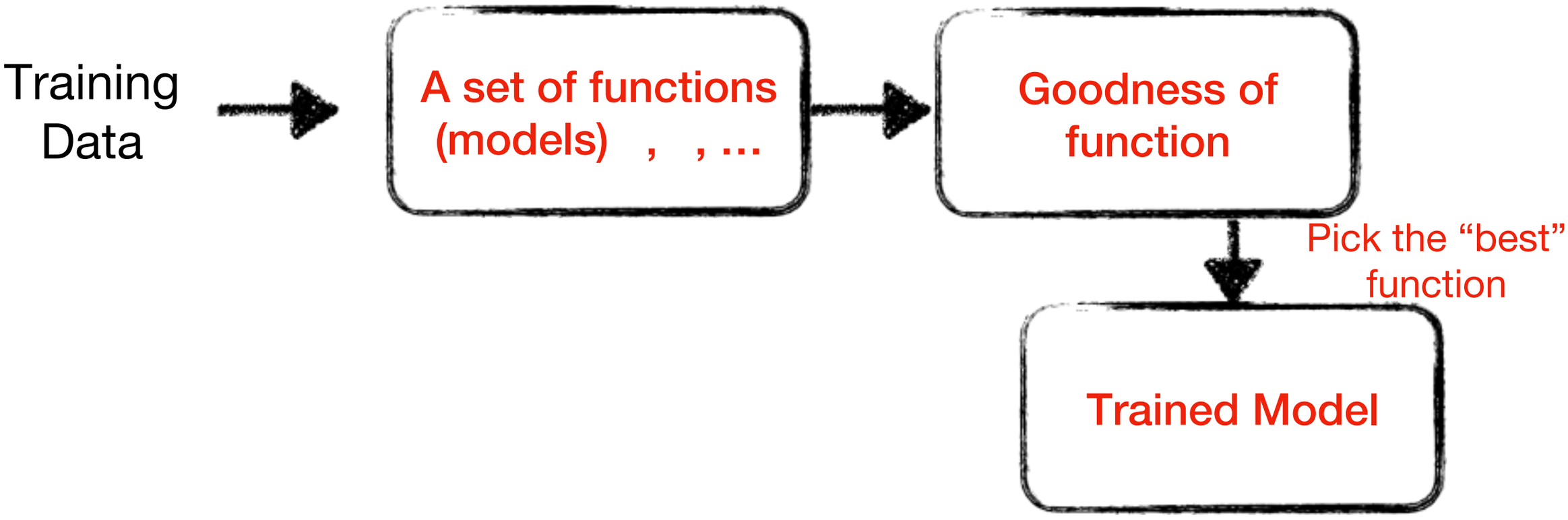
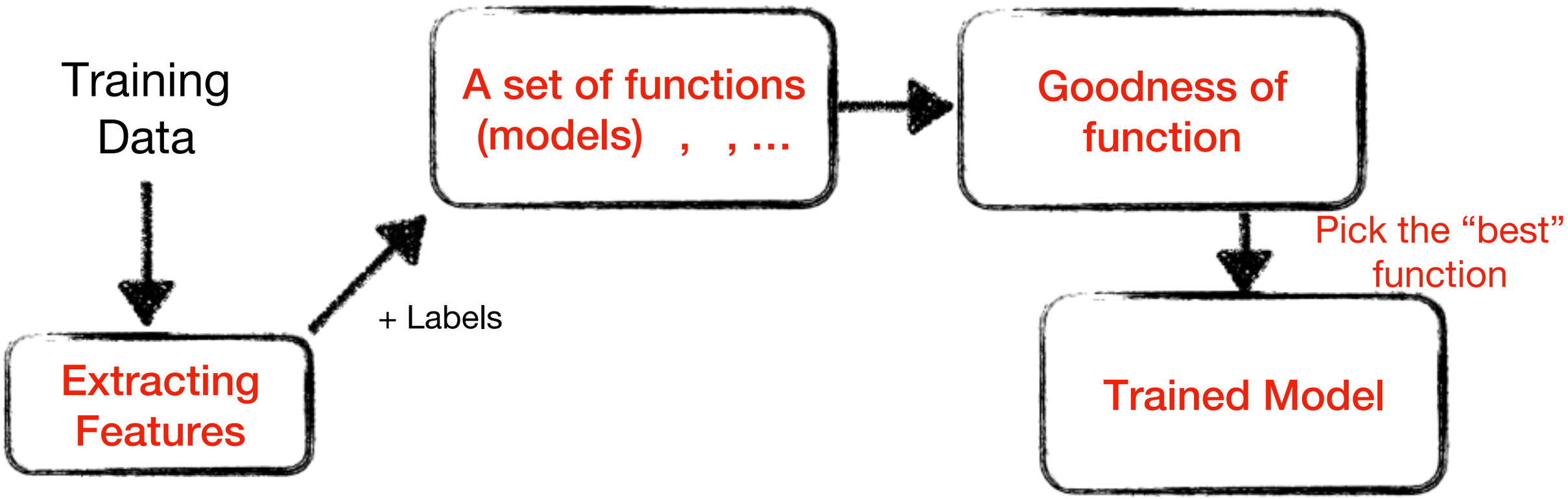
Lecture 3
Feature Extraction

