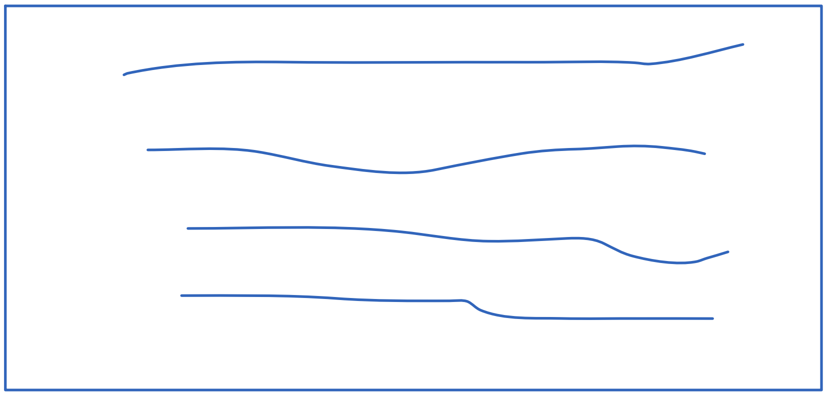
Section 1.2: ML vs. Rule-Based Systems

Monday, September 5, 2022 17:14

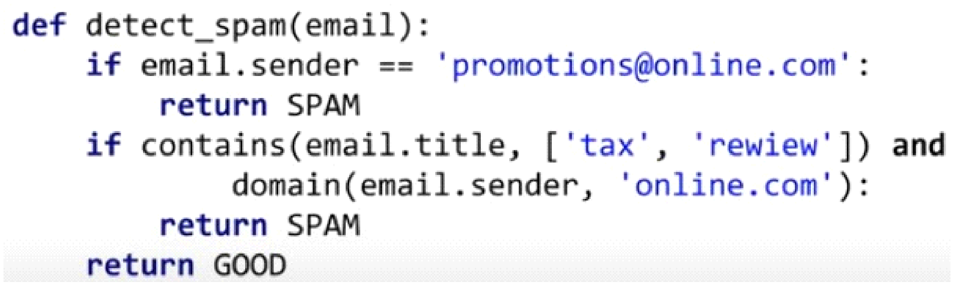
Plan:

- A rule-based system for spam detection.
- Using ML for spam detection.
- Extracting features for ML.

Consider an email system:



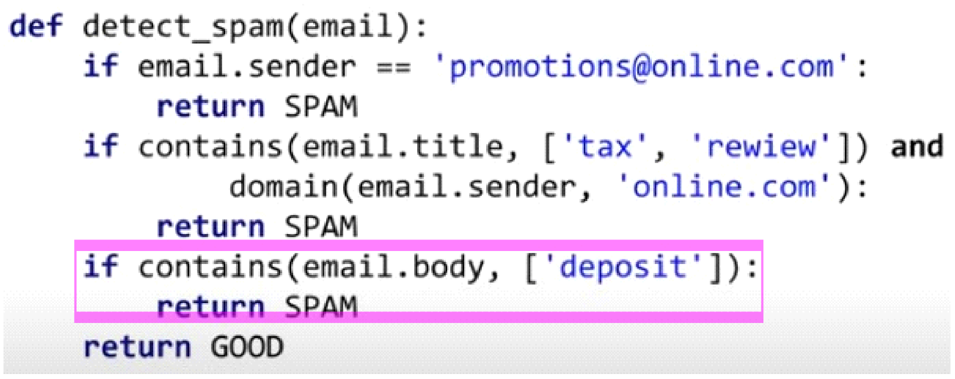
- In the drawing, each line across represents an email, but you begin to receive unsolicited emails (read as: spam).
- To combat against the unsolicited emails, a spam folder is composed; to actually get the emails into the folder, we want to build a classifier that categorizes content as either "spam" or "not spam."
- We can establish some rules:
  - If sender = [for example] promotions@online.com, then "spam."
  - If title contains [for example] "tax review" and sender domain is "online.com" then "spam."
  - Otherwise, "good email" or "not spam."
- What we can do then is write a simple program code using [for example] Python:



```
def detect_spam(email):
    if email.sender == 'promotions@online.com':
        return SPAM
    if contains(email.title, ['tax', 'rewiew']) and
            domain(email.sender, 'online.com'):
        return SPAM
    return GOOD
```

So, in other words, everything so far has been a scenario surrounding a rule-based system. This rule-based system is sustainable until complaints surface about other unsolicited messages. Now you need to establish **another rule** within your established set of rules.

