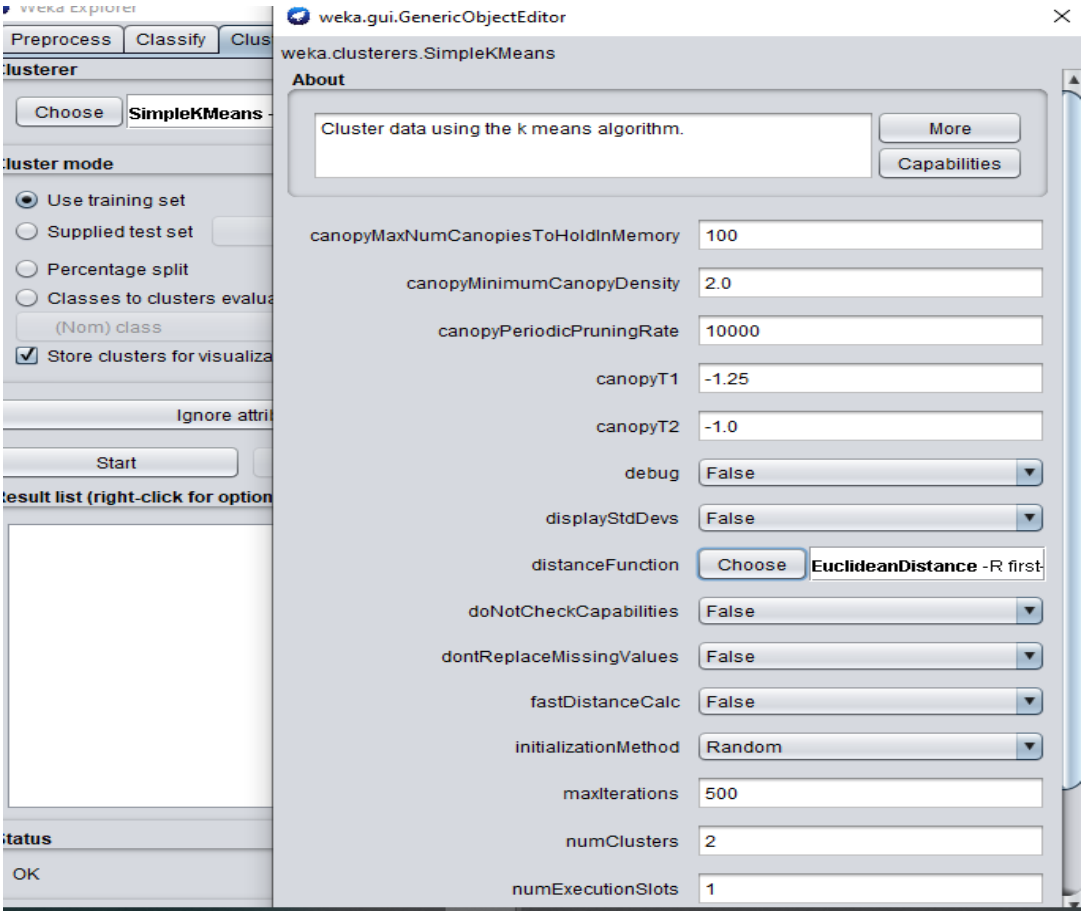
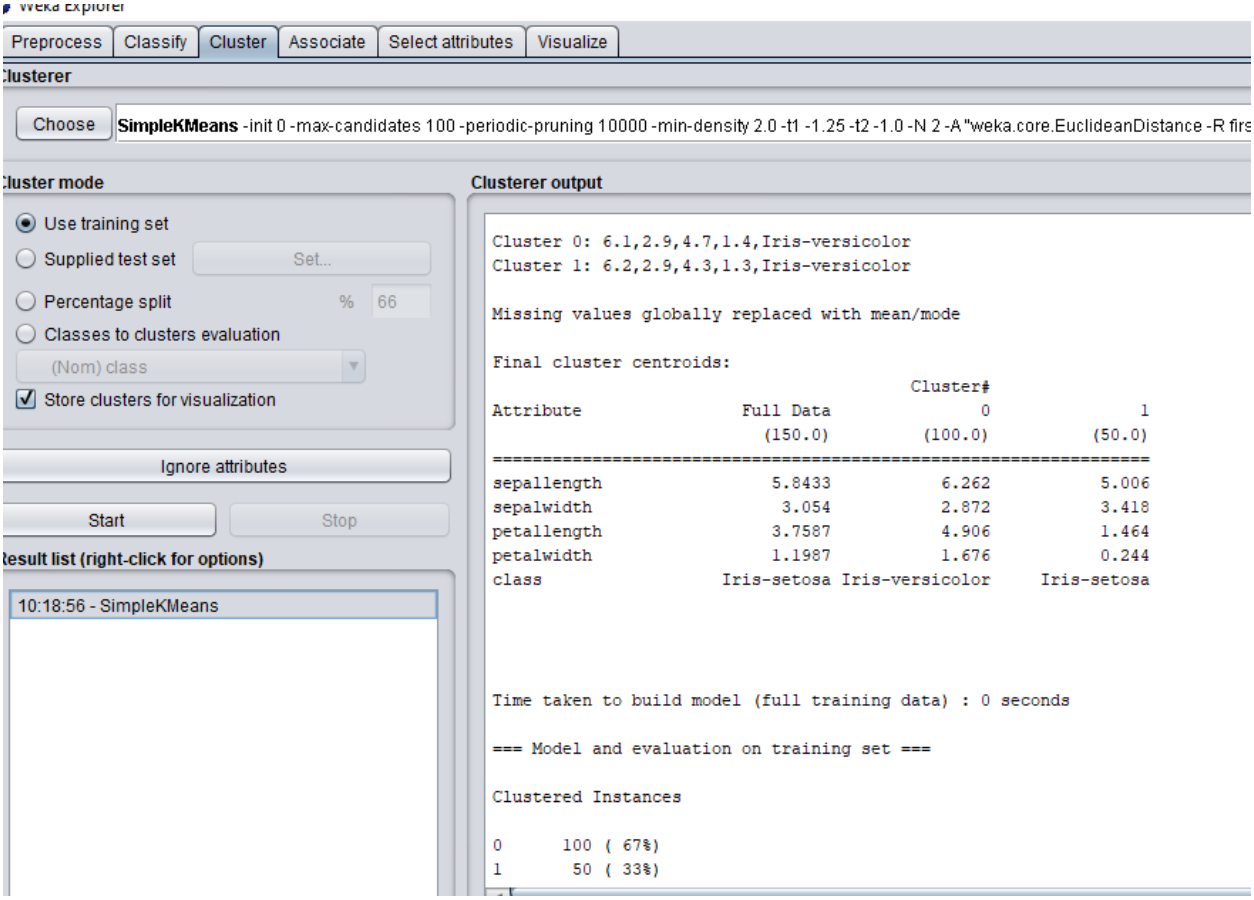
**6. Demonstrate clustering algorithms for given data sets and report the model.**
**STEPS:**