Xu Yuan
University of Louisiana at Lafayette

Data Input = Features Input



Data Input = Feature Input



Real-World Example

When recognizing a person, we compare the face with those stored in memory.

We cannot always remember all the details of a face, but we can still recognize a person.

Previous Example

```
from sklearn import svm

X = [[0, 1],[1, 2],[2, 1],[2, 3],[1, 3],[2, 2]]

y = ['a', 'a', 'b', 'b', 'a', 'b']

clf = svm.SVC()

clf.fit(X, y)

result1 = clf.predict([[3, 1]])

print(result1)

result2 = clf.predict([[0, 2]])

print(result2)
```

Feature representation

```
['b']
['a']
```

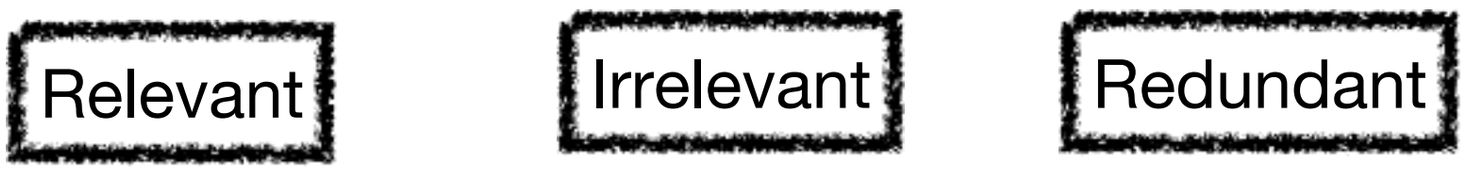
Features

- **A set of attributes with their values can represent a data record**
- **Data record = Feature vectors**
- **With their determinant values, a machine learning model can determine its class**

  - 80% of the time spent on an AI project is wrangling training data, including data labeling
  - Both quality and quantity of training data determine the success of AI

Feature Selection

- **A preprocessing step to choose a subset of original features according to certain criterion**
  - Find the representative data
  - Remove redundant or meaningfulness data
  - Reduce effect of irrelevant data
  - Toward learning accuracy



A feature is good if it is relevant to the class task but is not redundant to any of the other relevant features!

Objective of Feature Selection

- **A preprocessing step to choose a subset of original features according to certain criterion**
  - Avoid overfitting and achieve better generalization ability
  - Improve the prediction performance of the predictors
  - Provide a faster and more cost-effective predictors
  - Provide a better understanding of the underlying process that are generating the data
- **Feature Dimensionality Reduction Technique**
  - Principle Component Analysis (PCA)
  - Independent Component Analysis (ICA)
  - Linear Discriminant Analysis (LDA)
  - Local Linear Embedding (LLE)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
  - Autoencoders
  - https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be

Scikit-learn: VarianceThreshold()

