**Why Gradient Boosted Tree (GBT) algorithm?**

Spark MLlib (Machine Learning Library) offers many algorithms. Generally, it offers algorithms within a) Unsupervised Learning or b) Supervised Learning method.

Algorithms include:

* Classification: logistic regression, **naive Bayes**,...
* Regression: generalized linear regression, survival regression,...
* Decision trees, **random forests**, and **gradient-boosted trees**
* Recommendation: alternating least squares (ALS)
* Clustering: **K-means**, Gaussian mixtures (GMMs),...
* Topic modeling: latent Dirichlet allocation (LDA)
* Frequent itemsets, association rules, and sequential pattern mining

However, my job is to think in the realm of [Plura.io](https://www.plura.io/). That is, my job is to find the most efficient algorithm that will aid in the use of machine learning to detect SQL Injection.

The first step was to figure out which of the two learning methods would be more efficient between Unsupervised and Supervised methods. Although both methods have their own strengths and weaknesses, I believe that in order to best detect SQL injection through machine learning, Supervised Learning algorithms would prove to be most efficient.

The difference between the two learning methods is simple.

Unsupervised method is when a computer is not given an labeled output. This focuses on finding relationship between the given inputs. Some examples include k-means clustering, and GMM.

Supervised method, on the other hand, is when a computer “learns by example”. Therefore, the input and output variables must be known and labeled, so that the computer knows how to get from point A to point B, and what they are. Think of it like a relationship between a teacher and a student. The teacher tells the student what the problem is, and the best answer to that problem. The student’s job is to learn how to solve the problem to reach the answer.

Another real-world example of supervised learning would be face detection. Basically, we want the computer to detect faces, and who they belong to. Logically, we would have to tell the computer what a face is, and what it is not. “This is the result/answer. Now, learn it”. However, the important aspect of Supervised Learning method that we must emphasize, is that of *generalization*. Generalization is basically making the computer able to learn one thing, and then making sure it can predict 10 other things we don’t teach it. This way the computer will not “freak out” when it is introduced “never seen before” data as input – data not encountered in training. Essentially this is the purpose of machine learning anyway – to train the machine to recognize patterns so that it can make predictions.

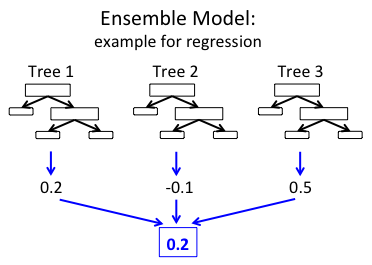
An important distinction to note is that between the training and test datasets. Training set is example used for learning where target value is known. Test set, however, should never be combined or used in training dataset so that we can avoid bias. With the goal of detecting and predicting SQL injection, I chose to use supervised learning method.

Within Supervised Learning method, I would like to divide the method into two sub-methods: Classification & Regression. Some example algorithms for classification include Support Vector Machine (SVM), logistic regression, decision tree classifier, and naive Bayes. Example algorithms for regression include linear regression, isotonic regression, and survival regression. We chose classification as the method to detect SQL injections.

Spark MLlib offers these algorithms, but my job is to pick the most efficient one, as mentioned earlier. I believe the best way is to study each algorithm to find out what they do and how. We would need to, then, run each algorithm with test data, and compare the results. But that might take too much time and resource, so after studying each, we narrowed the algorithms to a handful of them, namely SVM & Gradient-boosted tree classifier (GBT). I was responsible for testing GBT, while my coworker, Scott, tested SVM.

Accordingly, I had to study what GBT does to find out its pros and cons to see if it would be efficient in helping us reach our goal. So what is GBT, and why or why not should we use it?

Gradient Boosted (decision) Tree is a predicition model that is offered both in the realm of regression and classification. Basically, it is an ensemble (sequential collection) of weak prediction models. That is, GBT sequentially adds together individual decision trees one after another to make the overall ensemble model more accurate and powerful. Let’s look at an example diagram of an ensemble model below.



We can see that this example has three trees, with each tree predicting different values. These three predictions are, ultimately, combined together to produce the ensemble’s final prediction. “Boosting” is essentially an algorithm that aims to convert weak learners to strong learners, with the goal of decreasing error and increasing accuracy.

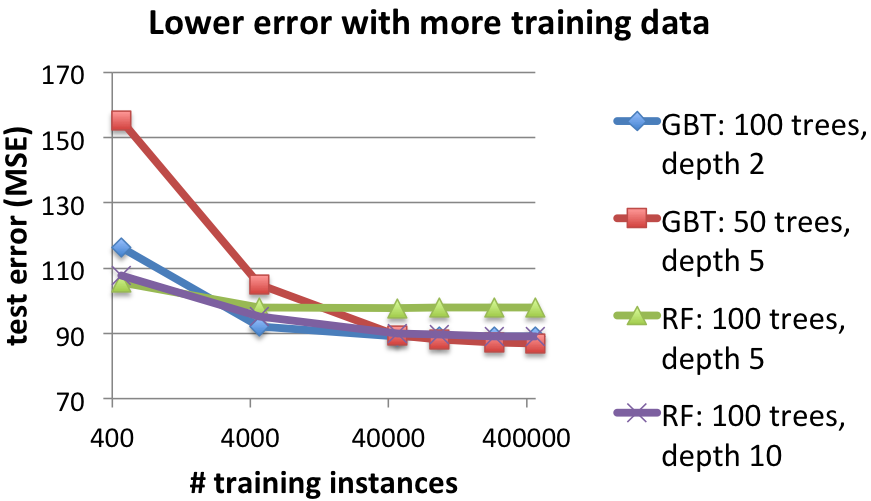
In a similar manner, GBT *sequentially* combines weak decision trees to make, for a lack of a better word, a bigger and more complex ensemble tree. So the decision tree starts off small, but grows as each successive tree is added. More importantly, the reason or purpose for the sequential summation of decision trees is to compensate for the error of the previous weak model (a technical term). Each successive tree should minimize the training error of the overall ensemble. Overall, the goal would be to reduce and minimize the residual between the prediction of the ensemble and the target function (as this is Supervised Learning method).

