CPSC 340: Machine Learning and Data Mining

Generative Models

Fall 2016

Admin

- Assignment 1 is out, due September 23rd.
  - Setup your CS undergrad account ASAP to use Handin:
    - https://www.cs.ubc.ca/getacct
    - Instructions for handin will be posted to Piazza.
  - Try to do the assignment this week, BEFORE add/drop deadline.
    - The material will be getting much harder and the workload much higher.
    - I'll give alternatives to p-files for Octave after class.
  - Tutorial slides posted.
- Registration:
  - Keep checking your registration, if could change quickly.
  - You need to be registered in a tutorial section to stay enrolled.

Should you trust them?

- Scenario 1:
  - "I built a model based on the data you gave me."
  - "It classified your data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably not:
  - They are reporting training error.
  - This might have nothing to do with test error.
  - E.g., they could have fit a very deep decision tree.
- Why 'probably'?
  - If they only tried a few very simple models, the 98% might be reliable.
  - E.g., they only considered decision stumps with simple 1-variable rules.

Should you trust them?

- Scenario 2:
  - "I built a model based on half of the data you gave me."
  - "It classified the other half of the data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably:
  - They computed the validation error once.
  - This is an unbiased approximation of the test error.
  - Trust them if you believe they didn't violate the golden rule.

Should you trust them?

- Scenario 3:
  - "I built 10 models based on half of the data you gave me."
  - "One of them classified the other half of the data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably:
  - They computed the validation error a small number of times.
  - Maximizing over these errors is a biased approximation of test error.
  - But they only maximized it over 10 models, so bias is probably small.
  - They probably know about the golden rule.

Should you trust them?

- Scenario 4:
  - "I built 1 billion models based on half of the data you gave me."
  - "One of them classified the other half of the data with 98% accuracy."
  - "It should get 98% accuracy on the rest of your data."

- Probably not:
  - They computed the validation error a huge number of times.
  - Maximizing over these errors is a biased approximation of test error.
  - They tried so many models, one of them is likely to work by chance.
- Why 'probably'?
  - If the 1 billion models were all extremely-simple, 98% might be reliable.

Should you trust them?

- Scenario 5:
  - “I built 1 billion models based on the first third of the data you gave me.”
  - “One of them classified the second third of the data with 98% accuracy.”
  - “It also classified the last third of the data with 98% accuracy.”
  - “It should get 98% accuracy on the rest of your data.”

- Probably:
  - They computed the first validation error a huge number of times.
  - But they had a second validation set that they only looked at once.
  - The second validation set gives unbiased test error approximation.
  - This is ideal, as long as they didn’t violate golden rule on second set.
  - And assuming you are using IID data in the first place.

The ‘Best’ Machine Learning Model

- Decision trees are not always most accurate.
- What is the ‘best’ machine learning model?
- First we need to define generalization error:
  - Test error on new examples (excludes test examples seen during training).
- No free lunch theorem:
  - There is **no** ‘best’ model achieving the best generalization error for every problem.
  - If model A generalizes better to new data than model B on one dataset, there is another dataset where model B works better.
- This question is like asking which is ‘best’ among “rock”, “paper”, and “scissors”.