- **This feature selector removes all low-variance features**

```
# if not installed, install sklearn
!pip install sklearn

from sklearn.feature_selection import VarianceThreshold
# dataset with three boolean features
X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0],
[0, 1, 1]]
sel = VarianceThreshold(threshold=(.8 * (1 - .8))) # set
threshold value
sel.fit_transform(x) #Reduce X to the selected features
```

```
array([[0, 1],
       [1, 0],
       [0, 0],
       [1, 1],
       [1, 0],
       [1, 1]])
```

Scikit-learn: VarianceThreshold()

- **This feature selector removes all low-variance features**

```
from sklearn.feature_selection import VarianceThreshold
#Sample dataset integer features, two of which are the same
in every sample
X = [[0, 2, 0, 3], [0, 1, 4, 3], [0, 1, 1, 3]]
selector = VarianceThreshold()
selector.fit_transform(X) #Reduce X to the selected
features
```

```
array([[2, 0],
       [1, 4],
       [1, 1]])
```

Scikit-learn: SelectKBest()

- **Select k features according to the highest scores**

```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2
X, y = load_digits(return_X_y=True)
print(X.shape)
X_new = SelectKBest(chi2, k=20).fit_transform(X, y)# use
chi-squared stats and select k = 20 features
print(X_new.shape)
```

```
(1797, 64)
(1797, 20)
```

Scikit-learn: SelectPercentile()

- **Select features according to a percentile (percent of features to keep) of the highest score**

```
from sklearn.datasets import load_digits # load existing
data
from sklearn.feature_selection import SelectPercentile,
chi2
X, y = load_digits(return_X_y=True)
print(X.shape)

X_new = SelectPercentile(chi2,
percentile=10).fit_transform(X, y)# Percent of features to
keep = 10
print(X_new.shape)
```

```
(1797, 64)
(1797, 7)
```

Scikit-learn: GenericUnivariateSelect()

- **Perform univariate feature selection with a configurable strategy**

```
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import
GenericUnivariateSelect, chi2
X, y = load_breast_cancer(return_X_y=True)
print(X.shape)
```

```
transformer = GenericUnivariateSelect(chi2, mode='k_best',
param=20)# k_best, 20 features. The mode can be any from
the set {‘percentile’, ‘k_best’, ...}.
X_new = transformer.fit_transform(X, y)
print(X_new.shape)
```

```
(569, 30)
(569, 20)
```

Preview the Data

- **Be familiar with the data before deciding the features**

Friends/Followers

- **Number of Following/Followers**

  Following count and Follower count are important social network features.

Tweet JSON

"follow_request_sent":false,
"followed_by":false,
"followers_count":2720235,
"friends_count":793,
"has_custom_timelines":true,
"is_translator":false,

Friends/Followers

- **Number of Following/Followers**

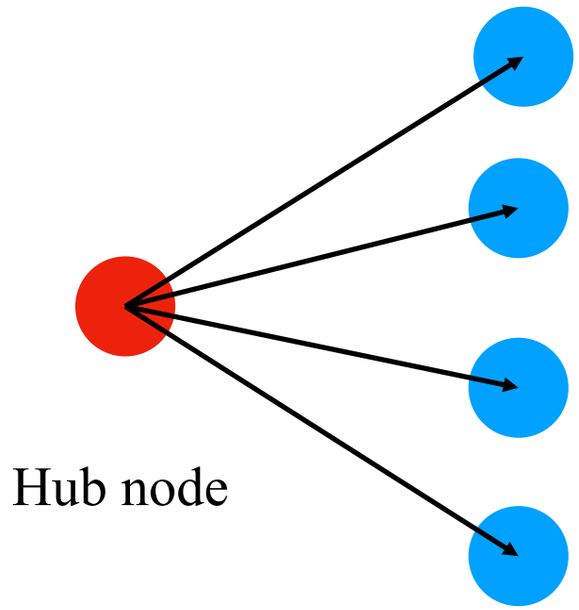
  Different types of accounts.

