


Logistic Regression vs Decision Trees vs SVM: Part II

 edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part2



This is the 2nd part of the series. Read the first part here: [Logistic Regression Vs Decision Trees Vs SVM: Part I](#)

In this part we'll discuss how to choose between Logistic Regression , Decision Trees and Support Vector Machines. The most correct answer as mentioned in the [first part](#) of this 2 part article , still remains it depends. We'll continue our effort to shed some light on, it depends on what. All three of these techniques have certain properties inherent by their design, we'll elaborate on some in order to provide you with few pointers on their selection for your particular business problem.

We'll start with **Logistic Regression** , the most prevalent algorithm for solving industry scale problems, although its losing ground to other techniques with progress in efficiency and implementation ease of other complex algorithms.

A very convenient and useful side effect of a logistic regression solution is that it doesn't give you discrete output or outright classes as output. Instead you get probabilities associated with each observation. You can apply many standard and custom performance metrics on this probability score to get a cutoff and in turn classify output in a way which best fits your business problem. A very popular application of this property is scorecards in the financial industry ,where you can adjust your threshold [cutoff] to get different results for classification from the same model. Very few other algorithms provide such scores as a direct result. Instead their outputs are discreet direct classifications. Also, logistic

regression is pretty efficient in terms of time and memory requirement. It can be applied on distributed data and it also has online algorithm implementation to handle large data on less resources.

In addition to above , logistic regression algorithm is robust to small noise in the data and is not particularly affected by mild cases of multi-collinearity. Severe cases of multi-collinearity can be handled by implementing logistic regression with L2 regularization, although if a parsimonious model is needed , L2 regularization is not the best choice because it keeps all the features in the model.

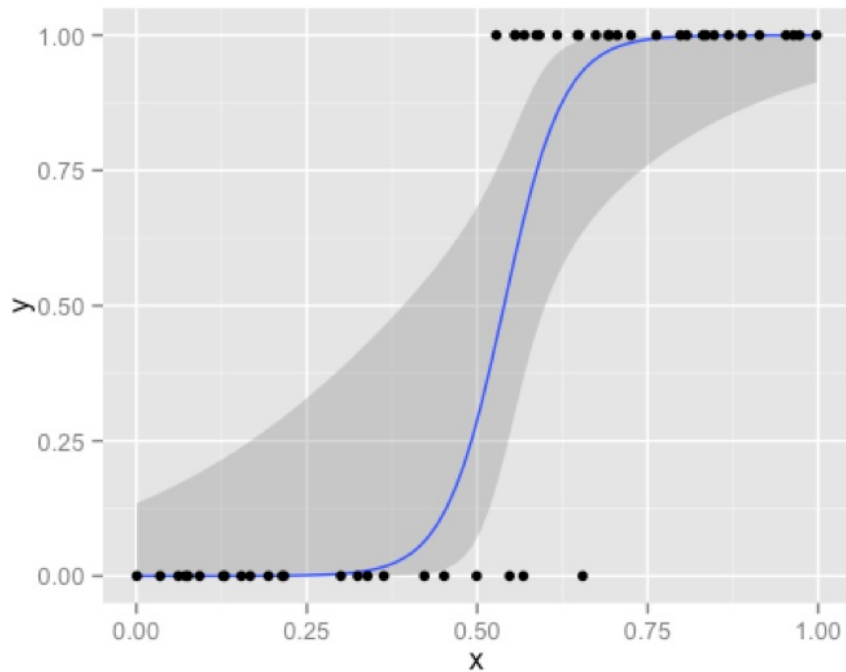
Where logistic regression starts to falter is , when you have a large number of features and good chunk of missing data. Too many categorical variables are also a problem for logistic regression. Another criticism of logistic regression can be that it uses the entire data for coming up with its scores. Although this is not a problem as such , but it can be argued that “obvious” cases which lie at the extreme end of scores should not really be a concern when you are trying to come up with a separation curve. It should ideally be dependent on those boundary cases, some might argue. Also if some of the features are non-linear, you’ll have to rely on transformations, which become a hassle as size of your feature space increases. We have picked few prominent pros and cons from our discussion to summaries things for logistic regression.

Logistic Regression Pros:

- Convenient probability scores for observations
- Efficient implementations available across tools
- Multi-collinearity is not really an issue and can be countered with L2 regularization to an extent
- Wide spread industry comfort for logistic regression solutions [oh that’s important too!]

Logistic Regression Cons:

- Doesn’t perform well when feature space is too large
- Doesn’t handle large number of categorical features/variables well
- Relies on transformations for non-linear features
- Relies on entire data [Not a very serious drawback I’d say]



Let's discuss Decision Trees and Support Vector Machines .

Decision trees are inherently indifferent to monotonic transformation or non-linear features [this is different from non linear correlation among predictors] because they simply cut feature space in rectangles [or (hyper)cuboids] which can adjust themselves to any monotonic transformation. Since decision trees anyway are designed to work with discrete intervals or classes of predictors, any number of categorical variables are not really an issue with decision trees. Models obtained from decision tree is fairly intuitive and easy to explain to business. Probability scores are not a direct result but you can use class probabilities assigned to terminal nodes instead. This brings us to the biggest problem associated with Decision Trees, that is, they are highly biased class of models. You can make a decision tree model on your training set which might outperform all other algorithms but it'll prove to be a poor predictor on your test set. You'll have to rely heavily on pruning and cross validation to get a non-over-fitting model with Decision Trees.

