

INDEX

Sl.No	Name Of The Experiment	Page No.
1	Credit Risk Assessment : 1.List all the categorical (or nominal) attributes and the real-valued attributes separately	3 4
	2. What attributes do you think might be crucial in making the credit assessment? Come up with some simple rules in plain English using your selected attributes.	5
	3. One type of model that you can create is a Decision Tree - train a Decision Tree using the complete dataset as the training data. Report the model obtained after raining.	7
	4. Suppose you use your above model trained on the complete dataset, and classify credit good/bad for each of the examples in the dataset. What % of examples can you classify correctly? (This is also called testing on the training set) Why do you think you cannot get 100 % training accuracy?	11
	5. Is testing on the training set as you did above a good idea? Why or Why not ?	12
	6. One approach for solving the problem encountered in the previous question is using cross-validation? Describe what is cross-validation briefly. Train a Decision Tree again using cross-validation and report your results. Does your accuracy increase/decrease? Why?	12
	7. Check to see if the data shows a bias against "foreign workers" (attribute 20), or "personal-status" (attribute 9). One way to do this (perhaps rather simple minded) is to remove these attributes from the dataset and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. To remove an attribute, you can use the preprocess tab in Weka's GUI Explorer. Did removing these attributes have any significant effect? Discuss.	17
	8. Another question might be, do you really need to input so many attributes to get good results? Maybe only a few would do. For example, you could try just having attributes 2, 3, 5, 7, 10, 17 (and 21, the class attribute (naturally)). Try out some combinations. (You had removed two attributes in problem 7. Remember to reload the arff data file to get all the attributes initially before you start selecting the ones you want.)	18
	9. Sometimes, the cost of rejecting an applicant who actually has a good credit (case 1) might be higher than accepting an applicant who has bad credit (case 2). Instead of counting the misclassifications equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. You can do this by using a cost matrix in Weka. Train your Decision Tree again and report the Decision Tree and crossvalidation results. Are	22

	they significantly different from results obtained in problem 6 (using equal cost)?	
	10. Do you think it is a good idea to prefer simple decision trees instead of having long complex decision trees? How does the complexity of a Decision Tree relate to the bias of the model?	24
	11. You can make your Decision Trees simpler by pruning the nodes. One approach is to use Reduced Error Pruning - Explain this idea briefly. Try reduced error pruning for training your Decision Trees using cross-validation (you can do this in Weka) and report the Decision Tree you obtain? Also, report your accuracy using the pruned model. Does your accuracy increase?	25
	12.(Extra Credit): How can you convert a Decision Trees into "if-then-else rules". Make up your own small Decision Tree consisting of 2-3 levels and convert it into a set of rules. There also exist different classifiers that output the model in the form of rules - one such classifier in Weka is rules. PART, train this model and report the set of rules obtained. Sometimes just one attribute can be good enough in making the decision, yes, just one ! Can you predict what attribute that might be in this dataset ? OneR classifier uses a single attribute to make decisions (it chooses the attribute based on minimum error). Report the rule obtained by training a one R classifier. Rank the performance of j48, PART and oneR.	28
2	Hospital Management System	32
	1. Star Schema	36
	2. Snowflake Schema	37
	3. Fact constellation schema	38

EXPERIMENT 1

Credit Risk Assessment Description:

The business of banks is making loans. Assessing the credit worthiness of an applicant is of crucial importance. You have to develop a system to help a loan officer decide whether the credit of a customer is good, or bad. A bank's business rules regarding loans must consider two opposing factors. On the one hand, a bank wants to make as many loans as possible. Interest on these loans is the bank's profit source. On the other hand, a bank cannot afford to make too many bad loans. Too many bad loans could lead to the collapse of the bank. The bank's loan policy must involve a compromise not too strict, and not too lenient.

To do the assignment, you first and foremost need some knowledge about the world of credit. You can acquire such knowledge in a number of ways.

1. Knowledge Engineering. Find a loan officer who is willing to talk. Interview her and try to represent her knowledge in the form of production rules.
2. Books. Find some training manuals for loan officers or perhaps a suitable textbook on finance. Translate this knowledge from text form to production rule form.
3. Common sense. Imagine yourself as a loan officer and make up reasonable rules which can be used to judge the credit worthiness of a loan applicant.
4. Case histories. Find records of actual cases where competent loan officers correctly judged when not to, approve a loan application.

The German Credit Data :

Actual historical credit data is not always easy to come by because of confidentiality rules. Here is one such dataset (original) Excel spreadsheet version of the German credit data (download from web). In spite of the fact that the data is German, you should probably make use of it for this assignment, (Unless you really can consult a real loan officer !) A few notes on the German dataset :

- DM stands for Deutsche Mark, the unit of currency, worth about 90 cents Canadian (but looks and acts like a quarter).
- Owns_telephone. German phone rates are much higher than in Canada so fewer people own telephones.
- Foreign_worker. There are millions of these in Germany (many from Turkey). It is very hard to get German citizenship if you were not born of German parents.
- There are 20 attributes used in judging a loan applicant. The goal is to classify the applicant into one of two categories, good or bad.

SUBTASKS:

1.List all the categorical (or nominal) attributes and the real-valued attributes separately.

AIM: To separate the nominal attributes and also the real valued attributes by using the weka tool.

PROCEDURE:

- 1) Open the Weka GUI Chooser.
- 2) Select EXPLORER present in Applications.
- 3) Select Preprocess Tab.
- 4) Go to OPEN file and browse the file that is already stored in the system “credi-g.arff”.
- 5) Clicking on any attribute in the left panel will show the basic statistics on that selected attribute.

