Chapter 4: Classification & Prediction

- 4.1 Basic Concepts of Classification and Prediction
- 4.2 Decision Tree Induction
- 4.3 Bayes Classification Methods
- 4.4 Rule Based Classification
  - 4.4.1 The principle
  - 4.4.2 Extracting Rules form a Decision Tree
  - 4.4.3 Rule Induction using a Sequential Covering Algorithm
- 4.5 Lazy Learners
- 4.6 Prediction
- 4.7 How to Evaluate and Improve Classification
4.4.1 The Principle

- The model learned in Rule-Based classification is represented as a set of **IF-THEN** rules

  **IF** condition **THEN** conclusion

- **Example**
  
  **R1**: IF age=youth AND student=yes THEN buys_computer=yes

- **Terminology**

  - The “IF” part is known as the **rule antecedent** or **precondition**
    - Consists of one or more attributes
  
  - The “THEN” part is known as **rule consequent**
    - Contains a class prediction
  
  - If the condition in a rule antecedent holds true we say
    - The condition is **satisfied**
    - The rule **covers** the tuple
How to Assess the Rules

- A rule $R$ can be assessed by
  - Coverage
  - Accuracy

- **Methodology**
  - Tuple $X$
  - Class labeled Data set $D$

Consider
- $N_{covers}$: the number of tuples covered by $R$
- $n_{correct}$: the number of tuples correctly classified by $R$
- $|D|$: the total number of tuples in $D$

\[ coverage(R) = \frac{n_{covers}}{|D|} \quad \text{accuracy}(R) = \frac{n_{correct}}{n_{covers}} \]
How To Use Rules for Classification

- Predict the class label for tuple X
  - If a rule is satisfied by X, the rule is said to be **triggered**
  - If a rule $R$ is the only one satisfied by X, the rule **fires** by returning the class prediction of X

**Important**
- Triggering $\neq$ firing
- More than one rule can be satisfied

**Problems**
- What if no rule is satisfied by X?
  - **Solution**: use a **default rule** that fires, for example, the most frequent class
- If more than one rule is triggered, what if each rule specifies a different class?
Conflicting Rules

\[ X(\text{age=youth}, \text{student=yes}, \text{income=low}) \]

\textbf{R1: IF } \text{age=youth AND student=yes } \text{THEN buys\textunderscore computer=yes}

\textbf{R2: IF } \text{income=low } \text{THEN buys\textunderscore computer=no}

\>

Need a \textbf{conflict resolution strategy}

\>

\textbf{Size ordering approach}

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>

\>
4.4.2 Rule Extraction from a Decision Tree

- One rule is created for each path from the root to a leave node.
- Each splitting criterion along a given path is logically ANDed to form the rule antecedent (IF part).
- The leaf node holds the class prediction (the rule consequent).

**Example:**

- **R1:** IF age=youth AND student=no THEN buys_computer=no
- **R2:** IF age=youth AND student=yes THEN buys_computer=yes
- **R3:** IF age=middle-aged THEN buys_computer=yes
- **R4:** IF age=senior AND credit_rating=excellent THEN buys_computer=yes
- **R5:** IF age=senior AND credit_rating=fair THEN buys_computer=no
Characteristics of Decision Tree Rules

Decision tree rules are mutually **exclusive** and **exhaustive**

- **Exclusive**
  - No rule conflict, no two rules triggered for the same tuple
  - One rule per leaf and any tuple is mapped to only one leaf

- **Exhaustive**
  - One rule for each attribute-value combination
  - The set of rules does not require a default rule

**Note**: The order of rules does not matter when extracted from a decision tree

- **Pruning rules**
  - Any rule that does not improve accuracy can be pruned
  - Pruning may generates non-Mutually exclusive and non-exhaustive rules:
  - C4.5 uses class-based ordering
4.4.3 Sequential Covering Algorithm

- IF-THEN rules are **directly** extracted from training data
- Rules are learned sequentially (one at a time)

  → **Note**: In decision trees rules are learned simultaneously

- Each rule for a given class ideally covers many tuples of that class and hopefully no tuples from other classes

- When a rule is learned, the tuples covered by the rule are removed (need of accurate rules but not necessarily high coverage)

- The process repeats on the remaining tuples until a stopping condition:
  → No tuples left
  → The quality measure of a rule is below a threshold
How are Rules Learned?

