Data Mining Project for Orange Data

Prepared for: Russel Pears

Prepared by: Mao Chuan Li

Student ID: 14854389

Submit Date: 2015/06/01

Paper Name: Data Mining and Machine Learning

Paper Number: COMP809

Abstract

Targeting a specific portion of customers to promote new services or products is widely needed in current businesses. The French Telecommunications giant Orange is no exception. A dataset with 5000 customers detail information each featured with 230 attributes is given to build a classification model to predict those potential customers willing to respond to the “up-selling” offers.

A clear winner has been identified with a high 0.86 Area under ROC value, which is a little lower than the KDD Cup 2009 winner IBM’s AUC value 0.9. Naïve Bayes with ‘useSupervisedDiscretization” flag has beaten all the other 4 algorithms used in the experiment. With a boosting method applied on top of that, although the Bagging boosting did not yield a better performance, but still it maintained the same performance of 0.86 AUC value.

Contents

1 Introduction

Almost each business company saves their customers' information as one of their most importantly valuable assets. The French Telecommunication giant Orange has such data with 100,000 customer records, each containing 15,000 attributes. A small subset of this data with only 5000 customers and 230 attributes is used in this research. The same as the purpose in the KDD Cup 2009, this research is to identify those customers who will respond to "up-selling" offers at the time of their purchase.

At first, an introduction of the application domain of the problem is given briefly. After that, a thorough dataset analysis and detailed explanation of the experiments followed. For the experiments, a few techniques were harassed to pre-process the data to tackle the missing values problems and imbalance issue; for dealing with the problem of curse of dimensionality, 2 feature selection methods were introduced; 5 different mining schemes were used with different set of parameters.

A few winners of the combination of the above techniques and classifiers are identified with the help of AUC metric, and then 3 boosting methods are implemented trying to improve the performance further.

At last, the winner of all combinations is identified, and then compared to the winner algorithm used by the KDD 2009 winner IBM.

2 Background

2.1 Application Domain

Market management is one of the most popular application domains that Data Mining technique is good at and have been applied for. The dataset being used in this report is just another one falling into this category.

All the customers are labelled either as "-1", which denotes they will not show interest to "up-selling" offers, or "1" which denotes that they have strong potential desire to respond to the offers.

With only 2 labels involved, all we need to do is to separate the customers into 2 groups, which makes the classification problem as simple as a binary or binominal classification problem. For such a requirement, a list of classification or regression models could be used, such as decision trees, Bayesian networks, support vector machines, artificial neural networks.

2.2 Purpose of the Research

As with any other marketing management applications, the only purpose of this research is to precisely identify those customers who are willing to respond to the "up-selling" offers, and grip those customers to promote Orange company's services or products to effectively and efficiently increase the sales.

Besides the precision of targeting-customers requirement, one other important factor should be the timing of that, because Orange sales persons should grab the golden time when a customer is making a purchase to decide if he needs to further promote other products.

3 Experimental Study

This section describes how the experiment was conducted. At first, the dataset is analysed in depth, followed by a list of mining schemes used in the experiment, and their strength and limitations analysis. And then the pre-processing techniques are explained how to tackle the missing values and imbalance issues. After pre-processing, 2 feature selection methods are used to dramatically decrease the dimension of each customer's attributes. Following that, a few different parameters for each mining scheme are explained.

For all combinations, a performance metric is defined to measure the mining algorithms. With it, the top 5 classifiers are selected. Afterward, to boost the performance of these 5 classifiers, 3 different boosting methods are used.

At last one of the best combination of one classifier and one boosting method is selected according to the metric. With the winner, a comparison with the KDD Cup 2009 winner is shown.

3.1 Data Set Description

The researcher was given 2 datasets that were extracted from the KDD Cup 2009 Orange data by professor Russel Pears. One with 1676 records is used for training, and the other with 3333 records is used for testing and measuring the performance.

Each customer record has a high dimension of 230 attributes, in which 40 of them are categorical. The last attribute is the class attribute which has 2 labels: -1 and 1.

At first glance of the 2 datasets, the class distribution is highly imbalanced, with only 8% of total customers are labelled with “1”, who are willing to accept the “up-selling” offers. The imbalance issue could severely impact the performance of each classifier, so 3 methods to tackle this issue are utilized as shown in section 3.3.

Second issue of the datasets is the huge numeric attributes that has a range of scales, ranging from 0~0(var90) to -5365400~5001520(var113).

Third issue is that some of the 40 nominal attributes have more than too many values. One extreme example is the attribute var217, which has 1329 distinct values.

The fourth issue comes to the huge amount of missing values in both training and test datasets.

The last but not least issue is the attribute names and the values are all scrambled because of need of maintaining the confidentiality of Orange customers.

3.2 Mining Schemes

Firstly, the requirement of the project is to separate the customers into 2 groups, each of whom holds a label. This makes this a supervised learning problem. Because there are only 2 classes (-1, 1) of the customers, both classification models and regression models could be used to mining the data.

In this research, 5 different classifiers are used:

3.2.1 weka.classifiers.trees.J48

J48 is an improved version of C4.5 decision tree, which can deal with numeric data, missing values and noisy data. Its advantage is the extreme fast speed of construction of the tree and classification of unknown data. Also the generated tree could easily be recognized by human to understand the logic of splitting data.

3.2.2 weka.classifiers.bayes.NaiveBayes

Naïve Bayes is another robust and easy to understand classifier as Decision Trees, which assumes all attributes are independent from each other, and calculate the unknown data’s probability according to history data so as to predict the class. It can deal well with unknown/missing values and numeric data with binning techniques. The only pitfall of it is the assumption of all attributes’ independence, which is not realistic most of the time. Even with this limitation, it still has good performance in practice.

3.2.3 weka.classifiers.bayes.BayesNet

Bayesian Network is an improved version of Naïve Bayes model which tries to relax the assumption of attributes independence by generating a directed acyclic graph (DAG) to represent a set of random variables and their dependence relationship.

The Bayesian Network model could easily handle missing values as Naïve Bayes does, and additionally provide a transparent representation of relationships between attributes. Again as Naïve Bayes, it has difficulty dealing with the continuous data.

3.2.4 weka.classifiers.functions.SMO

Support Vector Machine (SVM) is a powerful method for classification, especially for 2-class classification. It uses a kernel function to transform the data into higher dimension so as to find a separation line between 2 class instances, and uses quadratic programming to search for the best classifier boundary.

The advantages of SVM is that it is based on sound mathematical theory, and robust even there is error in the training dataset. The drawbacks of it is it takes long time to train the model, and difficult to under. Also the correctly set parameters for the model has to be tried again and again to get a satisfactory result.

3.2.5 weka.classifiers.lazy.k-NN

K-nearest-neighbour is one lazy instance-based learning classifier, compared to the previous four classifiers, which does not train the model upfront, but delay the training as late as seeing the new unknown data. By looking for *k* nearest instances in the history data to predict the class of the unknown data with the majority class of the nearest neighbours. (Witten, Frank, & Hall, 2011, p. 78).

