Internet of Things
Data Analytics - Part 4
Classification and Time Series Analysis

Aveek Dutta
Assistant Professor
Electrical Engineering and Computer Science
University of Kansas
e-mail: aveekd@ku.edu
http://www.ittc.ku.edu/~aveekd
Classifiers

- Classification: assign labels to objects.
- Usually supervised: training set of pre-classified examples.
- Our examples:
  - Naïve Bayesian
  - Decision Trees
  - (and Logistic Regression)

Where in the catalog should I place this product listing?
Is this email spam?
Is this politician Democrat/Republican/Independent?
Naïve Bayesian Classifier

- Determine the most probable class label for each object
  - Based on the observed object attributes
    - Naïvely assumed to be conditionally independent of each other
      - Presence or absence of a particular feature is unrelated to presence or absence of others
  - Example:
    - Based on the object's attributes \{shape, color, weight\}
    - A given object that is \{spherical, yellow, < 60 grams\}, may be classified as a tennis ball
  - Class label probabilities are determined using Bayes’ Law

- Input variables are discrete (continuous variables are discretized)

- Output:
  - Probability score – proportional to the true probability
  - Class label – based on the highest probability score
Naïve Bayesian Classifier - Use Cases

- Preferred method for many text classification problems.
  - Try this first; if it doesn't work, try something more complicated

- Use cases
  - Spam filtering, other text classification tasks
  - Fraud detection
  - Recommendation systems
Building a Training Dataset to Predict Good or Bad Credit

- Predict the credit behavior of a credit card applicant from applicant's attributes:
  - Personal status
  - Job type
  - Housing type
  - Savings amount

- These are all categorical variables and are better suited to Naïve Bayesian Classifier than to logistic regression.

<table>
<thead>
<tr>
<th>personal_status</th>
<th>job</th>
<th>housing</th>
<th>savings_status</th>
<th>credit_class</th>
</tr>
</thead>
<tbody>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>for free</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>for free</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>for free</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>500&lt;X&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>high qualif/self emp/mgm</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male div/sep</td>
<td>unskilled resident</td>
<td>own</td>
<td>&gt;=1000</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>high qualif/self emp/mgm</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>unskilled resident</td>
<td>own</td>
<td>100&lt;=X&lt;500</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>high qualif/self emp/mgm</td>
<td>for free</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>500&lt;=X&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>rent</td>
<td>500&lt;=X&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>skilled</td>
<td>own</td>
<td>100&lt;=X&lt;500</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
</tbody>
</table>
Classification Problem

Should we color it Green or Yellow?

Depends on two things:
- What is the overall probability of a point $X$ being Green or Yellow ($P(Y)$, $P(G)$) (also called Prior)
- What is the likelihood of the point $X$ being green or yellow ($P(X|G)$, $P(X|Y)$)
- Task: Calculate the probability that the point is Green or Yellow ($P(G|X)$ and $P(Y|X)$) (Posterior Probability)
Technical Description - Bayes' Theorem

\[ P(C \mid A) = \frac{P(A \cap C)}{P(A)} = \frac{P(A \mid C)P(C)}{P(A)} \]

- C is the class label:
  - \( C \in \{C_1, C_2, \ldots, C_n\} \)
- A is the observed object attributes
  - \( A = (a_1, a_2, \ldots, a_m) \)
- \( P(C \mid A) \) is the probability of C given A is observed
  - Called the conditional probability or posterior probability
For observed attributes $A = (a_1, a_2, ... a_m)$, we want to compute

$$P(C_i | A) = \frac{P(a_1, a_2, ..., a_m | C_i)P(C_i)}{P(a_1, a_2, ..., a_m)} \quad i = 1, 2, ..., n$$

and assign the classifier, $C_i$, with the largest $P(C_i | A)$

Two simplifications to the calculations

- Apply naïve assumption - each $a_j$ is conditionally independent of each other, then

$$P(a_1, a_2, ..., a_m | C_i) = P(a_1 | C_i)P(a_2 | C_i) ... P(a_m | C_i) = \prod_{j=1}^{m} P(a_j | C_i)$$

- Denominator $P(a_1, a_2, ..., a_m)$ is a constant and can be ignored
Building a Naïve Bayesian Classifier

