DATA MINING
SUPERVISED LEARNING

Regression
Classification
    Decision Trees
Evaluation
Supervised learning

- In **supervised learning**, except for the feature variables that describe the data, we also have a **target variable**.
- The goal is to **learn** a function (model) that can predict the value of the target variable given the features.
  - We learn the function using a labeled **training set**.
- **Regression**: The target variable is numerical and continuous.
  - The price of a stock, the grade in a class, the height of a child, the life expectancy etc.
- **Classification**: The target variable is discrete.
  - Will the stock go up or down? Will the student pass or fail? Is a transaction fraudulent or not? What is the topic of an article?
- Predictive modeling is in the heart of the data science revolution.
LINEAR REGRESSION
Regression

- We assume that we have $k$ feature variables:
  - Also known as covariates, or independent variables
- The target variable is also known as dependent variable.
- We are given a dataset of the form $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where, $x_i$ is a $k$-dimensional feature vector, and $y_i$ a real value
- We want to learn a function $f$ which given a feature vector $x_i$ predicts a value $y_i' = f(x_i)$ that is as close as possible to the value $y_i$
- Minimize sum of squares:
  $$\sum_i (y_i - f(x_i))^2$$
Linear regression

- The simplest form of $f$ is a linear function
- In linear regression the function $f$ is typically of the form:

$$f(x_i) = w_0 + \sum_{j=1}^{k} w_j x_{ij}$$
One-dimensional linear regression

In the simplest case we have a single variable and the function is of the form:

\[ f(x_i) = w_0 + w_1 x_i \]

Minimizing the error gives:

\[ w_0 = \bar{y} - w_1 \bar{x} \]
\[ w_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = r_{xy} \frac{\sigma_y}{\sigma_x} \]

\( \bar{x} \): mean value of \( x_i \)’s
\( \bar{y} \): mean value of \( y_i \)’s
\( r_{xy} \): correlation coefficient between \( x, y \)
Multiple linear regression

• In the general case we have \( k \) features, and \( x_i, w \) are vectors.
• We simplify the notation:
  \[
  x_i = (1, x_{i1}, ..., x_{ik}) \\
  w = (w_0, w_1, ..., w_k) \\
  f(x_i, w) = x_i^T w
  \]
• Let \( X \) be the \( n \times (k + 1) \) matrix with vectors \( x_i \) as rows.
• Let \( y = (y_1, ..., y_n) \)
• We can write the SSE function as:
  \[
  SSE = \|Xw - y\|^2
  \]
• There is a closed-form solution for \( w \):
  \[
  w = (X^T X)^{-1} X^T y
  \]
• Matrix inversion may be too expensive. Other optimization techniques are often used to find the optimal vector (e.g., Gradient Descent)
Outliers

• Regression is sensitive to outliers:
  • The line will “tilt” to accommodate very extreme values
• Solution: remove the outliers
  • But make sure that they do not capture useful information
Normalization

- In the regression problem sometimes our features may have very different scales:
  - For example: predict the GDP of a country using as features the percentage of home owners and the income
  - The weights in this case will not be interpretable
- Solution: Normalize the features by replacing the values with the z-scores
More complex models

- The model we have is **linear** with respect to the parameters $w$ but the features we consider may be **non-linear functions** of the $x_i$ values.
- To capture more complex relationships we can take a transformation of the input (e.g., logarithm $\log x_{ij}$), or add polynomial terms (e.g., $x_{ij}^2$).
  - However this may increase a lot the number of features
Interpretation and significance

- A regression model is useful for making predictions for new data.
- The coefficients for the linear regression model are also useful for understanding the effect of the independent variables to the value of the dependent variable.
  - The \( w_j \) value is the effect of the increase of \( x_{ij} \) by one to the value \( y_i \).
- We can also compute the significance of the value of \( w_j \) by testing the null hypothesis that \( w_j = 0 \).

<table>
<thead>
<tr>
<th>Covariate</th>
<th>Least Squares Estimate</th>
<th>Estimated Standard Error</th>
<th>t value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>-589.39</td>
<td>167.59</td>
<td>-3.51</td>
<td>0.001 **</td>
</tr>
<tr>
<td>Age</td>
<td>1.04</td>
<td>0.45</td>
<td>2.33</td>
<td>0.025 *</td>
</tr>
<tr>
<td>Southern State</td>
<td>11.29</td>
<td>13.24</td>
<td>0.85</td>
<td>0.399</td>
</tr>
<tr>
<td>Education</td>
<td>1.18</td>
<td>0.68</td>
<td>1.7</td>
<td>0.093</td>
</tr>
<tr>
<td>Expenditures</td>
<td>0.96</td>
<td>0.25</td>
<td>3.86</td>
<td>0.000 ***</td>
</tr>
<tr>
<td>Labor</td>
<td>0.11</td>
<td>0.15</td>
<td>0.69</td>
<td>0.493</td>
</tr>
<tr>
<td>Number of Males</td>
<td>0.30</td>
<td>0.22</td>
<td>1.36</td>
<td>0.181</td>
</tr>
<tr>
<td>Population</td>
<td>0.09</td>
<td>0.14</td>
<td>0.65</td>
<td>0.518</td>
</tr>
<tr>
<td>Unemployment (14-24)</td>
<td>-0.68</td>
<td>0.48</td>
<td>-1.4</td>
<td>0.165</td>
</tr>
<tr>
<td>Unemployment (25-39)</td>
<td>2.15</td>
<td>0.95</td>
<td>2.26</td>
<td>0.030 *</td>
</tr>
<tr>
<td>Wealth</td>
<td>-0.08</td>
<td>0.09</td>
<td>-0.91</td>
<td>0.367</td>
</tr>
</tbody>
</table>