The big advantage of k-NN is that it is as simple as Decision Trees, robust to noisy training data, has no training time, and the characteristics of all attributes do not need to be understood. The drawbacks are not trivial on the other hand. The biggest problem goes to the prediction time which increases when the volume of history data grows. Another issue with it is that more attributes of a sample has, more prediction time is needed. The last one is about the dealing method with nominal attributes, which is always difficult to calculate the distances between nominal values.

3.3 Pre-Processing

With the analysis of the Orange dataset, a series of actions were taken on dataset to clean the data. Firstly, those attributes with more than 90% values missed were removed with a newly developed Weka Filter. Secondly the records with 50% missing values are deleted, which led to only 9 (6%) loss of the minority class records. With the 2 actions, there are 70 attributes left (seriously dropped from 230) in the training dataset with 1487 records left. For details of the attributes, please check appendix 7.1.

For dealing with the high imbalance issue of the datasets, 3 different techniques were utilized:

- Over Sampling – weka.filters.supervised.instance.SMOTE filter is used to increase the minority class to 1134 from 126
- Under Sampling – weka.filters.supervised.instance.SpreadSubsample filter is used to shrink the majority class down to 150 from 1361
- Cost Matrix – a cost matrix is defined beforehand for each classifier(this has nothing to do at the pre-processing stage, later at training time, a weka.classifiers.meta.CostSensitiveClassifier meta classifier will be used):

0	1
10	0

For each of the numeric attributes has different scales, an additional normalization for the numeric attributes with help of weka.filters.unsupervised.instance.Normalize filter is conducted.

There is still a small portion of missing values in the dataset after these actions, they are left to the mining algorithms which could all deal with this problem. For outlier's issue, a few attempts were made to identify and delete them, but it turned out to delete too many minority class samples. So at last all attempts were abandoned, and the issue was left to each algorithm again.

3.4 Feature Selection

After pre-processing stage, there are 70 features left, which is still a large number. Two feature selection methods are used to further decrease the dimensions:

- Information Gain Attribute Ranking – “This is one of the simplest (and fastest) attribute ranking methods” (Hall & Holmes, 2003). weka.attributeSelection.InfoGainAttributeEval evaluator and weka.attributeSelection.Ranker search method are used to search and rank the attributes, each of which provides a significant information for predicting the class.
- Correlation-based Feature Selection – in contrast with Ranking method, CFS takes all attributes into account and tries to evaluate the subsets of attributes as a whole to select the most predictable attributes. weka.attributeSelection.CfsSubsetEval evaluator and weka.attributeSelection.LinearForwardSelection selector are used.

After applying the pre-processing actions and feature selections, a combination of 12 pairs of training and testing datasets are generated with different samples and features as shown in Table 1. For details of the attributes, please see appendix 7.2.

	Over Sampling	Under Sampling	Cost Matrix	None
Ranking	Samples: 2495 Attributes:33	Samples: 276 Attributes: 26	Samples: 1487 Attributes: 18	Samples: 1487 Attributes: 18
CFS	Samples: 2495 Attributes: 22	Samples: 276 Attributes: 6	Samples: 1487 Attributes: 3	Samples: 1487 Attributes: 3
None	Samples: 2495 Attributes:70	Samples: 276 Attributes: 70	Samples: 1487 Attributes: 70	Samples: 1487 Attributes: 70

Table 1 - Datasets to be trained and tested status

3.5 Parameter Tuning

For each of the mining schemes, besides the default settings suggested by Weka package, a few more tuning to each of the algorithms are shown in Table 2:

- weka.classifiers.trees.J48 - for there are many nominal attributes in the datasets, so “binarySplits = true” is firstly set. After that, the tree node’s minimal leave number is increased to 5 with “minObjNum = 5”, meanwhile “reducedErrorPruning = true” is enabled to help reduce the tree size and improve accuracy of prediction.

- weka.classifiers.bayes.NaiveBayes - the “useSupervisedDiscretization” flag is turned on to explicitly discretize those numeric features.
- weka.classifiers.bayes.BayesNet - the search algorithm "weka.classifiers.bayes.net.search.local.TAN" is used to search the space of all possible combinations of edges (Friedman, Geiger, & Goldszmidt, 1997).
- weka.classifiers.functions.SMO – the (Gaussian) radial basis function kernel is also used.
- weka.classifiers.lazy.k-NN – the default k value in weka is 1, 2 more values 5 and 10 are used additionally.

<table>
<tr><th></th><th>Classifier</th><th>Parameter Tuning</th></tr>
<tr><td>1</td><td rowspan="5">J48</td><td>default</td></tr>
<tr><td>2</td><td>binarySplits = true</td></tr>
<tr><td rowspan="3">3</td><td>binarySplits = true</td></tr>
<tr><td>minObjNum = 5</td></tr>
<tr><td>reducedErrorPruning = true</td></tr>
<tr><td>4</td><td rowspan="2">Naïve Bayes</td><td>default</td></tr>
<tr><td>5</td><td>useSupervisedDiscretization=true</td></tr>
<tr><td>6</td><td rowspan="2">Bayesian Network</td><td>default</td></tr>
<tr><td>7</td><td>searchAlgorithm = TAN –S Bayes</td></tr>
<tr><td>8</td><td rowspan="2">SMO</td><td>default</td></tr>
<tr><td>9</td><td>Kernel = RBFKernel</td></tr>
<tr><td>10</td><td rowspan="3">kNN</td><td>default</td></tr>
<tr><td>11</td><td>kNN = 5</td></tr>
<tr><td>12</td><td>kNN = 10</td></tr>
</table>

Table 2 - Classifier with different parameters

3.6 Performance Metrics

Each data mining application domain has different requirements. The purpose of this research is to precisely identify the customers who are willing to respond to the “up-selling” offers at the time of purchase of a service or product. So the following metrics in order are used to evaluate the performance of each classifier model.

- Area under ROC curve (AUC) - used as a measure of quality of a probabilistic classifier (Miha & Tomaz, 2006). The greater the AUC value is, the more accurate the classifier model is.

- The model evaluation time – because the Orange company sales persons need to evaluate if the customer is potential customers that will respond to "up-selling" offers, and try his/her best to entice the customer within a very short period.
- The model training time – with customers' data growing, the classifier model needs to periodically rebuild.

3.7 Boosting Techniques

Ensemble Data Mining Methods are those machine learning methods that take advantage of more than one single model to provide better accuracy than any one of the models alone (Maclin & Opitz, 2011). In Weka, Bagging and Adaboost are 2 popular boosting techniques, both of which use a single learner and create multiple models to help improve the prediction accuracy. Another method "Voting" that supports multiple learning methods is used as well in this research.