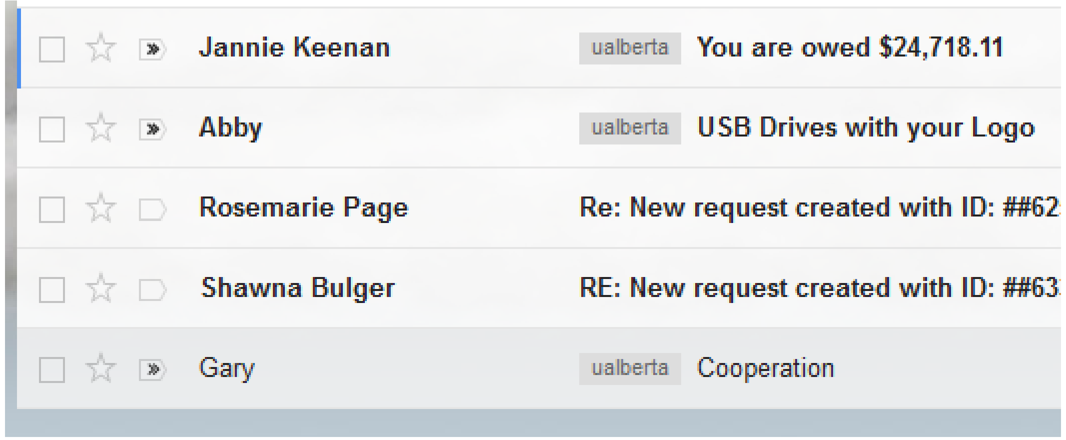
The ‘Best’ Machine Learning Model

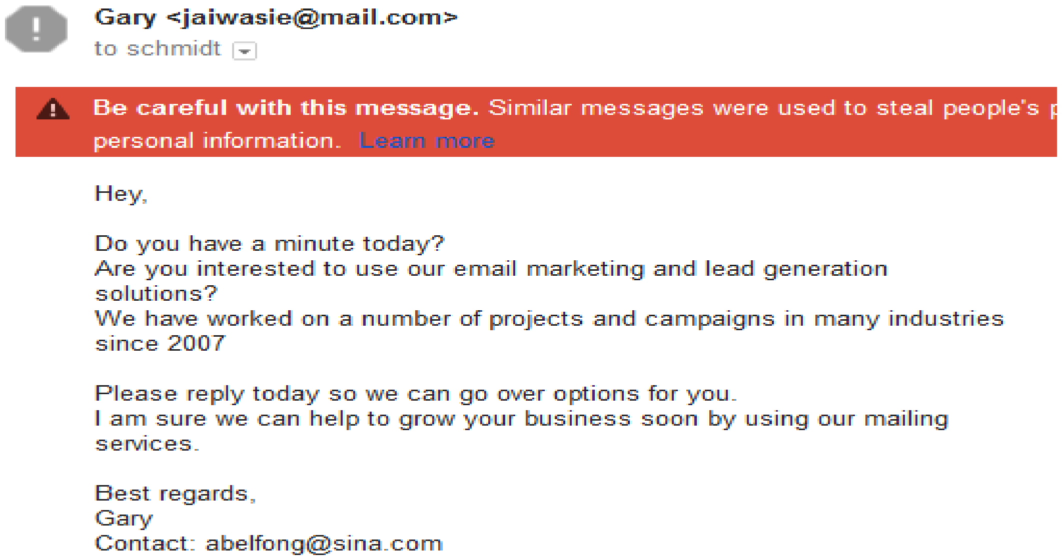
- Implications of the lack of a ‘best’ model:
  - We need to learn about and try out multiple models.
- So which ones to study in CPSC 340?
  - We’ll usually motivate a method by a specific application.
  - But we’ll focus on models that are effective in many applications.

- Caveat of no free lunch (NFL) theorem:
  - The world is very structured.
  - Some datasets are more likely than others.
  - Model A really could be better than model B on every real dataset in practice.
- Machine learning research:
  - Large focus on models that are useful across many applications.

Application: E-mail Spam Filtering

- Want a build a system that filters spam e-mails.

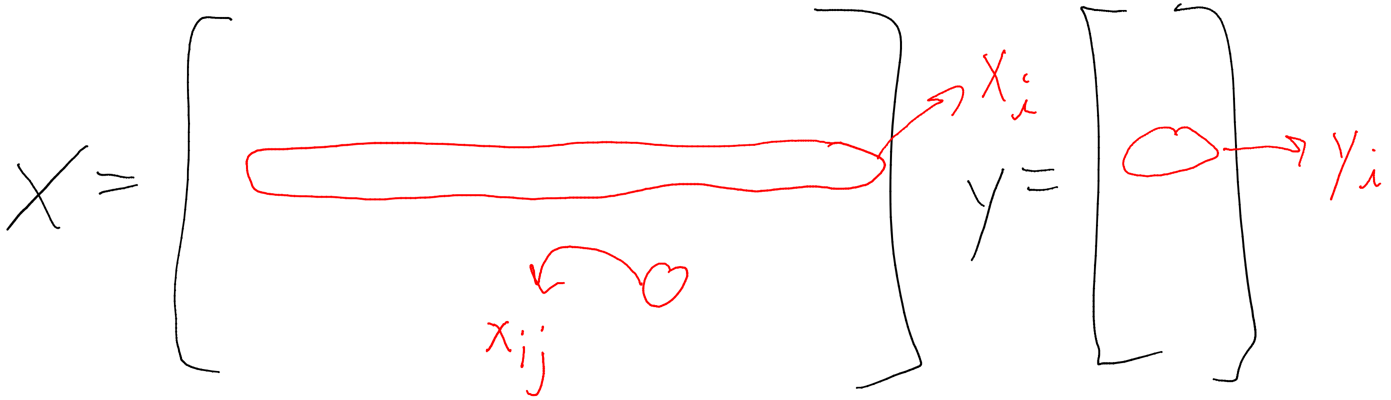




- We have a big collection of e-mails, labeled by users.
- Can we formulate as supervised learning?