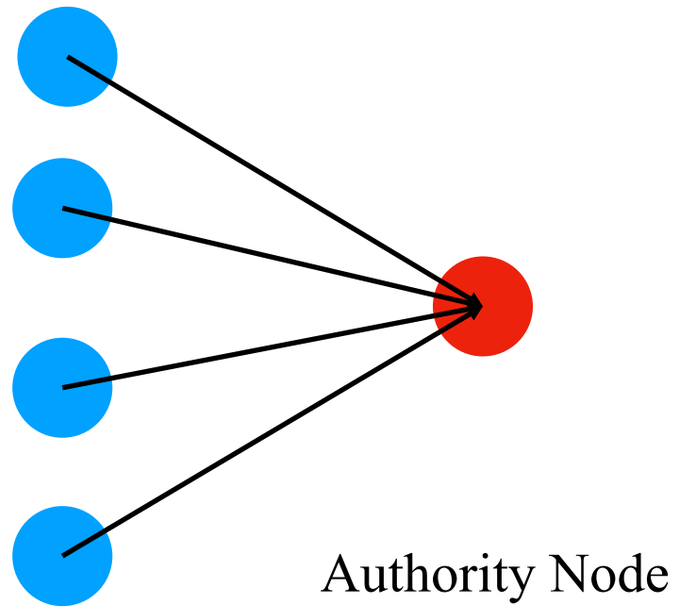
Non-famous Follower count Famous

Friends/Followers

- **Number of Following/Followers**

  Different types of accounts based on Followers count and Following count.

