Matrix Scatter Plot

from pandas.tools.plotting import scatter_matrix


Cheaper homes = Worse transit

Better shopping = better food

More diversity = lower creativity?

Wellness not correlated with other factors
Recap: Information Gain

Titanic Entropy = 0.96

Weighted Entropy: 466/1309 * 0.25 + 843 / 1309 * 0.21 = 0.22
Information Gain for split: 0.96 – 0.22 = 0.74
### Before:

14 records, 9 are “yes”

\[
- \left( \frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14} \right) = 0.94
\]

If we choose **outlook**:

- **overcast**: 4 records, 4 are “yes”
  \[
  - \left( \frac{4}{4} \log_2 \frac{4}{4} \right) = 0
  \]

- **rainy**: 5 records, 3 are “yes”
  \[
  - \left( 3 \log_2 \frac{3}{5} + 2 \log_2 \frac{2}{5} \right) = 0.97
  \]

- **sunny**: 5 records, 2 are “yes”
  \[
  - \left( \frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5} \right) = 0.97
  \]

**Expected new entropy:**

\[
\frac{4}{14} \times 0.0 + \frac{5}{14} \times 0.97 + \frac{5}{14} \times 0.97
\]

\[= 0.69\]
Before: 14 records, 9 are “yes”

\[- \left( \frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14} \right) = 0.94 \]

If we choose \textit{windy}:

FALSE: 8 records, 6 are “yes”

0.81 = -(6/8*log(6/8)+2/8*log(2/8))

TRUE: 6 records, 3 are “yes”

Expected new entropy:

\[0.81(8/14) + 1 (6/14) = 0.89\]
<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
</tbody>
</table>

Before: 14 records, 9 are “yes”

\[
- \left( \frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14} \right) = 0.94
\]

If we choose temperature:

- cool : 4 records, 3 are “yes”
  \[0.81\]
- rainy : 4 records, 2 are “yes”
  \[1.0\]
- sunny : 6 records, 4 are “yes”
  \[0.92\]

Expected new entropy:

\[0.81(4/14) + 1.0(4/14) + 0.92(6/14)\]

\[= 0.91\]
<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Before:** 14 records, 9 are “yes”

\[ - \left( \frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14} \right) = 0.94 \]

If we choose **humidity**:

- **normal** : 7 records, 6 are “yes”
  \[ 0.59 \]
- **high** : 7 records, 2 are “yes”
  \[ 0.86 \]

Expected new entropy:

\[ 0.59(7/14) + 0.86(7/14) \]

\[ = 0.725 \]
<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>TRUE</td>
<td>yes</td>
</tr>
</tbody>
</table>

Before: 14 records, 9 are “yes”

$$- \left( \frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14} \right) = 0.94$$

- **outlook**
  - 0.94 – 0.69 = 0.25  
    highest gain

- **temperature**
  - 0.94 – 0.91 = 0.03

- **humidity**
  - 0.94 – 0.725 = 0.215

- **windy**
  - 0.94 – 0.87 = 0.07
Outlook

- Rainy
- Sunny
- Overcast
Clicker: Document Classification

Question (assuming equal size splits):
a) Falcon’s Information Gain is higher
b) Mars’ Information Gain is higher
Building a Decision Tree (ID3 Algorithm)

• Assume attributes are discrete  
  – Discretize continuous attributes
• Choose the attribute with the highest Information Gain
• Create branches for each value of attribute
• Examples partitioned based on selected attributes
• Repeat with remaining attributes
• Stopping conditions  
  – All examples assigned the same label  
  – No examples left
Problems

- Expensive to train
- Prone to overfitting
  - Drive to perfection on training data, bad on test data
  - Pruning can help: remove or aggregate subtrees that provide little discriminatory power (C45)
## C4.5 Extensions

### Continuous Attributes

<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>overcast</td>
<td>cool</td>
<td>60</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>80</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>63</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>81</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>58</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>90</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>54</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>92</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>59</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>90</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>89</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>90</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>60</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>62</td>
<td>TRUE</td>
<td>yes</td>
</tr>
</tbody>
</table>
Consider every possible binary partition; choose the partition with the highest gain

<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>rainy</td>
<td>mild</td>
<td>54</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>58</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>59</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>60</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>60</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>62</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>63</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>80</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>81</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>89</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>90</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>90</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>90</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>92</td>
<td>FALSE</td>
<td>no</td>
</tr>
</tbody>
</table>