- In a **general-to-specific** manner

**Example**
- In loan-application data, customers have (age, income, education level, residence, credit-rating, and term of the loan)
- **Two classes:** loan_decision=accept and loan_decision=reject

Start with a general rule for class accept:

```
IF
THEN   loan_decision=accept
```

- Consider each possible attribute test that may be added to the rule
- Adopt a greedy depth-first strategy choosing the rule with high quality (use beam search where the k best attributes are maintained)
- Repeat the process till the rule reached an acceptable quality level

```
IF  income=high AND credit_rating=excellent
    THEN  loan_decision=accept
```
Accuracy seems to be natural as a quality measure, but
- R1: correctly classifies 18 tuples out of 20 (accuracy=90%)
- R2: correctly classifies 2 tuples out of 2 (accuracy=100%)

→ Accuracy alone is not enough
→ Coverage alone is not enough (cover many tuples of ≠ classes )
→ Use Entropy
Rule Quality Measures

- Using **Entropy** (Information Gain)

  \[ R: \text{IF condition THEN class}=c \]

- If logically ANDing a given attribute test to **condition** we obtain **condition’**

  \[ R’: \text{IF condition’ THEN class}=c \]

  - Test the potential rule R’ using entropy
  - Compute the **entropy** based on probabilities \( p_i \), where \( p_i \) is the probability of a class \( C_i \) in \( D \)
  - \( D \) is the set of tuples covered by R’
  - Entropy prefers conditions that cover a large number of tuples of a single class and few tuples of other classes
Summary of Section 4.4

- Rule-based classification builds a model that is a set of rules.

- Rules can be extracted from a decision tree or directly from training data.

- Rule quality measures are important to assess the rules and to define orders for conflict resolution.
Chapter 4: Classification & Prediction

- 4.1 Basic Concepts of Classification and Prediction
- 4.2 Decision Tree Induction
- 4.3 Bayes Classification Methods
- 4.4 Rule Based Classification
- 4.5 Lazy Learners
  - 4.5.1 K-Nearest-Neighbor Classifiers
  - 4.5.2 Shortcomings of K-NN Algorithms
- 4.6 Prediction
- 4.7 How to Evaluate and Improve Classification
4.5 Lazy Learners

- The classification algorithms presented before are **eager learners**
  - Construct a generalization model before receiving new tuples to classify
  - Learned models are ready and eager to classify previously unseen tuples

- **Lazy learners**
  - The learner waits till the last minute
  - Before doing any model construction
  - In order to classify a given test tuple
  - Store training tuples
  - Wait for test tuples
  - Perform generalization based on similarity between test and the stored training tuples

<table>
<thead>
<tr>
<th>Eager Learners</th>
<th>Lazy learners</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do lot of work on training data</td>
<td>Do less work on training data</td>
</tr>
<tr>
<td>Do less work when test tuples are presented</td>
<td>Do more work when test tuples are presented</td>
</tr>
</tbody>
</table>
4.5.1 k-Nearest Neighbor Classifiers

- Nearest-neighbor classifiers compare a given test tuple with training tuples that are similar
  - Training tuples are described by \( n \) attributes
  - Training tuples are stored in \( n \)-dimensional space
  - Find the \( k \)-nearest tuples from the training set to the unknown tuple

- The closeness between tuples is defined in terms of distance metric
  - \( X_1(x_{11}, \ldots, x_{1n}) \)
  - \( X_2(x_{21}, \ldots, x_{2n}) \)
  - E.g., Euclidian distance

\[
dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}
\]
4.5.1 k-Nearest Neighbor Classifiers

Classification

→ The unknown tuple is assigned the most common class among its $k$ nearest neighbor

→ When $k=1$ the unknown tuple is assigned the class of the training tuple that is closest to it

→ 1-NN scheme has a miss-classification probability that is no worse than twice that of the situation where we know the precise probability density of each function

Prediction

→ Nearest neighbor classifiers can also be used for prediction

→ Return a real-valued prediction for a given unknown tuple

→ The classifier returns the average value of the real-valued labels associated with the $k$-nearest neighbors of the unknown tuple
How to Determine the Value of K

- In typical applications, $k$ is in **units or tens** rather than in hundreds or thousands.

