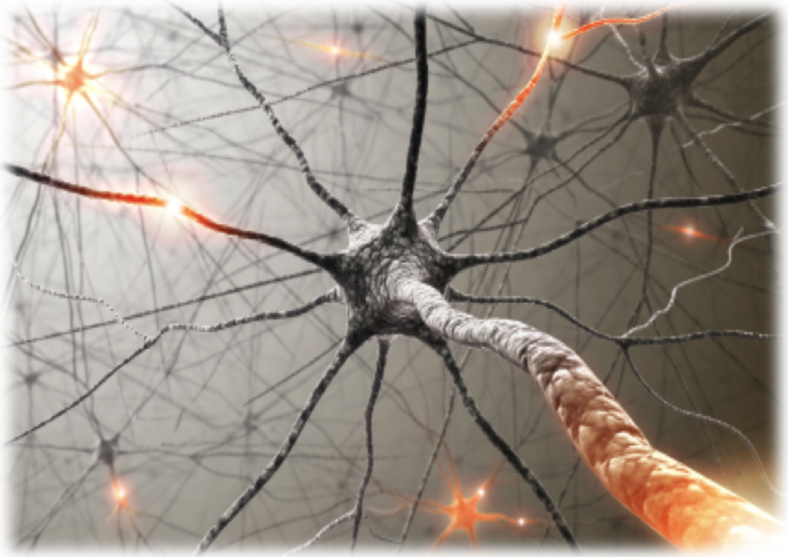
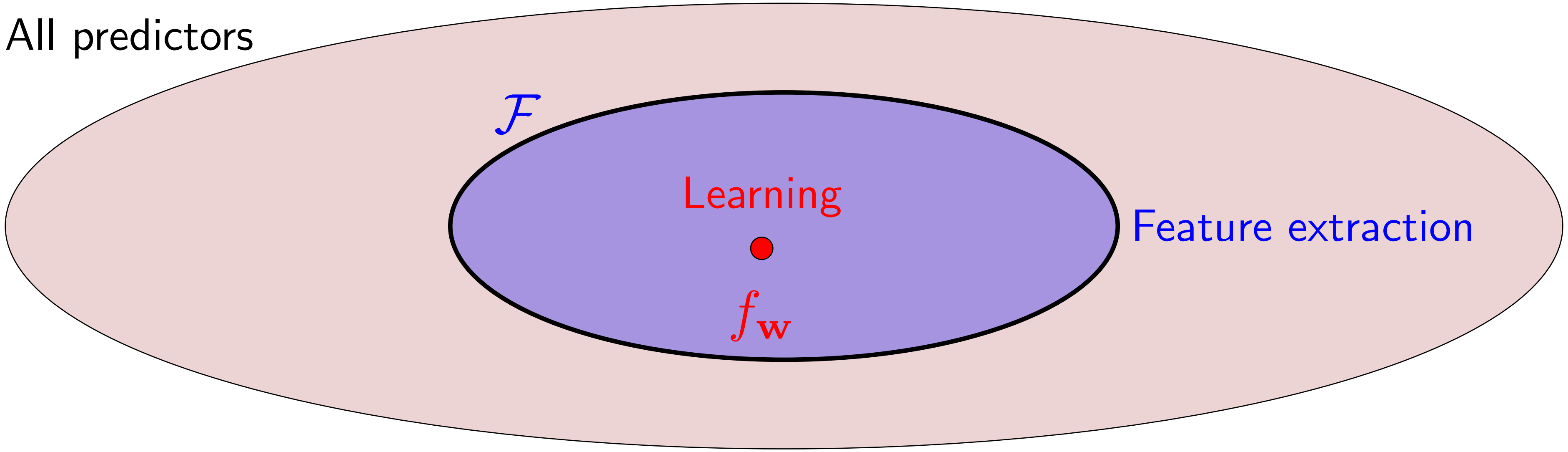
Machine learning: feature templates



Feature extraction + learning

$$\mathcal{F} = \{f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x)) : \mathbf{w} \in \mathbb{R}^d\}$$

