Suppose you have a car insurance company and you have collected the following historical data:

Example: if $16 < \text{age} < 25$ AND car type is sports or truck, then risk is high.

<table>
<thead>
<tr>
<th>Age</th>
<th>CarType</th>
<th>HighRisk</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Sedan</td>
<td>F</td>
</tr>
<tr>
<td>30</td>
<td>Sports</td>
<td>F</td>
</tr>
<tr>
<td>36</td>
<td>Sedan</td>
<td>F</td>
</tr>
<tr>
<td>25</td>
<td>Truck</td>
<td>T</td>
</tr>
<tr>
<td>30</td>
<td>Sedan</td>
<td>F</td>
</tr>
<tr>
<td>23</td>
<td>Truck</td>
<td>T</td>
</tr>
<tr>
<td>30</td>
<td>Truck</td>
<td>F</td>
</tr>
<tr>
<td>25</td>
<td>Sports</td>
<td>T</td>
</tr>
<tr>
<td>18</td>
<td>Sedan</td>
<td>F</td>
</tr>
</tbody>
</table>
Classification & Regression Rules

- **Classification rules**
  - Dependent attribute is categorical

- **Regression rules**
  - Dependent attribute is numerical

- **Support for C1 → C2**
  - Percentage of tuples that satisfy C1 and C2

- **Confidence for C1 → C2**
  - Percentage of tuples satisfying C1 that also satisfy C2.

<table>
<thead>
<tr>
<th>Age</th>
<th>CarType</th>
<th>Expenditure</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Sedan</td>
<td>200</td>
</tr>
<tr>
<td>30</td>
<td>Sports</td>
<td>150</td>
</tr>
<tr>
<td>36</td>
<td>Sedan</td>
<td>300</td>
</tr>
<tr>
<td>25</td>
<td>Truck</td>
<td>220</td>
</tr>
<tr>
<td>30</td>
<td>Sedan</td>
<td>400</td>
</tr>
<tr>
<td>23</td>
<td>Truck</td>
<td>80</td>
</tr>
<tr>
<td>30</td>
<td>Truck</td>
<td>100</td>
</tr>
<tr>
<td>25</td>
<td>Sports</td>
<td>125</td>
</tr>
<tr>
<td>18</td>
<td>Sedan</td>
<td>500</td>
</tr>
</tbody>
</table>
Decision Trees

- Classification rules that can be structured as trees.
- Trees for regression rules are called regression trees.
- A decision tree $T$ encodes a (a classifier or regression function) in form of a tree.
- A node $t$ in $T$ without children is called a leaf node. Otherwise $t$ is called an internal node.
Top-down Decision Tree Algorithm

- Examine training database and find best splitting predicate for the root node
- Partition training database
- Recurse on each child node

**BuildTree**(Node $t$, Training database $D$, Split Selection Method $S$)

1. Apply $S$ to $D$ to find splitting criterion
2. if ($t$ is not a leaf node)
3. Create children nodes of $t$
4. Partition $D$ into children partitions
5. Recurse on each partition
6. endif
Split Selection Method

• Two decisions:
  – What is the splitting attribute
  – What is the splitting criterion

• **Numerical** or ordered attributes:
  – Find a split point that separates the (two) classes

• **Categorical attributes:**
  – evaluate all possible partitions
Clustering

- Decision trees learn to predict class labels given predictor attributes -- *supervised learning*.
- What if no class labels are available?
  - Find patterns via *unsupervised learning*
- **Clustering** is the process of organizing objects into groups whose members are similar in some way
  - Given:
    - Data Set D (training set)
    - Similarity/distance metric/information
  - Find:
    - Partitioning of data
    - Groups of similar/close items
Clustering Visually

Clustering

centroids
Why are clusters useful?
Applications

- **Marketing**
  - finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records;

- **Biology**
  - classification of plants and animals given their features;

- **Insurance**
  - identifying groups of motor insurance policy holders with a high average claim cost
  - identifying frauds;

- **Earthquake studies**
  - clustering observed earthquake epicenters to identify dangerous zones;

- **WWW**
  - document classification; clustering weblog data to discover groups of similar access patterns.
Minkowski Distance (L_p Norm)

• Consider two records $x=(x_1,\ldots,x_d)$, $y=(y_1,\ldots,y_d)$:

$$d(x, y) = \sqrt[p]{|x_1 - y_1|^p + |x_2 - y_2|^p + \ldots + |x_d - y_d|^p}$$

Special cases:
• $p=1$: Manhattan distance

$$d(x, y) = |x_1 - y_1| + |x_2 - y_2| + \ldots + |x_p - y_p|$$

• $p=2$: Euclidean distance

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_d - y_d)^2}$$
Properties of Distances: Metric Spaces

• A metric space is a set $S$ with a global distance function $d$. For every two points $x, y$ in $S$, the distance $d(x,y)$ is a nonnegative real number.

• A metric space must also satisfy
  - $d(x,y) = 0$ iff $x = y$
  - $d(x,y) = d(y,x)$ (symmetry)
  - $d(x,y) + d(y,z) \geq d(x,z)$ (triangle inequality)
Clustering: Informal Problem Definition

Input:
• A data set of $N$ records each given as a $d$-dimensional data feature vector.

Output:
• Determine a natural, useful “partitioning” of the data set into a number of (k) clusters and noise such that we have:
  – High similarity of records within each cluster (intra-cluster similarity)
  – Low similarity of records between clusters (inter-cluster similarity)
Clustering Algorithms

• Partitioning-based clustering
  – K-means clustering
  – K-medoids clustering
  – EM (expectation maximization) clustering

• Hierarchical clustering
  – Divisive clustering (top down)
  – Agglomerative clustering (bottom up)

• Density-Based Methods
  – Regions of dense points separated by sparser regions of relatively low density
K-means Clustering Algorithm

Initialize k cluster centers

Do

1. **Assignment step**: Assign each data point to its closest cluster center

2. **Re-estimation step**: Re-compute cluster centers

While (there are still changes in the cluster centers)

Visualization at:

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html
Issues with K-means

Why is K-Means working:

• How does it find the cluster centers?
• Does it find an optimal clustering
• What are good starting points for the algorithm?
• What is the right number of cluster centers?
• How do we know it will terminate?