearch we’ll find out the ideal number of neighbors, which gives us the best F1 score, which turn out to be Y = 11, see the table above

We can notice that the tuned classifier improve the perdition from 0.8 to ~0.85,which turn out to be around %6.25 improvement in model prediction