Errors & Fairness in Machine Learning

S. Verma

Machine Learning - Errors

- Bayes' error
  - how much error is there in the best possible classifier
- Approximation error
  - What is the cost of restricting to some hypothesis class
- Estimation error
  - how much do you pay because you only have finite samples
- Optimization error
  - how much do you pay because you didn't find a global optimum to your optimization problem

Steps - debugging strategy

- First, ensure that the optimizer is not the problem. This can be done by adding "cheating" features
  - a feature that correlates perfectly with the label.

- Make sure you can successfully overfit the training data.

If not, this is probably either an optimizer problem or a too-small-sample problem.

Steps - debugging strategy

- Remove all the features except the cheating feature and make sure you can overfit then. Assuming that works, add feature back in incrementally (usually at an exponential rate).

If at some point, things stop working, then probably you have too many features or too little data.

Steps - debugging strategy

- Remove the cheating features and make your hypothesis class much bigger; e.g., by adding lots of quadratic features.

Make sure you can overfit.

If you can't overfit, maybe you need a better hypothesis class.

Steps - debugging strategy

- Cut the amount of training data in half.
- We usually see test accuracy asymptote as the training data size increases,

if cutting the training data in half has a huge effect, you're not yet asymptoted and you might do better to get some more data.

Problem of this normal breakdown

- These error terms come from theory
  - sometimes theory misses out on something because of a particular abstraction that has been taken.

- This abstraction is due to the fact that the overall goal has already been broken down into an iid/PAC style learning problem
  - Hence, some types of error are hidden because the abstraction hides them.

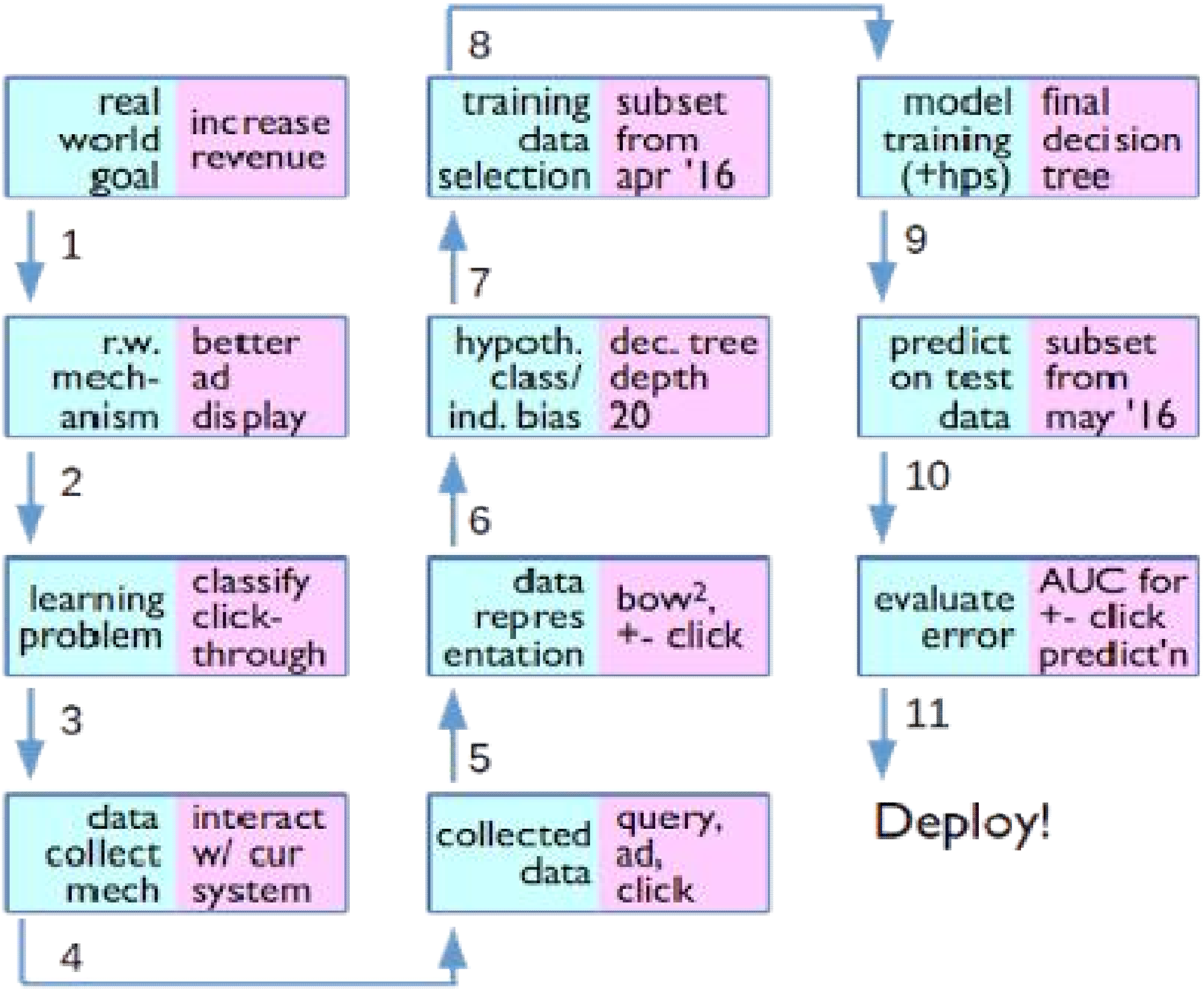
1st Step

- Real World Goal - increasing revenue for our company
- Proposed Solution - *Improve our ad displays.*

- Consequence
- This immediately upper bounds how much increased revenue we can hope for because –
  - *maybe ads are the wrong thing to target.*
  - *Maybe I would do better by building a better product.*

- This is sort of a "business" decision, but it's perhaps the most important question you can ask:
  - *am I even going after the right things?*

Errors in Machine Learning



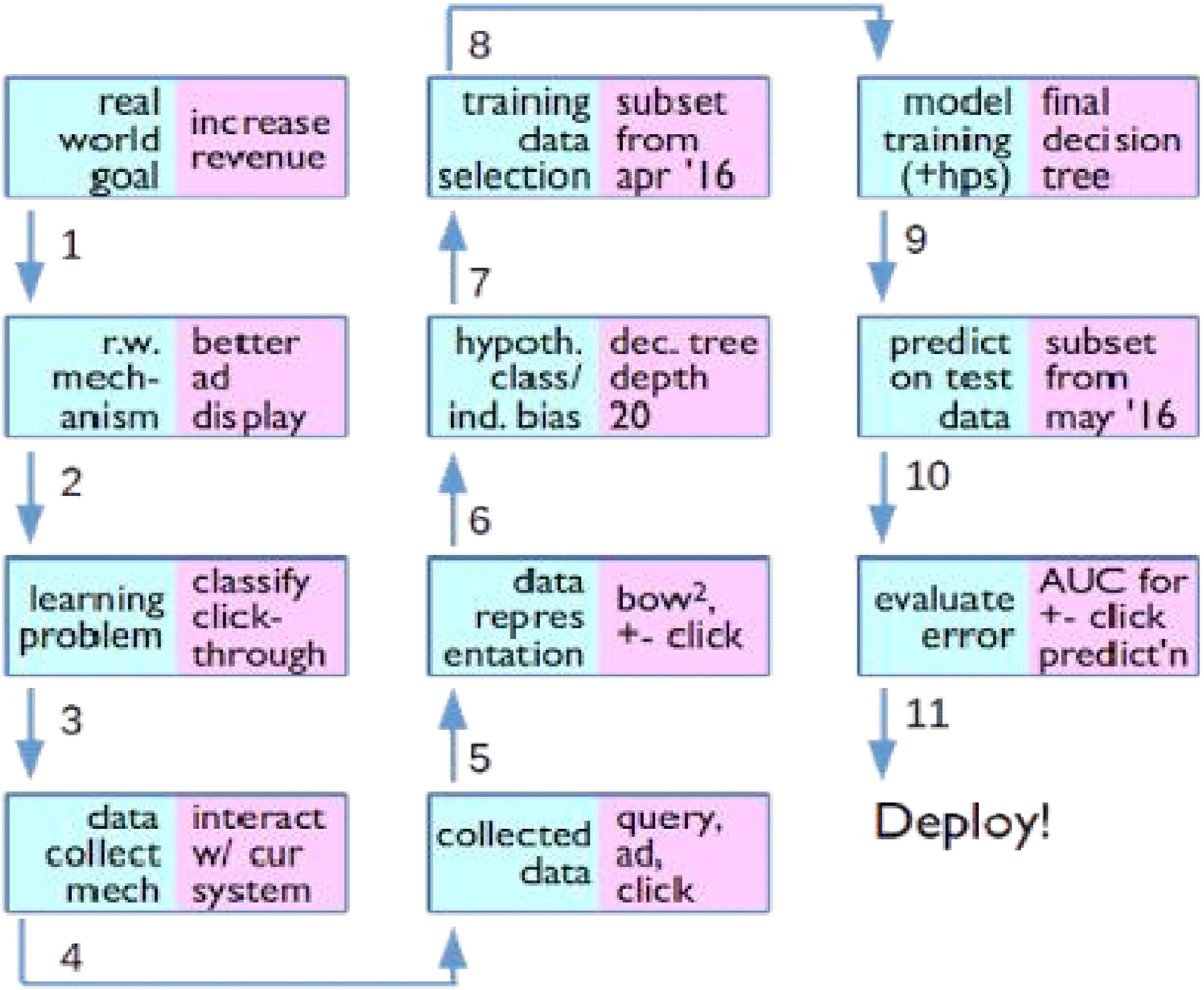
2nd Step

- Goal- better ad placement
- Proposed Solution - *turn it into a learning problem (say) by*
  - *trying to predict clickthrough, and*
  - *Use those predictions to place ads. better*