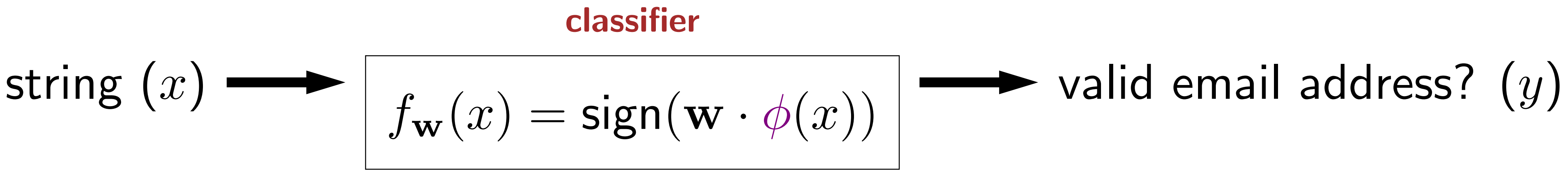


- Feature extraction: choose $\mathcal{F}$ based on domain knowledge
- Learning: choose $f_{\mathbf{w}} \in \mathcal{F}$ based on data

Want $\mathcal{F}$ to contain good predictors but not be too big

Feature extraction with feature names

Example task:

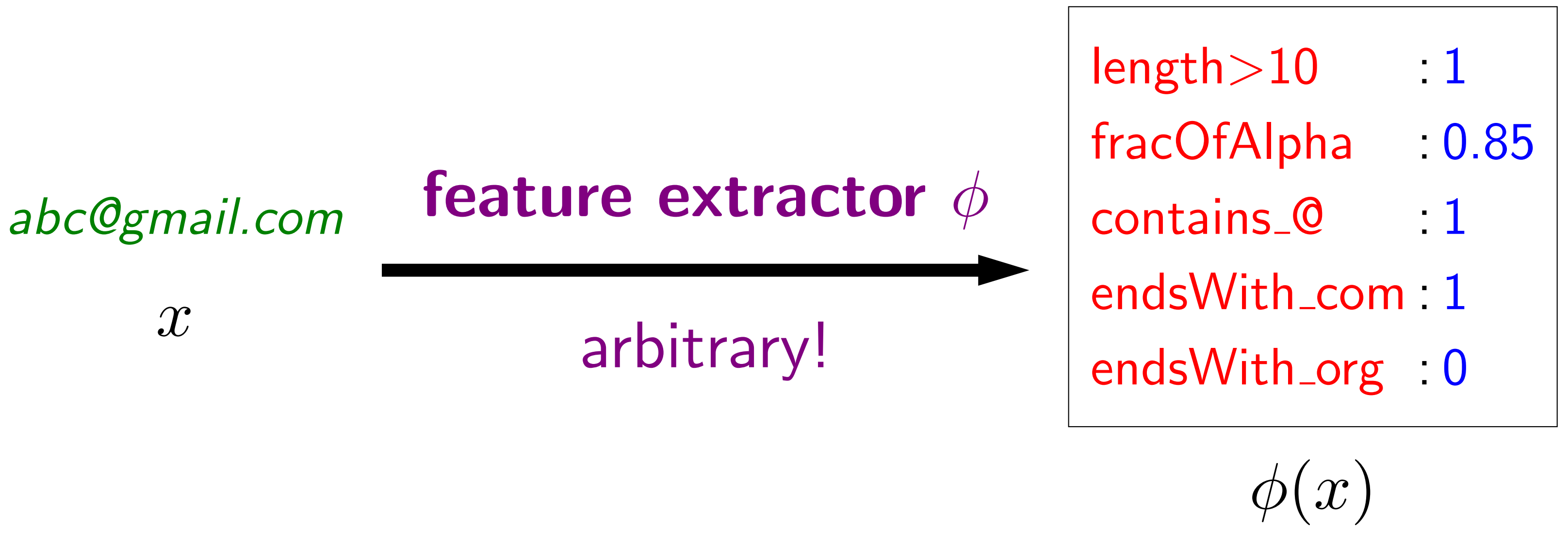


classifier

string (x) → $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$ → valid email address? (y)

Question: what properties of x **might be** relevant for predicting y?