$\text{Expect} = \frac{8}{14} \times 0.95 + \frac{6}{14} \times 0 = 0.54$

$\text{Expect} = \frac{10}{14} \times 0.47 + \frac{4}{14} \times 0 = 0.33$
Ensemble Methods

Some slides are due to Zhuowen Tu, Robert Schapire and Pier Luca Lnzi
Ensemble Methods

Bagging (Breiman 1994, …)

Boosting (Freund and Schapire 1995, Friedman et al. 1998, …)

Random forests (Breiman 2001, …)

Predict class label for unseen data by aggregating a set of predictions (classifiers learned from the training data).
General Idea

Training Data

Multiple Data Sets
- $S_1$
- $S_2$
- $S_n$

Multiple Classifiers
- $C_1$
- $C_2$
- $C_n$

Combined Classifier $H$
Build Ensemble Classifiers

• Basic idea:

  Build different “experts”, and let them vote

• Advantages:

  Improve predictive performance
  Other types of classifiers can be directly included
  Easy to implement
  No too much parameter tuning

• Disadvantages:

  The combined classifier is not so transparent (black box)
  Not a compact representation
Why do they work?

• Suppose there are 25 base classifiers
• Each classifier has error rate, $\varepsilon = 0.35$
• Assume independence among classifiers
• Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1 - \varepsilon)^{25-i} = 0.06$$
Bagging

• **Training**
  
  o Given a dataset $S$, at each iteration $i$, a training set $s_i$ is sampled **with replacement** from $S$ (i.e. bootstraping)
  
  o A classifier $c_i$ is learned for each $s_i$

• **Classification**: given an unseen sample $x$,
  
  o Each classifier $c_i$ returns its class prediction
  
  o The bagged classifier $h$ counts the votes and assigns the class with the **most votes** to $x$

• **Regression**: can be applied to the prediction of continuous values by taking the **average value** of each prediction.
Bagging

• Bagging works because it reduces variance by voting/averaging
  o In some pathological hypothetical situations the overall error might increase
  o Usually, the more classifiers the better

• Problem: we only have one dataset.

• Solution: generate new ones of size $n$ by bootstrapping, i.e. sampling it with replacement

• Can help a lot if data is noisy.
When does Bagging work?

• **Learning algorithm is unstable**: if small changes to the training set cause large changes in the learned classifier.

• If the learning algorithm is unstable, then Bagging almost always improves performance

• Some candidates:

  Decision tree, linear regression, SVMs, decision stump, regression tree,

**Questions**: Why are decision trees or SVMs unstable?
Randomization

• Can randomize learning algorithms instead of inputs

• Some algorithms already have random component: e.g. random initialization

• Most algorithms can be randomized
  o Pick from the N best options at random instead of always picking the best one
  o Split rule in decision tree

• Random projection in kNN (Freund and Dasgupta 08)
Ensemble Methods

Bagging (Breiman 1994, ...)

Boosting (Freund and Schapire 1995, Friedman et al. 1998, ...)

Random forests (Breiman 2001, ...)

Boosting

• Idea: Build a classifier based on many weak learners
**Boosting**

- **Idea:** Build a classifier based on many weak learners
- **Defines a classifier using an additive model:**

\[ F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots \]

- **Strong classifier**
- **Weak classifier**
- **Features vector**
- **Weight**
It is a sequential procedure:

Each data point has a class label:
\[ y_t = \begin{cases} +1 & (\circ) \\ -1 & (\bullet) \end{cases} \]

and a weight:
\[ w_t = 1 \]
Toy example

Weak learners from the family of lines

Each data point has a class label:
\[ y_t = \begin{cases} 
+1 & (\bullet) \\
-1 & (\circ) 
\end{cases} \]

and a weight:
\[ w_t = 1 \]

\[ h \Rightarrow p(\text{error}) = 0.5 \] it is at chance
Each data point has a class label:

\[ y_t = \begin{cases} 
+1 & (\circ) \\
-1 & (\bullet) 
\end{cases} \]

and a weight:

\[ w_t = 1 \]

This one seems to be the best

This is a ‘weak classifier’: It performs slightly better than chance.
We set a new problem for which the previous weak classifier performs at chance again.

Each data point has a class label:

\[ y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
We set a new problem for which the previous weak classifier performs at chance again.

Each data point has a class label:

\[ y_t = \begin{cases} +1 & \text{(red)} \\ -1 & \text{(blue)} \end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
We set a new problem for which the previous weak classifier performs at chance again.