- Applying the two simplifications

\[ P(C_i | a_1, a_2, ..., a_m) \propto \left( \prod_{j=1}^{m} P(a_j | C_i) \right) P(C_i) \quad i = 1, 2, ..., n \]

- To build a Naïve Bayesian Classifier, collect the following statistics from the training data:
  - \( P(C_i) \) for all the class labels (Priors)
  - \( P(a_j | C_i) \) for all possible \( a_j \) and \( C_i \) (Likelihoods)
  - Assign the classifier label, \( C_i \), that maximizes the value of

\[ \left( \prod_{j=1}^{m} P(a_j | C_i) \right) P(C_i) \quad i = 1, 2, ..., n \]
Naïve Bayesian Classifiers for the Credit Example

- Class labels: \{good, bad\}
  - \(P(\text{good}) = 0.7\)
  - \(P(\text{bad}) = 0.3\)

- Conditional Probabilities (or likelihood)
  - \(P(\text{own}|\text{bad}) = 0.62\)
  - \(P(\text{own}|\text{good}) = 0.75\)
  - \(P(\text{rent}|\text{bad}) = 0.23\)
  - \(P(\text{rent}|\text{good}) = 0.14\)
  - ... and so on

<table>
<thead>
<tr>
<th>personal_status</th>
<th>job</th>
<th>housing</th>
<th>savings_status</th>
<th>credit_class</th>
</tr>
</thead>
<tbody>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>for free</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>for free</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>for free</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>500&lt;=x&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>high qualif/self emp/mgm</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>high qualif/self emp/mgm</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>unskilled resident</td>
<td>own</td>
<td>100&lt;=x&lt;500</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known savings</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>high qualif/self emp/mgm</td>
<td>for free</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>500&lt;=x&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>rent</td>
<td>500&lt;=x&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>100&lt;=x&lt;500</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>skilled</td>
<td>own</td>
<td>no known savings</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
</tbody>
</table>
Naïve Bayesian Classifier for a Particular Applicant

- Given applicant attributes of 
  \( A = \{\text{female single, owns home, self-employed, savings > $1000}\} \)

- Since \( P(\text{good}|A) > (\text{bad}|A) \), assign the applicant the label "good" credit

\[
\begin{array}{|c|c|c|}
\hline
a_j & C_i & P(a_j|C_i) \\
\hline
\text{female single} & \text{good} & 0.28 \\
\text{female single} & \text{bad} & 0.36 \\
\text{own} & \text{good} & 0.75 \\
\text{own} & \text{bad} & 0.62 \\
\text{self emp} & \text{good} & 0.14 \\
\text{self emp} & \text{bad} & 0.17 \\
\text{savings>1K} & \text{good} & 0.06 \\
\text{savings>1K} & \text{bad} & 0.02 \\
\hline
\end{array}
\]

\[
P(\text{good}|A) \sim (0.28 \times 0.75 \times 0.14 \times 0.06) \times 0.7 = 0.0012
\]

\[
P(\text{bad}|A) \sim (0.36 \times 0.62 \times 0.17 \times 0.02) \times 0.3 = 0.0002
\]
Naïve Bayesian Implementation Considerations

- **Numerical underflow**
  - Resulting from multiplying several probabilities near zero
  - Preventable by computing the logarithm of the products

- **Zero probabilities due to unobserved attribute/classifier pairs**
  - Resulting from rare events
  - Handled by smoothing (adjusting each probability by a small amount)

- **Assign the classifier label, \( C_i \), that maximizes the value of**

\[
\sum_{j=1}^{m} \log P'(a_j | C_i) + \log P(C_i)
\]

where \( i = 1, 2, \ldots, n \) and \( P' \) denotes the adjusted probabilities
Smoothing

- If one of the $P(a_j | C_i) = 0$ (i.e., not observed in the training data)
  - The numerator product of conditional probabilities $= 0$ (akin to overfitting)
- Smoothing adds "small non-zero" value to rare events
- Laplace Smoothing ($\varepsilon < 0.1$)
  \[ P' = \frac{\text{count}(x) + \varepsilon}{\sum_x [\text{count}(x) + \varepsilon]} \]
- Most software packages incorporate smoothing during calculation
Diagnostics

- **Hold-out data**
  - How well does the model classify new instances?

