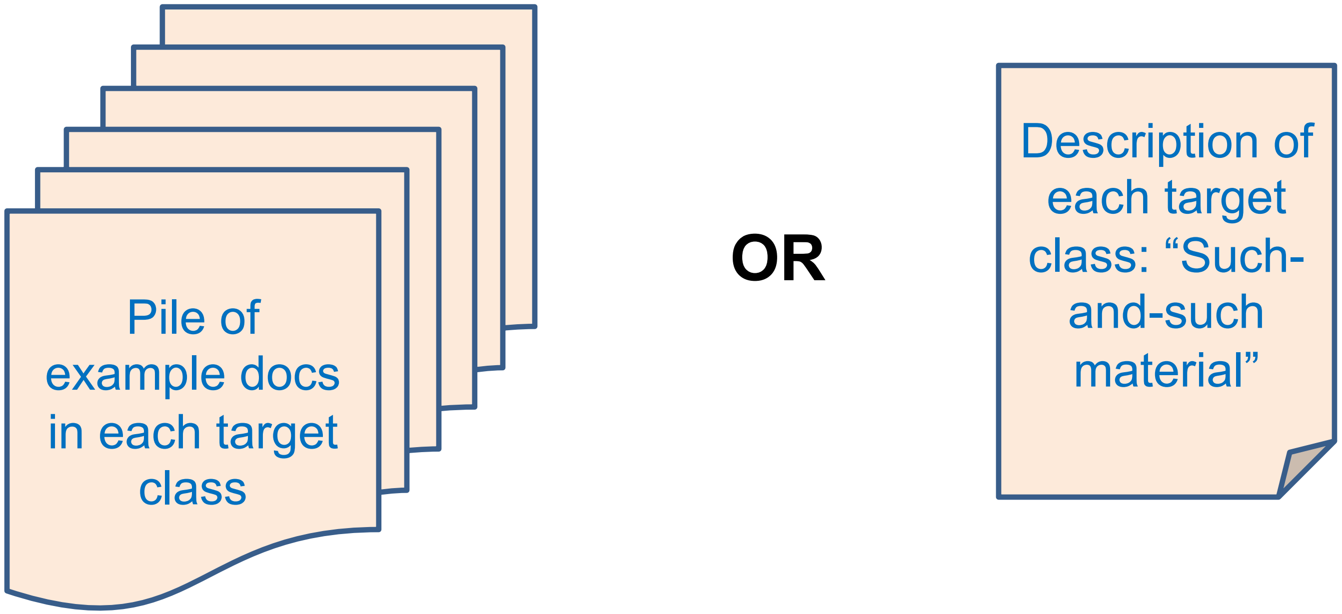
Automated Classifying of Documents

Two Types of Text Classification

Text classification can be based on two different types of reference:

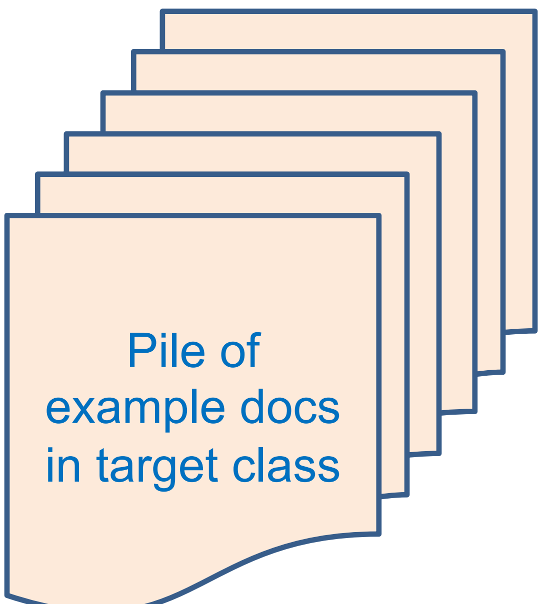
- Content-based classification
- Descriptor-based classification



Two Types of Text Classification

Text classification can be based on two different types of reference:

- **Content-based classification**
- Descriptor-based classification



We start with two or more classes of existing content, where our task is to classify documents into the same categories.

Two Types of Text Classification

Text classification can be based on two different types of reference:

- Content-based classification
- **Descriptor-based classification**

Instead of example documents, there is a user-inputted description of the content desired.

Description of each target class: “Such-and-such material”

Examples of Content-Based Text Classification

Most applications of content-based text classification involve the placement of documents into a binary classification, or into a preestablished subject-matter hierarchy.

- An example of the binary classifier is a spam filter (or an offensive content filter).
- An example of a multiclass application is the browsing taxonomy of broad content network (e.g., news website, blog network).

Examples of Descriptor-Based Text Classification

Applications of descriptor-based text classification can also be binary or multiclass.

- Binary classification:
  - Legal discovery orders in court cases
  - FOIA* requests
- Multiclassification
  - This would be, for example, the *initial* populating of taxonomies with documents.
  - This assumes a set of well-defined taxonomy nodes that lack preexisting example documents.

**Freedom of Information Act*

Text Classification vs. Document Classification

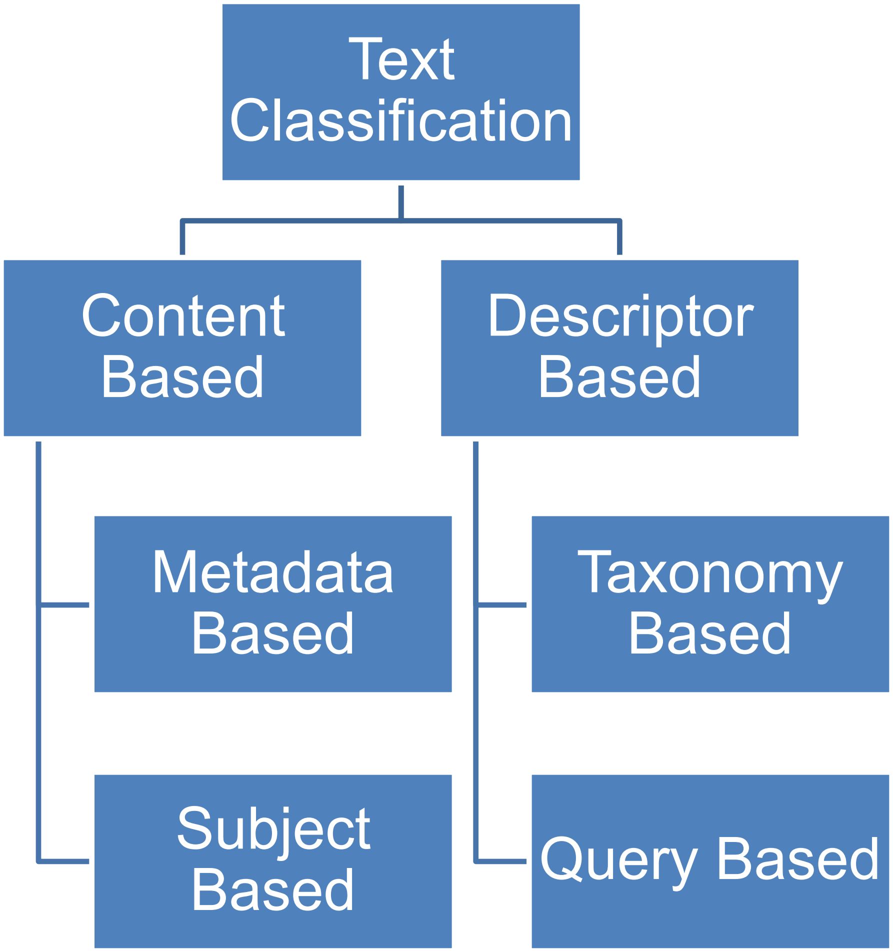
- It’s possible to classify short texts such as queries, headlines, subtitles, i.e., texts that are not prototypical “documents.”
- When we say “document classification,” we tend to mean classifying relatively *discursive* texts, i.e., those having multiple complete sentences concerning the same subject matter or narrative.

The Text-Classification Landscape

- The NLP community is heavily engaged with *supervised learning* techniques on discursive documents. There are many packages and a wealth of literature on these things.
- In contrast, there is much less available today for automated descriptor-based classification.

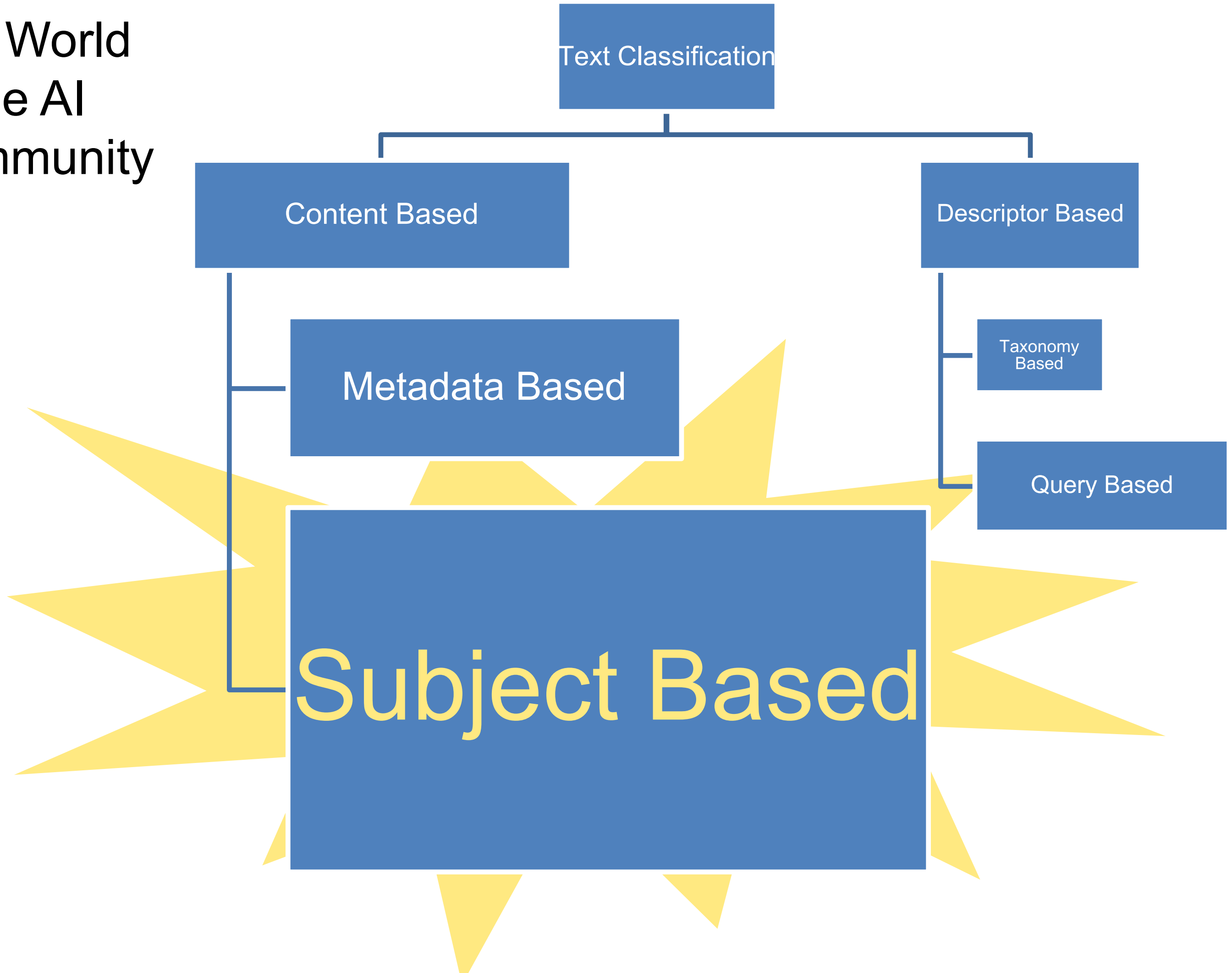
The Text-Classification Landscape

The
Actual
World



The Text-Classification Landscape

The World of the AI Community



DataScience@SMU

Subject-Based Classification

Two Approaches to Subject-Based Automated Document Classification

We'll look more at these subject-based models:

- Multinomial Naïve Bayes classifiers
- SVM*-based classifiers

**Support Vector Machine*

Terminology for Naïve Bayes Models

- The *class* is what we're trying to predict from a set of attributes collectively termed the *predictor.*
- The *prior probability* of the class is the global distribution of individuals into that class.
- Likewise, the prior probability of the *predictor* is the global distribution of individuals into that predictor (items sharing that set of attributes).