This table is typical of the output of a multiple regression program. The “t-value” is the Wald test statistic for testing \( H_0 : \beta_j = 0 \) versus \( H_1 : \beta_j \neq 0 \). The asterisks denote “degree of significance” with more asterisks being significant at a smaller level. The example raises several important questions. In particular: (1) should we eliminate some variables from this model? (2) should we interpret this relationships as causal? For example, should we conclude that low crime prevention expenditures cause high crime rates? We will address question (1) in the next section. We will not address question (2) until a later chapter.
CLASSIFICATION
Classification

- Similar to the regression problem we have features and a target variable that we want to model/predict
- The target variable is now discrete. It is often called the class label
  - In the simplest case, it is a binary variable.
Example: Catching tax-evasion

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Tax-return data for year 2011

A new tax return for 2012
Is this a cheating tax return?

An instance of the classification problem: learn a method for discriminating between records of different classes (cheaters vs non-cheaters)
Classification

- **Classification** is the task of *learning a target function* $f$ that maps attribute set $x$ to one of the predefined class labels $y$.

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

One of the attributes is the class attribute
In this case: Cheat

Two class labels (or classes): Yes (1), No (0)

Figure 4.2. Classification as the task of mapping an input attribute set $x$ into its class label $y$. 
Why classification?

• The target function \( f \) is known as a **classification model**

• **Descriptive modeling:** *Explanatory tool* to distinguish between objects of different classes (e.g., understand why people cheat on their taxes, or what makes a hipster)

• **Predictive modeling:** Predict a class of a **previously unseen** record
Examples of Classification Tasks

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Categorizing news stories as finance, weather, entertainment, sports, etc
- Identifying spam email, spam web pages, adult content
- Understanding if a web query has commercial intent or not

Classification is everywhere in data science. Big data has the answers to all questions.
General approach to classification

• Obtain a training set consisting of records with known class labels

• Training set is used to build a classification model

• A labeled test set of previously unseen data records is used to evaluate the quality of the model.

• The classification model is applied to new records with unknown class labels

• Important intermediate step: Decide on what features to use
Illustrating Classification Task

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
Evaluation of classification models

- Counts of test records that are correctly (or incorrectly) predicted by the classification model
- Confusion matrix

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class = 1</td>
<td>Class = 0</td>
</tr>
<tr>
<td>Class = 1</td>
<td>$f_{11}$</td>
</tr>
<tr>
<td>Class = 0</td>
<td>$f_{01}$</td>
</tr>
</tbody>
</table>

Accuracy = \[
\frac{\text{# correct predictions}}{\text{total # of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}
\]

Error rate = \[
\frac{\text{# wrong predictions}}{\text{total # of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}
\]
Classification Techniques

• Decision Tree based Methods
• Rule-based Methods
• Memory based reasoning
• Neural Networks
• Naïve Bayes and Bayesian Belief Networks
• Support Vector Machines
• Logistic Regression
Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
Decision Trees

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
### Example of a Decision Tree

#### Training Data

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### Model: Decision Tree

- **Refund**
  - Yes
  - NO
  - No

- **MarSt**
  - Single, Divorced
  - > 80K
    - NO
    - > 80K
      - YES

- **TaxInc**
  - < 80K
    - NO
  - > 80K
    - YES

#### Splitting Attributes

- Refund
- Marital Status (MarSt)
- Taxable Income (TaxInc)

#### Test outcome

- Yes
- No
- YES
- NO

#### Class labels

- NO
- YES
Another Example of Decision Tree

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Married

MarSt

Refund

TaxInc

< 80K

> 80K

There could be more than one tree that fits the same data!
Decision Tree Classification Task

### Training Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Test Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>

Tree Induction algorithm

- Induction
- Deduction

Learn Model

Apply Model

Model

Decision Tree

Test Set

Training Set
Apply Model to Test Data

Start from the root of tree.

Refund

Yes

NO

No

MarSt

Yes

Married

NO

No

Single, Divorced

TaxInc

< 80K

NO

> 80K

YES

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund

Yes

No

Marital Status

Single, Divorced

Married

Taxable Income

< 80K

> 80K

Cheat

No

YES
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>
Apply Model to Test Data

Refund

MarSt

TaxInc

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

No, Married, 80K, ?
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Refund: Yes → NO, No → MarSt
Marital Status: Single, Divorced, Married
Taxable Income: < 80K, > 80K
Cheat: NO, YES
Apply Model to Test Data

Test Data

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

Assign Cheat to “No”
## Decision Tree Classification Task

### Training Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Test Set

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>

---

**Tree Induction Algorithm**

- **Induction**
- **Deduction**

---

**Apply Model**

**Learn Model**

**Decision Tree**

**Model**
Tree Induction

• **Goal**: Find the tree that has low classification error in the training data (training error)

• Finding the **best** decision tree (lowest training error) is **NP-hard**

• **Greedy** strategy.
  • Split the records based on an attribute test that optimizes certain criterion.

