Rashmi Mariyappa
Data Mining for Business Class

Assignment 1: Decision Tree Analysis

Question 1: Explore the Data

Missing Values
Once the data set was imported into Rapidminer, we checked for missing values. There were seven variables with missing values: NEW_CAR, USED_CAR, FURNITURE, RADIO/TV, EDUCATION, RETRAINING, and AGE. Of these seven variables six, with the exception of AGE, are used to determine the purpose of the loan, with a value of 1 meaning 'yes' this is what the loan was for and a value of 0 meaning 'no' this is not what the loan was used for. Rather than being coded as 0, the 'no' values for these variables loaded as missing. Therefore, the missing values for these variables were converted to 0. Since the AGE attribute represented continuous numerical values, the decision on what to replace the missing values with was not as straightforward. Considering there were only 8 missing values for this attribute, we replaced them with the average.

Data Questions
Upon examining the data several oddities gave rise to further questions. First, in the original data set as compared to the transformed data set, many of the categorical variables were divided out into single binary variables. Should we combine these variable into a single categorical variable and would this make the analysis more accurate? Additionally, in the original data set personal status and sex were one categorical variable. The working data set has divided this out into 3 variables and only to classify males. This leaves all women in the data set as negative space and not directly coded in. Would it be better to code all sex/personal status categories back for both males and females? Next, in the variable PRESENT_RESIDENT the coding key has the values of 0, 1, 2, 3, however, the actual data uses 1, 2, 3, 4. Is this simply an error in the key offset by one, or is it a larger error that might affect the results? The values for the variables NUM_DEPENDENTS and INSTALL_RATE do not seem correct. All the values for number of dependents were either 1 or 2. It is difficult to believe that dependents fall entirely in this range. For example, a married couple with 3 kids would list 4 dependents. Also, a single person might not be supporting anyone so the minimum value should be 0. Finally, the values in INSTALL_RATE are all integers between 0 and 4. It seems like values would vary more and be decimals rather than integers.

Data Exploration
There were 1000 observations for this data set. Seventy percent of the cases were classified as "good" credit and thirty percent as "bad." The mean, standard deviations, and frequencies for the individual attributes were examined. The next method that was used to explore the data were various plots and charts. We started by examining each attribute against the RESPONSE variable credit rating using scatterplots. With visual inspection of the scatterplots it proved difficult to determine any clear relationships because the size of the points did not get bigger with an increased frequency. Even when turning the 'jitter' setting up, it was still too difficult to easily determine the frequencies in major groupings. This prompted questions such as, "Is this a 55% - 45% split or 70% - 30%?" The next charts we used were color histograms, where the frequency of each variable was broken out by the response variable. This chart proved to be more useful in that we could estimate whether the frequency of the answers for each attribute against the response variable fell approximately along the overall split of the

data, 70% - 30%, or whether it deviated. Based on this examination, the variables that seem to play a role in determining credit rating were: CHK_ACCT, SAV_ACCT, HISTORY, AMOUNT, EMPLOYMENT, RENT, OWN_RES, REAL_ESTATE, PROP_UNKN_NONE, OTHER_INSTALL, CO-APPLICANT, GUARANTOR, USED_CAR, RADIO/TV, EDUCATION.

CHK_ACCT seemed to be the attribute that most divided the sample into “good” and “bad” cases. However, the breakout was not completely as one would expect. The examination showed that the more money one had in their checking account the less likely they were to be a “bad” case, which would be expected. However, the surprising finding was that people who had no checking account were more likely to be “good” credit cases. This finding appeared to be counter-intuitive and contradicted the positive link between amount in the checking account and credit rating. Does ‘no checking account’ mean strictly at this bank or at any bank? SAV_ACCT matched the findings of CHK_ACCT, with a positive link between amount in the account and “good” credit.

HISTORY also showed curious results in that people who paid back all credit at the bank duly were more likely to be “bad” credit cases and ‘critical accounts’ were more likely to be “good” cases. The question arises then, What exactly does “critical” mean in the coding? Does it mean in critical condition/bad shape or does it mean critical/very important? The more understandable finding was that people with ‘no credit taken’ were more likely to be “bad” credit cases.

AMOUNT, OTHER_INSTALL and EMPLOYMENT showed results as one might expect. As the amount of the loan increased so too did the chance that the credit rating was bad. Also, if the person had other installment plans, they were more likely to be a “bad” credit case. Additionally, a longer employment history at the applicant’s present job corresponded with a greater likelihood that his or her credit rating was “good”.

RENT, OWN_RES, REAL_ESTATE, PROP_UNKN_NONE all related to whether the person owned property or rented. All showed a similar relationship to credit rating. If the person owned some type of property he or she was more likely to have a “good” credit rating compared to renters who were more likely to have a “bad” rating.

USED_CAR, RADIO/TV, and EDUCATION were all variables used to classify the purpose of the loan. These three purposes seemed to have a small linkage to whether the credit was “good” or “bad.” Loans for used cars and radio/tvs were “good”. Surprisingly, loans for education were more likely to be “bad.”

