EECS 738 Final Project Report (Incomplete Draft before Presentation)

Su Yan
Wei Fei
Jiahui Wang

I. Introduction:

This project studies on the problem of the high dropout rate on MOOC learning platforms. To predict whether a user will drop a course, we conducted our preliminary experiments on the provided training and testing data set in order to find the suitable classifiers to use. In phase one of project, we show the exploratory analysis of the classification accuracy of many classifiers. The accuracy of our prediction results in this phase was poor. In phase two of project, we select three models with good performance from the candidate models in phase one. Then we optimize the parameters roughly based on all features of provided training data and testing data, the accuracy and AUC are not optimal for the affect of meaningless features. Hence, we do the feature normalization to reduce the meaningless features and then do the parameter optimization.

II. Problem statement:

In the given training data and testing data, we pad the first row with indices of attributes to make sure Weka can recognize the data. We define the second column as the class attribute which represents whether or not the student will dropout from the course. Also, we replaced 0s and 1s with 'No' and 'Yes' correspondingly to make it a nominal class. The rest 50 columns are the 50 features. The first problem we have is to find three optimal classifiers to use. Second, we do the feature selections to reduce the meaningless features, and then optimize parameters based on the performance of models

III. Exploratory analysis of the data set:

- Training data:
    - Features: 50
    - Samples: 72326
    - Type of features: integer number, floating point number
    - Label: Yes or No

- Testing data:
    - Features: 50
    - Samples: 48220
    - Type of features: integer number
    - Label: Yes or No

In terms of preprocessing, we transformed the class attributes from 0 and 1's to 'No' and 'Yes' correspondingly to let weka recognized as nominal class. We also performed normalization over the 50 features to avoid

IV. Experiment Design:

1) How do you build your models, how to optimize your models, how do you perform feature selection, and how do you evaluate your model?

1. Model selection

We select our models based on two principles. In the previous stage, we obtained the list of prediction accuracies of most weka's built-in classifiers using their default parameters and 10-fold cross validation (Indicated in following table). We select models with top accuracy and built-in regularizations from the list. They are Logistic, RandomForest and RBFNetwork.

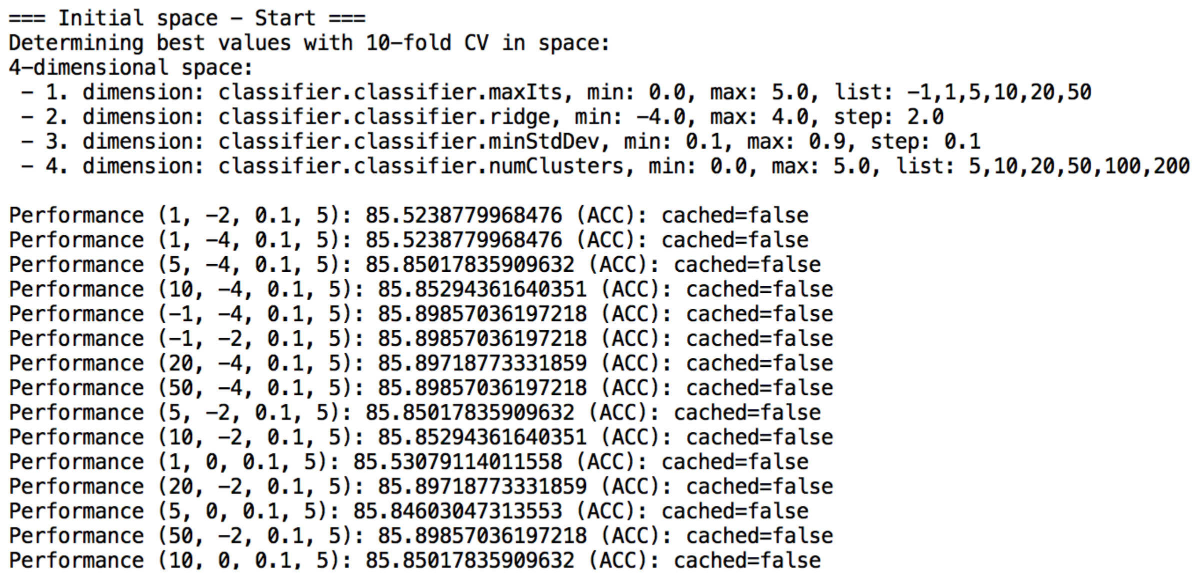
Classifier Name	Accuracy (with default parameters)
RandomForest	87.91%
LMT (Logistic Model Trees)	87.90%
Logistic	87.65%
JRip	87.63%
MultilayerPerceptron	87.26%
PART	87.25%
REPTree	87.11%
SMO	87.08%
DecisionTable	86.95%
HoeffdingTree	86.95%
SGD	86.87%
OneR	85.90%
J48	85.29%
NaiveBayes	84.89%
LWL	84.52%
DecisionStump	84.48%
BayesNet	81.41%
VotedPerceptron	81.36%
IBk	81.33%
RandomTree	81.09%

ZeroR	79.29%
NaiveBayesMultinomial	70.69%
KStar	(No result. Computing time too long.)