• Many Algorithms:
  • Hunt’s Algorithm (one of the earliest)
  • CART
  • ID3, C4.5
  • SLIQ, SPRINT
General Structure of Hunt’s Algorithm

• Let $D_t$ be the set of training records that reach a node $t$

• General Procedure:
  • If $D_t$ contains records that belong the same class $y_t$, then $t$ is a leaf node labeled as $y_t$
  • If $D_t$ contains records with the same attribute values, then $t$ is a leaf node labeled with the majority class $y_t$
  • If $D_t$ is an empty set, then $t$ is a leaf node labeled by the default class, $y_d$
  • If $D_t$ contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.

• Recursively apply the procedure to each subset.
Hunt’s Algorithm

Refund

Don’t Cheat

Yes

Marital Status

Single, Divorced

Cheat

Don’t Cheat

No

Don’t Cheat

Marital Status

Single, Divorced

Cheat

Don’t Cheat

Taxable Income

< 80K

Don’t Cheat

Cheat

>= 80K

Refund

Don’t Cheat

Yes

No

Married

Taxable Income

120K

No
Constructing decision-trees (pseudocode)

**GenDecTree**(Sample $S$, Features $F$)

1. **If** stopping_condition($S,F$) = true **then**
   a. leaf = createNode()
   b. leaf.label = Classify($S$)
   c. return leaf

2. root = createNode()

3. root.test_condition = findBestSplit($S,F$)

4. $V = \{v \mid v$ a possible outcome of root.test_condition}$

5. **for each** value $v \in V$
   a. $S_v = \{s \mid root.test_condition(s) = v \text{ and } s \in S\}$;
   b. child = GenDecTree($S_v,F$);
   c. Add child as a descent of root and label the edge (root→child) as $v$

6. return root
Tree Induction

• Issues
  • How to **Classify** a leaf node
    • Assign the **majority class**
    • If leaf is empty, assign the **default class** – the class that has the highest popularity (overall or in the parent node).
  • Determine how to split the records
    • How to specify the attribute test condition?
    • How to determine the best split?
  • Determine when to stop splitting
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

- Depends on number of ways to split
  - 2-way split
  - Multi-way split
Splitting Based on Nominal Attributes

• **Multi-way split**: Use as many partitions as distinct values.

- CarType
  - Family
  - Sports
  - Luxury

• **Binary split**: Divides values into two subsets. Need to find optimal partitioning.

- CarType
  - {Sports, Luxury} OR {Family}
  - {Sports}
  - {Family, Luxury}

OR

- CarType
  - {Sports, Luxury} OR {Family}
  - {Sports}
  - {Family, Luxury}
Splitting Based on Ordinal Attributes

- **Multi-way split:** Use as many partitions as distinct values.

- **Binary split:** Divides values into two subsets – respects the order. Need to find optimal partitioning.

- What about this split?
Splitting Based on Continuous Attributes

- Different ways of handling
  - **Discretization** to form an **ordinal** categorical attribute
    - **Static** – discretize once at the beginning
    - **Dynamic** – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

- **Binary Decision**: \((A < v)\) or \((A \geq v)\)
  - consider all possible splits and finds the best cut
  - can be more computationally intensive
Splitting Based on Continuous Attributes

(i) Binary split

(ii) Multi-way split
How to determine the Best Split

Before Splitting: 10 records of class 0, 10 records of class 1

Own Car?
Yes
C0: 6  C0: 1
C1: 4  C1: 7
No
C0: 4  C0: 0
C1: 6  C1: 1

Car Type?
Family
C0: 1  C0: 0
C1: 3  C1: 7
Sports
Luxury
C0: 8  C0: 1
C1: 0  C1: 1

Student ID?
c1  c10  c11  c20
C0: 1  C0: 1  C0: 1  C0: 0
C1: 0  C1: 0  C1: 1  C1: 1

Which test condition is the best?
How to determine the Best Split

- **Greedy** approach:
  - Creation of nodes with *homogeneous* class distribution is preferred
- Need a measure of node *impurity*:

  \[
  \begin{array}{c|c}
  \text{C0} & 5 \\
  \text{C1} & 5 \\
  \end{array}
  \quad \quad \quad
  \begin{array}{c|c}
  \text{C0} & 9 \\
  \text{C1} & 1 \\
  \end{array}
  \]