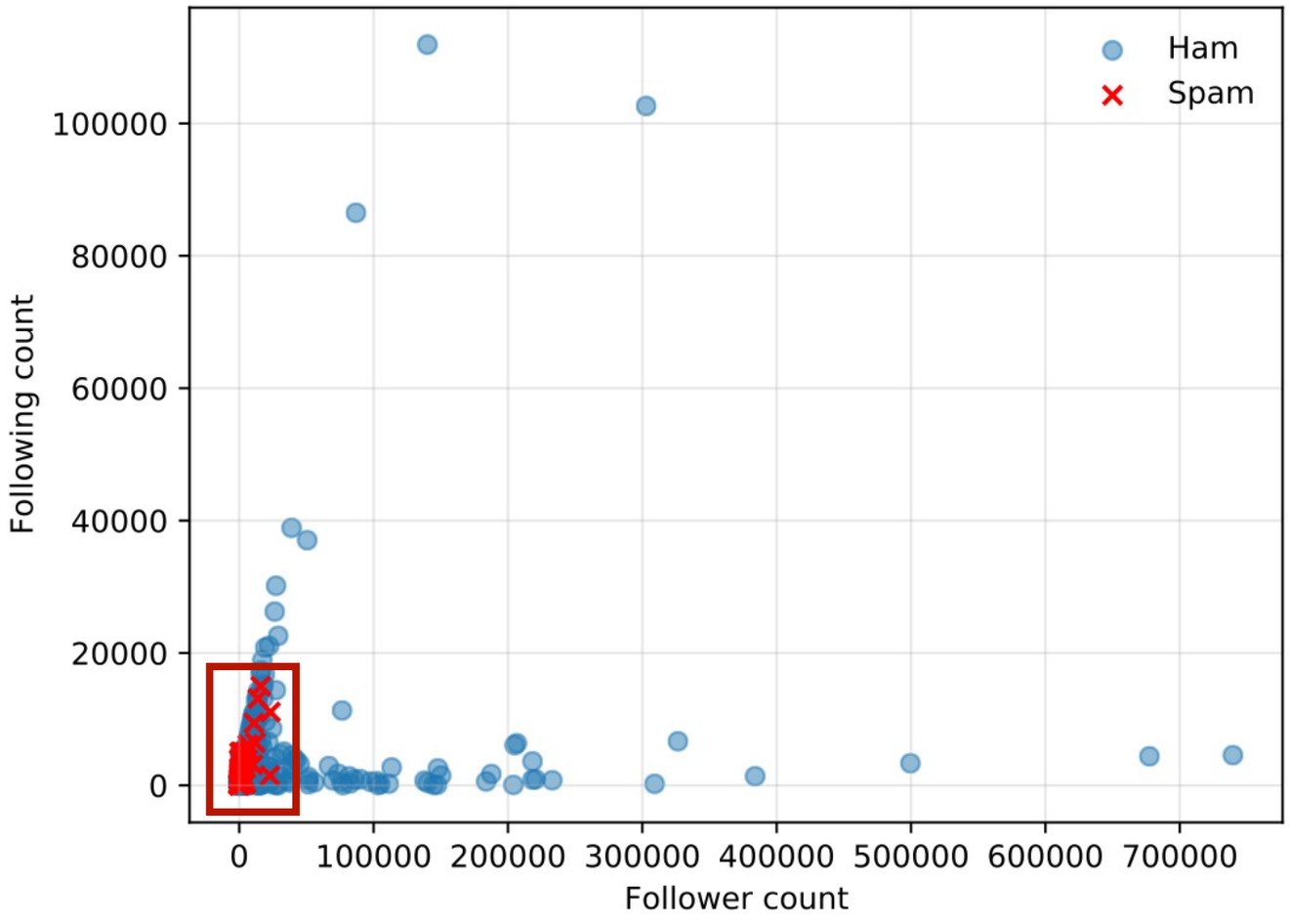




Friends/Followers

- **Number of Following/Followers**

  Followers count and Following count for 1000 randomly sample spams and hams.



Friends/Followers

- **Number of Following/Followers**

  Code example

```
with open(file_path,'r') as read_file:
    for line in read_file:
        tweet_json = json.loads(line)
        tweet_obj = Tweet(line)
        n_friends = tweet_obj.user.get_friends_count()
        n_followers = tweet_obj.user.get_followers_count()
```

```
class Tweet():
    '''
    TwitterTweet class can used to save a tweet