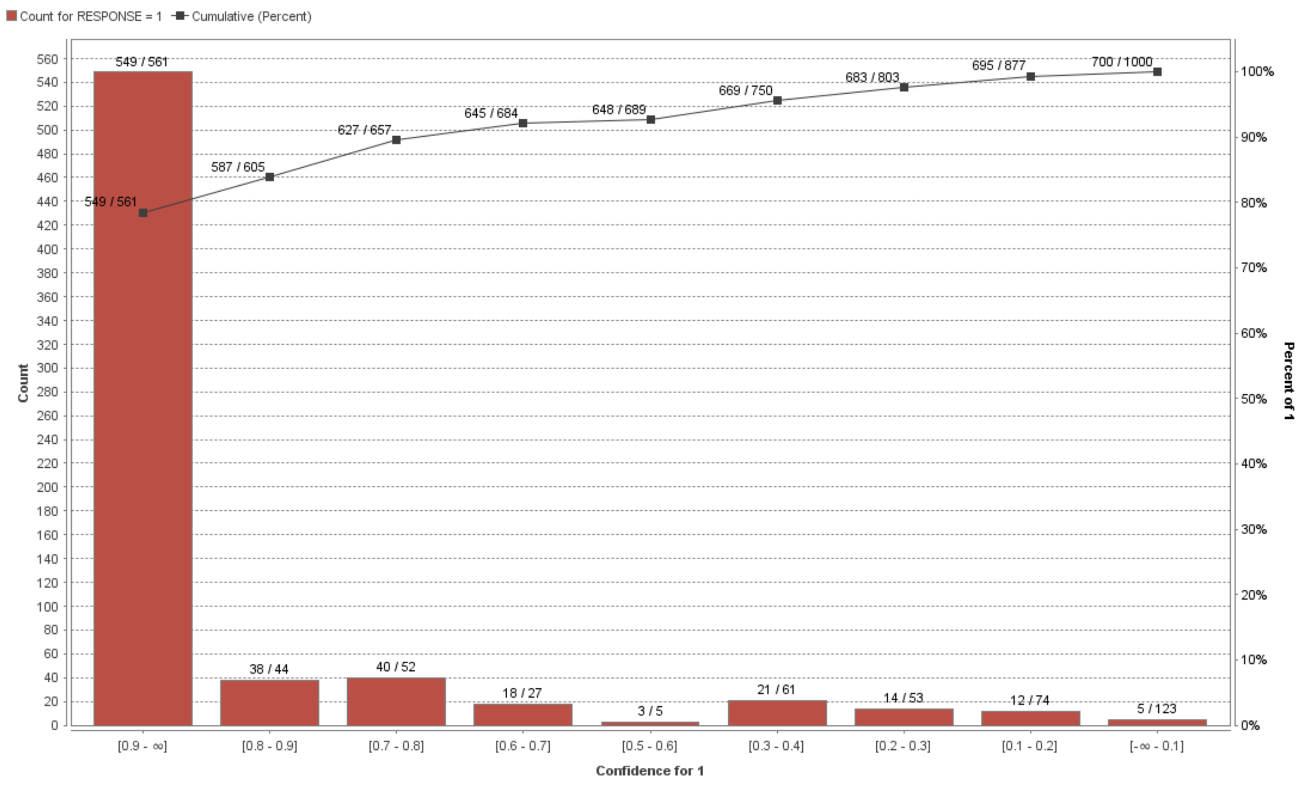
Question 2: Descriptive Model

In the first task, we focused on creating a descriptive model using the full data set. The decision tree parameters we used to get a good model were: gini index, no maximal depth restriction, default pruning confidence of .25, no prepruning minimal gain, a minimal leaf size of 3, and a minimal size for split of 6. These settings were selected because they provided the best tree determined by overall accuracy in the 90% range, as well as high recall, specificity, precision, and negative predictive value – all in the 80% - 90% range, while keeping the leaf size and minimal size for split as large as possible. Although having smaller leaf and split size could lead to overfit, these sizes were still chosen for the reasons above because we have no way of telling what models are overfit without a test data set.

The variable that was most important in differentiating “good” from “bad” credit cases was CHK_ACCT. It is the root node upon which the tree first branches. This was the case in most of the experimental decision trees created while exploring different tree criteria. These findings matched the expectations of the findings from exploring the data using the color histograms, where CHK_ACCT was determined to be the most influential variable in that examination as well. Depending on the branch of the split on CHK_ACCT, the tree next split on HISTORY, AMOUNT, REAL_ESTATE, and OTHER_INSTALL. Therefore, these attributes were the next most important variables in determining credit rating in the subgroups. All these variables were also identified as important in the color histogram exploration of the data. However, the decision tree could identify subgroups that were important. For example, for people who have checking accounts with 0 - 200 DM the next most important attribute to examine is the amount of the loan. The color histogram analysis was only on the overall data set.

The overall level of accuracy of the model was 90.70% with a 9.3% error rate. The recall for the “good” case was 92.57% or 648 out of 700 cases predicted correctly. The specificity for the “bad” case was 86.33% or 259 out of 300 cases predicted correctly. Below is the lift chart for the selected model.

Lift Chart for Descriptive Model



This chart shows that for the positive class of “good” credit the most probable cases in the top confidence score bracket, 0.9 - 1, identify the majority of the positive cases, 549/700. All the other confidence brackets ranging from 0 - 0.8 add a much smaller count for positive cases, which is why the cumulative lift curve levels off past the 0.9 - 1 confidence score bracket.

This is not a reliable and robust model due to the fact that the model is never tested. All the data goes into the training set. Without being able to test the model on unseen data, we can never know how good this model is. It is likely that the high accuracy is simply due to overfit.

Question 2: Predictive Model

Next, we developed a model for prediction using the (standard) decision tree operator in Rapidminer, dividing the data set into 50% - 50% partitions for training and testing. The parameters used for the descriptive model above were tested and it proved to be an overfitted model as the training accuracy was high but the test accuracy fell sharply. After experimentation, the model parameters that most effected performance were increasing the leaf size and the minimum number of cases for a spilt. This prevented the model from overfitting. We increased the minimum leaf size to 10 and minimal size for split to 20. We also changed parameters around pruning. Changing the confidence level for pruning from the default 0.25 did not seem to improve the model. However, adding a minimal gain of 0.05 in the prepruning criteria improved the model by adding a minimum requirement that must be gained by the split and increased overall accuracy of the model.

We selected a model that had 73.00% accuracy and a 27.00% error rate on the validation data. The performance values we considered most when selecting the model were a combination of accuracy, recall, and specificity. While we are interested in the positive case – identifying "good" credit applications, we cannot ignore the performance on the negative case – identifying "bad" credit applications. Improperly identified negative cases are costly due to the default on the loan. Therefore, we tried to balance as high of a recall as possible while keeping the specificity high as well. It was a tradeoff between the two. Additionally, accuracy, the AUC figures, and the ROC curve were considered to help make the decision on best performance. We also examined lift charts to see how the most probable "good" cases based on confidence scores assigned by the model were classified correctly.

<u>Different Decision Tree Operators</u>

Next, we considered two additional decision tree operators: J48 and CART. The best model for each operator was obtained. See the chart on the next page for a summary of the parameters and performance of the selected decision tree from each type of operator.

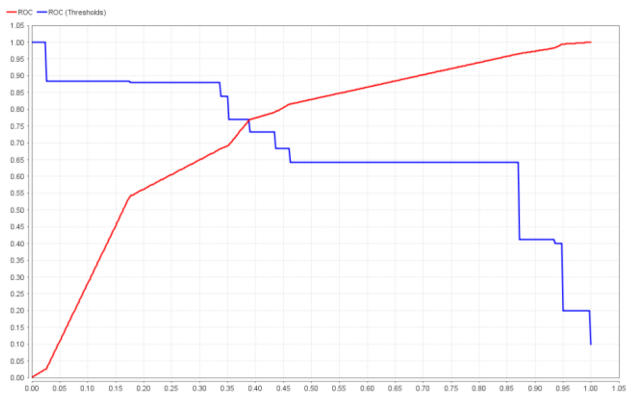
The performance across all three decision tree operators was similar for the best model selected from each learner. All had accuracy ranging from 72% - 73%. Recall for the different models ranged from 81% - 82%. Similarly, the specificity, which we were trying to get high in order to properly identify the "bad" credit applicants, ranged from 50% - 53%. Precision and negative predictive value were also kept in mind and had similar values across the three.