  Non-homogeneous, 
  High degree of impurity

  Homogeneous, 
  Low degree of impurity

- Ideas?
Measuring Node Impurity

- We are at a node $D_t$ and the samples belong to classes $\{1, \ldots, c\}$
  - $p(i|t)$: fraction of records associated with node $D_t$ belonging to class $i$
- Impurity measures:
  
  \[
  \text{Entropy}(D_t) = - \sum_{i=1}^{c} p(i|t) \log p(i|t)
  \]
  - Used in ID3 and C4.5

  \[
  \text{Gini}(D_t) = 1 - \sum_{i=1}^{c} p(i|t)^2
  \]

  \[
  \text{Classification Error}(D_t) = 1 - \max p(i|t)
  \]
  - Used in CART, SLIQ, SPRINT.
Gain

- *Gain* of an attribute split into children \( \{v_1, ..., v_k\} \): compare the impurity of the parent node with the average impurity of the child nodes

\[
\Delta = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)
\]

- *Maximizing the gain*
  - \( \Leftrightarrow \) *Minimizing* the weighted average **impurity** of children nodes
  - \( \Leftrightarrow \) *Maximizing* average **purity**
- If \( I() = \text{Entropy()} \), then \( \Delta_{\text{info}} \) is called **information gain**
### Example

<table>
<thead>
<tr>
<th>C1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>6</td>
</tr>
</tbody>
</table>

\[
P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1
\]

\[
\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0
\]

\[
\text{Entropy} = - 0 \log 0 - 1 \log 1 = - 0 - 0 = 0
\]

\[
\text{Error} = 1 - \max (0, 1) = 1 - 1 = 0
\]

<table>
<thead>
<tr>
<th>C1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>5</td>
</tr>
</tbody>
</table>

\[
P(C1) = 1/6 \quad P(C2) = 5/6
\]

\[
\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278
\]

\[
\text{Entropy} = - (1/6) \log_2 (1/6) - (5/6) \log_2 (1/6) = 0.65
\]

\[
\text{Error} = 1 - \max (1/6, 5/6) = 1 - 5/6 = 1/6
\]

<table>
<thead>
<tr>
<th>C1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>4</td>
</tr>
</tbody>
</table>

\[
P(C1) = 2/6 \quad P(C2) = 4/6
\]

\[
\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444
\]

\[
\text{Entropy} = - (2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92
\]

\[
\text{Error} = 1 - \max (2/6, 4/6) = 1 - 4/6 = 1/3
\]
Impurity measures

- All of the impurity measures take value zero (minimum) for the case of a pure node where a single value has probability 1.
- All of the impurity measures take maximum value when the class distribution in a node is uniform.
Comparison among Splitting Criteria

For a 2-class problem:

The different impurity measures are consistent
Categorical Attributes

• For binary values split in two
• For multivalued attributes, for each distinct value, gather counts for each class in the dataset
  • Use the count matrix to make decisions

<table>
<thead>
<tr>
<th>CarType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
</tr>
<tr>
<td>Gini</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CarType</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Sports, Luxury}</td>
</tr>
<tr>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
</tr>
<tr>
<td>Gini</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CarType</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Sports}</td>
</tr>
<tr>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
</tr>
<tr>
<td>Gini</td>
</tr>
</tbody>
</table>

Multi-way split

Two-way split
(find best partition of values)
Continuous Attributes

- Use Binary Decisions based on one value
- Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, $A < v$ and $A \geq v$
- Exhaustive method to choose best $v$
  - For each $v$, scan the database to gather count matrix and compute the impurity index
  - Computationally Inefficient! Repetition of work.
Continuous Attributes

• For efficient computation: for each attribute,
  • **Sort** the attribute on values
  • Linearily scan these values, each time **updating** the count matrix and computing impurity
  • Choose the split position that has the least impurity

<table>
<thead>
<tr>
<th>Cheat</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxable Income</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>70</td>
<td>75</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>120</td>
<td>125</td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>65</td>
<td>72</td>
<td>80</td>
<td>87</td>
<td>92</td>
<td>97</td>
<td>110</td>
<td>122</td>
<td>172</td>
<td>230</td>
</tr>
<tr>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td>&lt;= &gt;</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>No</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Gini</td>
<td>0.420</td>
<td>0.400</td>
<td>0.375</td>
<td>0.343</td>
<td>0.417</td>
<td>0.400</td>
<td>0.300</td>
<td>0.343</td>
<td>0.375</td>
<td>0.400</td>
</tr>
</tbody>
</table>
Splitting based on impurity

- Impurity measures favor attributes with large number of values
- A test condition with large number of outcomes may not be desirable
  - # of records in each partition is too small to make predictions
Splitting based on INFO

Figure 4.12. Multiway versus binary splits.
Gain Ratio

- Splitting using information gain

\[
GainRATIO_{\text{split}} = \frac{GAIN_{\text{split}}}{\text{SplitINFO}}
\]

\[
\text{SplitINFO} = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}
\]

Parent Node, p is split into k partitions
\( n_i \) is the number of records in partition i

- Adjusts Information Gain by the entropy of the partition (SplitINFO). Higher entropy partition (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of impurity
Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- **Early termination** (to be discussed later)
Decision Tree Based Classification

• Advantages:
  • Inexpensive to construct
  • Extremely fast at classifying unknown records
  • Easy to interpret for small-sized trees
  • Accuracy is comparable to other classification techniques for many simple data sets
Example: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.