Feature extractor: Given x, produce set of (feature name, feature value) pairs



abc@gmail.com

x

feature extractor ϕ

arbitrary!

length>10	: 1
fracOfAlpha	: 0.85
contains_@	: 1
endsWith_com	: 1
endsWith_org	: 0

$\phi(x)$

Prediction with feature names

Weight vector $\mathbf{w} \in \mathbb{R}^d$

length>10	:-1.2
fracOfAlpha	:0.6
contains_@	:3
endsWith_com	:2.2
endsWith_org	:1.4

Feature vector $\phi(x) \in \mathbb{R}^d$

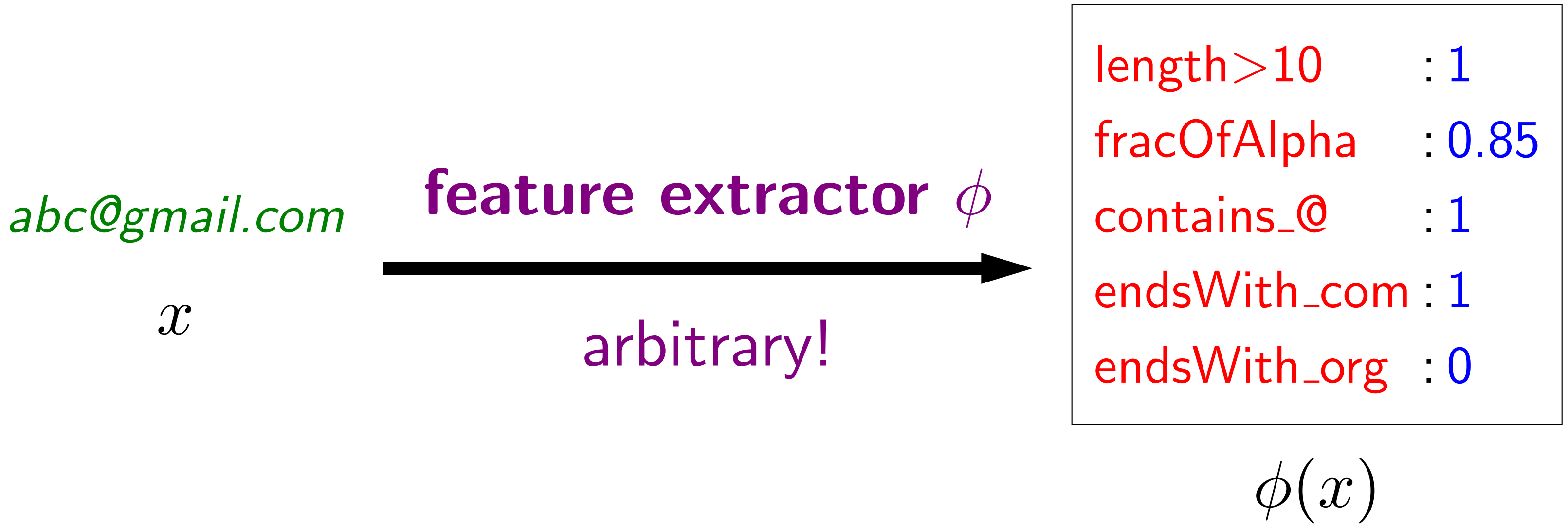
length>10	:1
fracOfAlpha	:0.85
contains_@	:1
endsWith_com	:1
endsWith_org	:0

Score: weighted combination of features

$$\mathbf{w} \cdot \phi(x) = \sum_{j=1}^{d} w_j \phi(x)_j$$

Example: $-1.2(1) + 0.6(0.85) + 3(1) + 2.2(1) + 1.4(0) = 4.51$

Organization of features?



abc@gmail.com

x

feature extractor ϕ

arbitrary!

length>10	: 1
fracOfAlpha	: 0.85
contains_@	: 1
endsWith_com	: 1
endsWith_org	: 0