Decision Tree Model Comparison

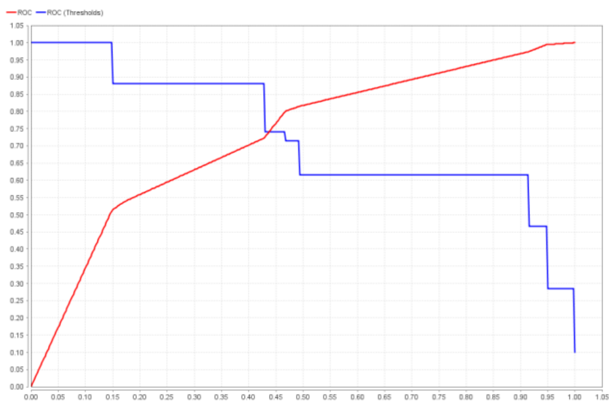
		Decision Tree (Standard)	J48	CART
Parameters	**Criterion**	gini index	N/A	N/A
	Max Depth	None	N/A	N/A
	Confidence Threshold for Pruning	0.25	0.25	N/A
	Minimal Gain	0.05	N/A	N/A
	Minimal Leaf Size	10	8	2
	Minimal Size for Split	20	N/A	N/A
	Other Method Specific	# of Prepruning Alternatives: 30	Used Reduced Error Pruning with 11 folds	Number of Folds for Cost-Complexity Pruning: 7.0
Performance	**Overall Accuracy**	73.00%	72.00%	72.80%
	Sensitivity / Recall	81.50%	81.50%	82.61%
	Specificity	53.90%	50.65%	50.97%
	Precision	79.89%	78.77%	78.95%
	Negative Predictive Value	56.46%	54.93%	56.83%
	ROC: AUC – Value Listed	0.728	0.724	.663

The ROC curves were similar, see below, for the standard decision tree and J48, and they also had a similar AUC value of 0.728 and 0.724, respectively.

ROC Curve – Decision Tree (Standard)

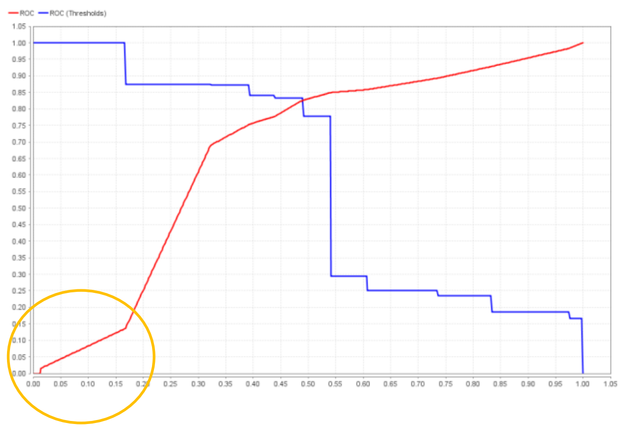


ROC Curve – J48



The ROC curve for the CART model, see below, differed slightly in that in the beginning of the curve the true positive rate to false positive rate was lower signaling a worse performance in that portion of the model. The area is marked by the yellow circle in the graph. This is reflected with a lower AUC of 0.663 compared to the standard decision tree and J48 models.

ROC Curve – CART



The lift charts across the three models were similar. The majority of good cases were identified with high confidence scores for good credit and then steeply dropped off. The exact confidence thresholds that the drop occurred at varied across the three models: 0.9 for standard decision tree, 0.874 for J48, and 0.87 for CART.

All three decision tree models have CHK_ACCT as the root node. The standard decision tree and J48 models are very similar. Both have a depth of 5 and only differ in the number of leaf nodes by 2, 10 to 8 respectively. Additionally, both models split on DURATION if CHK_ACCT equals 0 and AMOUNT if CHK_ACCT equals 1. Both models also do not further branch on CHK_ACCT if they equal 2 or 3. The model obtained using CART is different, however. It is much larger having a depth of 8 and 20 leaf nodes. The CART model is able to combine categorical values. As a result, the split on CHK_ACCT is into two branches for CART: equal to 2 or 3, not equal to 2 or 3. The CART model, similarly to standard decision tree and J48, branches on AMOUNT and DURATION in further splits. Although there are so many additional branches in this model, we would not call it very similar to the standard decision tree and J48 models.

Changing the Random Seed

To check if our models were stable we changed the random seed in the split data operator for each model to various randomly selected numbers. This provided different samples for training and test data upon which to run the model. We were surprised to see that the models changed dramatically upon changing the random seed for all three decision tree models. Not only did the size of the tree and the attributes that were split on change, but the performance measures also did. The accuracy, recall, specificity, ROC curve, AUC, and lift chart all varied as well. Therefore we determined that our models, and decision tree models in general, are unstable.

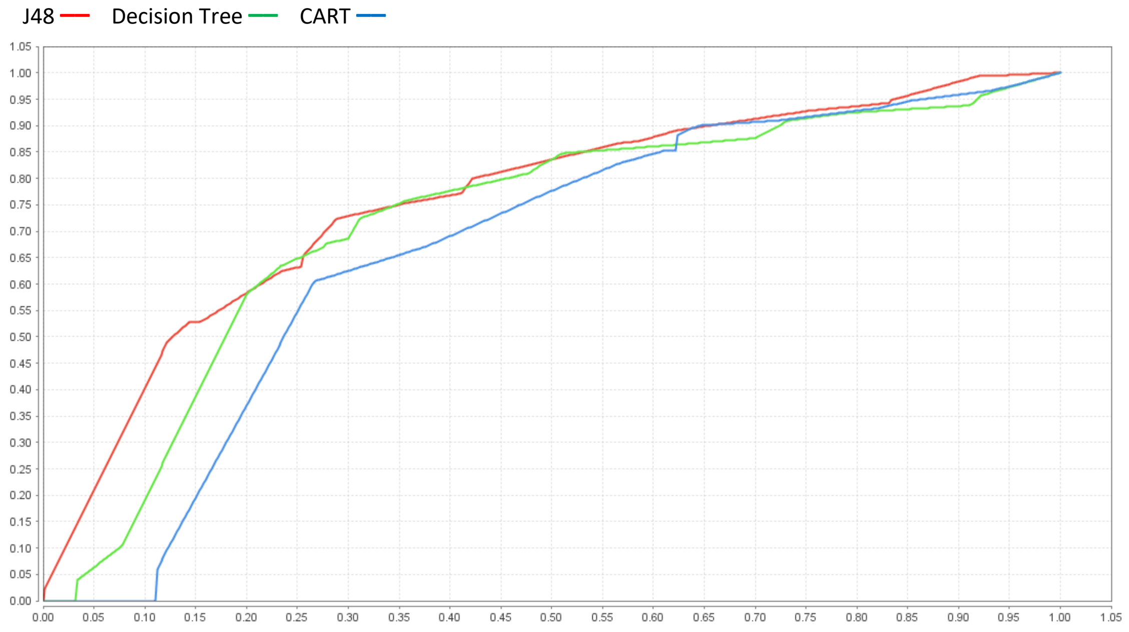
Changing the Training and Test Partitions