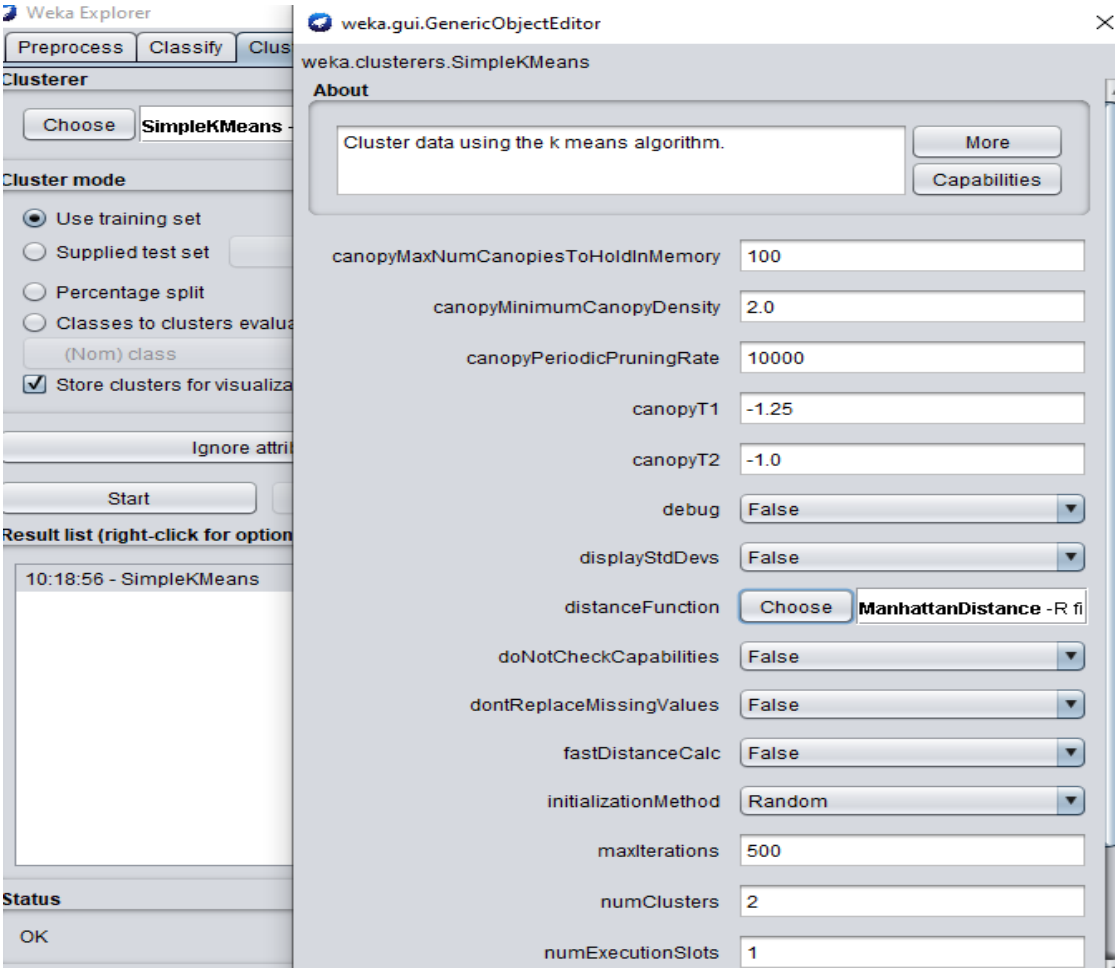
1) Open WEKA 3.8 Tool.
2) Select EXPLORER present in Applications.
3) Select Preprocess Tab.
4) Go to OPEN file and browse the file that is already stored in the system “iris. arff”.
5) Go to Cluster tab.
6) Choose “SimpleKmeans” in Classifiertab
7) Click on “simpleKmeans” properties
8) Choose “Distance function” as “EuclideanDistance”
9) Save the changes, click OK and click on Start
10) We can see the output details in the Classifier output
11) Likewise, choose the required distance function for other outputs.

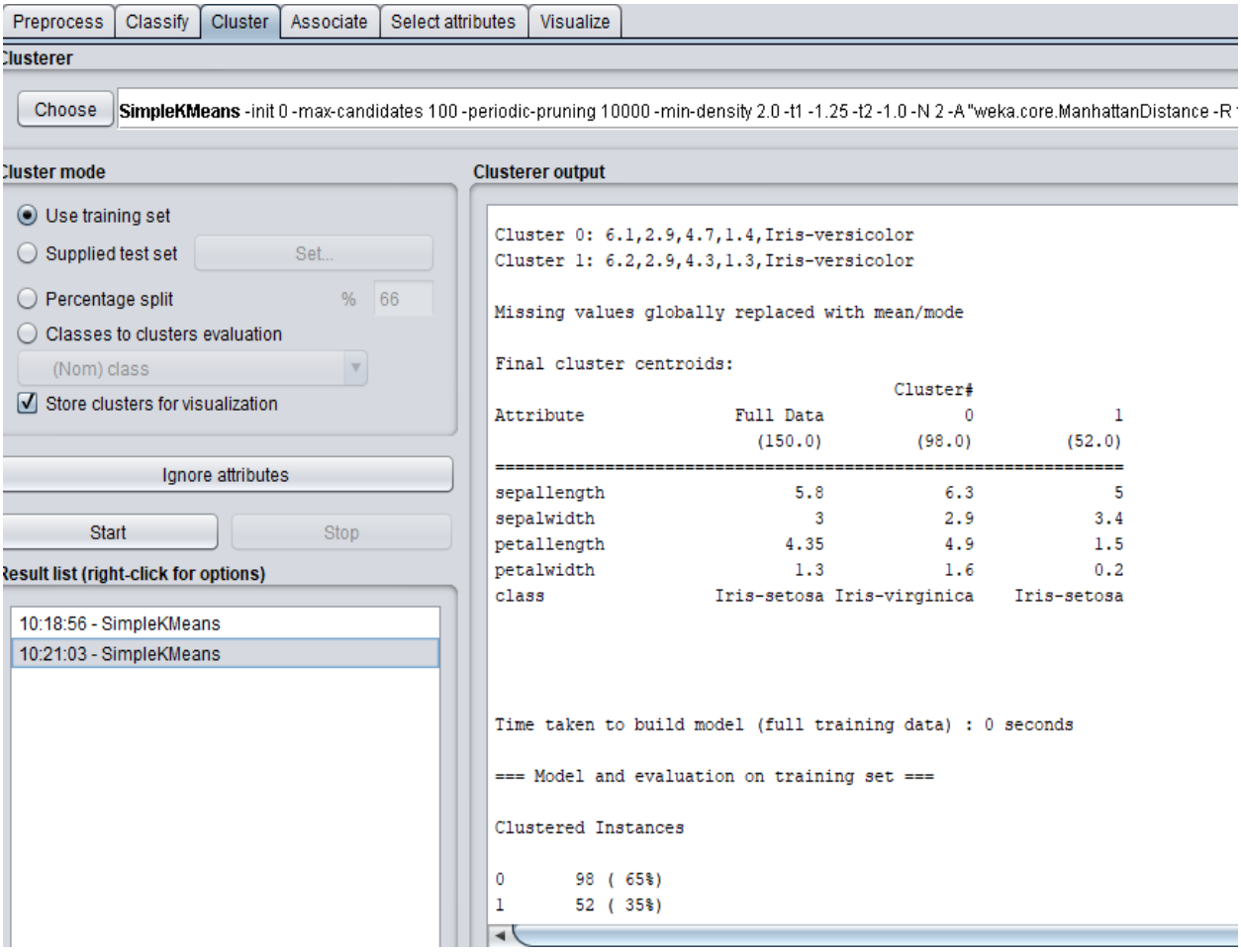
**Euclidean Distance:**




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R firs
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
10:18:56 - SimpleKMeans
Clusterer output
Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1
(150.0) (100.0) (50.0)
sepallength 5.8433 6.262 5.006
sepalwidth 3.054 2.872 3.418
petallength 3.7587 4.906 1.464
petalwidth 1.1987 1.676 0.244
class Iris-setosa Iris-versicolor Iris-setosa
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 100 ( 67%)
1 50 ( 33%)