This problem of over-fitting is overcome to large extent by using Random Forests, which are nothing but a very clever extension of decision trees. But random forest take away easy to explain business rules because now you have thousands of such trees and their majority votes to make things complex. Also by decision trees have forced interactions between variables , which makes them rather inefficient if most of your variables have no or very weak interactions. On the other hand this design also makes them rather less susceptible to multicollinearity. Whew!

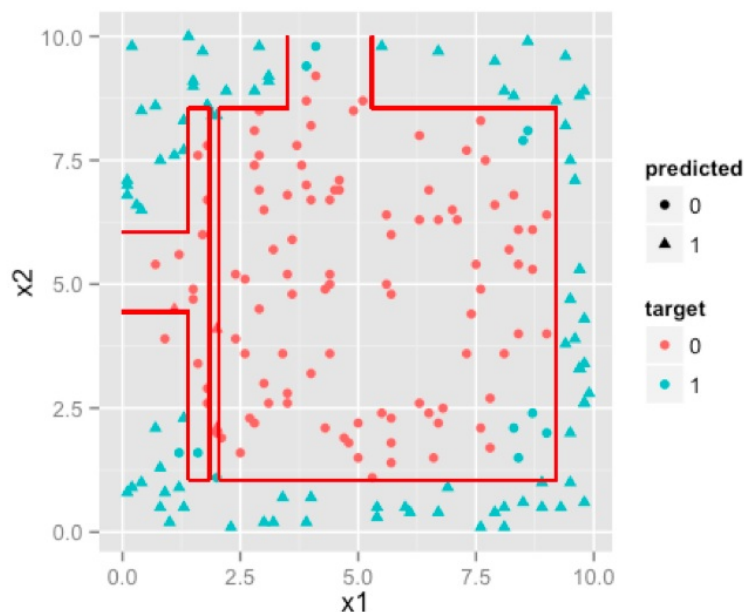
Summarizing Decision Trees:

Decision Trees Pros:

- Intuitive Decision Rules
- Can handle non-linear features
- Take into account variable interactions

Decision Trees Cons:

- Highly biased to training set [Random Forests to your rescue]
- No ranking score as direct result



Now to **Support Vector Machines**. The best thing about support vector machines is that they rely on boundary cases to build the much needed separating curve. They can handle non linear decision boundaries as we saw earlier. Reliance on boundary cases also enables them to handle missing data for “obvious” cases. SVM can handle large feature spaces which makes them one of the favorite algorithms in text analysis which almost always results in huge number of features where logistic regression is not a very good choice.

Result of SVMs are not as not as intuitive as decision trees for a layman. With non linear kernels, SVMs can be very costly to train on huge data. In Summary:

SVM Pros:

- Can handle large feature space
- Can handle non-linear feature interactions
- Do not rely on entire data

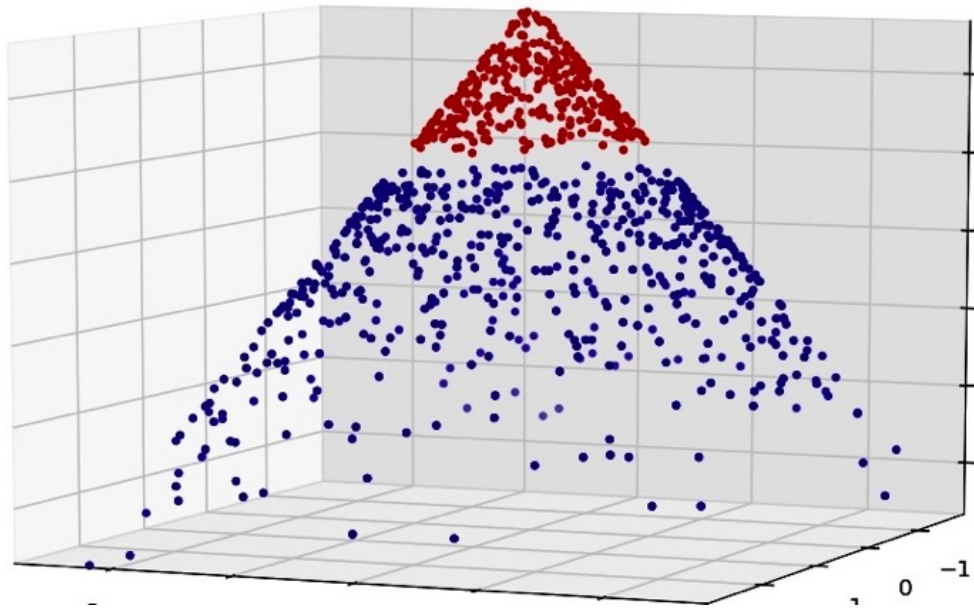
SVM Cons:

- Not very efficient with large number of observations
- It can be tricky to find appropriate kernel sometimes

I have tried to compile a simple workflow for you to decide which algorithm to use out of these three, which is as follows:

- Always start with logistic regression, if nothing then to use the performance as baseline

- See if decision trees (Random Forests) provide significant improvement. Even if you do not end up using the resultant model, you can use random forest results to remove noisy variables
- Go for SVM if you have large number of features and number of observations are not a limitation for available resources and time



At the end of the day, remember that good data beats any algorithm anytime. Always see if you can engineer a good feature by using your domain knowledge. Try various iterations of your ideas while experimenting with feature creation. Another thing to try with efficient computing infra available these days is to use ensembles of multiple models. We'll discuss them next, so, stay tuned!

- [About](#)
- [Latest Posts](#)



Lalit Sachan

Share this on



Follow us on