Upon changing the split between training and test data from 50% - 50% to 70% - 30% and 80% - 20%, we once again noticed that the model and performance changed across all three decision tree models. Regarding the models, some of the attributes that were split on changed based on different training - testing ratios. However, the root node consistently stayed CHK_ACCT across all three models and on all three divisions of training and test data. This indicates that checking account is the most important variable in dividing between “good” and “bad” cases across the entire data set. The accuracy, recall, sensitivity, ROC curve, AUC, and lift chart all varied as well when changing the training - test ratio. Sometimes this variation was by just a couple percent in accuracy, but in other cases the variation was more significant, such as a 20% drop in specificity. Additionally, the same parameters did not necessarily provide the best models when applied across the different divisions of training - test data. Once the division of training and test data is altered, we can tweak the parameters to create a model that performs in a different way that we might consider better. For example, in the standard decision tree operator we could increase the specificity from 53.90% (in the 50% - 50% training - test ratio) to 62.37% (in the 70% - 30% training - test ratio) simply by increasing the minimal leaf size from 10 to 15.

Selecting a Model for Implementation

Given what we learned in question 2, that decision tree models are unstable and slight changes in the data set can have profound impacts, it is difficult to strongly believe in any of the models. However, since we must pick a model, it would be best to pick a model based on the performance on the 70% - 30% split in training - test data. We tested our top three models from all three decision tree operators, listed above in the chart on page 5, on the 70% - 30% data split. Below is the chart comparing the ROC curves from all three decision tree models. The J48 decision tree performed the best on the 70% - 30% training - test data division based not only on the ROC curve, as can be seen below, but also across other measures such as accuracy, recall, and specificity.

ROC Curve 3 Model Comparison



We checked if we could achieve better performance on the original J48 model with the 70% - 30% split in training - test data by changing parameters. In this case, the original parameters worked the best. Therefore, this is the model we chose for implementation.

Question 3: Misclassification Costs

Next, we used the costs for misclassification to assess the performance of the selected J48 model. We started with the standard confidence threshold of 0.5, which had the average misclassification cost as 94.33 DM. We then moved the threshold up by 0.05 until we reached 1.0. The results of this analysis are summarized in the chart below.

Average Misclassification Costs by Increasing Confidence Threshold

Confidence Threshold	**.50**	**.55**	**.60**	**.65**	**.70**	**.75**	**.80**	**.85**	**.90**	**.95**	**1.0**
Average Misclassification Costs (DM)	94.33	80.00	80.00	80.00	77.67	62.67	61.33	54.67	69.00	69.00	70.00

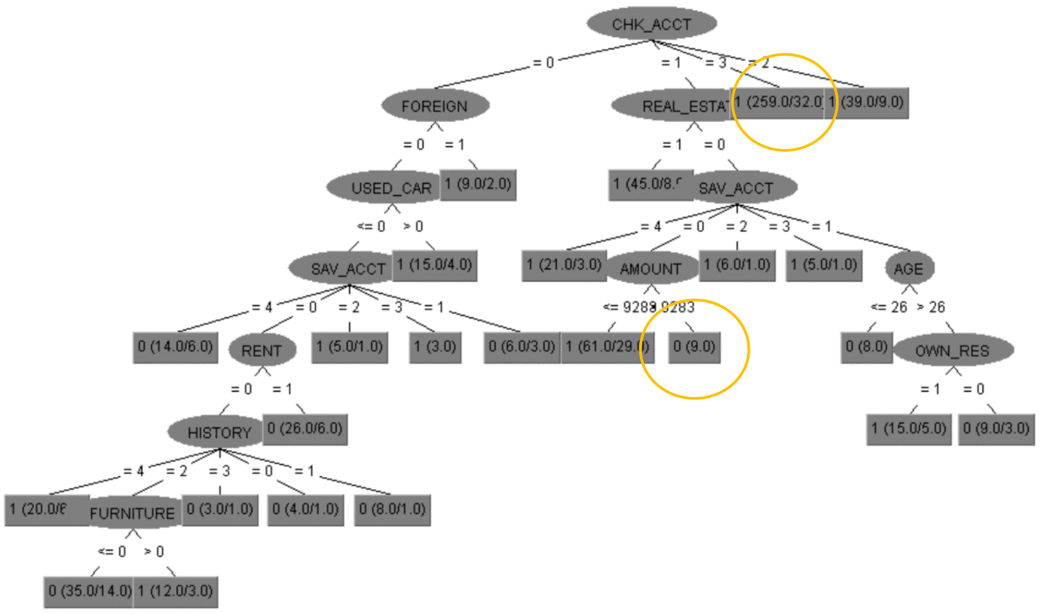
We can see from the chart that the average misclassification costs are minimized when the confidence threshold is set around 0.85. Since the cost for misclassifying a "bad" credit case as "good" is much higher (500 DM) than the opportunity cost of misclassifying a "good" credit case as "bad" (100 DM) it is logical that to minimize misclassification costs we would want to set the confidence level much higher for predicting a case as "good" credit. This minimizes risk and therefore cost. If there were different misclassification costs then the confidence level that we would set for predicting a "good" credit case would change. For example, if the costs were reversed, 500 DM for misclassifying a "good" credit case as "bad" and 100 DM for misclassifying a "bad" credit case a "good," the confidence level for classifying a "good" credit case would be much lower.

Question 4: "Best" Decision Tree Model

For our "best" decision tree model we used the J48 operator with the same parameters as listed in the chart in question 2. The tree had a depth of 8, with 12 nodes, and 24 leaves. The variable at the top of the tree, the root node was CHK_ACCT. This was as we predicted in question 1. The majority of the experimental decision trees had CHK_ACCT as the root node. We can be confident with this consistency that the checking account value plays a very important role in predicting whether someone will have "good" or "bad" credit. The other variables near the top of the tree were also ones identified in question 1: REAL_ESTATE, SAVE_ACCT, USED_CAR. The variable FOREIGN was not identified in question 1, but some of the experimental trees that we created branched on this variable. A picture of our best decision tree is shown on the next page.

