EECS 738 – Machine Learning Final Project

Report

On

Dropout Prediction on a MOOC Learning Platform

Chinnaswamy, Ragaprabha

Parikh, Bijal

Thippabhotla, Sirisha

Contents

1. Problem Statement 2
2. Exploratory Analysis of the Data Set 2
   2.1 Transformation of dataset 3
   2.2 Sampling 3
3. Model Design 4
   3.1 System Design 4
   3.2 Selecting Classifiers 4
   3.3 Parameter Tuning 5
4. Feature Selection 7
   4.1 RandomForest 7
   4.2 ADTree 7
   4.3 Logistic Regression: 8
5. Model Evaluation with Tuned Parameters and Optimized Feature Sets 9
7. Experiment Results – T test 13
   7.1 T-Test on Random Forest and ADTree: 13
   7.2 T-Test on Random Forest and Logistics: 14
   7.3 T-Test on ADTree and Logistics: 14
8. Conclusion 16
9. References 16

1. Problem Statement

Of late, multiple learning courses have been made available to students online, which provide the flexibility of being taken at any location and at any time. These online learning techniques have gained a lot of popularity in recent times. Coursera, edX, Udacity are some such learning platforms that offer online courses. However, these courses experience high dropout rates. The aim of this project is to apply machine learning algorithms to predict likelihood of dropout for students from online courses.

2. Exploratory Analysis of the Data Set

- The training data set was obtained from the file EECS738_Train.csv
- The training data set has the following properties

Number of instances	72326
Number of attributes	52
Type of Attributes	Numeric
Missing information	0%
Positive/Negative label ratio	57349/14977

- Testing data set was obtained from the file EECS738_Test.csv
- Testing Data set has the following properties

Number of instances	48216
Number of attributes	51
Types of Attributes	Numeric
Missing information	0% (excluding label column)

Observations

1) The dataset is a slightly imbalanced dataset, as the number of cases with decision '1' is more than 3 times the number of '0'.We have experimented with sampling the dataset, the details of which are provided in section 2.2
2) All attributes are numeric attributes.
3) The number of samples in the training dataset (72326) is large compared to the number of features (52).

2.1 Transformation of dataset

1. We converted the decision label from a numerical attribute to a categorical attribute in the training dataset. This helps in applying a wider range of machine learning algorithms to our training data. The modified decision label contains values- True for 1 and False for 0.
2. We converted both the training and testing csv files to arff files, as the latter stores information regarding attribute type. This helps in getting predictions for the test dataset, as WEKA then reads the decision attribute as categorical as opposed to numeric.
3. The testing data set does not contain the decision label. Therefore we created a new column in the test file to represent the label information.
4. We split the training dataset in to 2 parts using WEKA *RemovePercentage filter*. We retained 70% of the training dataset for model creation and tested our model on the remaining 30% of the data. This helped in verifying the accuracy of the model against labeled data.

2.2 Sampling

As the training data given to us was we slightly imbalanced, during the pre-processing we wanted to sample to data in order to balance it. Two Sampling approaches were used. They were SMOTE and SpreadSubSample. Mentioned below is a brief description of the strategies applied:

1) SMOTE- SMOTE injects synthetic data in to a dataset, to balance the number of positive and negative labels. It is a type of over sampling of data.
2) SpreadSubSample- Spread Sub Sampling is a strategy used to create a subset from the original dataset, so that the number of instances for both the decision are equal. It is a type of under sampling strategy.

- For both the sampling techniques we applied the techniques on 70% Training dataset.
- The sampled data was then randomized using Random filter in Weka.
- We then built a model (using Random Forest as our classifier with 10 fold cross validation) and re-evaluated it on the remaining 30% training dataset.

Sampling Results

Strategy	Area under the curve	Accuracy Prediction
No Sampling	0.871	87.6717%
SMOTE	0.87	86.9297%
SpreadSubSample	0.873	82.2011%

Dataset used was 70% if original training data – Tested on remaining 30% data.

Observations

1. We observed that the Area under the curve and Accuracy reduced after sampling when compared to the original data set without sampling.
2. Also, Sampling may sometimes lead to the risk of overfitting. We therefore decided not to use sampled data for building our model.