Terminology for Naïve Bayes Models

- The *posterior probability* of the *predictor* is the probability, just among items in the target class, of having the predictor attributes.
- The *posterior probability* of the *class* is the probability, just among items sharing the predictor attributes, of falling into the target class.
- What a Naïve Bayes classifier essentially does is calculate the *posterior probability of the class.*

Why Are These Things Called What They Are Called?

What's this funny language about "prior" and "posterior" probabilities?

- The language was derived from similar terminology in a branch of philosophy called "epistemology"—the theory of knowledge (how we know what we know, including both before and after we make observations).
- So the key is to think of "prior" and "posterior" as meaning "before we observe something" and "after we observe something."

Why Are These Things Called What They Are Called?

The terms really do make sense.

- The "prior probability of the class" is an answer to the question "What's the probability of the class *before* we know the predictor attributes?"
- And correspondingly, the "posterior probability of the class" is an answer to the question "What's the probability of the class *after* we know the predictor attributes?"

Naïve Bayes

Putting the foregoing terms together:

Posterior probability of the predictor (once you know the class)

Prior probability of the class (before you know the predictor)

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Posterior probability of the class (once you know the predictor)

Prior probability of the predictor (before you know the class)

where $P(\mathrm{x}|\mathrm{c}) = P(\mathrm{x}_1|\mathrm{c}) \times P(\mathrm{x}_2|\mathrm{c}) \times \ldots \times P(\mathrm{x}_n|\mathrm{c})$

Example of Naïve Bayes Text Classifier

Suppose we are tired of getting all those spam emails for Viagra, so we are building a spam filter. We have 74 emails as our training data:

	all 74 emails	contains "penis"	contains ”Viagra"
Not Spam	44	31	1
Spam	30	20	24

Example of Naïve Bayes Text Classifier

Then for a new email that contains both "penis" and "Viagra":

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(spam|penis, viagra)$$

$$= \frac{P(penis|spam) * P(viagra|spam) * P(spam)}{P(penis) * P(viagra)}$$

$$= \frac{\frac{20}{30} * \frac{24}{30} * \frac{30}{74}}{\frac{51}{74} * \frac{25}{74}} = 0.928$$

	all 74 emails	contains "penis"	contains "Viagra"
Not Spam	44	31	1
Spam	30	20	24

After Nedelcu 2012

Multinomial Naïve Bayes Classifier

- The multinomial Bayes classifier lets us extend the previous case to applications where there is a multiclass rather than binary outcome.
- The **`scikit-learn`** package has an implementation of this for Python, called **`MultinomialNB`**.

DataScience@SMU

Classification with SVMs

SVM Classifier

- Every SVM classifier is a binary classifier—but we can combine multiple SVMs to make a multiclass solution.
- First we’ll start by understanding the binary classifier implementation.

SVM Classifier

Ideally, there is a linear space within the margins of feature vector space, separating two classes.

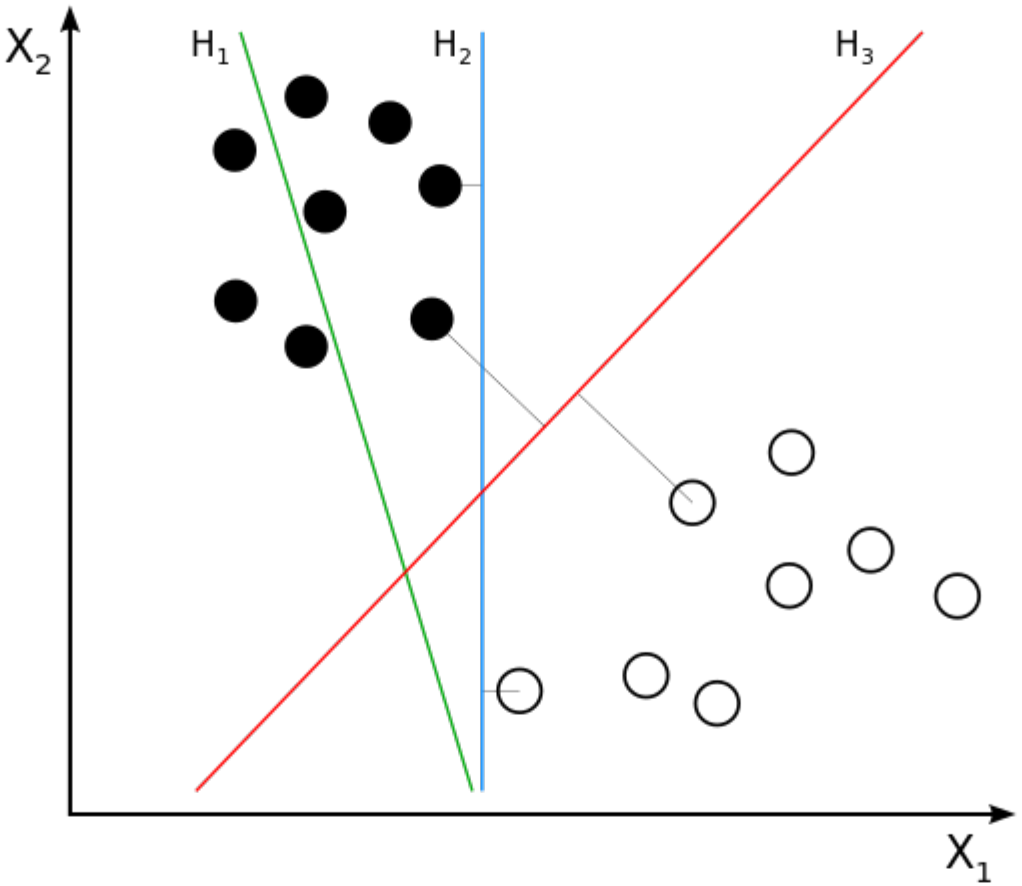


Image from Weinberg 2012

SVM Classifier

The possible demarcations between classes are called "hyperplanes."

For now you can think of them as separation lines. Here you see three of them.

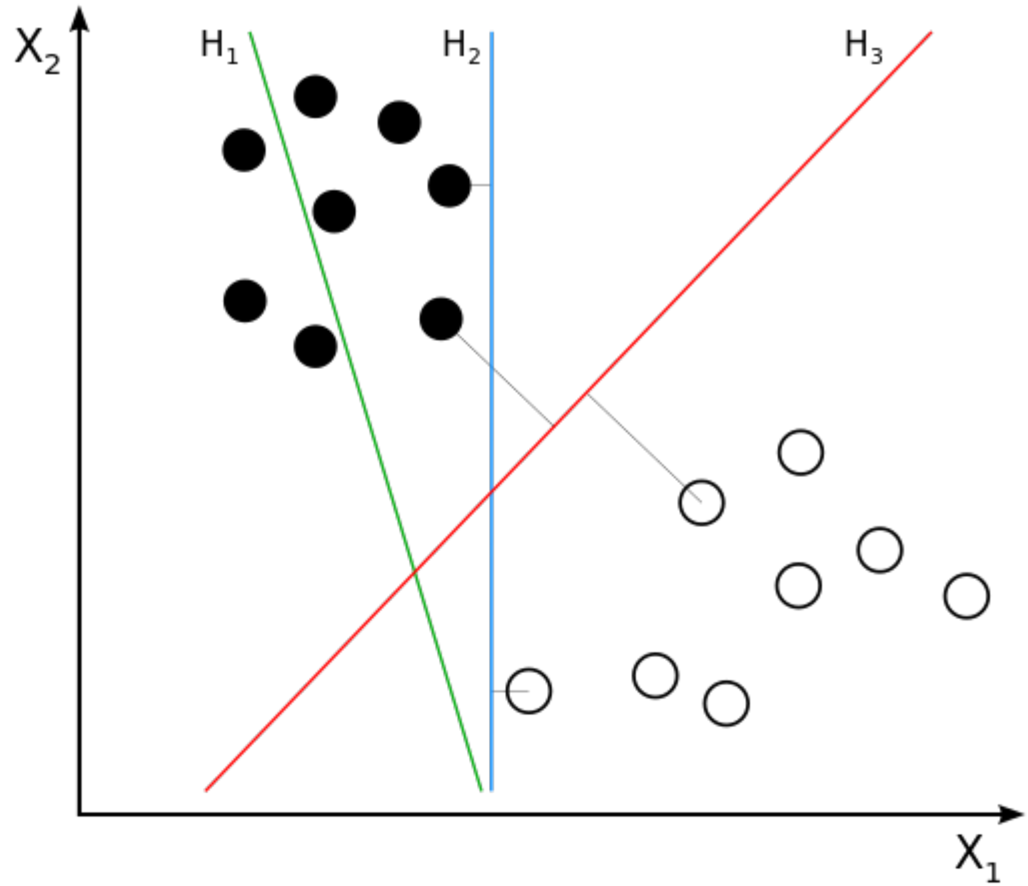


Image from Weinberg 2012

SVM Classifier

- H1 does not separate the classes.
- H2 separates them but only by a small margin.
- H3 actually has the *maximum margin* possible—so it is the hyperplane that would be determined by the SVM.