J48 – Decision Tree

(70% - 30% Training to Test Partitions)



<u>Pure Leaf Nodes</u>

Two examples of relatively pure leaf nodes are as follows. One is near the top of the tree, circled in yellow. It is the leaf identified by: IF (CHK_ACCT = 3) THEN (RESPONSE = 1, “Good” Credit). This leaf node has 291 cases of which 259 are “good” credit and 32 are “bad” credit. Therefore, the probability of “good” credit in this leaf is 89% and the probability for “bad” is 11%. There is a leaf node that is pure near the middle of the tree, circled in yellow. It is the leaf identified by: IF (CHK_ACCT = 1) AND (REAL_ESTATE = 0) AND (SAV_ACCT = 0) AND (AMOUNT > 9293) THEN (RESPONSE = 0, “Bad” Credit). This leaf node has 9 cases of which all 9 are “bad” credit cases. Therefore, the probability of “good” credit in this leaf is 0% and the probability for “bad” 100%.

<u>Decision Tree Rules</u>

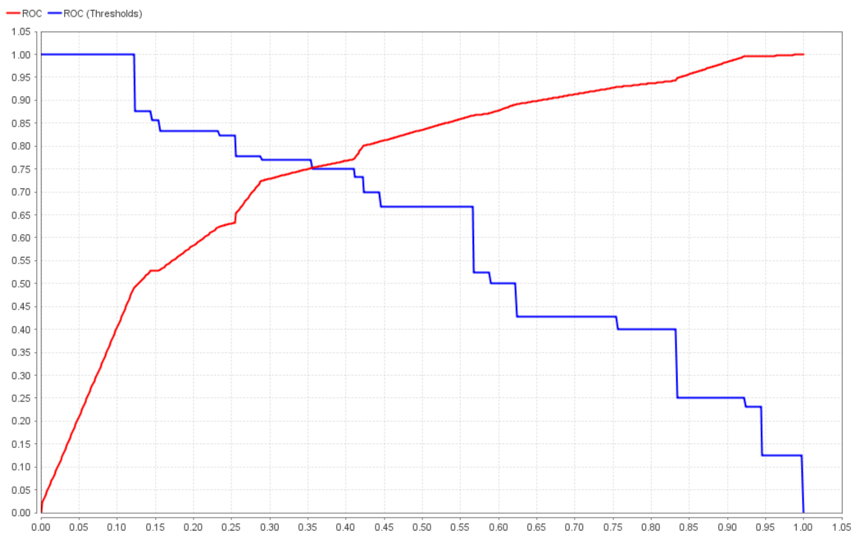
Two rules in addition to the two listed in the section above are as follows. We picked one shorter rule and one that is lengthier.

1. IF (CHK_ACCOUNT = 2) THEN (RESPONSE = 1, “Good” Credit)
2. IF (CHK_ACCT = 1) AND (REAL_ESTATE = 0) AND (SAV_ACCT = 1) AND (AGE > 26) AND (OWN_RES = 1) THEN (RESPONSE = 1, “Good” Credit)

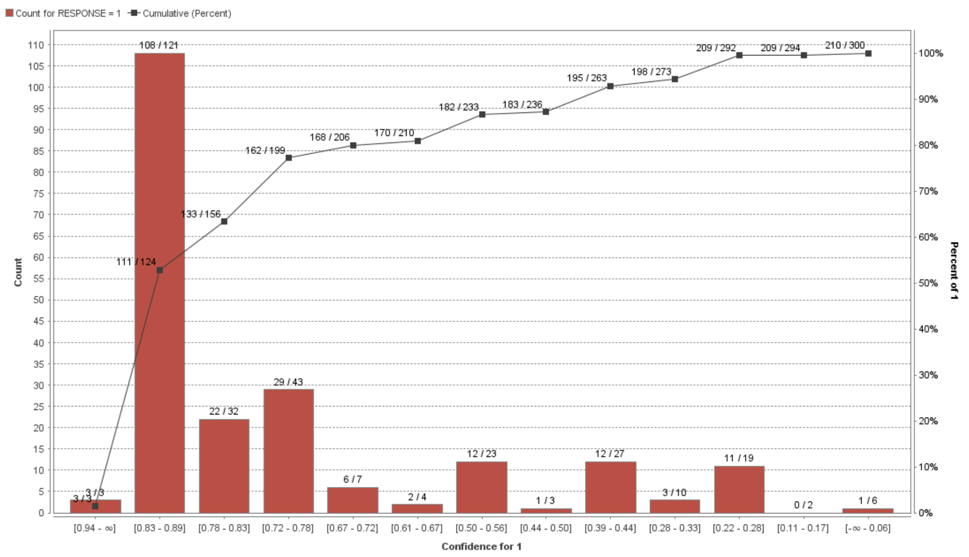
<u>Performance and Overall Summary of Selected Model</u>

The J48 decision tree that we selected to implement on the data with the training - test split of 70% - 30% had an AUC of 0.757. Which was higher than the AUC of the models listed in question two. The ROC curve showed the model having a steep true positive rate to false positive rate in beg section of the curve. The lift curve shows that most of the “good” cases are identified with the higher confidence predictions for “good” cases in the 0.83 – 0.89 bracket. In this bracket 108 out of 121 cases are correctly identified. There is a steep drop off past this confidence bracket, with the next bracket from 0.78 – 0.83 only identifying 22 out of 32 cases for “good” credit. The ROC and lift curves for the selected model are shown on the next page.