    '''
    def __init__(self, tweet_json):
        self.tweet_json = tweet_json
        self.user = User(tweet_json)
```

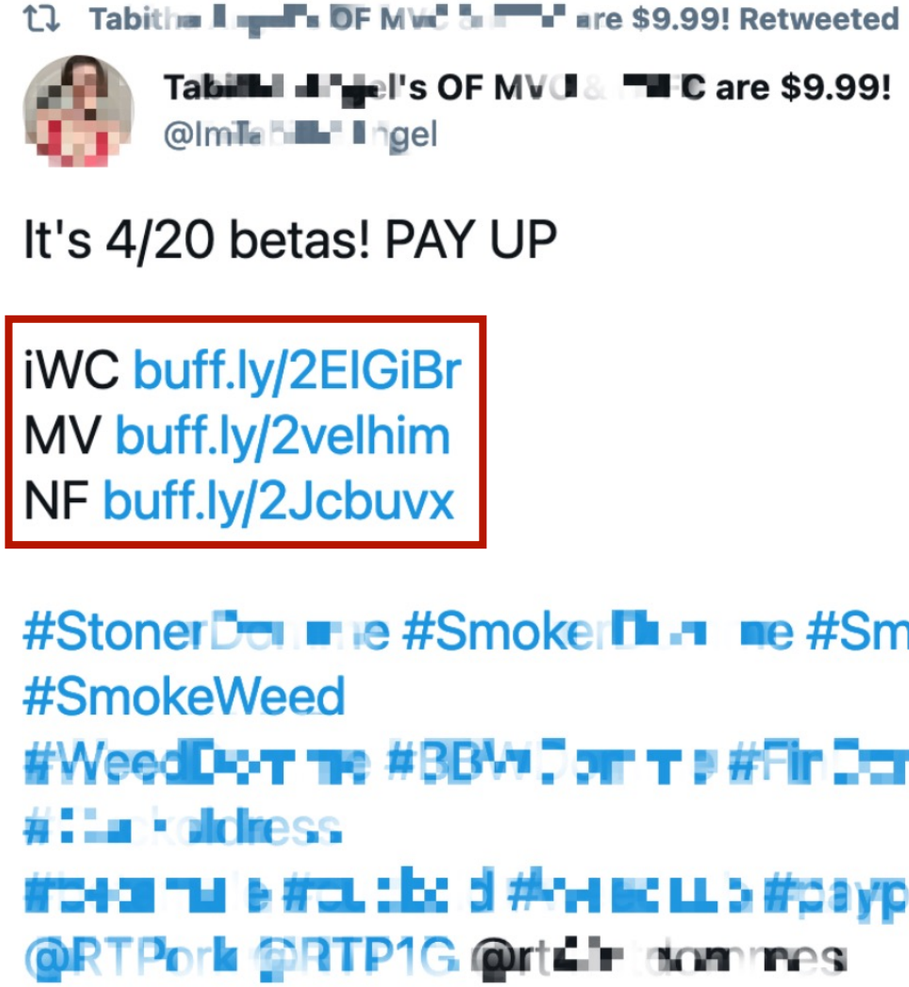
```
def get_followers_count(self):
    '''
    return follower count
    '''
    followers_count = self.user_json['followers_count']
    if followers_count == 0:
        followers_count = 1
    return followers_count
```

```
def get_friends_count(self):
    '''
    return friends count
    '''
    friends_count = self.user_json['friends_count']
    if friends_count == 0:
        friends_count = 1
    return friends_count
```