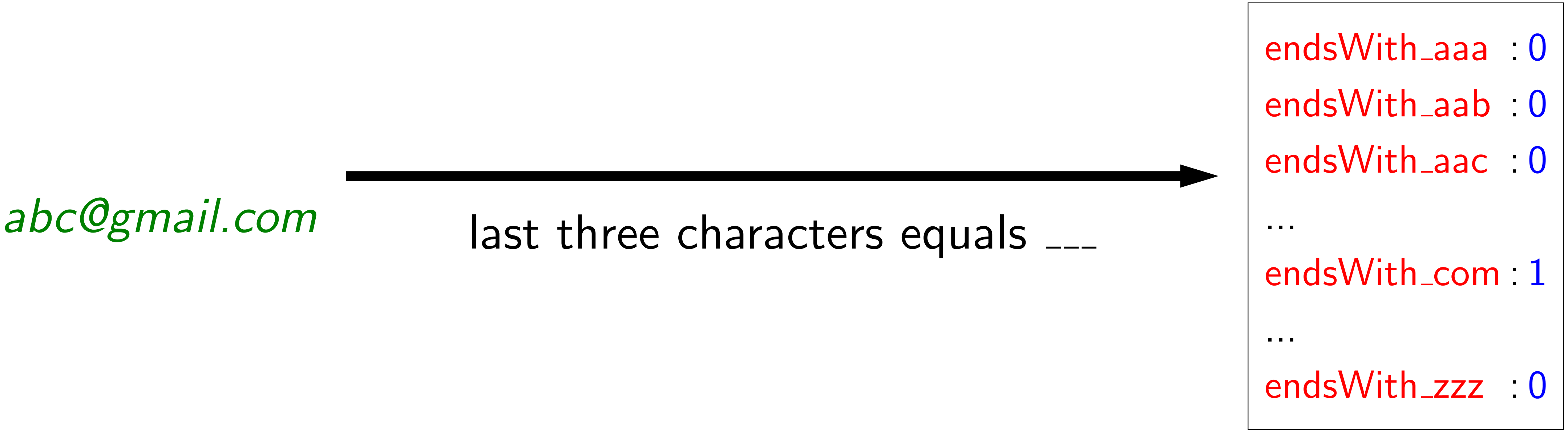
$\phi(x)$

Which features to include? Need an organizational principle...

Feature templates

Definition: feature template

A **feature template** is a group of features all computed in a similar way.



abc@gmail.com

last three characters equals ___

endsWith_aaa : 0
endsWith_aab : 0
endsWith_aac : 0
...
endsWith_com : 1
...
endsWith_zzz : 0

Define types of pattern to look for, not particular patterns

Feature templates example 1

Input:

abc@gmail.com

Feature template	Example feature	
Last three characters equals ___	Last three characters equals *com*	: 1
Length greater than ___	Length greater than *10*	: 1
Fraction of alphanumeric characters	Fraction of alphanumeric characters	: 0.85

Feature templates example 2

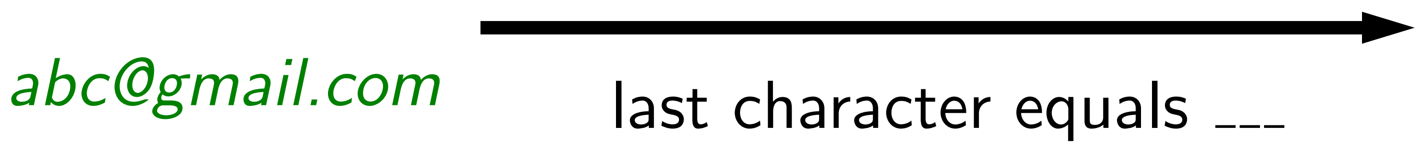
Input:

Latitude: 37.4068176

Longitude: -122.1715122

Feature template	Example feature name	
Pixel intensity of image at row ___ and column ___ (___ channel)	Pixel intensity of image at row *10* and column *93* (*red* channel)	: 0.8
Latitude is in [___, ___] and longitude is in [___, ___]	Latitude is in [*37.4*, *37.5*] and longitude is in [*-122.2*, *-122.1*]	: 1