First a bit more supervised learning notation

- We have been using the notation ‘X’ and ‘y’ for supervised learning:



- X is matrix of all features, y is vector of all labels.
- Need a way to refer to the features and label of specific object ‘i’.
  - We use y_i for the label of object ‘i’ (element ‘i’ of ‘y’).
  - We use x_i for the features object ‘i’ (row ‘i’ of ‘X’).
  - We use x_{ij} for feature ‘j’ of object ‘i‘.

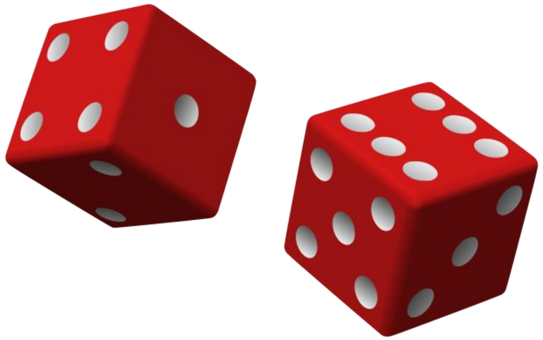
Feature Representation for Spam

- How do we make label 'y_i' of an individual e-mail?
  - (y_i = 1) means 'spam', (y_i = 0) means 'not spam'.
- How do we construct features 'x_i' for an e-mail?
  - Use bag of words:
    - "hello", "vicodin", "$".
    - "vicodin" feature is 1 if "vicodin" is in the message, and 0 otherwise.
  - Could add phrases:
    - "be your own boss", "you're a winner", "CPSC 340".
  - Could add regular expressions:
    - <recipient>, <sender domain == "mail.com">

Probabilistic Classifiers

- For years, best spam filtering methods used naïve Bayes.
  - Naïve Bayes is a probabilistic classifier based on Bayes rule.
  - It's "naïve" because it makes a strong conditional independence assumption.
  - But it tends to work well with bag of words.
- Probabilistic classifiers model the conditional probability, $p(y_i \mid x_i)$.
  - "If a message has words x_i, what is probability that message is spam?"
- If $p(y_i = \text{'spam'} \mid x_i) > p(y_i = \text{'not spam'} \mid x_i)$, classify as spam.

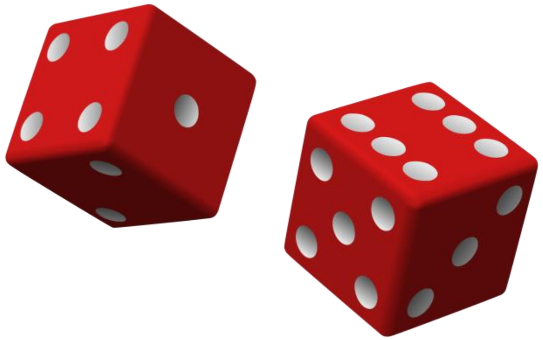
Digression to Review Probabilities…



https://en.wikipedia.org/wiki/Dice_throw_%28review%29

Digression to Review Probabilities…

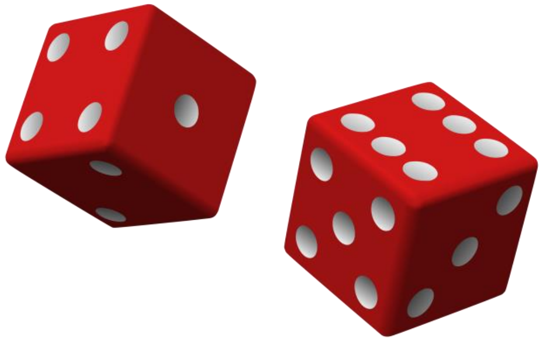
- Dungeons & Dragons scenario:
  - You roll dice 1:
    - Roll 5 or 6 you sneak past monster.
    - Otherwise, you are eaten.
  - If you survive, you roll dice 2:
    - Roll 4-6, find pizza.
    - Otherwise, you find nothing.



https://en.wikipedia.org/wiki/Dice_throw_%28review%29
http://www.dungeonsdragonscartoon.com/2011/11/cloak.html

Digression to Review Probabilities…

- Dungeons & Dragons scenario:
  - You roll dice 1:
    - Roll 5 or 6 you sneak past monster.
    - Otherwise, you are eaten.
  - If you survive, you roll dice 2:
    - Roll 4-6, find pizza.
    - Otherwise, you find nothing.



