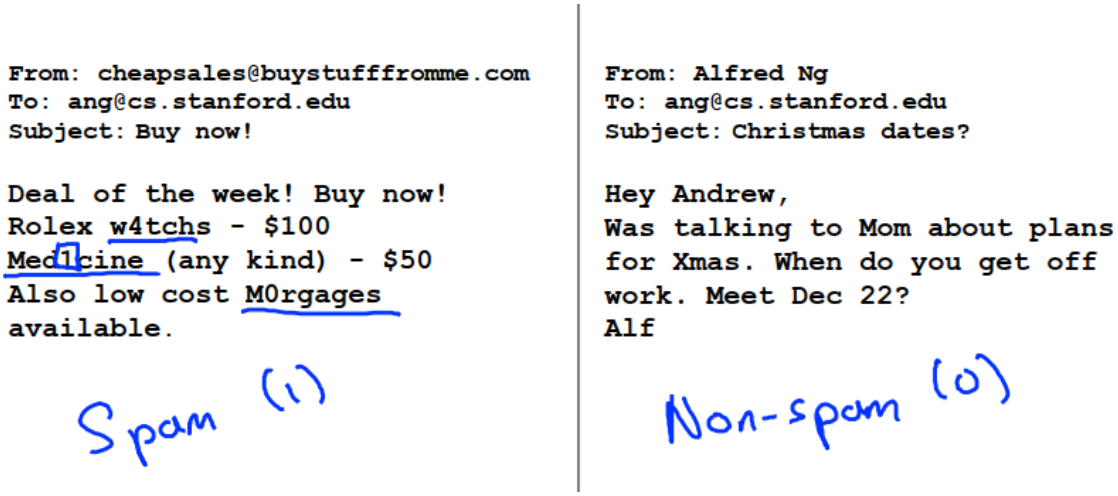
Spam / Non-spam classifier project

Prioritizing what to work on

▼ **Spam / Non-spam example**



▼ ***Where*** do I need ***to focus*** / ***How*** can I spend my time on?

- Collect lots of data
- Develop sophisticated ***features based on email routing information*** (from email header)
- Develop sophisticated ***features for message body***
  - E.g. should "discount" and "discounts" be ***treated as the same word***?
  - How about "deal" and "Dealer"?
  - Features about ***punctuation***?
- Develop sophisticated algorithm to ***detect misspellings*** (e.g. m0rtgage, med1cine, w4tches.)

Recommended approach

1. ***Start with a simple algorithm*** that you can implement quickly. ***Implement it and test*** it on your ***cross-validation data***
2. ***Plot learning curves*** to decide if more data, more features, etc are likely to help
3. ***Error analysis***: Manually examine the examples (in cross-validation set) that your algorithm made errors on.
    - See if you spot any ***systematic trend*** in what ***type of examples*** it is ***making error on***

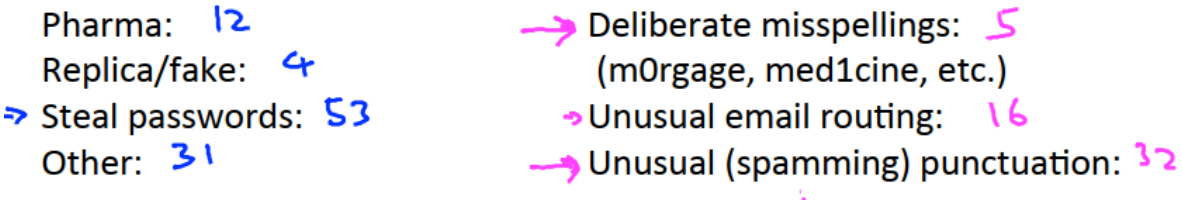
Error analysis

▼ Cross-validation set has 500 examples

- Algorithm ***misclassifies 100 emails*** in cross-validation set

—> ***Manually examine the 100 errors***, and categorize them based on:

1. What ***type of email*** it is? (E.g. Pharma, replica _ email to fake product, steal password)
2. What cue (***features***) you think ***would have helped the algorithm*** classify them correctly?



