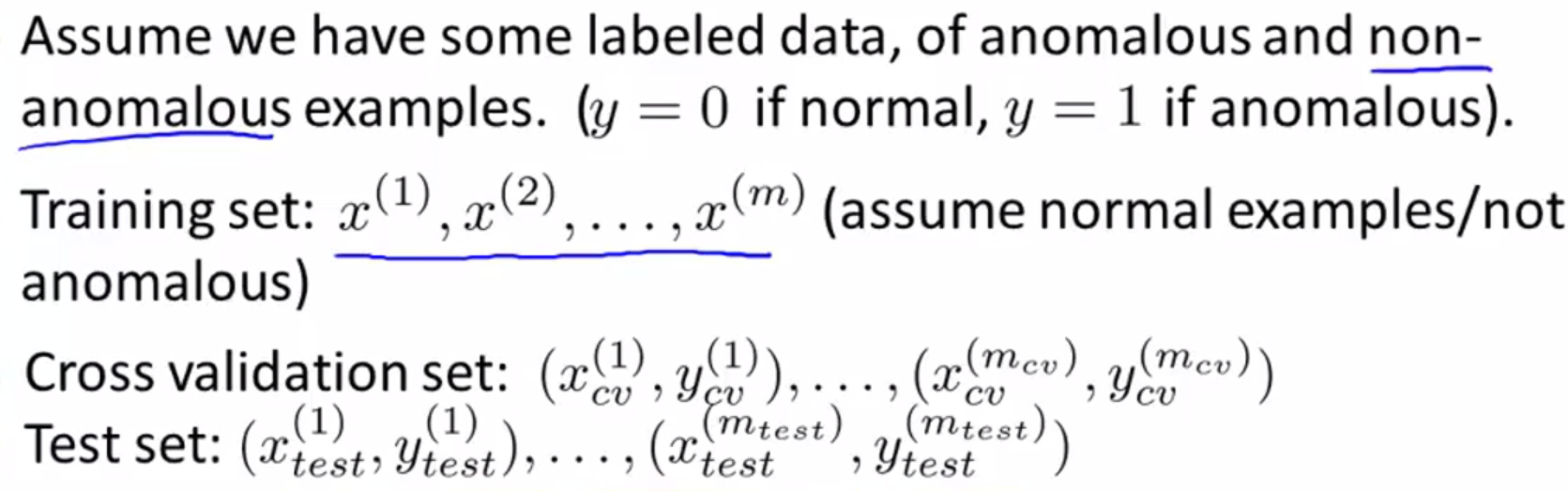
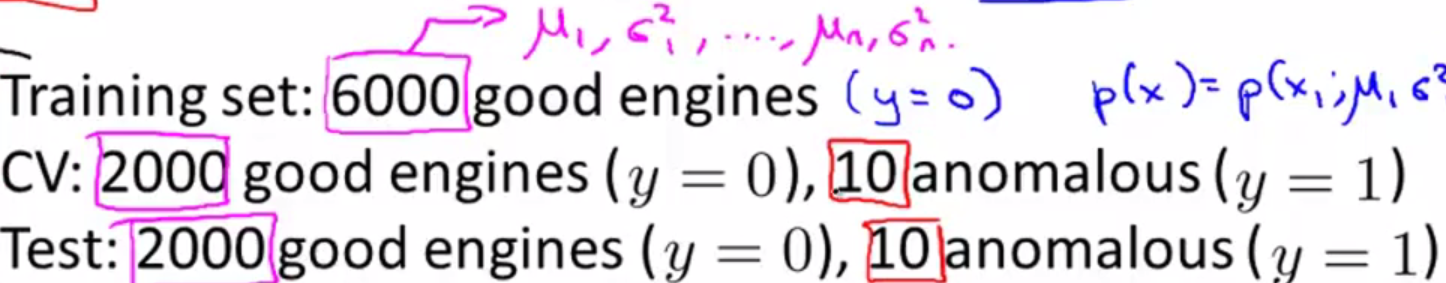
***Building an Anomaly Detection System***

