

FINAL INDEPENDENT STUDY REPORT (MIS 6950)

Subhadeep Nag (Deep)

A01631525

Course: MIS 6950 Independent Study

Instructor: Dr. Y.S. Kim

TOPIC: Feature Selection and Transfer Learning

AREA: Data Mining, Machine Learning

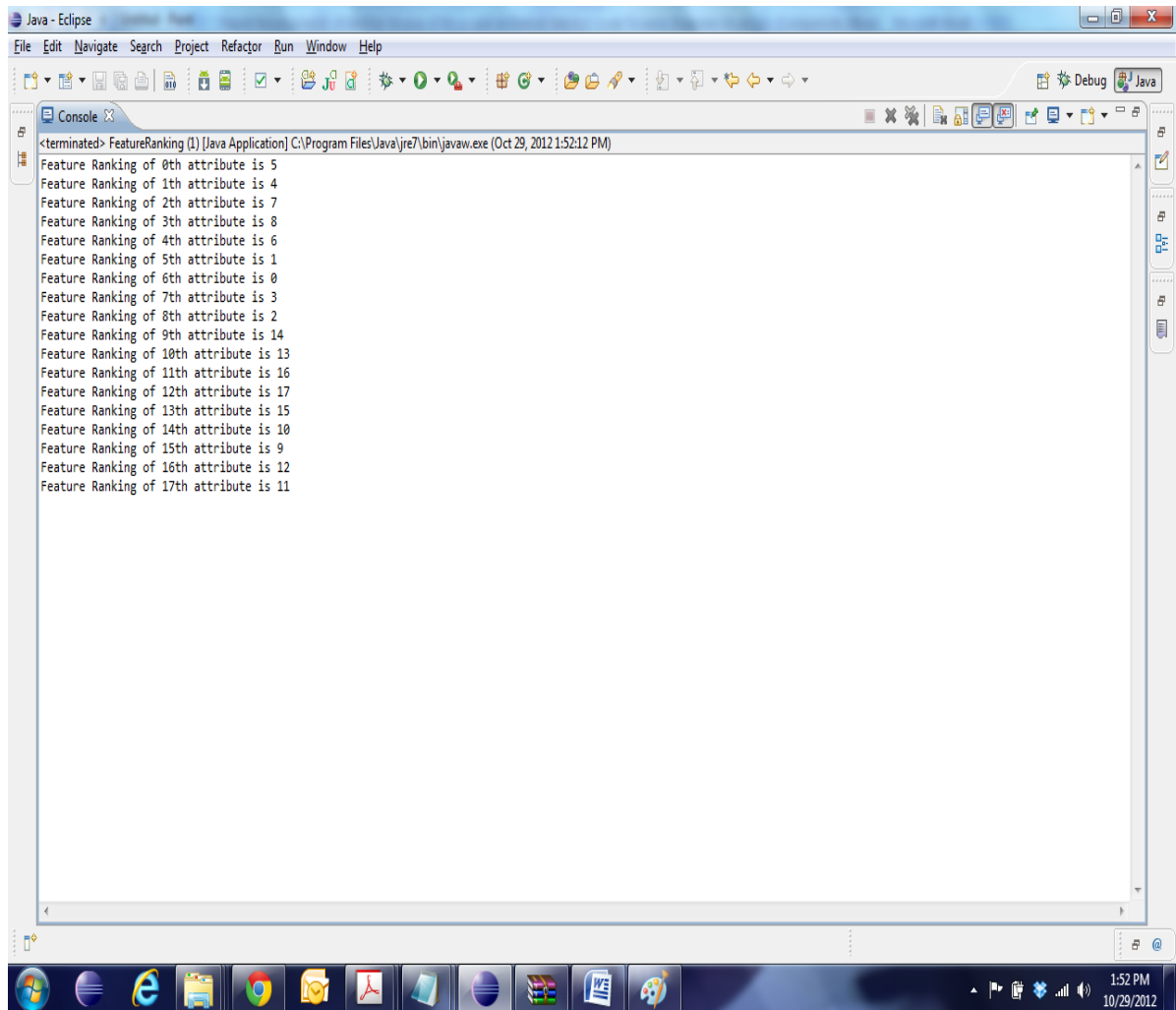
Abstract: Churn Prediction is used to increase the value of customers in a business enterprise. In this project, two phases are performed. In the first phase, a Series of Feature Selection Algorithms is applied on a Customer Churn Data-Set to find the best features in a Customer churn. In the second phase, A TransferBoost algorithm is designed via the AdaBoost algorithm and implementing sequential minimal optimization algorithm for training a support vector classifier using polynomial or RBF kernels. The new TransferBoost Algorithm is applied on a Multi-Domain Sentiment Data-Set, to find the accuracy of the classifier applied on the Source and Target Domains. This method performs boosting at both the instance level and the task level using a series of SVMs, assigning higher weight to those source tasks that show positive transferability to the target task, and adjusting the weights of individual instances within each source task via AdaBoost.