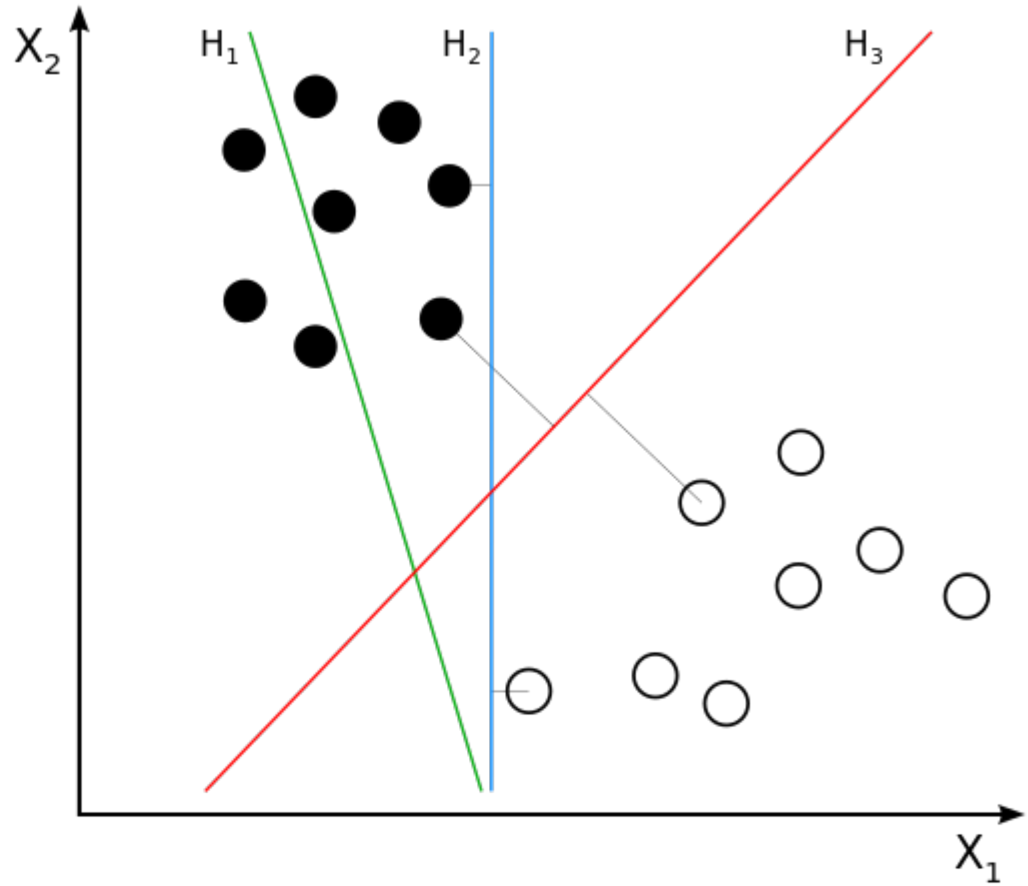


Image from Weinberg 2012

Harder Cases

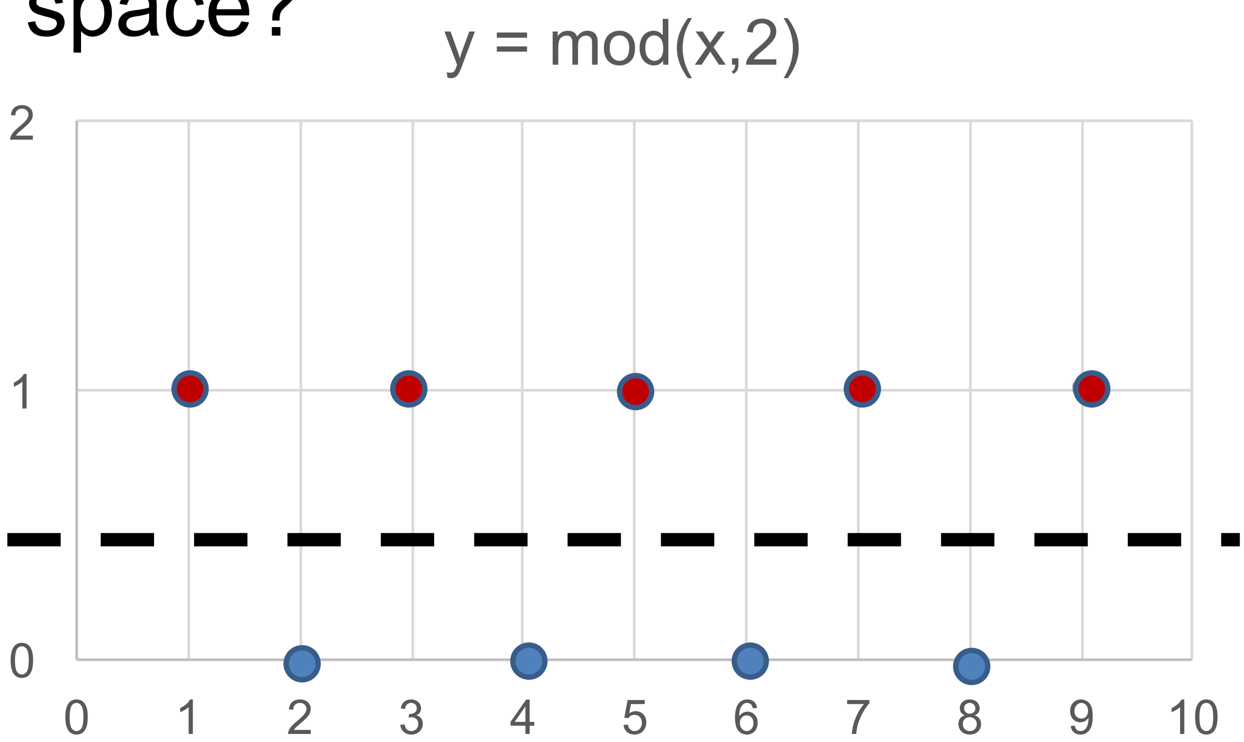
- In actuality, there might not be a nice straight line all the time—unless the space is projected into higher dimensions.
- What does that mean?

Projecting to Higher Dimensionality

- Imagine this one-dimensional vector space with two classes (**red** and **blue**):
  1 2 3 4 5 6 7 8 9
- There's no line that separates the two classes.

Projecting to Higher Dimensionality

But what happens if we map the data by a simple function onto a two-dimensional space?



This illustrates that we can easily increase the dimensionality of data, so it can be possible to draw a line of separation that would not exist otherwise.

Harder Cases

- What if we can't draw a line just because of one weird case?

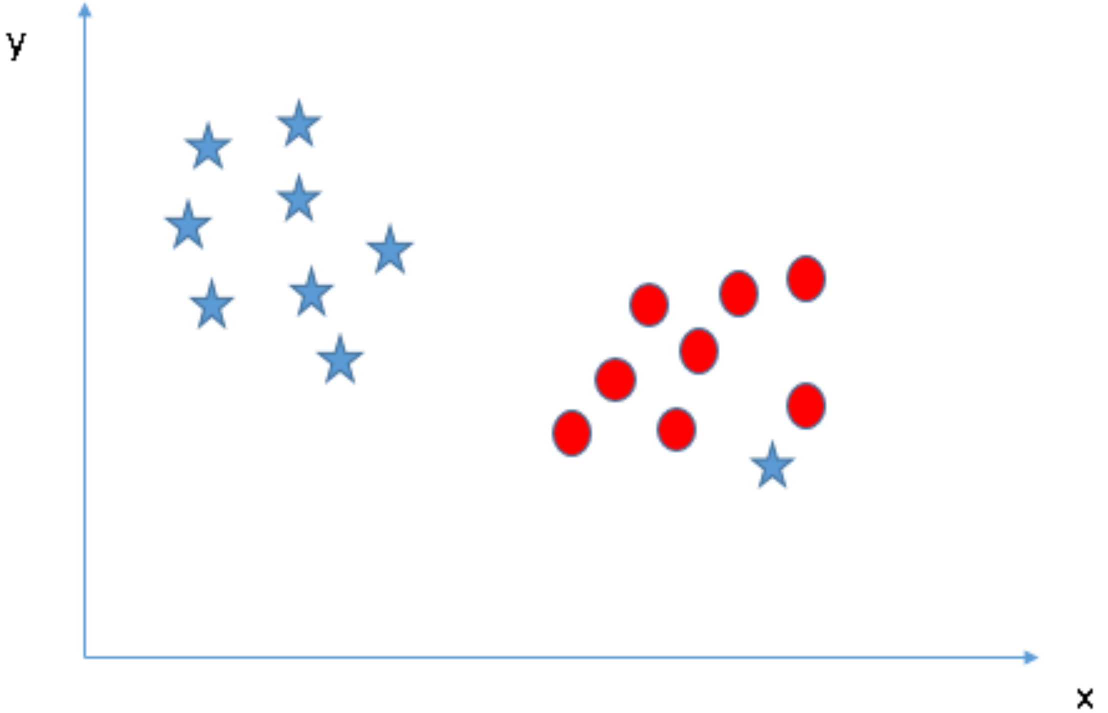


Image from Ray 2017

Harder Cases

- Not to fear, outlier detection is here!

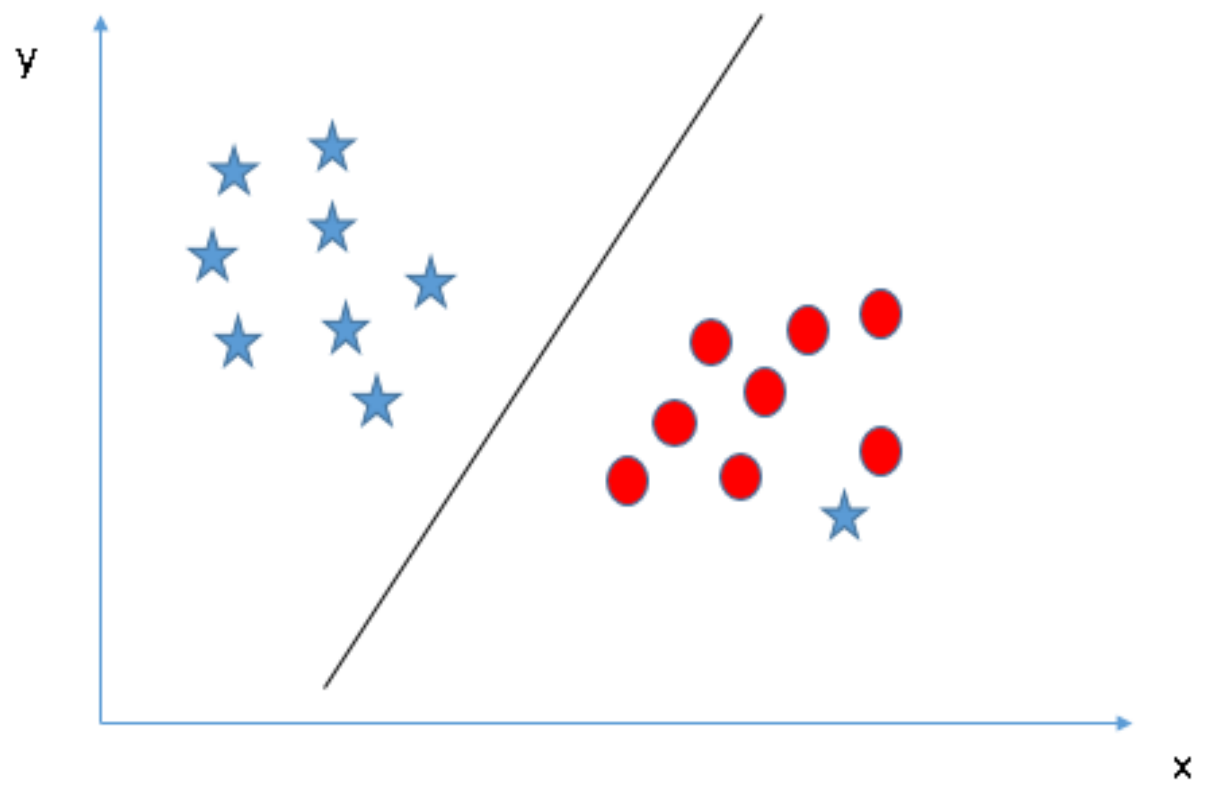


Image from Ray 2017

Harder Cases

This doesn't look doable.