3.8 Experimental Setups

In the pre-processing stage, 12 pairs of datasets have been generated with different kinds of sampling and feature selection methods. In section 3.5, 5 different mining algorithms with either 2 or 3 different parameter tunings have been defined. Every algorithm with each parameter setting is trained on each of the training datasets, and then evaluated with the corresponding test dataset. In total, there are 144 models created.

With the 144 models evaluation results, 5 best of the models are identified according to the metrics pre-defined. All the models are ranked by the AUC value from top to down. The training time for the 1667 (over sampling) / 276 (under sampling) samples and evaluation time for the 3333 samples are all below a second, ranging from 208 to 967 milliseconds as shown in Table 3. For a whole list of 144 models performance, please check appendix 7.3.

ReSample	Selector	Classifier	Options	Training Time	Test Time	Precision	Recall	AUC
under	CfsSubset	NaiveBayes	-D	208	443	22.36%	72.69%	0.86
under	CfsSubset	J48	default	359	394	18.82%	74.17%	0.84
under	CfsSubset	J48	-C 0.25 -B -M 5	225	334	18.82%	74.17%	0.84
under	CfsSubset	BayesNet	default	437	681	14.38%	94.46%	0.84
under	Ranker	J48	-C 0.25 -B -M 5	967	409	20.40%	67.16%	0.83

Table 3 - top 5 models with best AUC

With above test results, 2 pairs of training and testing datasets are proved to provide the best information for the 5 algorithms: Under sampling with CfsSubset and Under sampling with Ranker. And 4 classifier models performed better than any others.

Afterward, 3 boosting methods are applied to the above 5 combinations to boost the performance of them. The 2 boosting techniques Bagging and Adaboost1 are applied with the above winner models each, and Voting just takes all the 4 winner algorithms and runs against the 2 pairs of dataset. 18 models with different boosting methods are run as the appendix 7.4 show. By measuring the AUC values again, the overall winner of all combinations is shown as the Table 4 which yields the best AUC value and reasonable training and testing time (below 1 second):

ReSample	Selector	Boosting	Options	Training Time	Test Time	Precision	Recall	AUC
under	CfsSubset	Bagging	NaiveBayes -D	525	471	22.42%	73.06%	0.86
under	CfsSubset	Vote	all models	789	1101	17.52%	84.13%	0.85
under	CfsSubset	AdaBoostM1	J48 -B -M 5	1085	218	23.01%	71.59%	0.85
under	CfsSubset	Bagging	J48 -B -M 5	968	251	20.56%	70.85%	0.85
under	CfsSubset	Bagging	BayesNet	1108	682	14.75%	92.99%	0.84

Table 4 - Boosting methods performance

4 Results

4.1 Analysis

4.1.1 Resampling

The Orange dataset is highly imbalanced in class distribution. Three techniques were utilized to encounter this problem: under resampling, over resampling and using Cost Matrix. Relatively, under resampling and cost matrix have successfully avoided the imbalance problem, especially the under resampling performed the best.

Num	ReSample	Selector	Training Time	Test Time	True Negative	False Positive	False Negative	True Positive	Precision	Recall	AUC
55	under	CfsSubset	208	443	2378	684	74	197	22.36%	72.69%	0.86
115	matrix	CfsSubset	333	463	2888	174	137	134	43.51%	49.45%	0.81
139	none	CfsSubset	384	488	3009	53	187	84	61.31%	31.00%	0.81
2	over	CfsSubset	811	238	2892	170	185	86	33.59%	31.73%	0.79

Table 5-best resampling method

4.1.2 Feature Selection

There were 2 feature selection methods used based on single variance and multiple variances. In all test cases, the CfsSubset evaluator and LinearForwardSelection selector outperformed the Information Gain evaluator and Ranker selector.

Num	ReSample	Selector	Train Time	Test Time	True Negative	False Positive	False Negative	True Positive	Precision	Recall	AUC
55	under	CfsSubset	208	443	2378	684	74	197	22.36%	72.69%	0.86
98	under	Ranker	967	409	2352	710	89	182	20.40%	67.16%	0.83

Table 6- best selector

4.1.3 Parameter Tuning

In all 5 mining schemes, Naïve Bayes and IBk classifiers have shown slight performance improvement, especially the Naïve Bayes. All other 3 did not show any improvement, in contrast, a significant downgrade has been observed in the case of BayesNet.

Num	Options	Train Time	Test Time	True Negative	False Positive	False Negative	True Positive	Precision	Recall	AUC
J48										
49	default	359	394	2195	867	70	201	18.82%	74.17%	0.84
50	defaultt + 5 leaves + binarySplit	225	334	2195	867	70	201	18.82%	74.17%	0.84
NaiveBayes										
55	default + -D	208	443	2378	684	74	197	22.36%	72.69%	0.86
115	default + -D	333	463	2888	174	137	134	43.51%	49.45%	0.81
139	default + -D	384	488	3009	53	187	84	61.31%	31.00%	0.81
54	default	86	769	1909	1153	59	212	15.53%	78.23%	0.80
BayesNet										
56	default	437	681	1538	1524	15	256	14.38%	94.46%	0.84
140	default	564	538	3009	53	186	85	61.59%	31.37%	0.79
116	default	567	674	2884	178	137	134	42.95%	49.45%	0.79
9	default + TAN	2498	708	3060	2	267	4	66.67%	1.48%	0.73
IBK										
60	default + 10 NN	39	1399	2763	299	132	139	31.74%	51.29%	0.78
120	default + 10 NN	155	2281	2814	248	137	134	35.08%	49.45%	0.76
144	default + 10 NN	120	2060	2983	79	144	127	61.65%	46.86%	0.75
59	default + 5 NN	38	1541	2692	370	134	137	27.02%	50.55%	0.75
119	default + 5 NN	121	2455	2792	270	138	133	33.00%	49.08%	0.75

143	default + 5 NN	68	2355	2985	77	145	126	62.07%	46.49%	0.75
58	default	87	1066	2585	477	136	135	22.06%	49.82%	0.70
				SMO						
52	default	12151	5715	2125	937	76	195	17.23%	71.96%	0.71
64	default	30026	20245	1961	1101	104	167	13.17%	61.62%	0.63
100	default	27739	19315	1940	1122	104	167	12.96%	61.62%	0.62
4	default	6654	748	2834	228	199	72	24.00%	26.57%	0.60
125	default + RFBKernel	3E+06	2E+06	2667	395	189	82	17.19%	30.26%	0.59

Table 7 - tuning effects

4.1.4 Boosting Techniques

In the experiment, 3 boosting techniques were harassed: bagging, boosting and voting. In general, all these boosting methods did not significantly improve the performances as expected, whereas the training time all increased significantly.