**Manhattan Distance:**

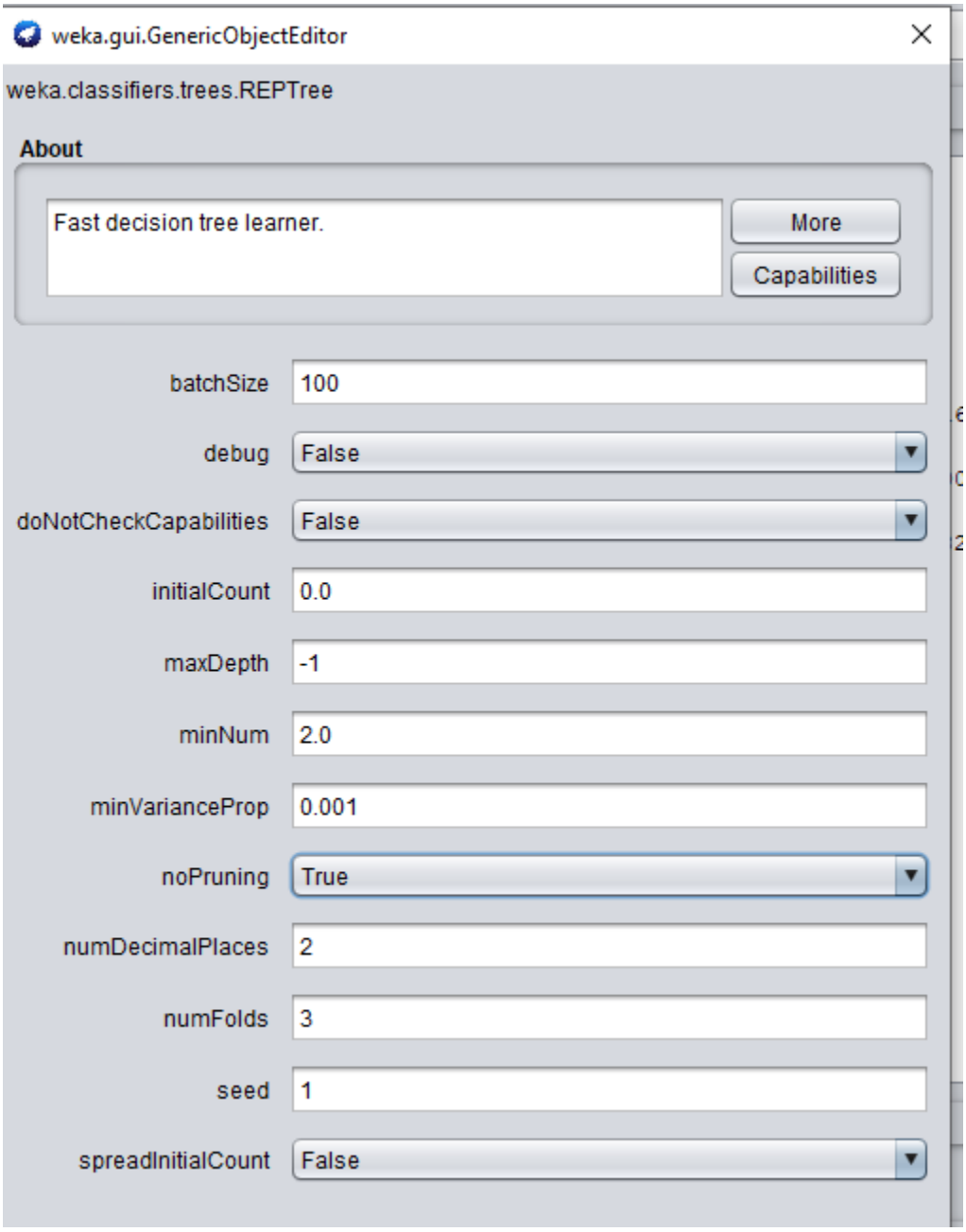



Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.ManhattanDistance -R
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
10:18:56 - SimpleKMeans
10:21:03 - SimpleKMeans
Clusterer output
Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1
(150.0) (98.0) (52.0)
sepallength 5.8 6.3 5
sepalwidth 3 2.9 3.4
petallength 4.35 4.9 1.5
petalwidth 1.3 1.6 0.2
class Iris-setosa Iris-virginica Iris-setosa
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 98 ( 65%)
1 52 ( 35%)

## LABCYCLE-4

**1.)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?**

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose creditg data set and open file.
8. Choose Classifier “Tree”
9. Select “REPTree.
10. Select Test options “Cross Validation”
11. Right click on the text box besides choose button, select show properties
12. Now change unprune mode “false” to “true”. .
13. Now start weka.
14. Now we can see the output details in the Classifier output.
15. Right click on the result list and select ” visualize tree “option.


weka.gui.GenericObjectEditor
weka.classifiers.trees.REPTree
About
Fast decision tree learner.
More
Capabilities
batchSize
100
debug
False
doNotCheckCapabilities
False
initialCount
0.0
maxDepth
-1
minNum
2.0
minVarianceProp
0.001
noPruning
True
numDecimalPlaces
2
numFolds
3
seed
1
spreadInitialCount
False

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose **REPTree** -M 2 -V 0.001 -N 3 -S 1 -L -1 -P -I 0.0

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start | Stop

**Result list (right-click for options)**

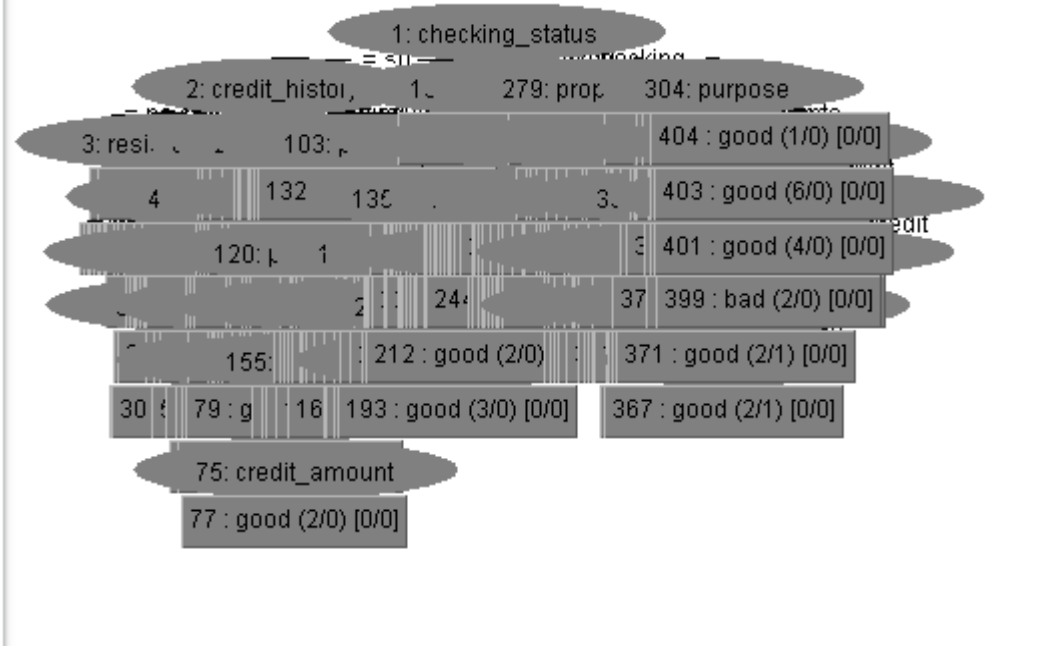
09:30:50 - rules.OneR
09:41:16 - trees.REPTree

**Classifier output**

```
Time taken to build model: 0.05 seconds

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

Correctly Classified Instances         720               72      %
Incorrectly Classified Instances       280               28      %
Kappa statistic                          0.3164
Mean absolute error                      0.3
Root mean squared error                  0.5008
Relative absolute error                 71.4    %
Root relative squared error            109.2825 %
Total Number of Instances              1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.819    0.510    0.789      0.819    0.804      0.317    0.672     0.786     good
                 0.490    0.181    0.536      0.490    0.512      0.317    0.672     0.459     bad
Weighted Avg.    0.720    0.411    0.713      0.720    0.716      0.317    0.672     0.688

=== Confusion Matrix ===

   a   b   <-- classified as
 573 127 |   a = good
 153 147 |   b = bad
```

Weka Classifier Tree Visualizer: 09:41:16 - trees.REPTree (german_credit)

**Tree View**



As the output given by the classifier, in the previous question the accuracy given by J48 Algorithm shows 85.5% correct data records when compared to REP Tree algorithm which shows 72% correct data records. So J48 Algorithm is more accurate than REP Tree.

**2.(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.**

1.Open WEKA 3.8 tool.
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 “creditg.arff”.
5. Go to Classify tab.
6. Choose Classifier “rules”
7. Select any of the classification Model
8. Select Test options “Use training set”
9. Now select “start” button.
10. Now we can see the output details in the Classifier output.

**PROCEDURE for “ONER”:**

**Steps:**

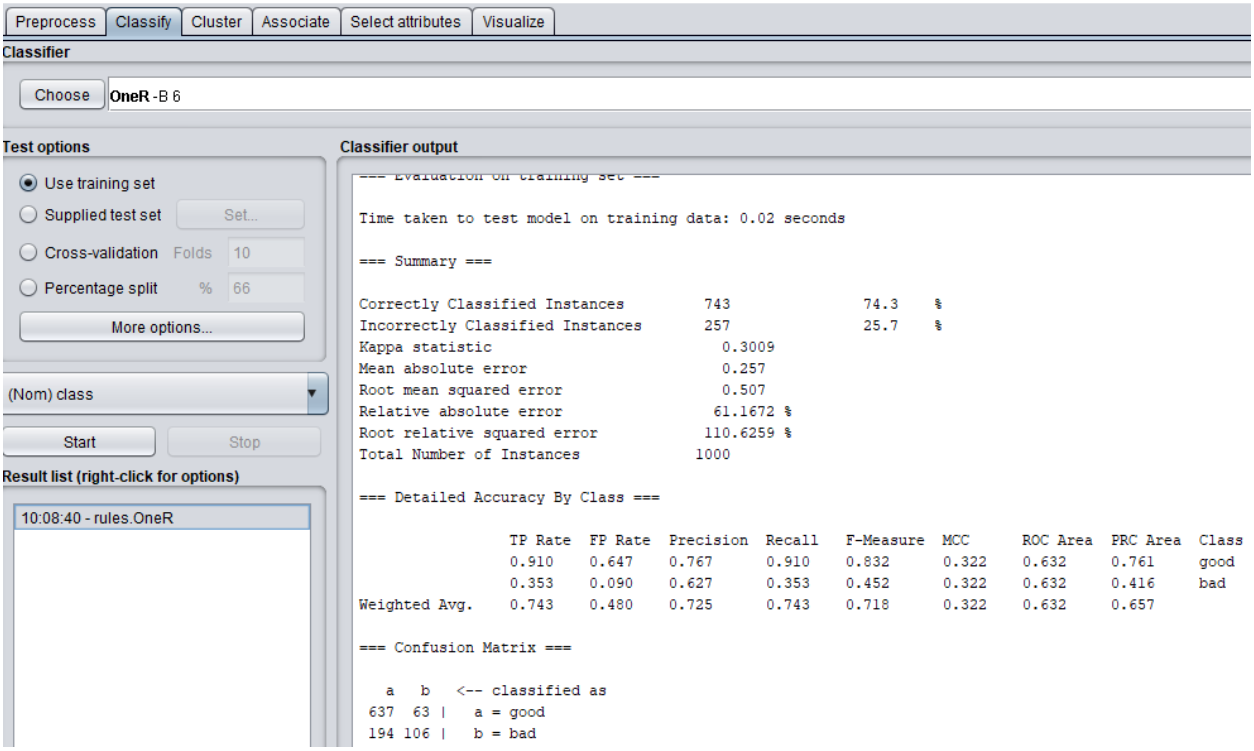
1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose “credit-g.arff” data set and open file.
8. Choose Classifier “Rules”
9. Select “OneR”.
10. Select Test 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.