Phase 1: Feature Selection:

The Performance and Output of a Series of Feature Selection Algorithms is tested on a Customer Churn Data-Set, the process is described as follows:

- 1) **The First Feature Selection:** The first feature selection is performed on the whole dataset using linear Support Vector Machine, which runs in linear time. For a linear SVM, the feature importance can be derived from the weight vector of the hyper-plane, a procedure known as *Recursive Feature Elimination SVM* (SVM-RFE). The SVM-RFE ranks the features, and the best 10 features are selected from the dataset. *Lower the rank, better is the attribute.*

The Output is displayed as follows:



```
<terminated> FeatureRanking (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Oct 29, 2012 1:52:12 PM)
Feature Ranking of 0th attribute is 5
Feature Ranking of 1th attribute is 4
Feature Ranking of 2th attribute is 7
Feature Ranking of 3th attribute is 8
Feature Ranking of 4th attribute is 6
Feature Ranking of 5th attribute is 1
Feature Ranking of 6th attribute is 0
Feature Ranking of 7th attribute is 3
Feature Ranking of 8th attribute is 2
Feature Ranking of 9th attribute is 14
Feature Ranking of 10th attribute is 13
Feature Ranking of 11th attribute is 16
Feature Ranking of 12th attribute is 17
Feature Ranking of 13th attribute is 15
Feature Ranking of 14th attribute is 10
Feature Ranking of 15th attribute is 9
Feature Ranking of 16th attribute is 12
Feature Ranking of 17th attribute is 11
```

This SVM limits to 3 best ranked features – “Weeks Since first Purchase”, “Week Since Last Purchased”, and “Non-book Purchases”

- 2) **The Second Feature Selection:** The Second Feature Selection is performed by *Greedy Forward Selection Method* on the features selected by the First Feature Selection. The Greedy Forward Selection selects 5 best subset features from the 10 features selected by the SVM-RFE.

The output is displayed here:

```
Java - Feature Selection/src/FeatureSubsetSelection.java - Eclipse
File Edit Source Navigate Search Project Refactor Run Window Help

Package Explorer
src
  (default package)
    FeatureRanking.java
  JRE System Library [JavaSE-1.7]
  Referenced Libraries
    Ensemble_Ranking_Output_Dataset
    Ensemble_Ranking_output.txt
    iris.data
    P1-CustomerPurchaseData.csv
    sparse.tsv
  Feature Scoring
  Feature Selection
    src
      (default package)
        FeatureSubsetSelection.java
        ajt-2.9.jar
        gmdh_shell.ini
        iris.data
        P1-CustomerPurchaseData.csv

FeatureScoring.java
FeatureSubsetSelection.java
FeatureRanking.java

BufferedWriter out = new BufferedWriter(fstream);

/*
 * Construct a greedy forward subset selector that will use the Pearson
 * correlation to determine the relation between each attribute and the
 * class label. The first parameter indicates that only one, i.e. 'the
 * best' attribute will be selected.
 */

//System.out.println(data);
GreedyForwardSelection gff = new GreedyForwardSelection(5, new PearsonCorrelationCoeffi

/* Apply the algorithm to the data set */
gff.build(data);

/* Print out the attribute that has been selected */
System.out.println("The Selected Attributes are "+gff.selectedAttributes());
out.write("The Selected Attributes are "+gff.selectedAttributes());
out.close();

FileHandler.exportDataset(data,new File("Subset_Selection_output_dataset.txt"));

}

catch (Exception e)
{
    System.err.println("Error: " + e.getMessage());
}

}

Console
<terminated> FeatureSubsetSelection [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Oct 29, 2012 3:29:30 PM)
The Selected Attributes are [4, 5, 6, 7, 8]
```

According to this Method, the best 3 features are
“Week since First Purchased”, “Books Purchased” and “Total Purchase”