ReSample	Selector	Classifier	Options	Train Time	Test Time	Precision	Recall	AUC
under	CfsSubset	NaiveBayes	-D	208	443	22.36%	72.69%	0.86
under	CfsSubSet	Bagging	NaiveBayes -D	525	471	22.42%	73.06%	0.86
under	CfsSubset	J48	default	359	394	18.82%	74.17%	0.84
under	CfsSubSet	Bagging	J48	706	273	61.65%	46.86%	0.84
under	CfsSubset	J48	-C 0.25 -B -M 5	225	334	18.82%	74.17%	0.84
under	CfsSubSet	AdaBoostM1	J48 -B -M 5	1085	218	23.01%	71.59%	0.85
under	CfsSubset	BayesNet	default	437	681	14.38%	94.46%	0.84
under	CfsSubSet	Bagging	BayesNet	1108	682	14.75%	92.99%	0.84
under	Ranker	J48	-C 0.25 -B -M 5	967	409	20.40%	67.16%	0.83
under	Ranker	Bagging	BayesNet	2446	1589	15.24%	64.58%	0.73
under	CfsSubSet	Vote	all models	789	1101	17.52%	84.13%	0.85
under	Ranker	Vote	all models	2046	543	16.93%	64.21%	0.78

Table 8 - Boosting Effects

4.1.5 Run time

In terms of training time and test time, SMO mining scheme took the longest time to train (78 minutes) and evaluate (18 minutes). BayesNet has spent around 19 minutes to train, while the test time became reasonable, being 2.7 seconds. All other 3 schemes have maintained a reasonable runtime for both training and testing. Even IBk, the lazy classifier only spent 65 seconds for the testing.

All the following results are all records with the longest runtime for each mining scheme, one notable matter of these records is that all of them are related to the over sampling methods.

Num	ReSample	Selector	Classifier	Options	Train Time	Test Time
17	over	none	SMO	default + RFBKernel	4707470	1088846
93	matrix	none	BayesNet	default + TAN	1142027	2773
14	over	none	J48	default + binarySplit + 5 leaves	8250	738
19	over	none	NaiveBayes	default + -D	1860	838
22	over	none	IBK	default	182	65645

4.1.6 Comparison of Schemes

In the experiment, there are 5 mining schemes were applied to the Orange datasets. As the above Table 3 – top 5 models shows, Naïve Bayes, J48 and Bayes Net all yielded good performances, in which Naïve Bayes outperformed all the others with 0.02 AUC value and similar training and test time. Neither SMO nor IBk schemes performed well, besides that, the training and test time both went out of expectation.

4.2 Comparison with Previous Work

The Orange dataset used in this research was the customer data provided by the French Telecom company Orange for KDD Cup 2009 competition to predict the "the propensity of customers to switch providers (churn), buy new products or services (appetency), or buy upgrades or add-ons (up-selling)" (Niculescu-Mizil, 2009). This research is targeting the up-selling problem in the slow challenge. Here is the comparison of the winner in this report with the KDD Cup winner IBM.

Classifier Type	Competitor	Scheme	Up-Selling AUC
Boosting	This research	Bagging	0.86
	IBM	Ensemble Selection	0.9091
Best Single	This research	Naïve Bayes	0.86
	IBM	Boosted Trees	0.9025

Table 9 - the comparison between this research and IBM

Apparently, the performance of IBM schemes is better than this research. This result might be due to the following 2 major differences:

- Feature Selection and Construction - a new set of features based on the features with high mutual information was added to help increase the prediction accuracy in IBM schemes. In this research, there is not.

- Hundreds of models – IBM tried thousands of models, and finally put 500-1000 models in the final ensemble method models, which helped improve the performance dramatically. In contrast, this research only tried 5 models.

5 Conclusion

This research is to experiment on an Orange dataset which has 5000 customer data to predict the "up-selling" customers with 5 mining schemes and 3 boosting techniques. It turned out the Naïve Bayes performed the best out of all models with a 0.86 AUC value, a little lower than the KDD Cup 2009 winner IBM.

Being compared to IBM, the pre-processing of this this research is evidently deficient without more delicate analysis and processing. Another disadvantage is the small number of learning models. In future work, improving the work of these 2 aspects should be expected to approach the IBM score.

6 References

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning, 29*(2-3), 131-163. doi: 10.1023/A:1007465528199

Hall, M. A., & Holmes, G. (2003). Benchmarking attribute selection techniques for discrete class data mining. *Knowledge and Data Engineering, IEEE Transactions on, 15*(6), 1437-1447. doi: 10.1109/TKDE.2003.1245283

Isabelle Guyon, A. e. E. (2003). An_Introduction_to_Variable_and_Feature_Selection. *Journal of Machine Learning Research*(3), 1157-1182.

Maclin, R., & Opitz, D. (2011). Popular Ensemble Methods: An Empirical Study. doi: 10.1613/jair.614

Miha, V., & Tomaz, C. (2006). ROC Curve, Lift Chart and Calibration Plot. *Metodoloski Zvezki, 3*(1), 89.

Niculescu-Mizil, A., Perlich, C., Swirszcz, G., Sindhwani, V., Liu, Y., Melville, P., ... & Zhu, Y. F. (2009). Winning the KDD cup orange challenge with ensemble selection. *The 2009 Knowledge Discovery in Data Competition (KDD Cup 2009) Challenges in Machine Learning, 3*(21).

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining: practical machine learning tools and techniques*. Burlington, MA: Morgan Kaufmann.

7 Appendix

7.1 Dataset Status after Cleaning Missing Values

Relation Name: orange_train-weka.filters…..

