Density Estimation

- Our Joint Distribution learner is our first example of something called Density Estimation
- A Density Estimator learns a mapping from a set of attributes to a Probability

There are many different kinds of Estimators

Density Estimation

- Compare it against the two other major kinds of models:

  - Input Attributes → Density Estimator → Probability
  - Input Attributes → Classifier → Prediction of categorical output
  - Input Attributes → Regressor → Prediction of real-valued output
Why are we doing this again?

- Density Estimators can do many magical things. They can answer:
  - \( P(\text{chips}|\text{beer}) \)
    - If you just want to know probabilities
  - \( P(\text{canadian}|\text{features}), P(\text{icelander}|\text{features}) \)
    - If you want to know which is more likely, so that you can classify
  - \( P(x_1 \leq \text{life_span} \leq x_2) \)
    - Like regression, but better.

- I.e., they can do all three of our first examples. This is pretty awesome.

---

Evaluating Density Estimation

Test-set criterion for estimating performance on future data*

* See the Decision Tree or Cross Validation lecture for more detail

- **Classifier**
  - Prediction of categorical output
  - **Test set Error Rate** (# errors)

- **Regressor**
  - Prediction of real-valued output
  - **Test set Error Rate** (MSE)

- **Density Estimator**
  - Probability
  - ???
Evaluating a density estimator

• Given one record \( x \), a density estimator \( M \) can tell you how likely the record is:

\[
\hat{P}(x|M)
\]

• Given a dataset with \( R \) records, a density estimator can tell you how likely the dataset is:

(Under the assumption that all records were independently generated from the Density Estimator’s JD)

\[
\hat{P}(\text{dataset}|M) = \hat{P}(x_1 \land x_2 \land \ldots \land x_R|M) = \prod_{k=1}^{R} \hat{P}(x_k|M)
\]

You can think of \( M \) as being similar to \( w \) from regression...

A small dataset: Miles Per Gallon

From the UCI repository (thanks to Ross Quinlan)
A small dataset: Miles Per Gallon

192 Training Set Records

\[ \hat{P}(\text{dataset}|M) = \hat{P}(x_1 \wedge x_2 \ldots \wedge x_R|M) = \prod_{k=1}^{R} \hat{P}(x_k|M) \]

= (in this case) \(3.4 \times 10^{-203}\)
Log Probabilities

Since probabilities of datasets get so small we usually use log probabilities

$$\log \hat{P}(\text{dataset}|M) = \log \prod_{k=1}^{R} \hat{P}(x_k|M) = \sum_{k=1}^{R} \log \hat{P}(x_k|M)$$

A small dataset: Miles Per Gallon

<table>
<thead>
<tr>
<th>mpg</th>
<th>modelyear</th>
<th>maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>79874</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>79874</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>79874</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>79874</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>79874</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>79874</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>79874</td>
<td>asia</td>
</tr>
<tr>
<td>good</td>
<td>79874</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>79874</td>
<td>europe</td>
</tr>
</tbody>
</table>

$$\log \hat{P}(\text{dataset}|M) = \log \prod_{k=1}^{R} \hat{P}(x_k|M) = \sum_{k=1}^{R} \log \hat{P}(x_k|M)$$

= (in this case) = -466.19
Summary: The Good News

- We have a way to learn a Density Estimator from data.
- Density estimators can do many good things...
  - Can sort the records by probability, and thus spot weird records (anomaly detection)
  - Can do inference: $P(E_1|E_2)$
    - Automatic Doctor / Help Desk etc
  - Ingredient for Bayes Classifiers (see later)

Summary: The Bad News

- Density estimation by directly learning the joint is trivial, mindless and dangerous
Using a test set

<table>
<thead>
<tr>
<th>Set Size</th>
<th>Log likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>196</td>
</tr>
<tr>
<td>Test Set</td>
<td>196</td>
</tr>
</tbody>
</table>

An independent test set with 196 cars has a worse log likelihood (actually it’s a billion quintillion quintillion quintillion quintillion times less likely)

Density estimators can overfit. And the full joint density estimator is the overfittiest of them all!

Remember the lookup table classifier with $2^m$ hypotheses?