Mathematically, let P(x) represent the prediction of the ensemble, f(x) represents the target function (where we want to be), and R(x) be the residual between the two. P(x) = Ptree1(x) + Ptree2(x) + … (sum of individual trees).

Accordingly, R(x) = f(x) – P(x), which is what we are trying to minimize in order to increase accuracy.

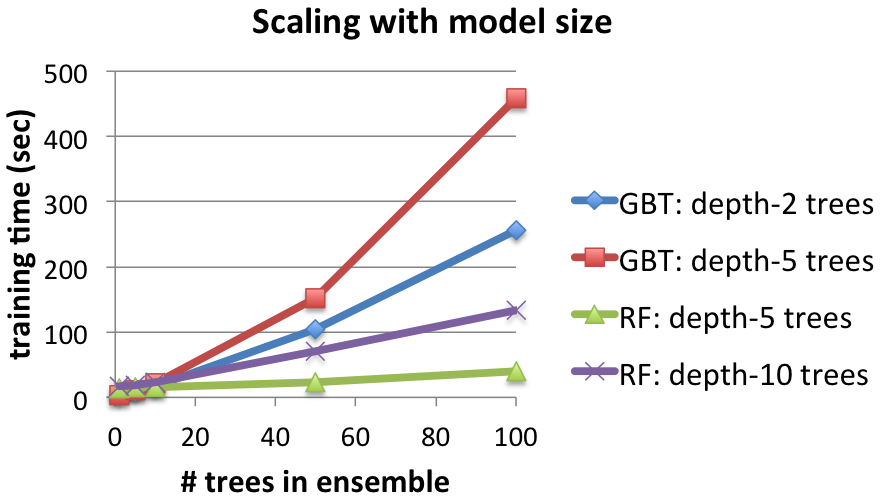
So you might be wondering what a “weak model” is. Let’s take an example of spam emails. To distinguish between spam and benign emails, we would set up arbitrary rules. For instance, promotional images, emails with only links, strings like “You won a prize of $$$” would raise suspicion for a spam email. On the other hand, non-spam emails would be emails that contain well-known domain names, or known sources. However, as you might have already realized, these rules are meager and not strong enough by themselves to successfully classify an email. These arbitrary rules are the “weak learners”. We would need to combine these into one ensemble so that the result is more accurate.

Boosted trees are well-known and widely-used for their relative accuracy and predictive ability.



We see in the example above that GBT with 50 trees, depth 5 produced the lowest test error (greater accuracy).

However, GBT does have its weaknesses. Mainly there are two that we must consider. **First**, GBT is prone to overfitting a training dataset quickly, and tuning is harder. Because GBT adds trees sequentially, test error increases as well after too many iterations. This probably means that we need to stop after a certain iteration even if the result is not 100% accurate. **Second**, GBT is prone to taking a longer average time than other algorithms, as training is not in parallel, but sequential. In other words, there is an inverse relationship between the number of weak learners (trees), and learning rate.



From this example, we can see that GBT took a significantly longer time with the addition of more depth.   
\_\_\_

With these ideas in mind, we need to make a decision whether or not GBT would be an efficient algorithm in training SQL injection datasets. To determine this, I will try to explain the concept of SQL injection as clearly as I can.

Hacking occurs when an unauthorized individual attempts and succeeds in obtaining private information from an invidual, organization, or company. There are many different methods that are deemed as hacking - SQL injection is just one of them. SQL injection, as the name suggests, is an attack in which the attacker injects malicious SQL statements in order to control the web application’s database server. More specifically, when a user (or attacker) makes an HTTP request, that request will be sent to the server, which will access the database through queries. Take a look at an example architecture below:



Our mission was to use machine learning algorithms to train the machine to not only recognize malicious SQL injection queries, but also to prevent them, if possible. We would do this by telling the machine which queries are clean/benign, and which are malicious. There are different types of SQL injections, which we will ultimately have to take into account individually through classification. For instance, there are error-based, Boolean-based, union-based, and blind SQL injection.

Personally, I believe that decision tree algorithms, including GBT, is, overall, not a bad idea. I believe every algorithm has its own strengths and weaknesses, and GBT is not an exception. If we were to use GBT to detect SQL injections, we *must* make sure that we do not overfit our dataset. This week, my goal is to use training datasets to get results using the gradient boosted decision tree classifer algorithm. Results will come out. Stay tuned ^^

References:

<https://spark.apache.org/docs/latest/ml-classification-regression.html#gradient-boosted-tree-classifier>

<https://spark.apache.org/docs/latest/ml-classification-regression.html#gradient-boosted-trees-gbts>

<http://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/>

<http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf>

<http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3103196/> (I like)

<https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf>

<https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html> (I like)

<https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/>

<https://www.analyticsvidhya.com/blog/2015/09/complete-guide-boosting-methods/>