- **Cross-validation**

- **ROC curve/AUC**
  - Each point correspond to a threshold.
  - Higher Area Under Curve (AUC) is better

---

![Ideal Classifier]

![Useless Classifier (Everything Positive)]

![Random Classifier]

![Useless Classifier (Everything Negative)]
### Diagnostics: Confusion Matrix

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Prediction</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>good</td>
<td>bad</td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>671</td>
<td>29</td>
<td>700</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>38</td>
<td>262</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td></td>
<td>709</td>
<td>291</td>
<td>1000</td>
<td></td>
</tr>
</tbody>
</table>

- **true positives (TP)**: 671
- **false positives (FP)**: 38
- **false negatives (FN)**: 29
- **true negatives (TN)**: 262

**Overall success rate (or accuracy):**

\[
\frac{(TP + TN)}{(TP+TN+FP+FN)} = \frac{671+262}{1000} = 0.93
\]

- **TPR:**\[ \frac{TP}{(TP + FN)} = \frac{671}{(671+29)} = 671/700 = 0.96 \]
- **FPR:**\[ \frac{FP}{(FP + TN)} = \frac{38}{(38 + 262)} = 38/300 = 0.13 \]
- **FNR:**\[ \frac{FN}{(TP + FN)} = \frac{29}{(671 + 29)} = 29/700 = 0.04 \]

**Precision:**\[ \frac{TP}{(TP + FP)} = \frac{671}{709} = 0.95 \]

**Recall (or TPR):** \[ \frac{TP}{(TP + FN)} = 0.96 \]
### Naïve Bayesian Classifier - Choose (+) and Cautions (-)

<table>
<thead>
<tr>
<th>Reasons to Choose (+)</th>
<th>Cautions (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handles missing values quite well</td>
<td>Numeric variables have to be discrete (categorized) Intervals</td>
</tr>
<tr>
<td>Robust to irrelevant variables</td>
<td>Sensitive to correlated variables</td>
</tr>
<tr>
<td></td>
<td>&quot;Double-counting&quot;</td>
</tr>
<tr>
<td>Easy to implement</td>
<td>Not good for estimating probabilities</td>
</tr>
<tr>
<td></td>
<td>Stick to class label or yes/no</td>
</tr>
<tr>
<td>Easy to score data</td>
<td></td>
</tr>
<tr>
<td>Resistant to over-fitting</td>
<td></td>
</tr>
<tr>
<td>Computationally efficient</td>
<td></td>
</tr>
<tr>
<td>Handles very high dimensional problems</td>
<td></td>
</tr>
<tr>
<td>Handles categorical variables with a lot of levels</td>
<td></td>
</tr>
</tbody>
</table>
Naïve Bayesian in R

```r
> install.packages("e1071")
> library(e1071)

> sample<-read.table("sample1.csv", header=TRUE, sep="",)
> head(sample)

Age Income JobSatisfaction    Desire Enrolls
1     <=30   High              No         Fair          No
2     <=30   High              No     Excellent     No
3  31 to 40   High            No       Fair           Yes
4      >40   Medium         No        Fair          Yes
5      >40    Low               Yes        Fair           Yes
6      >40    Low               Yes      Excellent    No

> Traindata = as.data.frame(sample[1:14,])
> Testdata = as.data.frame(sample[15,])
> Testdata

Age Income JobSatisfaction Desire Enrolls
<=30 Medium Yes Fair TBD

> model<- naiveBayes(Enrolls~Age+Income+JobSatisfaction+Desire, Traindata)
> model

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y                  No       Yes
0.0000000 0.3571429 0.6428571

Conditional probabilities:
Age
Y          <=30       >40  31 to 40
No  0.6000000 0.4000000 0.0000000
Yes  0.2222222 0.3333333 0.4444444

Income
Y          High       Low    Medium
No  0.4000000 0.2000000 0.4000000
Yes  0.2222222 0.3333333 0.4444444

JobSatisfaction
Y            No       Yes
No  0.8000000 0.2000000
Yes  0.3333333 0.6666667

Desire
Y      Excellent      Fair
No  0.6000000 0.4000000
Yes 0.3333333 0.6666667

> results<-predict(model,Testdata)
> results

[1] Yes
Levels:  No Yes
```
Decision Trees
What is it?

