Data Mining I
Classification, Part 1

Heiko Paulheim
Outline

1. What is Classification?
2. k Nearest Neighbors and Nearest Centroids
3. Naïve Bayes
4. Decision Trees
5. Evaluating Classification
6. The Overfitting Problem
7. Rule Learning
8. Other Classification Approaches
9. Parameter Tuning
A Couple of Questions

- What is this?
- Why do you know?
- How have you come to that knowledge?
Introductory Example

• Learning a new concept, e.g., "Tree"

"tree"  "tree"  "tree"

"not a tree"  "not a tree"  "not a tree"
Introductory Example

• Example: learning a new concept, e.g., "Tree"
  – we look at (positive and negative) examples
  – ...and derive a *model*
    • e.g., "Trees are big, green plants"

• Goal: Classification of new instances

Warning: Models are only approximating examples! Not guaranteed to be correct or complete!
What is Classification?

- **Classic programming:**
  - if more than 10 orders/year and more than $100k spent
  
  ```
  set customer.isPremiumCustomer = true
  ```

- **The prevalent style of programming computers**
  - works well as long as we know the rules
  - e.g.: what makes a customer a premium customer?
What is Classification?

- Sometimes, it's not so easy
- E.g., due to missing knowledge
  - if customer is likely to order a new phone
    send advertisement for new phones
- E.g., due to difficult formalization as an algorithm
  - if customer review is angry
    send apology

Diagram:
- give instructions
- compute results
What is Classification?

• A different paradigm:
  – User provides computer with examples
  – Computer finds model by itself
  – Notion: the computer *learns* from examples (term: *machine learning*)

• Example
  – labeled examples of angry and non-angry customer reviews
  – computer finds model for telling if a customer is angry
Classification: Formal Definition

• Given:
  – a set of labeled records, consisting of
    • data fields (a.k.a. attributes or features)
    • a class label (e.g., true/false)

• Generate
  – a function f(r)
    • input: a record
    • output: a class label
  – which can be used for classifying previously unseen records

• Variants:
  – single class problems (e.g., only true/false)
  – multi class problems
  – multi label problems (more than one class per record, not covered in this lecture)
  – hierarchical multi class/label problems (with class hierarchy, e.g., product categories)
What is Classification?

- Classification is a **supervised** learning problem
  - i.e., given labeled data, learn a prediction function for those labels

http://dilbert.com/strip/2013-02-02
The Classification Workflow

### 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>

### Unseen Records

<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>

### Workflow Diagram

1. **Learning Algorithm**
2. **Induction**
3. **Learn Model**
4. **Apply Model**
5. **Deduction**
6. **Model**
Classification Applications – Examples

• Attributes: a set of symptoms (headache, sore throat...)  
  – class: does the patient suffer from disease X?

• Attributes: the values in your tax declaration  
  – class: are you trying to cheat?

• Attributes: your age, income, debts, …  
  – class: are you getting credit by your bank?

• Attributes: the countries you phoned with in the last 6 months  
  – class: are you a terrorist?
Classification Applications – Examples

- Attributes: words in a product review
  - Class: Is it a fake review?

- Attributes: words and header fields of an e-mail
  - Class: Is it a spam e-mail?
Classification Applications – Examples

• A controversial example
  – Class: whether you are searched by the police
  – Class: whether you are selected at the airport for an extra check

http://lubbockonline.com/stories/030609/loc_405504016.shtml
Classification Algorithms

• Recap:
  – we give the computer a set of labeled examples
  – the computer learns to classify new (unlabeled) examples

• How does that work?
k Nearest Neighbors

• Problem
  – find out what the weather is in a certain place
  – where there is no weather station
  – how could you do that?
k Nearest Neighbors

• Idea: use the average of the nearest stations

• Example:
  – 3x sunny
  – 2x cloudy
  – result: sunny

• Approach is called
  – “k nearest neighbors”
  – where k is the number of neighbors to consider
  – in the example: k=5
  – in the example: “near” denotes geographical proximity
k Nearest Neighbors

• Further examples:
• Is a customer going to buy a product?
  → have similar customers bought that product?
• What party are you going to vote for?
  → what party do your (closest) friends/family members vote for?
• Is a film going to win an oscar?
  → have similar films won an oscar?

