CS 539 Machine Learning

Project 2 - Decision Trees

Chris Winsor

Contents:

Understanding the Census-Income Dataset

To get a better understanding of the Census-Income Dataset the following steps were taken:

Reviewed the data files understand the attributes and prepare the files for reading
Evaluated missing data values and identified a plan to resolve these
Used Weka Explorer to review each attribute relative to the class variable
Investigated the subject of categorical data and identified a plan here
Investigated cross-correlation between between numeric attributes

These steps are detailed in this section.

Review and Prepare the Data Files

We started with adult.test (test data), adult.data (training data), adult.names (description of attributes).

The conversion of the original data involved

- In adult.test - removed the "." suffix from the class variable values as this is inconsistent with that in adult.data.
- Created adult.test.cvs and adult.data.cvs using attribute names from adult.names

Missing Data Values

There are 4262 missing data values in names.data and 2200 missing data values in the names.test.

Samples with missing values (denoted by "?" in the original file) were manually replaced by an empty sample (nothing). As the Matlab function "dataset" reads the file it will replace any numeric attributes with a NaN, and any nominal attribute with a "" (empty string). These values will allow us to provide appropriate values for missing data during preprocessing.

Read Data From File into dataset array

```
%%%  bring in training or test data into A from file
A = dataset('file',
'C:\Users\cwinsor\Documents\weka_files\project_2\adult.data.data','delimiter', ',');
A = dataset('file',
'C:\Users\cwinsor\Documents\weka_files\project_2\adult.test.data','delimiter', ',');
```

Categorical Data

Numeric attributes can be ordered or unordered. If an attribute is numeric-ordered then the Matlab decision tree function (classregtree) can use the value directly (using its < = > comparators). Attributes which are numeric-unordered do not have an underlying numeric order. These attributes must be binned in a manner consistent with that attribute's underlying structure. classregtree has a "categorical" option to identify those attributes which are numeric-unordered.

Normal attributes are always non-categorical (the values are always unstructured).

As can be seen in Table 1 there are NO attributes which are numeric-unordered.

However it is important to note - that Matlab's "classregtree" cannot take "Nominal" values as input - it requires nominal attributes be converted to some numeric equivalent. Therefore ALL nominal attributes will be converted to their index value (which is numeric). For these the classregtree "categorical" switch will be set so they are not interpreted as being ordered.

Cross-correlation between Numeric Attributes

I investigated the possibility that there might be cross-correlation between normal attributes (and thereby the possibility that one or more could be eliminated). For example capital_gain and capital_loss would seem to be natural candidates for cross-correlation but this simply did not turn out to be true. The cross-correlation matrix for

{age, education_num, capital_gain, capital_loss, hours_per_week}

was:

cc =

1.0000	0.0401	-0.0268	0.0842	-0.0369
0.0401	1.0000	0.0959	0.1431	0.2669
-0.0268	0.0959	1.0000	-0.0469	0.0355
0.0842	0.1431	-0.0469	1.0000	0.1280
-0.0369	0.2669	0.0355	0.1280	1.0000

There is no opportunity to eliminate attributes based on cross-correlation.

Summary Table

The following table identifies the attributes and their category (numeric-ordered, numeric-unordered, nominal), and the plan with respect to each variable.

Attribute Num	Attribute Name	Category	The Plan	Visual Inspection using Weka Explorer
1	age	Numeric, Ordered	Allow classregtree to use as is	<=50K peaks at ~30. >50K peaks at ~40
2	workclass	Nominal (8)	Convert to numeric	Self-employed seems to be a predictor is >50K. Otherwise not a strong predictor.
3	fnlwgt	Numeric	Do not use	THIS ATTRIBUTE TO BE REMOVED
4	education	Nominal (16)	Convert to numeric	Many values for this attribute.
5	education-num	Numeric, Ordered	Allow classregtree to use as is	This attribute shows a strong correlation to the class variable - above 12 years significantly increases likelihood of >50K.
6	marital-status	Nominal (7)	Convert to numeric	Interestingly the Married-civ-spouse is correlated equally with <=50K and >50K. All other values are predominantly <50K.
7	occupation	Nominal	Convert to numeric	A handful of the nominal values seem

		(14)		to predict the class variable, namely Priv-house-serv" and Other-service"
8	relationship	Nominal (6)	Convert to numeric	Some of the values predict <50K. Specifically Own-child, Not-in-family, Other-relative.
9	race	Nominal (5)	Convert to numeric	Some of the attribute values predict the class - namely Amer-Indian-Eskimo, Other.
10	sex	Nominal (2)	Convert to numeric	Female seems to be more strongly associated with <50K.
11	capital-gain	Numeric, Ordered	Allow classregtree to use as is	The majority are centered around 0 but there are outliers.
12	capital-loss	Numeric, Ordered	Allow classregtree to use as is	Similar to capital_gain
13	hours-per-week	Numeric, Ordered	Allow classregtree to use as is	Unclear if there is any correlation to the class
14	native-country	Nominal (41)	Convert to numeric	Many values - there should be an opportunity to simplify this.
15	class	Nominal (2)	Convert to numeric	

Objectives of the Experiments

The questions I will be looking to answer in the Report are:

Is it advantageous to use all the bins that Nominal attributes have? Since each bin will result in a branch departing that decision node a large number of bins would result in a wide tree. I suspect that reducing the number of bins in nominal data attributes will significantly simplify the tree.

With respect to numeric-ordered attributes (age, education_num, capital-gain, capital-loss). Is there any reason to introduce binning on these attributes ? My plan is to not attempt to bin or otherwise preprocess this data. By leaving these as "normal" values classregtree will be free to choose its own decision point. We will attempt to use switches into the model so it uses these normal attributes wisely.