Sparsity in feature vectors



abc@gmail.com

endsWith_a	: 0
endsWith_b	: 0
endsWith_c	: 0
endsWith_d	: 0
endsWith_e	: 0
endsWith_f	: 0
endsWith_g	: 0
endsWith_h	: 0
endsWith_i	: 0
endsWith_j	: 0
endsWith_k	: 0
endsWith_l	: 0
endsWith_m	: **1**
endsWith_n	: 0
endsWith_o	: 0
endsWith_p	: 0
endsWith_q	: 0
endsWith_r	: 0
endsWith_s	: 0
endsWith_t	: 0
endsWith_u	: 0
endsWith_v	: 0
endsWith_w	: 0
endsWith_x	: 0
endsWith_y	: 0
endsWith_z	: 0

Compact representation:

```
{"endsWith_m": 1}
```

Two feature vector implementations

Arrays (good for dense features):

pixelIntensity(0,0)	:	0.8
pixelIntensity(0,1)	:	0.6
pixelIntensity(0,2)	:	0.5
pixelIntensity(1,0)	:	0.5
pixelIntensity(1,1)	:	0.8
pixelIntensity(1,2)	:	0.7
pixelIntensity(2,0)	:	0.2
pixelIntensity(2,1)	:	0
pixelIntensity(2,2)	:	0.1

```
[0.8, 0.6, 0.5, 0.5, 0.8, 0.7, 0.2, 0, 0.1]
```

Dictionaries (good for sparse features):

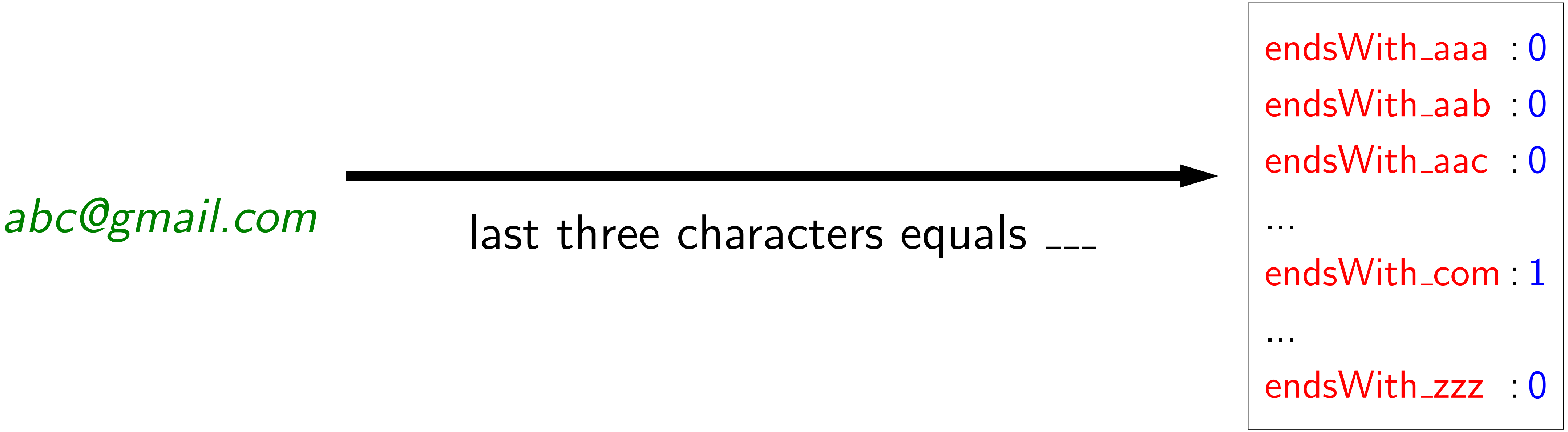
fracOfAlpha	:	0.85
contains_a	:	0
contains_b	:	0
contains_c	:	0
contains_d	:	0
contains_e	:	0
...		
contains_@	:	1
...		

```
{"fracOfAlpha": 0.85, "contains_@": 1}
```

Summary

$$\mathcal{F} = \{f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x)) : \mathbf{w} \in \mathbb{R}^d\}$$

Feature template:



abc@gmail.com

last three characters equals ___

endsWith_aaa : 0
endsWith_aab : 0
endsWith_aac : 0
...
endsWith_com : 1
...
endsWith_zzz : 0

Dictionary implementation:

```
{"endsWith_com": 1}
```