Model Stacking | Scott Schwartz

May 20, 2020

1 Introduction

Much like subject matter experts, different models have different characteristics. This can lead different models to exhibit differential predictive performance strength (i.e., "specializations") in different domains within a given problem. So just as it is intuitively understandable that there are benefits available from "group-informed decision making", so too is it naturally apparent that "combining predictions" from different models could boost predictive performance. This technique is readily available for us to use via

```
sklearn.ensemble.
StackingClassifier([e for e in estimators],
                   stack_method=<...>, passthrough=<...>,
                   final_estimator=meta_classifier).fit(X_train, Y_train)
```

Using `stack_method='predict_proba'` will consider prediction strength from each model used (enumerated in `estimators`), while a more democratic approach can be taken by setting this value to `'predict'`. Another key considerations is `passthrough`. Setting `passthrough=False` stabilizes prediction towards a central tendency via some sort of weighted averaging or bagging (dictated by the `meta_classifier` choice), while setting this to `True` passes the original features `X_train` to the `meta_classifier`. With this latter approach the "mediator" model `meta_classifier` attempts to learn the parts of the feature space in which each estimator outperforms the others and uses this information to inform the final adjudication process.

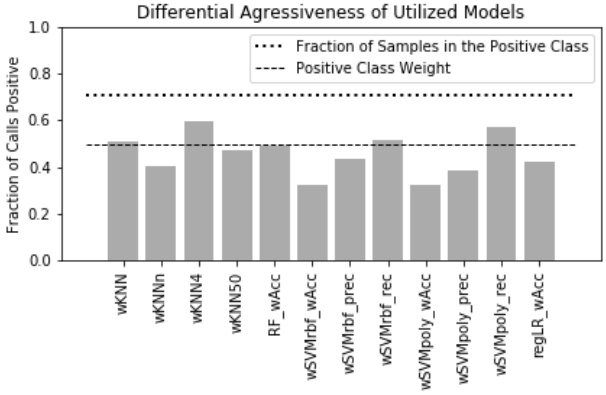
2 Practical Advice

In an application of this technique – known as **Model Stacking** – to the Indian Liver Patient Records (ILPR) dataset, a number of useful, practical considerations were encountered:

- n=583 (71.4% positive class) is a small data set, so we must remain vigilant to the risk of overfitting.
- As the class imbalance (∼2.4:1) does not represent any obvious population level prevalence, we chose to evaluate model performance using weighted accuracy $\frac{\frac{TP}{TP+FN}+\frac{TN}{TN+FP}}{2}$.
  - Other choices for performance evaluation can reasonably be considered. E.g., we decided against a recall-biased metric *only* because we didn't have clear expectations of actual FN and FP costs.
  - However, blindly evaluating *de facto* accuracy "as is" filters assessment through class imbalance, which reduces interpretability since in our case class imbalance seemingly appears to be arbitrary.
  - *Weighted accuracy* can be implemented as just a "plain" accuracy calculation, but where the observations from class $j = 0, 1$ are given weight $\frac{1}{2\pi_j}$, which redistributes $n = \sum_i \frac{1}{2\pi_{y_i}}$ by class.
  - A *final hold out test set* of 75 positive and 25 negative cases was set aside and not used for model fitting. This balance was chosen to limit the number of negative cases removed from the training data. Thus, for this set of data, negative class observations are thus worth $\frac{1}{2\pi_0} = \frac{1}{2\frac{1}{4}} = 2$ and positive cases are worth $\frac{1}{2\pi_0} = \frac{1}{2\frac{3}{4}} = \frac{2}{3}$, which redistributes $100 = n = 2 \times 25 + \frac{2}{3} \times 75$ by class.
  - Most `sklearn` models can be passed `class_weights='balanced'` which will cause the model to fit on a balanced class representation; however, note that `probability='True'` for `SVC` does not similarly calibrate the probabilities returned from calls to the `.predict_proba()` method.
  - **We are interested in fitting models on balanced class data representations so that we do not introduce any class preference bias into our final models solely as a result of any arbitrary (ungeneralizable) class imbalance present in our current data set.**

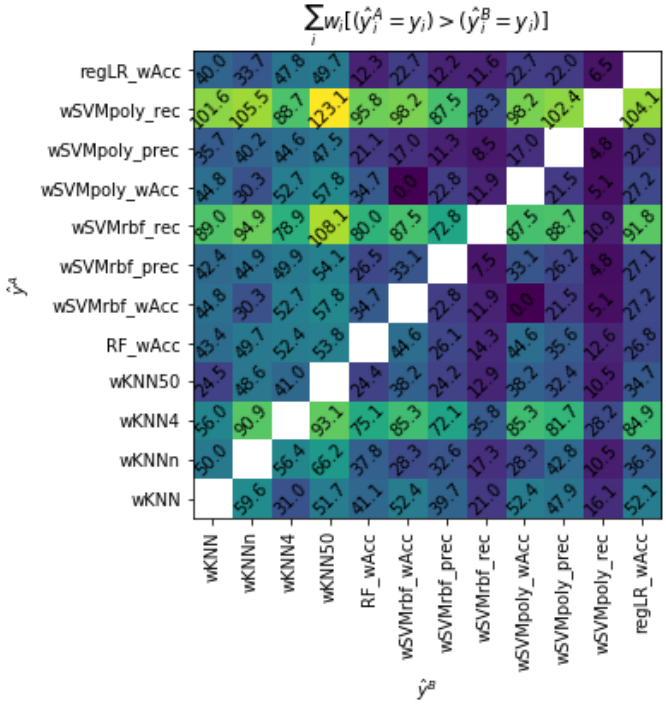
- `estimators` should include diverse models; but given this set, `meta_classifier` is trained on the test-fold predictions from cross validation fitting of submodels, which provides us the following rationale:
  - If, to create `estimators`, different model classes are individually tuned and fitted using `X_train`, then directly fitting `meta_classifier` with submodel predictions on `X_train` introduces leakage since the submodels were optimized for `Y_train`: instead, using test-fold predictions to fit `meta_classifier` only relies on submodel tuning parameters being optimized to `Y_train`, but the submodel fits themselves are not optimized to these predictions. This means that the `meta_classifier` determines adjudication between submodels on the basis of their observed generalized performance. I.e., `meta_classifier` is fit using observed "out of sample" capability.