Performance Metrics

The performance metrics are

- classification accuracy - this is measured by creating the model using training dataset, then running it against the test dataset. The classification accuracy is the percent of predictions that are correct.
- size of the tree - this is measured as number of nodes of the tree (t.size())
- readability of the tree - this is a subjective measure which is evaluated through the view(t) tree diagram.

Preprocessing the Data

Overview:

- Starting with the dataset array A which we read from file above - we remove the fnlwgt attribute to produce dataset B
- Next we clean up the data including removing missing data values. As a result of this we have a dataset array C which is fully populated.
- Then we establish index values for the Normal attributes. As a result of this we have dataset array D. This data is suitable for use by classregtree

Remove the fnlwgt Attribute:

```
B = A;
B.fnlwgt = [];
```

Clean Up The Data (numeric attributes)

For numeric attributes (Nan) these are substituted by a normal random value with mean + std of the attribute

```
% Replace any NaN in "age" with random value
```

```
nm = nanmean(B.age);  % find mean ignoring NaNs
ns = nanstd(B.age); % find std ignoring NaNs
[nr,nc] = size(B.age);
for n = 1:nr
   r = nm + ns.*randn(1,1);  % random with mean and stddev of rest of column
  B.age(n) = r;  % replace NaN with the random value
end

% Replace any NaN in "education_num" with random value
nm = nanmean(B.education_num);  % find mean ignoring NaNs
ns = nanstd(B.education_num); % find std ignoring NaNs
[nr,nc] = size(B.education_num);
for n = 1:nr
   r = nm + ns.*randn(1,1);  % random with mean and stddev of rest of column
  B.education_num(n) = r;  % replace NaN with the random value
end

% Replace any NaN in "capital_gain" with random value
nm = nanmean(B.capital_gain);  % find mean ignoring NaNs
ns = nanstd(B.capital_gain); % find std ignoring NaNs
[nr,nc] = size(B.capital_gain);
for n = 1:nr
   r = nm + ns.*randn(1,1);  % random with mean and stddev of rest of column
  B.capital_gain(n) = r;  % replace NaN with the random value
end

% Replace any NaN in "capital_loss" with random value
nm = nanmean(B.capital_loss);  % find mean ignoring NaNs
ns = nanstd(B.capital_loss); % find std ignoring NaNs
[nr,nc] = size(B.capital_loss);
for n = 1:nr
   r = nm + ns.*randn(1,1);  % random with mean and stddev of rest of column
  B.capital_loss(n) = r;  % replace NaN with the random value
end

% Replace any NaN in "hours_per_week" with random value
nm = nanmean(B.hours_per_week);  % find mean ignoring NaNs
ns = nanstd(B.hours_per_week); % find std ignoring NaNs
[nr,nc] = size(B.hours_per_week);
for n = 1:nr
   r = nm + ns.*randn(1,1);  % random with mean and stddev of rest of column
  B.hours_per_week(n) = r;  % replace NaN with the random value
end
```

Clean Up The Data (normal attributes)

For discrete attributes with a value of "" is converted to "no_data".

```
nf = 0;
[nr,nc] = size(B.workclass);
for n = 1:nr
  if (strcmp(B.workclass(n),  ''))      B.workclass(n) = cellstr('no_data');  nf=nf+1;
  end;
  if (strcmp(B.education(n), ''))       B.education(n) = cellstr('no_data');  nf=nf+1;
  end;
  if (strcmp(B.marital_status(n), '')) B.marital_status(n) = cellstr('no_data'); nf=nf+1;
  end;
  if (strcmp(B.occupation(n), ''))      B.occupation(n) = cellstr('no_data');    nf=nf+1;
  end;
  if (strcmp(B.relationship(n), ''))    B.relationship(n) = cellstr('no_data');  nf=nf+1;
  end;
  if (strcmp(B.race(n), ''))            B.race(n) = cellstr('no_data');          nf=nf+1;
  end;
  if (strcmp(B.sex(n), ''))             B.sex(n) = cellstr('no_data');           nf=nf+1;
  end;
  if (strcmp(B.native_country(n), '')) B.native_country(n) = cellstr('no_data'); nf=nf+1;
  end;
  if (strcmp(B.class(n), ''))           B.class(n) = cellstr('no_data');         nf=nf+1;
  end;
end;
nf
```

As a result of the preprocessing we have dataset B which has no missing data samples.

Experiment 1 - Decision Tree Constructed With Default Values

Objectives

The experiment intends to establish a baseline using the default values for classregtree.

Data

In this experiment

- all attributes are used
- the NaN values (found in the discrete attributes) have been replaced with "no_data"

Additional Pre processing

Establish Index Values for Discrete Attributes

Attributes which are discrete will need index values, not strings, in order to be used by classregtree. We use grp2idx for this.

```
C = B;
C.workclass = grp2idx(B.workclass);
C.education = grp2idx(B.education);
C.marital_status = grp2idx(B.marital_status);
C.occupation = grp2idx(B.occupation);
C.relationship = grp2idx(B.relationship);
C.race = grp2idx(B.race);
C.sex = grp2idx(B.sex);
C.native_country = grp2idx(B.native_country);
C.class = grp2idx(B.class);

R = Q;
R.workclass = grp2idx(Q.workclass);
R.education = grp2idx(Q.education);
R.marital_status = grp2idx(Q.marital_status);
R.occupation = grp2idx(Q.occupation);
R.relationship = grp2idx(Q.relationship);
R.race = grp2idx(Q.race);
R.sex = grp2idx(Q.sex);
R.native_country = grp2idx(Q.native_country);
R.class = grp2idx(Q.class);
```

Split out the Class Variable, Make A List Of Attribute Names, Make A List of Categorical Attributes

```
D_atts = [C.age, C.workclass, C.education, C.education_num, C.marital_status, C.occupation,
C.relationship, C.race, C.sex, C.capital_gain, C.capital_loss, C.hours_per_week,
C.native_country];
D_attNames = properties(C); D_attNames(end) = []; D_attNames(end) = [];
D_class = C.class;
D_categoricalAttList = [2 3 5 6 7 8 9];

S_atts = [R.age, R.workclass, R.education, R.education_num, R.marital_status, R.occupation,
R.relationship, R.race, R.sex, R.capital_gain, R.capital_loss, R.hours_per_week,
R.native_country];
S_attNames = properties(C); S_attNames(end) = []; S_attNames(end) = [];
S_class = R.class;
S_categoricalAttList = [2 3 5 6 7 8 9];
```

Summary
We now have two training dataset arrays (dataset D_atts and D_class) and testing dataset arrays S_atts and S_class which are suitable for classregtree.