Attributes:-

1. checking_status
2. duration
3. credit history
4. purpose
5. credit amount
6. savings_status
7. employment duration
8. installment rate
9. personal status
10. debtors
11. residence_since
12. property
14. installment plans
15. housing
16. existing credits
17. job
18. num_dependents
19. telephone
20. foreign worker

Categorical or Nominal attributes:-

1. checking_status
2. credit history
3. purpose
4. savings_status
5. employment
6. personal status

- 7. debtors
- 8. property
- 9. installment plans
- 10. housing
- 11. job
- 12. telephone
- 13. foreign worker

Real valued attributes:-

- 1. duration
- 2. credit amount
- 3. credit amount
- 4. residence
- 5. age
- 6. existing credits
- 7. num_dependents

2.What attributes do you think might be crucial in making the credit assessment ? Come up with some simple rules in plain English using your selected attributes.

AIM: To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best known constraints are minimum thresholds on support and confidence. The support supp(X) of an itemset X is defined as the proportion of transactions in the data set which contain the itemset. In the example database, the itemset {milk, bread} has a support of $2 / 5 = 0.4$ since it occurs in 40% of all transactions (2 out of 5 transactions).

ALGORITHM:

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two sub problems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Suppose one of the large itemsets is L_k , $L_k = \{I_1, I_2, \dots, I_k\}$, association rules with this itemsets are generated in the following way: the first rule is $\{I_1, I_2, \dots, I_{k-1}\} \rightarrow \{I_k\}$, by checking the confidence this rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness

of them. Those processes iterated until the antecedent becomes empty. Since the second subproblem is quite straight forward, most of the researches focus on the first subproblem.

The Apriori algorithm finds the frequent sets L In Database D.

- Find frequent set L_{k-1} .
- Join Step.
- C_k is generated by joining L_{k-1} with itself
- Prune Step.

o Any $(k-1)$ itemset that is not frequent cannot be a subset of a frequent k itemset, hence should be removed.

Where · (C_k : Candidate itemset of size k)

· (L_k : frequent itemset of size k)

Apriori Pseudocode

Apriori (T, ℓ)

$L < \{ \text{Large 1itemsets that appear in more than } \ell \text{ transactions} \}$

while $L(k) \neq \emptyset$

$C(k) < \text{Generate}(L_{k-1})$

for transactions $t \in T$

$C(t) \text{Subset}(C_k, t)$

for candidates $c \in C(t)$

$\text{count}[c] < \text{count}[c] + 1$

$L(k) < \{ c \in C(k) | \text{count}[c] \geq \ell \}$

$K < K + 1$

return $\bigcup L(k)$.

PROCEDURE:

- 1) Given the Credit database for mining.
- 2) Select EXPLORER in WEKA GUI Chooser.
- 3) Load “Credit-g.arff” in Weka by Open file in Preprocess tab.
- 4) Select only Nominal values.
- 5) Go to Associate Tab.
- 6) Select Apriori algorithm from “Choose” button present in Associator weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
- 7) Select Start button
- 8) Now we can see the sample rules.

According to me the following attributes may be crucial in making the credit risk assessment.

1. Credit_history
2. Employment
3. Property_magnitude
4. job
5. duration

6. credit_amount
7. installment
8. existing credit

Basing on the above attributes, we can make a decision whether to give credit or not.

3. One type of model that you can create is a Decision Tree - train a Decision Tree using the complete dataset as the training data. Report the model obtained after training.

AIM: Implementing the decision tree analysis and the training data in the data set.

Different Classification Algorithms: Oracle Data Mining provides the following algorithms for classification:

Decision Tree - Decision trees automatically generate rules, which are conditional statements that reveal the logic used to build the tree.

Naive Bayes - Naive Bayes uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.

PROCEDURE:

- 1) Open Weka GUI Chooser.
- 2) Select EXPLORER present in Applications.
- 3) Select Preprocess Tab.
- 4) Go to OPEN file and browse the file that is already stored in the system “Credit-g.arff”.
- 5) Go to Classify tab.
- 6) j48 in Java and can be selected by clicking the button ,choose and select tree j48 .

Output:

J48 pruned tree

```
-----
checking_status = <0
| foreign_worker = yes
| | duration <= 11
| | | existing_credits <= 1
| | | | property_magnitude = real estate: good (8.0/1.0)
| | | | property_magnitude = life insurance
| | | | | own_telephone = none: bad (2.0)
| | | | | own_telephone = yes: good (4.0)
| | | | property_magnitude = car: good (2.0/1.0)
| | | | property_magnitude = no known property: bad (3.0)
| | | | existing_credits > 1: good (14.0)
| | | duration > 11
| | | | job = unemp/unskilled non res: bad (5.0/1.0)
| | | | job = unskilled resident
| | | | purpose = new car
```

```
||||| own_telephone = none: bad (10.0/2.0)
||||| own_telephone = yes: good (2.0)
||| purpose = used car: bad (1.0)
||| purpose = furniture/equipment
||| employment = unemployed: good (0.0)
||| employment = <1: bad (3.0)
||| employment = 1<=X<4: good (4.0)
||| employment = 4<=X<7: good (1.0)
||| employment = >=7: good (2.0)
| purpose = radio/tv
| existing_credits <= 1: bad (10.0/3.0)
| existing_credits > 1: good (2.0)
| purpose = domestic appliance: bad (1.0)
| purpose = repairs: bad (1.0)
| purpose = education: bad (1.0)
| purpose = vacation: bad (0.0)
| purpose = retraining: good (1.0)
| purpose = business: good (3.0)
| purpose = other: good (1.0)
job = skilled
| other_parties = none
| duration <= 30
| | savings_status = <100
| | credit_history = no credits/all paid: bad (8.0/1.0)
| | credit_history = all paid: bad (6.0)
| | credit_history = existing paid
| | | own_telephone = none
| | | | existing_credits <= 1
| | | | | property_magnitude = real estate
| | | | | age <= 26: bad (5.0)
| | | | | age > 26: good (2.0)
| | | | | property_magnitude = life insurance: bad (7.0/2.0)
| | | | | property_magnitude = car
| | | | | credit_amount <= 1386: bad (3.0)
| | | | | credit_amount > 1386: good (11.0/1.0)
| | | | | property_magnitude = no known property: good (2.0)
| | | | | existing_credits > 1: bad (3.0)
| | | | | own_telephone = yes: bad (5.0)
| | | | | credit_history = delayed previously: bad (4.0)
| | | | | credit_history = critical/other existing credit: good (14.0/4.0)
| | | | | savings_status = 100<=X<500
| | | | | credit_history = no credits/all paid: good (0.0)
| | | | | credit_history = all paid: good (1.0)
| | | | | credit_history = existing paid: bad (3.0)
| | | | | credit_history = delayed previously: good (0.0)
| | | | | credit_history = critical/other existing credit: good (2.0)
| | | | | savings_status = 500<=X<1000: good (4.0/1.0)
| | | | | savings_status = >=1000: good (4.0)
```

```
||| savings_status = no known savings
|||| existing_credits <= 1
||||| own_telephone = none: bad (9.0/1.0)
||||| own_telephone = yes: good (4.0/1.0)
|||||| existing_credits > 1: good (2.0)
||||| duration > 30: bad (30.0/3.0)
|||| other_parties = co applicant: bad (7.0/1.0)
|||| other_parties = guarantor: good (12.0/3.0)
||| job = high qualif/self emp/mgmt: good (30.0/8.0)
| foreign_worker = no: good (15.0/2.0)
checking_status = 0<=X<200
| credit_amount <= 9857
|| savings_status = <100
||| other_parties = none
||| duration <= 42
|||| personal_status = male div/sep: bad (8.0/2.0)
|||| personal_status = female div/dep/mar
||||| purpose = new car: bad (5.0/1.0)
||||| purpose = used car: bad (1.0)
||||| purpose = furniture/equipment
||||| duration <= 10: bad (3.0)
||||| duration > 10
|||||| duration <= 21: good (6.0/1.0)
|||||| duration > 21: bad (2.0)
||||| purpose = radio/tv: good (8.0/2.0)
||||| purpose = domestic appliance: good (0.0)
||||| purpose = repairs: good (1.0)
||||| purpose = education: good (4.0/2.0)
||||| purpose = vacation: good (0.0)
||||| purpose = retraining: good (0.0)
||||| purpose = business
||||| residence_since <= 2: good (3.0)
||||| residence_since > 2: bad (2.0)
||||| purpose = other: good (0.0)
||||| personal_status = male single: good (52.0/15.0)
||||| personal_status = male mar/wid
||||| duration <= 10: good (6.0)
||||| duration > 10: bad (10.0/3.0)
||||| personal_status = female single: good (0.0)
||| duration > 42: bad (7.0)
||| other_parties = co applicant: good (2.0)
||| other_parties = guarantor
||| purpose = new car: bad (2.0)
||| purpose = used car: good (0.0)
||| purpose = furniture/equipment: good (0.0)
||| purpose = radio/tv: good (18.0/1.0)
||| purpose = domestic appliance: good (0.0)
||| purpose = repairs: good (0.0)
```

```

|||| purpose = education: good (0.0)
|||| purpose = vacation: good (0.0)
|||| purpose = retraining: good (0.0)
|||| purpose = business: good (0.
|||| purpose = other: good (0.0)
|| savings_status = 100<=X<500
|| purpose = new car: bad (15.0/5.0)
|| purpose = used car: good (3.0)
|| purpose = furniture/equipment: bad (4.0/1.0)
|| purpose = radio/tv: bad (8.0/2.0)
|| purpose = domestic appliance: good (0.0)
|| purpose = repairs: good (2.0)
|| purpose = education: good (0.0)
|| purpose = vacation: good (0.0)
|| purpose = retraining: good (0.0)
|| purpose = business
||| housing = rent
|||| existing_credits <= 1: good (2.0)
|||| existing_credits > 1: bad (2.0)
||| housing = own: good (6.0)
||| housing = for free: bad (1.0)
||| purpose = other: good (1.0)
|| savings_status = 500<=X<1000: good (11.0/3.0)
|| savings_status = >=1000: good (13.0/3.0)
|| savings_status = no known savings: good (41.0/5.0)
| credit_amount > 9857: bad (20.0/3.0)
checking_status = >=200: good (63.0/14.0)
checking_status = no checking: good (394.0/46.0)
Number of Leaves : 103
Size of the tree : 140
Time taken to build model: 0.03 seconds

```

==== Evaluation on training set ====

==== Summary ====

Correctly Classified Instances	855	85.5 %
Incorrectly Classified Instances	145	14.5 %
Kappa statistic	0.6251	
Mean absolute error	0.2312	
Root mean squared error	0.34	
Relative absolute error	55.0377 %	
Root relative squared error	74.2015 %	
Total Number of Instances	1000	

4:Suppose you use your above model trained on the complete dataset, and classify credit good/bad for each of the examples in the dataset. What % of examples can you classify

correctly? (This is also called testing on the training set) Why do you think you cannot get 100 % training accuracy?

AIM: Determining and classifying the credit good or bad in the dataset with an Accuracy.

PROCEDURE:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “Credit-g.arff”.
- 6) Go to Classify tab.
- 7) Choose Classifier “Tree”
- 8) Select J48
- 9) Select Test options “Use training set”
- 10) if need select attribute.
- 11) Now start weka.
- 12) Now we can see the output details in the Classifier output.

In the above model we trained complete dataset and we classified credit good/bad for each of the examples in the dataset.

