Machine Learning Nanodegree
P2: Student Intervention System
Jose Augusto Montiel

Which type of supervised machine learning problem is this, classification or regression? Why?

This represents a **Classification** problem since we need to identify each data entry either as a possible or not dropout case given the compiled training data.

Facts about the Dataset

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%**

Training and Evaluating Models

Three algorithms were chosen to model the given dataset: SVM, K-NN and Gradient Boosting.

Gradient Boosting was chosen due to its effectiveness in separating linear separable data, however, tuning it can be complexed due to the number its number of hyperparameters. As second model we have K-NN, it allows us to predict the classification of a given set by comparing it to its closest examples in the training data. It is considered a “Lazy Learner” since it doesn’t use the entire data set to generalize.

Support vector machines allows us to separate our dataset for classification while maximizing the the margins between our labels. It provided a very good result and its ease of tuning made it an excellent choice for selecting as the final algorithm for the given dataset.

Algorithm	Time Complexity	Space Complexity
SVM	**Learning:** O(n^2 * m)	**Learning:** O(n)

	Querying: O(1)	**Querying:** O(1)

K-NN	**Learning:**
	O(1)
	
	Querying:
	O(Log n + k)
		Learning:
		O(n)
		
		Querying:
		O(n)

Gradient Boosting	**Learning:**
	O(n^m)
	where m is the number of iterations for minimizing the loss function.
	
	Querying:
	O(1)
		Learning:
		O(n)
		
		Querying:
		O(1)

Algorithm	Strengths	Weaknesses
SVM	Provides the best plane that separates our labels while minimizing the generalization error. Through “kernel tricks”, it is able to take non linearly separable data into higher dimensions where it can be separated.	Learning time significantly increases as the data size and dimensionality increases
K-NN	Basically non-existent learning/training time (Lazy learner). Insensitivity to outliers.	Lack of generalization and High memory/space consumption
Gradient Boosting	Provides a generalized model by implementing weak learners and minimizing a loss function.	Difficult to tune due to its number of hyper-parameters. Hard/Slow to train in comparison to other models and algorithms.

Training and Prediction Results

Algorithm (Data Size)	**Time - Training (in secs)**	**Time - Querying (in secs)**	**F1 Score - Training**	**F1 Score - Querying**
SVM (100)	0.001	0.001	0.85	0.784
SVM (200)	0.003	0.002	0.869	0.776
SVM (300)	0.006	0.005	0.869	0.759
K-NN (100)	0.001	0.002	1.0	0.792
K-NN (200)	0.001	0.003	1.0	0.753
K-NN (300)	0.001	0.005	1.0	0.792
Gradient Boosting (100)	0.064	0.001	1.0	0.742
Gradient Boosting (200)	0.078	0.001	0.985	0.788
Gradient Boosting (300)	0.098	0.001	0.974	0.767

Choosing the Best Model

Examining the performance results exposed above we can conclude that Support Vector Machines in 0.001s and with only a third of the training data, was able to able to predict data with a F1 score of 0.784, only to be matched by K-NN's 0.792 score, but that comes with a lack of generalization and higher memory consumption due to its Lazy Learner nature; or by Gradient Boosting's 0.788 score, which needed twice the training data size and 78 times the training time. Overall, SVM seems to be the most efficient of all the options considered.

Support Vector Machine constructs a divisory line that successfully splits our data into different categories, while through an iterative procedure it maximizes the separation between the available categories, minimizing the cases of possible misclassification. In case a given data is not separable as-is, SVM can implement mathematical tricks in order to transport the data into higher dimensions where the data can be separated.

Given the training data, SVM was optimized using Gridsearch and the following results were obtained:

Hyperparameters

Kernel: RBF **Gamma**: 0.001 **C**: 100

Algorithm (Data Size)	Time - Training (in secs)	Time - Querying (in secs)	F1 Score - Training	F1 Score - Querying
SVM (300)	0.007	0.003	0.874	0.797

After tuning the hyperparameters of our Support Vector Machine we were able to reduce the querying time by 40% and increase the F1 Score by 5% while increasing the Training time by 16.7%.