Each data point has a class label:

\[ y_t = \begin{cases} 
+1 & (\bullet) \\
-1 & (\circ) 
\end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
We set a new problem for which the previous weak classifier performs at chance again.

Each data point has a class label:

\[ y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases} \]

We update the weights:

\[ w_t \leftarrow w_t \exp\{-y_t H_t\} \]
The strong (non-linear) classifier is built as the combination of all the weak (linear) classifiers.
Given: m examples \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)

Initialize \(D_1(i) = 1/m\)

For \(t = 1\) to \(T\)

1. Train learner \(h_t\) with min error
2. Compute the hypothesis weight
   \[\varepsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i]\]
   \[\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right)\]
3. For each example \(i = 1\) to \(m\)
   \[D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} 
e \alpha_t & \text{if } h_t(x_i) = y_i \varepsilon_t & \text{if } h_t(x_i) \neq y_i \end{cases}\]

Output
\[H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)\]

**AdaBoost Algorithm**

The goodness of \(h_t\) is calculated over \(D_t\) and the bad guesses.

The weight **Adapts**. The bigger \(\varepsilon_t\) becomes the smaller \(\alpha_t\) becomes.

Boost example if incorrectly predicted.

**Z_t** is a normalization factor.

Linear combination of models.
Toy Example
\[ H_{\text{final}} = \text{sign}(0.42 + 0.65 + 0.92) \]

= 

=
Assume a weak classifier can only use one feature (e.g., \( x \leq 5 \rightarrow +1 \) and \( x > 5 \rightarrow -1 \)).

How many weak classifiers are needed to correctly identify all data items in the example above?

A) One weak classifier

B) Two weak classifiers with equal weights

C) Two weak classifiers with unequal weights

D) Three weak classifiers with equal weights

E) Three weak classifiers with unequal weights
Advantages of Boosting

• Simple and easy to implement
• Flexible—can combine with any learning algorithm
• No requirement on data metric—data features don’t need to be normalized, like in kNN and SVMs (this has been a central problem in machine learning)
• Feature selection and fusion are naturally combined with the same goal for minimizing an objective error function
• No parameters to tune
• No prior knowledge needed about weak learner
• Provably effective
• Versatile—can be applied on a wide variety of problems
• Non-parametric
Caveats

• Performance of AdaBoost depends on data and weak learner

• Consistent with theory, AdaBoost can fail if
  o weak classifier too complex—overfitting
  o weak classifier too weak -- underfitting

• Empirically, AdaBoost seems especially susceptible to uniform noise
Ensemble Methods

Bagging (Breiman 1994, ...)

Boosting (Freund and Schapire 1995, Friedman et al. 1998, ...)

Random forests (Breiman 2001, ...)
Motivation: You have To Hire A New Expert
Willow is one of the interviewers
But he is not really good in interviewing (yet)
So We Need More Interviewers
Setting I
Setting 2
Random Forests

- Random forests (RF) are a combination of tree predictors
- Each tree depends on the values of a random vector sampled in dependently
- The generalization error depends on the strength of the individual trees and the correlation between them
The Random Forests Algorithm

Given a training set $S$

For $i = 1$ to $k$ do:

- Build subset $s_i$ by sampling with replacement from $S$

  Learn tree $t_i$ from $s_i$

  At each node:

  - Choose best split from random subset of $F$ features

  - Each tree grows to the largest extent, and no pruning

Make predictions according to majority vote of the set of $k$ trees.
Features of Random Forests

• Boosting-decision tree (C4.5) often works very well.
• It runs efficiently on large data bases.
• It can handle thousands of input variables without variable deletion.
• It gives estimates of what variables are important in the classification.
• It generates an internal unbiased estimate of the generalization error as the forest building progresses.
• It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
• It has methods for balancing error in class population unbalanced data sets.
Compared with Boosting

Pros:

• It is more robust.

• It is faster to train (no reweighting, each split is on a small subset of data and feature).

• Can handle missing/partial data.

• Is easier to extend to online version.

Cons:

• The feature selection process is not explicit.

• Feature fusion is also less obvious.

• Has weaker performance on small size training data.
Ensemble Methods

• Random forests (also true for many machine learning algorithms) is an example of a tool that is useful in doing analyses of scientific data.

• But the cleverest algorithms are no substitute for human intelligence and knowledge of the data in the problem.

• Take the output of random forests not as absolute truth, but as smart computer generated guesses that may be helpful in leading to a deeper understanding of the problem.

Leo Brieman