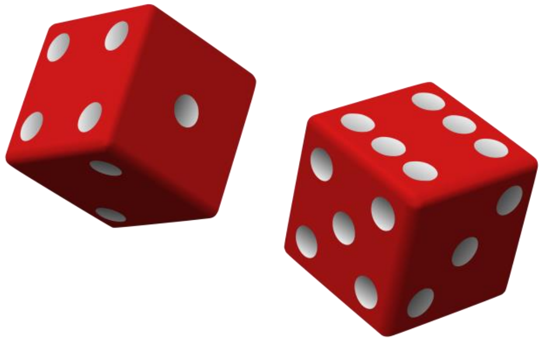
- Probabilities defined on 'event space':

D1\D2	1	2	3	4	5	6
1						
2						
3		$D_1=3,D_2=2$				
4						
5						
6						

https://en.wikipedia.org/wiki/Dice_throw_%28review%29
http://www.dungeonsdragonscartoon.com/2011/11/cloak.html

Digression to Review Probabilities…

- Dungeons & Dragons scenario:
  - You roll dice 1:
    - Roll 5 or 6 you sneak past monster.
    - Otherwise, you are eaten.
  - If you survive, you roll dice 2:
    - Roll 4-6, find pizza.
    - Otherwise, you find nothing.



- Probabilities defined on ‘event space’:

D1\D2	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

¬Survive

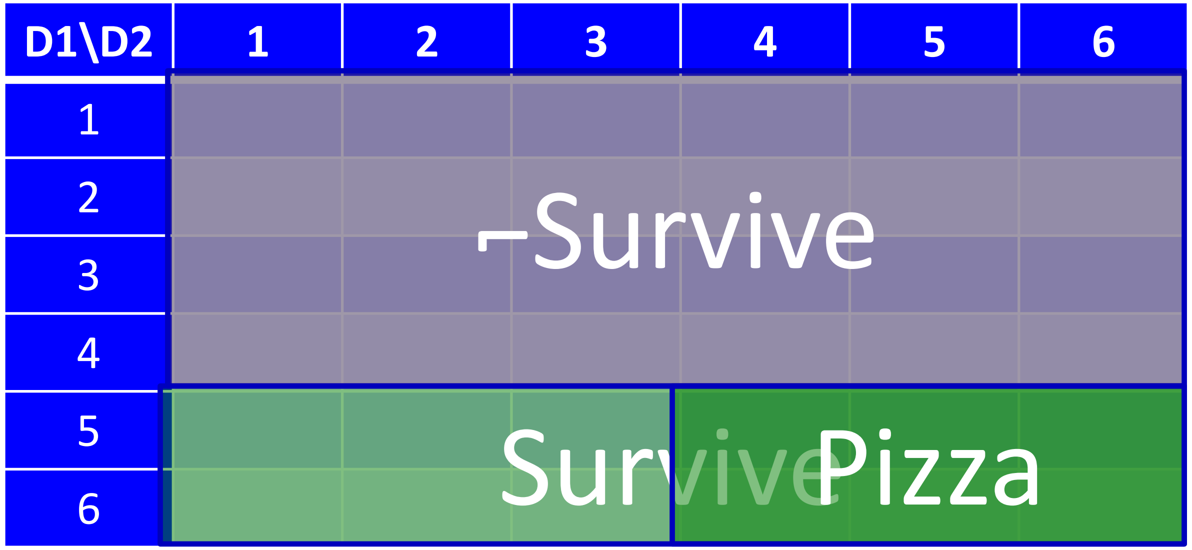
Survive

Pizza

https://en.wikipedia.org/wiki/Dice_throw_%28review%29
http://www.dungeonsdragonscartoon.com/2011/11/cloak.html

Calculating Basic Probabilities

- Probability of event ‘A’ is ratio:
  - p(A) = Area(A)/TotalArea.
  - “Likelihood” that ‘A’ happens.
- Examples:
  - p(Survive) = 12/36 = 1/3.
  - p(Pizza) = 6/36 = 1/6.
  - p(¬Survive) = 1 – p(Survive) = 2/3.