For example:

IF

purpose=vacation THEN

credit=bad

ELSE

purpose=business THEN

Credit=good

In this way we classified each of the examples in the dataset.

We classified 85.5% of examples correctly and the remaining 14.5% of examples are incorrectly classified. We can't get 100% training accuracy because out of the 20 attributes, we have some unnecessary attributes which are also been analyzed and trained.

Due to this the accuracy is affected and hence we can't get 100% training accuracy.

5) Is testing on the training set as you did above a good idea? Why or Why not?

AIM: Testing the training set is it necessary to analyze the data.

THEORY: Comparing the "Correctly Classified Instances" from this test set with the "Correctly

Classified Instances" from the training set, we see the accuracy of the model, which indicates

that the model will not break down with unknown data, or when future data is applied to it.

PROCEDURE:

- i) In Test options, select the Supplied test set radio button
- ii) Click Set
- iii) Choose the file which contains records that were not in the training set we used to create the model.
- iv) Click Start(WEKA will run this test data set through the model we already created.)
- v) Compare the output results with that of the 4th experiment.

According to the rules, for the maximum accuracy, we have to take 2/3 of the dataset as training set and the remaining 1/3 as test set. But here in the above model we have taken complete dataset as training set which results only 85.5% accuracy.

This is done for the analyzing and training of the unnecessary attributes which does not make a crucial role in credit risk assessment. And by this complexity is increasing and finally it leads to the minimum accuracy.

If some part of the dataset is used as a training set and the remaining as test set then it leads to the accurate results and the time for computation will be less.

This is why, we prefer not to take complete dataset as training set.

6. One approach for solving the problem encountered in the previous question is using cross-validation?

Describe what is cross-validation briefly. Train a Decision Tree again using cross-validation and report your results. Does your accuracy increase/decrease? Why?

Cross validation:-

In k-fold cross-validation, the initial data are randomly portioned into 'k' mutually exclusive subsets or folds D₁, D₂, D₃, , D_k. Each of approximately equal size. Training and testing is

performed 'k' times. In iteration I, partition D_i is reserved as the test set and the remaining partitions are

collectively used to train the model. That is in the first iteration subsets D₂, D₃, , D_k collectively

serve as the training set in order to obtain as first model. Which is tested on D_i. The second trained on the subsets D₁, D₃, , D_k and test on the D₂ and so on....

AIM: To test the cross validation while analyzing the data.

PROCEDURE:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “credit-g.arff”. Go to Classify tab.
- 6) Choose Classifier “Tree”
- 7) Select J48
- 8) Select Test options “Cross-validation”.
- 9) Set “Folds” Ex:10
- 10) if need select attribute.
- 11) now Start weka.

Now we can see the output details in the Classifier output. Compare the output results with that of the 4 th experiment check whether the accuracy increased or decreased?

OUTPUT:

J48 pruned tree :-

```
-----
| checking_status = <0
| | foreign_worker = yes
| | | duration <= 11
| | | | existing_credits <= 1
| | | | | property_magnitude = real estate: good (8.0/1.0)
| | | | | property_magnitude = life insurance
| | | | | | own_telephone = none: bad (2.0)
| | | | | | own_telephone = yes: good (4.0)
| | | | | property_magnitude = car: good (2.0/1.0)
| | | | | property_magnitude = no known property: bad (3.0)
| | | | | existing_credits > 1: good (14.0)
| | | | duration > 11
| | | | | job = unemp/unskilled non res: bad (5.0/1.0)
| | | | | job = unskilled resident
| | | | | | purpose = new car
| | | | | | | own_telephone = none: bad (10.0/2.0)
| | | | | | | own_telephone = yes: good (2.0)
| | | | | | | purpose = used car: bad (1.0)
| | | | | | | purpose = furniture/equipment
| | | | | | | employment = unemployed: good (0.0)
```

||||| employment = <1: bad (3.0)
||||| employment = 1<=X<4: good (4.0)
||||| employment = 4<=X<7: good (1.0)
||||| employment = >=7: good (2.0)
||| purpose = radio/tv
||| existing_credits <= 1: bad (10.0/3.0)
||| existing_credits > 1: good (2.0)
||| purpose = domestic appliance: bad (1.0)
||| purpose = repairs: bad (1.0)
||| purpose = education: bad (1.0)
||| purpose = vacation: bad (0.0)
||| purpose = retraining: good (1.0)
||| purpose = business: good (3.0)
||| purpose = other: good (1.0)
|| job = skilled
||| other_parties = none
||| duration <= 30
||| savings_status = <100
||| credit_history = no credits/all paid: bad (8.0/1.0)
||| credit_history = all paid: bad (6.0)
||| credit_history = existing paid
||| own_telephone = none
||| existing_credits <= 1
||| property_magnitude = real estate
||| age <= 26: bad (5.0)
||| age > 26: good (2.0)
||| property_magnitude = life insurance: bad (7.0/2.0)
||| property_magnitude = car
||| credit_amount <= 1386: bad (3.0)
||| credit_amount > 1386: good (11.0/1.0)
||| property_magnitude = no known property: good (2.0)
||| existing_credits > 1: bad (3.0)
||| own_telephone = yes: bad (5.0)
||| credit_history = delayed previously: bad (4.0)
||| credit_history = critical/other existing credit: good (14.0/4.0)
|| savings_status = 100<=X<500
||| credit_history = no credits/all paid: good (0.0)
||| credit_history = all paid: good (1.0)
||| credit_history = existing paid: bad (3.0)
||| credit_history = delayed previously: good (0.0)
||| credit_history = critical/other existing credit: good (2.0)
||| savings_status = 500<=X<1000: good (4.0/1.0)
||| savings_status = >=1000: good (4.0)
||| savings_status = no known savings
||| existing_credits <= 1
||| own_telephone = none: bad (9.0/1.0)
||| own_telephone = yes: good (4.0/1.0)
||| existing_credits > 1: good (2.0)