3. Model Design

3.1 System Design

Designing a model for predictions with large dataset is a compute intensive task .The processing took longer in Eclipse. We also faced a lot of memory issues and heap space errors. In order to overcome these problems, we made use of the ITTC cluster. Jobs were submitted to clusters using pbs script. The pbs script was coded in such a way that it prints both the error and output log.

3.2 Selecting Classifiers

In our initial exploratory analysis, we considered the classifiers RandomForest, RepTree and Naïve Bayes. The AUC and the accuracy for Naïve Bayes was low compared to RandomForest, so we decided to do an empirical study to see if some other classifiers gave us a better AUC.

We decided to build models on a reduced size of our training dataset and use 5 folds cross validation to see if other classifiers give us a better AUC. Here is a result of our initial analysis

Classifier	**AUC**	**Accuracy**
BayesianLogisticRegression	0.662	82.46%
BayesNet	0.732	80.6987%
NaiveBayes	0.862	84.95%
RandomForest	***0.886***	***87.826%***
LogisticRegression	***0.884***	***87.79%***
SimpleLogisticRegression	0.872	87.68%
ADTree	***0.881***	***87.38%***
SimpleCART	0.845	87.582%
SMO	0.737	87.107%

After running those tests, we narrowed down on the following classifiers to build our model:

1. Random Forest
2. Logistic
3. ADTree

3.3 Parameter Tuning

3.3.1 RandomForest

We tuned the parameters for random forest in the following range

1. MaxDepth: 0 to 11 [Incrementing in steps of 1]
2. NumOfTrees: 90 to 1000 [Incrementing it in steps of 100]

After comparing the weighted MCC values obtained on training the model using 10 folds cross validation, we used the parameters which gave the highest MCC value.

Upon observation, we found that for all the above combinations, the highest weighted MCC was obtained when maxDepth was set to 0.

Since the model was built 100+ times, instead of reporting the comprehensive list of results obtained, below table only contains some of the highest weighted MCC values.

NumOfTrees(I)	**MaxDepth(K)**	**Weighted MCC**
800	0	0.61884
700	0	0.61944
700	11	0.61707
900	0	0.61871
90	10	0.61592
400	0	0.61946
500	***0***	***0.61963***
500	11	0.61667

3.3.2 ADTree

The classifier ADTree has the following parameter:

- numOfBoostingIterations: Sets the number of boosting iterations to perform.

The parameter was tuned using the below ranges:

- numOfBoostingIterations : 10 to 100 [Incrementing in steps of 10]

We observed the weighted MCC and weighted PRC values for each iterations and chose the numOfBoostingIterations with the highest values among them. The following table displays the observed values on evaluating the model using 10-fold cross validation:

NumOfBoostingIterations	Weighted PRC	Weighted ROC	Weighted MCC
10	0.90822	0.88235	0.60282
20	0.91483	0.88702	0.60741
30	0.91551	0.88786	0.60948
40	0.91566	0.88792	0.61028
50	0.91549	0.88759	0.60914
60	0.91513	0.88768	0.61132
70	0.91531	0.88768	0.61246
80	0.91486	0.88735	0.61278
90	0.91474	0.88721	0.61441
100	**0.91455**	**0.88692**	**0.61485**

3.3.3 Logistic Regression

The classifier Logistic has the following parameters:

- maxIts – Used to specify the number of iterations to perform. We have considered the default value of -1 needed for convergence.
- Ridge – Used for setting the ridge value in the log-likelihood function.

The parameters were tuned using the below ranges:

- Ridge - Base: 100 to 1000 [Incrementing in steps of 100]
  - Exponent: 0 to 11 [Incrementing in steps of 1]

We observed the weighted MCC and weighted PRC values for each iteration and chose the ridge with the highest values among them. The following table displays a set of highest observed values on evaluating the model using 10-fold cross validation:

Ridge	Weighted PRC	Weighted ROC	Weighted MCC
1.00E-05	0.91184829	0.883948499	0.60821
1.00E-04	0.911847501	0.883946307	0.60824
6.25E-06	0.911848419	0.883949202	0.60808
0.001	0.911847325	0.883946006	0.60824
1.25E-04	0.911847609	0.883946344	0.60824
3.70E-05	0.911848078	0.883946928	0.60824
1.56E-05	0.911848567	0.883948268	0.60821
8.00E-06	0.911848197	0.883948827	0.60808
2.92E-06	0.9118484	0.883950253	0.60820