- You can download the software from:
  http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz
OTHER CLASSIFICATION ISSUES

Expressiveness
Overfitting
Evaluation
EXPRESSIVENESS
Expressiveness

- A classifier defines a function that discriminates between two (or more) classes.
- The expressiveness of a classifier is the class of functions that it can model, and the kind of data that it can separate.
  - When we have discrete (or binary) values, we are interested in the class of boolean functions that can be modeled.
  - If the data-points are real vectors we talk about the decision boundary that the classifier can model.
• Border line between two neighboring regions of different classes is known as decision boundary

• Decision boundary is parallel to axes because test condition involves a single attribute at-a-time
Limitations of single attribute-based decision boundaries

Both positive (+) and negative (o) classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.

The resulting boundary is very complex.
Oblique Decision Trees

- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive
Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree

- Less expressive for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time
OVERFITTING
Classification Errors

- **Training errors** (apparent errors)
  - Errors committed on the training set

- **Test errors**
  - Errors committed on the test set

- **Generalization errors**
  - Expected error of a model over random selection of records from same distribution
Example Data Set

Two class problem:

+ : 5400 instances
  - 5000 instances generated from a Gaussian centered at (10,10)
  - 400 noisy instances added

o : 5400 instances
  - Generated from a uniform distribution

10 % of the data used for training and 90% of the data used for testing
Increasing number of nodes in Decision Trees
Decision Tree with 4 nodes
Decision Tree with 50 nodes

Decision Tree

Decision boundaries on Training data

Error

Train Error

Number of Nodes
Which tree is better?
Model Overfitting

- **Underfitting**: when model is too simple, both training and test errors are large

- **Overfitting**: when model is too complex, training error is small but test error is large

- As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing
Bias – Variance tradeoff

• **Bias**: Measures how good the model is with respect to the training data
  - High Bias: Underfitting.
  - We have a poor model (e.g., a tree with a single decision node)

• **Variance**: Measures how sensitive the model error is with respect to changes in the training data
  - High Variance: Overfitting.
  - We have a very specific model (e.g., a tree with a single sample per leaf). Small changes in the data cause errors in the model

• There is a **tradeoff** between these two: decreasing one will increase the other.
Model Overfitting

Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model
Model Overfitting

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model.
Reasons for Model Overfitting

• Limited Training Size

• High Model Complexity
  • Multiple Comparison Procedure
Effect of Multiple Comparison Procedure

- Consider the task of predicting whether stock market will rise/fall in the next 10 trading days

- Random guessing:
  \[ P(\text{correct}) = 0.5 \]

- Make 10 random guesses in a row:

  \[
P(\# \text{correct} \geq 8) = \frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547
\]
Effect of Multiple Comparison Procedure

• Approach:
  • Get 50 analysts
  • Each analyst makes 10 random guesses
  • Choose the analyst that makes the most number of correct predictions

• Probability that at least one analyst makes at least 8 correct predictions

\[ P(\# correct \geq 8) = 1 - (1 - 0.0547)^{50} = 0.9399 \]
Effect of Multiple Comparison Procedure

- Many algorithms employ the following greedy strategy:
  - Initial model: $M$
  - Alternative model: $M' = M$, where $\gamma$ is a component to be added to the model (e.g., a test condition of a decision tree)
  - Keep $M'$ if improvement, $\Delta(M, M') > \alpha$

- Often times, $\gamma$ is chosen from a set of alternative components, $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$

- If many alternatives are available, one may inadvertently add irrelevant components to the model, resulting in model overfitting
Use additional 100 noisy variables generated from a uniform distribution along with X and Y as attributes.

Use 30% of the data for training and 70% of the data for testing.
Notes on Overfitting

• Overfitting results in decision trees that are more complex than necessary

• Training error no longer provides a good estimate of test error, that is, how well the tree will perform on previously unseen records
• We say that the model does not generalize well

• Generalization: The ability of the model to predict data points that it has not already seen.

• Need ways for estimating generalization errors
Model Selection

- Performed during model building
- Purpose is to ensure that model is not overly complex (to avoid overfitting)
- Need to estimate generalization error
  - Using Validation Set
  - Incorporating Model Complexity
Model Selection: Using Validation Set

• Divide training data into two parts:
  • Training set:
    • Use for model building
  • Validation set:
    • Use for estimating generalization error
    • Note: validation set is not the same as test set since it affects the creation of the model (e.g. in tuning a parameter)

• Drawback:
  • Less data available for training
Occam’s Razor

- **Occam’s razor**: All other things being equal, the simplest explanation/solution is the best.
  - A good principle for life as well

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

- For complex models, there is a greater chance that it was fitted accidentally by errors in data

- Therefore, one should include model complexity when evaluating a model
Minimum Description Length (MDL)

- \( \text{Cost(Model,Data)} = \text{Cost(Model)} + \text{Cost(Data|Model)} \)
  - Search for the least costly model.

- \( \text{Cost(Model)} \) encodes the decision tree
  - node encoding (number of children) plus splitting condition encoding.
- \( \text{Cost(Data|Model)} \) encodes the misclassification errors.