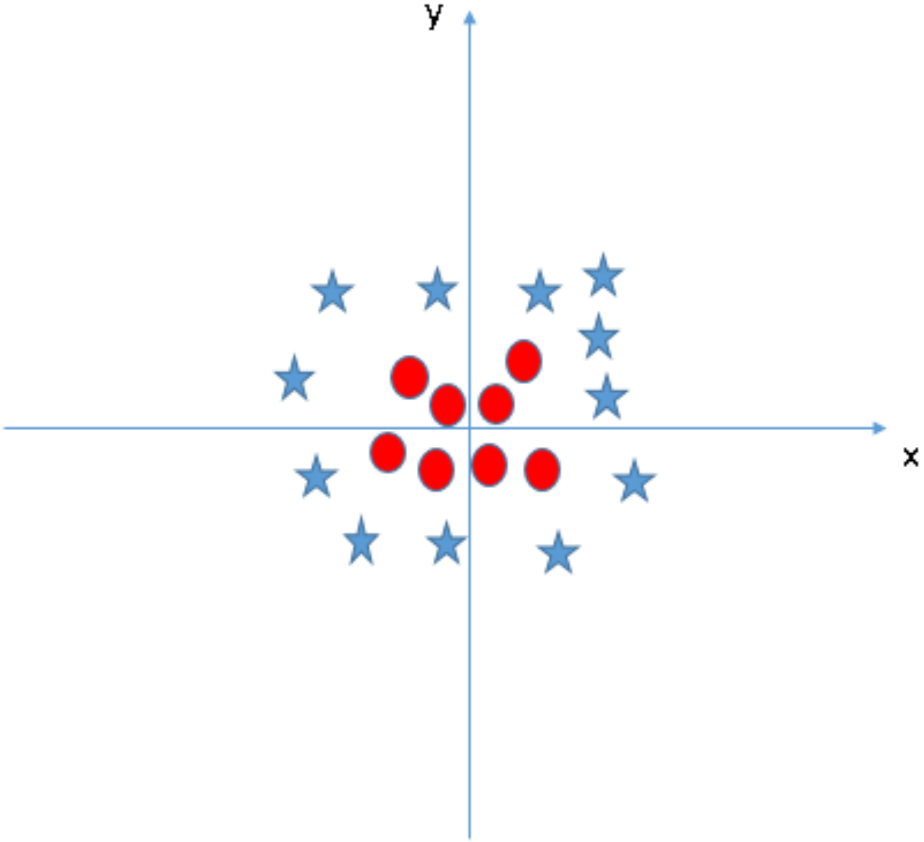


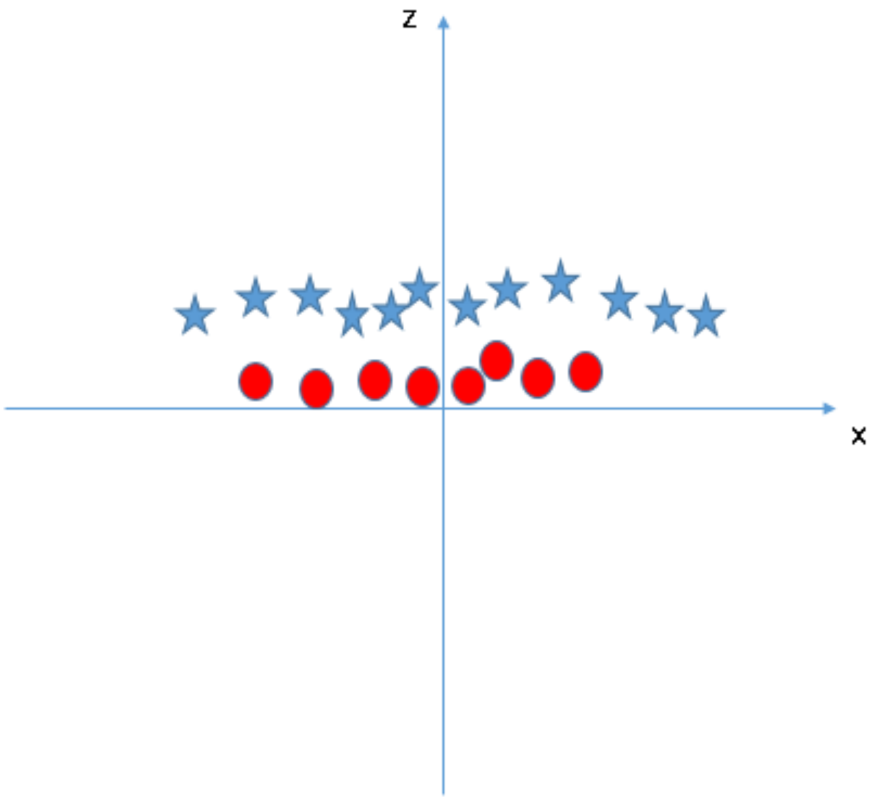
Image from Ray 2017

Harder Cases

But let's define the function z, such that:

$z(x,y) = x^2 + y^2$

Look at the resulting plot of the x and z axes.



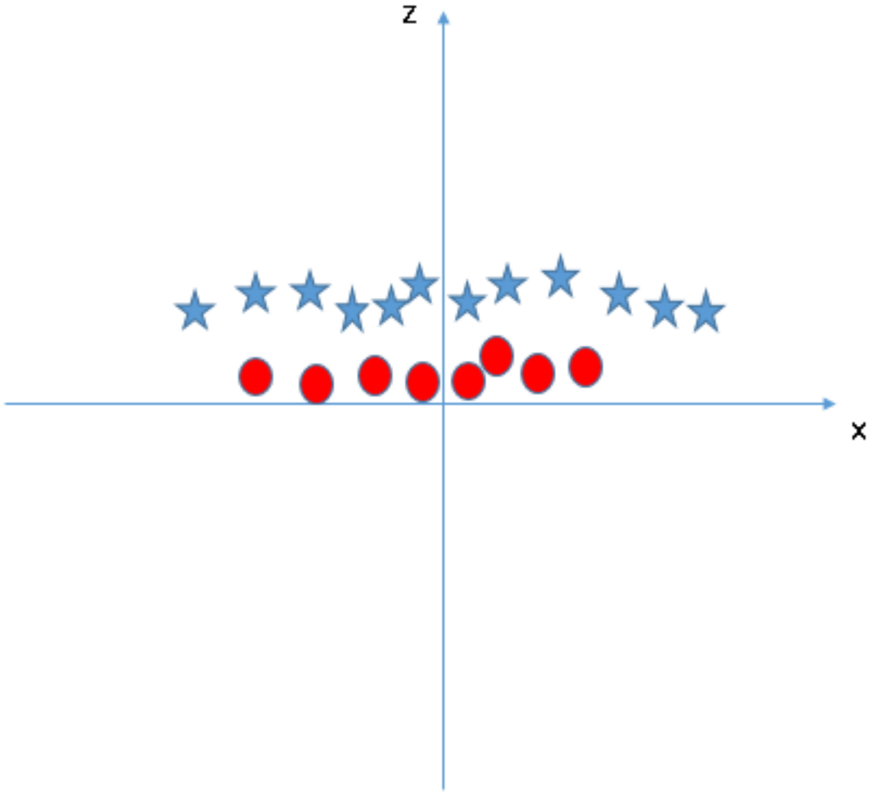
What happened? Since negative numbers become positive when squared, voila! Linear-separable data!

Image from Ray 2017

The Kernel Trick

$z = x^2 + y^2$

Look at the resulting plot of the x and z axes.



SVM has a library of functions such as this, called kernels, that are used to transform data. Doing so is cutesily called “the kernel trick.”

Terminology Again

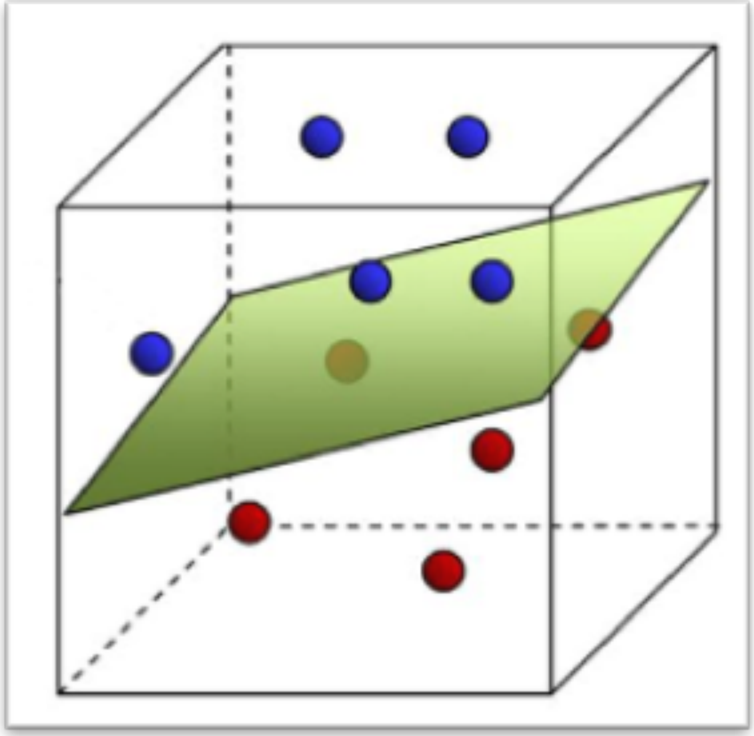
Q: Why do we have to call the separation between classes a “hyperplane”? Why not just call it a line?

Terminology Again

Q: Why do we have to call the separation between classes a “hyperplane”? Why not just call it a line?

A: In a simple 2D space, it was just a line.

But in a 3D space, we need a *plane* in order to divide the space in two.



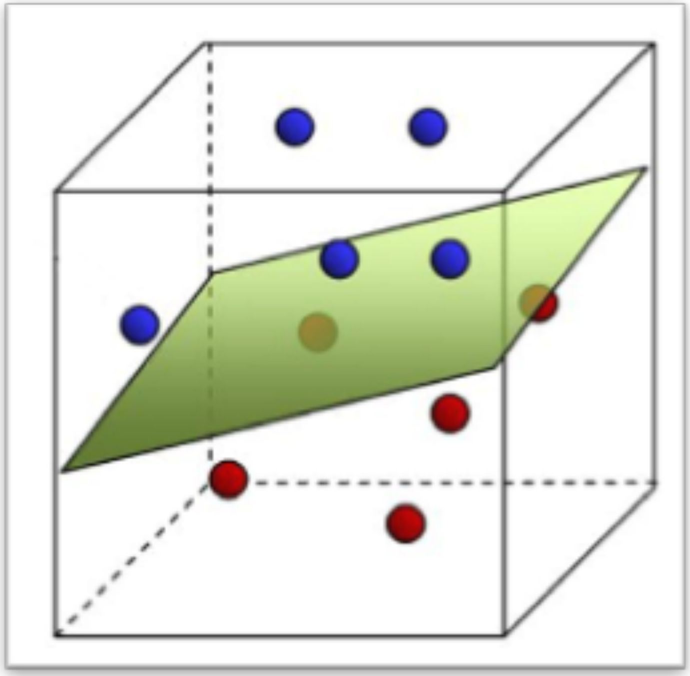
Terminology Again

Q: Why do we have to call the separation between classes a "hyperplane"? Why not just call it a line?