• and so on...
Experiment

• Trying to predict: do you want to watch “Ad Astra” (coming to cinemas tomorrow)?

• Binary attributes: have you watched these 2019 films?
  1) Replicas
  2) Lego Movie 2
  3) Captain Marvel
  4) The Kid
  5) Shazam!
  6) Long Shot
  7) Dark Phoenix
  8) Secret Life of Pets 2
  9) Angry Birds Movie 2
Recap: Similarity and Distance

• k Nearest Neighbors
  – requires a notion of similarity (i.e., what is “near”?)

• Review: similarity measures for clustering
  – similarity of individual data values
  – similarity of data points

• Think about scales and normalization!

• Which similarity measure was used in our experiment?
  – we could have used different ones
  – probably with different outcomes
Nearest-Neighbor Classifiers

- Requires three things
  - The set of stored records
  - A distance metric to compute distance between records
  - The value of $k$, the number of nearest neighbors to retrieve
Nearest-Neighbor Classifiers

• To classify an unknown record:
  – Compute distance to each training record
  – Identify k nearest neighbors
  – Use class labels of nearest neighbors to determine the class label of unknown record
    • by taking majority vote
    • by weighing the vote according to distance
Definition of the k Nearest Neighbors

The k nearest neighbors of a record \( x \) are data points that have the k smallest distance to \( x \).

(a) 1-nearest neighbor
(b) 2-nearest neighbor
(c) 3-nearest neighbor
Choosing a Good Value for k

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes

- Rule of thumb: Test k values between 1 and 10.
Discussion of K-Nearest Neighbor

- Often very accurate
- ... but slow as training data needs to be searched
- Can handle decision boundaries which are not parallel to the axes
- Assumes all attributes are equally important
  - Remedy: Attribute selection or using attribute weights
Decision Boundaries of a k-NN Classifier

- $k=1$
- Single noise points have influence on model
Decision Boundaries of a k-NN Classifier

- $k=3$
- Boundaries become smoother
- Influence of noise points is reduced
scaler = MinMaxScaler()
features_norm = scaler.fit_transform(features)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(features_norm,label)
Applying the Model

test_norm = scaler.transform(test)
model.predict(test_norm)
Contrast: Nearest Centroids

- a.k.a. Rocchio classifier
- Training: compute centroid for each class
  - center of all points of that class
  - like: centroid for a cluster
- Classification:
  - assign each data point to nearest centroid

- RapidMiner:
  - available in Mannheim RapidMiner Toolbox Extension
- Python:
  - `scikit_learn.neighbors.NearestCentroid`

Sounds pretty much just like k-NN, huh?
k-NN vs. Nearest Centroid

- Basic problem: two circles
  - Both k-NN and Nearest Centroid are rather perfect
k-NN vs. Nearest Centroid

- Some data points are mislabeled
  - k-NN loses performance
  - Nearest Centroid is stable
k-NN vs. Nearest Centroid

- One class is significantly smaller than the other
  - k-NN loses performance
  - Nearest Centroid is stable
**k-NN vs. Nearest Centroid**

- Outliers are contained in the dataset
  - k-NN is stable
  - Nearest Centroid loses performance

![Graph depicting the comparison between k-NN and Nearest Centroid](image-url)
k-NN vs. Nearest Centroid

- **k-NN**
  - slow at classification time (linear in number of data points)
  - requires much memory (storing all data points)
  - robust to outliers

- **Nearest Centroid**
  - fast at classification time (linear in number of classes)
  - requires only little memory (storing only the centroids)
  - robust to label noise
  - robust to class imbalance