Overfitting Density Estimators

If this ever happens during training, it means there are certain combinations that we learn are impossible. The probabilities will be set to 0.

$$\log \hat{P}(\text{testset}|M) = \log \prod_{k=1}^{R} \hat{P}(x_k|M) = \sum_{k=1}^{R} \log \hat{P}(x_k|M)$$

= $-\infty$ if for any $k \hat{P}(x_k|M) = 0$
Using a test set

<table>
<thead>
<tr>
<th>Set</th>
<th>Size</th>
<th>Log likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>196</td>
<td>-466.1905</td>
</tr>
<tr>
<td>Test</td>
<td>196</td>
<td>-614.6157</td>
</tr>
</tbody>
</table>

The only reason that our test set didn’t score -infinity is that my code is hard-wired to always predict a probability of at least one in $10^{20}$

*We need Density Estimators that are less prone to overfitting*

Naïve Density Estimation

The problem with the Joint Estimator is that it just mirrors the training data.

We need something which generalizes more usefully.

The naïve model generalizes strongly:

Assume that each attribute is distributed independently of any of the other attributes.
Independently Distributed Data

- Let $x[i]$ denote the $i$th field of record $x$.
- The independently distributed assumption says that for any $i, v, u_1, u_2, ..., u_{i-1}, u_{i+1}, ..., u_M$

$$P(x[i] = v | x[1] = u_1, x[2] = u_2, ..., x[i-1] = u_{i-1}, x[i+1] = u_{i+1}, ..., x[M] = u_M) = P(x[i] = v)$$

"Given everything except $x[i]"$

- Or in other words, $x[i]$ is independent of
  - $\{x[1], x[2], ..., x[i-1], x[i+1], ..., x[M]\}$
  - This is often written as

$$x[i] \perp \{x[1], x[2], ..., x[i-1], x[i+1], ..., x[M]\}$$

A note about independence

- Assume $A$ and $B$ are Boolean Random Variables. Then
  "A and B are independent"

if and only if

$$P(A|B) = P(A)$$

- "A and B are independent" is often notated as

$$A \perp B$$
Independence Theorems

- Assume $P(A|B) = P(A)$
- Then $P(A \cap B) = P(A) P(B)$

- Assume $P(A|B) = P(A)$
- Then $P(B|A) = P(B)$

Independence Theorems

- Assume $P(A|B) = P(A)$
- Then $P(\neg A|B) = P(\neg A)$

- Assume $P(A|B) = P(A)$
- Then $P(A|\neg B) = P(A)$
Multivalued Independence

For multivalued Random Variables A and B,

\[ A \perp B \]

if and only if

\[ \forall u, v : P(A = u \mid B = v) = P(A = u) \]

from which you can then prove things like...

\[ \forall u, v : P(A = u \land B = v) = P(A = u)P(B = v) \]

\[ \forall u, v : P(B = u \mid A = v) = P(B = v) \]

Back to Naïve Density Estimation

- Let x[i] denote the i'th field of record x:
- Naïve DE assumes x[i] is independent of \{x[1], x[2], ... x[i-1], x[i+1], ... x[M]\}
- Example:
  - Suppose that each record is generated by randomly shaking a green dice and a red dice
  - Dataset 1: A = red value, B = green value
  - Dataset 2: A = red value, B = sum of values
  - Dataset 3: A = sum of values, B = difference of values
- Which of these datasets violates the naïve assumption?
- Think: What do I know about A? What do I know about A if I know B? That is: What is \( P(A) \)? What is \( P(A|B) \)? What about \( P(A|B=5) \)?
Using the Naïve Distribution

- Once you have a Naïve Distribution you can easily compute any row of the joint distribution.
- Suppose $A$, $B$, $C$ and $D$ are independently distributed. What is $P(A^\sim B^\sim C^\sim D)$?

\[
P(A^\sim B^\sim C^\sim D) = P(A) P(\sim B) P(C) P(\sim D)
\]
Naïve Distribution General Case

- Suppose $x[1], x[2], ... x[M]$ are independently distributed.

$$P(x[1] = u_1, x[2] = u_2, ... x[M] = u_M) = \prod_{k=1}^{M} P(x[k] = u_k)$$

- So if we have a Naïve Distribution we can construct any row of the implied Joint Distribution on demand.

- So we can do any inference

- But how do we learn a Naïve Density Estimator?