- A diverse model ensemble `estimators` can be created by drawing upon a wide variety of model classes; and further, within model classes, by generating multiple characteristically distinct models with different tuning parameter choices. The creation of differently tuned models within the same model class can be achieved by using different optimization metrics. For example, a model selection metric might be *weighed accuracy* directly, or it might be an *Fbeta-measure* to bias the choice towards a more *recall* or *sensitivity* oriented model.



- The plot immediately above on the right shows our ensemble of submodels characterized by their propensity to positive prediction. All models were trained on class balanced data representations; however, not all final tuning parameter choices were made to directly optimize for *weighed accuracy*. This can be seen by the variation in propensity to positive prediction observed across the ensemble.

- Model performance diversity provides a diagnostic indicator of the potential utility of model stacking:



In the figure on the left, the model on each row is compared to the model on each column as follows: predictions for each model are made and compared to the truth, and the number of times the row model is correct while the column model is wrong are tallied. The table is not symmetric since the models will outperform each other for different observations. The more isolated the feature space localizations of differential performance, the more opportunity for preferential (specialized) subestimator usage.

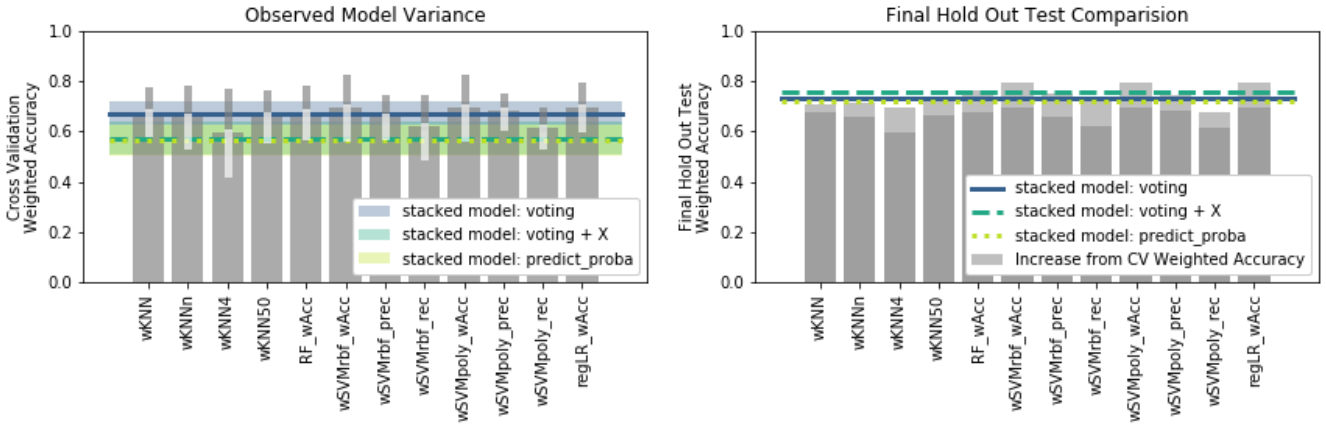
3 Page 3

3.1 Left Panel of Figure Below

Initial expected performance of the ensemble (stacked) estimator did not definitively exceed the performance of individual submodels. However, model variance was generally characterizable as "high" across the board, while the stabilization expected from a stacking approach such as that which we used did indeed appear to be present. At this stage, moving forward with the standard voting ensemble appears reasonable.

3.2 Right Panel of Figure Below

Interestingly, despite the more attractive performance of the voting ensemble during the initial cross validation scoring, all stacked methods performed approximately equally well in the final hold out test set. This might be indicative of the generally excellent generalization character of stacked ensemble methods; or, the testing set may just not be that challenging. In favor of the latter explanation – and very indicative of the generally challenging nature of our under powered context – all submodels performed better on the final hold data than they did in the initial cross validation context.[1] Nonetheless, the stacked methods performance was generally on par with the submodels comprising them.



4 Conclusions

While this data set did not provide a clear demonstration of the performance improvement opportunities available from model stacking, this perhaps isn't too surprising. This is a very small data set. And additionally, the predictive signal in the data was never observed to be particularly strong. I.e., despite the extremely diverse set of prediction techniques applied to this data, the predictive (weighted) accuracy was always observed to be between ∼70-75%. This is fairly low, and indeed, ROC curves (not shown) demonstrated that high sensitivity levels are only obtained at the cost of an extremely high number of false positive calls. While the counts of the heat matrix on page 2 are enticing, if there is no systematic differentiation between model performance (or, worse yet, spuriously detectable differentiation) then there's nothing to do except average together differentially performing models, which would lead to relatively "average" performance of the kind we observed. Indeed, the only consistency observed between repeated runs of the full pipeline was that there wasn't any exhibited consistency. My feeling is that the `StackedClassifier sklaern.Pipeline` developed in this effort will prove to be incredibly useful resource whose value moving forward will make itself known in more data rich contexts.

[1] That's right, the lighter gray color in the plot on the right is not our usual "out of sample" performance drop; rather, in this data, for this hold out set, it is performance improvement.(!)