**I. DEVELOPING AND EVALUATING AN ANOMALY DETECTION SYSTEM**

* The importance of **real number evaluation** 🡪 when trying to develop a learning algorithm for a specific application, often need to make a lot of choices (choosing features to use, etc.) + making decisions about all these choices is often much easier w/ a way to evaluate a learning algorithm that just *gives you back a number*.
* Have an idea for 1 extra feature? Run the algorithm w/ the feature + w/out the feature + see if it improved or worsened performance via a real number
* So in order to be able to develop an anomaly detection system quickly, it’d be a really helpful to have a way of *evaluating* an anomaly detection system.
* We're actually going to assume we have some labeled data
* So far, we've been treating anomaly detection as unsupervised (using unlabeled data)
* But if you have labeled data that specifies some anomalous + non-anomalous examples, this is the standard way of evaluating an anomaly detection algorithm.
* Some labeled data of a few anomalous examples of aircraft engines manufactured in the past + also have some non-anomalous examples
* y = 0 denotes normal/non-anomalous example + y = 1 denotes anomalous examples.
* The process of developing and evaluating an anomaly detection algorithm is as follows.
* Think of it as a training set as unlabeled = large collection of normal, non-anomalous examples.
* it's okay if a few anomalies slip into your unlabeled training set.
* Define a CV + a test set w/ which to evaluate a particular anomaly detection algorithm



* Ex: 10k examples of normal engines (assume vast majority of these are non-anomalous engines) + 20 anomalous engines
* For a typical application of anomaly detection, non-anomalous examples may range from 20-50.
* Usually have a much larger number of good examples.
* A fairly typical way to split it into the training, CV, + test sets would be a 60/20/20 split:



* 6k into unlabeled training set (unlabeled but all examples correspond to y = 0, as far as we know)
* We will use this to fit p(x(i) parametrized by μ(i), δ^2(i)) 🡪 to estimate parameters μ1, δ^2
* Halve remaining examples into CV + test sets w/ 2k examples
* Also have 20 flawed aircraft engines 🡪 split + put 10 in CV set + 10 in test set
* Like to think of the CV set + test set as being completely different data sets to each other, but, in anomaly detection, sometimes people use the same set of good examples in both the CV sets + the test sets, sometimes people use exactly the same sets of anomalous examples in these sets
* Certainly using the same data in the CV set + the test set, that is not considered a good ML practice.
* So, given the training, CV + test sets, here's how you develop and evaluate an algorithm.
* Take the training set + we fit the model p(x) 🡪 fit all those parameters for all the Gaussians to the m unlabeled examples
* Then imagine the anomaly detection algorithm is actually making predictions
* On the CV or test set, given example X, think of the algorithm as predicting that y = 1 when p(x) < epsilon + y = 0 if p(x) >= epsilon.
* Can evaluate it by seeing how often it gets these labels right, similar to evaluation metrics used in supervised learning.
* Labels will be very skewed b/c y = 0 are normal examples, usually much more common than y = 1 anomalous examples.
* B/c the data is very skewed (i.e. b/c y = 0 is much more common) classification accuracy would NOT be a good evaluation metric
* W/ a very skewed data set, predicting y = 0 all the time will give very high classification accuracy.
* Instead, use evaluation metrics like computing the **fraction of TP’s, FP’s, FN’s, TN’s, etc**.
* Or compute the **position** + **recall** of this algorithm or compute **the f1 score** (a single real number way of summarizing the position + recall values
* **Epsilon** is this threshold used to decide when to flag something as an anomaly.
* If you have a CV set, another way to + to choose this parameter epsilon would be to try many different values of epsilon + then pick the value that maximizes f1 score, or that otherwise does well on your CV set
* More generally, the way to use the training, testing, + CV sets, is when trying to make decisions (what features to include, tune the parameter epsilon), we’d continually evaluate the algorithm on the CV sets + make all those decisions for a set of features, or a value of epsilon we're happy w/
* We can then take the final model + do a final evaluation of the algorithm on the test sets.
* Being able to evaluate an algorithm w/ a single real number evaluation, like an F1 score, often allows you to much more efficient use of time when trying to develop an anomaly detection system

**II. ANOMALY DETECTION VS. SUPERVISED LEARNING**

**III. CHOOSING WHAT FEATURES TO USE**