credit_amount:

| | |
|---|---|
| < 718.0 | -> good |
| < 759.5 | -> bad |
| < 883.0 | -> good |
| < 922.0 | -> bad |
| < 938.0 | -> good |
| < 979.5 | -> bad |
| < 1206.5 | -> good |
| < 1223.5 | -> bad |
| < 1267.5 | -> good |
| < 1286.0 | -> bad |
| < 1821.5 | -> good |
| < 1865.5 | -> bad |
| < 3913.5 | -> good |
| < 3969.0 | -> bad |
| < 4049.5 | -> good |
| < 4329.5 | -> bad |
| < 4726.0 | -> good |
| < 5024.0 | -> bad |
| < 6322.5 | -> good |
| < 6564.0 | -> bad |
| < 6750.0 | -> good |
| < 6917.5 | -> bad |
| < 7760.5 | -> good |
| < 8109.5 | -> bad |
| < 9340.5 | -> good |
| < 10331.5 | -> bad |
| < 11191.0 | -> good |
| >= 11191.0 | -> bad |

(743/1000 instances correct)




**PROCEDURE for "PART":**

**Steps:**

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose "credit-g.arff" data set and open file.
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.

**Rules:**
**PART decision list**
---

checking_status = no checking AND
other_payment_plans = none AND
credit_history = critical/other existing credit: good (134.0/3.0)

checking_status = no checking AND
existing_credits <= 1 AND
other_payment_plans = none AND
purpose = radio/tv: good (49.0/2.0)

checking_status = no checking AND
foreign_worker = yes AND
employment = 4<=X<7: good (35.0/2.0)

foreign_worker = no AND
personal_status = male single: good (21.0)

checking_status = no checking AND
purpose = used car AND
other_payment_plans = none: good (23.0)

duration <= 15 AND
other_parties = guarantor: good (22.0/1.0)

duration <= 11 AND
credit_history = critical/other existing credit: good (29.0/3.0)

checking_status = >=200 AND
num_dependents <= 1 AND
property_magnitude = car: good (20.0/3.0)

checking_status = no checking AND
property_magnitude = real estate AND
other_payment_plans = none AND
age > 23: good (25.0)

savings_status = >=1000 AND
property_magnitude = real estate: good (10.0)

savings_status = 500<=X<1000 AND

employment = >=7: good (13.0/1.0)

credit_history = no credits/all paid AND
housing = rent: bad (9.0)

savings_status = no known savings AND
checking_status = 0<=X<200 AND
existing_credits > 1: good (9.0)

checking_status = >=200 AND
num_dependents <= 1 AND
property_magnitude = life insurance: good (9.0)

installment_commitment <= 2 AND
other_parties = co applicant AND
existing_credits > 1: bad (5.0)

installment_commitment <= 2 AND
credit_history = delayed previously AND
existing_credits > 1 AND
residence_since > 1: good (14.0/3.0)

installment_commitment <= 2 AND
credit_history = delayed previously AND
existing_credits <= 1: good (9.0)

duration > 30 AND
savings_status = 100<=X<500: bad (13.0/3.0)

credit_history = all paid AND
other_parties = none AND
other_payment_plans = bank: bad (16.0/5.0)

duration > 30 AND
savings_status = no known savings AND
num_dependents > 1: good (5.0)

duration > 30 AND
credit_history = delayed previously: bad (9.0)

duration > 42 AND
savings_status = <100 AND
residence_since > 1: bad (28.0/3.0)

purpose = used car AND
credit_amount <= 8133 AND
existing_credits > 1: good (11.0)

purpose = used car AND
credit_amount > 8133: bad (8.0/1.0)

purpose = used car AND
employment = 1<=X<4: good (7.0)

purpose = used car: good (16.0/3.0)

purpose = furniture/equipment AND
other_payment_plans = stores: good (8.0)

credit_history = all paid AND
other_parties = none AND
other_payment_plans = none: bad (10.0)

purpose = business AND
residence_since <= 1: good (9.0)

other_payment_plans = stores AND
purpose = radio/tv AND
personal_status = male single: bad (6.0/1.0)

purpose = radio/tv AND
employment = >=7 AND
num_dependents <= 1: good (20.0/1.0)

installment_commitment <= 3 AND
purpose = furniture/equipment AND
other_parties = none AND
own_telephone = yes: good (19.0/3.0)

checking_status = no checking AND
savings_status = no known savings AND
personal_status = male single: good (11.0/1.0)

checking_status = 0<=X<200 AND
employment = 4<=X<7 AND
personal_status = male single AND
residence_since > 2: good (9.0)

purpose = other: good (5.0/1.0)

installment_commitment <= 2 AND
foreign_worker = yes AND
credit_history = existing paid AND
residence_since > 1 AND
other_parties = none AND
other_payment_plans = none AND
housing = rent AND
installment_commitment <= 1: good (9.0)

housing = rent AND
other_payment_plans = none AND
purpose = new car: bad (13.0/2.0)

other_payment_plans = stores AND
property_magnitude = life insurance: bad (4.0/1.0)

other_payment_plans = bank AND
other_parties = none AND
housing = rent: bad (7.0/1.0)

installment_commitment > 3 AND
existing_credits <= 1 AND
savings_status = <100 AND
credit_history = existing paid AND
purpose = new car: bad (17.0/5.0)

checking_status = >=200 AND
job = unskilled resident: bad (5.0)

duration <= 15 AND
property_magnitude = real estate: good (38.0/8.0)

foreign_worker = yes AND
property_magnitude = real estate AND
other_payment_plans = none AND
other_parties = none AND
duration <= 33 AND
own_telephone = yes: bad (7.0)

foreign_worker = yes AND
checking_status = <0 AND
purpose = education: bad (9.0/1.0)

foreign_worker = yes AND
purpose = education AND
checking_status = 0<=X<200: good (5.0)

foreign_worker = yes AND
checking_status = <0 AND
savings_status = 100<=X<500 AND
num_dependents <= 1: bad (6.0/1.0)

foreign_worker = yes AND
savings_status = >=1000 AND
checking_status = <0: good (4.0)

foreign_worker = yes AND
savings_status = 100<=X<500 AND
personal_status = male single: good (10.0/2.0)

foreign_worker = yes AND
existing_credits > 2: good (11.0/2.0)

foreign_worker = yes AND
other_parties = guarantor AND
other_payment_plans = none AND
existing_credits <= 1: good (6.0)

foreign_worker = yes AND
num_dependents > 1 AND
personal_status = male single AND
savings_status = <100 AND
job = skilled AND
duration > 16: bad (7.0)

foreign_worker = yes AND
other_parties = guarantor AND
purpose = radio/tv: bad (3.0)

foreign_worker = yes AND
credit_history = critical/other existing credit AND
job = unskilled resident: bad (6.0)

foreign_worker = yes AND
credit_history = no credits/all paid AND
housing = own: good (9.0/4.0)

foreign_worker = yes AND
credit_history = delayed previously AND
savings_status = <100 AND
existing_credits <= 1: bad (5.0)

foreign_worker = yes AND
credit_history = delayed previously AND
num_dependents <= 1: good (5.0)

foreign_worker = yes AND
credit_history = delayed previously AND
job = skilled: good (3.0/1.0)

foreign_worker = yes AND
credit_history = critical/other existing credit AND
other_parties = none AND
housing = own AND
savings_status = <100 AND
existing_credits > 1 AND
installment_commitment > 2 AND
credit_amount > 2181: bad (6.0)