Calculating Basic Probabilities

- Probability of event ‘A’ is ratio:
  - p(A) = Area(A)/TotalArea.
  - “Likelihood” that ‘A’ happens.
- Examples:
  - p(Survive) = 12/36 = 1/3.
  - p(Pizza) = 6/36 = 1/6.
  - p($\neg$Survive) = 1 – p(Survive) = 2/3.
  - p(D_1 is even) = 18/36 = ½.

D1\D2	1	2	3	4	5	6
1						
2	D_1 is even					
3						
4	D_1 is even					
5						
6	D_1 is even					

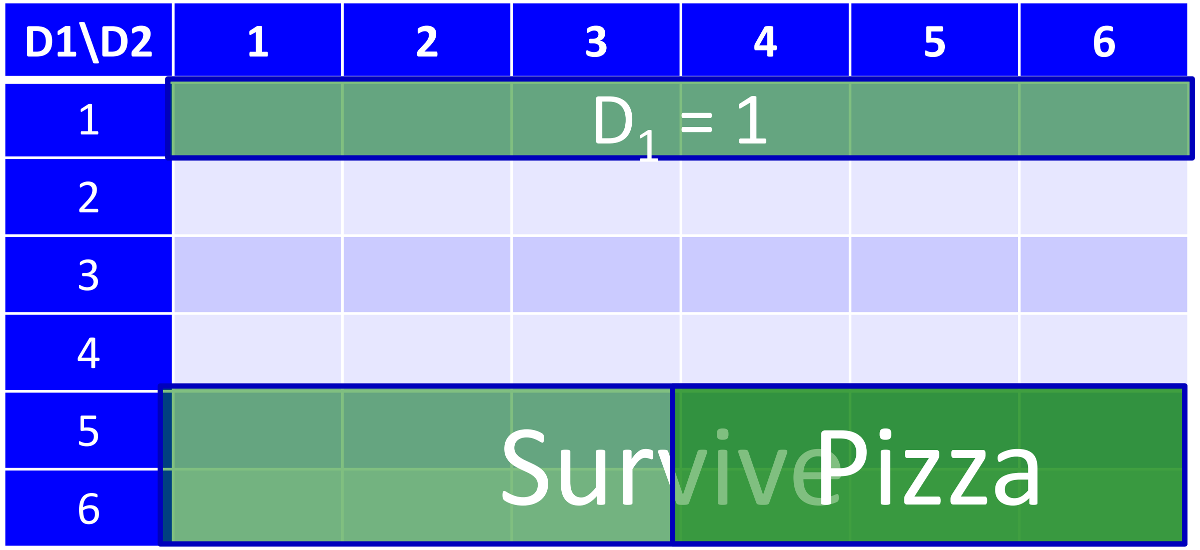
Random Variables and ‘Sum to 1’ Property

- Random variable: variable whose value depends on probability.
- Example: event (D_1 = x) depends on random variable D_1.
- Convention:
  - We’ll use p(x) to mean p(X = x), when random variable X is obvious.
- Sum of probabilities of random variable over entire domain is 1:
  - $\sum_x p(x) = 1$.
  - E.g, $\sum_i p(D_1 = i)$ = 1/6+1/6 + …
    = 1.

<table>
<tr><th>D1\D2</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th></tr>
<tr><td>1</td><td colspan="6">D_1 =1</td></tr>
<tr><td>2</td><td colspan="6">D_1 =2</td></tr>
<tr><td>3</td><td colspan="6">D_1 = 3</td></tr>
<tr><td>4</td><td colspan="6">D_1 = 4</td></tr>
<tr><td>5</td><td colspan="6">D_1 = 5</td></tr>
<tr><td>6</td><td colspan="6">D_1 = 6</td></tr>
</table>