Num Instances: 1487

Num Attributes: 70

	Name	Type	Nom	Int	Real	Missing	Unique	Dist
1	Var6	Num	0%	99%	0%	13 / 1%	195 / 13%	448
2	Var7	Num	0%	99%	0%	15 / 1%	0 / 0%	6
3	Var13	Num	0%	99%	0%	15 / 1%	393 / 26%	589
4	Var21	Num	0%	99%	0%	13 / 1%	78 / 5%	207
5	Var22	Num	0%	100%	0%	0 / 0%	78 / 5%	207
6	Var24	Num	0%	95%	0%	76 / 5%	13 / 1%	33
7	Var25	Num	0%	100%	0%	0 / 0%	27 / 2%	85
8	Var28	Num	0%	9%	91%	0 / 0%	380 / 26%	466
9	Var35	Num	0%	100%	0%	0 / 0%	1 / 0%	8
10	Var38	Num	0%	100%	0%	0 / 0%	1107 / 74%	1127
11	Var44	Num	0%	100%	0%	0 / 0%	1 / 0%	4
12	Var57	Num	0%	0%	100%	0 / 0%	1426 / 96%	1456
13	Var65	Num	0%	99%	0%	15 / 1%	2 / 0%	9
14	Var72	Num	0%	60%	0%	588 / 40%	1 / 0%	6
15	Var73	Num	0%	100%	0%	0 / 0%	7 / 0%	103
16	Var74	Num	0%	99%	0%	15 / 1%	47 / 3%	124
17	Var76	Num	0%	100%	0%	0 / 0%	1056 / 71%	1083
18	Var78	Num	0%	100%	0%	0 / 0%	2 / 0%	8
19	Var81	Num	0%	10%	89%	13 / 1%	1449 / 97%	1450
20	Var83	Num	0%	100%	0%	0 / 0%	18 / 1%	40
21	Var85	Num	0%	100%	0%	0 / 0%	23 / 2%	50
22	Var94	Num	0%	60%	0%	588 / 40%	844 / 57%	862
23	Var109	Num	0%	95%	0%	76 / 5%	27 / 2%	65
24	Var112	Num	0%	100%	0%	0 / 0%	31 / 2%	72
25	Var113	Num	0%	39%	61%	0 / 0%	1460 / 98%	1461
26	Var119	Num	0%	99%	0%	13 / 1%	169 / 11%	409
27	Var123	Num	0%	100%	0%	0 / 0%	35 / 2%	72
28	Var125	Num	0%	99%	0%	15 / 1%	886 / 60%	1002
29	Var126	Num	0%	72%	0%	412 / 28%	2 / 0%	49
30	Var132	Num	0%	100%	0%	0 / 0%	1 / 0%	13
31	Var133	Num	0%	100%	0%	0 / 0%	1292 / 87%	1309
32	Var134	Num	0%	100%	0%	0 / 0%	1163 / 78%	1190
33	Var140	Num	0%	99%	0%	15 / 1%	335 / 23%	518
34	Var143	Num	0%	100%	0%	0 / 0%	1 / 0%	3
35	Var144	Num	0%	99%	0%	13 / 1%	0 / 0%	7

36	Var149	Num	0%	95%	0%	76 / 5%	687 / 46%	704
37	Var153	Num	0%	100%	0%	0 / 0%	1374 / 92%	1388
38	Var160	Num	0%	100%	0%	0 / 0%	48 / 3%	118
39	Var163	Num	0%	100%	0%	0 / 0%	846 / 57%	873
40	Var173	Num	0%	100%	0%	0 / 0%	1 / 0%	3
41	Var181	Num	0%	100%	0%	0 / 0%	1 / 0%	4
42	Var192	Nom	99%	0%	0%	8 / 1%	30 / 2%	235
43	Var193	Nom	100%	0%	0%	0 / 0%	7 / 0%	29
44	Var195	Nom	100%	0%	0%	0 / 0%	4 / 0%	13
45	Var196	Nom	100%	0%	0%	0 / 0%	0 / 0%	2
46	Var197	Nom	100%	0%	0%	5 / 0%	25 / 2%	148
47	Var198	Nom	100%	0%	0%	0 / 0%	508 / 34%	789
48	Var199	Nom	100%	0%	0%	1 / 0%	335 / 23%	550
49	Var202	Nom	100%	0%	0%	0 / 0%	911 / 61%	1151
50	Var203	Nom	100%	0%	0%	5 / 0%	0 / 0%	3
51	Var204	Nom	100%	0%	0%	0 / 0%	3 / 0%	100
52	Var205	Nom	96%	0%	0%	60 / 4%	0 / 0%	3
53	Var206	Nom	99%	0%	0%	13 / 1%	0 / 0%	20
54	Var207	Nom	100%	0%	0%	0 / 0%	1 / 0%	10
55	Var208	Nom	100%	0%	0%	5 / 0%	0 / 0%	2
56	Var210	Nom	100%	0%	0%	0 / 0%	0 / 0%	6
57	Var211	Nom	100%	0%	0%	0 / 0%	0 / 0%	2
58	Var212	Nom	100%	0%	0%	0 / 0%	12 / 1%	48
59	Var216	Nom	100%	0%	0%	0 / 0%	168 / 11%	296
60	Var217	Nom	99%	0%	0%	14 / 1%	1030 / 69%	1196
61	Var218	Nom	99%	0%	0%	14 / 1%	0 / 0%	2
62	Var219	Nom	91%	0%	0%	133 / 9%	1 / 0%	9
63	Var220	Nom	100%	0%	0%	0 / 0%	508 / 34%	789
64	Var221	Nom	100%	0%	0%	0 / 0%	0 / 0%	7
65	Var222	Nom	100%	0%	0%	0 / 0%	508 / 34%	789
66	Var223	Nom	91%	0%	0%	133 / 9%	0 / 0%	4
67	Var226	Nom	100%	0%	0%	0 / 0%	0 / 0%	23
68	Var227	Nom	100%	0%	0%	0 / 0%	0 / 0%	6
69	Var228	Nom	100%	0%	0%	0 / 0%	3 / 0%	22
70	Var230	Nom	100%	0%	0%	0 / 0%	0 / 0%	2

7.2 12 Pairs of Training and Testing Datasets Detail

	Over Sampling	Under Sampling	Cost Matrix	None
Ranking	Ranked attributes: 0.92714 49 Var202 0.88847 60 Var217 0.74634 48 Var199	Ranked attributes: 0.94381 49 Var202 0.85052 60 Var217 0.74897 47 Var198	Ranked attributes: 0.3544 49 Var202 0.3469 60 Var217 0.2551 47 Var198	As in Cost Matrix

	0.66583 65 Var222 0.66583 63 Var220 0.66583 47 Var198 0.65352 42 Var192 0.57416 21 Var85 0.53693 27 Var123 0.52196 20 Var83 0.50494 51 Var204 0.49507 24 Var112 0.45878 59 Var216 0.45448 13 Var65 0.44677 35 Var144 0.43398 46 Var197 0.41873 7 Var25 0.40027 2 Var7 0.39884 23 Var109 0.36761 6 Var24 0.35942 67 Var226 0.30872 8 Var28 0.20689 4 Var21 0.20053 29 Var126 0.19023 30 Var132 0.17614 61 Var218 0.16074 53 Var206 0.13817 16 Var74 0.13259 52 Var205 0.12654 39 Var163 0.11342 15 Var73 0.10019 58 Var212	0.74897 63 Var220 0.74897 65 Var222 0.65649 48 Var199 0.52058 42 Var192 0.35058 59 Var216 0.30859 46 Var197 0.26949 51 Var204 0.18802 29 Var126 0.09757 8 Var28 0.08878 57 Var211 0.0713 43 Var193 0.06371 67 Var226 0.06269 69 Var228 0.05185 58 Var212 0.05072 53 Var206 0.02757 62 Var219 0.02548 44 Var195 0.02523 13 Var65 0.01361 56 Var210 0.01178 61 Var218 0.01127 50 Var203 0.01103 68 Var227	0.2551 63 Var220 0.2551 65 Var222 0.1922 48 Var199 0.1091 59 Var216 0.1048 42 Var192 0.0846 29 Var126 0.0704 46 Var197 0.0488 51 Var204 0.0217 8 Var28 0.0204 58 Var212 0.0198 57 Var211 0.019 43 Var193 0.0119 69 Var228 0.0114 67 Var226	
CFS	Var7 Var24 Var25 Var28 Var57 Var65 Var83 Var85 Var94 Var112 Var123 Var126	Var28 Var126 Var202 Var211 Var217	Var126 Var202	As in Cost Matrix