1.95E-06	0.911848317	0.883950295	0.60818
0.01	0.911847556	0.883946246	0.60811
0.0025	0.911847328	0.883946004	0.60817
0.001111111	0.911847335	0.883946032	0.60824
6.25E-04	0.911847398	0.883946127	0.60824
4.00E-04	**0.911847322**	**0.88394613**	**0.60831**
2.78E-04	0.911847365	0.883946132	0.60831
2.04E-04	0.9118474	0.883946214	0.60831
1.56E-04	0.911847564	0.8839463	0.60831
1.23E-04	0.911847581	0.883946323	0.60824

4. Feature Selection

4.1 RandomForest

- CfsSubsetEval was used alongwith GreedyStepwise as a search method.

- We tried used Greedy Stepwise as the search method and tried optimizing the parameters of Greedy Stepwise as by changing numToSelect as -1 and then from 10 to 20 in steps of 1

Parameters	Weighted MCC	Weighted PRC
numToSelect = -1, searchDirection = forward threshold = default	0.6024	0.9014
numToSelect = 10 searchDirection = forward threshold = default	0.60405	0.9022

4.2 ADTree

Attribute Evaluator: CfsSubsetEval

- Classifier: ADTree [Parameter: numOfBoostingIterations-100]
- SearchMethod: BestFirst, GreedyStepwise

Evaluator	Search Method	Parameters (for search method)	Weighted MCC	Weighted PRC
CfsSubsetEval	BestFirst	Direction: Forward lookupCacheSize: 5 searchTermination: 1	**0.60928**	**0.90735**
CfsSubsetEval	GreedyStepWise	ConservativeForwardSelection: False Generate Ranking: False numToSelect: -1 SearchBackwards: False Threshold: -1.7976931348623157E308 *[Default]*	0.60842	0.90748

4.3 Logistic Regression:

Attribute Evaluator: ClassifierSubsetEval

- Classifier: Logistic Regression [Parameters: Ridge-4.0E-4, maxIts -1]
- SearchMethod: GreedyStepwise

Evaluator	Search Method	Parameters (for search method)	Weighted MCC	Weighted PRC
ClassifierSubsetEval-Logistic	GreedyStepWise	ConservativeForwardSelection: False Generate Ranking: False numToSelect: -1 SearchBackwards: False Threshold: -1.7976931348623157E308 *[Default]*	**0.60863**	**0.91030**
ClassifierSubsetEval-Logistic	GreedyStepWise	ConservativeForwardSelection: True Generate Ranking: False numToSelect: 10 SearchBackwards: False Threshold: -1.7976931348623157E308	0.60683	0.90830

5. Model Evaluation with Tuned Parameters and Optimized Feature Sets

- After performing Feature Selection, the original dataset was reduced based on the number of features selected.
- The model was then built using this reduced dataset for all the three Classifiers separately.
- Presented in the below table are the Weighted MCC values for all the three classifiers before and after applying Feature Selection techniques.