A: In a simple 2D space, it is just a line.

But in a 3D space, we need a *plane* in order to divide the space in two.

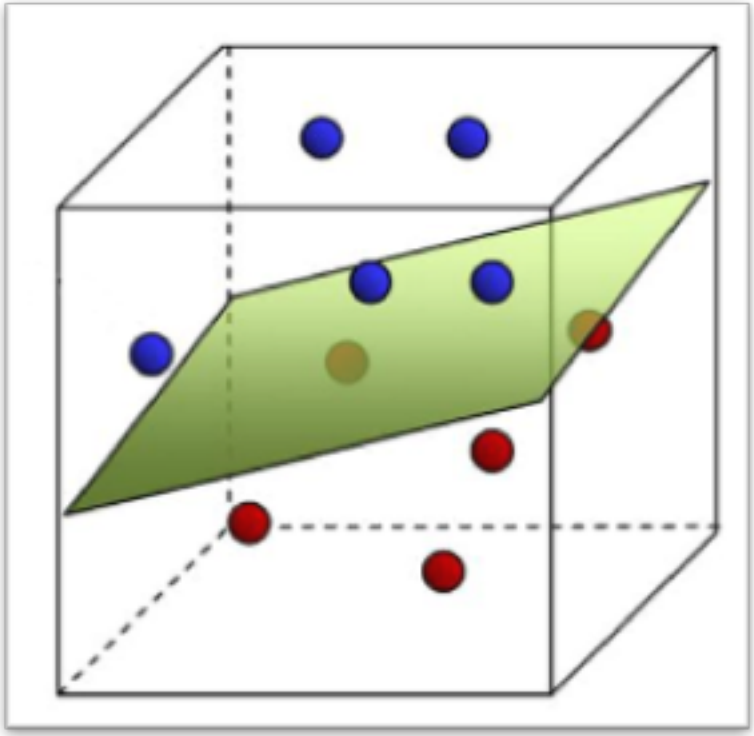
Think about the pattern: Where *n* is the number of dimensions in our data, *n*-1 is the number of dimensions our class separator needs to have.



Terminology Again

Q: Why do we have to call the separation between classes a "hyperplane"? Why not just call it a line?

A: Where *n* is the number of dimensions in our data, *n*-1 is the number of dimensions our class-separator needs to have.



So when we go beyond three dimensions, we need to go beyond a plane, i.e., a "hyperplane."

DataScience@SMU

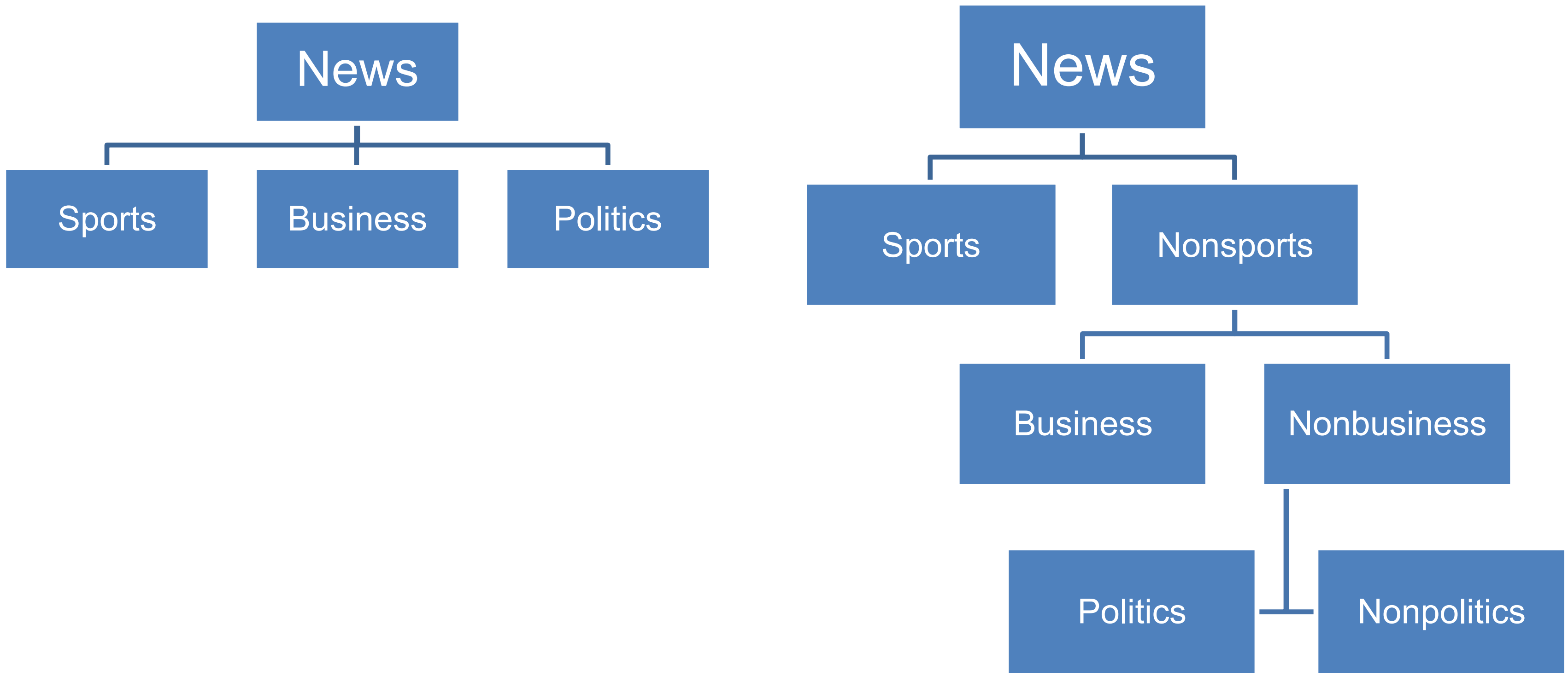
Architecting a Classification System

Multiclass with SVM

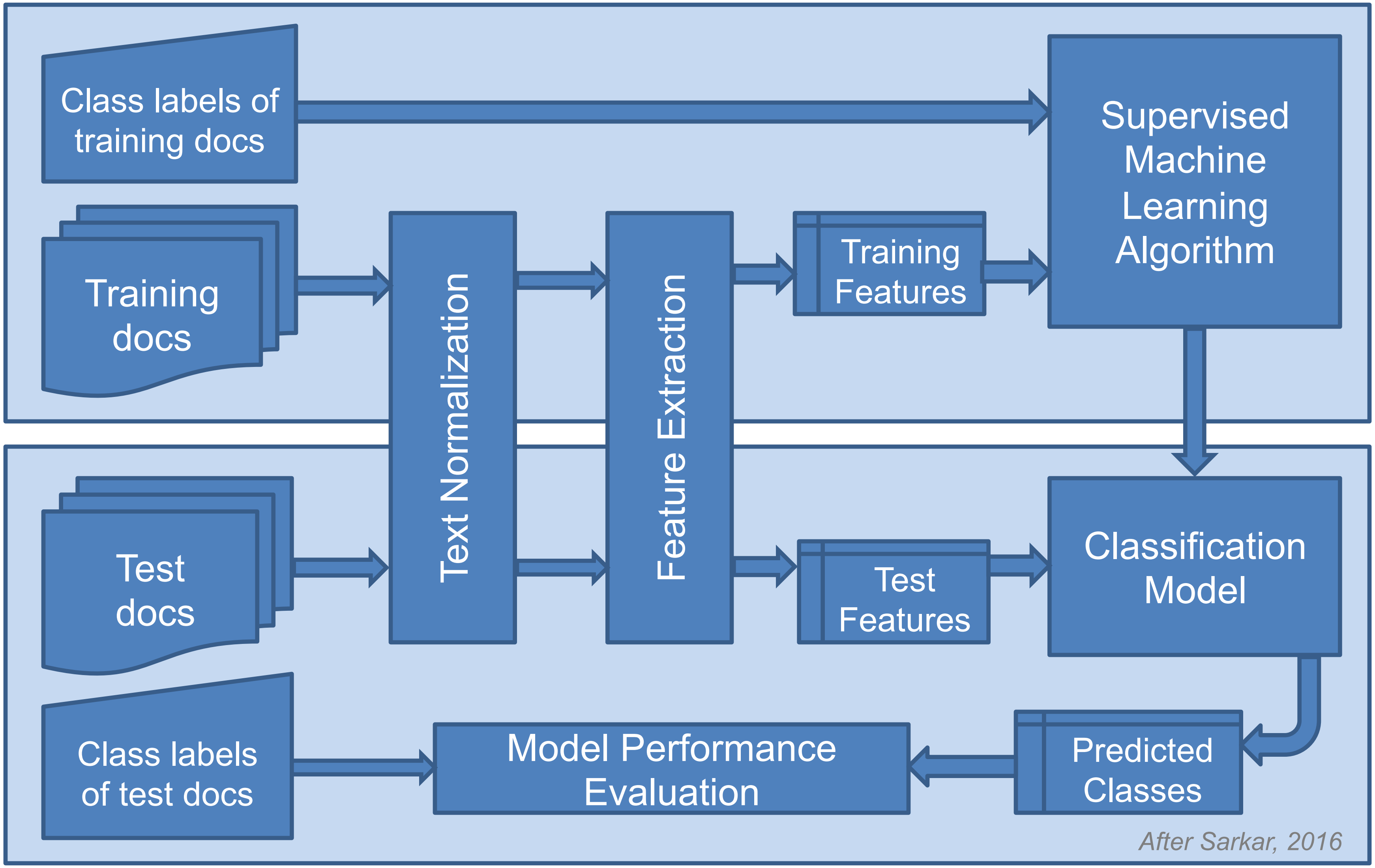
- An SVM only looks at a binary classification.
- But we often need far more than binary classification.
  - We can map any multiclassification problem to a series or binary classification tasks.
  - It means we train a plurality of SVMs and organize them into a system.