Table: Performance of weka classifiers with default parameter

2. Model parameter optimization (before feature selection)

From preliminary results, we could observe that models with only default parameters are not reliable and good enough. We mainly use the *MultiSearch* (https://github.com/fracpete/multisearch-weka-package) library in this stage. The library allows the optimization of any arbitrary number of parameters and supports both ranged parameter values and array of values. *Multisearch* provides exhaustive search over all given parameter combinations and pick one with best performance.



```
=== Initial space - Start ===
Determining best values with 10-fold CV in space:
4-dimensional space:
 - 1. dimension: classifier.classifier.maxIts, min: 0.0, max: 5.0, list: -1,1,5,10,20,50
 - 2. dimension: classifier.classifier.ridge, min: -4.0, max: 4.0, step: 2.0
 - 3. dimension: classifier.classifier.minStdDev, min: 0.1, max: 0.9, step: 0.1
 - 4. dimension: classifier.classifier.numClusters, min: 0.0, max: 5.0, list: 5,10,20,50,100,200

Performance (1, -2, 0.1, 5): 85.5238779968476 (ACC): cached=false
Performance (1, -4, 0.1, 5): 85.5238779968476 (ACC): cached=false
Performance (5, -4, 0.1, 5): 85.85017835909632 (ACC): cached=false
Performance (10, -4, 0.1, 5): 85.85294361640351 (ACC): cached=false
Performance (-1, -4, 0.1, 5): 85.89857036197218 (ACC): cached=false
Performance (-1, -2, 0.1, 5): 85.89857036197218 (ACC): cached=false
Performance (20, -4, 0.1, 5): 85.89718773331859 (ACC): cached=false
Performance (50, -4, 0.1, 5): 85.89857036197218 (ACC): cached=false
Performance (5, -2, 0.1, 5): 85.85017835909632 (ACC): cached=false
Performance (10, -2, 0.1, 5): 85.85294361640351 (ACC): cached=false
Performance (1, 0, 0.1, 5): 85.53079114011558 (ACC): cached=false
Performance (20, -2, 0.1, 5): 85.89718773331859 (ACC): cached=false
Performance (5, 0, 0.1, 5): 85.84603047313553 (ACC): cached=false
Performance (50, -2, 0.1, 5): 85.89857036197218 (ACC): cached=false
Performance (10, 0, 0.1, 5): 85.85017835909632 (ACC): cached=false
```

Figure: Example log of optimization by Multisearch

In general, we use the prediction accuracy as the standard to determine the performance of a model and 10-fold. The result of parameter optimization will be shown in part **V. Experimental Study Result.**

3. Feature selection with another round of parameter optimization

We perform feature selection by wrapping the target classifier (e.g. Logistic) by two meta-classifiers *MultiSearch and AttributeSelectedClassifier*. The *MultiSearch* at the outside layer is still in charge of parameter optimization through 10-fold cross validation. *AttributeSelectedClassifier* at the middle layer perform feature selection with given algorithm and passes the selected feature to target classifier (e.g. Logistic) for performance evaluation.

This layered structure has great performance overhead since we need to perform feature selection each time when evaluating each parameter combination. But this sturcture can prevent possible information leaking to classifiers since we always perform feature selection on the training part of the data after the split in cross validation.

4. Iterative feature selection with wrapper method

In previous step, we select important features by evaluating the relations between features and relations between features and class attribute. In other words, we select features using the property of the training dataset. In wrapper method, we have different philosophy. Optimal features should also be different for each specific learning algorithm and training set and should not only corresponds to their correlations. "The optimal features depend on the specific biases and heuristics of the learning algorithm, and hence the wrapper approach naturally fits with this definition." (Ron Kohavi, George H. John (1997). Wrappers for feature subset selection. Artificial Intelligence. 97(1-2):273-324)

With the implementation of package *weka.attributeSelection.WrapperSubsetEval* we come up with the following iterative method:

- Give WrapperSubsetEval a classifier and its current optimal parameters
- Get a set of attributes that maximizes the prediction performance of give classifer
- Perform parameter optimization on resulting attribute set
- Give WrapperSubsetEval again the same classifier and newly optimized parameters
- Do this until convergence…

This method gives us an opportunity to get model-specific optimal feature set. However, the difficulty is the long computation time. Using the three-layer structure described in step 4, one iteration of feature selection and parameter optimization takes about 7 days and 16 hours to finish. This exceeds the time limit of a single job on KU ITTC Clusters, which is 168 hours. Unitl now we haven't get any result from the wrapper method.

5. Model Evaluation

From the results generated by program, we mainly use AUC (Area under curve) and accuracy to determine which model has better performance after cross validation. The evaluation result will be shown in next section.