- Is clickthrough a good thing to use to predict increased revenue?
  - *This itself is an active research area.*

- Consequence of clickthrough prediction
  - *suffer some loss because of a mismatch between that prediction task and the goal of better ad placement.*

Errors in Machine Learning



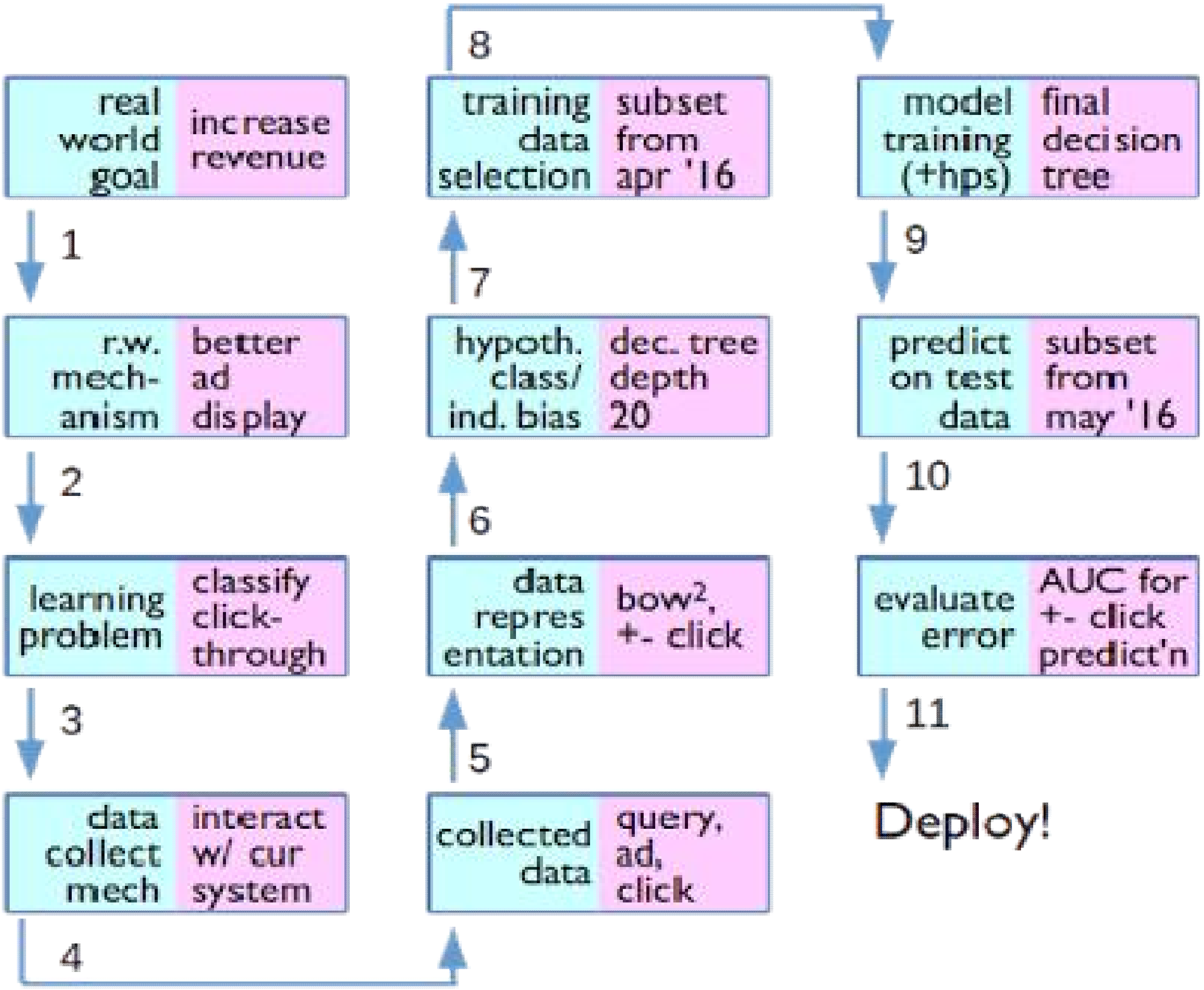
3rd Step

- Goal- collect some data.
- Proposed Solution - *do this by logging interactions with a currently deployed system.*

- Consequence
- all sorts of biases are introduced
  - *because the collected data is not from the final system to be build and deployed,*
  - *pay for this in terms of distribution drift.*

•

Errors in Machine Learning



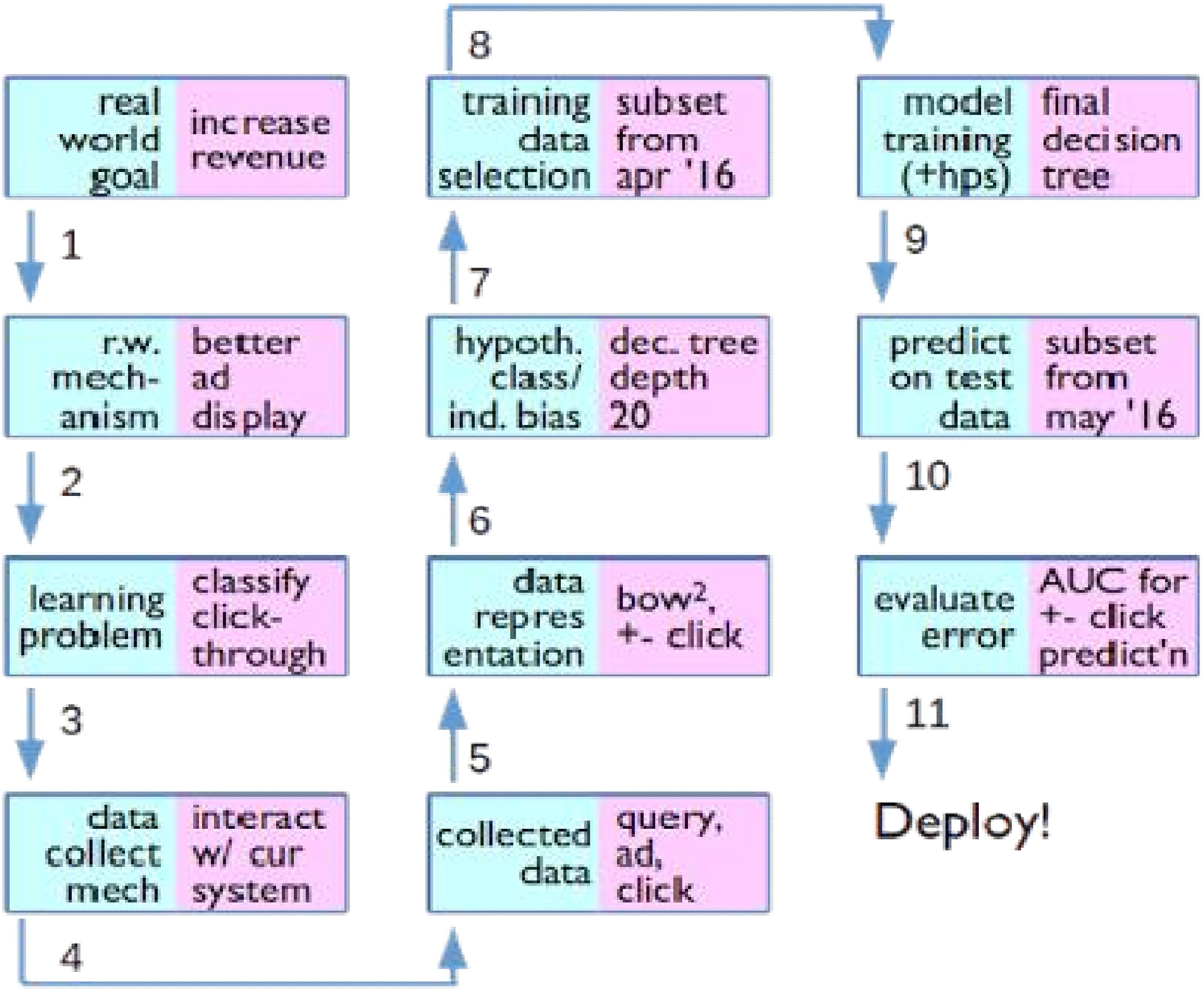
4th Step

- Problem: Cannot possibly log everything that the current system is doing
- Solution: *only log a subset of things. (say, log queries, ads, and clicks)*