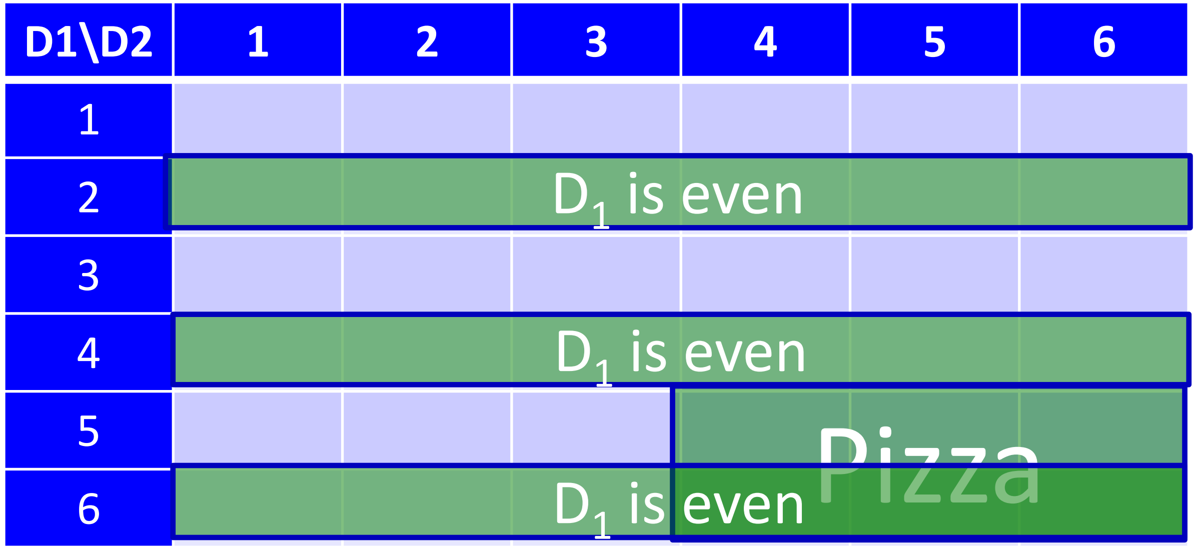
Joint Probability

- Joint probability: probability that A and B happen, written ‘p(A,B)’.
  - Intersection of Area(A) and Area(B).
- Examples:
  - $p(D_1 = 1, \text{Survive}) = 0$.
  - p(Survive, Pizza) = 6/36 = 1/6.



Joint Probability

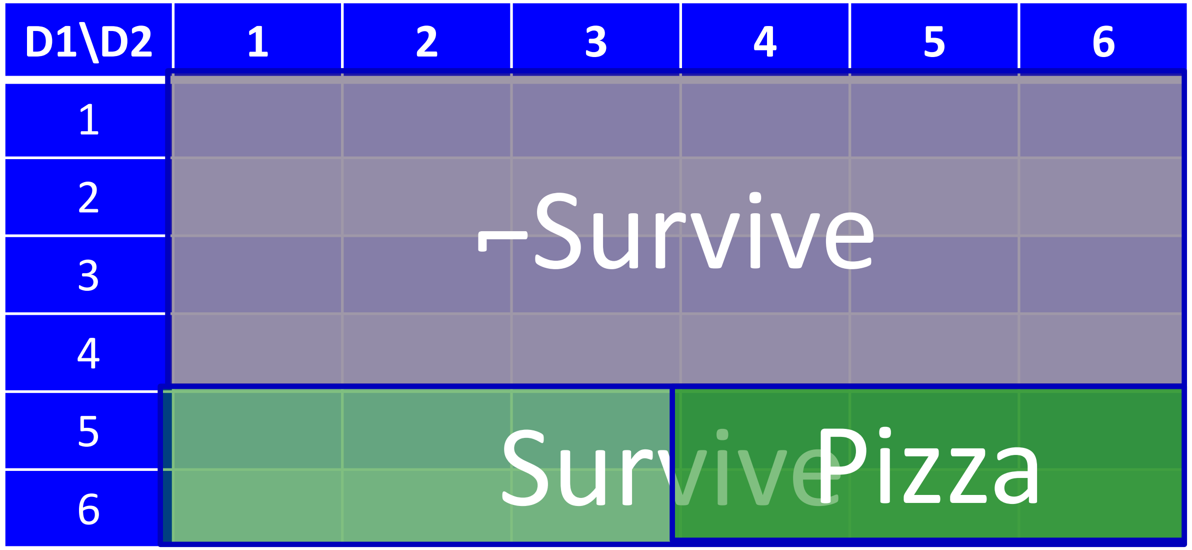
- Joint probability: probability that A and B happen, written ‘p(A,B)’.
  - Intersection of Area(A) and Area(B).
- Examples:
  - $p(D_1 = 1, \text{Survive}) = 0$.
  - $p(\text{Survive}, \text{Pizza}) = 6/36 = 1/6$.
  - $p(D_1 \text{ even}, \text{Pizza}) = 3/36 = 1/12$.