- **Higher values** of $k$ provide smoothing that **reduces** the risk of **overfitting** due to noise in the training data.

- Value of $k$ can be chosen based on error rate measures.

- We should avoid over-smoothing by choosing $k=n$, where $n$ is the total number of tuples in the training data set.

- Let’s see how to choose $k$ via an example.
<table>
<thead>
<tr>
<th>RID</th>
<th>Income($000's)</th>
<th>lot Size (000's sq.ft)</th>
<th>class: Owners =1</th>
<th>Non-Owners=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
<td>18.4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>85.5</td>
<td>16.8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>64.8</td>
<td>21.6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>61.5</td>
<td>20.8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>87</td>
<td>23.6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>110.1</td>
<td>19.2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>17.6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>82.8</td>
<td>22.4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>69</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>93</td>
<td>20.8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>51</td>
<td>22</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>81</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>75</td>
<td>19.6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>52.8</td>
<td>20.8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>64.8</td>
<td>17.2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>43.2</td>
<td>20.4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>84</td>
<td>17.6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>49.2</td>
<td>17.6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>59.4</td>
<td>16</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>66</td>
<td>18.4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>47.4</td>
<td>16.4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>33</td>
<td>18.8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>51</td>
<td>14</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>63</td>
<td>14.8</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Example

We randomly divide the data into 18 training cases and 6 test cases: tuples 6,7,12,14,19, 20. Use training cases to classify test cases and compute error rates.
If we choose \( k=1 \) we will classify in a way that is very sensitive to the local characteristics of our data.

If we choose a large value of \( k \) we average over a large number of data points and average out the variability due to the noise associated with data points.

If we choose \( k=18 \) we would simply predict the most frequent class in the data set in all cases.

- Very stable but completely ignores the information in the independent variables.

We would choose \( k=11 \) (or possibly 13) in this case.

<table>
<thead>
<tr>
<th>( k )</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misclassification error %</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>17</td>
<td>17</td>
<td>50</td>
</tr>
</tbody>
</table>
4.5.2 Shortcomings of k-NN Algorithms

- **First**: no time requires to estimate parameters from training data, but the time to find the nearest neighbor can be prohibitive

- **Some ideas to overcome this problem**

  - Reduce the time taken to compute distances by working in reduced dimension (use PCA)
  
  - Use sophisticated data structure such as trees to speed up the identification of the nearest neighbor
  
  - Edit the training data to remove redundant or almost redundant points.
    - E.g., remove observations in the training data that have no effect on the classification because they are surrounded by observations that all belong to the same class
4.5.2 Shortcomings of k-NN Algorithms

- **Second:** “the Curse of Dimensionality”
  - Let $p$ be the number of dimensions
  - The expected distance to the nearest neighbor goes up dramatically with $p$ unless the size of the training data set increases exponentially with $p$

- **Some ideas to overcome this problem**
  - Reduce the dimensionality of the space of attributes
  - Select subsets of the predictor variables by combining them using methods such as principal components, singular value decomposition and factor analysis
Chapter 4: Classification & Prediction

4.1 Basic Concepts of Classification and Prediction
4.2 Decision Tree Induction
4.3 Bayes Classification Methods
4.4 Rule Based Classification
4.5 Lazy Learners
4.6 Prediction
  4.6.1 Definitions
  4.6.2 Linear Regression
  4.6.3 Nonlinear Regression
  4.6.4 Generalized Linear Models: Logistic Regression
4.7 How to Evaluate and Improve Classification
### 4.6.1 Definitions

- **Numeric Prediction** (or prediction) is the task of predicting continuous (or ordered) values for given input.