Learning a Naïve Density Estimator

$$\hat{P}(x[i] = u) = \frac{\# \text{records in which } x[i] = u}{\text{total number of records}}$$

Another trivial learning algorithm!
### Contrast

<table>
<thead>
<tr>
<th>Joint DE</th>
<th>Naïve DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can model anything</td>
<td>Can model only very boring distributions</td>
</tr>
<tr>
<td>No problem to model</td>
<td>Outside Naïve’s scope</td>
</tr>
<tr>
<td>“C is a noisy copy of A”</td>
<td></td>
</tr>
<tr>
<td>Given 100 records and more than 6 Boolean</td>
<td>Given 100 records and 10,000 multivalued</td>
</tr>
<tr>
<td>attributes will screw up badly</td>
<td>attributes will be fine</td>
</tr>
</tbody>
</table>

### Empirical Results: “Hopeless”

The "hopeless" dataset consists of 40,000 records and 21 Boolean attributes called a, b, c, ... u. Each attribute in each record is generated 50-50 randomly as 0 or 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>Parameters</th>
<th>LogLik</th>
<th>+/-  StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model1</td>
<td>joint</td>
<td>submodel=gauss</td>
<td>-272625</td>
<td>301.109</td>
</tr>
<tr>
<td></td>
<td></td>
<td>guasstype=general</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model2</td>
<td>naive</td>
<td>submodel=gauss</td>
<td>-58225.6</td>
<td>0.554747</td>
</tr>
<tr>
<td></td>
<td></td>
<td>guasstype=general</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Despite the vast amount of data, "Joint" overfits hopelessly and does much worse
Empirical Results: “Logical”

The “logical” dataset consists of 40,000 records and 4 Boolean attributes called a, b, c, d where a, b, c are generated 50-50 randomly as 0 or 1. D = A^~C, except that in 10% of records it is flipped.

### The DE learned by “Joint”

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.11335</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.01202</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.11152</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.01295</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.01222</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.01233</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.11358</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.01232</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.01323</td>
</tr>
</tbody>
</table>

### The DE learned by “Naive”

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.500325</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.499675</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.50046</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.49995</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.50165</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.49935</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.69945</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.30055</td>
</tr>
</tbody>
</table>
Empirical Results: “MPG”

The “MPG” dataset consists of 392 records and 8 attributes

A tiny part of the DE learned by “Joint”

The DE learned by “Naive”

Empirical Results: “MPG”

The “MPG” dataset consists of 392 records and 8 attributes

A tiny part of the DE

The DE learned by “Naive”
Empirical Results: “Weight vs. MPG”
Suppose we train only from the "Weight" and "MPG" attributes

The DE learned by "Joint"

The DE learned by "Naive"

---

Empirical Results: “Weight vs. MPG”
Suppose we train only from the "Weight" and "MPG" attributes

<table>
<thead>
<tr>
<th>mpg</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>bad</td>
<td>low 0.193878</td>
</tr>
<tr>
<td>good</td>
<td>low 0.380102</td>
</tr>
</tbody>
</table>

| mpg | bad 0.602041 | good 0.397959 |
|-----|--------------|
| weight | low 0.57398 | high 0.42602 |

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>Parameters</th>
<th>LogLike</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model1</td>
<td>joint</td>
<td>submodel=gauss, gausstype=general</td>
<td>-44.3562 ± 2.27647</td>
</tr>
<tr>
<td>Model2</td>
<td>naive</td>
<td>submodel=gauss, gausstype=general</td>
<td>-53.2231 ± 0.610411</td>
</tr>
</tbody>
</table>
"Weight vs. MPG": The best that Naïve can do

The light color shades denote predicted densities. The dark shades are real data.

Reminder: The Good News

- We have two ways to learn a Density Estimator from data.
- * In other lectures we’ll see vastly more impressive Density Estimators (Mixture Models, Bayesian Networks, Density Trees, Kernel Densities and many more)
- Density estimators can do many good things...
  - Anomaly detection
  - Can do inference: P(E1|E2) Automatic Doctor / Help Desk etc
  - Ingredient for Bayes Classifiers
Bayes Classifiers

- A formidable and sworn enemy of decision trees

---

How to build a Bayes Classifier

- Assume you want to predict output $Y$ which has arity $n_Y$ and values $v_{y_1}, v_{y_2}, \ldots, v_{y_{n_Y}}$
- Assume there are $m$ input attributes called $X_1, X_2, \ldots, X_m$
- Break dataset into $n_Y$ smaller datasets called $DS_{y_1}, DS_{y_2}, \ldots, DS_{y_{n_Y}}$
- Define $DS_i = \text{Records in which } Y = v_i$
- For each $DS_i$, learn Density Estimator $M_i$ to model the input distribution among the $Y=v_i$ records.
How to build a Bayes Classifier

• Assume you want to predict output Y which has arity \( n_Y \) and values \( v_1, v_2, \ldots, v_{n_Y} \).
• Assume there are \( m \) input attributes called \( X_1, X_2, \ldots, X_m \).
• Break dataset into \( n_Y \) smaller datasets called \( DS_1, DS_2, \ldots, DS_{n_Y} \).
• Define \( DS_i \) = Records in which \( Y = v_i \).
• For each \( DS_i \), learn Density Estimator \( M_i \) to model the input distribution among the \( Y = v_i \) records.
• \( M_i \) estimates \( P(X_1, X_2, \ldots, X_m \mid Y = v_i) \).