Parameters and Settings

The experiment generates a model using the default values for classregtree, namely

- pruning ON
- minparent 10
- minleaf 1
- mergeleaves ON

The command run was:

```
t = classregtree( D_atts, D_class, 'names', D_attNames, 'categorical', D_categoricalAttList,
'method', 'classification', 'prune', 'on', 'minparent', 10, 'minleaf', 1, 'mergeleaves',
'on');
```

Resulting Model

The number of nodes = 3773
The readability = POOR

Performance of the Model

```
% testing accuracy
predictedClass = t.eval(S_atts);   % run the model using the testing data
good = (int2str(S_class(:)) == cell2mat(predictedClass(:)));
accuracy = sum(good) /   size(S_class,1)
accuracy =
   0.7576
```

The testing accuracy was 75.8 %

Experiment 2 - Reducing The Bin Count for Normal Attributes

Objectives

The experiment investigates the normal attributes with an eye toward reducing the number of bins for those attributes. For example "native_country" has 41 values so thereby this decision node would have 41 branches. If we could identify the ones that are associated with the class variable and bin those it should dramatically reduce the width of that node, and thereby the overall size of the decision tree.

Data

We start with the same training and testing dataset as in the prior experiment, but then we further process it to combine attribute values. This is done by evaluating each attribute with respect to the class variable to determine which attribute values are good predictors. We then bin the good predictors together so we end up with just two bins for each attribute.

Additional Pre processing

education: We combine all attributes which are 12th grade and below into one bin, and those above 12th grade into a second bin.

EDUCATION

```
[ag,agName] = grp2idx(B.education);
[cg, cgName] = grp2idx(B.class);
for attNum = 1:size(agName, 1)
  x = strcmp(B.education, agName(attNum));
  pos =  sum(  x & strcmp(B.class,'>50K'));
  neg =  sum ( x & strcmp(B.class,'<=50K'));
  total = pos + neg;
  if (total > 0)
    [ agName(attNum)   'pos'  pos/total  'neg'  neg/total]
  end
end
```

```
    'Doctorate'    'pos'    [0.7409]    'neg'    [0.2591]
    'Prof-school'  'pos'    [0.7344]    'neg'    [0.2656]
    'Masters'      'pos'    [0.5566]    'neg'    [0.4434]
    'Bachelors'    'pos'    [0.4148]    'neg'    [0.5852]

    'Assoc-acdm'   'pos'    [0.2484]    'neg'    [0.7516]
    'Assoc-voc'    'pos'    [0.2612]    'neg'    [0.7388]
    'Some-college' 'pos'    [0.1902]    'neg'    [0.8098]
    'HS-grad'      'pos'    [0.1595]    'neg'    [0.8405]
    '11th'         'pos'    [0.0511]    'neg'    [0.9489]
    '9th'          'pos'    [0.0525]    'neg'    [0.9475]
```

```
    '7th-8th'      'pos'    [0.0619]    'neg'    [0.9381]
    '5th-6th'      'pos'    [0.0480]    'neg'    [0.9520]
    '10th'         'pos'    [0.0665]    'neg'    [0.9335]
    '1st-4th'      'pos'    [0.0357]    'neg'    [0.9643]
    '12th'         'pos'    [0.0762]    'neg'    [0.9238]
    'Preschool'    'pos'    [0]    'neg'    [1]
```

We conclude that Bachelors and above should be binned together.

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.education(sampleNum),  'Doctorate'   )) || (strcmp(B.education(sampleNum),
'Prof-school' )) || (strcmp(B.education(sampleNum),  'Masters'     )) ||
(strcmp(B.education(sampleNum),  'Bachelors'   )))
    B.education{sampleNum} = 'education_high';
  else
    B.education{sampleNum} = 'education_low';
  end
end
```

WORKCLASS

```
[ag,agName] = grp2idx(B.workclass);
[cg, cgName] = grp2idx(B.class);
for attNum = 1:size(agName, 1)
  x = strcmp(B.workclass, agName(attNum));
  pos =  sum(  x & strcmp(B.class,'>50K'));
  neg =  sum ( x & strcmp(B.class,'<=50K'));
  total = pos + neg;
  if (total > 0)
    [ agName(attNum)   'pos'  pos/total  'neg'  neg/total]
  end
end
```

```
    'Self-emp-inc'        'pos'    [0.5573]    'neg'    [0.4427]

    'State-gov'           'pos'    [0.2720]    'neg'    [0.7280]
    'Self-emp-not-inc'    'pos'    [0.2849]    'neg'    [0.7151]
    'Private'             'pos'    [0.2187]    'neg'    [0.7813]
    'Federal-gov'         'pos'    [0.3865]    'neg'    [0.6135]
    'Local-gov'           'pos'    [0.2948]    'neg'    [0.7052]
    'no_data'             'pos'    [0.1040]    'neg'    [0.8960]
    'Without-pay'         'pos'    [0]    'neg'    [1]
    'Never-worked'        'pos'    [0]    'neg'    [1]
```

We conclude that Self-Emp-inc should be its own bin.

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.workclass(sampleNum), 'Self-emp-inc'   )))
```

```
    B.workclass{sampleNum} = 'workclass_high';
  else
    B.workclass{sampleNum} = 'workclass_low';
  end
end
```