- **Examples**
  - Given the profile of a customer, predict how much money he will spend.
  - Predict the potential sale of a new product given its price.

- The most widely used approach for prediction is **regression**.

- **Regression Analysis**
  - A statistical methodology.
  - Used to model the relationship between one or more independent (predictor) variable and a dependent (response) variable.
    - **Predictor variables**: the attributes describing a tuple.
    - **Response variable**: what we want to predict.

- Many prediction problems can be solved using **linear regression**.

- A **non-linear** problem can be converted to a linear one.
4.6.2 Linear Regression

- **Straight-line regression** analysis involves
  - A single predictor variable
  - A response variable

\[ y = b + wx \]

- The **variance** of \( y \) is **constant**
- \( b \) and \( w \) are **regression coefficients**
  - \( b \): Y-intercept
  - \( w \): the slope of the line

- Regression coefficients can also be considered as weights

\[ y = \beta_0 + \beta_1 x \]

- Need of estimating the regression coefficients
Method of Least Squares

- Estimate the best-fitting straight line as the one that minimizes the error between the actual data and the estimate of the line.
- Used to solve overdetermined systems (more equations than unknowns).
- \( f \) is the model function where
  \[
  y_i = f(x, \beta) = \beta_0 + \beta_1 x
  \]
- Minimize the sum, \( S \), of squared residuals
  \[
  S = \sum_{i=1}^{|D|} r_i^2 \quad r_i = y_i - f(x_i, \hat{\beta})
  \]
- \( D \): a set of training tuples with 1 predictor and 1 response each
  - \((x_1, y_1)\)
  - \((x_2, y_2)\)
  - ...
  - \((x_{|D|}, y_{|D|})\)
Method of Least Squares

- The minimum of the sum of squares is found by setting the gradient to zero. If the model contains \( m \) parameters there are \( m \) gradient equations:

\[
\frac{\partial S}{\partial \beta_j} = 2 \sum_i \frac{\partial r_i}{\partial \beta_j} = 0, \ j = 1, \ldots, m
\]

- When \( m=2 \), the regression coefficients are estimated by:

\[
\beta_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2}
\]

\[
\beta_0 = \bar{y} - \beta_1 \bar{x}
\]

where \( \bar{x} \) is the mean value of \( x_1, x_2, \ldots, y_{|D|} \)

\( \bar{y} \) is the mean value of \( y_1, y_2, \ldots, y_{|D|} \)
Example

- Four data points: \((1,6),(2,5),(3,7)\) and \((4,10)\)

- Model these data as \( y = f(x, \beta) = \beta_0 + \beta_1 x \)

- Find the parameters that approximately solve:

\[
\begin{align*}
\beta_0 + 1\beta_1 &= 6 \\
\beta_0 + 2\beta_1 &= 5 \\
\beta_0 + 3\beta_1 &= 7 \\
\beta_0 + 4\beta_1 &= 10
\end{align*}
\]

\[
S = [6 - (\beta_0 + 1\beta_1)]^2 + [5 - (\beta_0 + 2\beta_1)]^2 + [7 - (\beta_0 + 3\beta_1)]^2 + [10 - (\beta_0 + 4\beta_1)]^2
\]

- By determine the partial derivatives \(S\) with respect to \(\beta_1\) and \(\beta_1\) and setting then to zero, we find:

\[
\beta_0 = 3.5\quad\text{and}\quad\beta_1 = 1.4
\]
Multiple Linear Regression

- Involve more than one predictor variables
- Model a response variable as linear function of $n$ predictor variables $A_1, A_2, \ldots, A_n$
- D: a set of training tuples with $n$ predictors and 1 response each
  - $(X_1, y_1)$
  - $(X_2, y_2)$
  - ...
  - $(X_{|D|}, y_{|D|})$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

- The method of least square is used to estimate the coefficients. However the computation becomes long
  - Use statistical software packages (e.g., SAS, SPSS, and S-Plus)
How to model data that does not show a linear dependence?