||||| duration > 30: bad (30.0/3.0)
||| other_parties = co applicant: bad (7.0/1.0)
||| other_parties = guarantor: good (12.0/3.0)
|| job = high qualif/self emp/mgmt: good (30.0/8.0)
| foreign_worker = no: good (15.0/2.0)
checking_status = 0<=X<200
| credit_amount <= 9857
| | savings_status = <100
| | other_parties = none
| | | duration <= 42
| | | personal_status = male div/sep: bad (8.0/2.0)
| | | personal_status = female div/dep/mar
| | | purpose = new car: bad (5.0/1.0)
| | | purpose = used car: bad (1.0)
| | | purpose = furniture/equipment
| | | | duration <= 10: bad (3.0)
| | | | duration > 10
| | | | | duration <= 21: good (6.0/1.0)
| | | | | duration > 21: bad (2.0)
| | | | purpose = radio/tv: good (8.0/2.0)
| | | | purpose = domestic appliance: good (0.0)
| | | | purpose = repairs: good (1.0)
| | | | purpose = education: good (4.0/2.0)
| | | | purpose = vacation: good (0.0)
| | | | purpose = retraining: good (0.0)
| | | | purpose = business
| | | | | residence_since <= 2: good (3.0)
| | | | | residence_since > 2: bad (2.0)
| | | | purpose = other: good (0.0)
| | | personal_status = male single: good (52.0/15.0)
| | | personal_status = male mar/wid
| | | | duration <= 10: good (6.0)
| | | | duration > 10: bad (10.0/3.0)
| | | | personal_status = female single: good (0.0)
| | | duration > 42: bad (7.0)
| | other_parties = co applicant: good (2.0)
| | other_parties = guarantor
| | | purpose = new car: bad (2.0)
| | | purpose = used car: good (0.0)
| | | purpose = furniture/equipment: good (0.0)
| | | purpose = radio/tv: good (18.0/1.0)
| | | purpose = domestic appliance: good (0.0)
| | | purpose = repairs: good (0.0)
| | | purpose = education: good (0.0)
| | | purpose = vacation: good (0.0)
| | | purpose = retraining: good (0.0)
| | | purpose = business: good (0.0)
| | | purpose = other: good (0.0)

```

savings_status = 100<=X<500
| purpose = new car: bad (15.0/5.0)
| purpose = used car: good (3.0)
| purpose = furniture/equipment: bad (4.0/1.0)
| purpose = radio/tv: bad (8.0/2.0)
| purpose = domestic appliance: good (0.0)
| purpose = repairs: good (2.0)
| purpose = education: good (0.0)
| purpose = vacation: good (0.0)
| purpose = retraining: good (0.0)
|   | purpose = business
|   | housing = rent
|   |   | existing_credits <= 1: good (2.0)
|   |   | existing_credits > 1: bad (2.0)
|   | housing = own: good (6.0)
|   | housing = for free: bad (1.0)
|   | purpose = other: good (1.0)
|   | savings_status = 500<=X<1000: good (11.0/3.0)
|   | savings_status = >=1000: good (13.0/3.0)
|   | savings_status = no known savings: good (41.0/5.0)
|   | credit_amount > 9857: bad (20.0/3.0)
|   | checking_status = >=200: good (63.0/14.0)
|   | checking_status = no checking: good (394.0/46.0)
Number of Leaves : 103
Size of the tree : 140
Time taken to build model: 0.07 seconds

```

==== Stratified cross-validation ====
==== Summary ===

Correctly Classified Instances	705	70.5 %
Incorrectly Classified Instances	295	29.5 %
Kappa statistic		0.2467
Mean absolute error		0.3467
Root mean squared error		0.4796
Relative absolute error		82.5233 %
Root relative squared error		104.6565 %
Total Number of Instances		1000

==== Detailed Accuracy By Class ===

TP Rate
0.84
0.39
Weighted Avg.
FP Rate Precision Recall F-Measure ROC Area Class
0.610.763 0.84 0.799 0.639 good
0.160.511 0.39 0.442 0.639 bad

0.705 0.475 0.687 0.705 0.692 0.639

==== Confusion Matrix ====

a b <-- classified as

588 112 | a = good

183 117 | b = bad

7. Check to see if the data shows a bias against "foreign workers" (attribute 20),or "personal-status"(attribute 9). One way to do this (perhaps rather simple minded) is to remove these attributes from the dataset and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. To remove an attribute you can use the reprocess tab in Weka's GUI Explorer. Did removing these attributes have any significant effect? Discuss.

AIM: Analyzing the removing attributes from the dataset.

PROCEDURE:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “Credit-g.arff”.
- 6) In the "Filter" panel, click on the "Choose" button. This will show a popup window with list available filters.
- 7) Select “weka.filters.unsupervised.attribute.Remove”
- 8) Next, click on text box immediately to the right of the "Choose" button
- 9) In the resulting dialog box enter the index of the attribute to be filtered out (Make sure that the "invert Selection" option is set to false)
- 10) Then click "OK" . Now, in the filter box you will see "Remove -R 1"
- 11) Click the "Apply" button to apply this filter to the data. This will remove the "id" attribute and create a new working relation
- 12) To save the new working relation as an ARFF file, click on save button in the top panel.

- 13) Go to OPEN file and browse the file that is newly saved (attribute deleted file)
- 14) Go to Classify tab.
- 15) Choose Classifier “Tree.”
- 16) Select j48 tree
- 17) Select Test options “Use training set”
- 18) If need select attribute.
- 19) Now start weka.
- 20) Now we can see the output details in the Classifier output.
- 21) Right click on the result list and select ” visualize tree “option .
- 22) Compare the output results with that of the 4th experiment
- 23) Check whether the accuracy increased or decreased.
- 24) Check whether removing these attributes have any significant effect.