- Consequence
- hides any information not logged
  - *for instance, time of day or day of week might be relevant, user information might be relevant, etc.*

- *This again upper bounds the best possible revenue.*

Errors in Machine Learning



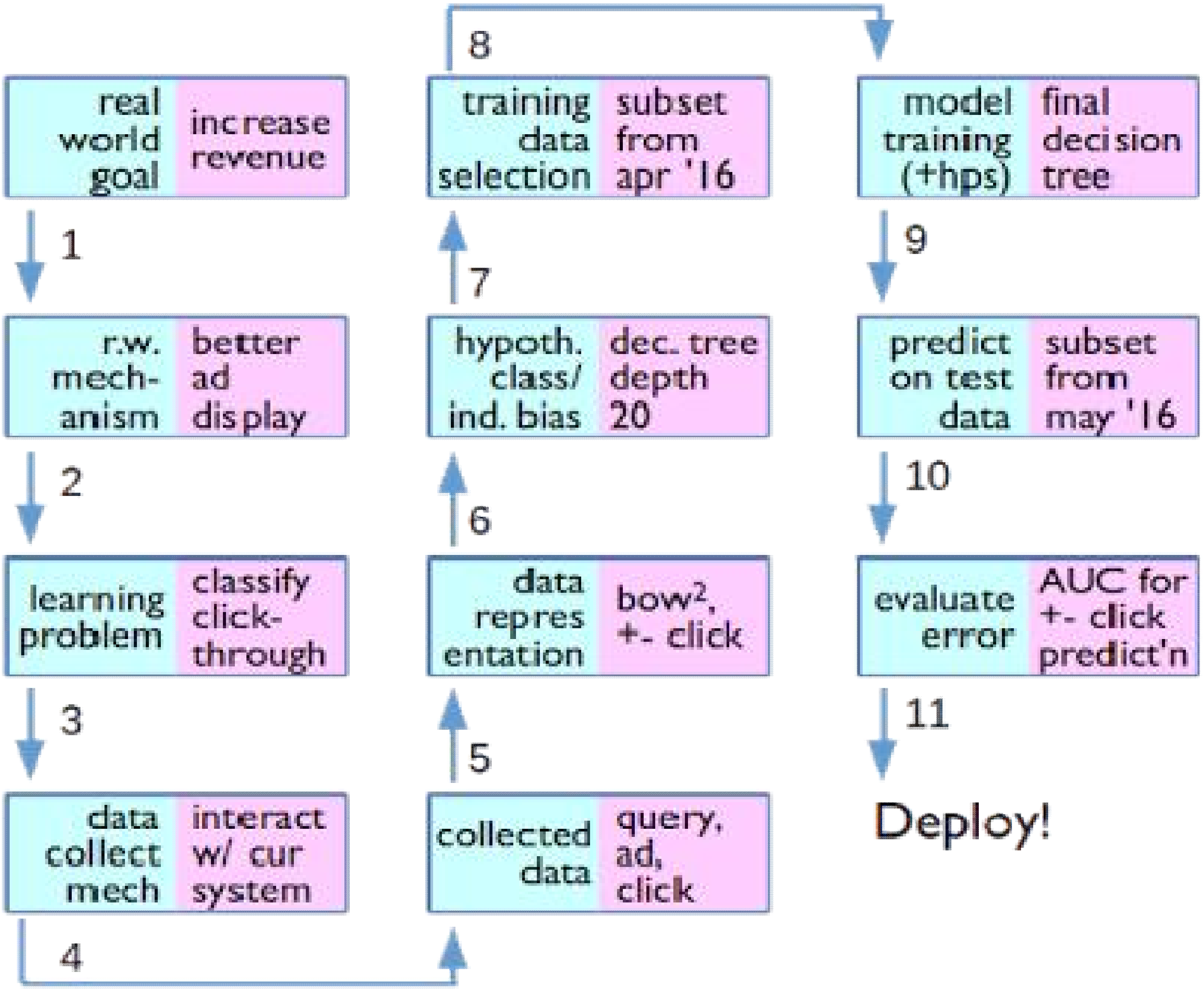
5th Step

- Action: Pick a data representation
  - *for instance quadratic terms between a bag of words on the query side and a bag of words on the ad side, paired with a +/- on whether the user clicked or not.*

- Consequence
  - *Use of theory words basically limits the best possible Bayes error.*

- *Inclusion of more information or better representation may result in a lower Bayes error.*

Errors in Machine Learning

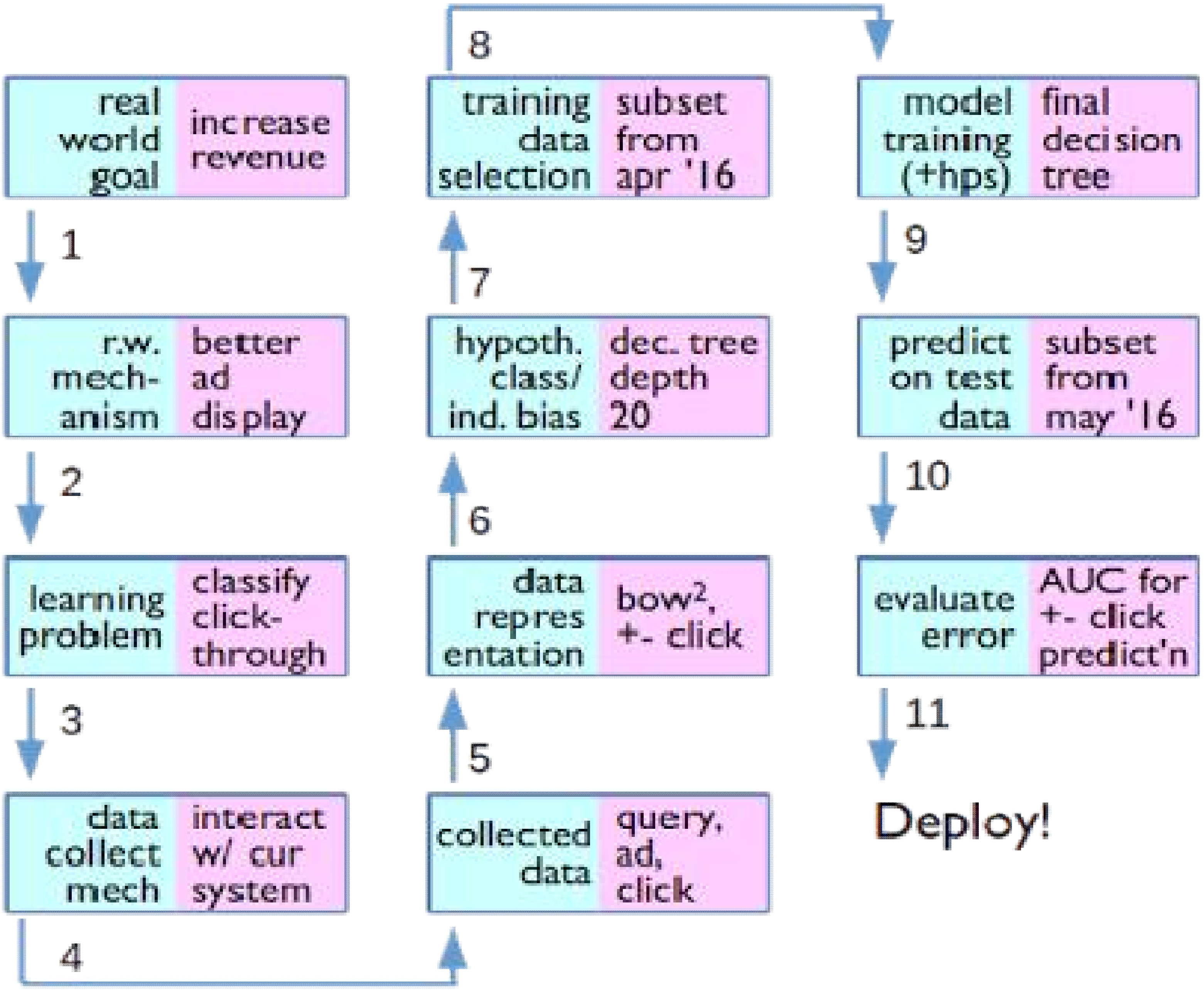


6th Step

- Choose a hypothesis class
  - Choose decision trees.