Example: **polynomial regression**
- Add polynomial terms to the basic linear model
- Apply transformations to variables
- Convert the nonlinear model to a linear one

Consider a cubic polynomial relationship given by:

\[ y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \]

To convert this equation to linear form, we define new variables

\[ x_1 = x \quad x_2 = x^2 \quad x_3 = x^3 \]

The equation becomes

\[ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \]
4.6.4 Generalized Linear Models

- Represent the theoretical foundation on which the linear regression can be applied to model classification.

- The variance of the response variable, is a function of the mean value of $y$, unlike the linear regression where the variance of $y$ is constant.

- Common types of generalized linear models include:
  - Poisson regression
  - Logistic regression

- **Logistic regression** models the probability of some event occurring as a linear function of a set of predictor variables.
Logistic Regression

- The logistic regression is used for binomial regression.
- It predicts the probability of occurrence of an event by fitting data to a logistic curve.
- $x$ represents the exposure to some set of risk factors.
- $f(x)$ represents the probability of a particular outcome, given that set of risk factors.

$$f(x) = \frac{e^x}{1 + e^x}$$
The variable \( x \) is a measure of the total contribution of all the risk factors (independent variables) used in the model and is known as the logit.

\[ x = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k \]

The logistic regression model is given by

\[
P(Y = 1 \mid x_1, x_2, \ldots, x_k) = \frac{e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k}}{1 + e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k}}
\]
Logistic Regression

\[
P(Y = 1 \mid x_1, x_2, \ldots x_k) = \frac{e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k}}{1 + e^{\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k}}
\]

- Estimate parameters using Maximum Likelihood Estimator
  - **Data:** \(y_j, x_{1j}, x_{2j}, \ldots, x_{pj}, \) \(j = 1, 2, \ldots, n\)
  - Likelihood Function is given by:
    \[
    L(\beta) = \prod_{j=1}^{n} \frac{e^{\beta_0 + \beta_1 x_{1j} + \ldots + \beta_p x_{pj}}}{1 + e^{\beta_0 + \beta_1 x_{1j} + \ldots + \beta_p x_{pj}}}
    \]
  - To simplify the computation, we can maximize the log likelihood function
- To estimate the parameters
  - Compute the partial derivatives of the loglikelihood
  - Equate each partial derivative to zero, and solve the resulting nonlinear equations
Summary of Section 4.6

- Numeric Prediction is the task of predicting *continuous values*

- **Regression analysis** is mostly used for prediction

- Regression can be of different forms *Linear* and *nonlinear*

- **Logistic regression** is used to model *binomial* regression
Chapter 4: Classification & Prediction

- 4.1 Basic Concepts of Classification and Prediction
- 4.2 Decision Tree Induction
- 4.3 Bayes Classification Methods
- 4.4 Rule Based Classification
- 4.5 Lazy Learners
- 4.6 Prediction

4.7 How to Evaluate and Improve Classification
  4.7.1 Accuracy and Error Measures
  4.7.2 Evaluating a Classifier or Predictor
  4.7.3 Increasing the Accuracy
Using training data to build and test a classifier can result in misleading overoptimistic estimates. Accuracy is better measured using test data that was not used to build the classifier.

- **Accuracy**: \([\text{Acc}(M)]\)- accuracy of model \(M\)
  - The percentage of test set tuples that are correctly classified
  - Referred to as the overall recognition rate of the classifier
  - Error rate or misclassification rate: \(1 - \text{Acc}(M)\)
  - When training data are used, the error rate is called resubstitution error
The confusion matrix as a table of at least \( m \) by \( m \) size. An entry \( \text{CM}_{i,j} \) indicated the number of tuples of class \( i \) that were labeled as class \( j \)

