**Support Vector Machine**

|  |
| --- |
| **What are the general applications of this model?** |
| Text Categorization,  Hand-Written Recognition,  Protein Classification,  Images Classification[[1]](#footnote-0) |
| **What are its strengths?** |
| Handle high dimensions problems,  Works if samples are fewer than dimensions,  Memory efficient,  Custom kernel allow multiples application. |
| **What are its weaknesses?** |
| All the datas must be labeled,  Not qualified for problem implying two classes,  BlackBox type problem[[2]](#footnote-1),  Work bad if samples are fewer than features.  Probability estimate is hard. |
| **Given what you know about the data so far, why did you choose this model to apply?** |
| The number of samples is much bigger than the number of features. We're probably in a high dimensional case. Since we're paying at CPU/Memory use, we're interested in a Fast/Memory efficient algorithm. We're especially interested in having a small testing time since we only have the 5th grade and dropouts to train on but we want to evaluate the entire school. SVMs are eager learners. We're not interested in probability estimate. Plus we know that SVMs work specially well as a binary classifier. |

**Decision Trees**

|  |
| --- |
| **What are the general applications of this model?** |
| Decision Analysis, Medical Decision, Data Mining[[3]](#footnote-2) |
| **What are its strengths?** |
| Drawable  Requires few preparation  Handle Numerical and Categorical datas  WhiteBox Model  Good on large dataset  You can use it when there’s inconsistencies[[4]](#footnote-3) |
| **What are its weaknesses?** |
| Might get stuck in local minima and has to be reset several times  Overfit if not pruned of depth-controlled  Cannot reach optimum trees due to neighbour only comparison  Mode affects the outcomes |
| **Given what you know about the data so far, why did you choose this model to apply?** |
| Contextually, we only have the 5th grade and dropouts to train on but we want to evaluate the entire school so we might be more interested in efficient testing than training. The dataset is filled with both numerical and categorical datas which can both be handled by decision trees. Since we may wish to feed the model with some kind of raw survey containing noises and unconsistencies, we need the robustness decision trees bring. We know that the dataset is unbalanced over the target feature 'passed?'. This leads to undefitted trees. We also know you can't prune using sklearn algorithm. This leads to overfitted trees. We hope that overfitting and underfitting might cancel each other out in some ways. |

**AdaBoost with Decision Trees as base learners**

|  |
| --- |
| **What are the general applications of this model?** |
| Spam Email  Data Mining  Natural Language Recognition |
| **What are its strengths?** |
| Tuning friendly,  Lots of differents parameters,  Efficient. |
| **What are its weaknesses?** |
| Pink noise sensitive,  Overfitting comes from the basic algorithm, so dependent on the base estimator,  May needs large data set in order to achieve the "weak learner" condition,  Lots of estimators choice to try. |
| **Given what you know about the data so far, why did you choose this model to apply?** |
| The Boosting-Decision Trees relation should take the robustness on outliers, data missing and inconsistencies of decision trees and add it to the fine tuning parameters of AdaBoost. Plus, because the tree shall remain small, we won't have to worry about the pruning problem. Small trees are also very easy to understand and may help to produce a understandable model. Plus it’s very efficient on testing. |

**Support Vector Machine**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Size* | *Training Time* | *Testing Time* | *F1-Score on training set* | *F1-Score on testing set* |
| *100* | 0.007 | 0.001 | 0.866141732283 | 0.797297297297 |
| *200* | 0.007 | 0.002 | 0.857142857143 | 0.823529411765 |
| *300* | 0.008 | 0.002 | 0.875 | 0.85161290322 |

**Decision Trees**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Size* | *Training Time* | *Testing Time* | *F1-Score on training set* | *F1-Score on testing set* |
| *100* | 0.002 | 0.000 | 1.0 | 0.65079350794 |
| *200* | 0.002 | 0.000 | 1.0 | 0.701492537313 |
| *300* | 0.003 | 0.000 | 1.0 | 0.811594202899 |

**AdaBoost with Decision Trees as base learners**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Size* | *Training Time* | *Testing Time* | *F1-Score on training set* | *F1-Score on testing set* |
| *100* | 0.164 | 0.006 | 0.991869918699 | 0.751879699248 |
| *200* | 0.123 | 0.007 | 0.888888888889 | 0.816901408451 |
| *300* | 0.120 | 0.007 | 0.857831325301 | 0.827586206897 |

**Choosing the Best Model**

|  |
| --- |
| **Based on the experiments you performed earlier, in 1-2 paragraphs explain to the board of supervisors what single model you chose as the best model. Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?** |
| Support Vector Classifier  It’s tradeoff between F1 (0.85) score and prediction time (0.002) was the best. Its F1 score is less than AdaBoost’s but it more reliable over resets and faster. It’s slower than Decision Tree but it’s F1 score is greater. Plus, it’s reliable on many sizes making it robust to real survey-like data sets.  It’s effectiveness leads it to be both memory and time efficient making it cheap to use in our contextual problem. Plus, seeing the high number of features (30) we’re happy to have an algorithm that can handle high dimensional cases and few samples. |
| **In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a Decision Tree or Support Vector Machine, how does it make a prediction).** |
| 2 different kind of points are placed on a paper, we wish to separate the two kinds using a line which is as the maximum distance from each sides nearest points.  We draw a line that roughly separate both of the categories and draw two border lines which are touching the two nearest points of each sides.  We erase the points that are very deep in their category, they won’t be affecting the outcomes.  We try to maximise the distance between each remaining points and the middle line while avoiding to have some points cross the border lines. This is equivalent as turning the line toward vertical angle until you can't continue without misclassifying some points. Best angle and initiality of the line is laying where the distance between all the remaining points and the center-line is minimum.  If there’s more than 2 kind of points, we add as many lines as we need to separate them all from each other in the same way we did. If we cannot separate the points with line we pitch one kind of point in three dimension (or more) and try separate it with a plane (or hyperplane). Then throw the points back in the dimension they’ve left and look at the shape it makes. |
| **What is the model's final F1 score?** |
| 0.831168831169 |

1. https://en.wikipedia.org/wiki/Support\_vector\_machine#Applications [↑](#footnote-ref-0)
2. https://en.wikipedia.org/wiki/Support\_vector\_machine#Issues [↑](#footnote-ref-1)
3. http://www.cbcb.umd.edu/~salzberg/docs/murthy\_thesis/survey/node32.html [↑](#footnote-ref-2)
4. https://en.wikipedia.org/wiki/Decision\_tree\_learning#Decision\_tree\_advantages [↑](#footnote-ref-3)