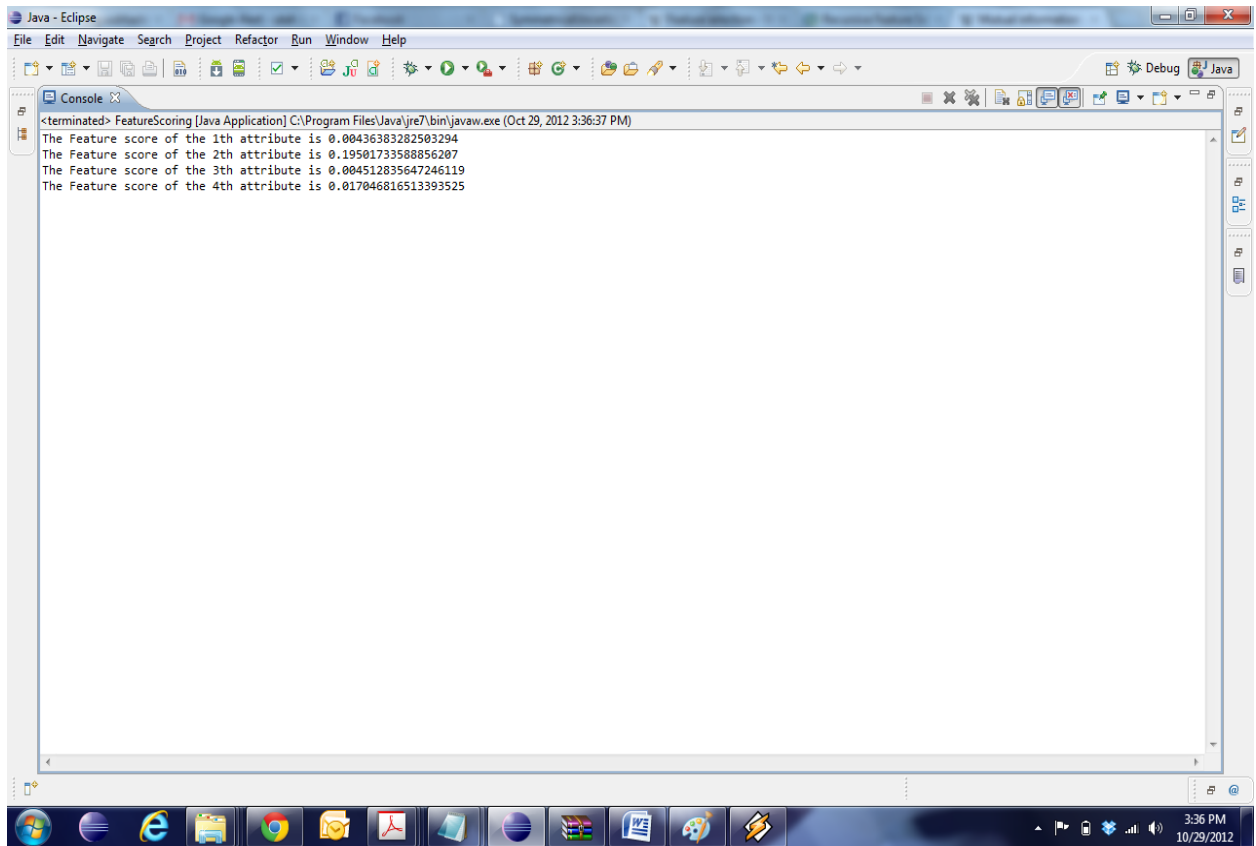
- 3) **The Third Feature selection:** The Third Feature Selection is performed using Symmetric Uncertainty Algorithm. *Symmetric uncertainty* is determined by the given formula:

$$U(X, Y) = 2R = 2 \frac{I(X; Y)}{H(X) + H(Y)}$$

which represents a weighted average of the two uncertainty coefficients. Here,

$I(X; Y)$ represents the mutual information of two discrete random variables X and Y

Symmetric Uncertainty scores the features based on the algorithm, and finally the best feature is selected. Higher the score, the better is the attribute.



```
<terminated> FeatureScoring [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Oct 29, 2012 3:36:37 PM)
The Feature score of the 1th attribute is 0.00436383282503294
The Feature score of the 2th attribute is 0.19501733588856207
The Feature score of the 3th attribute is 0.004512835647246119
The Feature score of the 4th attribute is 0.017046816513393525
```

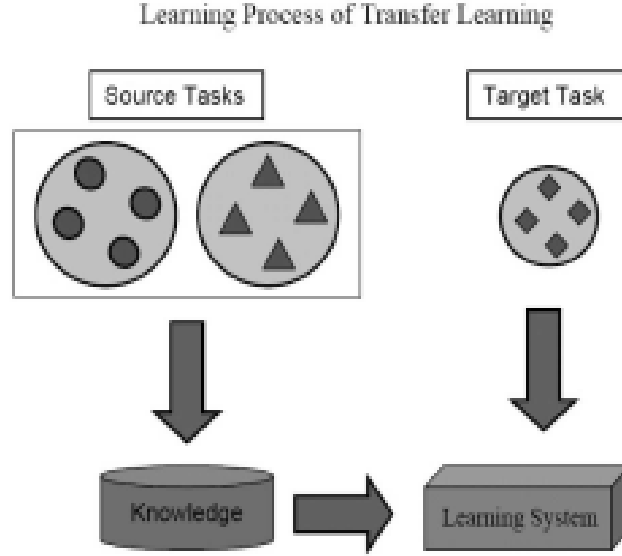
This algorithm limits its best 3 features as “Week Since First Purchased”, “Non-book Purchase” and “Total Purchase”

Analysis: In accordance with the feature selected by transfer learning using all the three mechanisms, the best feature is “Weeks since First Purchased” followed by the “Books Purchased” Feature.