- *This is where the approximation error comes from.*

Errors in Machine Learning

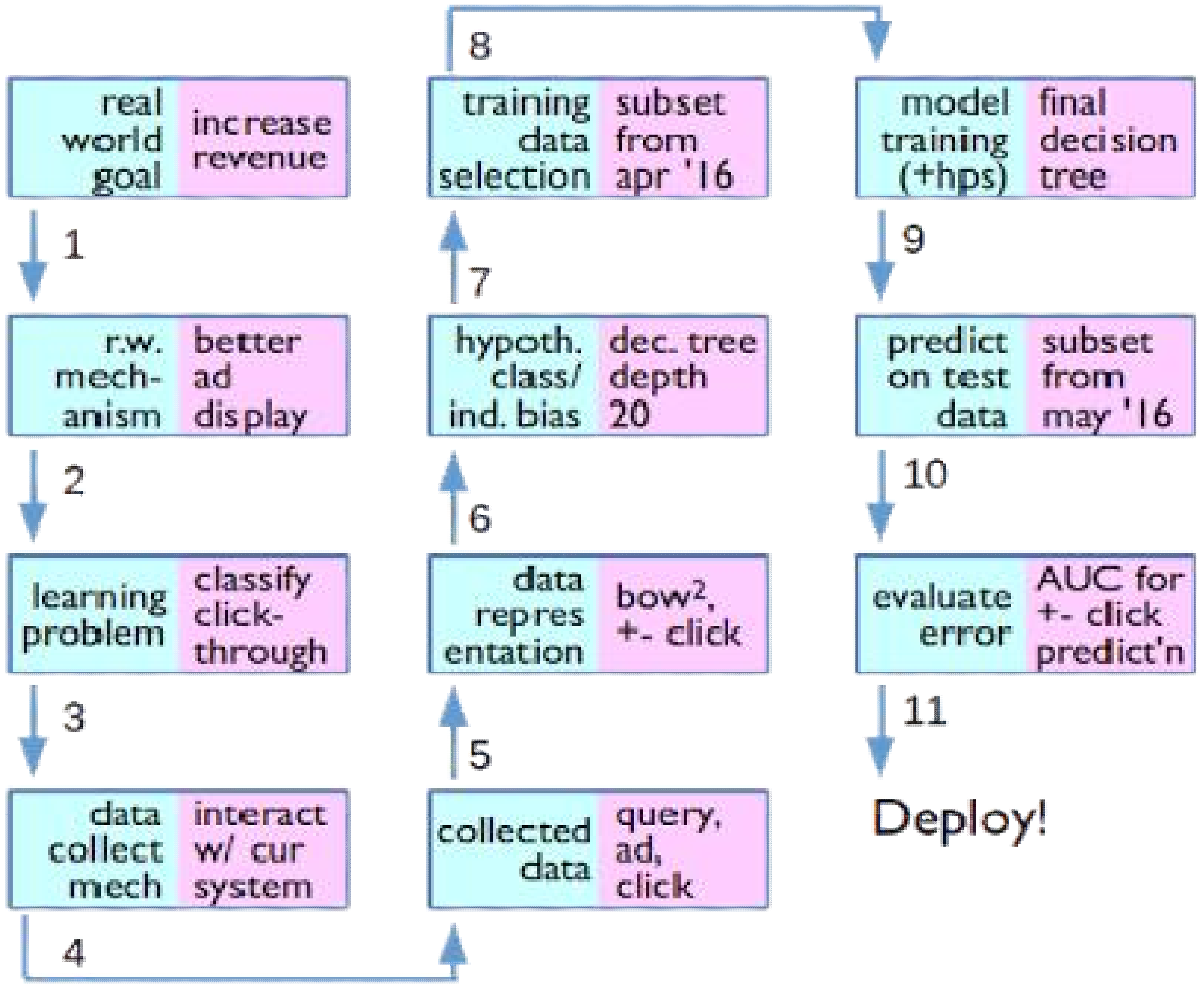


7th Step

- Goal: pick some training data.

- Consequence:
  - *The real world is basically never i.i.d., so any selected data is going to have some bias.*
  - *It might not be identically distributed with the test data (because things change month to month, for instance).*
  - *It might not be independent (because things don't change much second to second). You will pay for this.*

Errors in Machine Learning

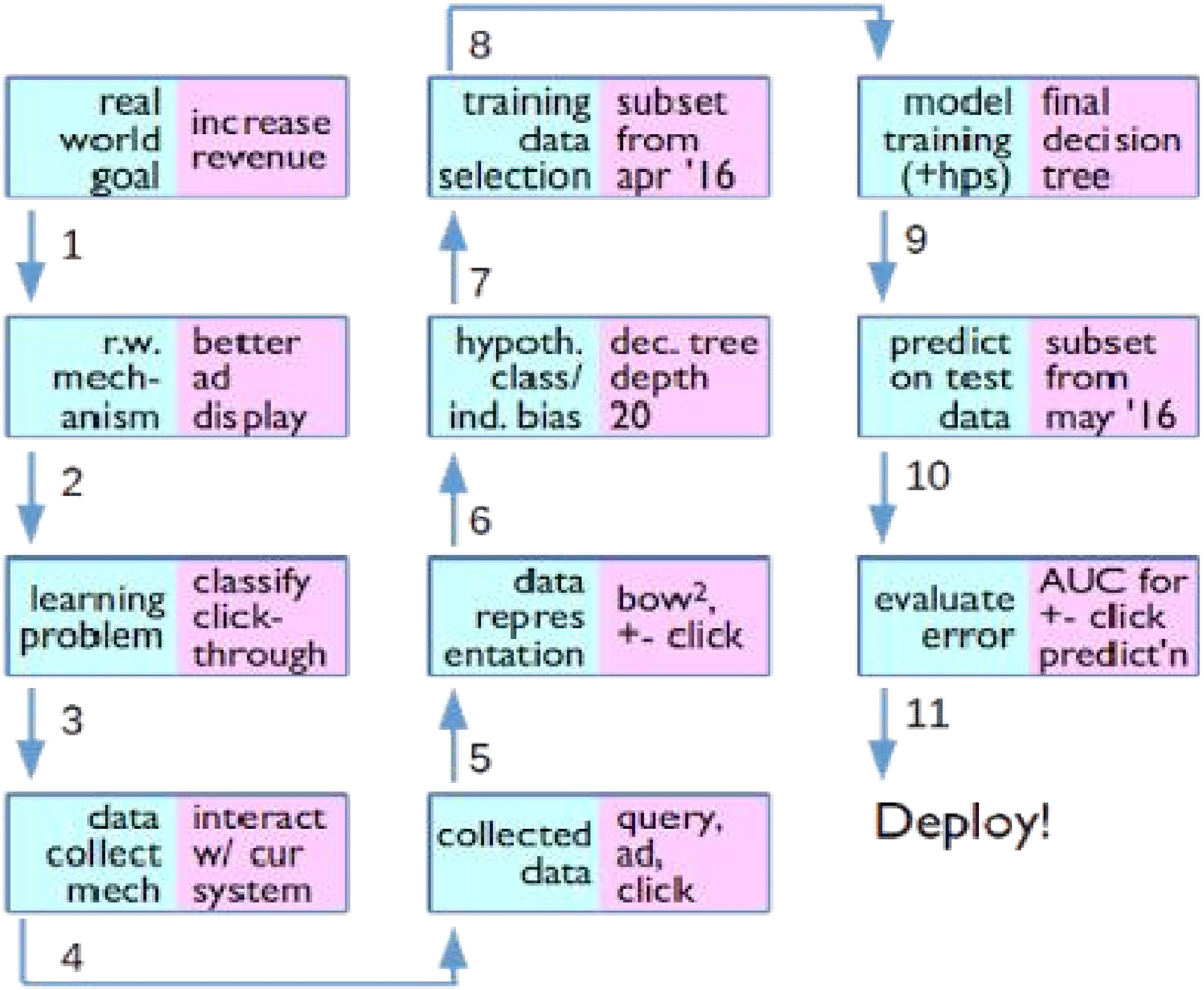


8th Step

- Goal: train the model on this data (probably tuning hyperparameters too).

- This is the usual estimation error.