foreign_worker = yes AND
credit_history = critical/other existing credit AND
other_payment_plans = bank: bad (5.0/1.0)

foreign_worker = yes AND
credit_history = critical/other existing credit AND
job = skilled AND
employment = 1<=X<4 AND
residence_since <= 3: good (6.0/1.0)

foreign_worker = yes AND
credit_history = critical/other existing credit: good (17.0/5.0)

foreign_worker = yes AND
credit_history = existing paid AND
checking_status = <0 AND
other_payment_plans = none AND
job = skilled AND
purpose = new car: bad (7.0/1.0)

foreign_worker = yes AND

credit_history = existing paid AND
checking_status = no checking AND
duration <= 30 AND
residence_since > 1 AND
own_telephone = yes: good (4.0)

foreign_worker = yes AND
credit_history = existing paid AND
savings_status = no known savings: bad (18.0/6.0)

foreign_worker = yes AND
credit_history = existing paid AND
checking_status = <0 AND
other_payment_plans = bank AND
housing = own: bad (3.0/1.0)

foreign_worker = yes AND
credit_history = existing paid AND
checking_status = <0 AND
other_payment_plans = none AND
purpose = radio/tv AND
job = skilled: bad (7.0/1.0)

foreign_worker = yes AND
credit_history = existing paid AND
existing_credits <= 1 AND
purpose = radio/tv AND
age > 22: good (11.0/1.0)

foreign_worker = yes AND
credit_history = existing paid AND
existing_credits <= 1 AND
installment_commitment > 3: bad (27.0/8.0)

foreign_worker = yes AND
credit_history = existing paid AND
other_payment_plans = bank: good (5.0/1.0)

foreign_worker = yes AND
credit_history = existing paid AND
own_telephone = yes AND
installment_commitment > 2: bad (4.0)

foreign_worker = yes AND

credit_history = existing paid AND
existing_credits <= 1 AND
employment = 1<=X<4 AND
personal_status = female div/dep/mar AND
credit_amount > 1474: good (5.0/1.0)

foreign_worker = yes AND
credit_history = existing paid AND
purpose = repairs: good (4.0/1.0)

foreign_worker = yes AND
credit_history = existing paid AND
purpose = furniture/equipment AND
property_magnitude = real estate: good (3.0)

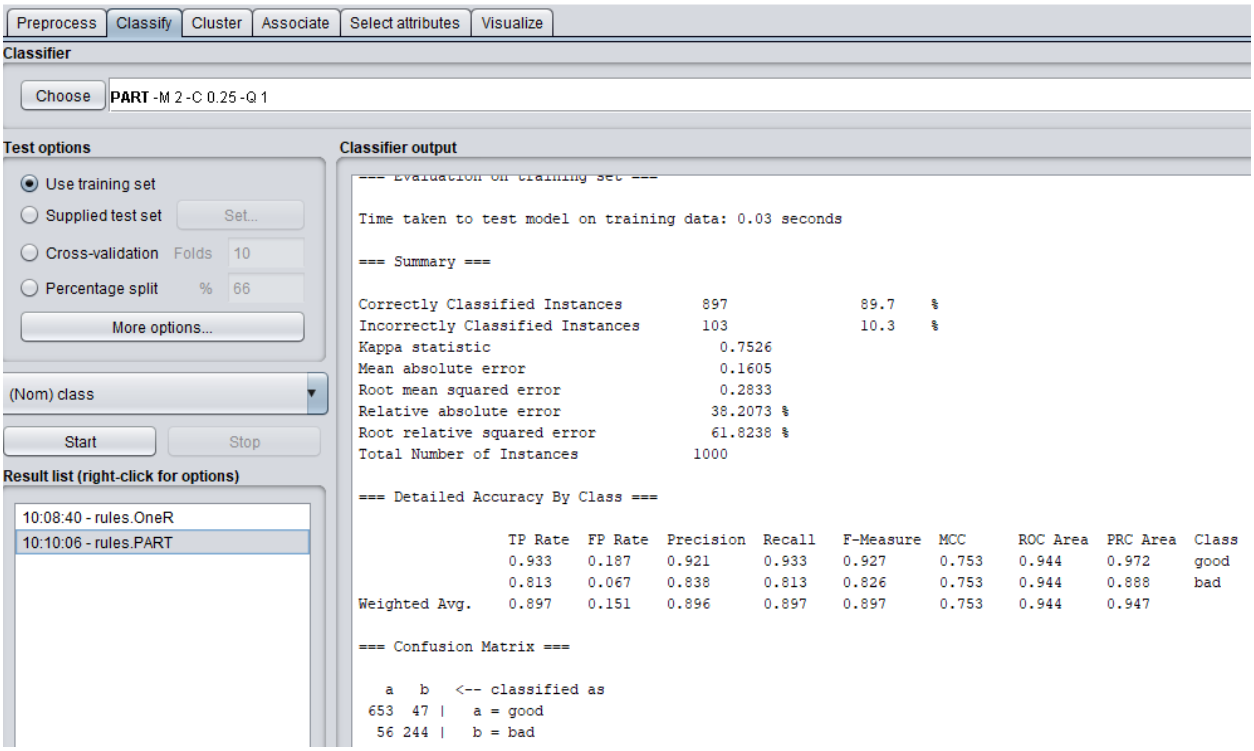
foreign_worker = yes AND
credit_history = existing paid AND
housing = own AND
property_magnitude = life insurance: bad (8.0/3.0)

num_dependents <= 1 AND
foreign_worker = yes AND
credit_history = existing paid AND
checking_status = no checking: good (4.0)

credit_history = existing paid AND
housing = own AND
residence_since > 1: bad (8.0/2.0)

existing_credits <= 1 AND
num_dependents <= 1: good (8.0/2.0)

: bad (5.0)