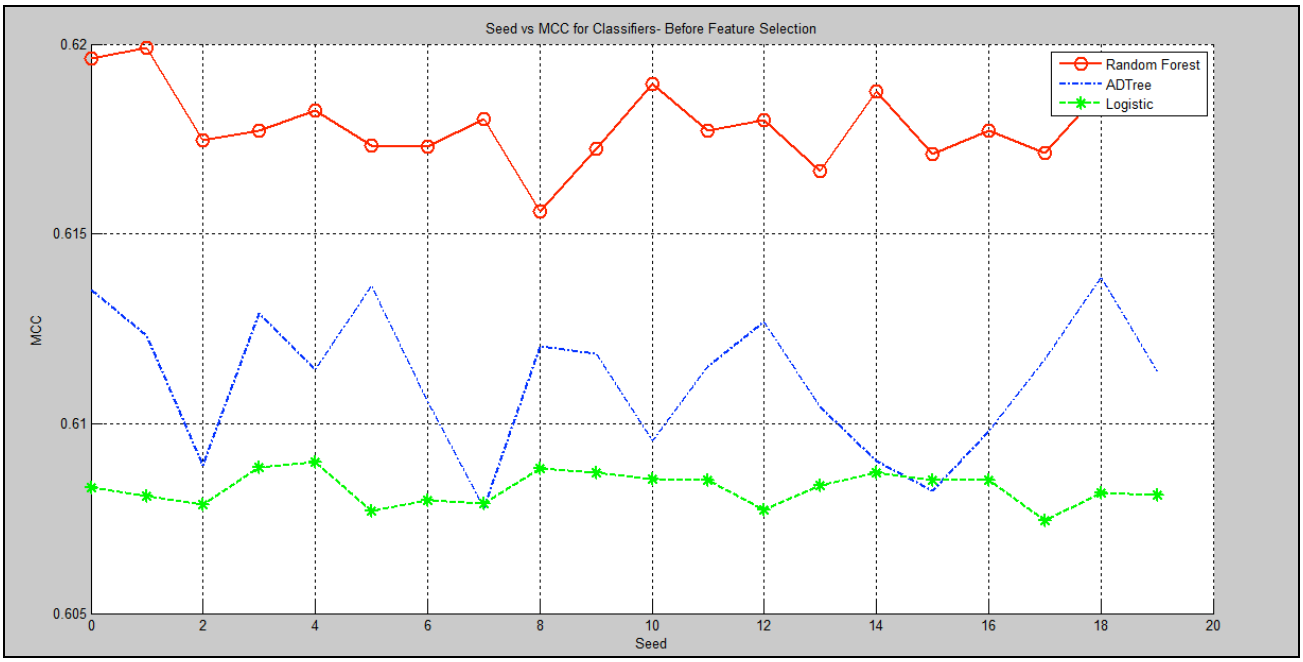
Weighted MCC:

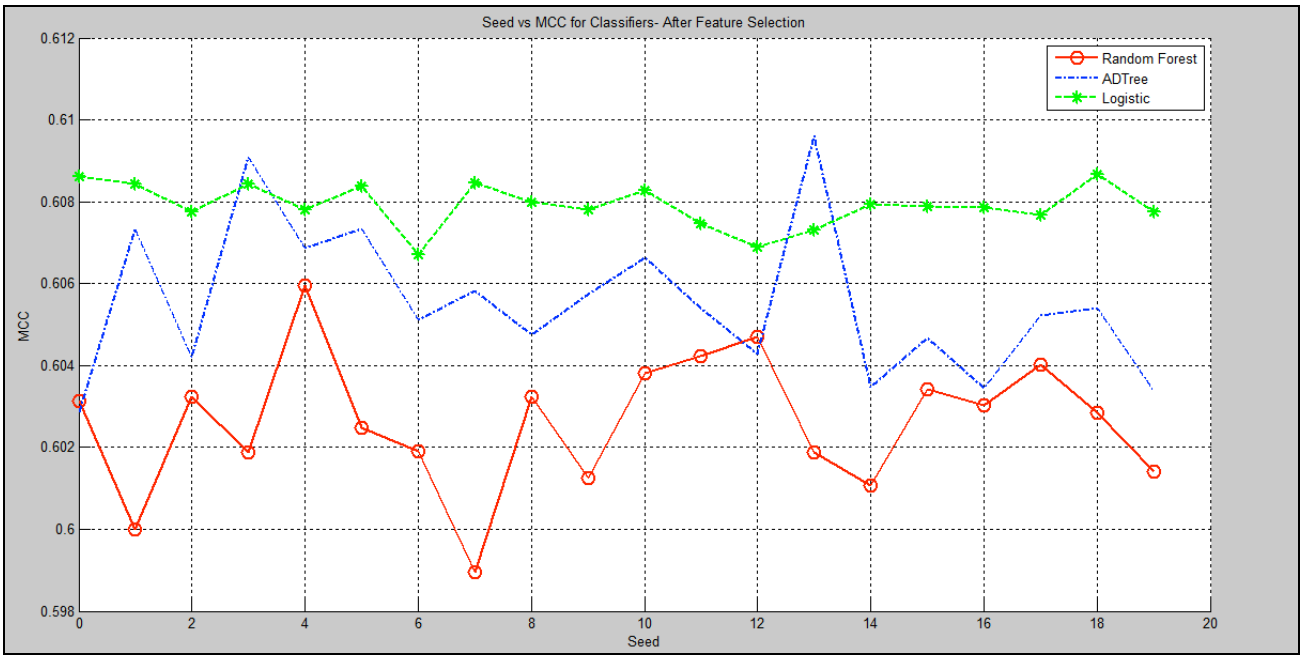
	Before Feature Selection [Original Dataset]			**After Feature Selection[Reduced Dataset]**		
Seed	**Random Forest**	**ADTree**	**Logistic**	**Random Forest**	**ADTree**	**Logistic**
0	0.61963	0.61352	0.60831	0.60312	0.60286	0.60863
1	0.61991	0.61232	0.60808	0.59999	0.60735	0.60844
2	0. 61748	0.60891	0.60787	0.60323	0.60420	0.60775
3	0. 61773	0.61291	0.60884	0.60187	0.60910	0.60845
4	0. 61825	0.61142	0.60899	0.60596	0.60688	0.60781
5	0. 61733	0.61362	0.60771	0.60248	0.60734	0.60838
6	0. 61729	0.61058	0.60799	0.60190	0.60512	0.60670
7	0. 61802	0.60779	0.60789	0.59895	0.60581	0.60847
8	0. 6156	0.61203	0.60883	0.60324	0.60474	0.60799
9	0. 61724	0.61185	0.60872	0.60125	0.60574	0.60782
10	0. 61895	0.60956	0.60855	0.60380	0.60664	0.60828
11	0. 61771	0.61152	0.6085	0.60423	0.60540	0.60746
12	0. 61801	0.61269	0.60774	0.60469	0.60428	0.60690
13	0. 61665	0.61043	0.60837	0.60187	0.60960	0.60730
14	0. 61875	0.60902	0.60872	0.60106	0.60348	0.60793
15	0. 61711	0.60823	0.60851	0.60341	0.60467	0.60784
16	0. 61773	0.60979	0.60851	0.60303	0.60344	0.60785
17	0. 61712	0.61169	0.60744	0.60402	0.60523	0.60769
18	0. 61877	0.61386	0.60818	0.60285	0.60540	0.60866
19	0. 61911	0.61137	0.60811	0.60141	0.60337	0.6077