- For example, this case # of "steal passwords" is the type algorithm misclassifies the most

▼ **Some problems** can be emerged

- ***Deliberate misspellings***
- ***Unusual email routing***
- ***Usual (spamming) punctuation***

▼ **Stemming technique**

- ***"Stemming"*** is the process of ***reducing inflection (sự biến đổi) in words*** to their root forms such as ***mapping a group of words to the same stem*** even if the stem itself is not a valid word in the Language

—> Should ***discount/discounts/discounted/discounting*** be ***treated as the same word***?

▼ **How** to ***decide*** whether ***using "Stemming" technique or not***?

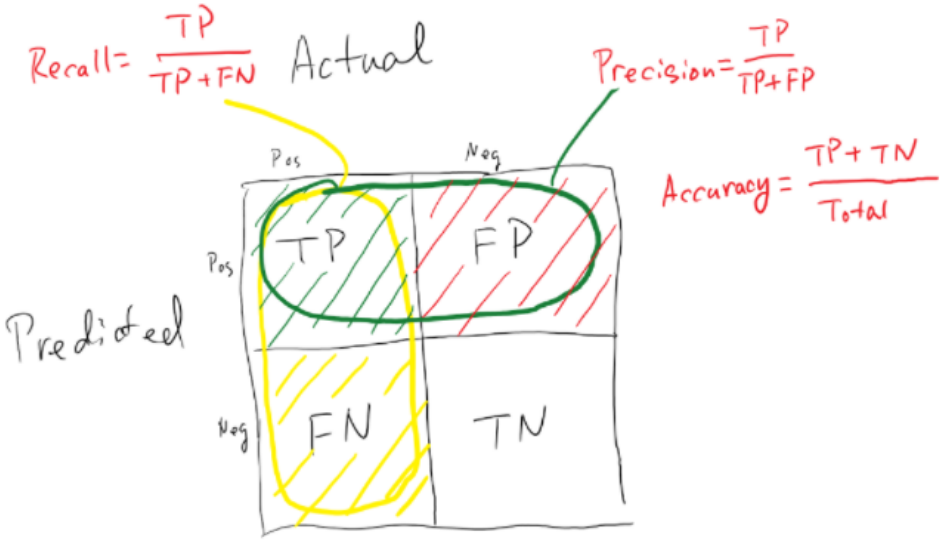
- **Comparing the performance** with or without using this technique.

Without stemming: 5% error With stemming: 3% error

Distinguish upper vs. lower case (Mom/mom): 3.2%

—> Using it!!!!

▼ **Error metrics** / **Confusion metrics**



▼ **F1 score**

$$F\text{-}measure = \frac{2*Recall*Precision}{Recall + Precision}$$

- F1 score shows the ***comparison between algorithms*** (1, 2, and 3)

--> Algorithm with the ***highest F1 score is the best algorithm***

(For more detail) Looking at Notion "kaggle"

▼ **Precision**

$$Precision = \frac{TP}{TP + FP}$$

- From the equation above, we can explain **the precision** as to ***From all classes we have predicted as positive how many of them are actually positive.***

> (Just remember) **All predicted positive —> Actually positive**

▼ **Recall**

$$Recall = \frac{TP}{TP + FN}$$

- From the equation above, we can explain **the recall** is that ***From all the set of actual positive classes, how many of them are predicted correctly***

> (Just remember) **All actual positive —> predicted correctly**

▼ **Relationship** between ***Type I/Type II error*** and ***Precision/Recall***

Type I + Type II Errors

Type I Error: Rejecting the null hypothesis when it is true.