- **Used for classification:**
  - Returns probability scores of class membership
    - Assigns label based on highest scoring class
    - Some Decision Tree algorithms return simply the most likely class
  - Regression Trees: a variation of regression
    - Returns average value at every node
    - Predictions can be discontinuous at the decision boundaries

- **Input:** Variables can be continuous or discrete

- **Output:**
  - A tree that describes the decision flow.
  - Leaf nodes return either a probability score, or simply a classification.
  - Trees can be converted to a set of "decision rules"
    - "IF income < $50,000 AND mortgage_amt > $100K THEN default=T with 75% probability"
Decision Tree - Example of Visual Structure

- **Female**
- **Male**

- **Gender**
  - **Female**
  - **Male**

- **Income**
  - ≤45,000
    - Yes
  - >45,000
    - No

- **Age**
  - ≤40
    - Yes
  - >40
    - No

- **Internal Node** – decision on variable
- **Branch** – outcome of test
- **Leaf Node** – class label
Decision Tree Classifier - Use Cases

- When a series of questions *(yes/no)* are answered to arrive at a classification
  - Biological species classification
  - Checklist of symptoms during a doctor’s evaluation of a patient
- When “if-then” conditions are preferred to linear models.
  - Customer segmentation to predict response rates
  - Financial decisions such as loan approval
  - Fraud detection
- Short Decision Trees are the most popular "weak learner" in ensemble learning techniques
Example: The Credit Prediction Problem

<table>
<thead>
<tr>
<th>personal_status</th>
<th>job</th>
<th>housing</th>
<th>savings_status</th>
<th>credit_class</th>
</tr>
</thead>
<tbody>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known status</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>for free</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>for free</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>for free</td>
<td>no known status</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>500&lt;=x&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>high qual/sel emp/ mgm</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male div/sep</td>
<td>unskilled resident</td>
<td>own</td>
<td>&gt;=1000</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>high qual/sel emp/ mgm</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>skilled</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>unskilled resident</td>
<td>own</td>
<td>100&lt;=x&lt;500</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>no known status</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>high qual/sel emp/ mgm</td>
<td>own</td>
<td>no known status</td>
<td>good</td>
</tr>
<tr>
<td>female div/dep/mar</td>
<td>high qual/sel emp/ mgm</td>
<td>for free</td>
<td>&lt;100</td>
<td>bad</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>500&lt;=x&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>skilled</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>rent</td>
<td>500&lt;=x&lt;1000</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>rent</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>100&lt;=x&lt;500</td>
<td>good</td>
</tr>
<tr>
<td>male single</td>
<td>unskilled resident</td>
<td>own</td>
<td>no known status</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
<tr>
<td>male mar/wid</td>
<td>unskilled resident</td>
<td>own</td>
<td>&lt;100</td>
<td>good</td>
</tr>
</tbody>
</table>

Housing

- Own: Good: 13, Bad: 4
- Rent: Good: 4, Bad: 2
- Free: Good: 2, Bad: 2

Ambiguous!! Ideally one of the attribute values should produce, Good = 0 or Bad = 0

If “Renting” then credit is always “Bad” - Unambiguous or Pure
After Classification

savings=(500:1000), >=1000, no known savings

personal=female, male div/sep

housing=free, rent

700/1000
p(good)=0.70

savings= <100, (100:500)

housing=own

245/294
p(good)=0.83

349/501
p(good)=0.70

personal=male mar/wid, male single

36/88
p(good) = 0.41

70/117
p(good)=0.60
Design goals

- Don’t have to test all possible paths
- Small and elegant (minimum, optimal, etc)
  - The training set can be incomplete
  - Complete training set is like a truth table (use logic optimization) - Easier problem
- Minimum tree - by identifying overriding variables
  - Similar to “Don’t Care” in logic design
General Algorithm

- To construct tree T from training set S
- If all examples in S belong to some class in C, or S is sufficiently "pure", then make a leaf labeled C
  - *Purity* is defined as a probability of the class (greater than a threshold)
  - Either all “Yes” or all “No” is PURE
- Otherwise:
  - Select the “most informative” attribute A
  - Partition S according to A’s values
  - Recursively construct sub-trees T1, T2, ..., for the subsets of S
- The details vary according to the specific algorithm – CART, ID3, C4.5
Entropy as a measure of Information (or lack of)