Using Binary Classifiers for a Multiclass Problem

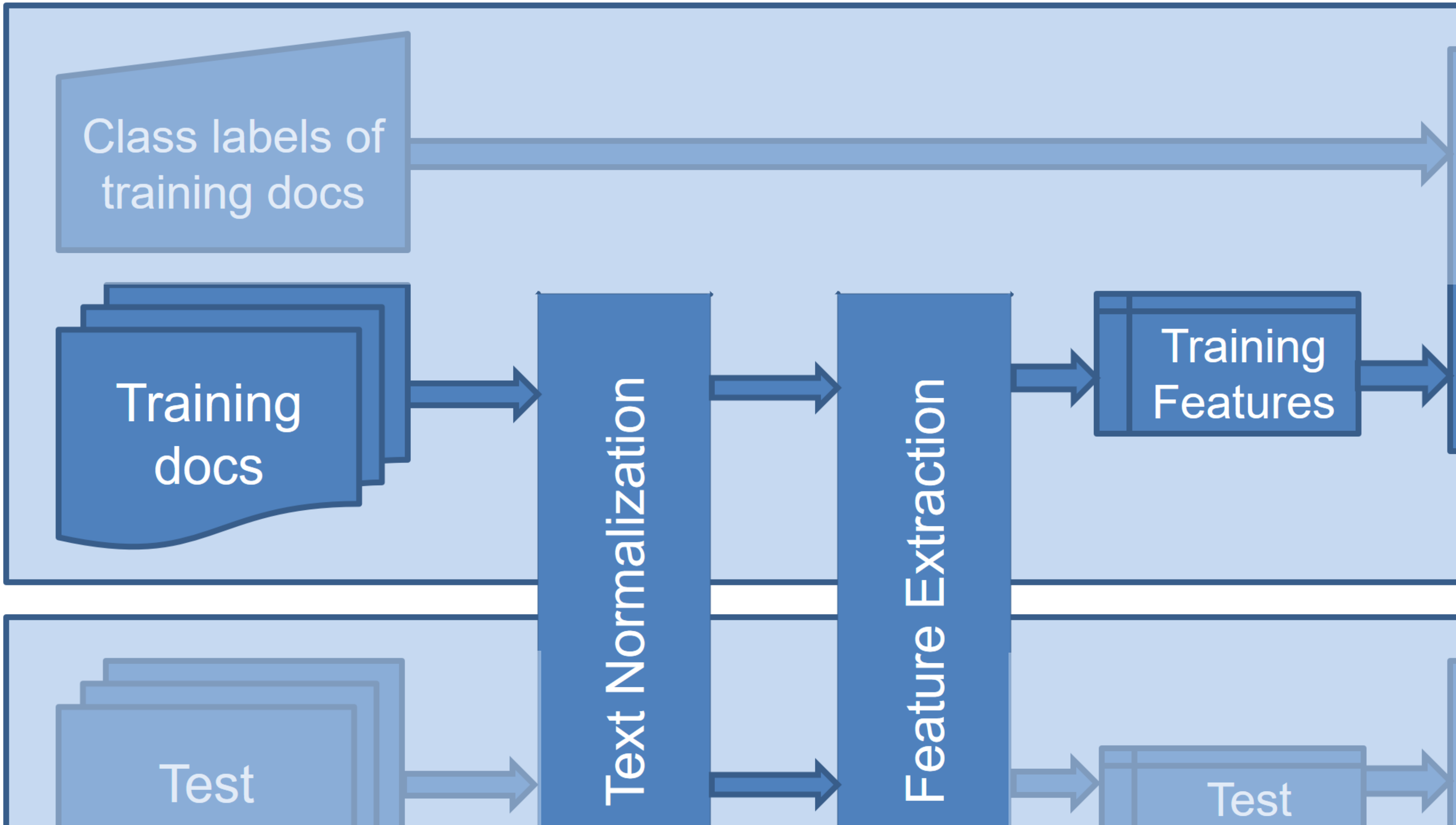
This… …becomes this.



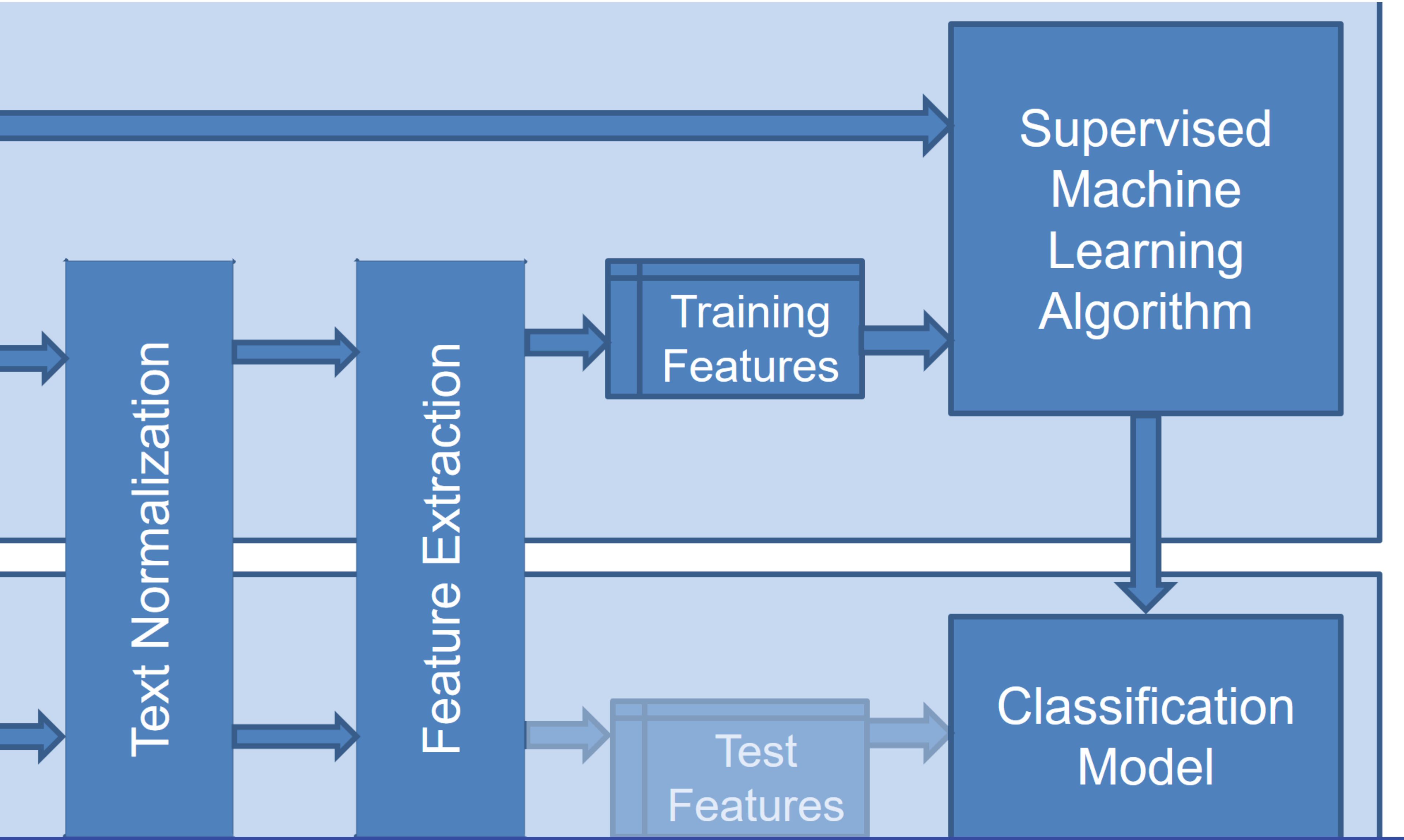
Content-Based Document Classification "Blueprint"



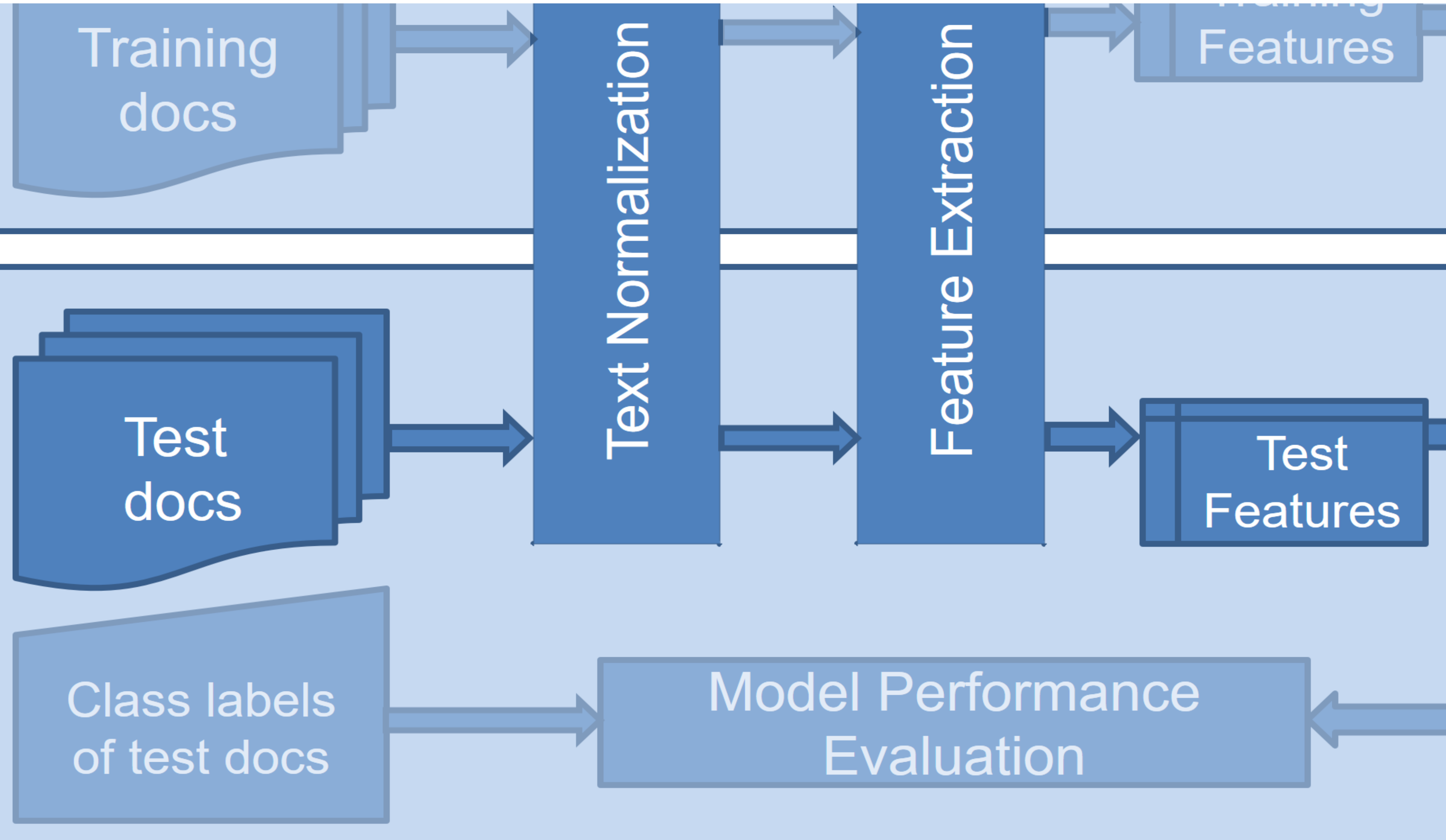
Content-Based Document Classification "Blueprint"