This increase in accuracy is because thus two attributes are not much important in training and analyzing by removing this, the time has been reduced to some extent and then it results in increase in the accuracy. The decision tree which is created is very large compared to the decision tree which we have trained now. This is the main difference between these two decision trees.

8. Another question might be, do you really need to input so many attributes to get good results? Maybe only a few would do. For example, you could try just having attributes 2, 3, 5, 7, 10, 17 (and 21, the class attribute (naturally)). Try out some combinations. (You had removed two attributes in problem 7. Remember to reload the arff data file to get all the attributes initially before you start selecting the ones you want.)

AIM: To removing the selected attributes and also to reload the arff data file to get all the attributes in the data set.

PROCEDURE:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.

- 5) Go to OPEN file and browse the file that is already stored in the system “credit-g.arff”
- 6) Select some of the attributes from attributes list which are to be removed. With this step only the attributes necessary for classification are left in the attributes panel.
- 7) Then go to Classify tab.
- 8) Choose Classifier “Tree”
- 9) Select j48
- 10) Select Test options “Use training set”
- 11) If need select attribute.
- 12) Now start Weka.
- 13) Now we can see the output details in the Classifier output.
- 14) Right click on the result list and select “visualize tree” option.
- 15) Compare the output results with that of the 4th experiment.
- 16) Check whether the accuracy increased or decreased?
- 17) Check whether removing these attributes have any significant effect.

==== Classifier model (full training set) ====

OUTPUT:

J48 pruned tree

credit_history = no credits/all paid: bad (40.0/15.0)

credit_history = all paid

| employment = unemployed

|| duration <= 36: bad (3.0)

|| duration > 36: good (2.0)

| employment = <1

|| duration <= 26: bad (7.0/1.0)

|| duration > 26: good (2.0)

| employment = 1<=X<4: good (15.0/6.0)

| employment = 4<=X<7: bad (10.0/4.0)

| employment = >=7

|| job = unemp/unskilled non res: bad (0.0)

|| job = unskilled resident: good (3.0)

|| job = skilled: bad (3.0)

|| job = high qualif/self emp/mgmt: bad (4.0)

credit_history = existing paid

| credit_amount <= 8648

|| duration <= 40: good (476.0/130.0)

|| duration > 40: bad (27.0/8.0)

| credit_amount > 8648: bad (27.0/7.0)

credit_history = delayed previously

| employment = unemployed

|| credit_amount <= 2186: bad (4.0/1.0)

|| credit_amount > 2186: good (2.0)

| employment = <1

|| duration <= 18: good (2.0)

|| duration > 18: bad (10.0/2.0)

| employment = 1<=X<4: good (33.0/6.0)

| employment = 4<=X<7

|| credit_amount <= 4530

||| credit_amount <= 1680: good (3.0)

||| credit_amount > 1680: bad (3.0)

|| credit_amount > 4530: good (11.0)

| employment = >=7

|| job = unemp/unskilled non res: good (0.0)

|| job = unskilled resident: good (2.0/1.0)

|| job = skilled: good (14.0/4.0)

|| job = high qualif/self emp/mgmt: bad (4.0/1.0)

credit_history = critical/other existing credit: good (293.0/50.0)

Number of Leaves : 27

Size of the tree : 40

Time taken to build model: 0.01 seconds

==== Evaluation on training set ====

==== Summary ====

Correctly Classified Instances 764 76.4 %

Incorrectly Classified Instances 236 23.6 %

Kappa statistic 0.3386

Mean absolute error 0.3488

Root mean squared error 0.4176

Relative absolute error 83.0049 %

Root relative squared error 91.1243 %

Total Number of Instances 1000

==== Classifier model (full training set) ====

J48 pruned tree

credit_history = no credits/all paid: bad (40.0/15.0)

credit_history = all paid

| employment = unemployed

|| duration <= 36: bad (3.0)

|| duration > 36: good (2.0)

| employment = <1

|| duration <= 26: bad (7.0/1.0)

|| duration > 26: good (2.0)

| employment = 1<=X<4: good (15.0/6.0)

| employment = 4<=X<7: bad (10.0/4.0)

| employment = >=7

|| job = unemp/unskilled non res: bad (0.0)

|| job = unskilled resident: good (3.0)

```

|| job = skilled: bad (3.0)
|| job = high qualif/self emp/mgmt: bad (4.0)
credit_history = existing paid
| credit_amount <= 8648
|| duration <= 40: good (476.0/130.0)
|| duration > 40: bad (27.0/8.0)
| credit_amount > 8648: bad (27.0/7.0)
credit_history = delayed previously
| employment = unemployed
|| credit_amount <= 2186: bad (4.0/1.0)
|| credit_amount > 2186: good (2.0)
| employment = <1
|| duration <= 18: good (2.0)
|| duration > 18: bad (10.0/2.0)
| employment = 1<=X<4: good (33.0/6.0)
| employment = 4<=X<7
|| credit_amount <= 4530
|| | credit_amount <= 1680: good (3.0)
|| | credit_amount > 1680: bad (3.0)
|| credit_amount > 4530: good (11.0)
| employment = >=7
|| job = unemp/unskilled non res: good (0.0)
|| job = unskilled resident: good (2.0/1.0)
|| job = skilled: good (14.0/4.0)
|| job = high qualif/self emp/mgmt: bad (4.0/1.0)
credit_history = critical/other existing credit: good (293.0/50.0)

```

Number of Leaves : 27

Size of the tree : 40

Time taken to build model: 0.01 seconds

==== Stratified cross-validation ===

==== Summary ===

Correctly Classified Instances	703	70.3 %
Incorrectly Classified Instances	297	29.7 %
Kappa statistic		0.1759
Mean absolute error	0.3862	
Root mean squared error	0.4684	
Relative absolute error		91.9029 %
Root relative squared error	102.2155 %	
Total Number of Instances	1000	

9. Sometimes, the cost of rejecting an applicant who actually has a good credit(case 1) might be higher than accepting an applicant who has bad credit (case 2). Instead of counting the misclassifications equally in both cases, give a higher cost to the first case

(say cost 5) and lower cost to the second case. You can do this by using a cost matrix in Weka. Train your Decision Tree again and report the Decision Tree and cross-validation results. Are they significantly different from results obtained in problem 6 (using equal cost)?