Average Weighted MCC Values:

Classifier	**Before Feature Selection**	**After Feature Selection**
Random Forest	0.61977	0.60261
ADTree	0.61352	0.60553
Logistic	0.60829	0.60791

Weighted MCC Graph Plots:





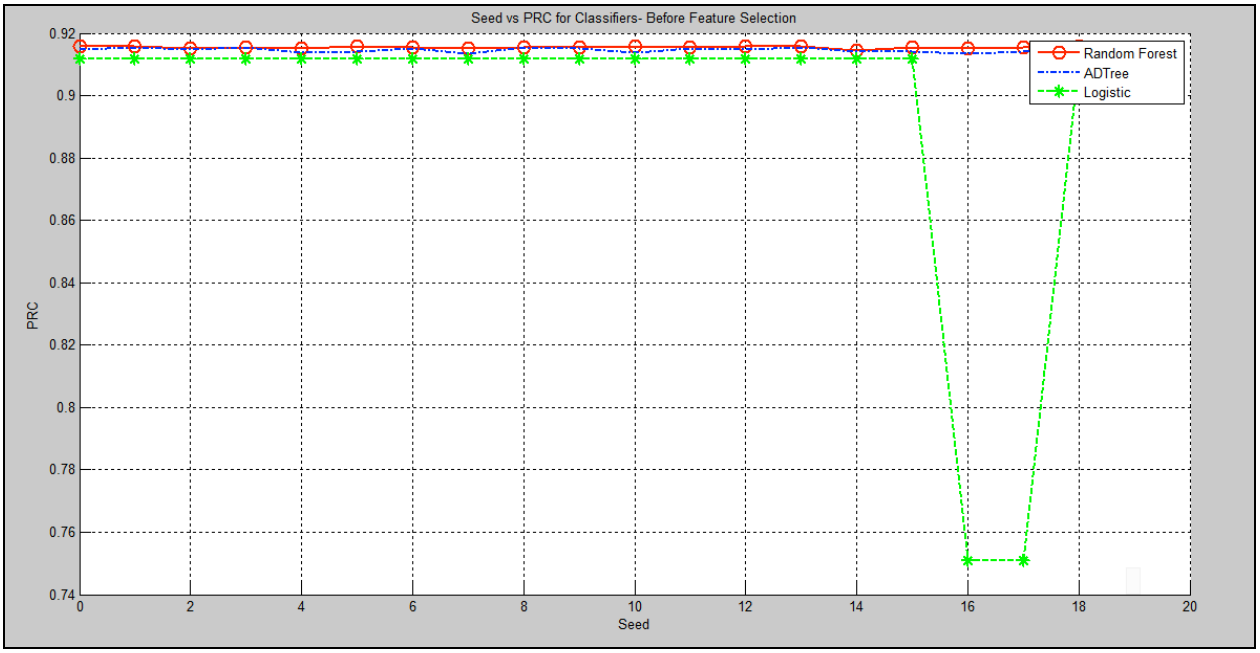
Weighted PRC Values:

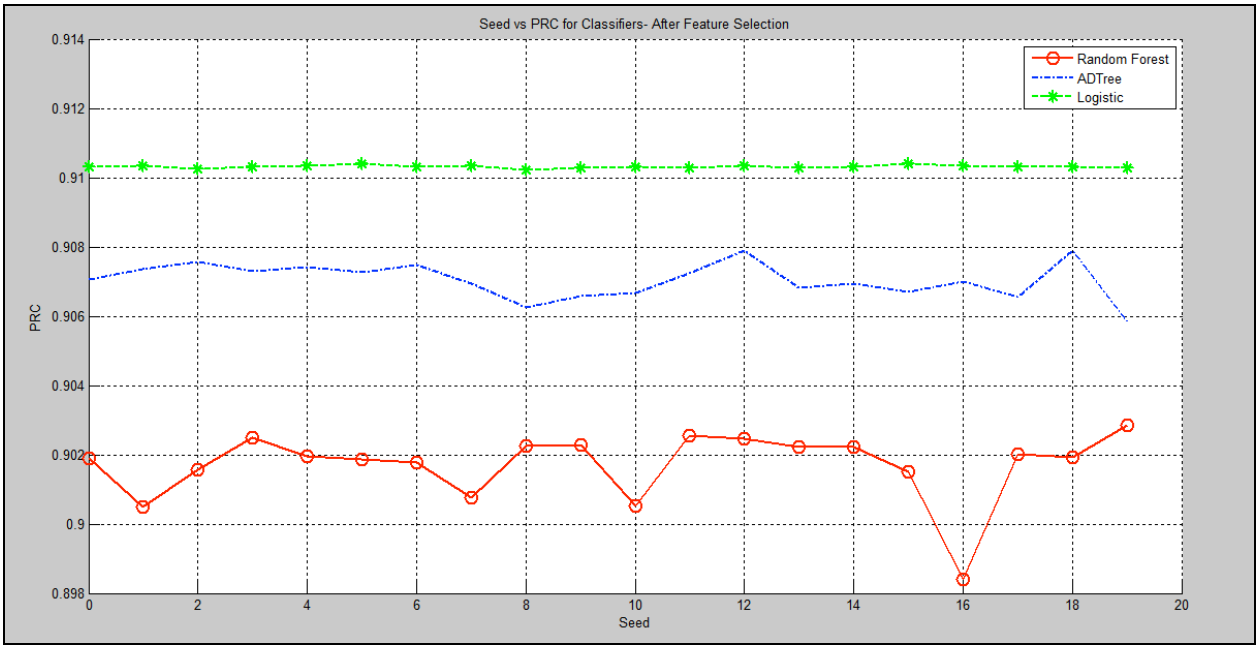
	Before Feature Selection[Original Dataset]			After Feature Selection[Reduced Dataset]		
Seed	**Random Forest**	**ADTree**	**Logistic**	**Random Forest**	**ADTree**	**Logistic**
0	0. 91570	0.91483	0.91184	0.90190	0.90707	0.91030
1	0. 91581	0.91536	0.91186	0.90050	0.90737	0.91033
2	0. 91515	0.91468	0.9118	0.90157	0.90756	0.91025
3	0. 91552	0.91534	0.91185	0.90248	0.90730	0.91031
4	0. 91529	0.91388	0.91182	0.90195	0.90743	0.91033
5	0. 91566	0.91408	0.91192	0.90187	0.90728	0.91040
6	0. 91532	0.91526	0.91184	0.90179	0.90747	0.9103
7	0. 91518	0.91336	0.91184	0.90075	0.90695	0.91033
8	0. 91542	0.91551	0.91176	0.90224	0.90626	0.91023
9	0. 91554	0.91505	0.91182	0.90228	0.90659	0.91027
10	0. 91571	0.91376	0.91184	0.90052	0.90667	0.91031
11	0. 91550	0.91514	0.91184	0.90255	0.90725	0.91027
12	0. 91576	0.91467	0.91187	0.90246	0.90789	0.91032
13	0. 91580	0.91533	0.91179	0.90221	0.90682	0.91028
14	0. 91459	0.91402	0.91186	0.90221	0.90693	0.91031
15	0. 91541	0.91421	0.91191	0.90150	0.90670	0.91040
16	0. 91512	0.91359	0.75083	0.89841	0.90699	0.91034
17	0. 91551	0.91407	0.75087	0.90202	0.90654	0.91031
18	0. 91565	0.91504	0.91183	0.90193	0.90791	0.91030
19	0. 91542	0.91535	0.91179	0.90284	0.90585	0.91029

