# Text Classification and Its Applications in Software Engineering

Shin Hong

23 Oct 2018

# Text Classification

- Modeling
  - Document $d \in X$
    Classes $C = \{c_1, c_2, \ldots, c_n\}$
    Training set $\{(d, c)\} \in X \times C$
  - Find a classification function $\gamma: X \to C$

- Problems
  - News categorization
  - Spam filtering
  - Sentiment analysis

# Document Representation (1/4)

- The topic/content of a document is characterized by words/phases appeared in the text
  - certain words appear more frequently on a specific class of documents, than on the others
  - the space of documents is typically high-dimensional because there are many words in universe

- Represent a document as a vector such that each index represents how a certain lexical feature is relevant to the document

# Document Representation (2/4)

- Normalization
  - Tokenization
  - Stop-words removal
  - Stemming

| | |
|---|---|
| *Original description* | `crashes when I Manage Bookmarks with a Personal Toolbar Folder link` |
| *After stop-words removal* | `crashes manage bookmarks personal toolbar folder link` |
| *After stemming* | `crash manag bookmark person toolbar folder link` |

- Control dictionary size

- (Embedding)
  - represent a word as a vector of features such that each feature declares a certain property of a word
  - make two similar words have close vector representations

# Document Representation (3/4)

- Bag-of-words model
  - consider a token as a document feature
  - each element of a document vector represents a word (in a dictionary)
  - the vector for a document counts the appearance of words

- TF-IDF normalization
  - Term Frequency: how frequently a word appears in a target document
  - Inverse Document Frequency: how frequently a word appears in all documents

# Document Representation (4/4)

- N-gram model
  - use N consecutive tokens as a feature of a document

- k-Skip-N-gram
  - use a sequence of N tokens in N+k consecutive sequence of tokens (i.e., allow k skip in a middle) as a document feature

- Bag vs. N-gram vs. k-Skip-N-gram ?

# Classifiers

- Naïve Bayes classifier
- Support Vector Machine classifier
- Decision Tree classifiers
- Neural-Net classifiers

# Naïve Bayes Classifiers

- Find how each feature is related to a class while assuming that each feature are independent to each other
- Find a likelihood of a document being classified into a class by counting an appearance of a feature as an independent event

Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)\,P(c)}{P(x)}$$

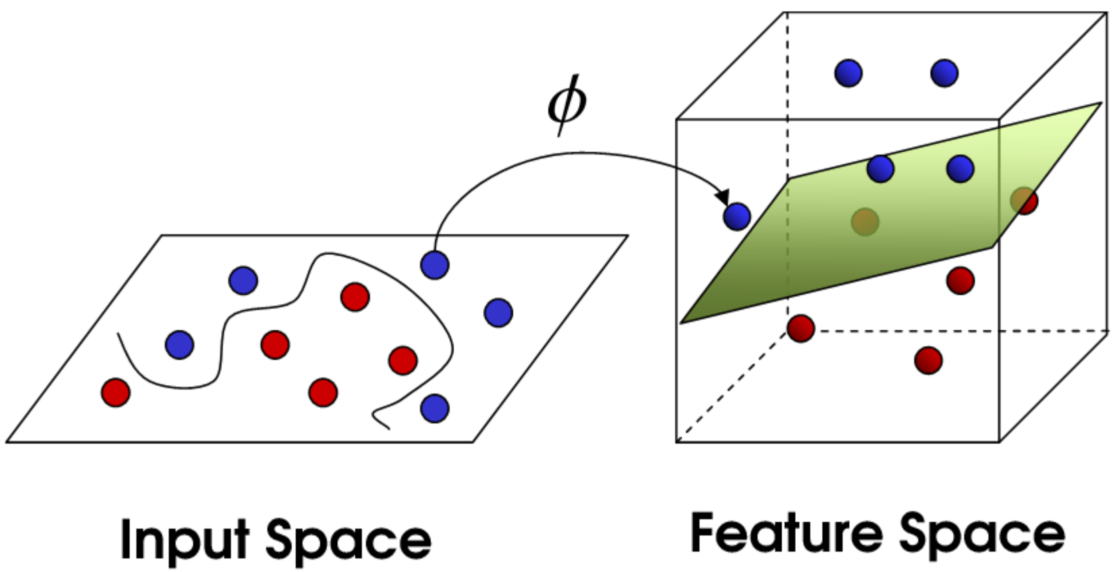
Posterior Probability

Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

# Support Vector Machine

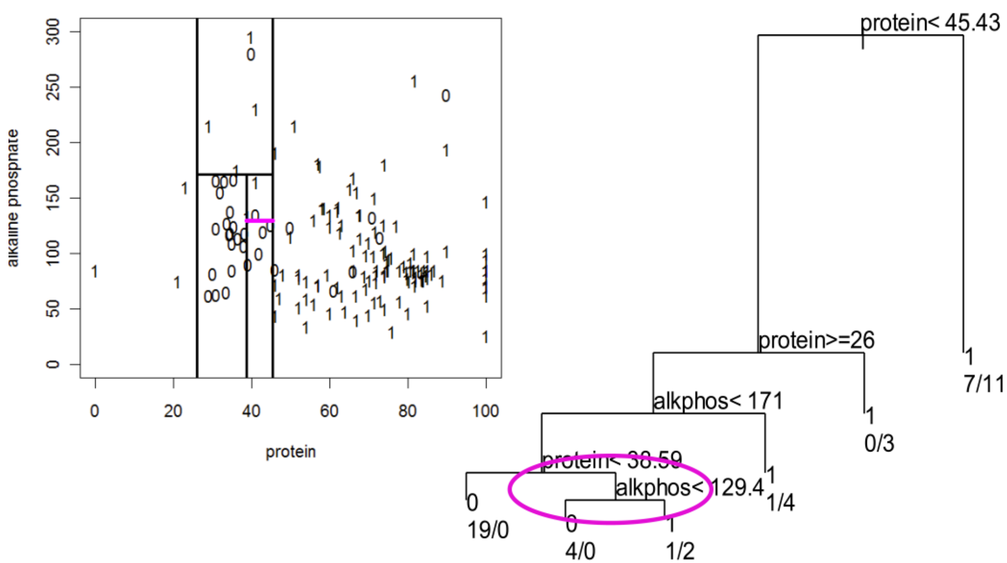
- Find a hyper-plane that well separates groups of datapoints in different classes

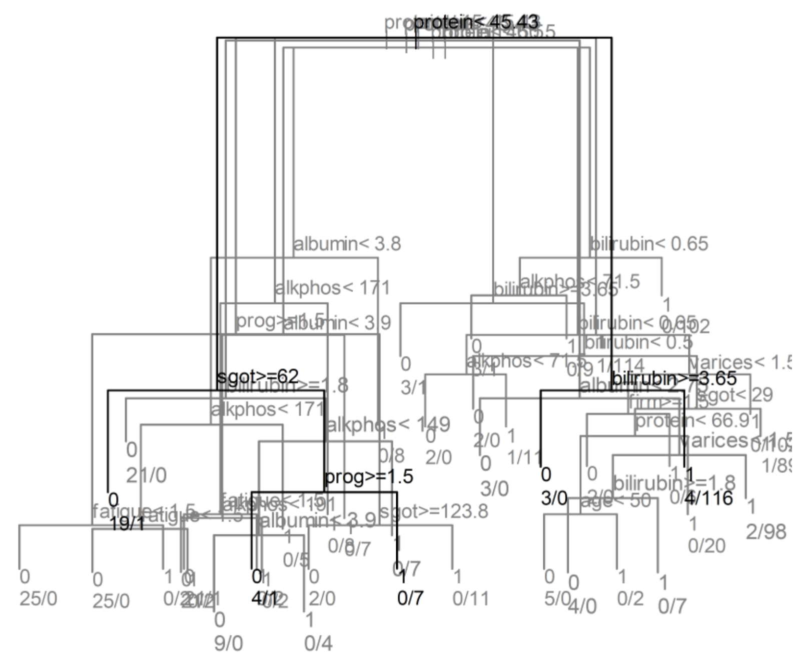


# Decision Tree

- Find criteria to divide a space into many subspaces where each subspace likely fits to a certain class