V. Experimental study results

1. Results of model parameter optimization (without feature selection)

After we selected three models and performed parameter optimizations without feature selection. We obtained the following results:

Models	Accuracy(%)	ROC Area	MCC	Parameters

Logistic	87.6296	0.881	0.592	ridge = 1.0E-4
				maxIts = -1

RandomForest	87.9241	0.883	0.604	numTrees = 200
				maxdepth = 0
				numFeatures = 14

RBFNetwork	86.3382	0.858	0.545	maxIts = -1
				ridge = 1.0E2
				minStdDev = 0.7
				numClusters = 200

From the above table, the performance of three classifiers are similar. After parameter optimization, the result become more reliable. For example, the accuracy of Logistic with default ridge parameters (1.0E-8) is 87.65%. The accuracy after parameter optimization is 87.63% with a ridge of 1.0E-4. Even though the accuracy after parameter optimization is a little lower than the classifier used default parameter, the model becomes more reliable since there's less chance of over-fitting. For the Random Forest classifier, the accuracy improves a little (Before: 87.91%, After: 87.924%).

2. Results of feature selection with another round of parameter optimization

In this section we performed feature selection for Logistic and RBFNetwork. We do not select features for RandomeForest for now since it will randomly select a number of features to build each decision tree by itself. Following eight attribute evaluation algorithms are used in this step.

- Correlation-based Subset Selection (weka.attributeSelection.CfsSubsetEval)
- Pearson's Correlation Coefficient (weka.attributeSelection.CorrelationAttributeEval)
- Gain Ratio (weka.attributeSelection.GainRatioAttributeEval)
- Information Gain (weka.attributeSelection.InfoGainAttributeEval)
- OneR Classifier (weka.attributeSelection.OneRAttributeEval)
- Principal Components Analysis (weka.attributeSelection.PrincipalComponents)
- RELIEF (weka.attributeSelection.ReliefFAttributeEval)
- Symmetrical Uncertainty Analysis (weka.attributeSelection.SymmetricalUncertAttributeEval)

The results are listed in the following table:

Logistic:

	Accuracy(%)	Attribute Selected	Optimized Parameter
CfsSubsetEval+BestFirst	87.3617	24 31 33 36 37 38 39 40 41 42 45 46 50	maxIts = 6, ridge = 1.0E-10
CfsSubsetEval+GreedyStepwise	87.2065	24 31 33 36 37 38 39 40 41 42 45 46 50	maxIts = 10, ridge = 1.0E1
CorrelationAttributeEval	87.6338	33,31,37,38,32,24,21,39,48,50,45,29,19,26	maxIts = -1, ridge = 1.0E-3
GainRatioAttributeEval	87.6337	45,33,38,31,37,43,42,46,32,24,26,36,34,35	maxIts = -1, ridge = 1.0E-3
InfoGainAttributeEval	87.4791	33,32,31,38,37,27,34,50,36,29,48,35,24,26	maxIts = -1, ridge = 1.0E-3
OneRAttributeEval	87.6338	38,33,32,37,31,34,27,36,35,39,45,50,24,41	maxIts = -1, ridge = 1.0E-3
PrincipalComponents	-------	-------	-------

ReliefFAttributeEval	87.5572	1,2,3,4,5,6,7,8,9, 10,11,12,13,14,15, 16,17,18	maxIts = -1, ridge = 1.0E-3
SymmetricalUncertAttributeEval	87.6338	33,38,45,31,37,32, 43,42,46,24,26,34, 36,35,50	maxIts = -1, ridge = 1.0E-3

RBFNetwork:

	Accuracy(%)	Attribute Selected	Optimized Parameter
CfsSubsetEval+BestFirst	-------	-------	-------
CfsSubsetEval+GreedyStepwise	-------	-------	-------
CorrelationAttributeEval	86.7544	33,31,37,38,32,24, 21,39,48,50,45,29, 19,26	maxIts = -1, ridge = 1.0E2, minStdDev = 0.5, numClusters = 200
GainRatioAttributeEval	86.7703	45,33,38,31,37,43, 42,46,32,24,26,36, 34,35	maxIts = -1, ridge = 1.0E2, minStdDev = 0.5, numClusters = 200
InfoGainAttributeEval	86.7580	33,32,31,38,37,27, 34,50,36,29,48,35, 24,26	maxIts = -1, ridge = 1.0E2, minStdDev = 0.5, numClusters = 200
OneRAttributeEval	-------	-------	-------
PrincipalComponents	-------	-------	-------
ReliefFAttributeEval	-------	-------	-------
SymmetricalUncertAttributeEval	86.6311	33,38,45,31,37,32, 43,42,46,24,26,34, 36,35,50	maxIts = -1, ridge = 1.0E2, minStdDev = 0.5, numClusters = 200

(Note: ----- means computation is still running)

3. Results of iterative feature selection with wrapper method

(We understand MCC and AUC are required for each model. Additional results analysis will be provided after we are done with the wrapper method of feature selection.)

VI. Results analysis and discussion

(To be finished. We haven't come up with final results yet)

VII. Final Model Construction

(To be finished. Final model has not been constructed yet)

VII. Predicted Result on Test Data

(To be finished)