- Which classifier is better?
  - that strongly depends on the problem at hand!
Bayes Classifier

- Based on Bayes Theorem
- Thomas Bayes (1701-1761)
  - British mathematician and priest
  - tried to formally prove the existence of God
- Bayes Theorem
  - important theorem in probability theory
  - was only published after Bayes' death
Conditional Probability and Bayes Theorem

• A probabilistic framework for solving classification problems

• Conditional Probability:

\[ P(C|A) = \frac{P(A,C)}{P(A)} \]

\[ P(A|C) = \frac{P(A,C)}{P(C)} \]

• Bayes theorem:

\[ P(C|A) = \frac{P(A|C)P(C)}{P(A)} \]
Conditional Probability and Bayes Theorem

• Bayes Theorem
  – Computes one conditional probability $P(C|A)$ out of another $P(A|C)$
  – given that the base probabilities $P(A)$ and $P(C)$ are known

• Useful in situations where $P(C|A)$ is unknown
  – while $P(A|C)$, $P(A)$ and $P(C)$ are known or easy to determine/estimate

• Example:
  – Given a symptom, what's the probability that I have a certain disease?
Example of Bayes Theorem

- **ELISA Test**
  - the most common test for HIV
- **Numbers:**
  - If you're infected, ELISA shows a positive result with $p=99.9\%$
  - If you're not infected, ELISA shows a negative result with $p=99.5\%$

- Assume you have a positive test
  - What's the probability that you're infected with HIV?

- Make a guess!
Example of Bayes Theorem

- We want to know $P(\text{HIV}|\text{pos})$
  - Bayes theorem:
    $$P(\text{HIV}|\text{pos}) = \frac{P(\text{pos}|\text{HIV})P(\text{HIV})}{P(\text{pos})}$$

- We still need $P(\text{pos})$
  - the probability of a positive test
    $$P(\text{pos}) = P(\text{pos}|\text{HIV} \lor \neg\text{HIV})$$
    $$= P(\text{pos}|\text{HIV}) \cdot P(\text{HIV}) + P(\text{pos}|\neg\text{HIV}) \cdot P(\neg\text{HIV})$$

- Putting the pieces together:
  $$P(\text{HIV}|\text{pos}) = \frac{P(\text{pos}|\text{HIV})P(\text{HIV})}{P(\text{pos}|\text{HIV}) \cdot P(\text{HIV}) + P(\text{pos}|\neg\text{HIV}) \cdot P(\neg\text{HIV})}$$

- 0.1% in Germany
Example of Bayes Theorem

• Now: numbers

\[
P(HIV | pos) = \frac{P(pos | HIV) \cdot P(HIV)}{P(pos | HIV) \cdot P(HIV) + P(pos | \neg HIV) \cdot P(\neg HIV)}
\]

\[
= \frac{0.999 \cdot 0.001}{0.999 \cdot 0.001 + 0.005 \cdot 0.999} = 0.167
\]

• That means:
  – at more than 80% probability, you are still healthy, given a positive test!

• Reason:
  – low overall apriori probability of being HIV positive
Example of Bayes' Theorem

http://xkcd.com/1236/
Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given a record with attributes \((A_1, A_2, \ldots, A_n)\)
  - Goal is to predict class \(C\)
  - Specifically, we want to find the value of \(C\) that maximizes \(P(C| A_1, A_2, \ldots, A_n)\)

- Can we estimate \(P(C| A_1, A_2, \ldots, A_n)\) directly from the data?
Bayesian Classifiers

• Approach:
  – compute the probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of $C$ using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  – Choose value of $C$ that maximizes $P(C \mid A_1, A_2, \ldots, A_n)$

  – Equivalent to choosing value of $C$ that maximizes $P(A_1, A_2, \ldots, A_n \mid C) \ P(C)$

• How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?
Naïve Bayes Classifier

• Assume independence among attributes $A_i$ when class is given:
  
  – $P(A_1, A_2, \ldots, A_n | C_j) = P(A_1 | C_j) \cdot P(A_2 | C_j) \cdots P(A_n | C_j)$
  
  – Can estimate $P(A_i | C_j)$ for all $A_i$ and $C_j$
  
  – New point is classified to $C_j$ if $P(C_j) \prod P(A_i | C_j)$ is maximal
How to Estimate Probabilities from Data?