Errors in Machine Learning



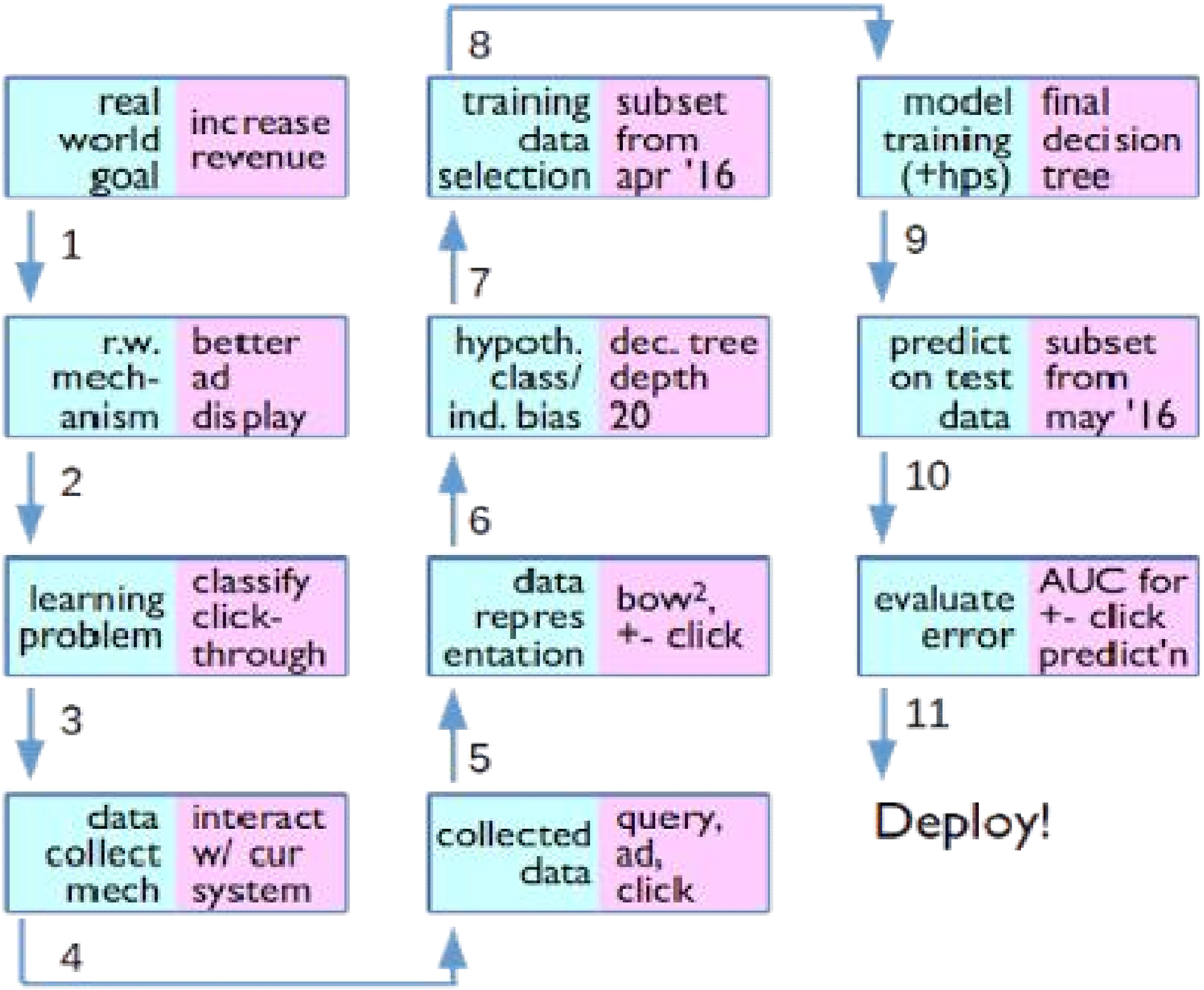
9th Step

- Goal: Pick some test data on which to measure performance.

- this test data is only going to be representative of how well the system will do in the future if this data is so representative.

- In practice, it won't be, typically at least because of concept drift over time.

Errors in Machine Learning



10th Step

- After we make predictions on this test data, we have to choose some method for evaluating success.

- We might use accuracy, f measure, area under the ROC curve, etc.

- The degree to which these measures correlate with what we really care about (ad revenue) is going to affect how well we're able to capture the overall task.

- If the measure anti-correlates, for instance, we'll head downhill rather than uphill.

Good Features

- Image: Pixel representation, Patch Representation

- Text categorization: Bag of words Representation
  - *throw away* all locality information in the image/text
  - Learning algorithms don't care about features: they only care about feature values (permutation does not effect feature values)

Irrelevant and Redundant Features

- One difference between learning models is Robustness to the addition of noisy or irrelevant features.

- Irrelevant Feature:
  - A feature f whose expectation does not depend on the label $E[f \mid Y] = E[f]$ might be irrelevant.

- Redundant Feature:
  - Two features are redundant if they are highly correlated, regardless of whether they are correlated with the task or not.

Decision Trees

- The model explicitly selects features that are highly correlated with the label.

- by limiting the depth of the decision tree, one can at least *hope* that the model will be able to throw away irrelevant features.

- Redundant features are almost certainly thrown out:
  - once you select one feature, the second feature now looks mostly useless.

- The only possible issue with irrelevant features is that even though they are irrelevant, they *happen to* correlate with the class label on the training data, but chance.

  - Perfect Correlation
  - Partial Correlation

K-nearest neighbors

- KNN weighs each feature just as much as another feature
- The introduction of irrelevant features can completely mess up KNN prediction
- In high dimensional space, randomly distributed points all look approximately the same distance apart

Perceptron

- It might learn to assign zero weight to irrelevant features. For instance
  - consider a binary feature is randomly one or zero independent of the label.
  - If the perceptron makes just as many updates for positive examples as for negative examples,
  - there is a reasonable chance this feature weight will be zero/very small.

Feature Pruning and Normalization

- Very low frequency of occurrence of a feature during training
- Keep or reject this as a feature?
  - It is hard to tell with just one training example if it is really correlated with the positive class, or is it just noise

- Feature pruning is important
- Binary features – easy
  - reject a feature if it only appears some small number *K* times

- Initially pruning does not hurt
  - but eventually interesting features may be deleted.

Feature Normalization

- We need to normalize the data so that it is consistent in some way.
- The goal is to make it *easier* for learning algorithm to learn

Two basic types:

- ***feature normalization:*** *go through each feature and adjust it the same way across all examples*
- ***example normalization***: *each example is adjusted individually.*

Feature Normalization

- Centering: moving the entire data set so that it is centered around the origin.
- *The goal of centering is to make sure that no features are arbitrarily large.*

- Scaling: rescaling each feature so that one of the following holds:
  - Each feature has variance 1 across the training data.
  - Each feature has maximum absolute value 1 across the training data.

- *The goal of scaling is to make sure that all features have roughly the same scale*

Example - Normalization

- The most standard normalization is to ensure that the length of each example vector is one:
  - namely, each example lies somewhere on the unit hypersphere.

The main advantage of example normalization is that it makes comparisons more straightforward across data sets.

Combinatorial Feature Explosion

- Linear models (like the perceptron) cannot solve the XOR problem.
- However, by performing a combinatorial feature explosion, they can learn.
  - But that came at the computational expense of gigantic feature vectors.
- perceptron is the one that has the most to gain by feature combination.
- decision tree is the one that has the least to gain. In fact, the decision tree construction is essentially building meta features

Combinatorial Feature Explosion

- This observation leads to a heuristic for constructing meta features for perceptrons from decision trees.

- The idea is to train a decision tree on the training data.
- From that decision tree, meta features can be extracted by looking at feature combinations along branches.

- Only those feature combinations can be added as meta features to the feature set for the perceptron

Evaluating Model Performance

Accuracy, precision, recall

- In some cases, *accuracy* may *not* be the *desirable measure of performance*

- *Precision* asks the question: *of all the X's that you found, how many of them were actually X's?*
- *Recall* asks the question : *of all the X's that were out there, how many of them did you find*

- Thus, if the system found nothing, precision is always perfect; and
- if there is nothing to find, recall is always perfect.

Example - Spam

Accuracy, precision, recall

- Suppose that you are attempting to identify spam.
- You run a learning algorithm to make predictions on a test set.
- But instead of just taking a "yes/no" answer, you allow your algorithm to produce its confidence.
  - For instance, in perceptron, you might use the distance from the hyperplane as a confidence measure.
- You can then sort all of your test emails according to this ranking.
- You may put the most spam-like emails at the top and the least spam-like emails at the bottom

Accuracy, precision, recall

- Once you have this sorted list, you can choose how aggressively you want your spam filter to be by setting a threshold anywhere on this list.
- One would hope that if you set the threshold very high, you are likely to have high precision (but low recall). If you set the threshold very low, you'll have high recall (but low precision).

- By considering every possible place you could put this threshold, you can trace out a curve of precision/recall values,

Accuracy, precision, recall

- Use harmonic mean of Precision and Recall: f-score

$$f - score\ F = \frac{2FP}{F + P}$$

- When Precision is more important than recall: weighted f-measure, which is parameterized by a weight β (user who cares β times as much about precision as about recall)

$$weighted\ f - measure\ \ F_\beta = \frac{(1 + \beta^2)PR}{\beta^2 P + R}$$

Sensitivity, specificity

- A sensitive classifier is one which almost always finds everything it is looking for: it has high recall.

  sensitivity is exactly the same as recall.

