

# 1 Scenario

Let us assume that we have a **small** labeled dataset from one domain (we are going to call this dataset the *TargetDataset*) and [usually] it is costly to obtain new labelled data from the same domain (so the quantity is limited).

Also, let us also assume that we have one or more **large** and labeled datasets from a *related* domains (we are going to call these datasets the *SourceDatasets*) and [usually] it is affordable to obtain new labelled data from these domains.

If we would like to build a model for the *TargetDataset* using it alone, then the model will likely perform undesirably as the dataset is quite small!

The idea is to make maximum use of the *SourceDatasets* to build a model for the *TargetDataset* and classify data from its domain. In other words, we want to reuse data from the source tasks to augment the target task's training data (this is Instance Transfer Learning)

In this experiment, I have used the technique explained in a paper called "*Selective Transfer Between Learning Tasks Using Task-Based Boosting*" to gain knowledge from the *Source* data and use it in training a classifier for the *Target* data. The proposed algorithm is called *TransferBoost* and it is based on the classical *AdaBoost* Algorithm!

## 2 How TransferBoost Works (from the abovementioned paper)

As the source and target data are from different but related domains, the two tasks (i.e. source and target) have different distributions. Yet, some of the source tasks' data could have been drawn from the target task's distribution. Such data could then be used as additional training data for the target task.

*TransferBoost* attempts to automatically select individual data from the source tasks to augment the target tasks training data. It automatically determines the weight to assign to each source instance in learning the target task's model, building on the *AdaBoost* algorithm. *TransferBoost* iteratively constructs an ensemble of classifiers, reweighting both the source and target data via two types of boosting: individual and task-based. It increases the weight of individual mispredicted instances following *AdaBoost*. In parallel, it also performs task-based boosting by reweighting all instances from each source task based on their aggregate transfer to the target task.

In effect, *TransferBoost* increases the weight of source tasks that show positive transfer to the target task, and then reweights the instances within each task via *AdaBoost*.

## 3 Experiment I

### 3.1 The Data

- By looking at the table provided with the Eve data, I have extracted and merged labeled data from the Sapphire Channel (Sapphire Active/Inactive) for assays TS3 and TS6. This

is for strain *Plasmodium vivax*. This is going to be our *SourceDataset*

- I have also extracted labeled data from the Venus Channel (Venus Active/Inactive) for assay TS6. This is for strain *Plasmodium falciparum*. This is going to be our *TargetDataset*
- I have split the *TargetDataset* into two subdatasets. One for training and one for testing. Notice that this training subdataset is going to be our actual *TargetDataset*
- Now our datasets look like:
  - *SourceDataset* has 2781 instances (for Pv from TS3 and TS6 – file TS3-TS6-Pv.arff)
  - *TargetDataset* has 46 (for Pf from TS6 – file TS6-Pf.arff)
  - *TestDataset* has 1389 (for Pf from TS6 – file TS6-Test-Pf.arff)

### 3.2 Experimental Setup

The authors of the paper have provided the java source code of their implementation so I have downloaded it, plugged it into WEKA's source code and recompiled WEKA.

Remember that our *TargetDataset* is quite small and our *SourceDataset* is large and from a related domain.

We will use our *TestDataset* for evaluation (it is from the same domain as *TargetDataset*)

Here is what I have done:

- I have built a classification model with the TransferBoost Algorithm using both the *Target* and *Source* Datasets to carry out Transfer Learning. I used *Decision Stump* as the base classifier.
- I have built classification models with WEKA's NaiveBayes, SVM, KNN and J48 Decision Trees using the *TargetDataset* only. This is because usually we build models using data from the same domain!

### 3.3 Experimental Results

After building the models as explained above, I have evaluated them using the *TestDataset* which is of the same domain as the *TargetDataset*. I have counted Actual vs Predicted results. The following table shows how many miss-classifications each model makes:

TransferBoost	NaiveBayes	SVM	KNN	J48 Decision Trees
6	115	10	9	25

Observe that the TransferBoost model (the one that does Transfer Learning) makes less classification errors meaning it outperforms models built using the *TargetDataset* alone.

```

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      44      95.6522 %
Incorrectly Classified Instances    2      4.3478 %
Kappa statistic                    0.7278
Mean absolute error                0.0498
Root mean squared error            0.2128
Relative absolute error            23.8126 %
Root relative squared error        67.5244 %
Total Number of Instances          46

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                1      0.4      0.953      1      0.976      0.976      Inactive
                0.6      0      1      0.6      0.75      0.976      Active
Weighted Avg.   0.957      0.357      0.959      0.957      0.952      0.976

=== Confusion Matrix ===
  a  b  <-- classified as
41  0  |  a = Inactive
 2  3  |  b = Active

```

(a) TransferBoost TL Algo

```

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      45      97.8261 %
Incorrectly Classified Instances    1      2.1739 %
Kappa statistic                    0.8969
Mean absolute error                0.0217
Root mean squared error            0.1474
Relative absolute error            10.391 %
Root relative squared error        46.7775 %
Total Number of Instances          46

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                0.976      0      1      0.976      0.988      1      Inactive
                1      0.024      0.833      1      0.909      1      Active
Weighted Avg.   0.978      0.003      0.982      0.978      0.979      1

=== Confusion Matrix ===
  a  b  <-- classified as
40  1  |  a = Inactive
 0  5  |  b = Active