URLs

- **Number of URLs**

  Spam contains more URLs.



This tweet is a spam

3 URLs

URLs

- **Number of URLs**

3 URLs in JSON

Twitter short URL

External URL, this is a malicious URL

[{'url': 'https://t.co/Bc07wW2S12', 'expanded_url': 'http://onlyfans.com/
[redacted]', 'display_url': 'onlyfans.com/mistressxmaya', 'indices': [49,
72]},

{'url': 'https://t.co/zdI7W21W6w', 'expanded_url': 'http://iwantclips.com/
stor[redacted]', 'display_url': 'iwantclips.com/store/74741/
mi…', 'indices': [73, 96]},

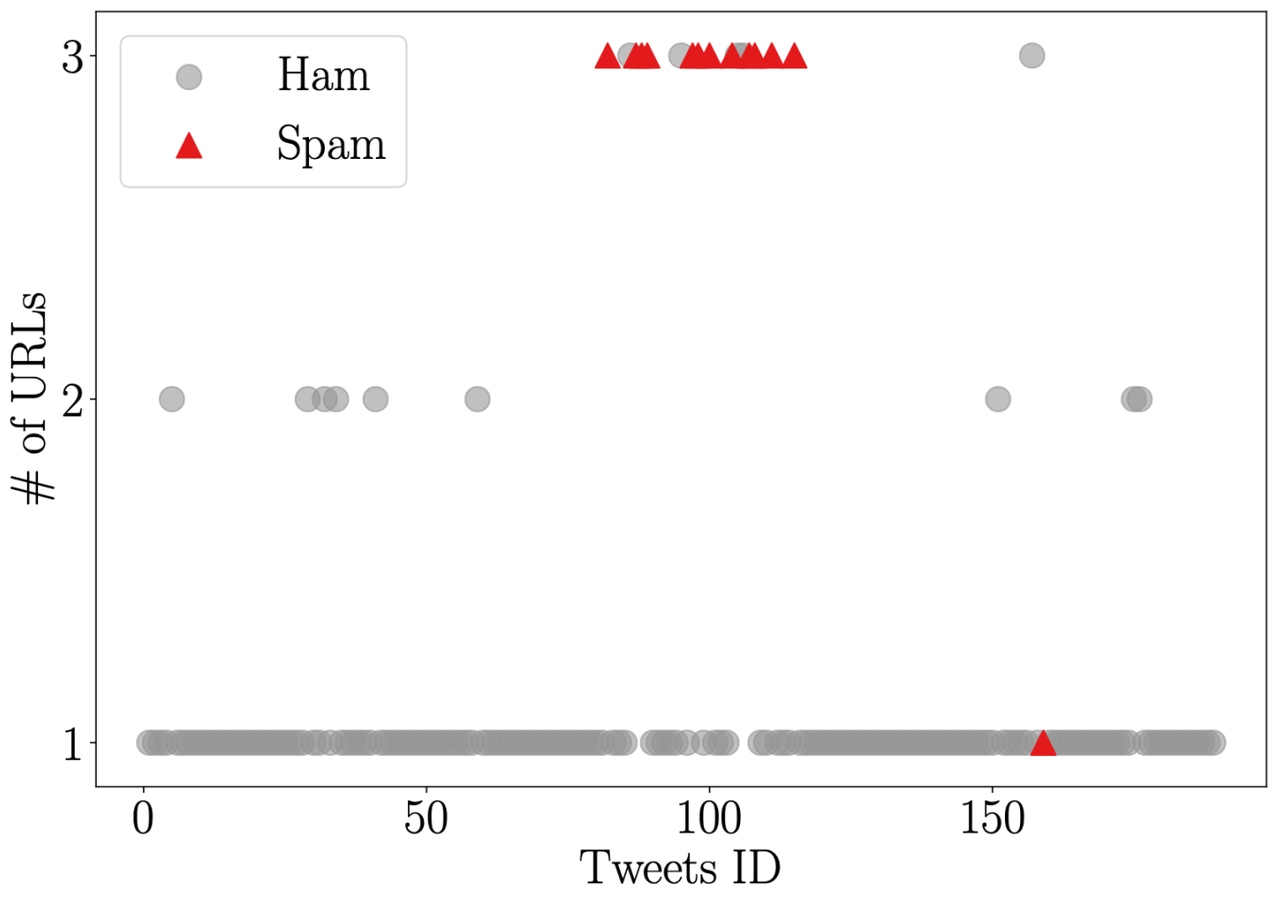
{'url': 'https://t.co/Z89sz24uyc', 'expanded_url': 'https://twitter.com/i/web/
status/987686235010461696', 'display_url': 'twitter.com/i/web/status/9…',
'indices': [116, 139]}]