MARITAL_STATUS

```
[ag,agName] = grp2idx(B.marital_status);
[cg, cgName] = grp2idx(B.class);
for attNum = 1:size(agName, 1)
  x = strcmp(B.marital_status, agName(attNum));
  pos =  sum(  x & strcmp(B.class,'>50K'));
  neg =  sum ( x & strcmp(B.class,'<=50K'));
  total = pos + neg;
  if (total > 0)
    [ agName(attNum)   'pos'  pos/total  'neg'  neg/total]
  end
end
```

```
    'Married-civ-spouse'    'pos'    [0.4468]    'neg'    [0.5532]
    'Married-AF-spouse'     'pos'    [0.4348]    'neg'    [0.5652]

    'Divorced'              'pos'    [0.1042]    'neg'    [0.8958]
    'Married-spouse-absent' 'pos'    [0.0813]    'neg'    [0.9187]
    'Separated'             'pos'    [0.0644]    'neg'    [0.9356]
    'Widowed'               'pos'    [0.0856]    'neg'    [0.9144]
    'Never-married'         'pos'    [0.0460]    'neg'    [0.9540]
```

We conclude Married-civ-spouse and Married-AF-spouse will be a bin, and others a second bin.

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.marital_status(sampleNum),  'Married-civ-spouse'    )) |
(strcmp(B.marital_status(sampleNum), 'Married-AF-spouse'  )))
    B.marital_status{sampleNum} = 'marital_status_high';
  else
    B.marital_status{sampleNum} = 'marital_status_low';
  end
end
```

OCCUPATION

```
[ag,agName] = grp2idx(B.occupation);
[cg, cgName] = grp2idx(B.class);
for attNum = 1:size(agName, 1)
  x = strcmp(B.occupation, agName(attNum));
  pos =  sum(  x & strcmp(B.class,'>50K'));
  neg =  sum ( x & strcmp(B.class,'<=50K'));
  total = pos + neg;
  if (total > 0)
    [ agName(attNum)   'pos'  pos/total  'neg'  neg/total]
  end
end
```

```
    'Exec-managerial'    'pos'    [0.4840]    'neg'    [0.5160]
    'Prof-specialty'     'pos'    [0.4490]    'neg'    [0.5510]
    'Tech-support'       'pos'    [0.3050]    'neg'    [0.6950]
    'Protective-serv'    'pos'    [0.3251]    'neg'    [0.6749]
    'Sales'              'pos'    [0.2693]    'neg'    [0.7307]

    'Craft-repair'       'pos'    [0.2266]    'neg'    [0.7734]
    'Transport-moving'   'pos'    [0.2004]    'neg'    [0.7996]
    'Adm-clerical'       'pos'    [0.1345]    'neg'    [0.8655]
    'Handlers-cleaners'  'pos'    [0.0628]    'neg'    [0.9372]
    'Other-service'      'pos'    [0.0416]    'neg'    [0.9584]
    'Farming-fishing'    'pos'    [0.1157]    'neg'    [0.8843]
    'Machine-op-inspct'  'pos'    [0.1249]    'neg'    [0.8751]
    'no_data'            'pos'    [0.1036]    'neg'    [0.8964]
    'Armed-Forces'       'pos'    [0.1111]    'neg'    [0.8889]
    'Priv-house-serv'    'pos'    [0.0067]    'neg'    [0.9933]
```

We bin the first 5 attribute values.

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.occupation(sampleNum), 'Exec-managerial'     ))
|(strcmp(B.occupation(sampleNum),'Prof-specialty'   )) |
(strcmp(B.occupation(sampleNum),'Tech-support'   )) |
(strcmp(B.occupation(sampleNum),'Protective-serv'   )) |
(strcmp(B.occupation(sampleNum),'Sales'     )))
    B.occupation{sampleNum} = 'occupation_high';
  else
    B.occupation{sampleNum} = 'occupation_low';
  end
end
```

RELATIONSHIP

```
[ag,agName] = grp2idx(B.relationship);
```

```
[cg, cgName] = grp2idx(B.class);
for attNum = 1:size(agName, 1)
  x = strcmp(B.relationship, agName(attNum));
  pos =  sum(  x & strcmp(B.class,'>50K'));
  neg =  sum ( x & strcmp(B.class,'<=50K'));
  total = pos + neg;
  if (total > 0)
    [ agName(attNum)   'pos'  pos/total  'neg'  neg/total]
  end
end
```

```
    'Husband'         'pos'    [0.4486]    'neg'    [0.5514]
    'Wife'            'pos'    [0.4751]    'neg'    [0.5249]

    'Not-in-family'   'pos'    [0.1031]    'neg'    [0.8969]
    'Own-child'       'pos'    [0.0132]    'neg'    [0.9868]
    'Unmarried'       'pos'    [0.0633]    'neg'    [0.9367]
    'Other-relative'  'pos'    [0.0377]    'neg'    [0.9623]
```

We bin the first 2 attribute values.