Run Times: The SVM-RFE runs in linear time $O(n)$, whereas the Greedy Forward Selection and Symmetric Uncertainty runs in Exponential time $O(n \log n)$.

Phase 2: Application of a TransferBoost Model

1) Architecture of TransferBoost Transfer Learning



Transfer learning deals with the ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks. It aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task. In contrast to multi-task learning, transfer learning cares most about the target task. The roles of the source and target tasks are not always symmetric in transfer learning. Traditional machine learning techniques try to learn each task from scratch, while transfer learning techniques try to transfer the knowledge from some previous tasks to a target task when the latter has fewer high-quality training data.

2) Base Algorithm:

Algorithm 1 TransferBoost ¹

Input: source tasks S_1, \dots, S_k , where $S_t = \{(x_j, y_j)\}_{j=1}^{|S_t|}$,
target training examples $T = \{(x_t, y_t)\}_{t=1}^n$.

- 1: Merge the source and target training data
 $D = S_1 \cup \dots \cup S_k \cup T$.
- 2: Initialize $w_1(x_t) = 1/|D|$, for $(x_t, y_t) \in D$. (Optionally, these initial weights could be specified by the user.)
- 3: **for** $t = 1, \dots, K$ **do**
- 4: Train model $h_t : X \rightarrow Y$ on D with weights $w_t(D)$.
- 5: Choose $\alpha_t^i \in \mathbb{R}$ for $i = 1, \dots, k$.
- 6: Choose $\beta_t \in \mathbb{R}$.
- 7: Update the weights for all $(x_j, y_j) \in D$:

$$w_{t+1}(x_j) = \begin{cases} \frac{w_t(x_j) \exp(-\beta_t y_j h_t(x_j) + \alpha_t^1)}{Z_t} & (x_j, y_j) \in S_1 \\ \frac{w_t(x_j) \exp(-\beta_t y_j h_t(x_j))}{Z_t} & (x_j, y_j) \in T \end{cases}$$

where Z_t normalizes $w_{t+1}(D)$ to be a distribution.

8: **end for**

Output: the hypothesis $H(x) = \text{sign} \left(\sum_{t=1}^K \beta_t h_t(x) \right)$

TransferBoost builds on AdaBoost to transfer source instances when learning a target distribution. The algorithm boosts each source task based on a notion of transferability from the source task to the target task. Transferability is the change in performance on the target task between learning with and without transfer. Transferability can be positive or negative, depending on whether transfer increases or decreases performance on the target task. TransferBoost boosts each set of instances from the same task, increasing the weights of all instances from a source task if it shows positive transferability to the target task. Similarly, it decreases the weights of all instances from source tasks that show negative transferability to the target. After several iterations, instances from source tasks that show positive transfer will have higher weights, and source tasks that show negative transfer will have lower weights. Additionally, the weight distribution within each source task and the target task will emphasize instances that are repeatedly mislabeled. In this manner, TransferBoost determines the instances that are likely from the target distribution. Modification in the algorithm is done in step 4, the model is trained on the source and target dataset merged together with weights by sending them in a set of 6 SVMs as a base classifier.

3) Dataset Used:

- 1) [Multi-domain Sentiment Dataset](#): In the dataset webpage, the [processed_stars.tar.gz](#) file is downloaded, and then only the “train” file is used as the source and target data for the different sets. After the data is downloaded, it is converted into a Weka Compatible arff file, and reduced to 10% to 20% of the attributes for training the classifier.

Note:

- i) The dimensions of the datasets have to be same.
- ii) The instances have to have same dimensions
- iii) No attributes can repeat exactly in two datasets, in that case, one of them should be manipulated manually using special characters or whitespaces.

- 2) [20 Newsgroup dataset](#): In this set, only the [20news-18828.tar.gz](#) is downloaded, and then only the “train” file is used as the source and target data for the different sets. After the data is downloaded, it is converted into a Weka Compatible arff file, and reduced to 10% to 20% of the attributes for training the classifier.

Note:

- i) The dimensions of the source and target datasets have to be same.
- ii) The instances have to have same dimensions
- iii) No attributes can repeat exactly in two datasets, in that case, one of them should be manipulated manually using special characters or whitespaces.

4) Mechanism of the data being fed into the classifier:

- i) The dataset consists of a category like books, DVD etc., which consists of positive and negative reviews as Sentiments. The training sets are divided into abstract number of sets and a randomizer is used to shuffle the reviews in each set. For the first source task, 20% of the train set is provided. For the second source task, 25% of the remaining dataset is provided, and the process continues till we keep feeding the source task in the Knowledge Matrix.
- ii) If a particular attribute is repeated in both the datasets, then one of them are manipulated manually using special characters or whitespaces. E.g: if the text “all_the” appears in both the Books and DVD dataset, then in one of them, “all_the” should be changed to “all \the” or all-\the” or something of such type. Enhancements about this issue are mentioned in the enhancement section.

