Machine Learning System Design:

- ➔ To prioritize your time spent

Eg Building a Spam Classifier

1) Find relevant features:
    a. Most frequently occurring n words in training set eg [10000-50000] words.

What options can we pursue to get the most efficient use out of our time?

- Collect More Data
    - Eg Honeypot email accounts to get more spam mail examples.
- Develop sophisticated features based on email routing information (from email header)
- Develop Sophisticated features from message body
    - 'discount' and 'discounts' considered the same word?
    - 'Dealer' and 'deal'?
- Punctuation?

Workflow Approach:

- ➔ Start with a simple algorithm that is easily implemented, a Quick and Dirty Solution.
- ➔ Implement and test on CV data
- ➔ Plot a learning curve to evaluate if more data/more features, etc will help.
- ➔ Error Analysis

Error Analysis:

Manually examine the examples (In CV set) that the algorithm makes errors on.

- See if there are systematic trends that can be found in what kind of examples that it makes errors on
    - Using such analysis, we can design new features based on these errors.

To test how effective these new features are, we conduct numerical evaluation:

Eg:

Should discount/discounts/discounted/discounting be treated the same?

- ➔ Use Stemming software / fuzzy string matching to catagorise
    - Note that such featuring may also adversely impact the algorithm in other ways like University/Universe => clustered together even though they mean completely different ways.

Then evaluate the altered model against the initial model

w/o stemming: 5% error with stemming: 3% error

Hence Yes it improves!

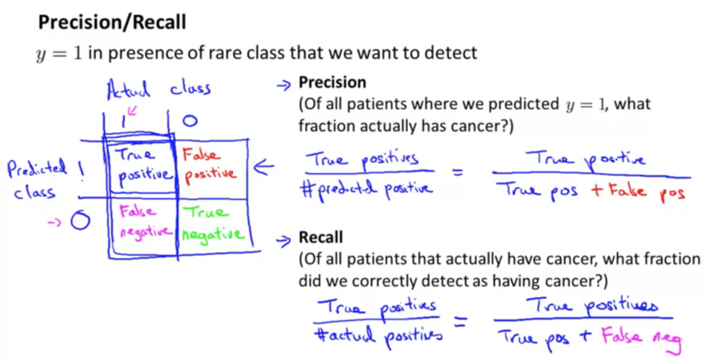
Distinguishing uppercase vs lowercase?

Without: 5% With: 10%

No it does not.

- ⇨ Hence by seeing what works and what doesn't with a simple implementation we can decide what features are most important, and circumvent the problem of premature optimization.

Error Metrics:
Often times, the proportion of datasets may not be balanced
ie. there may be skewed classes
Eg: Logistic Regression Model on Cancer Diagnoses
Perhaps there is only 1% of actual positive test results in the dataset.



Precision:
Percentage of true positives out of all predicted positives.
ie Specificity
Recall:
Percentage of True Positives out of actual number of positives
ie Sensitivity

We can adjust the sensitivity and recall by controlling the thereshold value.
Eg Logistic Regression:
Threshold = 0.7
Predict 1 if $h_\theta(x) \geq 0.7$
Predict 0 if $h_\theta(x) < 0.7$
ie we only predict positive if we are confident more than the threshold about the prediction.
This causes the precision to increase, at a cost to the value of the recall.

Threshold = 0.3
Predict 1 if $h_\theta(x) \geq 0.3$
Predict 0 if $h_\theta(x) < 0.3$
This causes the recall to increase, at a cost to the value of the precision.

ROC curve,
The Precision and Recall can be plot against each other in a ROC curve.

F1 Score:
F1 Score is a single metric that helps measure the total efficacy of a model.
The formula:

$$F_{score} = 2\frac{PR}{P + R}$$

Hence if P = 1 and R = 1, F score = 1
If P = 1, R = 0, F score = 0, same vice versa.

On Data:
Designing a high accuracy learning system.

Conditions for more Data as the solution:
Assuming Features $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately.
A useful heuristic test: Given input x, can a human expert confidently predict y?

Use a learning algorithm with many parameters
Eg Logistic Regression/Linear Regression with many features / Neural Network with many hidden units => Low bias algorithm + Low varience

$$J_{train}(\theta)\ is\ small$$

+ Use a large training set = Low chance of overfitting
  - ⇨ $J_{test}(\theta) \approx J_{train}(\theta)$
  - ⇨ Hence resulting in a effective model.