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.relationship(sampleNum), 'Husband'      )) |(strcmp(B.relationship(sampleNum),
'Wife'     )))
    B.relationship{sampleNum} = 'relationship_high';
  else
    B.relationship{sampleNum} = 'relationship_low';
  end
end
```

RACE

```
[ag,agName] = grp2idx(B.race);
[cg, cgName] = grp2idx(B.class);
for attNum = 1:size(agName, 1)
  x = strcmp(B.race, agName(attNum));
  pos =  sum(  x & strcmp(B.class,'>50K'));
  neg =  sum ( x & strcmp(B.class,'<=50K'));
  total = pos + neg;
  if (total > 0)
    [ agName(attNum)   'pos'  pos/total  'neg'  neg/total]
  end
end
```

```
    'White'               'pos'    [0.2559]    'neg'    [0.7441]
    'Asian-Pac-Islander'  'pos'    [0.2656]    'neg'    [0.7344]

    'Black'               'pos'    [0.1239]    'neg'    [0.8761]
    'Amer-Indian-Eskimo'  'pos'    [0.1158]    'neg'    [0.8842]
    'Other'               'pos'    [0.0923]    'neg'    [0.9077]
```

We bin the first 3 attribute values.

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.race(sampleNum), 'White'        )) |(strcmp(B.race(sampleNum),
'Asian-Pac-Islander'   )))
    B.race{sampleNum} = 'race_high';
  else
    B.race{sampleNum} = 'race_low';
  end
end
```

SEX

```
This attribute already has only 2 bins.
```

NATIVE_COUNTRY

```
[ag,agName] = grp2idx(B.native_country);
[cg, cgName] = grp2idx(B.class);
for attNum = 1:size(agName, 1)
  x = strcmp(B.native_country, agName(attNum));
  pos =  sum(  x & strcmp(B.class,'>50K'));
  neg =  sum ( x & strcmp(B.class,'<=50K'));
  total = pos + neg;
  if (total > 0)
    [ agName(attNum)   'pos'  pos/total  'neg'  neg/total]
  end
end
```

```
    'France'                'pos'    [0.4138]    'neg'    [0.5862]
    'India'                 'pos'    [0.4000]    'neg'    [0.6000]
    'Iran'                  'pos'    [0.4186]    'neg'    [0.5814]
    'Japan'                 'pos'    [0.3871]    'neg'    [0.6129]
    'Yugoslavia'            'pos'    [0.3750]    'neg'    [0.6250]
    'Hong'                  'pos'    [0.3000]    'neg'    [0.7000]
    'Cambodia'              'pos'    [0.3684]    'neg'    [0.6316]
    'England'               'pos'    [0.3333]    'neg'    [0.6667]
    'Canada'                'pos'    [0.3223]    'neg'    [0.6777]
    'Germany'               'pos'    [0.3212]    'neg'    [0.6788]
    'Philippines'           'pos'    [0.3081]    'neg'    [0.6919]
    'Italy'                 'pos'    [0.3425]    'neg'    [0.6575]
    'Taiwan'                'pos'    [0.3922]    'neg'    [0.6078]

    'United-States'         'pos'    [0.2458]    'neg'    [0.7542]
    'Cuba'                  'pos'    [0.2632]    'neg'    [0.7368]
    'no_data'               'pos'    [0.2504]    'neg'    [0.7496]
    'China'                 'pos'    [0.2667]    'neg'    [0.7333]
    'Scotland'              'pos'    [0.2500]    'neg'    [0.7500]
    'Greece'                'pos'    [0.2759]    'neg'    [0.7241]
    'Ireland'               'pos'    [0.2083]    'neg'    [0.7917]
    'Hungary'               'pos'    [0.2308]    'neg'    [0.7692]
    'Poland'                'pos'    [0.2000]    'neg'    [0.8000]
    'South'                 'pos'    [0.2000]    'neg'    [0.8000]
    'Jamaica'               'pos'    [0.1235]    'neg'    [0.8765]
    'Puerto-Rico'           'pos'    [0.1053]    'neg'    [0.8947]
    'Portugal'              'pos'    [0.1081]    'neg'    [0.8919]
    'Thailand'              'pos'    [0.1667]    'neg'    [0.8333]
    'Ecuador'               'pos'    [0.1429]    'neg'    [0.8571]
    'Laos'                  'pos'    [0.1111]    'neg'    [0.8889]
    'Mexico'                'pos'    [0.0513]    'neg'    [0.9487]
    'Honduras'              'pos'    [0.0769]    'neg'    [0.9231]
```

```
    'Columbia'                    'pos'    [0.0339]    'neg'    [0.9661]
    'Haiti'                       'pos'    [0.0909]    'neg'    [0.9091]
    'Dominican-Republic'          'pos'    [0.0286]    'neg'    [0.9714]
    'El-Salvador'                 'pos'    [0.0849]    'neg'    [0.9151]
    'Guatemala'                   'pos'    [0.0469]    'neg'    [0.9531]
    'Peru'                        'pos'    [0.0645]    'neg'    [0.9355]
    'Trinadad&Tobago'             'pos'    [0.1053]    'neg'    [0.8947]
    'Nicaragua'                   'pos'    [0.0588]    'neg'    [0.9412]
    'Vietnam'                     'pos'    [0.0746]    'neg'    [0.9254]
    'Outlying-US(Guam-USVI-etc)' 'pos'    [0]         'neg'    [1]
    'Holand-Netherlands'          'pos'    [0]         'neg'    [1]
```

We bin the first 13 attribute values.

```
for sampleNum = 1:size(B.age, 1)
  if (   (strcmp(B.native_country(sampleNum),   'France'     )) |
(strcmp(B.native_country(sampleNum),    'India'      )) |
(strcmp(B.native_country(sampleNum),    'Iran'       )) |
(strcmp(B.native_country(sampleNum),    'Japan'    )) |
(strcmp(B.native_country(sampleNum),    'Yugoslavia'   )) |
(strcmp(B.native_country(sampleNum),    'Hong'       )) |
(strcmp(B.native_country(sampleNum),     'Cambodia'      )) |
(strcmp(B.native_country(sampleNum),    'England'      )) |
(strcmp(B.native_country(sampleNum),    'Canada'        )) |
(strcmp(B.native_country(sampleNum),    'Germany'          )) |
(strcmp(B.native_country(sampleNum),    'Philippines'     )) |
(strcmp(B.native_country(sampleNum),    'Italy'                )) |
(strcmp(B.native_country(sampleNum),    'Taiwan'            )))
    B.native_country{sampleNum} = 'country_high';
  else
    B.native_country{sampleNum} = 'country_low';
  end
end
```