Classification tree (hepatitis)



# Case Study: Predicting Bug Report Severity (MSR 2010)

- A severity level of a bug report declares to what extent the bug impacts on successful executions of a target program
  - More severe bugs must be managed earlier than less sever ones
  - E.g., https://bugs.eclipse.org/bugs/show_bug.cgi?id=419729

- Bug reports issued by developers/end-users may give inappropriate severity labels, which may result in an obstacle in debugging

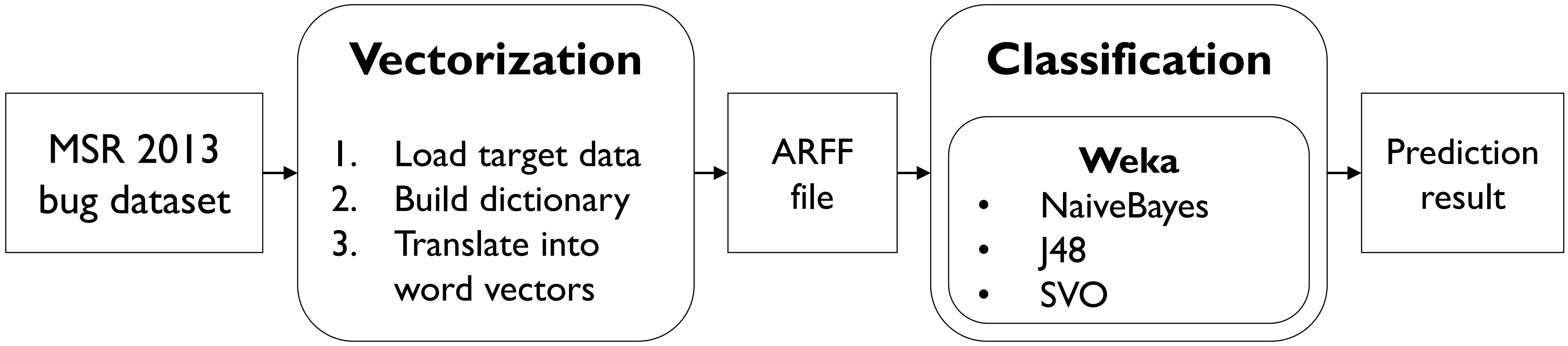
- Train a classifier that predicts the severity of a bug report based on the bug report title (or content)
  - Binary classification

# Data

- MSR 2013 Defect Tracking Benchmark
  - available at https://github.com/ansymo/msr2013-bug_dataset
  - gathered from Mozilla and Eclipse issue trackers
  - compiles all attributes of Bugzilla DB as Xml files
    - E.g. structure of component.xml, severity.xml, short_desc.xml
      - *Report*
        - *Update*
          - *When* ← timestamp at modification (update)
          - *What* ← content

- Use the bug reports found to be on the Layout components in the Mozilla Core module as the study material

# Workflow


MSR 2013
bug dataset
Vectorization
1. Load target data
2. Build dictionary
3. Translate into word vectors
ARFF
file
Classification
Weka
• NaiveBayes
• J48
• SVO
Prediction
result

# Task: Complete TODO's in Vectorinzation

- buildDictionary(descriptions, threshold)
  - dictionary: String → WordIndex
  - convert all characters in a token into lowercase
  - tokenize each description by one of the following delimiters: (, ), /, \, ", ', [, ], ;, :, ,, . , ?, !, <, >, |, ` and **whitespace**
  - reject a token if it appear less than *threshold* of descriptions

- getVector(dictionary, description)
  - tokenize the description as the same as buildDictionary does
  - vector: WordIndex → $\mathbb{R}$
    - Binary
    - Count
    - TF X IDF:
      - TF: # appearance of the word in a desc
      - IDF: log(# descriptions / # description with the word)