- Entropy-based methods are one common way

\[ H = - \sum_c p(c) \log_2 p(c) \]

- \( H = 0 \) if \( p(c) = 0 \) or 1 for any class
- So for binary classification, \( H=0 \) is a "pure" node
- \( H \) is maximum when all classes are equally probable
- For binary classification, \( H=1 \) when classes are 50/50
Step 1: Pick the most "informative" attribute

- First, we need to get the base entropy of the data

\[
H_{calculate} = -(0.7 \log_2(0.7) + 0.3 \log_2(0.3))
\]

\[
= 0.88
\]

- Then pick the Most “Informative” Attribute - Conditional Entropy

\[
H_{attr} = - \sum_v \sum_c p(v) p(c|v) \log_2 p(c|v)
\]

- The weighted sum of the class entropies for each value of the attribute
- Conditional entropy should be lower than unconditioned entropy
## Conditional Entropy Example

<table>
<thead>
<tr>
<th></th>
<th>for free</th>
<th>own</th>
<th>rent</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(\text{housing})$</td>
<td>0.108</td>
<td>0.713</td>
<td>0.179</td>
</tr>
<tr>
<td>$P(\text{bad} \mid \text{housing})$</td>
<td>0.407</td>
<td>0.261</td>
<td>0.391</td>
</tr>
<tr>
<td>$p(\text{good} \mid \text{housing})$</td>
<td>0.592</td>
<td>0.739</td>
<td>0.601</td>
</tr>
</tbody>
</table>

\[
H(\text{housing} \mid \text{credit}) = -[0.108 \times (0.407 \log_2(0.407) + 0.592 \log_2(0.592)) \\
+ 0.713 \times (0.261 \log_2(0.261) + 0.739 \log_2(0.739)) \\
+ 0.179 \times (0.391 \log_2(0.391) + 0.601 \log_2(0.601))]
= 0.868
\]
Step 1: Pick the Most “Informative” Attribute

(Continued) Information Gain

\[ \text{InfoGain}_{\text{attr}} = H - H_{\text{attr}} \]

- The information that you gain, by knowing the value of an attribute
- So the "most informative" attribute is the attribute with the highest \text{InfoGain}
Back to the Credit Prediction Example

\[
\text{InfoGain}_{\text{credit}} = H_{\text{credit}} - H_{\text{housing}|\text{credit}} \\
= 0.88 - 0.86 \\
\approx 0.013
\]

<table>
<thead>
<tr>
<th>Attribute</th>
<th>InfoGain</th>
</tr>
</thead>
<tbody>
<tr>
<td>job</td>
<td>0.001</td>
</tr>
<tr>
<td>housing</td>
<td>0.013</td>
</tr>
<tr>
<td>personal_status</td>
<td>0.006</td>
</tr>
<tr>
<td>savings_status</td>
<td>0.028</td>
</tr>
</tbody>
</table>
Step 2 & 3: Partition on the Selected Variable

- Step 2: Find the partition with the highest InfoGain
  - In our example the selected partition has InfoGain = 0.028
- Step 3: At each resulting node, repeat Steps 1 and 2
  - until node is "pure enough"
  - Pure nodes => no information gain by splitting on other attributes
- Note - A record can only belong to one node but attributes may appear in multiple splits
Decision Surface

- Always axis aligned splits
- Require deeper trees for comparable results to Linear regression or Naive Bayes, which may lead to over-fitting
- It is a greedy algorithm (not optimal, exponentially many trees)
Diagnostics