- **RECALL:** the previous set of rules:
    - If sender = [for example] promotions@online.com, then "spam."
    - If title contains [for example] "tax review" and sender domain is "online.com" then "spam."
    - **If body contains a word "deposit" then "spam".**
    - Otherwise, "good email" or "not spam."
- Then we implement a new Python code:



```
def detect_spam(email):
    if email.sender == 'promotions@online.com':
        return SPAM
    if contains(email.title, ['tax', 'rewiew']) and
            domain(email.sender, 'online.com'):
        return SPAM
    if contains(email.body, ['deposit']):
        return SPAM
    return GOOD
```

- The problem may arise when a legitimate email is erroneously categorized as "spam", thereby sending it to the spam folder.
- Then we conduct another analysis, and generate another new rule within our established rules:
    - If sender = [for example] promotions@online.com, then "spam."
    - If title contains [for example] "tax review" and sender domain is "online.com" then "spam."
    - **If body contains a word "deposit"**
        - **If sender domain is "test.com" then "spam".**
        - **If body >= 100 words then "spam"**
    - Otherwise, "good email" or "not spam."

The problem persists in the rule-based system as spam keeps changing or evolving; resulting in programmers having to write more code, and more code in an ever-repeating process.

More code being written means more code that requires maintenance, which is a nightmare to maintain.

Machine learning is the tool that we can use to solve this problem. Machine learning solves this problem in a different way:

- **Get data.**
    - What email providers do, is they have a SPAM folder that allows consumers to place specific emails into it where it is categorized as spam.
    - Then, the email provider can take all of the data that users generated and make use of it by training a model.
- **Define and calculate features.**
    - Features can be simple, but after the email provider gets the data surrounding spam emails, these features are defined.
    - The features can be based on the previous established set of

rules.
    - *Start with rules and then use these rules as features in a machine learning system.*
    - So consider the following set of features:
        - Length of title > 10? <u>true/false</u>
        - Length of body > 10? <u>true/false</u>
        - Sender "promotion@online.com"? <u>true/false</u>
        - Sender "hpYOSKmL@test.com"? <u>true/false</u>
        - Sender domain "test.com"? <u>true/false</u>
        - Description contains "deposit"? <u>true/false</u>
    - <u>The true/false are called binary features.</u>
- **Train and use the model.**
    - Using the binary data, you run it through the model to predict the certainty that a given email is spam or if it goes to the inbox.
    - Parameters can be specified; where if a value generated by the machine learning process produces, say, a result greater than or equal to 0.5, then the email is sent to the spam folder.

With software, you have data and code that is plugged into software to generate an outcome (spam/not spam) produced by us. We hard code the outcome in the code.

In machine learning, on the other hand, both the data <u>**AND**</u> the outcome (spam/not spam) are the input to the machine learning algorithm, and the final output is a model. The model is then used to make predictions for cases when we don't know the outcome.

$Data + Model = Outcome$

<u>**Next:**</u> Supervised ML
- A bit more formal definition.
- Examples: regression, classification, ranking.

<u>**Notes by Community:**</u>

The differences between ML and Rule-Based systems is explained with the example of a **spam filter**. The traditional systems are based on a set of characteristics that identify an email as spam, which has some drawbacks because the spam emails keep changing over time. The system must be upgraded with these adjustments, and this process is untraceable due to code maintenance and other issues.

The ML systems can be trained with features and targets extracted from the rules of Rule-Based systems, a model is obtained, and it can be used to predict new emails as spam or regular emails. The **predictions are probabilities**, and to make a decision it is necessary to define a threshold to classify emails as spam or normal ones.