Weighted PRC Average Values:

Classifier	**Before Feature Selection**	**After Feature Selection**
Random Forest	0.91545	0.90169
ADTree	0.91462	0.90704
Logistic	0.89573	0.91030

Weighted PRC Graph Plots:





7. Experiment Results – T test

Based on the weighted mcc, the performance of the classifier can be compared using paired T-test.

Strategy behind Paired T-test:

Consider δ(difference) as observed values of a set of i.i.d. random variables

• Null hypothesis: The 2 learning systems have the same accuracy

• Alternative hypothesis: One of the systems is more accurate than the other

• Hypothesis test:

– Use paired t-test do determine probability p that mean of δ's would arise from null hypothesis

– If p is sufficiently small (typically < 0.05) then reject the null hypothesis

Comparing systems using a paired t test:

- Calculate the sample mean

$$\bar{\delta} = \frac{1}{n}\sum_{i=1}^{n}\delta_i$$

- Calculate the t statistic

$$t = \frac{\bar{\delta}}{\sqrt{\frac{1}{n(n-1)}\sum_{i=1}^{n}(\delta_i - \bar{\delta})^2}}$$

- Determine the corresponding p-value, by looking up t in a table of values for the Student's t-distribution with n-1 degrees of freedom

7.1 T-Test on Random Forest and ADTree:

Paired T-Test has been performed on the weighted mcc values of Random forest and ADTree to determine the better performing algorithm.

The following table shows the different T-test values:

Sample mean	0.006764
T value	15.5369402
Degree of freedom	19
Significance level	0.05
Hypothesis	Two-tailed
P value	< .00001

Observation:

The result is significant at $p < .05$ and we can conclude that Random Forest performs better than ADTree for the given dataset.

7.2 T-Test on Random Forest and Logistics:

Paired T-Test has been performed on the weighted mcc values of Random forest and Logistics to determine the better performing algorithm.

The following table shows the different T-test values:

Sample mean	0.0096265
T value	37.276
Degree of freedom	19
Significance level	0.05
Hypothesis	Two-tailed
P value	< .00001

Observation:
The result is significant at $p < .05$ and we can conclude that Random Forest performs better than Logistics for the given dataset.

7.3 T-Test on ADTree and Logistics:

Paired T-Test has been performed on the weighted mcc values of ADTree and Logistics to determine the better performing algorithm.

The following table shows the different T-test values:

Sample mean	0.0028625
T value	6.760630749
Degree of freedom	19
Significance level	0.05
Hypothesis	Two-tailed
P value	< .00001

Observation:
The result is significant at $p < .05$ and we can conclude that ADTree performs better than Logistics for the given dataset.

8. Conclusion

In our experiments, the random forest classifier performed well in the classification of the training data set. Hence, it has been used for making predictions for the given data set considering they do not overfit because of the law of large numbers. In terms of accuracy, the accuracy without feature selection was comparably better than the accuracy with feature selection. The number of trees (K) and the number of attributes(I) have been tuned to achieve maximum accuracy. The tree gave us best results when K is set to 400 (a larger K value gives better performance). Despite the fact that the default value for I is the square root of the whole number of attributes, the tree gave us the best result when tuned to 0. Thus, according to our experiments, since the Random Forest algorithm performed better, it was used for predicting the dropout ratio of online courses.

9. References

- https://weka.wikispaces.com
- www.mathworks.com
- Learning from Data – Abu-Mostafa, Magdon-Ismail, Lin