**J48:**

J48 pruned tree

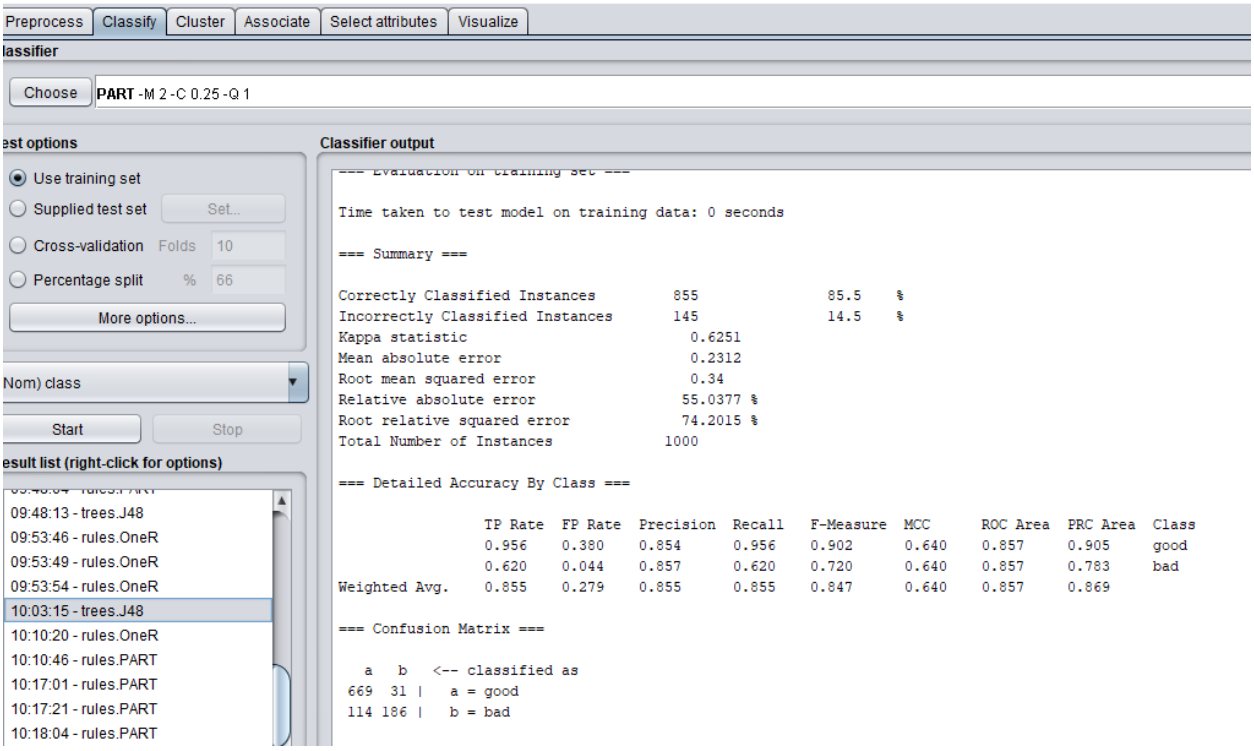
---

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)
```




| Algorithm | Correctly Classified Instances (%) | Incorrectly Classified Instances (%) | Accuracy |
|---|---|---|---|
| J48 | 85.5 | 14.5 | 85.5% |
| OneR | 74.3 | 25.7 | 74.3% |
| PART | 89.7 | 10.3 | 89.7% |

3) **Case Study: Hospital Management System**

**Design a Hospital Management system data warehouse (TARGET) consists of Patient, Medicine, Supplier, Time as dimensions. 'NO.OF.UNITS', UNIT PRICE as measures.**

**Assume the Relational database (SOURCE) table schemas as follows:**

**TIME (day, month, year),**

**PATIENT (patient_name, Age, Address, etc.,)**

**MEDICINE ( Medicine_Brand_name, Drug_name, Supplier, no_units, Uinit_Price, etc.,) SUPPLIER :( Supplier_name, Medicine_Brand_name, Address, etc., )**

**If each Dimension has 6 levels, decide the levels and hierarchies,Assume the level names suitably.**

**Design the Hospital Management system data warehouse using all schemas.Give the example 4-D cube with assumption names.**

Data Warehouse consists Dimension Table and Fact Table. Remember the following

Dimension The dimension object (Dimension):

_ Name

_ Attributes (Levels) , with one primary key

_ Hierarchies

One time dimension is must. About Levels and Hierarchies

Dimension objects (dimension) consist of a set of levels and a set of hierarchies defined over those levels. The levels represent levels of aggregation. Hierarchies describe parent-child relationships among a set of levels.

For example, a typical calendar dimension could contain five levels. Two hierarchies can be defined on these levels:

H1: YearL > QuarterL > MonthL > WeekL > DayL H2:YearL > WeekL > DayL

The hierarchies are described from parent to child, so that Year is the parent of Quarter, Quarter

the parent of Month, and so forth.

About Unique Key Constraints

When you create a definition for a hierarchy, Warehouse Builder creates an identifier key for each

level of the hierarchy and a unique key constraint on the lowest level (Base Level)

Design a Hospital Management system data warehouse (TARGET) consistig of Dimensions Patient, Medicine, Supplier, Time. Where measures are _ NO UNITS‘, UNIT PRICE.
Assume the Relational database (SOURCE) table schemas as follows TIME (day, month, year),
PATIENT (patient_name, Age, Address, etc.,)
MEDICINE ( Medicine_Brand_name, Drug_name, Supplier, no_units, Uinit_Price, etc.,)
SUPPLIER :( Supplier_name, Medicine_Brand_name, Address, etc., )
If each Dimension has 6 levels, decide the levels and hierarchies, Assume the level names suitably.

Design the Hospital Management system data warehouse using all schemas. Give the example 4-D cube with assumption names.

Data Warehouse consists Dimension Table and Fact Table

REMEMBER The following

Dimension

The dimension object

(Dimension)

_Name

_Attribute (Levels), with one primary key

_Hierarchies

One time dimension is must

About Levels and Hierarchies

Dimension objects (dimension) consist of set of levels and set of hierarchies defined over those levels. The levels represent levels of aggregation. Hierarchies describe parent-child relationships among a set of levels.

For example, a typical calendar dimension could contain five levels. Two hierarchies can be defined on these levels:

H1: YearL>QuarterL>Month>WeekL>DayL

H2: YearL>WeekL>DayL

The hierarchies are described from parent to child, So that Year is the parent of Quarter ,Quarter the parent of Month, and so forth.