	Var132 Var144 Var203 Var205 Var211 Var218 Var219 Var226 Var227			
None	As in Section 7.1	As in Section 7.1	As in Section 7.1	As in Cost Matrix

7.3 All Single Models Evaluation Results

Num	ReSample	Selector	Classifier	Options	**Train Time**	**Test Time**	True Negative	False Positive	False Negative	True Positive	Precision	Recall	**AUC**
55	under	CfsSubset	NaiveBayes	default + -D	208	443	2378	684	74	197	22.36%	72.69%	0.86
49	under	CfsSubset	J48	default	359	394	2195	867	70	201	18.82%	74.17%	0.84
50	under	CfsSubset	J48	default + binarySplit + 5 leaves	225	334	2195	867	70	201	18.82%	74.17%	0.84
56	under	CfsSubset	BayesNet	default	437	681	1538	1524	15	256	14.38%	94.46%	0.84
98	under	Ranker	J48	default + binarySplit + 5 leaves	967	409	2352	710	89	182	20.40%	67.16%	0.83
115	matrix	CfsSubset	NaiveBayes	default + -D	333	463	2888	174	137	134	43.51%	49.45%	0.81
85	matrix	none	J48	default	2236	325	2280	782	73	198	20.20%	73.06%	0.81
139	none	CfsSubset	NaiveBayes	default + -D	384	488	3009	53	187	84	61.31%	31.00%	0.81
123	matrix	Ranker	J48	default + binary + 5 leaves + reduced	2013	356	2902	160	140	131	45.02%	48.34%	0.80
54	under	CfsSubset	NaiveBayes	default	86	769	1909	1153	59	212	15.53%	78.23%	0.80
140	none	CfsSubset	BayesNet	default	564	538	3009	53	186	85	61.59%	31.37%	0.79
26	none	Ranker	J48	default + binarySplit + 5 leaves	2244	282	3034	28	190	81	74.31%	29.89%	0.79
116	matrix	CfsSubset	BayesNet	default	567	674	2884	178	137	134	42.95%	49.45%	0.79
39	none	none	J48	default + binary + 5 leaves + reduced	3536	368	3039	23	167	104	81.89%	38.38%	0.79
2	over	CfsSubset	J48	default + binarySplit + 5 leaves	811	238	2892	170	185	86	33.59%	31.73%	0.79
74	over	Ranker	J48	default + binarySplit + 5 leaves	4048	435	3003	59	195	76	56.30%	28.04%	0.78
51	under	CfsSubset	J48	default + binary + 5 leaves + reduced	294	464	2983	79	144	127	61.65%	46.86%	0.78
109	matrix	CfsSubset	J48	default	753	263	2983	79	144	127	61.65%	46.86%	0.78
111	matrix	CfsSubset	J48	default + binary + 5 leaves + reduced	468	218	2983	79	144	127	61.65%	46.86%	0.78
121	matrix	Ranker	J48	default	2214	313	2983	79	144	127	61.65%	46.86%	0.78
63	under	none	J48	default + binary + 5 leaves + reduced	1061	499	2031	1031	66	205	16.59%	75.65%	0.78
99	under	Ranker	J48	default + binary + 5 leaves + reduced	659	344	2031	1031	66	205	16.59%	75.65%	0.78
60	under	CfsSubset	IBK	default + 10 NN	39	1399	2763	299	132	139	31.74%	51.29%	0.78
3	over	CfsSubset	J48	default + binary + 5 leaves + reduced	683	223	2895	167	158	113	40.36%	41.70%	0.78
122	matrix	Ranker	J48	default + binarySplit + 5 leaves	3897	300	2798	264	152	119	31.07%	43.91%	0.77

38	none	none	J48	default + binarySplit + 5 leaves	5265	414	3024	38	222	49	56.32%	18.08%	0.77
110	matrix	CfsSubset	J48	default + binarySplit + 5 leaves	1483	543	2968	94	143	128	57.66%	47.23%	0.77
14	over	none	J48	default + binarySplit + 5 leaves	8250	738	2997	65	193	78	54.55%	28.78%	0.76
120	matrix	CfsSubset	IBK	default + 10 NN	155	2281	2814	248	137	134	35.08%	49.45%	0.76
144	none	CfsSubset	IBK	default + 10 NN	120	2060	2983	79	144	127	61.65%	46.86%	0.75
27	none	Ranker	J48	default + binary + 5 leaves + reduced	1867	272	3044	18	187	84	82.35%	31.00%	0.75
103	under	Ranker	NaiveBayes	default + -D	277	496	2209	853	102	169	16.54%	62.36%	0.75
59	under	CfsSubset	IBK	default + 5 NN	38	1541	2692	370	134	137	27.02%	50.55%	0.75
75	over	Ranker	J48	default + binary + 5 leaves + reduced	2761	419	3002	60	165	106	63.86%	39.11%	0.75
67	under	none	NaiveBayes	default + -D	424	829	2215	847	102	169	16.63%	62.36%	0.75
119	matrix	CfsSubset	IBK	default + 5 NN	121	2455	2792	270	138	133	33.00%	49.08%	0.75
143	none	CfsSubset	IBK	default + 5 NN	68	2355	2985	77	145	126	62.07%	46.49%	0.75
15	over	none	J48	default + binary + 5 leaves + reduced	5121	458	3001	61	168	103	62.80%	38.01%	0.74
9	over	CfsSubset	BayesNet	default + TAN	2498	708	3060	2	267	4	66.67%	1.48%	0.73
62	under	none	J48	default + binarySplit + 5 leaves	1780	545	2215	847	71	200	19.10%	73.80%	0.72
8	over	CfsSubset	BayesNet	default	930	508	3061	1	270	1	50.00%	0.37%	0.72
104	under	Ranker	BayesNet	default	458	690	2122	940	107	164	14.86%	60.52%	0.72
127	matrix	Ranker	NaiveBayes	default + -D	481	461	2629	433	177	94	17.84%	34.69%	0.72
68	under	none	BayesNet	default	1024	689	2114	948	107	164	14.75%	60.52%	0.72
86	matrix	none	J48	default + binarySplit + 5 leaves	8047	335	2913	149	170	101	40.40%	37.27%	0.72
91	matrix	none	NaiveBayes	default + -D	1063	787	2633	429	175	96	18.29%	35.42%	0.71
43	none	none	NaiveBayes	default + -D	1025	836	3023	39	253	18	31.58%	6.64%	0.71
31	none	Ranker	NaiveBayes	default + -D	681	516	3034	28	255	16	36.36%	5.90%	0.71
52	under	CfsSubset	SMO	default	12151	5715	2125	937	76	195	17.23%	71.96%	0.71
57	under	CfsSubset	BayesNet	default + TAN	1E+05	570	2785	277	260	11	3.82%	4.06%	0.70
87	matrix	none	J48	default + binary + 5 leaves + reduced	3516	342	2963	99	177	94	48.70%	34.69%	0.70
58	under	CfsSubset	IBK	default	87	1066	2585	477	136	135	22.06%	49.82%	0.70
7	over	CfsSubset	NaiveBayes	default + -D	927	448	3051	11	268	3	21.43%	1.11%	0.70
128	matrix	Ranker	BayesNet	default	1061	446	2681	381	197	74	16.26%	27.31%	0.69