- ROC/AUC, Confusion Matrix
- Do the splits (or the "rules") make sense?
  - What does the domain expert say?
  - Significance of splits, Pruning over time
- How deep is the tree?
  - Too many layers are prone to over-fit
- Do you get nodes with very few members?
  - Over-fit
# Decision Trees in R

```r
cpyinstall.packages("rpart.plot")
> library("rpart")
> library("rpart.plot")

# Read the data
> play_decision <- read.table("DTdata.csv", header=TRUE, sep="",)
> play_decision
Play  Outlook Temperature Humidity  Wind
1   yes    rainy        cool   normal FALSE
2    no    rainy        cool   normal  TRUE
3   yes overcast         hot     high FALSE
4    no    sunny        mild     high FALSE
5   yes    rainy        cool   normal FALSE
6   yes    sunny        cool   normal FALSE
7   yes    rainy        cool   normal FALSE
8   yes    sunny         hot   normal FALSE
9   yes overcast        mild     high  TRUE
10  no    sunny        mild     high  TRUE

# build the decision tree
> fit <- rpart(Play ~ Outlook + Temperature + Humidity + Wind,
method="class",
data=play_decision,
control=rpart.control(minsplit=1),
parms=list(split='information'))

> rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE,
varlen=0, faclen=0)

> newdata <- data.frame(Outlook="rainy", Temperature="mild",
Humidity="high", Wind=FALSE)
> predict(fit,newdata,type="class")
1

Levels: no yes
```
## Decision Tree Classifier - Choose (+) & Cautions (-)

<table>
<thead>
<tr>
<th>Reasons to Choose (+)</th>
<th>Cautions (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Takes any input type (numeric, categorical)</td>
<td>Decision surfaces can only be axis-aligned</td>
</tr>
<tr>
<td>In principle, can handle categorical variables with many distinct values (ZIP code)</td>
<td></td>
</tr>
<tr>
<td>Robust with redundant variables, correlated variables</td>
<td>Tree structure is sensitive to small changes in the training data</td>
</tr>
<tr>
<td>Naturally handles variable interaction</td>
<td>A &quot;deep&quot; tree is probably over-fit</td>
</tr>
<tr>
<td>Handles variables that have non-linear effect on outcome</td>
<td>Because each split reduces the training data for subsequent splits</td>
</tr>
<tr>
<td>Computationally efficient to build</td>
<td>Not good for outcomes that are dependent on many variables</td>
</tr>
<tr>
<td>Easy to score data</td>
<td>Related to over-fit problem, above</td>
</tr>
<tr>
<td>Many algorithms can return a measure of variable importance</td>
<td>Doesn't naturally handle missing values;</td>
</tr>
<tr>
<td>In principle, decision rules are easy to understand</td>
<td>However most implementations include a method for dealing with this</td>
</tr>
<tr>
<td></td>
<td>In practice, decision rules can be fairly complex</td>
</tr>
</tbody>
</table>
Time Series Analysis
Time Series Analysis

- **Time Series:** Ordered sequence of equally spaced values over time
- **Time Series Analysis:** Accounts for the **internal structure** of observations taken over time
  - Trend
  - Seasonality
  - Cycles
  - Random

- **Goals**
  - To identify the internal structure of the time series
  - To forecast future events
    - Example: Based on sales history, what will next December sales be?

- **Method:** Box-Jenkins (ARMA)
Box-Jenkins Method: What is it?

- Models historical behavior to forecast the future
  - Condition data and select Model
    - Identify and account for trend and seasonality
  - Estimate model parameters
  - Assess the model and return to Step 1, if necessary
- Applies ARMA (Autoregressive Moving Averages)
- Input: Time Series
  - Accounting for Trends and Seasonality components
- Output: Expected future value of the time series
Modeling a Time Series

- Let's model the time series as

\[ Y_t = T_t + S_t + R_t \quad \text{for } t=1,...,n. \]

- **\( T_t \): Trend term**
  - Air travel steadily increased over the last few years

- **\( S_t \): The seasonal term**
  - Air travel fluctuates in a regular pattern over the course of a year

- **\( R_t \): Random component**
  - To be modeled with ARMA
Stationary Sequences

- Box-Jenkins methodology assumes the random component is a stationary sequence
  - Constant mean
  - Constant variance
  - Autocorrelation does not change over time
    - Constant correlation of a variable with itself at different times
- In practice, to obtain a stationary sequence, the data must be:
  - De-trended
  - Seasonally adjusted
Stationarity

- Mean and Variance does not change over time

- Covariance of $y_t$ and $y_{t+h}$ is a function of $h=0,1,2,...$
  
  - $\text{Cov} (y_t, y_{t+h}) = E[(y_t - \mu)(y_{t+h} - \mu)]$
  
  - $h=0$ gives the variance

- Stationarity can be derived by differencing samples
  
  - $D_t = y_t - y_{t-1}$
  
  - Second order $D_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$
De-trending