- A specific classifier is one which does a good job not finding the things that it doesn't want to find.

  Specificity is precision on the negation of the task at hand.

- Sensitivity-specificity plot is referred to as the receiver operating characteristic (or ROC curve)

K-fold Cross Validation

- Break your training data up into (k=10) equally-sized partitions.
- Train a learning algorithm on 9 of them and test it on the remaining 1.
- Do this 10 times, each time holding out a different partition as the "development" part.
- average the performance over all ten part to *get an estimate of how well the model will perform in the future.*
- repeat this process for every possible choice of hyperparameters to get an estimate of which one performs best.

leave-one-out cross validation

- In the case that K = N-1, this is known as leave-one-out cross validation
  - *may be very expensive.*
  - *This is true for most learning algorithms except for K-nearest neighbors. For KNN, leave-one-out is actually very natural.*
- After running cross validation, there are two choices.
  - either select one of the K trained models as the final model to make predictions with, or
  - train a new model on all of the data, using the hyperparameters selected by cross-validation.
- Advantage of cross validation - *robustness*

Fairness - Machine Learning

Fairness in Classification

Advertising

Education

Financial aid

Banking

Health Care

Insurance

Taxation

many more...

Concern: Discrimination

- Certain attributes should be *irrelevant*!
- Population includes minorities
  - Ethnic, religious, medical, geographic
- Protected by law, policy, ethics

Discrimination arises even when nobody's *evil*

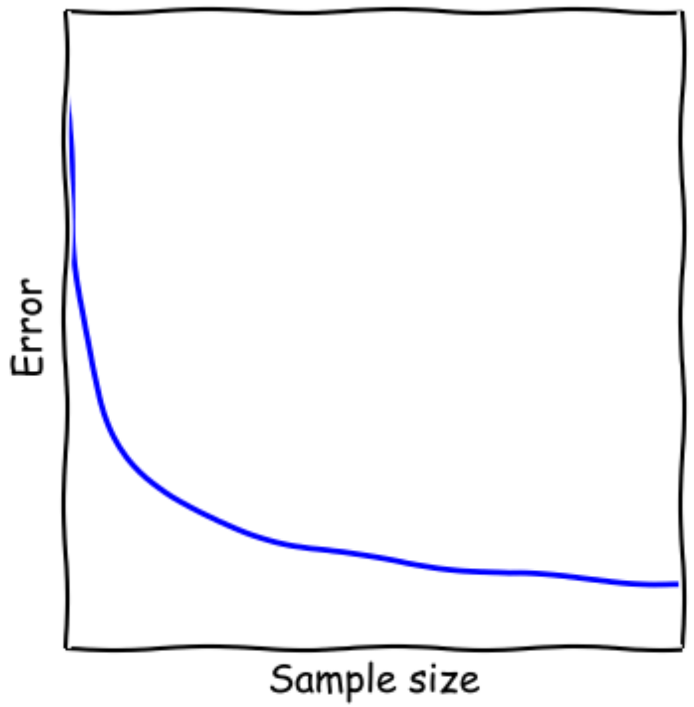
- Google+ tries to classify real vs fake names
- Fairness problem:
  - Most training examples standard white American names: John, Jennifer, Peter, Jacob, ...
  - Ethnic names often unique, much fewer training examples

Likely outcome: Prediction accuracy *worse on ethnic names*

"Due to Google's ethnocentricity I was prevented from using my real last name (my nationality is: Tungus and Sami)"

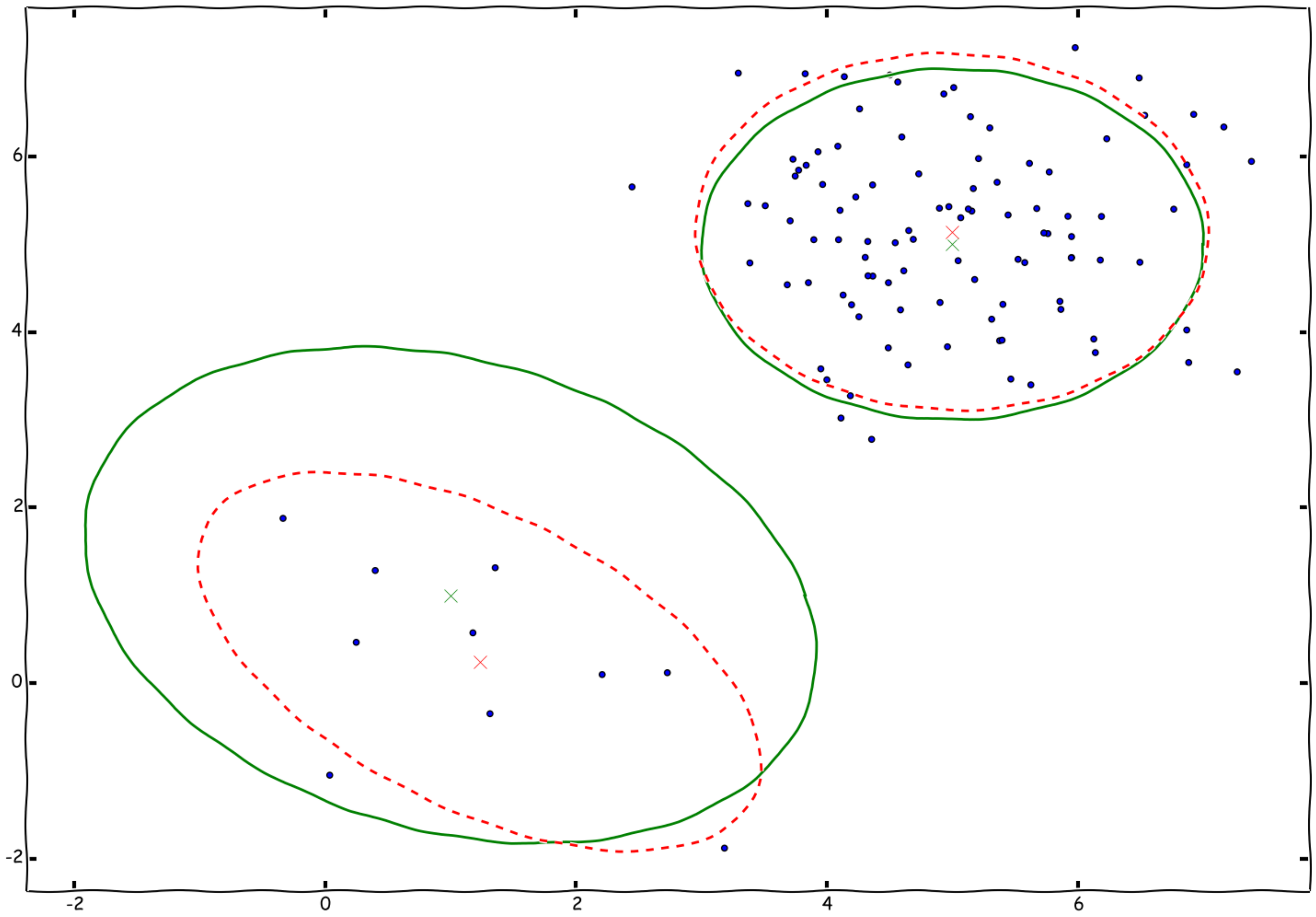
- Katya Casio. Google Product Forums.

Error vs sample size



Sample Size Disparity:

In a heterogeneous population, smaller groups face larger error


-2
0
2
4
6
-2
0
2
4
6

Disparate impact occurs when a selection process has widely different outcomes for different groups, even as it appears to be neutral.

E.g.:

refusing to hire people because of a poor credit rating, when minorities are disproportionately affected.