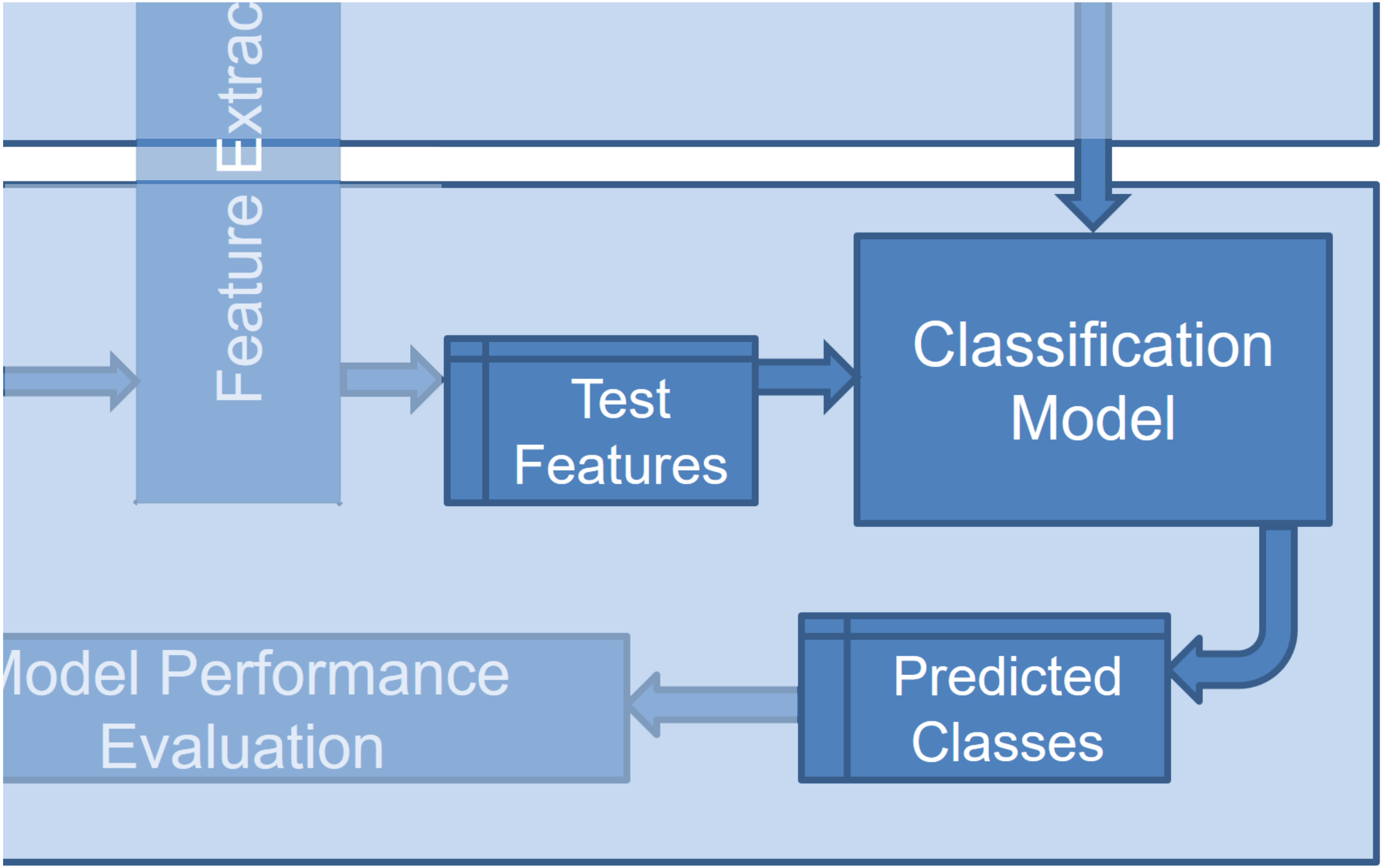
Content-Based Document Classification "Blueprint"



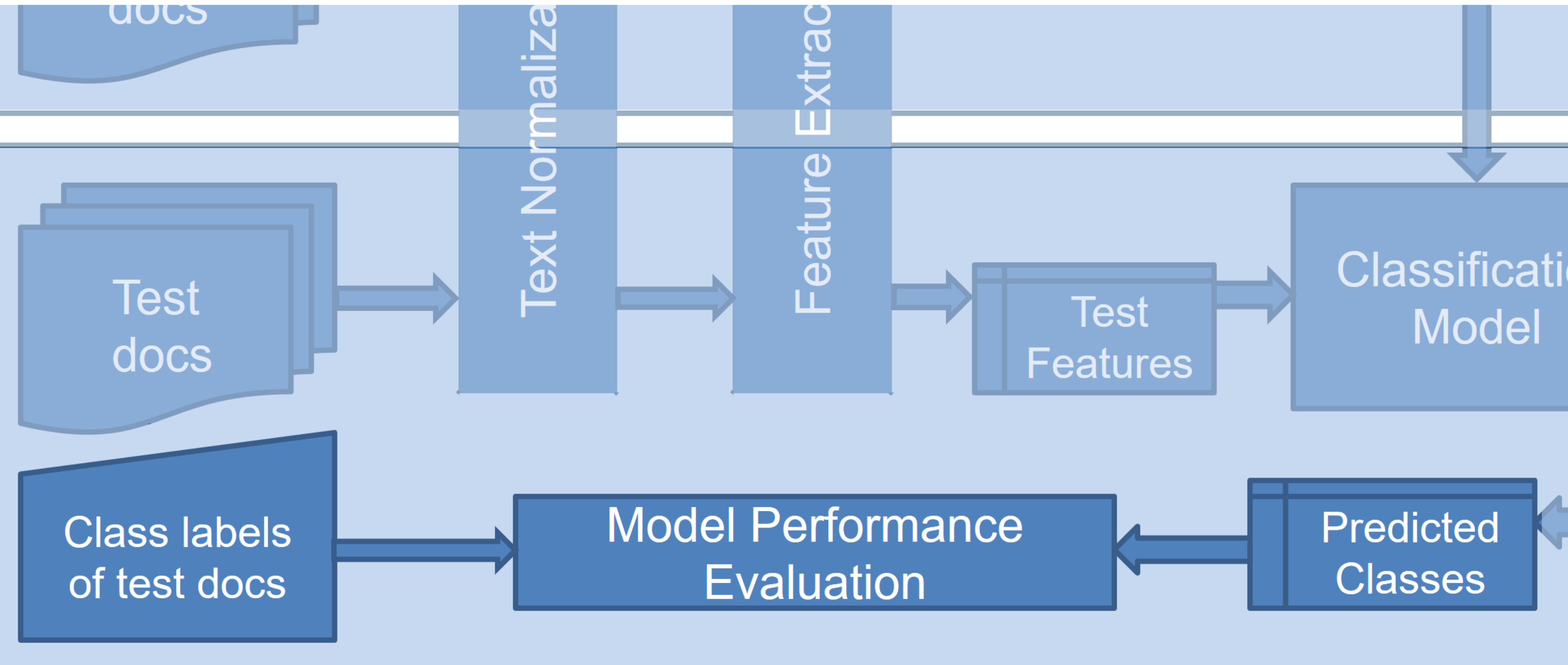
Content-Based Document Classification "Blueprint"



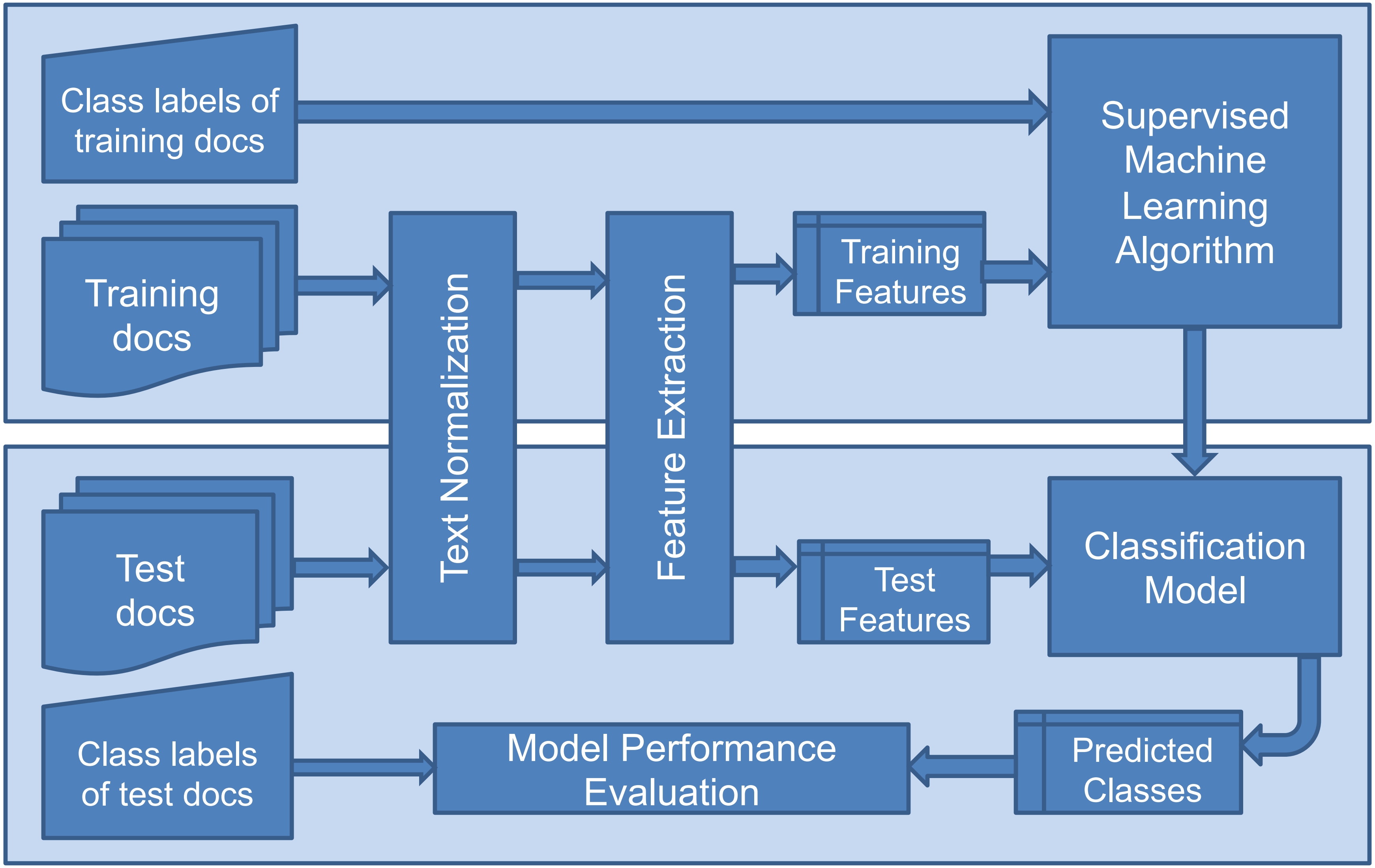
Content-Based Document Classification "Blueprint"



Content-Based Document Classification "Blueprint"



Content-Based Document Classification “Blueprint”

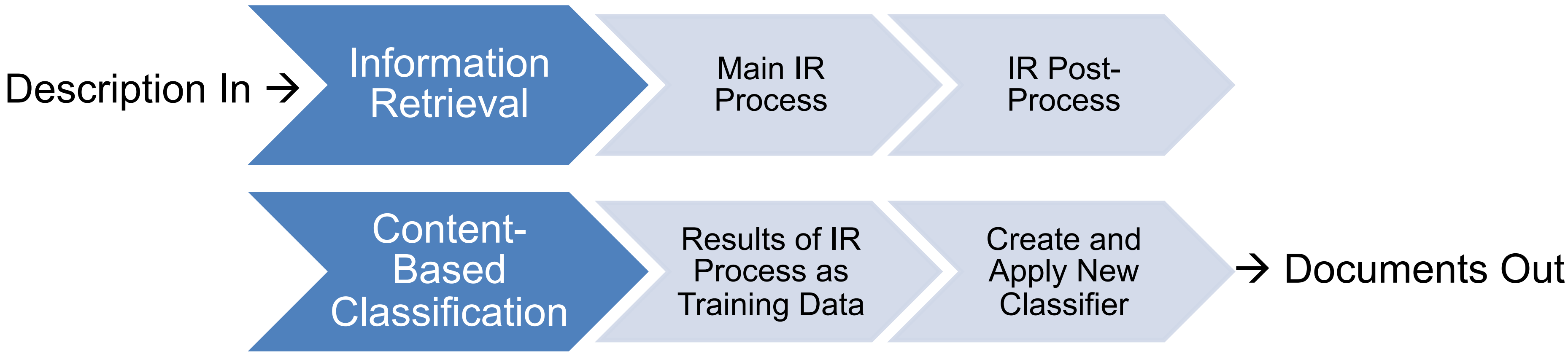


DataScience@SMU

Descriptor-Based Classifiers

Descriptor-Based Text Classification

One approach to descriptor-based text classification is to construe it as a two-phase process:



So we can treat the description-based classification task just as if it were content based, after we first fetch content that is a strong match to the description. The IR postprocess is where we separate the strong matches from weaker ones.

