Report for student intervention project

1. This project is classification machine learning problem. The difference between classification and regression problems is that regression model predicts continuous output, while classification model returns an object from some discrete set. In this project we should predict whether student might need early intervention or not, so this is classification ML problem.

2.

Total number of students:	395
Number of students who passed:	265
Number of students who failed:	130
Graduation rate of the class:	0.67%
Number of features:	31

3. For this project I have chosen three machine learning models: Support Vector Machine, Decision Tree Classifier and kNN.

Machine learning algorithms can be divided in two major categories: *instance-based learning or non-generalizing and generalizing* algorithms.

Instance-based learning algorithms construct hypotheses directly from the training set comparing object that needs to be predicted to other objects in the training set. That means that hypotheses complexity can grow with the data. Among such algorithms we can name kNN(that stores all the training data) and SVM(stores only few “important” points – so called support vectors).

I. k-nearest neighbors algorithm is a simple machine learning algorithm that needs no training(“lazy training”) of the model. It works by finding k nearest points to the given object X and after that using those points makes a prediction on object X. Depending on the situation we can assign different weights(e.q importance) to the given points.

The most popular distance metric is Euclidean distance though there are other several other metrics: Minkowski or Hamming distance for text data.
In the most primitive realization kNN model needs no training, takes O(n) space and prediction complexity is equal to O(n).

Algorithm	Training	Space	Prediction
kNN	No training	O(n)	O(n)
KNN using k-d tree	O(NlogN)	O(KN)	O(MlogN)

One possible choice is to use kNN with k-d tree that can increase prediction time.

	Training set size		
	100	200	300
Training time(secs)	0.001	0.000	0.001
Prediction time(secs)	0.001	0.003	0.006
F1 score for training set	0.834	0.847	0.8638
F1 score for test set	0.8129	0.8137	0.8082

As we can see, training time in all cases is equal, though prediction time grows linearly when training set increases. One of the key plusses of kNN is zero cost of the learning process, robust to noisy training data.

Drawbacks:

- computation cost is high for prediction
- needs to store all the training data in the memory

II. SVM

SVM is machine learning algorithm that constructs a hyper-plane in a high dimensional space and can be used for classification, regression or outlier detection.

	Training set size		
	100	200	300
Training time(secs)	0.002	0.003	0.006
Prediction time(secs)	0.001	0.002	0.004
F1 score for training set	0.886	0.881	0.867
F1 score for test set	0.827	0.8435	0.8496

- SVM is effective in high dimensional spaces.
- Uses only a subset of training data to make predictions, so it's memory efficient.
- Can use Kernel functions to use “domain-specific knowledge” of the data to make better predictions.
  At the same time SVM has the following complexity for training: between O(featuresN^2) and O(featuresN^3). That makes SVM almost unscalable on the data having more than 10.000 features.

Given the fact that dataset contains no more than 500 points SVM is a good choice, is memory-efficient and provides good F1-score.

3. Decision Tree Classifier

- The cost of using the tree is logarithmic. O(logN) for prediction

- Simple to understand and to interpret
- Uses a white box model. Can be easily explained by using boolean logic
  Drawbacks:
  - Can create over-complex models. Tend to overfit
  - Can be unstable
  - Can be hard to learn some concepts like XOR

	Training set size		
	100	200	300
Training time(secs)	0.005	0.007	0.002
Prediction time(secs)	0.000	0.000	0.000
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.731343283582	0.766917293233	0.75

As we can see from this table, Decision Tree Classifier makes really fast predictions and tends to overfit on the training data.

3. For this project I have chosen SVM because it is memory-efficient(it only needs to store some subset of the data), makes fast predictions and it's F1-score is better than the score of others models. The fact that SVM has a high training complexity doesn't matter a lot, because our dataset isn't very big.

SVM works by trying to build a hyperplane between points of two different classes with the highest margin.

Model-tuning of the support vector machine.

SVM comes with lots of different hyper-parameters that can be optimized. First of all, there are different **kernels** like “sigmoid”, “poly”, “linear”, “rbf” and so on. We can also use our domain-specific kernels to make better predictions.

The C parameter trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors. [http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html]. C is some sort of penalty factor. If C is small we can underfit the model, when it is big – we can overfit it. In case we have lots of noisy data it makes sense to decrease the C value.

For this project, I have used several SVM parameters to optimize the model prediction.

Kernels: linear, radial basis function and polynome.

C: although default value of C(1.0) is a reasonable choice I have also tested values

10,100,1000.
gamma: default is equal to 1/number of features.
There are also several parameters that are specific only to some kernels like coef0 or degree for polynome kernel.
The resulting model has the following score: 0.8535[*] with following params: C = 0.1, kernel – polynomial with the degree of 5 and with gamma equals 0.0001.

*Due to the fact that train/test data were randomly split there were a few different results.