Idea: When a new set of input values \((X_1 = u_1, X_2 = u_2, \ldots, X_m = u_m)\) come along to be evaluated predict the value of \( Y \) that makes \( P(X_1, X_2, \ldots, X_m \mid Y = v_i) \) most likely

\[
Y_{\text{predict}} = \arg\max_v P(X_1 = u_1 \cdot X_m = u_m \mid Y = v)
\]

Is this a good idea?
How to build a Bayes Classifier

- Assume you want to predict output $Y$ which has arity $n_Y$ and values $v_{Y1}, v_{Y2}, \ldots, v_{Yn_Y}$.
- Assume there are $m$ input attributes called $X_1, X_2, \ldots, X_m$.
- Break dataset into $n_Y$ smaller datasets called $DS_1, DS_2, \ldots, DS_{n_Y}$.
- Define $DS_i$ = Records in which $Y = v_i$.
- For each $DS_i$, learn Density Estimator $M_i$ to model the input distribution among the $Y = v_i$ records.
- $M_i$ estimates $P(X_1, X_2, \ldots, X_m | Y = v_i)$.

Idea: When a new set of input values $(X_1 = u_1, X_2 = u_2, \ldots, X_m = u_m)$ come along to be evaluated, predict the value of $Y$ that makes $P(Y = v_i | X_1, X_2, \ldots, X_m)$ most likely.

$$Y_{predict} = \arg\max_v P(X_1 = u_1 \cdot \cdot \cdot X_m = u_m | Y = v_i)$$

Is this a good idea?

This is a Maximum Likelihood classifier.

It can get silly if some $Y$s are very unlikely.

Much Better Idea
Terminology

• MLE (Maximum Likelihood Estimator):
  \[ Y^{\text{predict}} = \arg\max_v P(X_1 = u_1 \cdots X_m = u_m \mid Y = v) \]

• MAP (Maximum A-Posteriori Estimator):
  \[ Y^{\text{predict}} = \arg\max_v P(Y = v \mid X_1 = u_1 \cdots X_m = u_m) \]

Getting what we need

\[ Y^{\text{predict}} = \arg\max_v P(Y = v \mid X_1 = u_1 \cdots X_m = u_m) \]
Getting a posterior probability

\[
P(Y = v \mid X_1 = u_1 \cdots X_m = u_m) = \frac{P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)P(Y = v)}{P(X_1 = u_1 \cdots X_m = u_m)} = \frac{P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)P(Y = v)}{\sum_{j=1}^{n_Y} P(X_1 = u_1 \cdots X_m = u_m \mid Y = v_j)P(Y = v_j)}
\]

Bayes Classifiers in a nutshell

1. Learn the distribution over inputs for each value Y.
2. This gives P(X_1, X_2, \ldots, X_m / Y = v_i).
3. Estimate \( P(Y = v_i) \) as fraction of records with \( Y = v_i \).
4. For a new prediction:

\[
Y_{predict} = \arg\max_{v} P(Y = v \mid X_1 = u_1 \cdots X_m = u_m) = \arg\max_{v} P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)P(Y = v)
\]
Bayes Classifiers in a nutshell

1. Learn the distribution over inputs for each value $Y$.
2. This gives $P(X_1, X_2, \ldots, X_m \mid Y=v_i)$.
3. Estimate $P(Y=v_i)$ as fraction of records $Y=v_i$.
4. For a new prediction:
   $$Y^{\text{predict}} = \arg\max_v P(Y=v \mid X_1, X_2, \ldots, X_m = u_1, \ldots, u_m)$$

   We can use our favorite Density Estimator here.

   Right now we have two options:
   - Joint Density Estimator
   - Naïve Density Estimator

Joint Density Bayes Classifier

$$Y^{\text{predict}} = \arg\max_v P(X_1 = u_1, \ldots, X_m = u_m \mid Y=v)P(Y=v)$$

In the case of the joint Bayes Classifier this degenerates to a very simple rule:

$$Y^{\text{predict}} = \text{the most common value of } Y \text{ among records in which } X_1 = u_1, X_2 = u_2, \ldots, X_m = u_m.$$ 

Note that if no records have the exact set of inputs $X_1 = u_1, X_2 = u_2, \ldots, X_m = u_m$ then $P(X_1, X_2, \ldots, X_m \mid Y=v_i) = 0$ for all values of $Y$.