- Class: \( P(C) = \frac{N_c}{N} \)
  - e.g., \( P(\text{No}) = \frac{7}{10}, \quad P(\text{Yes}) = \frac{3}{10} \)

- For discrete attributes:
  \( P(A_i \mid C_k) = \frac{|A_{ik}|}{N_c} \)
  - where \(|A_{ik}|\) is number of instances having attribute \(A_i\) and belongs to class \(C_k\)
  - Examples:
    \[
    P(\text{Status=Married} \mid \text{No}) = \frac{4}{7} \\
    P(\text{Refund=Yes} \mid \text{Yes}) = 0
    \]
How to Estimate Probabilities from Data?

• For continuous attributes:
  – **Discretize** the range into bins
    • one binary attribute per bin
    • violates independence assumption
  – **Two-way split**: (A < v) or (A > v)
    • choose only one of the two splits as new attribute
  – **Probability density estimation**:
    • Assume attribute follows a normal distribution
    • Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    • Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$
How to Estimate Probabilities from Data?

- Normal distribution:
  \[ P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(A_i - \mu_j)^2}{2\sigma^2}} \]
  - One for each \((A_i, c_i)\) pair

- For \((\text{Income}, \text{Class}=\text{No})\):
  - If \(\text{Class}=\text{No}\)
    - sample mean = 110
    - sample variance = 2975

\[ P(\text{Income} = 120 \mid \text{No}) = \frac{1}{\sqrt{2\pi(54.54)}} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072 \]
How to Estimate Probabilities from Data?

• Example visualization:
  – normal distribution
  – mean = 110
  – variance = 2975

• $P(\text{Income}=120|\text{No}) = 0.0072$
Example of Naïve Bayes Classifier

Given a Test Record:

\[ X = (\text{Refund} = \text{No, Married, Income} = 120K) \]

naive Bayes Classifier:

\[
P(\text{Refund}=\text{Yes}|\text{No}) = 3/7
\]
\[
P(\text{Refund}=\text{No}|\text{No}) = 4/7
\]
\[
P(\text{Refund}=\text{Yes}|\text{Yes}) = 0
\]
\[
P(\text{Refund}=\text{No}|\text{Yes}) = 1
\]
\[
P(\text{Marital Status}=\text{Single}|\text{No}) = 2/7
\]
\[
P(\text{Marital Status}=\text{Divorced}|\text{No}) = 1/7
\]
\[
P(\text{Marital Status}=\text{Married}|\text{No}) = 4/7
\]
\[
P(\text{Marital Status}=\text{Single}|\text{Yes}) = 2/7
\]
\[
P(\text{Marital Status}=\text{Divorced}|\text{Yes}) = 1/7
\]
\[
P(\text{Marital Status}=\text{Married}|\text{Yes}) = 0
\]

For taxable income:

If class=No: sample mean=110
sample variance=2975

If class=Yes: sample mean=90
sample variance=25

\[ P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No}) \]
\[ \times P(\text{Married}|\text{Class}=\text{No}) \]
\[ \times P(\text{Income}=120K|\text{Class}=\text{No}) \]
\[ = 4/7 \times 4/7 \times 0.0072 = 0.0024 \]

\[ P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No}|\text{Class}=\text{Yes}) \]
\[ \times P(\text{Married}|\text{Class}=\text{Yes}) \]
\[ \times P(\text{Income}=120K|\text{Class}=\text{Yes}) \]
\[ = 1 \times 0 \times (1.2 \times 10^{-9}) = 0 \]

Since \( P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes}) \)
Therefore \( P(\text{No}|X) > P(\text{Yes}|X) \)
\[ \Rightarrow \text{Class} = \text{No} \]
Handling missing values

- Missing values may occur in training and classification examples.