About Unique Key Constraints

When you create a definition for hierarchy, warehouse Builder creates an identifier key for each levels of the hierarchy and a Unique Key constraint on the lowest level (Base Level)

Design a Hospital Management system data warehouse (TARGET) Consists of Dimensions patient, medicine, supplier, Time. Where measures are 'NO UNITS', UNIT PRICE.
Assume the Relational database (SOURCE) table schemas as
follows TIME (day, month, year).
PATIENT (patient-name, Age, Address, etc,.)
MEDICINE (Medicine_Brand_name, Drug_name, supplier, no_units, Unit_Price, etc.,)
SUPPLIER :( Supplier_name, Medicine_Brand_Name, Address, etc.,)
If each Dimension has 6 levels, decide the levels and hierarchies, Assume the level names suitably

The HMS consists of four dimensions named TIME, PATIENT, MEDICINE and SUPPLIER Step 1:- Each dimension should describe with the help of name attributes. One attribute as a primary Key and concept hierarchy with at least 6 levels
Step 2:-Each Dimension is created as a relational database with appropriate dataset.
Step 3:- The HMS cube is represented as a lattices of cuboids with 5 levels(starting from 0 level to Base level.).
Step 4:- The measures of HMS has to be identified as a attribute of fact table in the given problem. The measures are number of units and unit price.
Step 5:- Design star schema with the help of the above dimension and fact table description. If any refinement required then represent the data warehouse as snow flake schema.
Step 6:- Identify the other business (other system such as insurance) where HMS dimension is used as the local dimension for that system and represent it as snow flake schema.
Step 7:- Represent all concept hierarchy of HMS data warehouse with the help of star net query model (clearly specify at which level the each dimension is represented in data warehouse).
Step 8:-Express the schemas in the form of definition with the help of two language primitives one for queue definition and another for queue definition dimension. Step 9:-Express the concept hierarchy using DMQL.

**RESPONSIBILITIES:-**

- Design the mapping with changing dimension to keep track and historical data
- Design the mapping to load the data the data into the target based on primary, foreign key relationship
- Design a mapping to process the incremental change that exists is its source table.
- Design a mapping to remove duplicate source records.
- Designs a mapping into transforming are record into multiple data records.
- Design a mapping in getting the related values using data up aggregator.
- Design a mapping with multiple loading using file repository.

Finally look up for transformation which involved in the form of unit testing.

**IMPLEMENTATION:**

**STEP 1:-**

TIME:

àDimension name: time

àList of attributes :( day, month, year)

àPrimary key:time-key

PATIENT:

àDimension name: PATIENT.

àList of attributes : (patient name,age,address,etc,.

àPrimary key: patient key

àConcept hierarchy

Patient name

Disease name

Type of disease

Age

Address

MEDICINE:

àDimension name:MEDICINE

àList of attribute:(medicine-brand-name, drug name, supplier, no-units, unit-price,etc) àPrimary key: MEDICINE KEY

àConcept hierarchy:

Medicine name

Medicine-brand-name

Customer number

Number-units

Unit-price

Supplier

Drug name

SUPPLIER:

àDimension name: Supplier

àList of attribute:(Supplier-name, medicine-brand-name,address,etc,.)