- In this example, we see a linear trend, so we fit a linear model
  - $T_t = m \cdot t + b$

- The de-trended series is then
  - $Y_t = Y_t - T_t$

- In some cases, may have to fit a non-linear model
  - Quadratic
  - Exponential
Seasonal Adjustment

- Plotting the de-trended series identifies seasons
- For CO2 concentration, we can model the period as being a year, with variation at the month level
- Simple ad-hoc adjustment: take several years of data, calculate the average value for each month, and subtract that from $Y^1_t$

$$Y^2_t = Y^1_t - S_t$$
ARMA(p, q) Model

\[ Y_t = \frac{\delta + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p}}{\epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}} \]

- The simplest Box-Jenkins Model
  - \( Y_t \) is de-trended and seasonally adjusted
  - Need “stationary” time-series
- Combination of two process models
  - **Autoregressive**: \( Y_t \) is a linear combination of its last \( p \) values
  - **Moving average**: \( Y_t \) is a constant value plus the effects of a dampened white noise process over the last \( q \) time values (lags)
Autocorrelation (ACF)

- Measure of similarity between observations as function of their lag
  - $ACF(h) = \frac{cov(h)}{cov(0)}$
- A mathematical to find repeating pattern in a time series
Partial Auto Correlation Function (PACF)

- Measure of when does series decorrelates, i.e., after how many lags
- Used to select the order of the AR () model
- \( \text{PACF}(h) = \text{corr}(y_t - y^*_t, y_{t+h} - y^*_{t+h}) \)
  - \( y^* \)'s are linear combination of samples between \( t+1 \) and \( t+h-1 \)
ARIMA(p, d, q) Model

- ARIMA adds a differencing term, $d$, to the ARMA model
  - Autoregressive Integrated Moving Average
  - Includes the de-trending as part of the model
    - linear trend can be removed by $d=1$
    - quadratic trend by $d=2$
    - and so on for higher order trends

- The general non-seasonal model is known as ARIMA $(p, d, q)$:
  - $p$ is the number of autoregressive terms
  - $d$ is the number of differences
  - $q$ is the number of moving average terms
Model Selection

- Based on the data, the Data Scientist selects $p$, $d$ and $q$
- An "art form" that requires domain knowledge, modeling experience, and a few iterations
- Use a simple model when possible
  - AR model ($q = 0$)
  - MA model ($p = 0$)
- Multiple models need to be built and compared
- Using ACF and PACF
## Time Series Analysis - to Choose (+) & Cautions (-)

<table>
<thead>
<tr>
<th>Reasons to Choose (+)</th>
<th>Cautions (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal data collection</td>
<td>No meaningful drivers: prediction based only on past performance</td>
</tr>
<tr>
<td>Only have to collect the series itself</td>
<td>No explanatory value</td>
</tr>
<tr>
<td>Do not need to input drivers</td>
<td>Can't do &quot;what-if&quot; scenarios</td>
</tr>
<tr>
<td>Design to handle the inherent autocorrelation of lagged time series</td>
<td>Can't stress test</td>
</tr>
<tr>
<td>Accounts for trends and seasonality</td>
<td>It's an &quot;art form&quot; to select appropriate parameters</td>
</tr>
<tr>
<td></td>
<td>Only suitable for short term predictions</td>
</tr>
</tbody>
</table>
Time Series Analysis with R

- The function “ts” is used to create time series objects
  - `mydata <- ts(mydata, start=c(1999,1), frequency=12)`
- Visualize data
  - `plot(mydata)`
- De-trend using differencing
  - `diff(mydata)`
- Examine ACF and PACF
  - `acf(mydata)`: It computes and plots estimates of the autocorrelations
  - `pacf(mydata)`: It computes and plots estimates of the partial autocorrelations
Other Useful R Functions in Time Series Analysis

- **ar()**: Fit an autoregressive time series model to the data
- **arima()**: Fit an ARIMA model
- **predict()**: Makes predictions
  - "predict" is a generic function for predictions from the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument
- **arima.sim()**: Simulate a time series from an ARIMA model
- **decompose()**: Decompose a time series into seasonal, trend and irregular components using moving averages
  - Deals with additive or multiplicative seasonal component
- **stl()**: Decompose a time series into seasonal, trend and irregular components using loess