- **Training:** Instance is not included in frequency count for attribute value-class combination.

- **Classification:** Attribute will be omitted from calculation.

- Example:

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Evade</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>No</td>
<td>?</td>
<td>120k</td>
<td>?</td>
</tr>
</tbody>
</table>

Likelihood of “yes” = $1 \times (1.2 \times 10^{-9}) = 1.2 \times 10^{-9}$

Likelihood of “no” = $4/7 \times 0.0072 = 0.0041$
From Likelihoods to Probabilities

• A person can either evade or not
  – so why do the likelihoods not add up to 1?

• Recap:

\[
P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}
\]

We have ignored the denominator so far!
  – however, it is the same for all classes
  – so we can simply normalize to 1:

Likelihood of “yes” = 1 * (1.2 * 10^{-9}) = 1.2 * 10^{-9}
Likelihood of “no” = 4/7 * 0.0072 = 0.0041
P(“yes”) = 1.2 * 10^{-9} / (1.2 * 10^{-9} + 0.0041) = 0.0000003
P(“no”) = 0.0041 / (1.2 * 10^{-9} + 0.0041) = 0.9999997
Zero Frequency Problem

- If one of the conditional probabilities is zero, then the entire expression becomes zero.
- And it is not unlikely that an exactly same data point has not yet been observed.
- Probability estimation:

\[ P(A_i | C) = \frac{N_{ic}}{N_c} \]

Original: \( N_{ic} \) \( N_c \)

Laplace: \( P(A_i | C) = \frac{N_{ic} + 1}{N_c + c} \)

c: number of classes
Naïve Bayes in RapidMiner & Python

```python
model = GaussianNB()
model.fit(features, label)
```
Anatomy of a Naïve Bayes Model

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Parameter</th>
<th>no</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlook</td>
<td>value=rain</td>
<td>0.392</td>
<td>0.331</td>
</tr>
<tr>
<td>Outlook</td>
<td>value=overcast</td>
<td>0.014</td>
<td></td>
</tr>
<tr>
<td>Outlook</td>
<td>value=sunny</td>
<td>0.581</td>
<td></td>
</tr>
<tr>
<td>Outlook</td>
<td>value=unknown</td>
<td>0.014</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>mean</td>
<td>74.600</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>standard deviation</td>
<td>7.893</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>mean</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>standard deviation</td>
<td>9.618</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=true</td>
<td>0.589</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=false</td>
<td>0.397</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>value=unknown</td>
<td>0.014</td>
<td></td>
</tr>
</tbody>
</table>
Using Conditional Probabilities for Naïve Bayes

<table>
<thead>
<tr>
<th>Row No.</th>
<th>Play</th>
<th>confidence(no)</th>
<th>confidence(yes)</th>
<th>prediction(Play)</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>0.711</td>
<td>0.289</td>
<td>no</td>
<td>sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td>0.058</td>
<td>0.942</td>
<td>yes</td>
<td>overcast</td>
<td>80</td>
<td>90</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>0.014</td>
<td>0.986</td>
<td>yes</td>
<td>overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>0.412</td>
<td>0.588</td>
<td>yes</td>
<td>rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>0.460</td>
<td>0.540</td>
<td>yes</td>
<td>rain</td>
<td>68</td>
<td>80</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>no</td>
<td>0.336</td>
<td>0.664</td>
<td>yes</td>
<td>rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>0.010</td>
<td>0.990</td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>no</td>
<td>0.596</td>
<td>0.404</td>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>0.248</td>
<td>0.752</td>
<td>yes</td>
<td>sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
</tr>
<tr>
<td>10</td>
<td>no</td>
<td>0.407</td>
<td>0.593</td>
<td>yes</td>
<td>sunny</td>
<td>75</td>
<td>80</td>
<td>false</td>
</tr>
<tr>
<td>11</td>
<td>yes</td>
<td>0.496</td>
<td>0.504</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>yes</td>
<td>0.038</td>
<td>0.962</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>no</td>
<td>0.027</td>
<td>0.973</td>
<td>yes</td>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>true</td>
</tr>
<tr>
<td>14</td>
<td>yes</td>
<td>0.453</td>
<td>0.547</td>
<td>yes</td>
<td>rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
</tr>
</tbody>
</table>