### Classifier Accuracy Measures

<table>
<thead>
<tr>
<th>Real class \ Predicted class</th>
<th>Class(_1)</th>
<th>Class(_2)</th>
<th>...</th>
<th>Class(_m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class(_1)</td>
<td>( \text{CM}_{1,1} )</td>
<td>( \text{CM}_{1,2} )</td>
<td>...</td>
<td>( \text{CM}_{1,m} )</td>
</tr>
<tr>
<td>Class(_2)</td>
<td>( \text{CM}_{2,1} )</td>
<td>( \text{CM}_{2,2} )</td>
<td>...</td>
<td>( \text{CM}_{2,m} )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Class(_m)</td>
<td>( \text{CM}_{m,1} )</td>
<td>( \text{CM}_{m,2} )</td>
<td>...</td>
<td>( \text{CM}_{m,m} )</td>
</tr>
</tbody>
</table>

Ideally, most of the tuples would be represented along the diagonal of the confusion matrix.
Case of binary classification

- **Positive tuples**: tuples of the main class of interest (e.g., C1)
- **Negative tuples**: tuples of the other class (e.g., C2)

<table>
<thead>
<tr>
<th>Real class \ Predicted class</th>
<th>C₁</th>
<th>C₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>True positive</td>
<td>False negative</td>
</tr>
<tr>
<td>C₂</td>
<td>False positive</td>
<td>True negative</td>
</tr>
</tbody>
</table>

- **True positives**: positive tuples correctly labeled
- **True negatives**: negative tuples correctly labeled
- **False positives**: negative tuples incorrectly labeled
- **False negatives**: positive tuples incorrectly labeled
Other measures can be used when the accuracy measure is not acceptable:

- **Sensitivity**
  \[ sens = \frac{t\_pos}{pos} \]

- **Specificity**
  \[ spec = \frac{t\_neg}{neg} \]

- **Precision**
  \[ precision = \frac{t\_pos}{(t\_pos + f\_pos)} \]

- **Accuracy**
  \[ accuracy = sens \frac{pos}{(pos + neg)} + spec \frac{neg}{(pos + neg)} \]

- **t\_pos**: the number of true positives
- **t\_neg**: the number of true negatives
- **Neg**: number of positive tuples
- **Pos**: number of positive tuples
- **F\_pos**: number of false positives
The predictor returns continuous values

→ It is difficult to say whether the predicted value is correct or not
→ Measure how far the predicted value from the known value

Compute **loss functions**

\[
\text{Absolute error} = |y_i - y_i'| \\
\text{Squared error} = (y_i - y_i')^2
\]

→ Mean square error is more sensitive to outliers

The **test error** or **generalization error** is the average loss

\[
\text{Mean absolute error} = \frac{\sum_{i=1}^{D} |y_i - y_i'|}{|D|} \\
\text{Mean squared error} = \frac{\sum_{i=1}^{D} (y_i - y_i')^2}{|D|}
\]
The total loss can be normalized by dividing by the total loss incurred from always predicting the mean.

\[
Relative \ absolute \ error = \frac{\sum_{i=1}^{D} |y_i - y_i'|}{\sum_{i=1}^{D} |y_i - y'|}
\]

\[
Relative \ squared \ error = \frac{\sum_{i=1}^{D} (y_i - y_i')^2}{\sum_{i=1}^{D} (y_i - y)^2}
\]

In practice, the choice of error measure does not greatly affect prediction model selection.
4.7.2 Evaluating a Classifier or Predictor

- How can we use the measures described previously to obtain a reliable estimate of classifier accuracy (or predictor accuracy in terms of error)?

- Some common techniques used for this purpose are
  - Holdout Method and Random Subsampling
  - Cross-validation
  - Bootstrap

- They assess accuracy based on randomly sampled partitions of the given data
- These techniques increase the overall computation time
Holdout and Random Subsampling

- **Holdout**
  - Randomly partition the data into two independent sets: training set and test set.
  - Typically, two-thirds of the data are allocated to training set and one-third is allocated to test set.
  - The estimate is **pessimistic** because only a portion of the initial data is used to derive the model.