àPrimary key: SUPPLIER KEY
àConcept hierarchy:

Supplier-name
Address
Medicine-brand-name
Shop number

**STEP2:-**

**TIME:**

Create table time (time-key number primary key, year number, month number, week number, day number);

**PATIENT:**

Create table patient (patient-number primary key, patient-type varchar2 (20), disease-name varchar22 (20), affected area varchar2 (20), time number);

**MEDICINE:**

Create table medicine (medicine-key number primary key, medicine-brand-name varchar2 (20),drug-name varchar2(20),Supplier varchar2 (20),no-units number, unit=price number);

**SUPPLIER:**

Create table supplier (supplier-key number primary key, supplier-name varchar2 (20), medicine-brand-
name varchar2 (20), address varchar2 (20));

INSERTING DATA

TIME:

Insert into time values (1, 1983.11,48,21) ;
1 row inserted
Insert into time values (2, 1993, 1, 2, 9);
1 row inserted

PATIENT:

Insert into patient values (101, in, cancer, advance,
brain, 10); 1 row inserted
Insert into patient values (102, out, fewer, normal,
non, 1); 1 row inserted

MEDICINE:

Insert into medicine values (101, ex, penciline, pradeep,
5, 50); 1 row inserted

Insert into medicine values (102, ex, penciline, raju, 10, 60); 1 row inserted

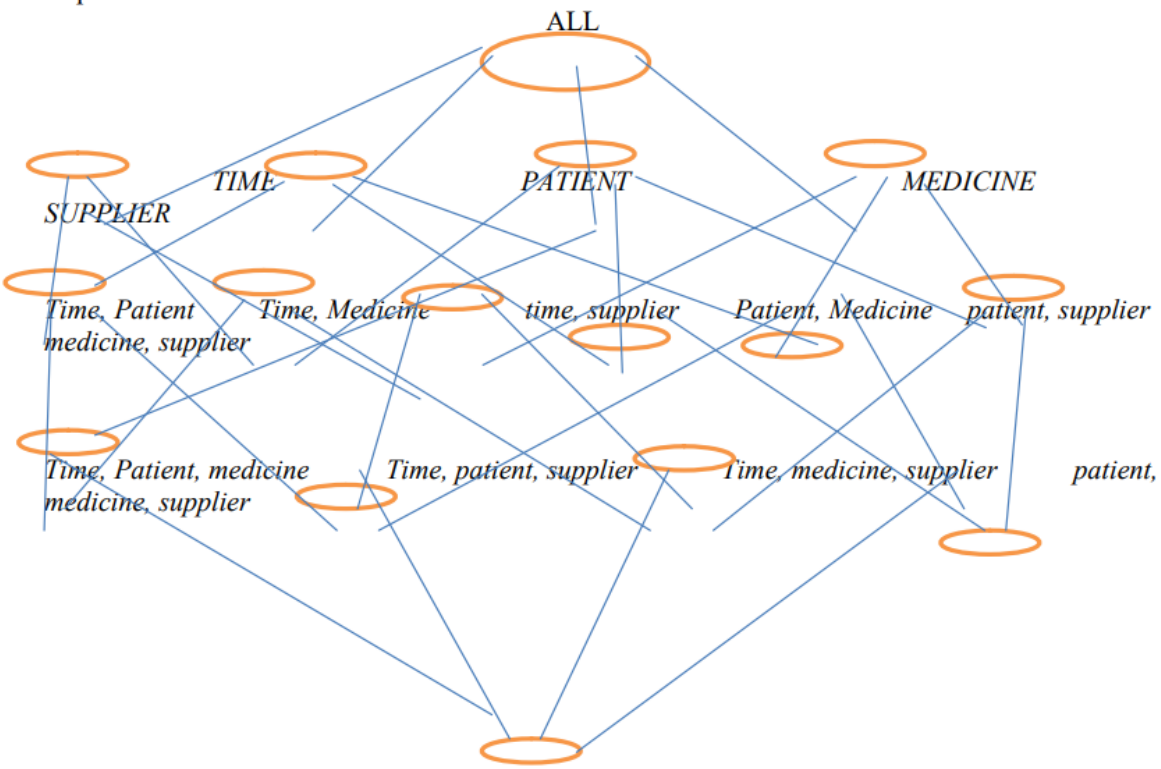
SUPPLIER:

Insert into supplier values (101, pradeep, ex, chittoor); 1 row inserted

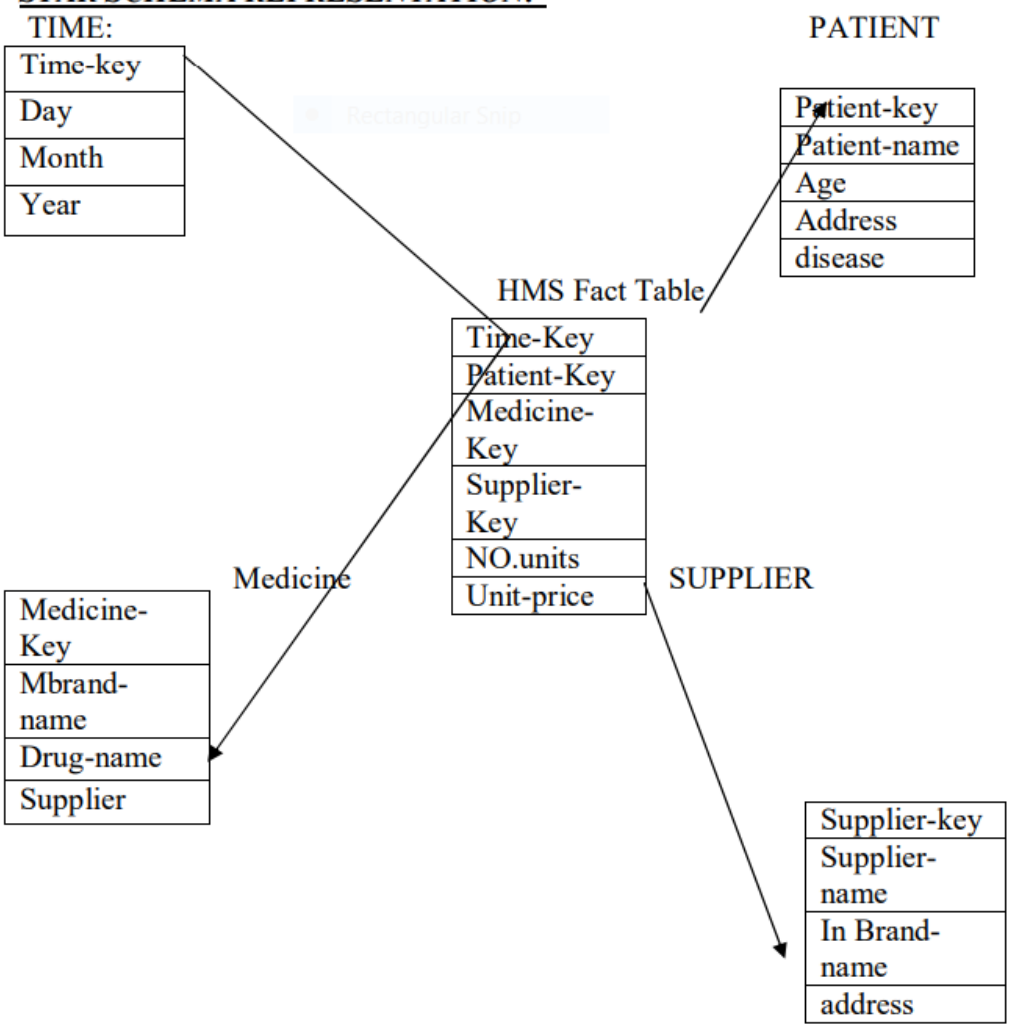
Insert into supplier values (102, raju, ex, Hyderabad); 1 row inserted

**STEP3:-**

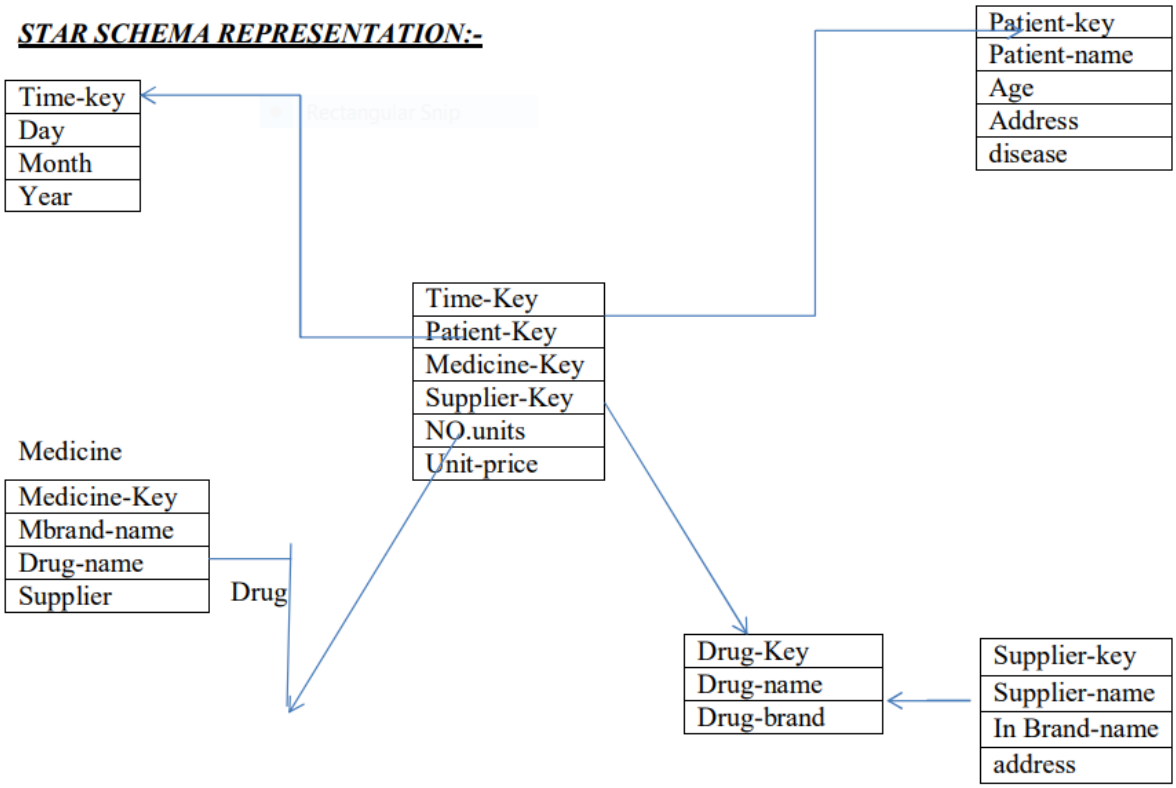
**R**epresentation of HMS cube in lattice of cuboid.



***STAR SCHEMA REPRESENTATION:-***

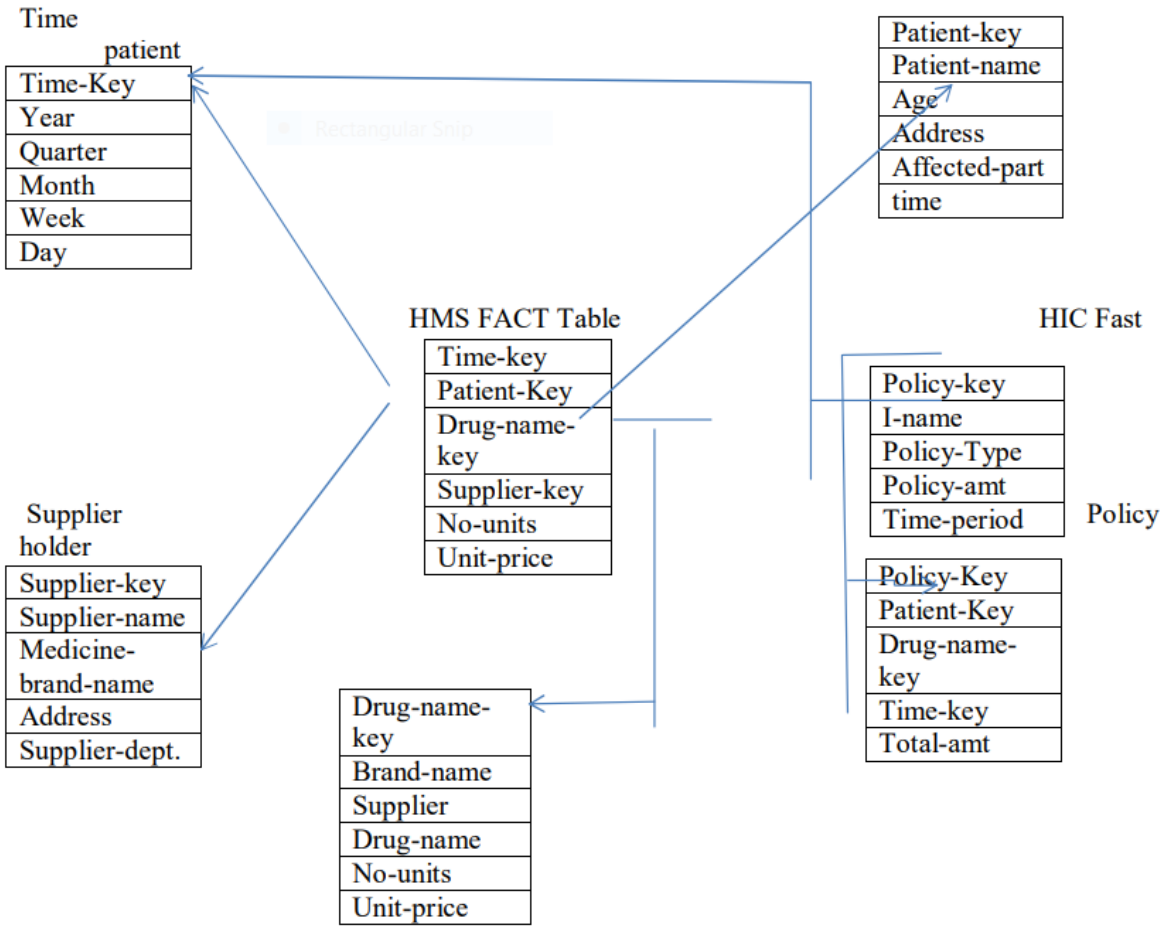


***STAR SCHEMA REPRESENTATION:-***



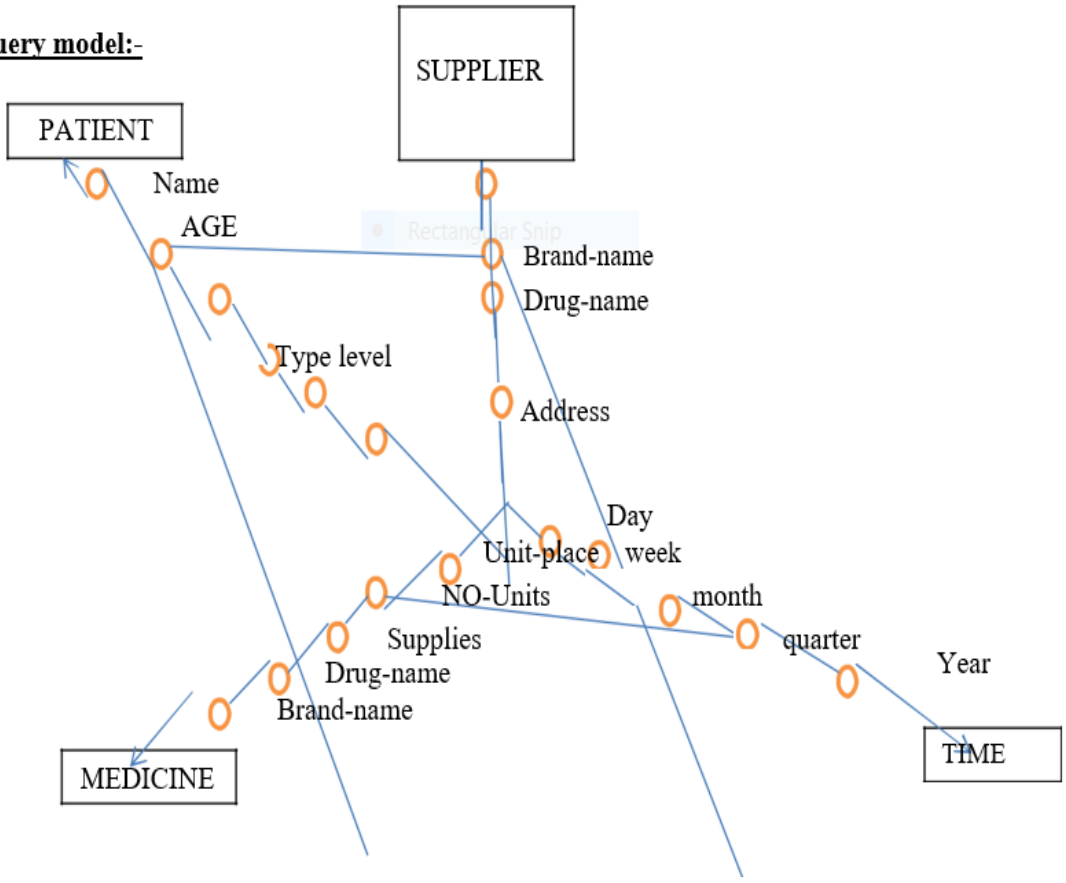

**Fact Constellation Schema Representation:-**

The health insurance company requires the dimensions like time, patient & medicine. These dimensions are already in hospital management system. So by using the Fact-constellation model without creating the dimensions once again, it can be reused for another cube.


Time
patient
Time-Key
Year
Quarter
Month
Week
Day
Patient-key
Patient-name
Age
Address
Affected-part
time
HMS FACT Table
Time-key
Patient-Key
Drug-name-key
Supplier-key
No-units
Unit-price
HIC Fast
Policy-key
I-name
Policy-Type
Policy-amt
Time-period
Policy
Supplier holder
Supplier-key
Supplier-name
Medicine-brand-name
Address
Supplier-dept.
Policy-Key
Patient-Key
Drug-name-key
Time-key
Total-amt
Drug-name-key
Brand-name
Supplier
Drug-name
No-units
Unit-price

**Step7:-**

**Start-net query model:-**




-

**Step8:-**

DMQL

Dmql for cube

Define cube hms (time, patient, medicine, supplier): No-units=count (*) Unit-price=sum (unit-price)/count (*);

Dmql for dimension

Define dimension

Define dimension time as (time-key, year, quarter, month, week, day)

Define dimension patient as (patient-key, patient-name, age, address, affected-part, time) Define dimension medicine as (drug-name-key, brand-name, supplier, drug-name, no-units,

unit price)

Define dimension supplier as(supplier-key, supplier name, medicine-brand-name, address, supplier-dept)

**STEP9:-**

HIERARCHY

Define hierarchy time-h on time as [day, week, month, quarter, year]

Define hierarchy sup-h as supplies as [address, drug-name, brand-name, supplier-name] Define hierarchy patient –h a patient as [affected –part, level, type, address, age, d-name]

Define hierarchy patient –h on medicine as [brand-name, drug-name, supplier, no-units, units price]