```

(b) NaiveBayes

```

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      43      93.4783 %
Incorrectly Classified Instances    3      6.5217 %
Kappa statistic                    0.543
Mean absolute error                0.0652
Root mean squared error            0.2554
Relative absolute error            31.1731 %
Root relative squared error        81.021 %
Total Number of Instances          46

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                1      0.6      0.932      1      0.965      0.7      Inactive
                0.4      0      1      0.4      0.571      0.7      Active
Weighted Avg.   0.935      0.535      0.939      0.935      0.922      0.7

=== Confusion Matrix ===
  a  b  <-- classified as
41  0  |  a = Inactive
 3  2  |  b = Active

```

(c) SVM

```

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances      42      91.3043 %
Incorrectly Classified Instances    4      8.6957 %
Kappa statistic                    0.6183
Mean absolute error                0.087
Root mean squared error            0.2949
Relative absolute error            41.5641 %
Root relative squared error        93.555 %
Total Number of Instances          46

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                0.927      0.2      0.974      0.927      0.95      0.863      Inactive
                0.8      0.073      0.571      0.8      0.667      0.863      Active
Weighted Avg.   0.913      0.186      0.931      0.913      0.919      0.863

=== Confusion Matrix ===
  a  b  <-- classified as
38  3  |  a = Inactive
 1  4  |  b = Active

```

(d) J48 Decision Tree

Figure 1: Stats after running 10 Fold CV

## 4 Experiment II

In this experiment, I am going to try and do TL at assay level. Meaning I will use Active/Inactive labelled datasets from the eve data (assays TS3,4,5,6,7)

### 4.1 The Data

- For the *SourceDataset*, I have used TS3 (1346 instances (4 Active) – file TS3-Labeled.arff)
- I have randomly split TS5 into two datasets:
  - *TargetDataset* ... 278 instances (3 Active) – file TS5-Labeled-Target.arff
  - *TestDataset* ... 1116 instances (5 Active) – file TS5-Labeled-Test.arff

### 4.2 Experimental Setup

Exactly the same as Experiment I

### 4.3 Experimental Results

After building the models as explained above, I have evaluated them using the *TestDataset* which is of the same domain as the *TargetDataset*. I have counted Actual vs Predicted results. The following table shows how many miss-classifications each model makes:

TransferBoost	NaiveBayes	SVM	KNN	J48 Decision Trees
3	8	3	5	5

Observe that the TransferBoost model (the one that does Transfer Learning) makes less classification errors meaning it outperforms models built using the *TargetDataset* alone.

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      277          99.6403 %
Incorrectly Classified Instances    1            0.3597 %
Kappa statistic                    0.7983
Mean absolute error                 0.0036
Root mean squared error             0.06
Relative absolute error             14.3434 %
Root relative squared error         57.8392 %
Total Number of Instances          278

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          1      0.333    0.996      1      0.998    0.727  Inactive
          0      0.667      0      1      0.667      0.8    0.683  Active
Weighted Avg.    0.996    0.33    0.996    0.996    0.996    0.727

=== Confusion Matrix ===

  a  b  <-- classified as
275  0 | a = Inactive
 1   2 | b = Active

```

### (a) TransferBoost TL

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      273          98.2014 %
Incorrectly Classified Instances    5            1.7986 %
Kappa statistic                    0.4363
Mean absolute error                 0.0193
Root mean squared error             0.1351
Relative absolute error             76.7381 %
Root relative squared error         130.2932 %
Total Number of Instances          278

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          0      0.333    0.996    0.985    0.991    0.813  Inactive
          1      0.667    0.015    0.333    0.667    0.444    0.676  Active
Weighted Avg.    0.982    0.33    0.989    0.982    0.985    0.812

=== Confusion Matrix ===

  a  b  <-- classified as
271  4 | a = Inactive
 1   2 | b = Active

```

### (b) NaiveBayes

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      274          98.5612 %
Incorrectly Classified Instances    4            1.4388 %
Kappa statistic                    0.3261
Mean absolute error                 0.0144
Root mean squared error             0.12
Relative absolute error             57.1057 %
Root relative squared error         115.6775 %
Total Number of Instances          278

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          0      0.993    0.667    0.993    0.993    0.993    0.663  Inactive
          1      0.333    0.007    0.333    0.333    0.333    0.663  Active
Weighted Avg.    0.986    0.66    0.986    0.986    0.986    0.663

=== Confusion Matrix ===

  a  b  <-- classified as
273  2 | a = Inactive
 2   1 | b = Active

```

### (c) SVM

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      273          98.2014 %
Incorrectly Classified Instances    5            1.7986 %
Kappa statistic                    -0.0087
Mean absolute error                 0.0187
Root mean squared error             0.1339
Relative absolute error             74.3517 %
Root relative squared error         129.1475 %
Total Number of Instances          278

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          0      0.993      1      0.989    0.993    0.991    0.613  Inactive
          1      0      0.007      0      0      0      0.613  Active
Weighted Avg.    0.982    0.989    0.978    0.982    0.98    0.613

=== Confusion Matrix ===

  a  b  <-- classified as
273  2 | a = Inactive
 3   0 | b = Active

```

### (d) J48 Decision Tree

Figure 2: Stats after running 10 Fold CV