This account has already been suspended by Twitter!

URLs

- **Number of URLs**

  Spam contains more URLs



URLs

- **Number of URLs**

  Code example

```
with open(file_path,'r') as read_file:
    for line in read_file:
        tweet_json = json.loads(line)
        tweet_obj = Tweet(line)
        n_url = tweet_obj.get_url_count()
```

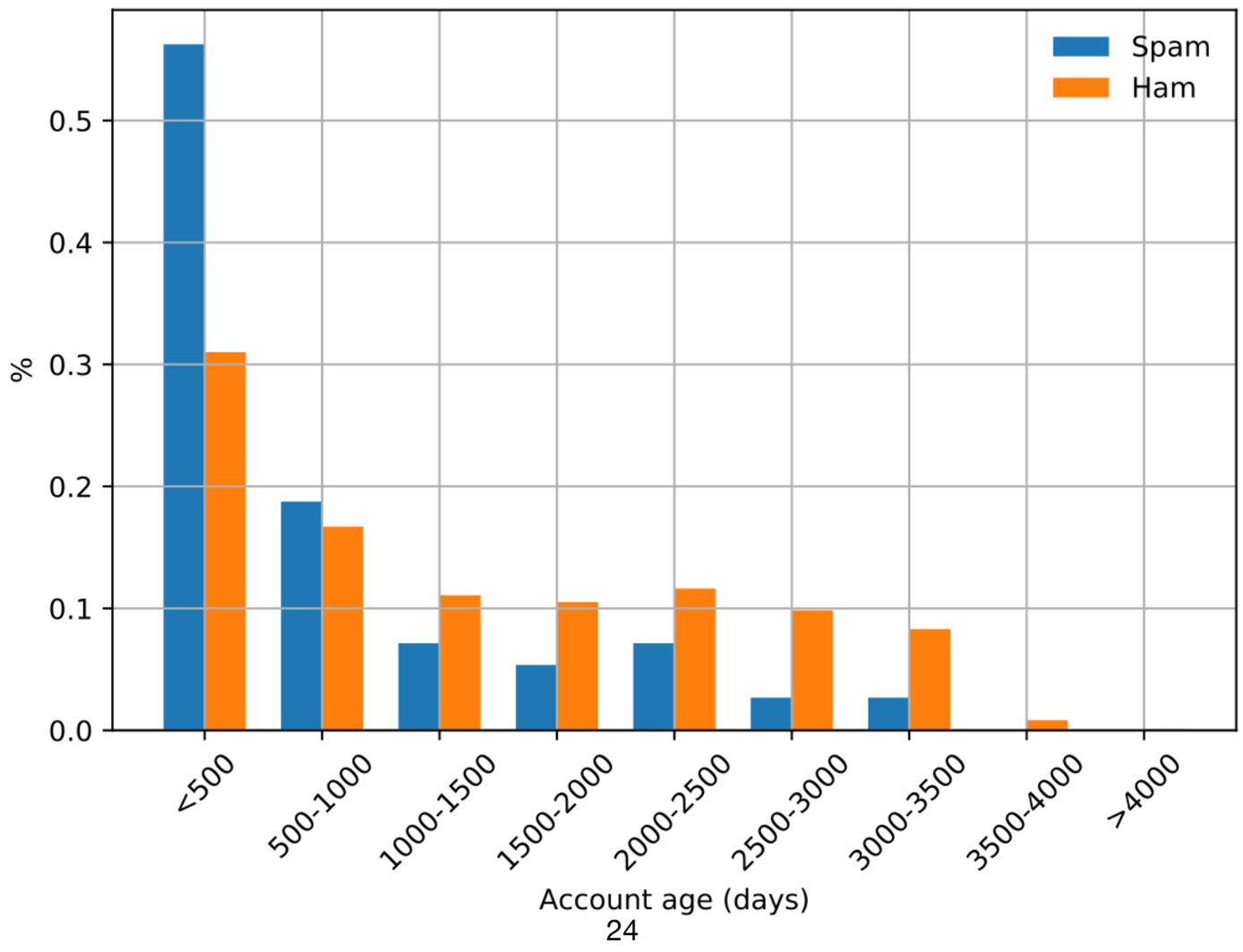
```
class Tweet():
    '''
    TwitterTweet class can used to save a tweet
    '''
    def __init__(self, tweet_json):
        self.tweet_json = tweet_json
        self.user = User(tweet_json)
```

```
    def get_url_count(self):
        '''
        get number of urls
        '''
        urls = self.tweet_json['entities']['urls']
        return len(urls)
```

Account Age

- **Account Age**

  Spam and ham age distribution are different.



Account Age

- **Account Age**

  Code example

```
def get_user_age(self):
    '''
    Age of an account
    get age feature of an account, remember call this function all the time. Time exchange
    '''
    account_start_time = self.json_date_to_os(self.user_json['created_at'])
    now_time = datetime.now()
    account_age = (now_time-account_start_time).days
    if account_age == 0:
        account_age = 1
    return account_age
```

Static Features

- **Account Profile**
  - Friends count, followers count, age, status count, average status, list count, average lists, average favorites, favorites count, verified status, default profile image, screen name length, name length, description length, description emoji count, and description digits count
- **Tweet Content**
  - Tweet status, tweet source platform, hashtag count, mention count, content length content emoji count, and content digits count
- **User Behaviors**
  - Reciprocity count, sender or receiver tweet distribution, sender or receiver source distribution, mention time, average tweet interval, environment score

https://ieeexplore.ieee.org/abstract/document/8809491
https://people.cmix.louisiana.edu/yuan/resources/pdf/19_DSN_Pseudo.pdf

Tweet Features on Lab 3

- **Feature List**

1: User account age	**get_user_age()**
2: The length of user description	**get_description_len()**
3: The number of followers	**get_followers_count()**
4: The number of following	**get_friends_count()**
5: The number of user favorites	**get_user_favorites()**
6: The number of user lists	**get_user_lists()**
7: The number of user statuses	**get_statuses_count()**
8: The number of tweet hashtags	**get_hashtag_count()**
9: The number of tweet mentions	**get_mention_count()**
10: The number of URLs	**get_url_count()**