- **Random Subsampling**
  - The holdout method is repeated $k$ times.
  - The overall accuracy is taken as the **average** of the accuracies obtained from each iteration.
Cross-validation

- Partition the data into $k$ mutually exclusive subsets or "folds", $D_1, D_2, \ldots, D_k$

- Training and testing is performed $k$ times
  - First iteration: use $D_2, \ldots, D_k$ and training and $D_1$ as test
  - Second iteration: use $D_1, D_3, \ldots, D_k$ as training and $D_2$ as test
  - ...

- Each sample is used the same number of times for training and once for testing
Cross-validation

- **Leave-one-out**
  - A **special case** of k-fold cross-validation
  - K is set to the initial number of tuples
  - Only **one** sample is **left out** at a time for the test set

- **Stratified cross-validation**
  - The class distribution of the tuples in each fold is approximately the same as in the initial data

- In general, stratified 10-fold cross validation is recommended for estimating accuracy due to its relatively low bias and variance
Sample training tuples uniformly with replacement

- i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set

Several bootstrap methods, and a common one is \( .632 \) bootstrap

- Suppose we are given a data set of \( d \) tuples
- The data set is sampled \( d \) times with replacement
- Result: a training set of \( d \) samples
- About \( 63.2\% \) of the original data will end up in the bootstrap, and the remaining \( 36.8\% \) will form the test set (since \( (1 - 1/d)^d \approx e^{-1} = 0.368 \))

Repeat the sampling procedure \( k \) times, overall accuracy of the model:

\[
acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})
\]
4.7.3 Increasing the Accuracy

- We have seen that pruning improves the accuracy of decision trees by reducing the overfitting effect.

- There are some general strategies for improving the accuracy of classifiers and predictors.

  - **Bagging** and **Boosting** are some of these strategies.

  - **Ensemble methods**: use a combination of models.
  - Combine a series of learned classifiers $M_1, M_2, \ldots, M_k$.
  - Find an improved **composite model** $M^*$.
Intuition

Ask diagnosis to one doctor

How accurate is this diagnosis?

diagnosis_1

diagnosis_2

diagnosis_3

Choose the diagnosis that occurs more than any of the others
Bagging

- K iterations
- At each iteration a training set $D_i$ is sampled with replacement
- The combined model $M^*$ returns the most frequent class in case of classification, and the average value in case of prediction

![Diagram of Bagging](image)
Assign different weights to the doctors based on the accuracy of their previous diagnosis.
Weights are assigned to each training tuple

A series of $k$ classifiers is iteratively learned

After a classifier $M_i$ is learned, the weights are adjusted to allow the subsequent classifier to pay more attention to training tuples misclassified by $M_i$

The final boosted classifier $M^*$ combines the votes of each individual classifier where the weight of each classifier is a function of its accuracy

This strategy can be extended for the prediction of continuous values
Given a set of \( d \) class-labeled tuples \((X_1, y_1), \ldots, (X_d, y_d)\)

Initially, all the weights of tuples are the same: \( 1/d \)

Generate \( k \) classifiers in \( k \) rounds.

At round \( i \), tuples from \( D \) are sampled (with replacement) to form a training set \( D_i \) of the same size.

Each tuple’s **chance of being selected** depends on its **weight**

A classification model \( M_i \) is **derived** and **tested** using \( D_i \)

If a tuple is **misclassified**, its **weight increases**, otherwise it decreases (use \( \text{err}(M_i)/(1-\text{err}(M_i)) \))
Error rate \( \text{err}(X_i) \) is the misclassification error of tuple \( X_i \)

Classifier \( M_i \) error rate is the sum of the weights of the misclassified tuples

\[
\text{error}(M_i) = \sum_{j=1}^{d} w_j \times \text{err}(X_j)
\]

- Tuple correctly classified: \( \text{err}(X_i) = 0 \)
- Tuple incorrectly classified: \( \text{err}(X_i) = 1 \)

The weight of classifier \( M_i \)'s vote is

\[
\log \left( \frac{1 - \text{error}(M_i)}{\text{error}(M_i)} \right)
\]
Accuracy is used to assess classifiers

Error measures are used to assess predictors

Stratified 10-fold cross validation is recommended for estimating accuracy

Bagging and boosting are used to improve the accuracy of classifiers and predictors