5) Number of SVMs:

The number of SVMs used in the Adaboost classifier is abstract, though in the original Adaboost Weka library, the source task is classified through a set of 6 SVMs. The response in the final accuracy differs with varied sets of SVMs.

6) Accuracy Analysis:

- i) Fix the number of SVMs to 6 (default)
- ii) Fix the number of iterations to 10 (default)

1) Sentiment dataset

Accuracy Comparisons:

Source vs Target Domain	Stochastic Gradient Descent (%)	TrAdaboost (%)
DVD vs Book	74.4	80.0
Electronics vs Book	69.5	74.9
Kitchen vs Book	67.6	68.1
Book vs DVD	78.2	74.4
Electronics vs DVD	71.9	73.6
Kitchen vs DVD	75.3	75.3
Book vs Electronics	68.3	76.7
DVD vs Electronics	73.9	73.1
Kitchen vs Electronics	80.0	85.9
Book vs Kitchen	71.2	76.4
DVD vs Kitchen	75.3	87.9
Electronics vs Kitchen	86.3	80.5

2) 20 NewsGroups dataset:

Accuracy Comparisons:

Source vs Target Domain	SVM (%)	TrAdaboost (%)
Rec vs talk	87.3	91.2
rec vs sci	83.3	93.5
sci vs talk	80.2	90.3
comp vs talk	90.3	95.7
comp vs sci	71.9	88.8

- 7) **Console Output:** An output of one sample implementation is attached here, where the accuracy of DVD dataset is checked with the Books dataset. (DVD vs Book)

Deep, Advisor: Dr. Y.S. Kim (2013). Transfer based Learning. Logan, UT.

Processing the Classifier...

Getting the Source Data...

Results of the Boosted Classifier in Iteration# 1 basing on Source and Target Data
=====

Correctly Classified Instances	0.4999	99.98 %
Incorrectly Classified Instances	0.0001	0.02 %
Kappa statistic	0	
Mean absolute error	0.0004	
Root mean squared error	0.0158	
Relative absolute error	0.1001 %	
Root relative squared error	3.9551 %	
Total Number of Instances	0.5	

Results of the Boosted Classifier in Iteration# 2 basing on Source and Target Data
=====

Correctly Classified Instances	0.0117	98.5049 %
Incorrectly Classified Instances	0.0002	1.4951 %
Kappa statistic	0	
Mean absolute error	0.0299	
Root mean squared error	0.1367	
Relative absolute error	6.0138 %	
Root relative squared error	27.4943 %	
Total Number of Instances	0.0119	

Results of the Boosted Classifier in Iteration# 3 basing on Source and Target Data
=====

Correctly Classified Instances	0.0008	82.6628 %
Incorrectly Classified Instances	0.0002	17.3372 %
Kappa statistic	0	
Mean absolute error	0.3467	
Root mean squared error	0.4655	
Relative absolute error	69.3625 %	
Root relative squared error	93.1238 %	
Total Number of Instances	0.0009	

Results of the Boosted Classifier in Iteration# 4 basing on Source and Target Data
=====

Correctly Classified Instances	0.0006	80.038 %
Incorrectly Classified Instances	0.0001	19.962 %
Kappa statistic	0	
Mean absolute error	0.3992	
Root mean squared error	0.4995	
Relative absolute error	79.8584 %	
Root relative squared error	99.9181 %	
Total Number of Instances	0.0007	

Results of the Boosted Classifier in Iteration# 5 basing on Source and Target Data
=====

Correctly Classified Instances	0.0007	80.0005 %
Incorrectly Classified Instances	0.0002	19.9995 %
Kappa statistic	0	
Mean absolute error	0.4	
Root mean squared error	0.5	
Relative absolute error	80.0105 %	
Root relative squared error	100.0143 %	
Total Number of Instances	0.0009	

Results of the Boosted Classifier in Iteration# 6 basing on Source and Target Data
=====

Correctly Classified Instances	0.0008	80 %
Incorrectly Classified Instances	0.0002	20 %
Kappa statistic	0	
Mean absolute error	0.4	
Root mean squared error	0.5	
Relative absolute error	80.015 %	
Root relative squared error	100.0188 %	
Total Number of Instances	0.001	

Results of the Boosted Classifier in Iteration# 7 basing on Source and Target Data
=====

Correctly Classified Instances	0.001	80	%
Incorrectly Classified Instances	0.0003	20	%
Kappa statistic	0		
Mean absolute error	0.4		
Root mean squared error	0.5		
Relative absolute error	80.0184	%	
Root relative squared error	100.0229	%	
Total Number of Instances	0.0013		

Results of the Boosted Classifier in Iteration# 8 basing on Source and Target Data
=====

Correctly Classified Instances	0.0012	80	%
Incorrectly Classified Instances	0.0003	20	%
Kappa statistic	0		
Mean absolute error	0.4		
Root mean squared error	0.5		
Relative absolute error	80.0224	%	
Root relative squared error	100.028	%	
Total Number of Instances	0.0016		