AIM: Determining the good credit and the bad credit in the decision tree dataset.

PROCEDURE:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “Credit-g.arff”.
- 6) Go to Classify tab.
- 7) Choose Classifier “Tree”
- 8) Select j48
- 9) Select Test options “Training set”.
- 10) Click on “more options”.
- 11) Select cost sensitive evaluation and click on set button
- 12) Set the matrix values and click on resize. Then close the window.
- 13) Click Ok and start
- 14) We can see the output details in the Classifier output
- 15) Select Test options “Cross-validation”.
- 16) Set “Folds” Ex: 10
- 17) if need select attribute.
- 18) Now start weka.
- 19) Now we can see the output details in the Classifier output.

In the Problem 6, we used equal cost and we trained the decision tree. But here, we consider two cases with different cost. Let us take cost 5 in case 1 and cost 2 in case 2. When we give such costs in both cases and after training the decision tree, we can observe that almost equal to that of the decision tree obtained in problem 6.

But we find some difference in cost factor which is in summary in the difference in cost factor.

Case1 (cost 5) Case2 (cost 5)

Total Cost 3820 1705

Average cost 3.82 1.705

We don't find this cost factor in problem 6. As there we use equal cost. This is the major difference between the results of problem 6 and problem 9.

The cost matrices we used here:

Case 1: 5 1

1 5

Case 2: 2 1

1 2

10. Do you think it is a good idea to prefer simple decision trees instead of having long complex decision trees? How does the complexity of a Decision Tree relate to the bias of the model?

AIM: Analyzing the decision tree into the shortest form by using the dataset.

PROCEDURE: This will be based on the attribute set, and the requirement of relationship among attribute we want to study. This can be viewed based on the database and user requirement.

When we consider long complex decision trees, we will have many unnecessary attributes in the tree which results in increase of the bias of the model. Because of this, the accuracy of the model can also effected. This problem can be reduced by considering simple decision tree. The attributes will be less and it decreases the bias of the model. Due to this the result will be more accurate. So it is a good idea to prefer simple decision trees instead of long complex trees.

11. You can make your Decision Trees simpler by pruning the nodes. One approach is to use Reduced Error Pruning - Explain this idea briefly. Try reduced error pruning for training your Decision Trees using cross-validation (you can do this in Weka) and

report the Decision Tree you obtain ? Also, report your accuracy using the pruned model. Does your accuracy increase ?

THEORY:

Reduced-error pruning:-

The idea of using a separate pruning set for pruning—which is applicable to decision trees as well as rule sets—is called reduced-error pruning. The variant described previously prunes a rule immediately after it has been grown and is called incremental reduced-error pruning. Another possibility is to build a full, unpruned rule set first, pruning it afterwards by discarding individual tests. However, this method is much slower. Of course, there are many different ways to assess the worth of a rule based on the pruning set. A simple measure is to consider how well the rule would do at discriminating the predicted class from other classes if it were the only rule in the theory, operating under the closed world assumption. If it gets p instances right out of the t instances that it covers, and there are P instances of this class out of a total T of instances altogether, then it gets p positive instances right. The instances that it does not cover include $N - n$ negative ones, where $n = t - p$ is the number of negative instances that the rule covers and $N = T - P$ is the total number of negative instances. Thus the rule has an overall success ratio of $[p + (N - n)] / T$, and this quantity, evaluated on the test set, has been used to evaluate the success of a rule when using reduced-error pruning.

AIM: Analyzing the decision trees is simpler by pruning the nodes in the dataset.

PROCEDURE:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
Select some of the attributes from attributes list
- 7) Go to Classify tab.
- 8) Choose Classifier “Tree”
- 9) Select “NBTree” i.e., Navie Bayesian tree.
- 10) Select Test options “Use training set”
- 11) Right click on the text box besides choose button, select show properties

- 12) Now change unprune mode “false” to “true”.
- 13) Change the reduced error pruning % as needed.
- 14) If need select attribute.
- 15) Now start weka.
- 16) Now we can see the output details in the Classifier output.
- 17) Right click on the result list and select ” visualize tree “option.