ROC Curve – J48 Selected Model



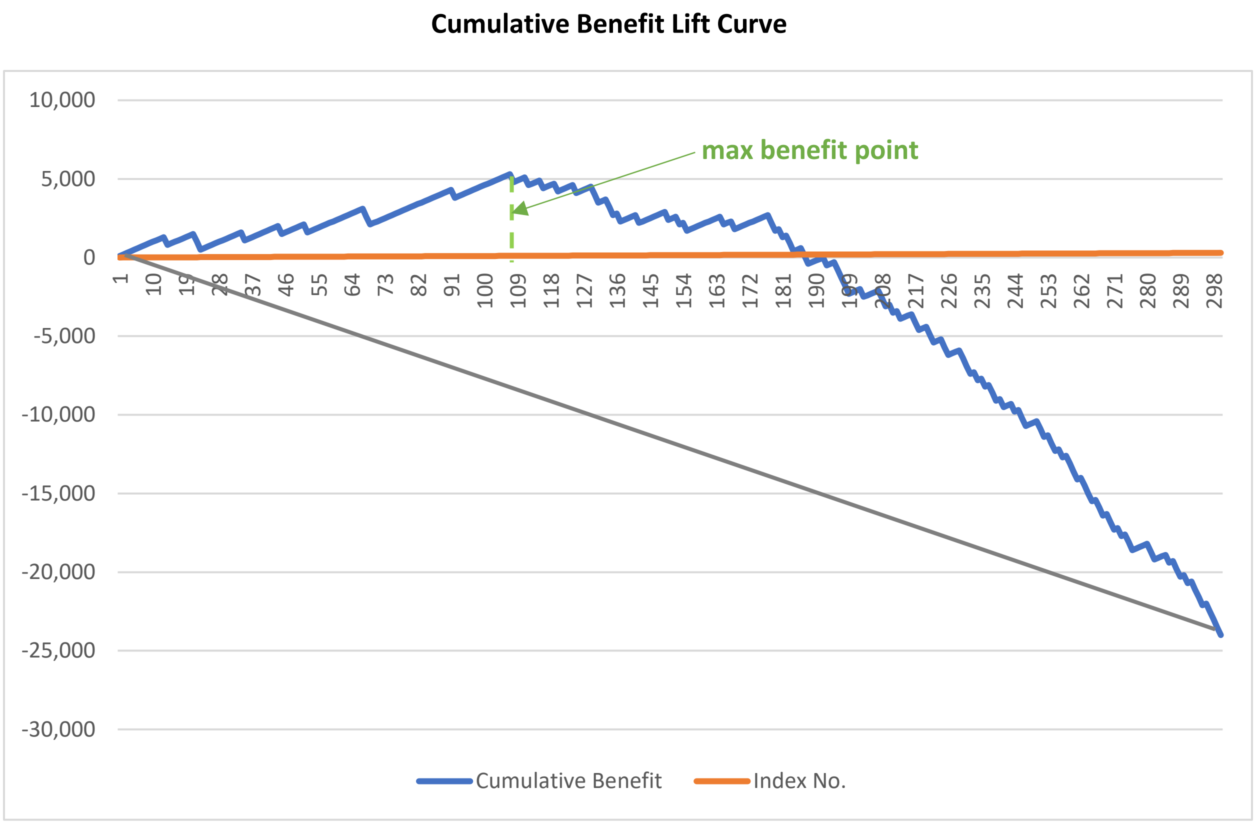
Lift Curve – J48 Selected Model



The confusion matrix changes based on the confidence threshold set for predicting the good credit rating. The lowest average cost for misclassification is 54.00 DM and it occurs when the confidence threshold for predicting good credit is set to 0.87644. This matches the cost/benefit analysis in question 5. Based on this threshold the accuracy drops to 60.67% (from 73.67% based on the 0.5 confidence threshold) and recall drops to 49.05% (from 86.67% based on the 0.5 confidence threshold). However, the specificity increased to 87.78% % (from 43.33% based on the 0.5 confidence threshold). This makes sense since the misclassification costs for a “bad” credit case as “good” is much higher than the reverse. Therefore, once costs are added into the analysis it is preferable, more profitable, to have more of the “bad” cases predicted correctly even if it is at the expense of an increased amount of “good” cases predicted incorrectly. Overall the tree had some large (approx. 300 cases) and some very small (approx. 4 cases) leaf nodes. It would have been preferable to have more even sized leaf nodes and larger minimum leaf sizes. This might have made the tree model more stable. However, the set of parameters used, obtained the best performing tree based on the factors described above, ROC, AUC, lift, and accuracy.

Question 5: Cost/Benefit Analysis

Finally, we did a cumulative cost / benefit analysis on the data to determine a cutoff value to obtain the maximum profit. The validation data was sorted by the confidence scores for predicting a "good" credit case. The cost and benefit was added to each case. "Good" models sorted in this matter should divide out the actual "good" credit cases at the top (with high confidence scores) and "bad" cases near the bottom (with low confidence scores). Therefore, the cumulative benefit for "good" models should pool at some point, a max, before declining when the "bad" credit cases cost start to get added in more heavily. The graph of the cumulative benefit lift curve is shown below.



The maximum cumulative profit is at index point 107 with a value of 5,300 DM. The confidence level for the "good" credit rating at this point is 0.87645. This is the cutoff value for predicted probability that we would recommend. However, at this predicted probability, we must look at the cumulative profit value for the largest indexed value with this probability. The index number for this case is 114 which corresponds to a cumulative profit value of 4,800 DM. To check to make sure this is the best value we look at the confidence scores immediately higher and lower than the selection. The confidence score higher is 1.00000 with a cumulative profit of 300 DM. We can rule this out as the profit is lower than the 4,800 DM that our selection yielded. The immediately lower confidence score is 0.85714 with a cumulative profit of 4,600 DM. This value can also be ruled out as the profit is again lower compared to our selection. An abbreviated list of the sorted values is shown on the next page to illustrate the above described cutoff. Please note that some index values are hidden so that the list could be condensed.

Abbreviated Cumulative Benefit by Confidence Score of 1 List

Confidence(1)	Cost/Benefit	Cumulative Benefit	Index No.	
1.00000	$100	$100	1	
1.00000	$100	$200	2	
1.00000	$100	$300	3	higher confidence value check
0.87645	$100	$400	4	
0.87645	$100	$4,900	103	
0.87645	$100	$5,200	106	
0.87645	$100	$5,300	107	highest cumulative profit reached
0.87645	-$500	$4,800	108	
0.87645	$100	$4,700	113	
0.87645	$100	$4,800	114	*confidence value cutoff and corresponding cumulative profit
0.85714	$100	$4,900	115	
0.85714	$100	$4,500	123	
0.85714	$100	$4,600	124	lower confidence value check
0.83333	-$500	$4,100	125	