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>1</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>0</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>0</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>1</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( x_n )</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>?</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>?</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>?</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>?</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( x_n )</td>
<td>?</td>
</tr>
</tbody>
</table>
Example

- **Regression**: find a polynomial for describing a set of values
  - **Model complexity** (model cost): polynomial coefficients
  - **Goodness of fit** (data cost): difference between real value and the polynomial value

Source: Grunwald et al. (2005) *Tutorial on MDL.*

MDL avoids **overfitting** automatically!
Model Selection: Incorporating Model Complexity

- **Occam’s razor**: All other things being equal, the simplest explanation/solution is the best.
  - A good principle for life as well

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

- For complex models, there is a greater chance that it was fitted accidentally

- Therefore, one should include model complexity when evaluating a model

\[
\text{Gen. Error(Model)} = \text{Train. Error(Model, Train. Data)} + \alpha \times \text{Complexity(Model)}
\]
Estimating the Complexity of Decision Trees

- **Resubstitution Estimate:**
  - Using *training error* as an optimistic estimate of *generalization error*
  - Referred to as *optimistic error estimate*

\[
e(T_L) = \frac{4}{24}
\]
\[
e(T_R) = \frac{6}{24}
\]
Estimating the Complexity of Decision Trees

- **Pessimistic Error Estimate** of decision tree $T$ with $k$ leaf nodes:

$$
err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}
$$

- $err(T)$: error rate on all training records
- $\Omega$: trade-off hyper-parameter (similar to $\alpha$)
  - Relative cost of adding a leaf node
- $k$: number of leaf nodes
- $N_{train}$: total number of training records
Estimating the Complexity of Decision Trees: Example

\[ e(TL) = \frac{4}{24} \]
\[ e(TR) = \frac{6}{24} \]
\[ \Omega = 1 \]

\[ e_{gen}(TL) = \frac{4}{24} + 1 \times \frac{7}{24} = \frac{11}{24} = 0.458 \]
\[ e_{gen}(TR) = \frac{6}{24} + 1 \times \frac{4}{24} = \frac{10}{24} = 0.417 \]
Minimum Description Length (MDL)

Cost(Model, Data) = Cost(Model) + Cost(Data|Model)

- Cost is the number of bits needed for encoding.
- Search for the least costly model.

- Cost(Model) encodes the decision tree
  - node encoding (number of children) plus splitting condition encoding.
- Cost(Data|Model) encodes the misclassification errors.
Example

- **Regression**: find a polynomial for describing a set of values
  - **Model complexity** (model cost): polynomial coefficients
  - **Goodness of fit** (data cost): difference between real value and the polynomial value

Source: Grunwald et al. (2005) *Tutorial on MDL.*
Model selection for Decision Trees

• Pre-Pruning (Early Stopping Rule)
  • Stop the algorithm before it becomes a fully-grown tree

• Typical stopping conditions for a node:
  • Stop if all instances belong to the same class
  • Stop if all the attribute values are the same

• More restrictive conditions:
  • Stop if number of instances is less than some user-specified threshold
  • Stop if class distribution of instance classes are independent of the available features (e.g., using $\chi^2$ test)
  • Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
Model selection for Decision Trees

• Post-pruning
  • Grow decision tree to its entirety
  • Trim the nodes of the decision tree in a bottom-up fashion
  • If generalization error improves after trimming, replace sub-tree by a leaf node (subtree pruning) or by the most probable subtree (subtree raising).
  • Class label of leaf node is determined from majority class of instances in the sub-tree

• Can use MDL for post-pruning
  • NP hard problem
Example of Post-Pruning

Optimistic (Training) Error (Before splitting) = 10/30
Pessimistic Error = (10 + 0.5)/30 = 10.5/30

Optimistic (Training) Error (After splitting) = 9/30
Pessimistic Error (After splitting) = (9 + 4 \times 0.5)/30 = 11/30

PRUNE!
Examples of Post-pruning

Decision Tree:

```
depth = 1:
  breadth > 7 : class 1
  breadth <= 7:
    breadth <= 3:
      ImagePages > 0.375 : class 0
      ImagePages <= 0.375:
        totalPages <= 6 : class 1
        totalPages > 6:
          breadth <= 6 : class 0
          breadth > 6:
            MultiAgent = 0:
            MultiAgent = 1:
              totalPages <= 81 : class 0
              totalPages > 81 : class 1
  width > 3:
    MultiIP = 0:
      ImagePages <= 0.1333 : class 1
      ImagePages > 0.1333:
        breadth <= 6 : class 0
        breadth > 6:
          RepeatedAccess <= 0.0322 : class 0
          RepeatedAccess > 0.0322 : class 1
    MultiIP = 1:
      MultiAgent = 0:
      MultiAgent = 1:
        totalPages <= 81 : class 0
        totalPages > 81 : class 1
  depth > 1:
    MultiAgent = 0:
      depth > 2 : class 0
      depth <= 2:
        MultiIP = 1: class 0
        MultiIP = 0:
          breadth <= 6 : class 0
          breadth > 6:
            RepeatedAccess <= 0.0322 : class 0
            RepeatedAccess > 0.0322 : class 1
    MultiAgent = 1:
      totalPages <= 81 : class 0
      totalPages > 81 : class 1
```

Simplified Decision Tree:

```
depth = 1:
  ImagePages <= 0.1333 : class 1
  ImagePages > 0.1333:
    breadth <= 6 : class 0
    breadth > 6:
      MultiAgent = 0:
      MultiAgent = 1:
        totalPages <= 81 : class 0
        totalPages > 81 : class 1
```
MODEL EVALUATION
Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?