- Note: order of A and B does not matter

Marginalization Rule

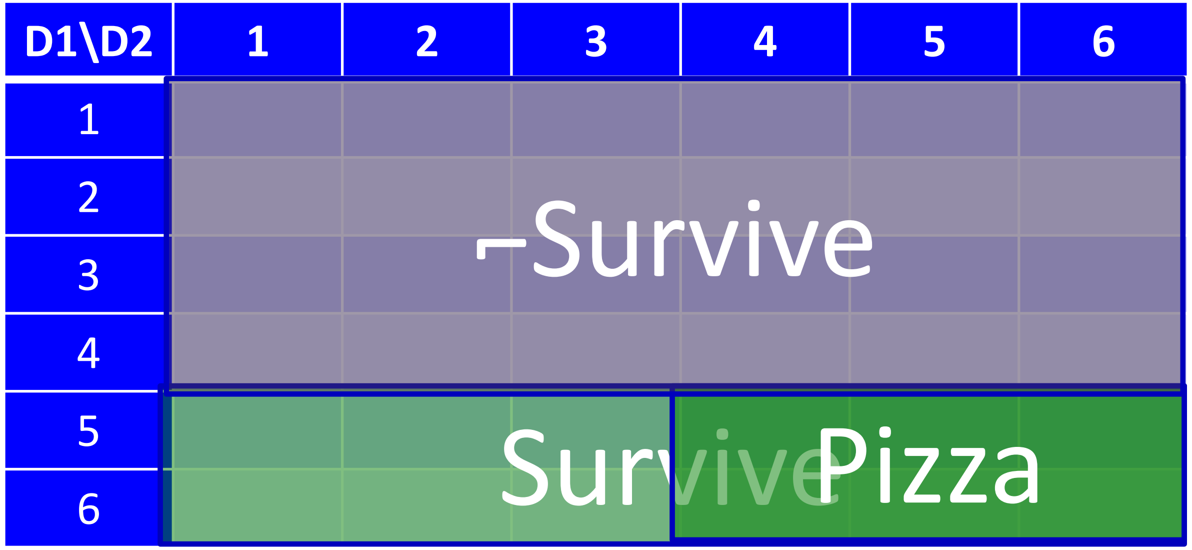
- Marginalization rule:
  - $P(A) = \sum_x P(A, X = x)$.
  - Summing joint over all values of one variable gives probability of the other.
  - Example: $P(Pizza) = P(Pizza, Survive) + P(Pizza, \neg Survive) = \frac{1}{6}$.



  - Applying rule twice: $\sum_x \sum_y p(Y = y, X = x) = 1$.

Conditional Probability

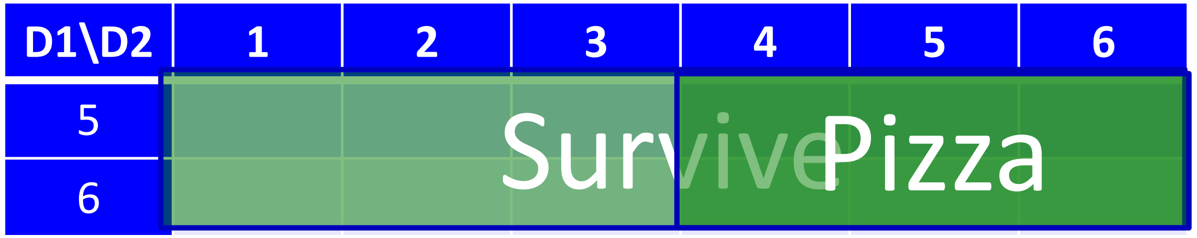
- Conditional probability:
  - probability that A will happen *if we know* that B happens.
  - “probability of A *restricted* to scenarios where B happens”.
  - Written p(A|B), said “probability of A given B”.
- Calculation:
  - Within area of B:
    - Compute Area(A)/TotalArea.
  - p(Pizza | Survive) =



Conditional Probability

- Conditional probability:
  - probability that A will happen *if we know* that B happens.
  - “probability of A *restricted* to scenarios where B happens”.
  - Written p(A|B), said “probability of A given B”.
- Calculation:
  - Within area of B:
    - Compute Area(A)/TotalArea.
  - p(Pizza | Survive) =
    p(Pizza, Survive)/p(Survive) = 6/12 = ½.
  - Higher than p(Pizza, Survive) = 6/36 = 1/6.
  - More generally, p(A | B) = p(A,B)/p(B).