Results of the Boosted Classifier in Iteration# 9 basing on Source and Target Data
=====

Correctly Classified Instances	0.0015	80	%
Incorrectly Classified Instances	0.0004	20	%
Kappa statistic	0		
Mean absolute error	0.4		
Root mean squared error	0.5		
Relative absolute error	80.0274	%	
Root relative squared error	100.0342	%	
Total Number of Instances	0.0019		

Results of the Boosted Classifier in Iteration# 10 basing on Source and Target Data
=====

Correctly Classified Instances	0.0019	80	%
Incorrectly Classified Instances	0.0005	20	%
Kappa statistic	0		
Mean absolute error	0.4		
Root mean squared error	0.5		
Relative absolute error	80.0334	%	
Root relative squared error	100.0417	%	
Total Number of Instances	0.0023		

Results of the Boosted Classifier basing on Source and Target Data

=====

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.8	1	0.889	0	0
	0	0	0	0	0	0	1
Weighted Avg.	0.8	0.8	0.64	0.8	0.711	0	

TransferBoost completeNumber of performed Iterations: 10

The output signifies that the classifier has a final accuracy of 80% after its 10th iteration.

```

Java - Eclipse
File Edit Run Navigate Search Project Refactor Window Help

<terminated> testMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Apr 30, 2013 12:32:56 PM)

Results of the Boosted Classifier in Iteration# 8 basing on Source and Target Data
=====
Correctly Classified Instances      0.0012      80    %
Incorrectly Classified Instances    0.0003      20    %
Kappa statistic                    0
Mean absolute error                 0.4
Root mean squared error             0.5
Relative absolute error             80.0224 %
Root relative squared error         100.028 %
Total Number of Instances          0.0016

Results of the Boosted Classifier in Iteration# 9 basing on Source and Target Data
=====
Correctly Classified Instances      0.0015      80    %
Incorrectly Classified Instances    0.0004      20    %
Kappa statistic                    0
Mean absolute error                 0.4
Root mean squared error             0.5
Relative absolute error             80.0274 %
Root relative squared error         100.0342 %
Total Number of Instances          0.0019

Results of the Boosted Classifier in Iteration# 10 basing on Source and Target Data
=====
Correctly Classified Instances      0.0019      80    %
Incorrectly Classified Instances    0.0005      20    %
Kappa statistic                    0
Mean absolute error                 0.4
Root mean squared error             0.5
Relative absolute error             80.0334 %
Root relative squared error         100.0417 %
Total Number of Instances          0.0023

Results of the Boosted Classifier basing on Source and Target Data
=====
      TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
      1        1        0.8        1      0.889      0        0
      0        0        0        0      0        0        1
Weighted Avg.  0.8      0.8      0.64     0.8      0.711     0

TransferBoost completeNumber of performed Iterations: 10

```

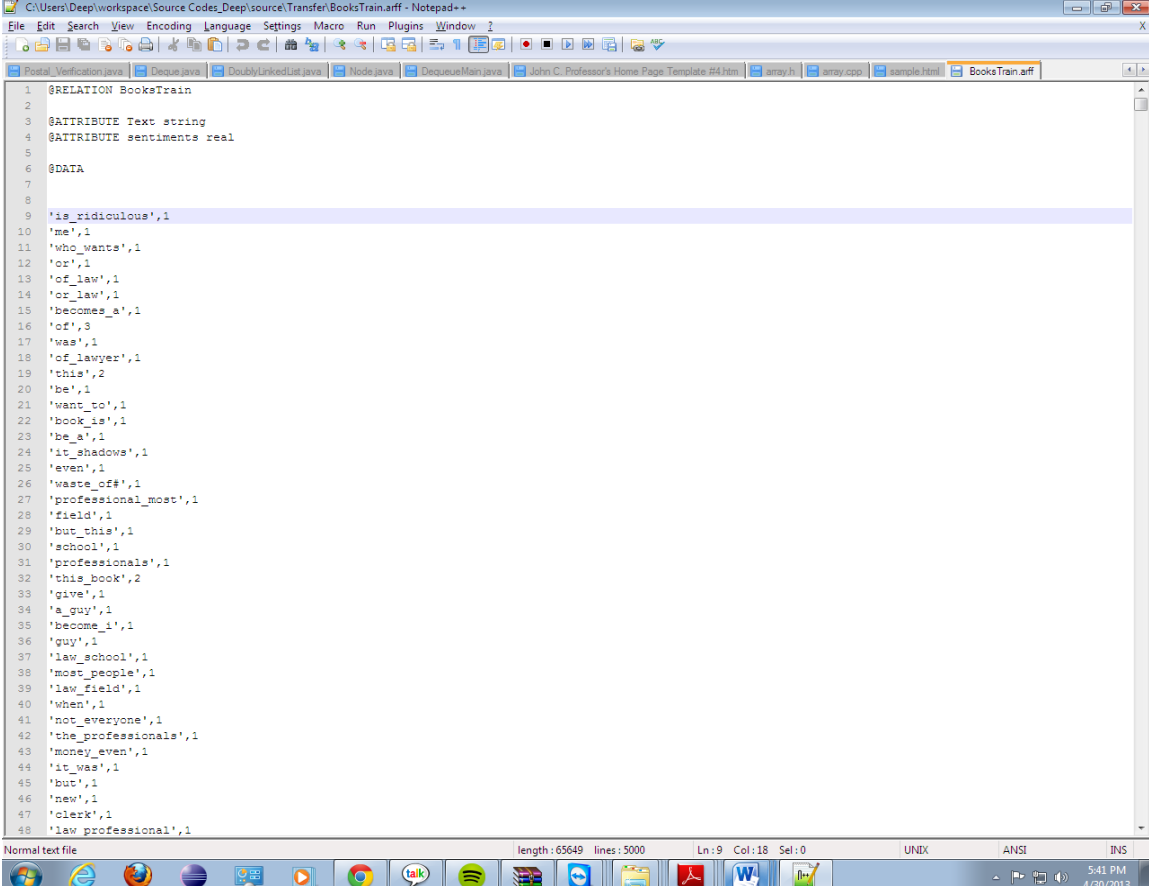
Technical Specifications:

- [Eclipse IDE](#)
- [JDK Standard Edition](#)
- [Java Virtual Machine](#)
- [Notepad++](#)
- Weka.jar (provided in library folder)
- Utils.jar (provided in library folder)
- Java Machine Learning Toolkit (javaml -- provided in library folder)
- LibSVM.jar file (provided in library folder)
- AJT Maven Library (provided in library folder)

➔ Converting a JAVA File into a WEKA compatible ARFF format:

- i) Download the train file in .txt format, and open the same with Notepad++
- ii) Put a relation name at the top of the text file as @RELATION <name>
- iii) Put the column names immediately after that as @ATTRIBUTE <name> <type>
- iv) Put the Data after the attribute field as @DATA
- v) Save the file as .arff

A typical .arff file looks like this:



```
1 @RELATION BooksTrain
2
3 @ATTRIBUTE Text string
4 @ATTRIBUTE sentiments real
5
6 @DATA
7
8
9 'is_ridiculous',1
10 'me',1
11 'who_wants',1
12 'or',1
13 'of_law',1
14 'or_law',1
15 'becomes_a',1
16 'of',3
17 'was',1
18 'of_lawyer',1
19 'this',2
20 'be',1
21 'want_to',1
22 'book_is',1
23 'be_a',1
24 'it_shadows',1
25 'even',1
26 'waste_of$',1
27 'professional_most',1
28 'field',1
29 'but_this',1
30 'school',1
31 'professionals',1
32 'this_book',2
33 'give',1
34 'a_guy',1
35 'become_i',1
36 'guy',1
37 'law_school',1
38 'most_people',1
39 'law_field',1
40 'when',1
41 'not_everyone',1
42 'the_professionals',1
43 'money_even',1
44 'it_was',1
45 'but',1
46 'new',1
47 'clerk',1
48 'law_professional',1
```

The screenshot shows a Notepad++ window with the file 'BooksTrain.arff' open. The file content is an ARFF format file. The first line is '@RELATION BooksTrain'. The next two lines are '@ATTRIBUTE Text string' and '@ATTRIBUTE sentiments real'. The third line is '@DATA'. The data section contains 48 lines of text and sentiment pairs, such as 'is_ridiculous',1, 'me',1, 'who_wants',1, 'or',1, 'of_law',1, 'or_law',1, 'becomes_a',1, 'of',3, 'was',1, 'of_lawyer',1, 'this',2, 'be',1, 'want_to',1, 'book_is',1, 'be_a',1, 'it_shadows',1, 'even',1, 'waste_of\$',1, 'professional_most',1, 'field',1, 'but_this',1, 'school',1, 'professionals',1, 'this_book',2, 'give',1, 'a_guy',1, 'become_i',1, 'guy',1, 'law_school',1, 'most_people',1, 'law_field',1, 'when',1, 'not_everyone',1, 'the_professionals',1, 'money_even',1, 'it_was',1, 'but',1, 'new',1, 'clerk',1, and 'law_professional',1. The status bar at the bottom shows 'Normal text file', 'length: 65649 lines: 5000', 'Ln: 9 Col: 18 Sel: 0', 'UNIX', 'ANSI', 'INS', and the date '4/30/2013'.

Note: Extract the Libraries folder from the zip file into the Eclipse workspace.