Disparate impact ("80 % rule")

Given:

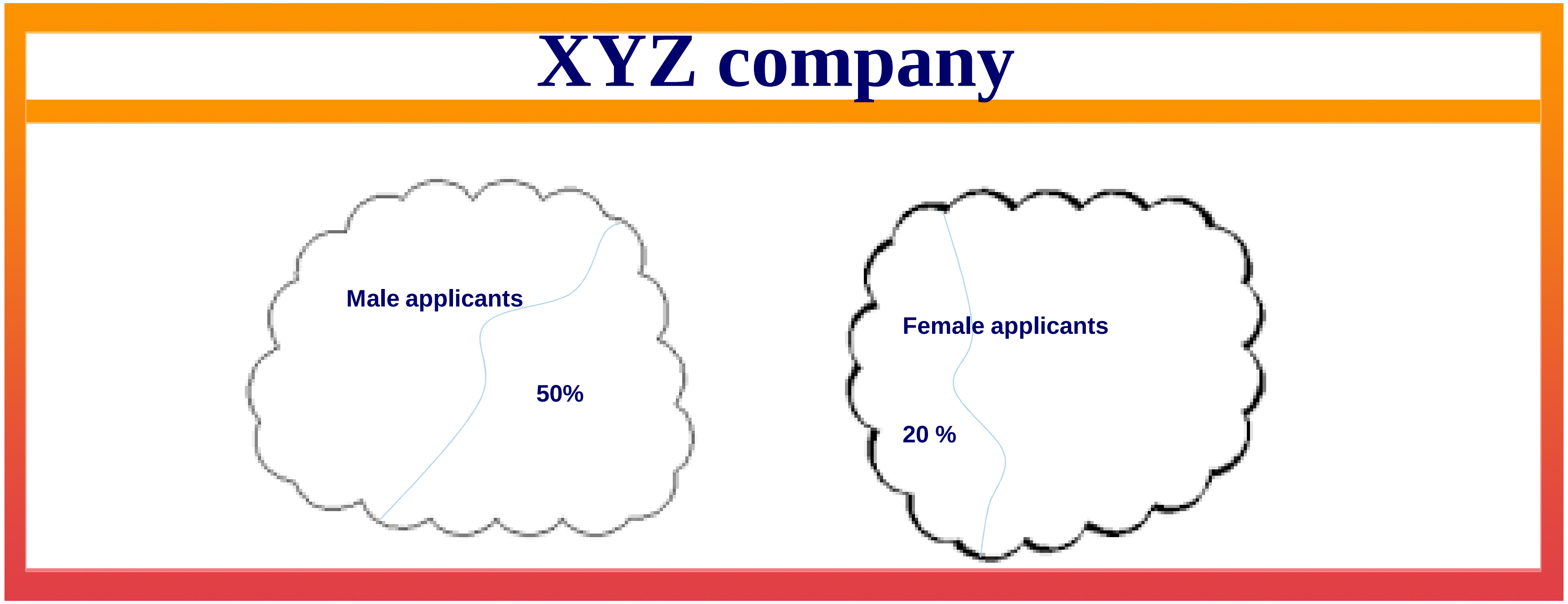
D = (X, Y, C)

X – protected attribute
Y – remaining attributes
C – binary class to be predicted

D has disparate impact if

$$\frac{\Pr(C=YES \mid X=0)}{\Pr(C=YES \mid X=1)} \leq \tau = 0.8$$

80 % rule example



$$\tau = \frac{\Pr(C{=}YES \mid X{=}FEMALE)}{\Pr(C{=}YES \mid X{=}MALE)} = \frac{20\ \%}{50\ \%} = 0.4 < 0.8$$

Disparate impact certification problem

- **Guarantee that given D, any classification algorithm aiming to predict some C’ from Y would not have disparate impact.**

Disparate impact removal problem

- **to take some data set D and return a data set $\bar{D} = (X, \bar{Y}, C)$ that can be certified as not having disparate impact.**

A confusion matrix

Outcome	$X = 0$	$X = 1$
$C = \text{NO}$	a	b
$C = \text{YES}$	c	d

The 80% rule can then be quantified as:

$$\frac{c/(a+c)}{d/(b+d)} \geq 0.8$$

The likelihood ratio positive:

$$LR_{+}(C,X) = \frac{d/(b+d)}{c/(a+c)}$$

$$\text{DI} = \frac{1}{\text{LR}_{+}(C,X)}$$

COMPUTATIONAL FAIRNESS

A: ~~X~~ , Y –> C

~~A~~

If Bob cannot predict X given the other attributes of D, then A is fair with respect to Bob on D.

Credit Application

User visits `capitalone.com`

Capital One uses tracking information provided by the tracking network [x+1] to personalize offers

Concern: *Steering* minorities into higher rates (illegal)

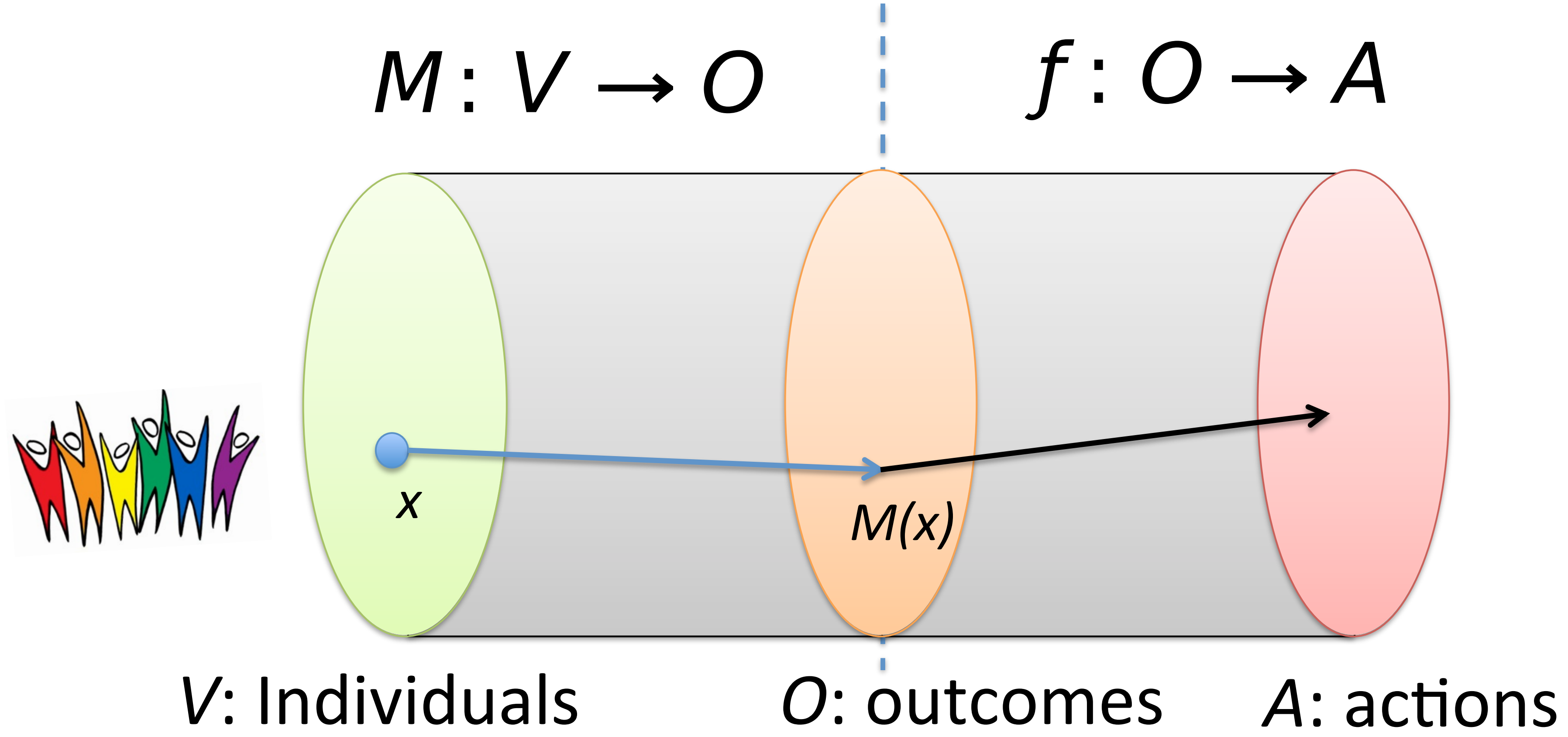
WSJ 2010

Classifier
(eg. ad network)

Vendor
(eg. capital one)