Geometrically: compute area of A on new space where B happened.



‘Sum to 1’ Properties and Bayes Rule.

- Conditional probability P(A | B) sums to one over all A:
  - $\sum_x P(x \mid B) = 1.$
  - P(Pizza | Survive) + P(¬ Pizza | Survive) = 1.
  - P(Pizza | Survive) + P(Pizza | ¬Survive) ≠ 1.
- Product rule: p(A,B) = p(A | B)p(B).
- Bayes Rule:

$$p(A \mid B) = \frac{p(B \mid A)\,p(A)}{p(B)}$$

  - Allows you to “reverse” the conditional probability.
- Example:
  - P(Pizza | Survive) = P(Survive | Pizza)P(Pizza)/P(Survive)
    = (1) * (1/6) / (1/3) = ½.
  - http://setosa.io/ev/conditional-probability

Back to E-mail Spam Filtering…

- Recall our spam filtering setup:
  - y_i: whether or not the e-mail was spam.
  - x_i: the set of words/phrases/expressions in the e-mail.
- To model conditional probability, naïve Bayes uses Bayes rule:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\, p(y_i = \text{"spam"})}{p(x_i)}$$

- Easy part #1: $p(y_i$ = 'spam') is the probability that an e-mail is spam.
  - Count of number of times (y_i = 'spam') divided by number of objects 'n'.
  - For (complicated) proof of this (simple) fact, see:
    - http://www.cs.ubc.ca/~schmidtm/Courses/540-F14/naiveBayes.pdf

Back to E-mail Spam Filtering…

- Recall our spam filtering setup:
  - y_i: whether or not the e-mail was spam.
  - x_i: set of words/phrases/expressions in the e-mail.
- To model conditional probability, naïve Bayes uses Bayes rule:

$$p(y_i = \text{"spam"} \mid x_i) = \frac{p(x_i \mid y_i = \text{"spam"})\,p(y_i = \text{"spam"})}{p(x_i)}$$

- Easy part #2: We don't need $p(x_i)$.

To test $p(y_i = \text{"spam"} \mid x_i)$ we just need to know if $p(y_i = \text{"spam"} \mid x_i) > p(y_i = \text{"not spam"} \mid x_i)$.

By Bayes rule this is equivalent to $\frac{p(x_i \mid y_i = \text{"spam"})\,p(y_i = \text{"spam"})}{p(x_i)} > \frac{p(x_i \mid y_i = \text{"not spam"})\,p(y_i = \text{"not spam"})}{p(x_i)}$

Denominators are the same so we just test $p(x_i \mid y_i = \text{"spam"})\,p(y_i = \text{"spam"}) > p(x_i \mid y_i = \text{"not spam"})\,p(y_i = \text{"not spam"})$

Generative Classifiers

- The hard part is estimating $p(x_i \mid y_i = \text{'spam'})$:
  - the probability of seeing the words/expressions x_i if the e-mail is spam.
- Classifiers based on Bayes rule are called generative classifier:
  - It needs to know the probability of the features, given the class.
    - How to "generate" features.
  - You need a model that knows what spam messages look like.
    - And a second that knows what non-spam messages look like.
  - This work well with tons of features compared to number of objects.

Generative Classifiers

- But does it need to know language to model $p(x_i \mid y_i)$???
- To fit generative models, usually make BIG assumptions:
  - Gaussian discriminant analysis (GDA):
    - Assume that $p(x_i \mid y_i)$ follows a multivariate normal distribution.
  - Naïve Bayes (NB):
    - Assume that each variables in x_i is independent of the others in x_i given y_i.

Summary

- No free lunch theorem: there is no “best” ML model.
- Joint probability: probability of A and B happening.
- Conditional probability: probability of A if we know B happened.
- Generative classifiers: build a probability of seeing the features.

- Next time:
  - A “best” machine learning model as ‘n’ goes to ∞.