Establish Index Values for Discrete Attributes

Attributes which are discrete will need index values, not strings, in order to be used by classregtree. We use grp2idx for this.

```
C = B;
C.workclass = grp2idx(B.workclass);
C.education = grp2idx(B.education);
C.marital_status = grp2idx(B.marital_status);
C.occupation = grp2idx(B.occupation);
C.relationship = grp2idx(B.relationship);
C.race = grp2idx(B.race);
C.sex = grp2idx(B.sex);
C.native_country = grp2idx(B.native_country);
C.class = grp2idx(B.class);
```

Split out the Class Variable, Make A List Of Attribute Names, Make A List of Categorical Attributes

% Training Data

```
D_atts = [C.age, C.workclass, C.education, C.education_num, C.marital_status, C.occupation,
C.relationship, C.race, C.sex, C.capital_gain, C.capital_loss, C.hours_per_week,
C.native_country];
D_attNames = properties(C); D_attNames(end) = []; D_attNames(end) = [];
D_class = C.class;
D_categoricalAttList = [2 3 5 6 7 8 9];
```

% Test data

```
T_atts = [C.age, C.workclass, C.education, C.education_num, C.marital_status, C.occupation,
C.relationship, C.race, C.sex, C.capital_gain, C.capital_loss, C.hours_per_week,
C.native_country];
T_attNames = properties(C); T_attNames(end) = []; T_attNames(end) = [];
T_class = C.class;
T_categoricalAttList = [2 3 5 6 7 8 9];
```

Summary

We now have two training dataset arrays (dataset D_atts and D_class) and testing dataset arrays T_atts and T_class which are suitable for classregtree.

Parameters and Settings

The experiment again uses the default values for classregtree:

- pruning ON
- minparent 10
- minleaf 1
- mergeleaves ON

The command run was:

```
% run with default settings
t = classregtree( D_atts, D_class, 'names', D_attNames, 'categorical', D_categoricalAttList,
'method', 'classification', 'prune', 'on', 'minparent', 10, 'minleaf', 1, 'mergeleaves',
'on');
```

Additional Pre or Post processing

None

Resulting Model

The number of nodes = 3085
The readability = POOR

Performance of the Model

% training accuracy (resubstitution accuracy)

```
predictedClass = t.eval(D_atts);   % run the model using the training data
good = (int2str(D_class(:)) == cell2mat(predictedClass(:)));
accuracy = sum(good) /   size(D_class,1)
```

accuracy =
   0.9070

```
% testing accuracy
predictedClass = t.eval(T_atts);   % run the model using the testing data
good = (int2str(T_class(:)) == cell2mat(predictedClass(:)));
accuracy = sum(good) /   size(T_class,1)
accuracy =
    0.8284
```

Testing accuracy was 82.8 %

Summary of This Experiment

The reduced binning of normal attributes helped slightly in reducing the complexity of the model.

An observation is that the model has many cases relating to the numeric attributes. Here is a small piece of the model code:

```
2678  if hours_per_week<47.5 then node 2740 elseif hours_per_week>=47.5 then node 2741 else 1
2679  if age<45.5 then node 2742 elseif age>=45.5 then node 2743 else 1
2680  class = 2
2681  if hours_per_week<43 then node 2744 elseif hours_per_week>=43 then node 2745 else 1
2682  if hours_per_week<42 then node 2746 elseif hours_per_week>=42 then node 2747 else 1
2687  class = 2
2688  if workclass=1 then node 2748 elseif workclass=2 then node 2749 else 2
2689  if age<54.5 then node 2750 elseif age>=54.5 then node 2751 else 1
2690  if race=1 then node 2752 elseif race=2 then node 2753 else 2
2691  if hours_per_week<35.5 then node 2754 elseif hours_per_week>=35.5 then node 2755 else 1
```

The next experiment will focus on getting the complexity down on the numeric attributes.

Experiment 3 - Varying "minparent" Model Parameter

Objectives

The experiment experiments with "minparent" switch into classregtree. This switch imposes a restriction that there are minparent samples in order to branch a node. It is believed this will force classregtree to be strategic in its use of normal attributes.

Data

We use the same data as Experiment 2.

Parameters and Settings

The experiment generates a model using the default values for classregtree, namely

- pruning ON
- minparent **varied**
- minleaf 1
- mergeleaves ON

%The command run was:

```
t = classregtree( D_atts, D_class, 'names', D_attNames, 'categorical', D_categoricalAttList,
'method', 'classification', 'prune', 'on', 'minparent', varied, 'minleaf', 1, 'mergeleaves',
'on');
```

% Accuracy was evaluated using:

```
% testing accuracy
predictedClass = t.eval(T_atts);   % run the model using the testing data
good = (int2str(T_class(:)) == cell2mat(predictedClass(:)));
accuracy = sum(good) /   size(T_class,1)
```