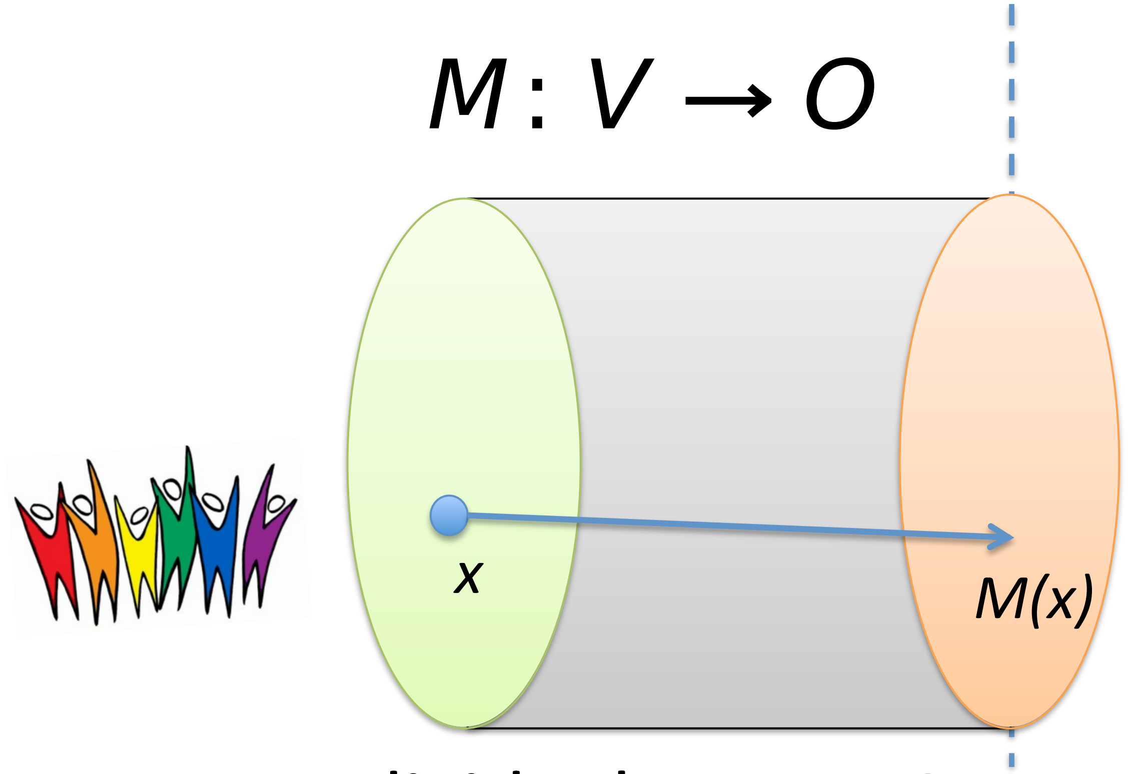
$M : V \rightarrow O$

$f : O \rightarrow A$



Our goal:

Achieve Fairness in the classification step



V: Individuals *O*: outcomes

Fairness through Blindness

Fairness through Blindness

Ignore all irrelevant/protected attributes

Point of Failure

You don’t need to *see* an attribute to be able to *predict* it with high accuracy

E.g.: User visits `artofmanliness.com`

... 90% chance of being male

Fairness through Privacy?

“It's Not Privacy, and It's Not Fair”

“At worst, **privacy solutions can hinder efforts to identify classifications that unintentionally produce objectionable outcomes**—for example, differential treatment that tracks race or gender—by limiting the availability of data about such attributes.“

Statistical Parity (Group Fairness)

Equalize two groups S, T at the level of outcomes

- E.g. S = minority, $T = S^c$

$$\Pr[\text{outcome } o \mid S] = \Pr[\text{outcome } o \mid T]$$

"Fraction of people in S getting credit same as in T."

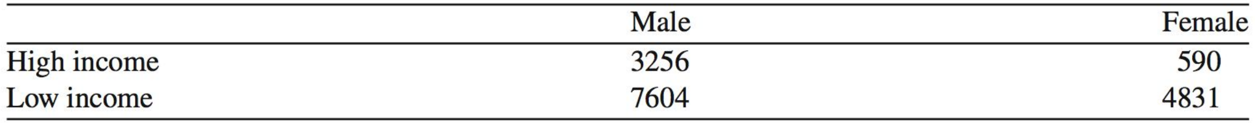
Naïve Bayes and Discrimination

Performance of Naive Bayes without discrimination awareness

Case:

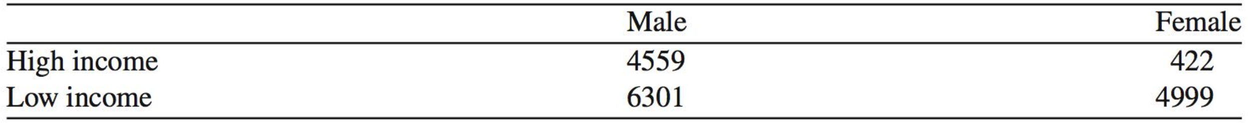
- Classifying individual as high-income or low-income
- Focus on gender discrimination
- Census Income Data set
  (UCI Machine Learning Repository)
- Highly discriminatory labeling
  - About 30% of all male individuals and only about 11% percent of all female individuals have a high income

Performance of Naive Bayes without discrimination awareness



	Male	Female
High income	3256	590
Low income	7604	4831

(training data)

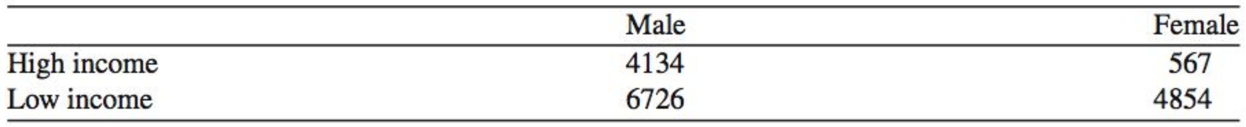


	Male	Female
High income	4559	422
Low income	6301	4999

(classification result)

“Learning this classifier results in about 42% of all males having a high income, and only 8% of all females”

Performance of Naive Bayes: Removal of discriminatory feature

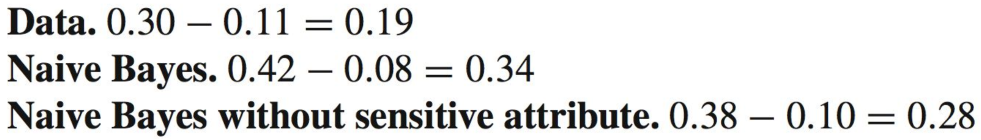


	Male	Female
High income	4134	567
Low income	6726	4854

- Slight improvement of results
  - (38% male classified positively, against 10% of female)
- Still more discriminatory than labeled data
  - Redlining with features that correlate with gender.

Measuring discrimination

Discrimination score: difference between the probability of a male and a female of being in the high-income class



Data. $0.30 - 0.11 = 0.19$
Naive Bayes. $0.42 - 0.08 = 0.34$
Naive Bayes without sensitive attribute. $0.38 - 0.10 = 0.28$

Zero would be ideal, assumes probability of positive classification should be the same

Measuring discrimination

First note: Discrimination score measure seems to be simplistic!

- Measures both direct and indirect discrimination together.
- Assumes sample is representative of whole population
  - This case is ok, since it’s a census
  - Counter example - Credit rating - hypothetic:
    - Only successful females access credit
    - Most men access credit
    - Discrimination score zero is still discriminatory to women

Solution: Modifying Probabilities

Statistical Parity (Group Fairness)

Equalize two groups S, T at the level of outcomes

- E.g. S = minority, $T = S^c$

$$\Pr[\text{outcome } o \mid S] = \Pr[\text{outcome } o \mid T]$$

"Fraction of people in S getting credit same as in T."

Main idea

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$$

Removing discrimination by modifying the probability distribution P(S|C) of the sensitive attribute values S given the class values C.

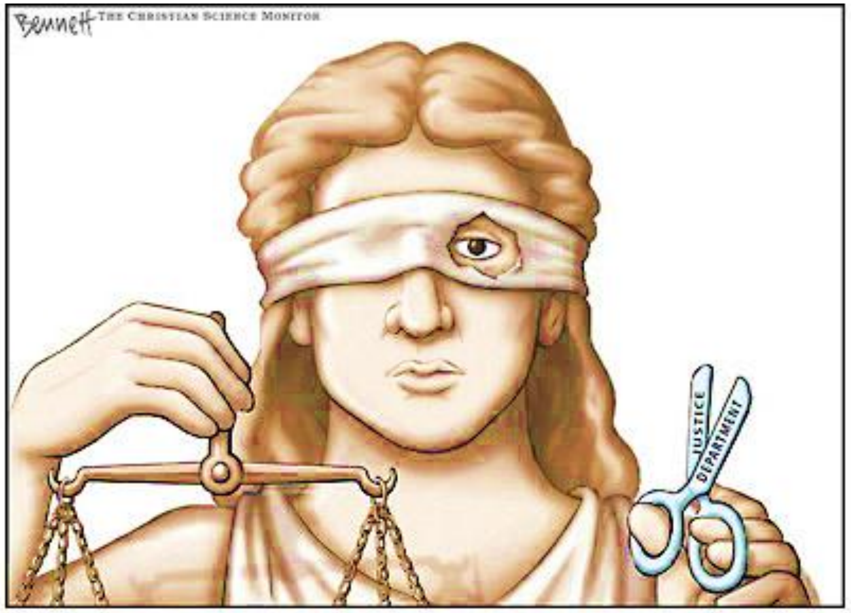
eg.

Increase P(S♀|C+) and reduce P(S♂|C+), if we want to positively discriminate ♀, or vice-versa

Fairness is task-specific

Fairness requires understanding of classification task and protected groups

AWARENESS


Bennett THE CHRISTIAN SCIENCE MONITOR
JUSTICE DEPARTMENT

Questions?

Thank you