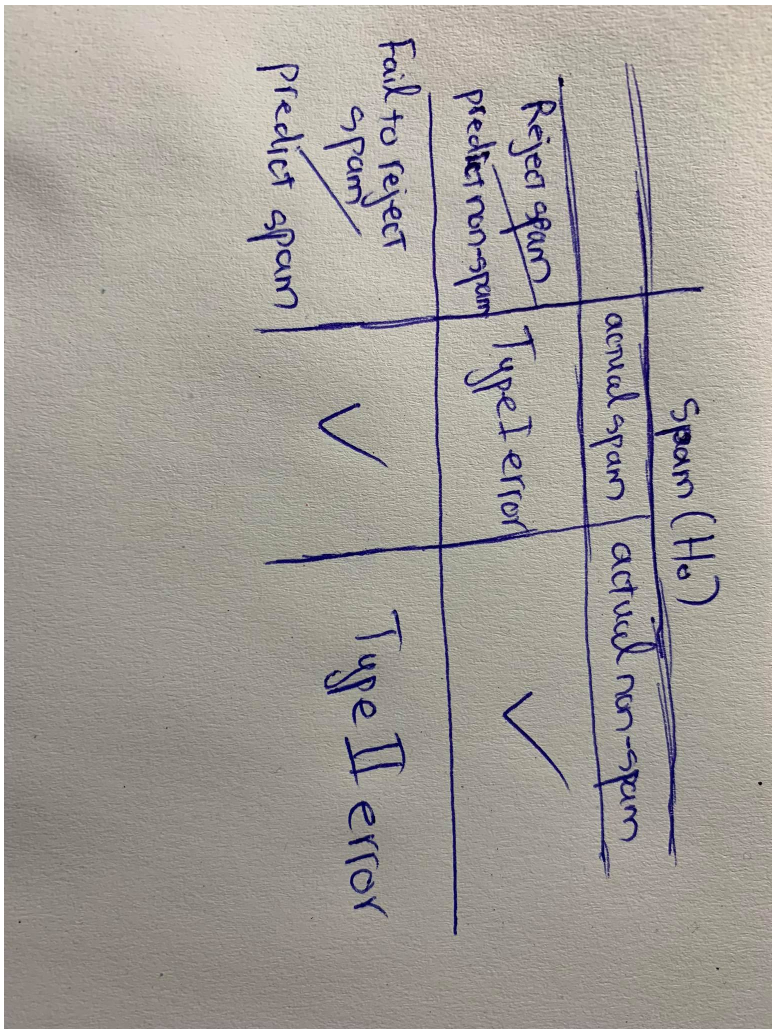
Type 2 Error: Not rejecting the null hypothesis when it is false.

P(type I error / Ho is true) = α

P(type II error / Ho is false) = β

P(rejecting a false Ho) = $1 - \beta$

	Ho	
	True	False
Reject Ho	Type I Error	✓
Fail to Reject Ho	✓	Type II Error



- **Null hypothesis (Ho):** The email is spam

- **Type I error:** The email is ***actually spam*** but the model ***predicts it as non-spam*** —> **FALSE NEGATIVE**

  > ***Avoid*** Type I error —> ***higher recall, lower precision***

- **Type II error**: The email is ***actually non-spam*** but the model ***predicts it as spam*** —> **FALSE POSITIVE**

  > ***Avoid*** Type II error —> ***Higher precision, lower recall***

- There is ***trade-off between Type I and Type II error***

(**NOTE**) Depending on each ***project's situation,*** we'll ***prioritize type I*** or ***type II error***

▼ **CONCLUSION**

- In situation classifying between spam / non-spam email
- We prefer avoiding missing information which comes from non-spam email. It means that we will ***prevent*** the case when the email is ***actually non-spam*** but the model ***predict it as spam***

  —> This case, we will ***prioritize "Type I error" —> Avoid "Type II error"***

Data in machine learning

▼ **Data rules** of Andrew Ng

- **If** using a learning algorithm with **many parameters + Powerful algorithm (1)** —> ***J_train should be small size*** (even large or small training set)
- If using ***(1) + very large training set*** -> ***Solving Overfit problem*** -> J_train and J_test are nearly the same -> ***J_test should be small size***