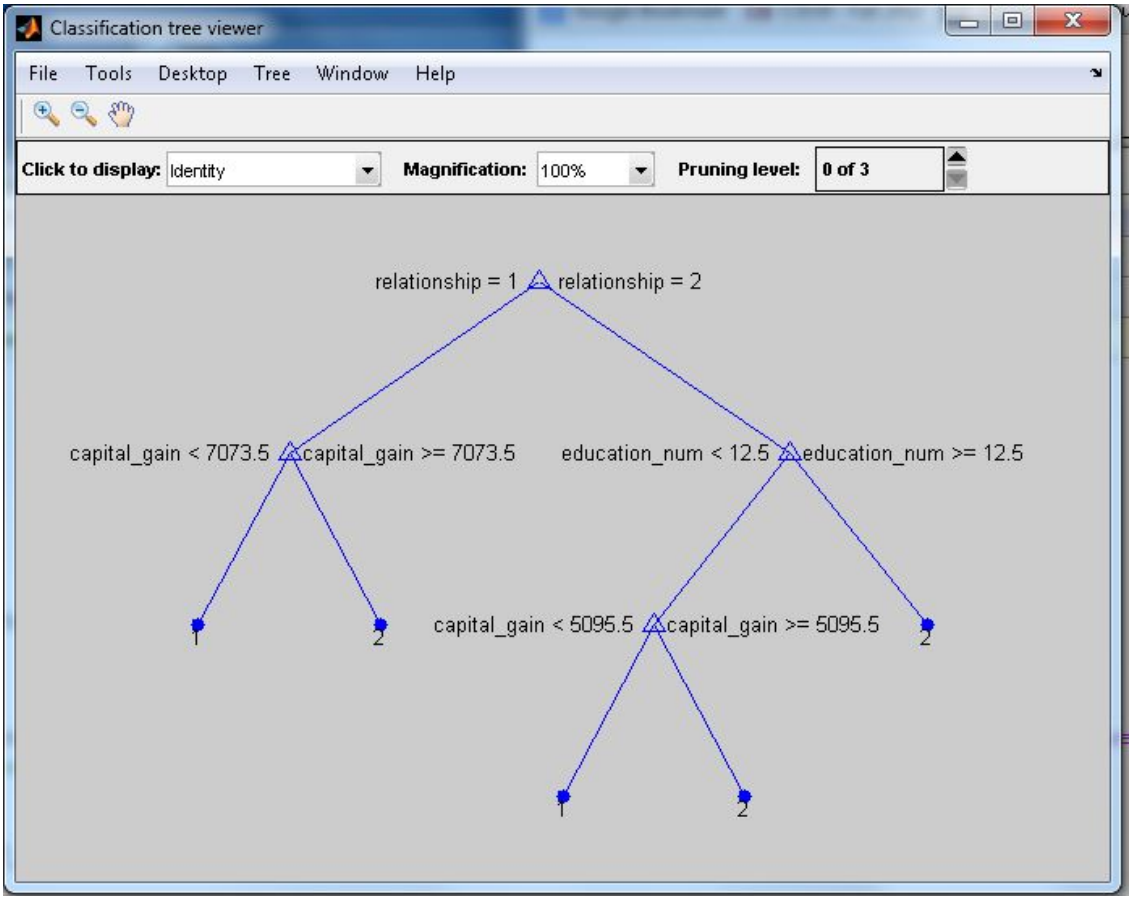
Resulting Models and Performance

minparent	Model size (number of nodes)	Readability	Testing Accuracy
10	3085	Poor	82.8
20	1913	Poor	83.6
50	993	Poor	84.3
100	553	Poor	84.6
200	347	Poor	85.6
500	177	Poor	85.9

1000	81	Fair	85.9
2000	37	Fair	85.6
5000	15	Good	84.5
10000	9	Very Good	84.5
20000	1 (zero-R !)	Excellent	76.4

The decision tree with minparent=10000 is as follows:

```
Decision tree for classification
1  if relationship=1 then node 2 elseif relationship=2 then node 3 else 1
2  if capital_gain<7073.5 then node 4 elseif capital_gain>=7073.5 then node 5 else 1
3  if education_num<12.5 then node 6 elseif education_num>=12.5 then node 7 else 1
4  class = 1
5  class = 2
6  if capital_gain<5095.5 then node 8 elseif capital_gain>=5095.5 then node 9 else 1
7  class = 2
8  class = 1
9  class = 2
```



Summary Of Results

All the models were able to predict with greater than 70% accuracy. Getting the number of nodes and complexity down required careful choice of bins for normal attributes. Then the tree was reduced by imposing the minparent restriction. Interestingly the reduction in tree complexity was accompanied by an INCREASE in its accuracyup to where the tree reduced to a Zero-R implementation.

Experiment	Description	Model size (number of nodes)	Readability	Testing Accuracy
1	Default Settings for classregtree	3773	Poor	75.8%
2	Reduced bins for normal attributes	3085	Poor	82.8%
3	minparent = 10	3085	Poor	82.8
3	minparent = 20	1913	Poor	83.6
3	minparent = 50	993	Poor	84.3
3	1minparent = 00	553	Poor	84.6
3	minparent = 200	347	Poor	85.6
3	minparent = 500	177	Poor	85.9
3	minparent = 1000	81	Fair	85.9
3	minparent = 2000	37	Fair	85.6
3	minparent = 5000	15	Good	84.5
3	minparent = 10000	9	Very Good	84.5
3	minparent = 20000	1 (zero-R !)	Excellent	76.4

Appendix

% Read Data From File into dataset array

```
%%%  bring in training or test data into A from file
A = dataset('file',
'C:\Users\cwinsor\Documents\weka_files\project_2\adult.data.data','delimiter', ',');
A = dataset('file',
'C:\Users\cwinsor\Documents\weka_files\project_2\adult.test.data','delimiter', ',');
```

% Remove the fnlwgt Attribute:

```
B = A;
B.fnlwgt = [];
```

%Clean Up The Data (normal attributes)