Additional Questions: Random Forest and AdaBoost

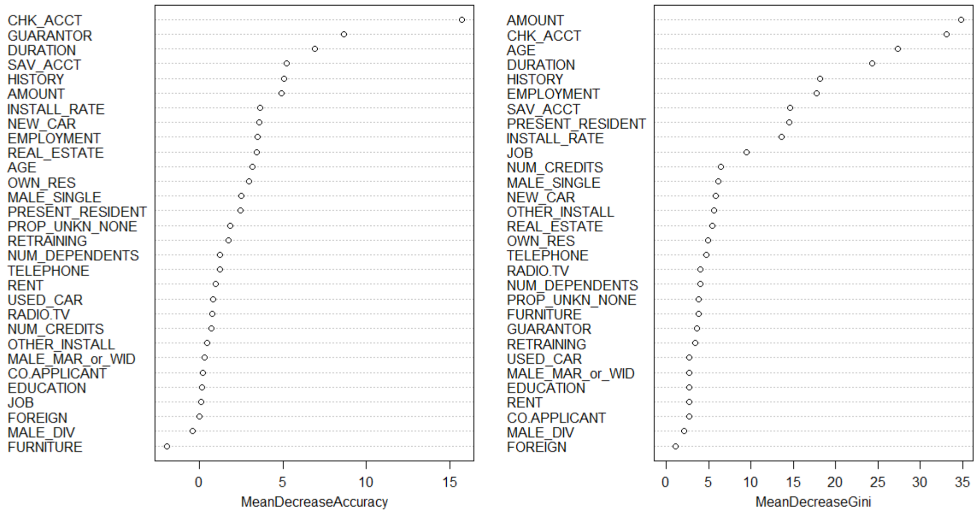
Next, we developed two additional decision tree models using random forest and AdaBoost.

Random Forest

There are two decision tree operators in Rapidminer for random forest, one in the base package – Random Forest, and the other in the Weka extension – W-Random Forest. We experimented with parameters on both operators to develop the best performing model for each type. We used the 70% - 30% training - test data split to develop our model. The parameters for the best W-Random Forest model were: 1500 as the number of trees and 6 of the attributes to consider for each tree. The parameters for the best Random Forest model were: 1500 as the number of trees, gini index, no max depth, confidence threshold of 0.25 for pruning, 0.01 minimal gain for pre-pruning, minimal leaf 6, minimum size for split 12, and the confidence voting strategy.

Additionally, we used R to create a random forest model on the data. While we did not select the model for this comparison study, one interesting feature was to use the model for variable importance. Below is a chart that plots the top variables by mean decrease in accuracy on the left and mean decrease in gini on the right. Similar to what we found in the exploratory data analysis, CHK_ACCT seems to be the top variable.

Top Variables Affecting Credit Rating – Using R Random Forest



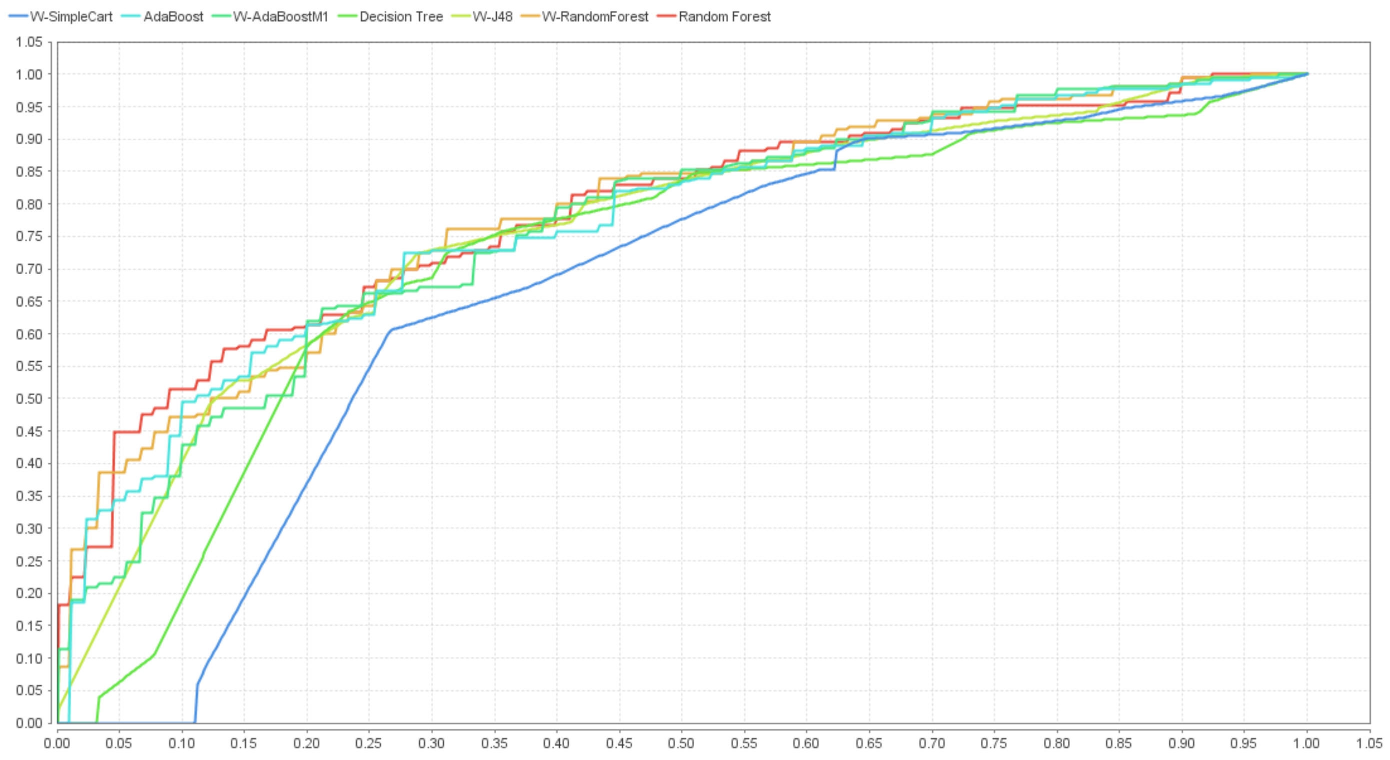
AdaBoost

Once again there are two decision tree operators for AdaBoost, one in the base package – AdaBoost. and the other in the Weka extension – W-AdaBoostM1. We decided to boost our top model from the original three trees selecting the top performer, J48. We used the original parameters for the J48 model and experimented with boosting parameters which is the number of iterations for boosting. For both models we used 18 iterations. While convention generally says to use 100 or more iterations, 18 iterations produced the top AUC values. In the AdaBoost model, iterations above 18 actually produced lower AUC values. In the W-AdaBoostM1 model, iterations above 18 leveled off and produced the same AUC values as 18 iterations.

Comparing Performance

We used the ROC curve and AUC as the main method for selection. This was because performance measures such as accuracy depend on the threshold set for the confidence score to classify a 1. In order to compare the ROC curves, we plotted them for all seven trees, the original three (Standard Decision Tree, J48, CART), the two Random Forest (Random Forest, W-Random Forest) and the two AdaBoost (AdaBoost, W-AdaBoostM1) on the same chart. This chart is shown below.