- Methods for Performance Evaluation
  - How to obtain reliable estimates?

- Methods for Model Comparison
  - How to compare the relative performance among competing models?
Metrics for Performance Evaluation

- Focus on the **predictive capability** of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>a</td>
</tr>
<tr>
<td>Class=No</td>
<td>c</td>
</tr>
</tbody>
</table>

a: TP (true positive)
b: FN (false negative)
c: FP (false positive)
d: TN (true negative)
### Metrics for Performance Evaluation…

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>Class=Yes</td>
</tr>
<tr>
<td></td>
<td>a (TP)</td>
</tr>
<tr>
<td>Class=No</td>
<td>b (FN)</td>
</tr>
<tr>
<td></td>
<td>c (FP)</td>
</tr>
<tr>
<td></td>
<td>d (TN)</td>
</tr>
</tbody>
</table>

- Most widely-used metric:

\[
\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}
\]
Precision-Recall

Precision (p) = \( \frac{a}{a + c} = \frac{TP}{TP + FP} \)

Recall (r) = \( \frac{a}{a + b} = \frac{TP}{TP + FN} \)

F-measure (F) = \( \frac{1}{\frac{1}{r} + \frac{1}{p}} \) = \( \frac{2rp}{r + p} \) = \( \frac{2a}{2a + b + c} \) = \( \frac{2TP}{2TP + FP + FN} \)

Assumption: The class YES is the one we care about.

- Precision is biased towards \( C(Yes|Yes) \) & \( C(Yes|No) \)
- Recall is biased towards \( C(Yes|Yes) \) & \( C(No|Yes) \)
- F-measure is biased towards all except \( C(No|No) \)
More Measures of Classification Performance

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>TP</td>
</tr>
<tr>
<td>No</td>
<td>FN</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

\(\alpha\) is the probability that we reject the null hypothesis when it is true.

This is a **Type I error** or a false positive (FP).

\(\beta\) is the probability that we accept the null hypothesis when it is false.

This is a **Type II error** or a false negative (FN).

**Accuracy**

\[
\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}
\]

**Error Rate**

\[
\text{Error Rate} = 1 - \text{accuracy}
\]

**Precision**

\[
\text{Precision} = \text{Positive Predictive Value} = \frac{TP}{TP + FP}
\]

**Recall = Sensitivity = TP Rate**

\[
\text{Recall} = \text{Sensitivity} = \text{TP Rate} = \frac{TP}{TP + FN}
\]

**Specificity = TN Rate**

\[
\text{Specificity} = \text{TN Rate} = \frac{TN}{TN + FP}
\]

**FP Rate**

\[
\text{FP Rate} = \alpha = \frac{FP}{TN + FP} = 1 - \text{specificity}
\]

**FN Rate**

\[
\text{FN Rate} = \beta = \frac{FN}{FN + TP} = 1 - \text{sensitivity}
\]

**Power = sensitivity = 1 - \beta**
ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- **ROC** curve plots TPR (true positive rate) (on the y-axis) against FPR (false positive rate) (on the x-axis)

Look at the **positive** predictions of the classifier and compute:

\[
TPR = \frac{TP}{TP + FN}
\]

What fraction of true **positive instances** are predicted **correctly**? (1-Type II error rate)

\[
FPR = \frac{FP}{FP + TN}
\]

What fraction of true **negative instances** were predicted **incorrectly**? (Type I error rate)

We want to strike a balance between these two
ROC (Receiver Operating Characteristic)

- Performance of a classifier represented as a point on the ROC curve
- Changing some parameter of the algorithm, sample distribution, or cost matrix changes the location of the point
ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)
- any points located at $x > t$ is classified as positive

At threshold $t$:
TP=0.5, FN=0.5, FP=0.12, FN=0.88
ROC Curve

(TP,FP):
• (0,0): declare everything to be negative class
• (1,1): declare everything to be positive class
• (1,0): ideal

Diagonal line:
• Random guessing
• Below diagonal line:
  • prediction is opposite of the true class

<table>
<thead>
<tr>
<th>Actual</th>
<th>PREDICTED CLASS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>a (TP)</td>
<td>b (FN)</td>
</tr>
<tr>
<td>No</td>
<td>c (FP)</td>
<td>d (TN)</td>
</tr>
</tbody>
</table>
Using ROC for Model Comparison

- No model consistently outperform the other
  - $M_1$ is better for small FPR
  - $M_2$ is better for large FPR

- Area Under the ROC curve (AUC)
  - Ideal: Area = 1
  - Random guess:
    - Area = 0.5
Precision-Recall plot

- Usually for parameterized models, it controls the precision/recall tradeoff
ROC curve vs Precision-Recall curve