- classifier is quite sure
- classifier is not sure
Decision Boundary of Naive Bayes Classifier

- Usually larger coherent areas
- Soft margins with uncertain regions
- Arbitrary (often curved) shapes
Naïve Bayes (Summary)

• Robust to isolated noise points
  – they have a small impact on the probabilities

• Handle missing values by ignoring the instance during probability estimate calculations

• Robust to irrelevant attributes

• Independence assumption may not hold for some attributes
  – Use other techniques such as Bayesian Belief Networks (BBN)
Why Naïve Bayes?

• Recap:
  – we assume that all the attributes are independent

• This does not hold for many real world datasets
  – e.g., persons: sex, weight, height
  – e.g., cars: weight, fuel consumption
  – e.g., countries: population, area, GDP
  – e.g., food: ingredients
  – e.g., text: word occurrences ("Donald", "Trump", "Duck")
  – ...
Naïve Bayes Discussion

- Naïve Bayes works surprisingly well.
  - even if independence assumption is clearly violated
  - Classification doesn’t require accurate probability estimates as long as maximum probability is assigned to correct class

- However: Adding too many redundant attributes will cause problems
  - Solution: Select attribute subset as Naïve Bayes often works as well or better with just a fraction of all attributes.

- Technical advantages:
  - Learning Naïve Bayes classifiers is computationally cheap as probabilities can be estimated doing one pass over the training data
  - Storing the probabilities does not require a lot of memory
Redundant Variables

- Consider two variables which are perfectly correlated
  - i.e., one is redundant
  - e.g.: a measurement in different units
- Violate independence assumption in Naive Bayes
  - Can, at large scale, skew the result
  - Consider, e.g., a price attribute in 20 currencies
    → price variable gets 20 times more influence
- May also skew the distance measures in k-NN
  - But the effect is not as drastic
  - Depends on the distance measure used
Irrelevant Variables

- Consider a random variable $x$, and two classes $A$ and $B$
  - For Naive Bayes: $p(x=v|A) = p(x=v|B)$ for any value $v$
  - Since it is random, it does not depend on the class variable
  - The overall result does not change

- For kNN:
Comparison kNN and Naïve Bayes

- Computation
  - Naïve Bayes is often faster

- Naïve Bayes uses *all* data points
  - Naive Bayes is less sensitive to label noise
  - k-NN is less sensitive to outliers

- *Redundant* attributes
  - are less problematic for kNN

- *Irrelevant* attributes
  - are less problematic for Naïve Bayes
  - attribute values equally distributed across classes
    → same factor for each class

- In both cases
  - attribute pre-selection makes sense (see Data Mining II)
Lazy vs. Eager Learning

- k-NN, and Naïve Bayes are all “lazy” methods
- They do not build an explicit model!
  - “learning” is only performed on demand for unseen records
- Nearest Centroid is a simple “eager” method
Lazy vs. Eager Learning

• We have seen three of the most common techniques for lazy learning
  – k nearest neighbors
  – Naïve Bayes
• ...and a very simple technique for eager learning
  – Nearest Centroids

• We will see more eager learning in the next lectures
  – where explicit models are built
  – e.g., decision trees
  – e.g., rule sets