In that case we just have to guess $Y$'s value.
Joint BC Results: "Logical"

The "logical" dataset consists of 40,000 records and 4 Boolean attributes called a, b, c, d where a, b, c are generated 50-50 randomly as 0 or 1. $D = A \land \neg C$, except that in 10% of records it is flipped.

Joint BC Results: "All Irrelevant"

The "all irrelevant" dataset consists of 40,000 records and 15 Boolean attributes called a, b, c, ..., o where a, b, c are generated 50-50 randomly as 0 or 1. $v$ (output) = 1 with probability 0.75, 0 with prob 0.25.
Naïve Bayes Classifier

\[ Y^{\text{predict}} = \arg\max_{v} P(X_1 = u_1 \cdots X_m = u_m \mid Y = v) P(Y = v) \]

In the case of the naive Bayes Classifier this can be simplified:

\[ Y^{\text{predict}} = \arg\max_{v} P(Y = v) \prod_{j=1}^{n_Y} P(X_j = u_j \mid Y = v) \]

Technical Hint:
If you have 10,000 input attributes that product will underflow in floating point math. You should use logs:

\[ Y^{\text{predict}} = \arg\max_{v} \left( \log P(Y = v) + \sum_{j=1}^{n_Y} \log P(X_j = u_j \mid Y = v) \right) \]
**BC Results: “XOR”**

The “XOR” dataset consists of 40,000 records and 2 Boolean inputs called a and b, generated 50-50 randomly as 0 or 1. c (output) = a XOR b

---

**Naive BC Results: “Logical”**

The “logical” dataset consists of 40,000 records and 4 Boolean attributes called a,b,c,d where a,b,c are generated 50-50 randomly as 0 or 1. D = A^~C, except that in 10% of records it is flipped.
Naive BC Results: “Logical”

The “logical” dataset consists of 40,000 records and 4 Boolean attributes called a,b,c,d where a,b,c are generated 50-50 randomly as 0 or 1. D = A^~C, except that in 10% of records it is flipped.

This result surprised Andrew until he had thought about it a little.

The data shown in the figure is merely a subsample of the full dataset. The light color shades denote predicted classes. The dark shades are real data.

### Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>Parameters</th>
<th>FracRight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model1</td>
<td>bayesclass</td>
<td>density=joint, submodel=gauss</td>
<td>0.00905 +/- 0.005315</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gaussity=general</td>
<td></td>
</tr>
<tr>
<td>Model2</td>
<td>bayesclass</td>
<td>density=naive, submodel=gauss</td>
<td>0.00905 +/- 0.005315</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gaussity=general</td>
<td></td>
</tr>
</tbody>
</table>

Naive BC Results: “All Irrelevant”

The “all irrelevant” dataset consists of 40,000 records and 15 Boolean attributes called a,b,c,d..o where a,b,c are generated 50-50 randomly as 0 or 1. v (output) = 1 with probability 0.75, 0 with prob 0.25.

The Classifier learned by “Naive BC”

### Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Model</th>
<th>Parameters</th>
<th>FracRight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model1</td>
<td>bayesclass</td>
<td>density=joint, submodel=gauss</td>
<td>0.70455 +/- 0.0053537</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gaussity=general</td>
<td></td>
</tr>
<tr>
<td>Model2</td>
<td>bayesclass</td>
<td>density=naive, submodel=gauss</td>
<td>0.75055 +/- 0.00281976</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gaussity=general</td>
<td></td>
</tr>
</tbody>
</table>
BC Results: “MPG”: 392 records

The Classifier learned by “Naive BC”

BC Results: “MPG”: 40 records
More Facts About Bayes Classifiers

• Many other density estimators can be slotted in*.
• Density estimation can be performed with real-valued inputs*.
• Bayes Classifiers can be built with real-valued inputs*.
• Rather Technical Complaint: Bayes Classifiers don’t try to be maximally discriminative---they merely try to honestly model what’s going on*.
• Zero probabilities are painful for Joint and Naïve. A hack (justifiable with the magic words “Dirichlet Prior”) can help*.
• Naïve Bayes is wonderfully cheap. And survives 10,000 attributes cheerfully!

What you should know

• Probability
  • Fundamentals of Probability and Bayes Rule
  • What’s a Joint Distribution
  • How to do inference (i.e. P(E1|E2)) once you have a JD

• Density Estimation
  • What is DE and what is it good for
  • How to learn a Joint DE
  • How to learn a naïve DE

*See future Andrew Lectures
What you should know

- Bayes Classifiers
  - How to build one
  - How to predict with a BC
  - Contrast between naïve and joint BCs