Area Under the Curve (AUC) as a single number for evaluation
Methods of Performance Estimation

- **Holdout**
  - Reserve \(\frac{2}{3}\) for training and \(\frac{1}{3}\) for testing

- **Random subsampling**
  - One sample may be biased -- Repeated holdout

- **Cross validation**
  - Partition data into \(k\) disjoint subsets
  - \(k\)-fold: train on \(k-1\) partitions, test on the remaining one
  - Leave-one-out: \(k=n\)
  - Guarantees that each record is used the same number of times for training and testing

- **Bootstrap**
  - Sampling with replacement
  - \(~63\%\) of records used for training, \(~27\%\) for testing
Class imbalance

• Consider a 2-class problem
  • Number of Class 0 examples = 9990
  • Number of Class 1 examples = 10

• If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9%
  • Accuracy is misleading because model does not detect any class 1 example
  • Precision and recall are better measures
Dealing with class Imbalance

- Class imbalance is a problem in training:
  - If the class we are interested in is very rare, then the classifier will ignore it.

- Solution
  - We can balance the class distribution
    - Sample from the larger class so that the size of the two classes is the same
    - Replicate the data of the class of interest so that the classes are balanced
      - Over-fitting issues
  - We can modify the optimization criterion by using a cost sensitive metric
Cost Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(i</td>
<td>j)</td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>C(Yes</td>
<td>Yes)</td>
<td>C(No</td>
</tr>
<tr>
<td>Class=No</td>
<td>C(Yes</td>
<td>No)</td>
<td>C(No</td>
</tr>
</tbody>
</table>

\[ C(i|j) : \text{Cost of classifying class } j \text{ example as class } i \]
### Weighted Accuracy

#### Confusion Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>( a )</td>
</tr>
<tr>
<td>Class=No</td>
<td>( c )</td>
</tr>
<tr>
<td>Class=No</td>
<td>( b )</td>
</tr>
</tbody>
</table>

- \( a \) (True Positive, TP)
- \( b \) (False Negative, FN)
- \( c \) (False Positive, FP)
- \( d \) (True Negative, TN)

#### Cost Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>Class=Yes</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>( w_1 )</td>
</tr>
<tr>
<td>Class=No</td>
<td>( w_2 )</td>
</tr>
<tr>
<td>Class=No</td>
<td>( w_3 )</td>
</tr>
</tbody>
</table>

- \( w_1 \) (Cost of Yes|Yes)
- \( w_2 \) (Cost of No|Yes)
- \( w_3 \) (Cost of Yes|No)
- \( w_4 \) (Cost of No|No)

#### Weighted Accuracy

\[
\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}
\]
# Computing Cost of Classification

## Cost Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>C(ij)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>100</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

## Model $M_1$

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>150</td>
<td>40</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>60</td>
<td>250</td>
</tr>
</tbody>
</table>

Accuracy = 80%
Weighted Accuracy = 8.9%

## Model $M_2$

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>250</td>
<td>45</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>5</td>
<td>200</td>
</tr>
</tbody>
</table>

Accuracy = 90%
Weighted Accuracy = 9%
## Classification Cost

### Confusion Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>Class=Yes</td>
</tr>
<tr>
<td></td>
<td>a (TP)</td>
</tr>
<tr>
<td>Class=No</td>
<td>c (FP)</td>
</tr>
</tbody>
</table>

### Cost Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>$C(i</td>
</tr>
<tr>
<td></td>
<td>$w_1 C(Yes</td>
</tr>
<tr>
<td>Class=No</td>
<td>$w_3 C(Yes</td>
</tr>
</tbody>
</table>

### Classification Cost

$$\text{Classification Cost} = w_1 a + w_2 b + w_3 c + w_4 d$$

Some weights can also be negative
Computing Cost of Classification

## Cost Matrix

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>C(ij)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>-1</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

### Model M₁

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>150</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>60</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>250</td>
</tr>
</tbody>
</table>

Accuracy = 80%
Cost = 3910

### Model M₂

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>250</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>45</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>5</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>200</td>
</tr>
</tbody>
</table>

Accuracy = 90%
Cost = 4255
### Cost vs Accuracy

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Cost</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>Class=Yes</td>
<td>a</td>
</tr>
<tr>
<td>Class=Yes</td>
<td>Class=No</td>
<td>b</td>
</tr>
<tr>
<td>Class=No</td>
<td>Class=Yes</td>
<td>c</td>
</tr>
<tr>
<td>Class=No</td>
<td>Class=No</td>
<td>d</td>
</tr>
</tbody>
</table>

\[
N = a + b + c + d
\]

\[
\text{Accuracy} = \frac{a + d}{N}
\]

\[
\text{Cost} = p(a + d) + q(b + c)
\]

\[
= p(a + d) + q(N - a - d)
\]

\[
= qN - (q - p)(a + d)
\]

\[
= N[q - (q - p) \times \text{Accuracy}]
\]

Accuracy is proportional to cost if
1. \(C(Yes|No)=C(No|Yes) = q\)
2. \(C(Yes|Yes)=C(No|No) = p\)