%For discrete attributes with a value of "" is converted to "no_data".

```
nf = 0;
[nr,nc] = size(B.workclass);
for n = 1:nr
  if (strcmp(B.workclass(n),  ''))      B.workclass(n) = cellstr('no_data');  nf=nf+1;
  end;
  if (strcmp(B.education(n), ''))       B.education(n) = cellstr('no_data');  nf=nf+1;
  end;
  if (strcmp(B.marital_status(n), '')) B.marital_status(n) = cellstr('no_data'); nf=nf+1;
  end;
  if (strcmp(B.occupation(n), ''))      B.occupation(n) = cellstr('no_data');    nf=nf+1;
  end;
  if (strcmp(B.relationship(n), ''))    B.relationship(n) = cellstr('no_data');  nf=nf+1;
  end;
  if (strcmp(B.race(n), ''))            B.race(n) = cellstr('no_data');          nf=nf+1;
  end;
  if (strcmp(B.sex(n), ''))             B.sex(n) = cellstr('no_data');           nf=nf+1;
  end;
  if (strcmp(B.native_country(n), '')) B.native_country(n) = cellstr('no_data'); nf=nf+1;
  end;
  if (strcmp(B.class(n), ''))           B.class(n) = cellstr('no_data');         nf=nf+1;
  end;
end;
nf
```

%As a result of the preprocessing we have dataset B which has no missing data samples.

['EDUCATION']

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.education(sampleNum),  'Doctorate'   )) || (strcmp(B.education(sampleNum),
'Prof-school' )) || (strcmp(B.education(sampleNum),  'Masters'     )) ||
(strcmp(B.education(sampleNum),  'Bachelors'   )))
    B.education{sampleNum} = 'education_high';
  else
    B.education{sampleNum} = 'education_low';
  end
end
```

['WORKCLASS']

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.workclass(sampleNum), 'Self-emp-inc'   )))
    B.workclass{sampleNum} = 'workclass_high';
  else
    B.workclass{sampleNum} = 'workclass_low';
  end
end
```

['MARITAL_STATUS']

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.marital_status(sampleNum),  'Married-civ-spouse'    )) |
(strcmp(B.marital_status(sampleNum), 'Married-AF-spouse'  )))
    B.marital_status{sampleNum} = 'marital_status_high';
  else
    B.marital_status{sampleNum} = 'marital_status_low';
  end
end
```

['OCCUPATION']

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.occupation(sampleNum), 'Exec-managerial'     ))
|(strcmp(B.occupation(sampleNum),'Prof-specialty'   )) |
(strcmp(B.occupation(sampleNum),'Tech-support'   )) |
(strcmp(B.occupation(sampleNum),'Protective-serv'   )) |
(strcmp(B.occupation(sampleNum),'Sales'     )))
    B.occupation{sampleNum} = 'occupation_high';
  else
    B.occupation{sampleNum} = 'occupation_low';
  end
end
```

['RELATIONSHIP']

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.relationship(sampleNum), 'Husband'      )) |(strcmp(B.relationship(sampleNum),
'Wife'     )))
    B.relationship{sampleNum} = 'relationship_high';
  else
    B.relationship{sampleNum} = 'relationship_low';
  end
end
```

['RACE']

```
for sampleNum = 1:size(B.age, 1)
  if ((strcmp(B.race(sampleNum), 'White'        )) |(strcmp(B.race(sampleNum),
'Asian-Pac-Islander'   )))
    B.race{sampleNum} = 'race_high';
  else
    B.race{sampleNum} = 'race_low';
  end
end
```

['NATIVE_COUNTRY']

```
for sampleNum = 1:size(B.age, 1)
  if (   (strcmp(B.native_country(sampleNum),   'France'     )) |
(strcmp(B.native_country(sampleNum),    'India'      )) |
(strcmp(B.native_country(sampleNum),    'Iran'       )) |
(strcmp(B.native_country(sampleNum),    'Japan'    )) |
(strcmp(B.native_country(sampleNum),    'Yugoslavia'   )) |
(strcmp(B.native_country(sampleNum),    'Hong'       )) |
(strcmp(B.native_country(sampleNum),     'Cambodia'      )) |
(strcmp(B.native_country(sampleNum),    'England'      )) |
(strcmp(B.native_country(sampleNum),    'Canada'        )) |
(strcmp(B.native_country(sampleNum),    'Germany'         )) |
(strcmp(B.native_country(sampleNum),    'Philippines'     )) |
(strcmp(B.native_country(sampleNum),    'Italy'               )) |
(strcmp(B.native_country(sampleNum),    'Taiwan'           )))
    B.native_country{sampleNum} = 'country_high';
  else
    B.native_country{sampleNum} = 'country_low';
  end
end
```

%Establish Index Values for Discrete Attributes

%Attributes which are discrete will need index values, not strings, in order to be used by classregtree. We use grp2idx for this.

```
C = B;
C.workclass = grp2idx(B.workclass);
C.education = grp2idx(B.education);
C.marital_status = grp2idx(B.marital_status);
C.occupation = grp2idx(B.occupation);
C.relationship = grp2idx(B.relationship);
C.race = grp2idx(B.race);
C.sex = grp2idx(B.sex);
C.native_country = grp2idx(B.native_country);
C.class = grp2idx(B.class);
```

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Split out the Class Variable, Make A List Of Attribute Names, Make A List of Categorical Attributes

% Training Data

```
D_atts = [C.age, C.workclass, C.education, C.education_num, C.marital_status, C.occupation,
C.relationship, C.race, C.sex, C.capital_gain, C.capital_loss, C.hours_per_week,
C.native_country];
D_attNames = properties(C); D_attNames(end) = []; D_attNames(end) = [];
D_class = C.class;
D_categoricalAttList = [2 3 5 6 7 8 9];
```

% Test data

```
T_atts = [C.age, C.workclass, C.education, C.education_num, C.marital_status, C.occupation,
C.relationship, C.race, C.sex, C.capital_gain, C.capital_loss, C.hours_per_week,
C.native_country];
T_attNames = properties(C); T_attNames(end) = []; T_attNames(end) = [];
T_class = C.class;
T_categoricalAttList = [2 3 5 6 7 8 9];
```