92	matrix	none	BayesNet	default	1942	712	2720	342	197	74	17.79%	27.31%	0.69
129	matrix	Ranker	BayesNet	default + TAN	8E+05	294	2722	340	201	70	17.07%	25.83%	0.69
44	none	none	BayesNet	default	1795	724	2990	72	245	26	26.53%	9.59%	0.69
32	none	Ranker	BayesNet	default	867	474	2996	66	247	24	26.67%	8.86%	0.69
1	over	CfsSubset	J48	default	643	338	2876	186	187	84	31.11%	31.00%	0.69
138	none	CfsSubset	NaiveBayes	default	147	362	3062	0	271	0	0.00%	0.00%	0.68
33	none	Ranker	BayesNet	default + TAN	9E+05	550	3061	1	269	2	66.67%	0.74%	0.67
142	none	CfsSubset	IBK	default	107	2433	2923	139	172	99	41.60%	36.53%	0.67
118	matrix	CfsSubset	IBK	default	127	1991	2809	253	167	104	29.13%	38.38%	0.67
114	matrix	CfsSubset	NaiveBayes	default	207	454	2384	678	138	133	16.40%	49.08%	0.67
69	under	none	BayesNet	default + TAN	1E+06	658	2062	1000	121	150	13.04%	55.35%	0.65
93	matrix	none	BayesNet	default + TAN	1E+06	2773	2825	237	228	43	15.36%	15.87%	0.65
102	under	Ranker	NaiveBayes	default	93	575	2107	955	134	137	12.55%	50.55%	0.65
12	over	CfsSubset	IBK	default + 10 NN	125	17934	2601	461	192	79	14.63%	29.15%	0.65
107	under	Ranker	IBK	default + 5 NN	48	2060	1922	1140	107	164	12.58%	60.52%	0.65
81	over	Ranker	BayesNet	default + TAN	1E+06	381	3062	0	270	1	100.00%	0.37%	0.64
105	under	Ranker	BayesNet	default + TAN	1E+06	736	2008	1054	113	158	13.04%	58.30%	0.64
13	over	none	J48	default	3624	478	2601	461	166	105	18.55%	38.75%	0.64
45	none	none	BayesNet	default + TAN	9E+05	929	3062	0	271	0	0.00%	0.00%	0.63
64	under	none	SMO	default	30026	20245	1961	1101	104	167	13.17%	61.62%	0.63
80	over	Ranker	BayesNet	default	1975	481	3062	0	271	0	0.00%	0.00%	0.63
126	matrix	Ranker	NaiveBayes	default	226	598	2543	519	193	78	13.07%	28.78%	0.63
100	under	Ranker	SMO	default	27739	19315	1940	1122	104	167	12.96%	61.62%	0.62
108	under	Ranker	IBK	default + 10 NN	69	2340	2231	831	156	115	12.16%	42.44%	0.62
79	over	Ranker	NaiveBayes	default + -D	1547	592	3062	0	271	0	0.00%	0.00%	0.62
6	over	CfsSubset	NaiveBayes	default	363	380	1754	1308	105	166	11.26%	61.25%	0.62
19	over	none	NaiveBayes	default + -D	1860	838	3059	3	270	1	25.00%	0.37%	0.62
11	over	CfsSubset	IBK	default + 5 NN	128	18447	2626	436	202	69	13.66%	25.46%	0.62
20	over	none	BayesNet	default	2263	850	3059	3	270	1	25.00%	0.37%	0.62

36	none	Ranker	IBK	default + 10 NN	129	7411	3062	0	270	1	100.00%	0.37%	0.62
132	matrix	Ranker	IBK	default + 10 NN	110	5747	1542	1520	82	189	11.06%	69.74%	0.62
30	none	Ranker	NaiveBayes	default	241	688	3035	27	262	9	25.00%	3.32%	0.61
21	over	none	BayesNet	default + TAN	9E+05	1077	3061	1	270	1	50.00%	0.37%	0.61
71	under	none	IBK	default + 5 NN	54	5954	1966	1096	136	135	10.97%	49.82%	0.60
4	over	CfsSubset	SMO	default	6654	748	2834	228	199	72	24.00%	26.57%	0.60
61	under	none	J48	default	874	360	591	2471	1	270	9.85%	99.63%	0.59
97	under	Ranker	J48	default	536	202	591	2471	1	270	9.85%	99.63%	0.59
96	matrix	none	IBK	default + 10 NN	122	31937	1482	1580	88	183	10.38%	67.53%	0.59
66	under	none	NaiveBayes	default	313	1149	2621	441	215	56	11.27%	20.66%	0.59
90	matrix	none	NaiveBayes	default	442	1282	1652	1410	108	163	10.36%	60.15%	0.59
84	over	Ranker	IBK	default + 10 NN	94	17745	2719	343	232	39	10.21%	14.39%	0.59
48	none	none	IBK	default + 10 NN	62	24367	3062	0	271	0	0.00%	0.00%	0.59
125	matrix	Ranker	SMO	default + RFBKernel	3E+06	2E+06	2667	395	189	82	17.19%	30.26%	0.59
72	under	none	IBK	default + 10 NN	62	5910	2390	672	185	86	11.35%	31.73%	0.58
42	none	none	NaiveBayes	default	412	1393	2630	432	203	68	13.60%	25.09%	0.58
24	over	none	IBK	default + 10 NN	166	60372	2691	371	217	54	12.71%	19.93%	0.58
89	matrix	none	SMO	default + RFBKernel	3E+06	2E+06	2691	371	198	73	16.44%	26.94%	0.57
23	over	none	IBK	default + 5 NN	128	62237	2775	287	228	43	13.03%	15.87%	0.57
35	none	Ranker	IBK	default + 5 NN	81	6203	3045	17	267	4	19.05%	1.48%	0.57
131	matrix	Ranker	IBK	default + 5 NN	117	5770	2074	988	149	122	10.99%	45.02%	0.56
18	over	none	NaiveBayes	default	630	1307	2114	948	153	118	11.07%	43.54%	0.56
83	over	Ranker	IBK	default + 5 NN	136	16627	2774	288	242	29	9.15%	10.70%	0.56
78	over	Ranker	NaiveBayes	default	393	855	2484	578	191	80	12.16%	29.52%	0.56
47	none	none	IBK	default + 5 NN	62	26148	3053	9	269	2	18.18%	0.74%	0.55
95	matrix	none	IBK	default + 5 NN	89	32083	2127	935	159	112	10.70%	41.33%	0.55
141	none	CfsSubset	BayesNet	default + TAN	669	473	3062	0	271	0	0.00%	0.00%	0.55
117	matrix	CfsSubset	BayesNet	default + TAN	775	518	2952	110	260	11	9.09%	4.06%	0.55
5	over	CfsSubset	SMO	default + RFBKernel	60538	11510	2879	183	232	39	17.57%	14.39%	0.54

