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:	67%
Number of features:	30

3. For this project I have chosen three machine learning models: Support Vector Machine, Gradient Boosting 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)	0.002	0.003	0.005
F1 score for training set	0.8484	0.8432	0.8571
F1 score for test set	0.7682	0.77	0.7552

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)	0.004	0.003	0.006
Prediction time(secs)	0.001	0.002	0.004
F1 score for training set	0.8554	0.8534	0.8665
F1 score for test set	0.79268	0.8176	0.797

- 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. Boosted Tree Classifier

Ensemble method uses multiple learning algorithms to obtain better performance than could be obtained from any individual model. Ensemble methods combine different hypotheses to produce a new one that will work better.

Boosted Tree Classifier is an ensemble method that uses a sequence of simple trees(DecisionTree) where each new tree is built for the prediction of the residuals of the preceding tree. Such model can describe complex non-linear functions.

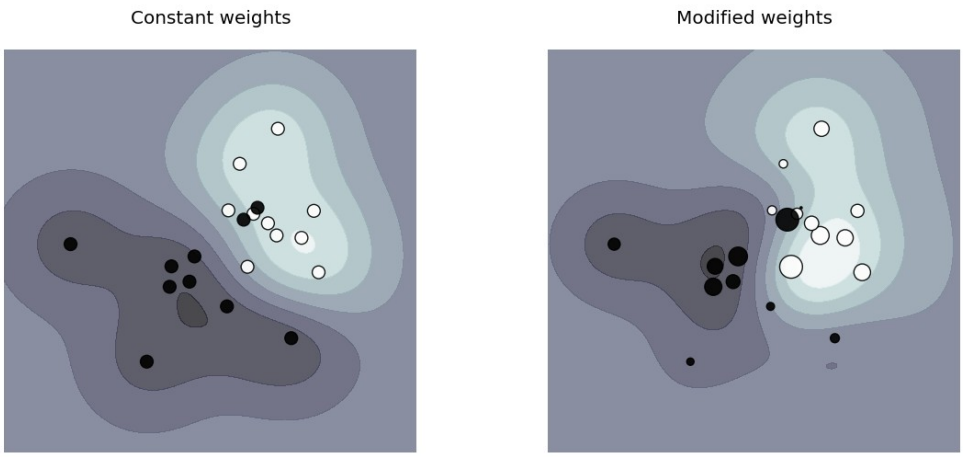
	Training set size		
	100	200	300
Training time(secs)	0.078	0.086	0.099
Prediction time(secs)	0.001	0.001	0.001
F1 score for training set	1.0	0.9849	0.96
F1 score for test set	0.7462	0.755	0.83687

3. **Reasons for choosing different algorithms:**

kNN:

- Zero training cost (one of the requirements of the project – was low training cost)
- The dataset is imbalanced (66%:33%) and kNN is robust to imbalanced problems
- robust to noisy data

SVM: For this project I have chosen SVM because it is memory-efficient(it only needs to store hyperplane), makes fast predictions, has lots of possibilities for tuning, ability to choose your own kernel function. SVM also provides some solutions for dealing with imbalanced problems like setting different class weights. The fact that SVM has a high training complexity doesn't matter a lot, because our dataset isn't very big.



- robust to outliers
- handling data of mixed type
- predictive power
- able to represent difficult non-linear functions
- fairly robust to overfitting

Model-tuning of the Gradient Boosted Classifier.

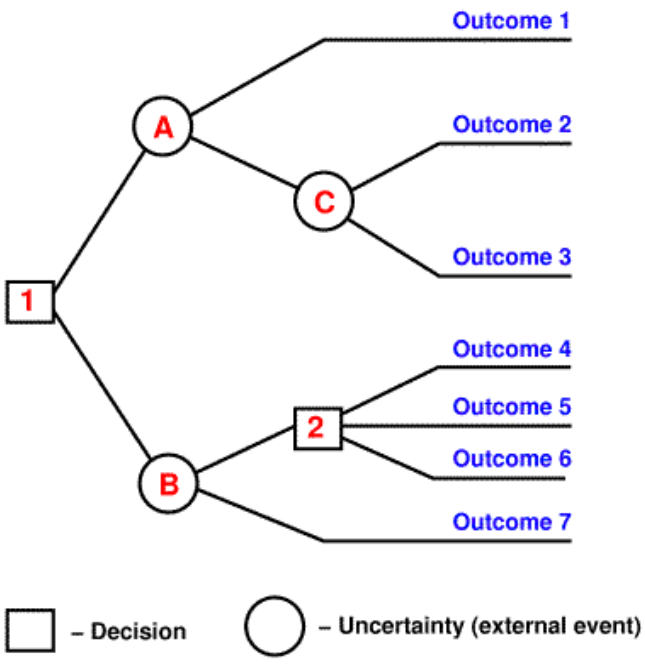
Gradient Boosted Classifier showed the best results among three models(the best F1-score) so I decided to tune the model to try to reach better results.

Gradient Boosted Classifier comes with lots of different hyper-parameters that can be optimized. First of all, number of estimators: the number of boosting stages to perform. Maximum depth – maximum of the individual estimator. Minimum samples leaf – the minimum number of samples allowed to be at the leaf node. This is important property, combining number of estimators with minimum samples can give good results. When model is overfitted by to many estimators we can solve this by increasing the number of samples in the leaf.

The final result of the model on the test set is: `0.8725 with n_estimators = 25 and minimum_number_of_samples = 16.`

Explanation of Gradient Boosted Classifier:

The idea behind GBC is pretty simple: it takes some simple algorithms, that can't describe some complex relations in the data and mix



them together to provide better results. In the case, of GBC it uses Decision Tree Classifier as a basic algorithm which is pretty straightforward (on the image above) and applies it to the dataset. After that, it takes those results and applies a new Decision Tree to the results(residuals) of the previous algorithm. Speaking less formally, what GBC does is that it continues to find better explanations of the data relations taking into the account “errors” of the previous models.

Resources:

http://www.alglib.net/other/nearestneighbors.php
http://www.cs.upc.edu/~bejar/apren/docum/trans/03d-algind-knn-eng.pdf
http://people.revoledu.com/kardi/tutorial/KNN/Strength%20and%20Weakness.htm
http://stats.stackexchange.com/questions/59630/test-accuracy-higher-than-training-how-to-interpret
http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html
http://www.svms.org/parameters
http://scikit-learn.org/
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.3344&rep=rep1&type=pdf
http://stats.stackexchange.com/questions/341/knn-and-unbalanced-classes
https://www.kaggle.com/c/otto-group-product-classification-challenge/forums/t/13719/intelligent-way-to-tune-parameters-in-gradient-boosting-classifiers
https://www.statsoft.com/Textbook/Boosting-Trees-Regression-Classification/button/1
https://en.wikipedia.org/wiki/Ensemble_learning