➔ How to Run the algorithms:

Transfer Learning

- 1) Extract the “Transfer” Folder from the “source” directory in the attached zip-files into the eclipse workspace.
- 2) Open Eclipse
- 3) Go to “file” -> Import -> Existing Projects into Workspace -> Select Root directory-> Browse
- 4) Locate the “Transfer” folder from the Workspace and press “Open” or Ok”
- 5) Check “Copy Projects into Workspace”
- 6) Click Finish
- 7) Right Click on the “Transfer” Folder in the “Package Explorer” to the left, and click “Properties”
- 8) As the window pops up, click “Java Build Path”, and then click “Libraries”.
- 9) Remove any files if they are marked by a red cross.
- 10) Click “Add External Jars”
- 11) Locate the Libraries Folder, and select “Weka.jar” and “Utils.jar” and click open
- 12) Click OK
- 13) In the “Package Explorer” to the left, double-click TestMain.java and run the program by pressing the run button, and running it as a JAVA application

Feature Ranking

- 1) Extract the “Feature Ranking” Folder from the “source” directory in the attached zip-files into the eclipse workspace.
- 2) Open Eclipse
- 3) Go to “file” -> Import -> Existing Projects into Workspace -> Select Root directory-> Browse
- 4) Locate the “Feature Ranking” folder from the Workspace and press “Open” or Ok”
- 5) Check “Copy Projects into Workspace”
- 6) Click Finish
- 7) Right Click on the “Feature Ranking” Folder in the “Package Explorer” to the left, and click “Properties”
- 8) As the window pops up, click “Java Build Path”, and then click “Libraries”.
- 9) Remove any files if they are marked by a red cross.
- 10) Click “Add External Jars”
- 11) Locate the Libraries Folder, and select “libsvm.jar”, “ajt-2.9.jar” and “javaml-0.1.7.jar” and click open
- 12) Click OK
- 13) In the “Package Explorer” to the left, double-click FeatureRankingSVMRFE.java and run the program by pressing the run button, and running it as a JAVA application

Feature Selection

- 1) Extract the “Feature Selection” Folder from the “source” directory in the attached zip-files into the eclipse workspace.
- 2) Open Eclipse
- 3) Go to “file” -> Import -> Existing Projects into Workspace -> Select Root directory-> Browse
- 4) Locate the “Feature Selection” folder from the Workspace and press “Open” or Ok”
- 5) Check “Copy Projects into Workspace”
- 6) Click Finish
- 7) Right Click on the “Feature Selection” Folder in the “Package Explorer” to the left, and click “Properties”
- 8) As the window pops up, click “Java Build Path”, and then click “Libraries”.
- 9) Remove any files if they are marked by a red cross.
- 10) Click “Add External Jars”
- 11) Locate the Libraries Folder, and select “libsvm.jar”, “ajt-2.9.jar” and “javaml-0.1.7.jar” and click open
- 12) Click OK
- 13) In the “Package Explorer” to the left, double-click FeatureSelectionGreedy.java and run the program by pressing the run button, and running it as a JAVA application

Feature Scoring

- 1) Extract the “Feature Scoring” Folder from the “source” directory in the attached zip-files into the eclipse workspace.
- 2) Open Eclipse
- 3) Go to “file” -> Import -> Existing Projects into Workspace -> Select Root directory-> Browse
- 4) Locate the “Feature Scoring” folder from the Workspace and press “Open” or Ok”
- 5) Check “Copy Projects into Workspace”
- 6) Click Finish
- 7) Right Click on the “Feature Scoring” Folder in the “Package Explorer” to the left, and click “Properties”
- 8) As the window pops up, click “Java Build Path”, and then click “Libraries”.
- 9) Remove any files if they are marked by a red cross.
- 10) Click “Add External Jars”
- 11) Locate the Libraries Folder, and select “libsvm.jar”, “ajt-2.9.jar” and “javaml-0.1.7.jar” and click open
- 12) Click OK
- 13) In the “Package Explorer” to the left, double-click SymmetricalUncertaintyScoring.java and run the program by pressing the run button, and running it as a JAVA application

Future Enhancements:

- 1) Instead of manually editing the common attribute in the dataset, the same can be done using a database.
- 2) The Feature Selection algorithms can be used to limit the attributes while training the model. By applying Feature Selection on the entire dataset, 25% of the dataset having the best features can be considered to train the target and source dataset.

References:

- 1) http://en.wikipedia.org/wiki/Feature_selection
- 2) <http://www.hindawi.com/journals/cmmm/2012/712542/>
- 3) http://en.wikipedia.org/wiki/Mutual_information
- 4) <http://cs.brynmawr.edu/~eeaton/papers/Eaton2011Selective.pdf>
- 5) <http://dl.acm.org/citation.cfm?id=1850545>
- 6) <http://www.computer.org/csdl/proceedings/ictai/2011/4596/00/4596a399-abs.html>
- 7) <http://weka.sourceforge.net/doc/weka/classifiers/meta/AdaBoostM1.html>