70	under	none	IBK	default	79	6221	1798	1264	137	134	9.59%	49.45%	0.54
10	over	CfsSubset	IBK	default	132	15452	2664	398	215	56	12.33%	20.66%	0.54
106	under	Ranker	IBK	default	60	1833	1826	1236	143	128	9.38%	47.23%	0.53
40	none	none	SMO	default	5E+05	16968	2979	83	246	25	23.15%	9.23%	0.53
28	none	Ranker	SMO	default	5E+05	19958	2987	75	247	24	24.24%	8.86%	0.53
124	matrix	Ranker	SMO	default	5E+05	19621	2985	77	249	22	22.22%	8.12%	0.53
88	matrix	none	SMO	default	5E+05	17881	2984	78	249	22	22.00%	8.12%	0.53
16	over	none	SMO	default	8E+05	16752	2986	76	252	19	20.00%	7.01%	0.52
65	under	none	SMO	default + RFBKernel	32267	1E+06	2902	160	247	24	13.04%	8.86%	0.52
22	over	none	IBK	default	182	65645	2776	286	238	33	10.34%	12.18%	0.51
101	under	Ranker	SMO	default + RFBKernel	31176	1E+06	2967	95	256	15	13.64%	5.54%	0.51
73	over	Ranker	J48	default	1886	276	2640	422	226	45	9.64%	16.61%	0.51
82	over	Ranker	IBK	default	119	12820	2749	313	239	32	9.28%	11.81%	0.51
76	over	Ranker	SMO	default	1E+06	20750	2976	86	260	11	11.34%	4.06%	0.51
46	none	none	IBK	default	60	31073	2844	218	249	22	9.17%	8.12%	0.50
94	matrix	none	IBK	default	119	27982	2844	218	249	22	9.17%	8.12%	0.50
34	none	Ranker	IBK	default	88	4539	2804	258	245	26	9.15%	9.59%	0.50
130	matrix	Ranker	IBK	default	100	4896	2802	260	245	26	9.09%	9.59%	0.50
136	none	CfsSubset	SMO	default	86437	1998	3004	58	265	6	9.38%	2.21%	0.50
112	matrix	CfsSubset	SMO	default	88351	2593	2942	120	260	11	8.40%	4.06%	0.50
25	none	Ranker	J48	default	2588	245	3062	0	271	0	0.00%	0.00%	0.50
37	none	none	J48	default	2349	311	3062	0	271	0	0.00%	0.00%	0.50
133	none	CfsSubset	J48	default	734	277	3062	0	271	0	0.00%	0.00%	0.50
134	none	CfsSubset	J48	default + binarySplit + 5 leaves	279	238	3062	0	271	0	0.00%	0.00%	0.50
135	none	CfsSubset	J48	default + binary + 5 leaves + reduced	300	280	3062	0	271	0	0.00%	0.00%	0.50
53	under	CfsSubset	SMO	default + RFBKernel	13287	552164	3062	0	271	0	0.00%	0.00%	0.50
29	none	Ranker	SMO	default + RFBKernel	7E+05	2E+06	3062	0	271	0	0.00%	0.00%	0.50
41	none	none	SMO	default + RFBKernel	7E+05	2E+06	3062	0	271	0	0.00%	0.00%	0.50
137	none	CfsSubset	SMO	default + RFBKernel	1E+05	298544	3062	0	271	0	0.00%	0.00%	0.50

113	matrix	CfsSubset	SMO	default + RFBKernel	1E+06	524284	3062	0	271	0	0.00%	0.00%	0.50
17	over	none	SMO	default + RFBKernel	5E+06	1E+06	3062	0	271	0	0.00%	0.00%	0.50
77	over	Ranker	SMO	default + RFBKernel	4E+06	1E+06	3061	1	271	0	0.00%	0.00%	0.50

7.4 All Boosted Models Evaluation Results

Num	ReSample	Selector	Classifier	Options	**Training Time**	**Test Time**	True Negative	False Positive	False Negative	True Positive	Precision	Recall	**AUC**
3	under	CfsSubSet	Bagging	NaiveBayes -D	525	471	2377	685	73	198	22.42%	73.06%	0.86
9	under	CfsSubSet	Vote	all models	789	1101	1989	1073	43	228	17.52%	84.13%	0.85
6	under	CfsSubSet	AdaBoostM1	J48 -B -M 5	1085	218	2413	649	77	194	23.01%	71.59%	0.85
2	under	CfsSubSet	Bagging	J48 -B -M 5	968	251	2320	742	79	192	20.56%	70.85%	0.85
4	under	CfsSubSet	Bagging	BayesNet	1108	682	1606	1456	19	252	14.75%	92.99%	0.84
1	under	CfsSubSet	Bagging	J48	706	273	2983	79	144	127	61.65%	46.86%	0.84
10	under	Ranker	Bagging	J48	3009	1068	1535	1527	25	246	13.87%	90.77%	0.84
11	under	Ranker	Bagging	J48 -B -M 5	7178	323	2326	736	78	193	20.78%	71.22%	0.83
18	under	Ranker	Vote	all models	2046	543	2208	854	97	174	16.93%	64.21%	0.78
15	under	Ranker	AdaBoostM1	J48 -B -M 5	5010	102	2290	772	95	176	18.57%	64.94%	0.78
7	under	CfsSubSet	AdaBoostM1	NaiveBayes -D	1320	261	1485	1577	51	220	12.24%	81.18%	0.78
14	under	Ranker	AdaBoostM1	J48	2882	286	2989	73	145	126	63.32%	46.49%	0.77
8	under	CfsSubSet	AdaBoostM1	BayesNet	2224	1157	1778	1284	40	231	15.25%	85.24%	0.77
12	under	Ranker	Bagging	NaiveBayes -D	1173	2533	2166	896	91	180	16.73%	66.42%	0.76
5	under	CfsSubSet	AdaBoostM1	J48	1274	731	2195	867	70	201	18.82%	74.17%	0.75
13	under	Ranker	Bagging	BayesNet	2446	1589	2089	973	96	175	15.24%	64.58%	0.73
16	under	Ranker	AdaBoostM1	NaiveBayes -D	1773	1091	2060	1002	91	180	15.23%	66.42%	0.71
17	under	Ranker	AdaBoostM1	BayesNet	2375	727	1997	1065	104	167	13.56%	61.62%	0.68