Example Description-Based Classifier

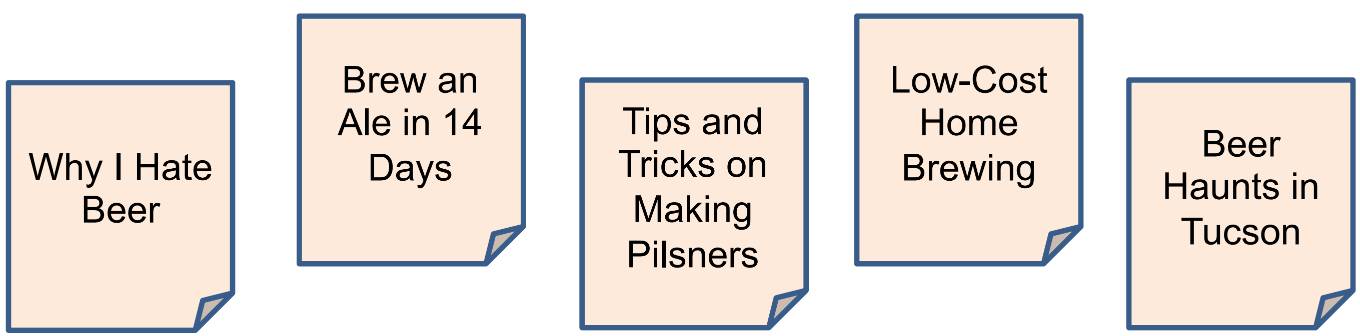
"Instruction on the brewing of beer, including lagers, ales, and barley wine; selecting and assembling the equipment and supplies needed for brewing, such as fermentation buckets, grains, hops and yeast; recipes for crafting various types of beer."

1. User inputs a robust description of the desired class of documents.

Example Description-Based Classifier

"Instruction on the brewing of beer, including lagers, ales, and barley wine; selecting and assembling the equipment and supplies needed for brewing, such as fermentation buckets, grains, hops and yeast; recipes for crafting various types of beer."

1. User inputs a robust description of the desired class of documents.
2. Our IR engine retrieves candidate documents based on keyword search from the description.



Example Description-Based Classifier

1. User inputs a robust description of the desired class of documents.
2. Our IR engine retrieves candidate documents based on keyword search from the description.
3. Our IR postprocessing identifies *strong hits* to the description and eliminates the rest.

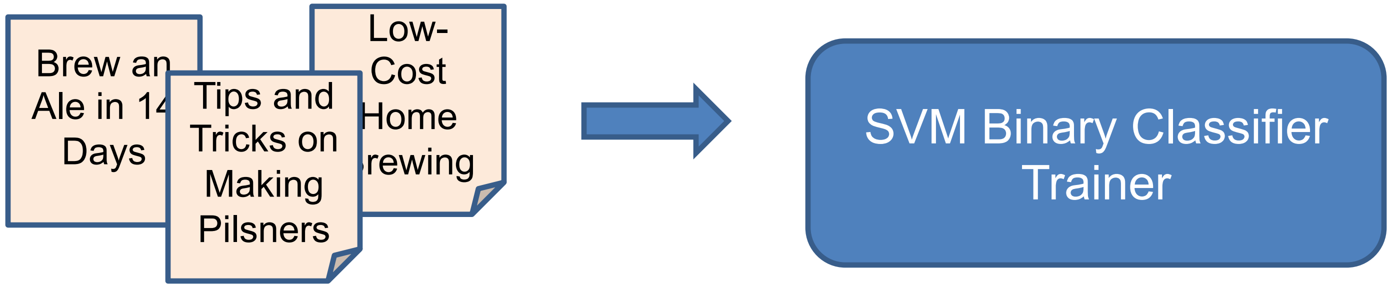
Example Description-Based Classifier

1. User inputs a robust description of the desired class of documents.
2. Our … based on k…
3. Our … the desc…

What is a *strong hit?* There is no standard definition, but the more of these features, the stronger the hit:

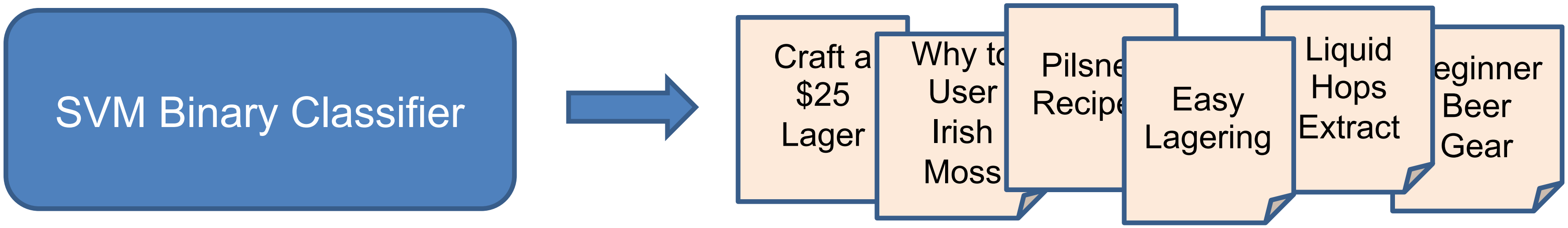
- TF-IDF of keywords found
- Phrase match
- Keyword sentence density of document
- Proximity of keywords
- Keywords found in title, subtitle, URL, filenames, image captions or alt tags, metadata keywords, hyperlinks, breadcrumb trail, etc.

Example Description-Based Classifier



1. User inputs a robust description of the desired class of documents.
2. Our IR engine retrieves candidate documents based on keyword search from the description.
3. Our IR postprocessing identifies *strong hits* to the description.
4. Our classifier trainer uses *only the strong hit* documents as training data.

Example Description-Based Classifier



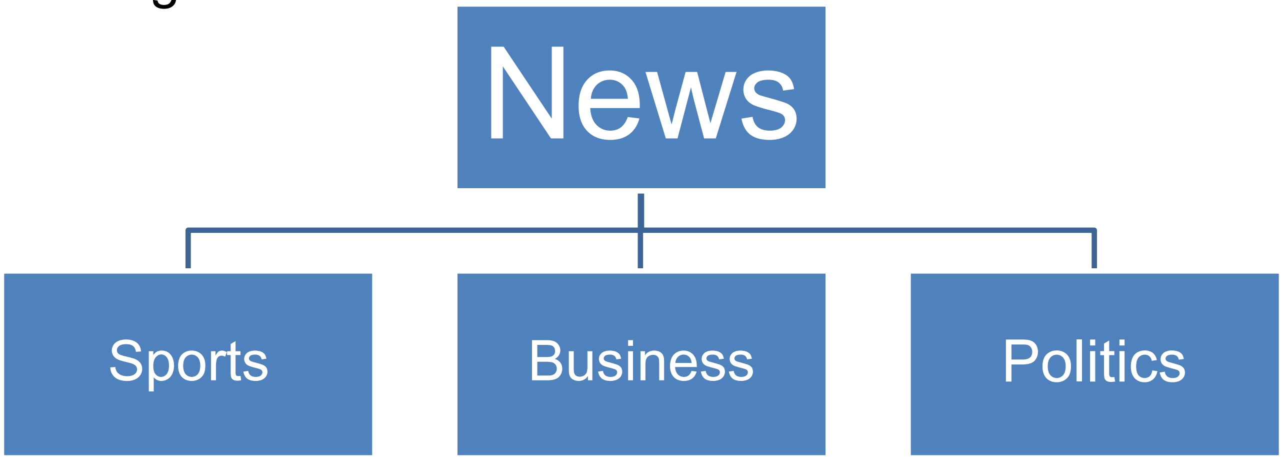
1. User inputs a robust description of the desired class of documents.
2. Our IR engine retrieves candidate documents based on keyword search from the description.
3. Our IR postprocessing identifies *strong hits* to the description.
4. Our classifier trainer uses *only the strong hit* documents as training data.
5. We build a binary content-based classifier and use it to gather more documents.

DataScience@SMU

Using Taxonomy Labels for Descriptor-Based Classifiers

Another Example: Empty Taxonomy as Input

Input is just a taxonomy? We can devise Boolean query strings to search for strong hits.



This taxonomy yields three Boolean queries:

Sports & ~(Business|Politics)

Business & ~(Sports|Politics)

Politics & ~(Sports|Business)

Other Examples

Input is just a taxonomy? If we can form disjunctive sets of equivalent terms (remember synonyms and hyponyms from WordNet?) for each node, we can make better queries.

This taxonomy + WordNet yields three queries:

(Sports|Athletics) & ~(Business|Finance|Investing|Politics|Government)

(Business|Finance|Investing) & ~(Sports|Athletics|Politics|Government)

(Politics|Government) & ~(Sports|Athletics|Business|Finance|Investing)

Why It Works

1. Not every sports document says “sports” or “athletics,” and not every business document says “business” or “finance.”

Why It Works

1. Not every sports document says “sports” or “athletics,” and not every business document says “business” or “finance.”
2. But if *enough* of them do, then we can still gather enough data to train a workable classifier.

Why It Works

1. Not every sports document says “sports” or “athletics,” and not every business document says “business” or “finance.”
2. But if *enough* of them do, then we can still gather enough data to train a workable classifier.
3. And if there’s not enough, we just need to get better sets of equivalent terms, e.g., add all the hyponyms of “sports” (there are a lot!), e.g., cycling, skiing, funambulism, and *hundreds* more.

Why It Works

1. Not every sports document says “sports” or “athletics,” and not every business document says “business” or “finance.”
2. But if *enough* of them do, then we can still gather enough data to train a workable classifier.
3. And if there’s not enough, we just need to get better sets of equivalent terms, e.g., add all the hyponyms of “sports” (there are a lot!), e.g., cycling, skiing, funambulism, and *hundreds* more.
4. A broadly expanded query might get an occasionally bad hit (e.g., “casting” is a hyponym of “sports”), but as long as the preponderance of hits are good hits, we’ll be OK since most classifiers (such as both SVM and MNB) can tolerate outliers.

How It Can Break

In short, it works about as well as the taxonomy fits the content.

If we use the Sports-Business-Politics taxonomy and our corpus is mostly cooking recipes, we can expect bad results!

DataScience@SMU