OUTPUT:

J48 pruned tree

```
checking_status = <0
| foreign_worker = yes
| credit_history = no credits/all paid: bad (11.0/3.0)
| credit_history = all paid: bad (9.0/1.0)
| credit_history = existing paid
|   other_parties = none
|     savings_status = <100
|       existing_credits <= 1
|         purpose = new car: bad (17.0/4.0)
|         purpose = used car: good (3.0/1.0)
|         purpose = furniture/equipment: good (22.0/11.0)
|         purpose = radio/tv: good (18.0/8.0)
|         purpose = domestic appliance: bad (2.0)
|         purpose = repairs: bad (1.0)
|         purpose = education: bad (5.0/1.0)
|         purpose = vacation: bad (0.0)
|         purpose = retraining: bad (0.0)
|         purpose = business: good (3.0/1.0)
|         purpose = other: bad (0.0)
|       existing_credits > 1: bad (5.0)
|       savings_status = 100<=X<500: bad (8.0/3.0)
|       savings_status = 500<=X<1000: good (1.0)
```

```

| | | | savings_status = >=1000: good (2.0)
| | | | savings_status = no known savings
| | | | job = unemp/unskilled non res: bad (0.0)
| | | | job = unskilled resident: good (2.0)
| | | | job = skilled
| | | | | own_telephone = none: bad (4.0)
| | | | | own_telephone = yes: good (3.0/1.0)
| | | | | job = high qualif/self emp/mgmt: bad (3.0/1.0)
| | | other_parties = co applicant: good (4.0/2.0)
| | | other_parties = guarantor: good (8.0/1.0)
| | credit_history = delayed previously: bad (7.0/2.0)
| | credit_history = critical/other existing credit: good (38.0/10.0)
| foreign_worker = no: good (12.0/2.0) checking_status = 0<=X<200
| other_parties = none
| | credit_history = no credits/all paid
| | | other_payment_plans = bank: good (2.0/1.0)
| | | other_payment_plans = stores: bad (0.0)
| | | other_payment_plans = none: bad (7.0)
| | credit_history = all paid: bad (10.0/4.0)
| | credit_history = existing paid
| | | credit_amount <= 8858: good (70.0/21.0)
| | | credit_amount > 8858: bad (8.0)
| | credit_history = delayed previously: good (25.0/6.0)
| | credit_history = critical/other existing credit: good (26.0/7.0)
| other_parties = co applicant: bad (7.0/1.0)
| other_parties = guarantor: good (18.0/4.0)
| checking_status = >=200: good (44.0/9.0)
| checking_status = no checking
| | other_payment_plans = bank: good (30.0/10.0)
| | other_payment_plans = stores: good (12.0/2.0)
| | other_payment_plans = none
| | | credit_history = no credits/all paid: good (4.0)
| | | credit_history = all paid: good (1.0)
| | | credit_history = existing paid
| | | | existing_credits <= 1: good (92.0/7.0)
| | | | existing_credits > 1
| | | | | installment_commitment <= 2: bad (4.0/1.0)
| | | | | installment_commitment > 2: good (5.0)
| | | credit_history = delayed previously: good (22.0/6.0)
| | | credit_history = critical/other existing credit: good (92.0/3.0)

```

Size of the tree : 64

Time taken to build model: 0.49 seconds

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	725	72.5	%
Incorrectly Classified Instances	275	27.5	%
Kappa statistic	0.2786		
Mean absolute error		0.3331	
Root mean squared error	0.4562		
Relative absolute error		79.2826 %	
Root relative squared error		99.5538 %	
Total Number of Instances	1000		

12. (Extra Credit): How can you convert a Decision Trees into "if-then-else rules".

Make up your own small Decision Tree consisting of 2-3 levels and convert it into a set of rules. There also exist different classifiers that output the model in the form of rules - one such classifier in Weka is rules.PART, train this model and report the set of rules obtained. Sometimes just one attribute can be good enough in making the decision, yes, just one ! Can you predict what attribute that might be in this dataset? OneR classifier uses a single attribute to make decisions (it chooses the attribute based on minimum error). Report the rule obtained by training a one R classifier. Rank the performance of j48, PART and oneR.

AIM: Converting the decision tree to the different set of rules based on the different levels in the dataset.

PROCEDURE for J48:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “credit-g.arff”.
- 6) Select some of the attributes from attributes list
- 7) Go to Classify tab.

- 8) Choose Classifier “Trees Rules”
- 9) Select “J48”.
- 10) Select Test options “Use training set”
- 11) If need select attribute.
- 12) Now start weka.
- 13) Now we can see the output details in the Classifier output.
- 14) Right click on the result list and select ” visualize tree “option .

Procedure for “OneR”:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “credit-g.arff”.
- 6) Select some of the attributes from attributes list
- 7) Go to Classify tab.
- 8) Choose Classifier “Rules”
- 9) Select “OneR”.
- 10) Select Test options “Use training set”
- 11) If need select attribute.
- 12) Now start weka.
- 13) Now we can see the output details in the Classifier output.

Procedure for “PART”:

- 1) Given the credit database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “credit-g.arff”.

- 6) Select some of the attributes from attributes list
- 7) Go to Classify tab.
- 8) Choose Classifier “Rules”.
- 9) Select “PART”.
- 10) Select Test options “Use training set”
- 11) if need select attribute.
- 12) Now start weka.
- 13) Now we can see the output details in the Classifier output.

Attribute relevance with respect to the class – relevant attribute (science) IF accounting=1
THEN class=A (Error=0, Coverage = 7 instance)

IF accounting=0 THEN class=B (Error=4/13, Coverage = 13 instances)

Converting Decision trees into “IF-THEN-ELSE” rules using rules.PART classifier:-

PART decision list

outlook = overcast: yes (4.0)

windy = TRUE: no (4.0/1.0)

outlook = sunny: no (3.0/1.0)

: yes (3.0)

Number of Rules : 4

Yes, sometimes just one attribute can be good enough in making the decision.

In this dataset (Weather), Single attribute for making the decision is “outlook”

outlook:

sunny -> no

overcast -> yes

rainy -> yes

(10/14 instances correct)

With respect to the time, the oneR classifier has higher ranking and J48 is in 2nd place and PART gets 3

rd place.

J48 PART oneR

TIME (sec) 0.12 0.14 0.04

RANK II III I

But if you consider the accuracy, The J48 classifier has higher ranking, PART gets second place and oneR

gets 1st place

J48 PART oneR

ACCURACY (%) 70.5 70.2% 66.8%

RANK I II III

Experiment 2:

Schema Definition