ROC Curve 7 Model Comparison

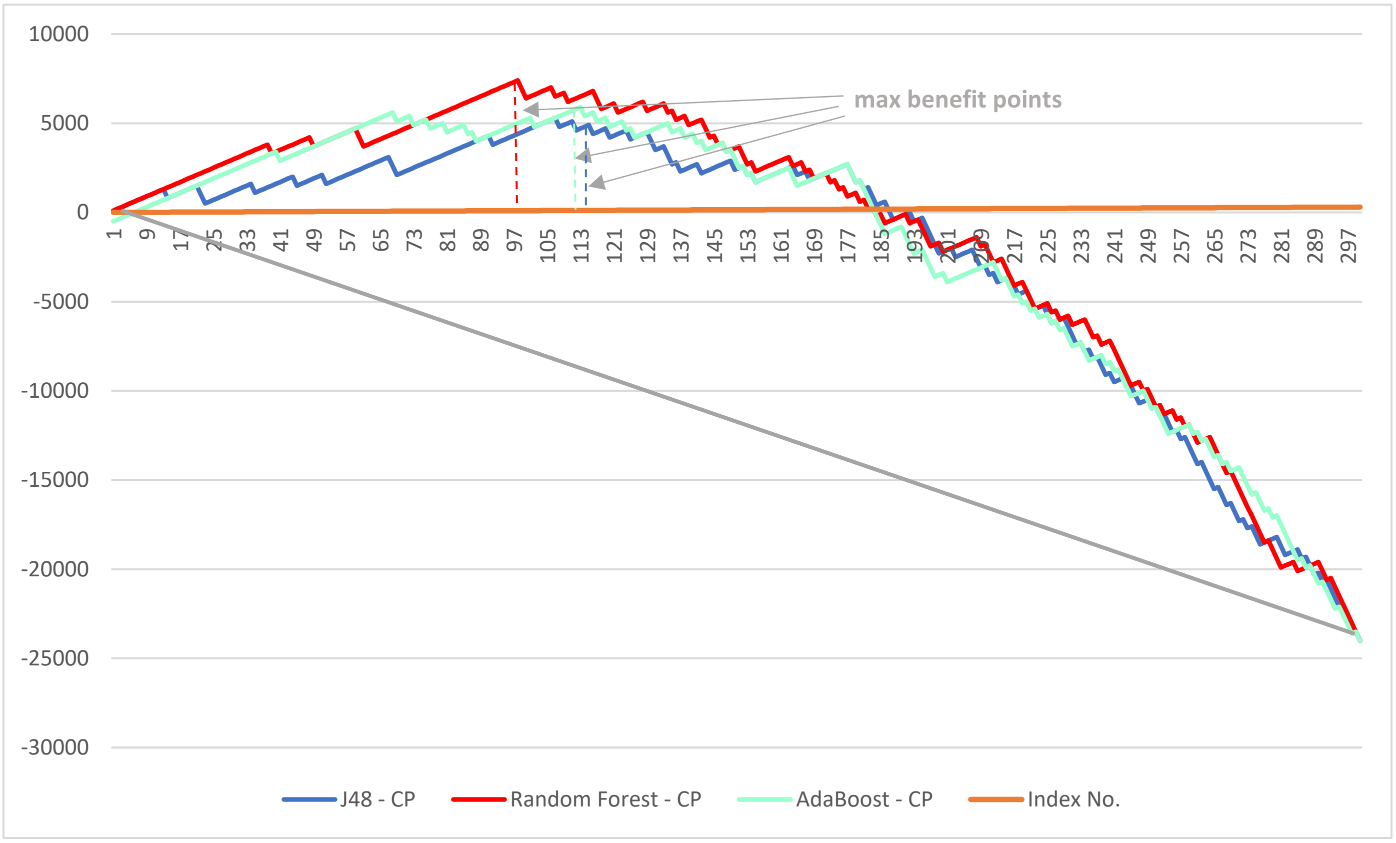


From the ROC curve above we can see that Random Forest performed better than W-Random Forest. Also, AdaBoost performed better than W-AdaBoostM1. This matched the AUC scores in which W-Random Forest had a score of .782 and Random Forest a score of .785. W-AdaBoostM1 had an AUC score of 0.762 and AdaBoost a score of .770. Therefore, we selected Random Forest and AdaBoost as the operators to use for each from their types. We can see, overall, the order of performance of all the models from highest to lowest is: Random Forest, W-Random Forest, AdaBoost (on J48), W-AdaBoostM1 (on J48), J48, Standard Decision Tree, CART.

Threshold and Business Performance Measure

For the final analysis we looked at the cost figures as we did in question 5 to see which of the newly selected models, Random Forest and AdaBoost, produces the most cumulative profit. We will include the previously selected and analyzed J48 model, for reason of comparison, in the new chart. We followed the same methodology as detailed in question 5. The graph of the cumulative benefit lift curve plotting all three is shown below.

Cumulative Benefit Lift Curve – J48, Random Forest, AdaBoost



The Random Forest model proved to be the most profitable with a maximum cumulative profit of 7,400 DM. This is at the threshold confidence for the 1 at a level of 0.72899 and above. The second most profitable was AdaBoost with a maximum cumulative profit of 5,900 DM at a threshold of 0.96043 and above. Lastly was J48, with the same results as in question 5, a maximum cumulative benefit of 4,800 DM at the threshold confidence level of 0.87645 and above. The summary chart of the performance of all three models is on the next page. Note that the accuracy, recall, and specificity measures are given at the confidence score for the 1 that maximizes cumulative benefit.

Summary of Various Performance Measures
J48, Random Forest, AdaBoost

		J48	Random Forest	AdaBoost of J48
Performance	Threshold	0.87645+	0.72899+	0.96043+
	Overall Accuracy	60.67%	60.00%	61.67%
	Sensitivity / Recall	49.05%	44.76%	49.52%
	Specificity	87.78%	95.56%	90.00%
	ROC: AUC – Value Listed	0.757	0.785	0.770
	Max Cumulative Profit	4,800	7,400	5,900

Random Forest created the best model across all the models tested. It produced the highest maximum cumulative profit as well as having the best overall ROC curve and AUC score. Conceptually, the idea of randomly selecting a subset of variables to develop trees out of is interesting. These random combinations produce more varying trees which provide the model less correlated trees and thus reduces error. Also, randomly selecting a subset of attributes from which to create the trees has the potential of producing interesting combinations of branching that might not have been found with the top-down greedy approach of decision trees. If we had to choose one model to implement it would be Random Forest.