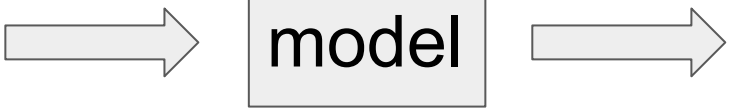
raw tabular data semantical mapping

first of all, what we want?

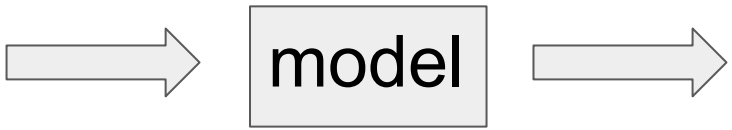
12/21/1970
6/19/2003
9/12/1941
6/15/1944
6/5/1981
3/17/1936
12/30/1957
10/9/2009
11/13/1970
6/27/2018



date: 0.8
country: 0.15
name: 0.01
city: 0.01
age: 0.01
. . .

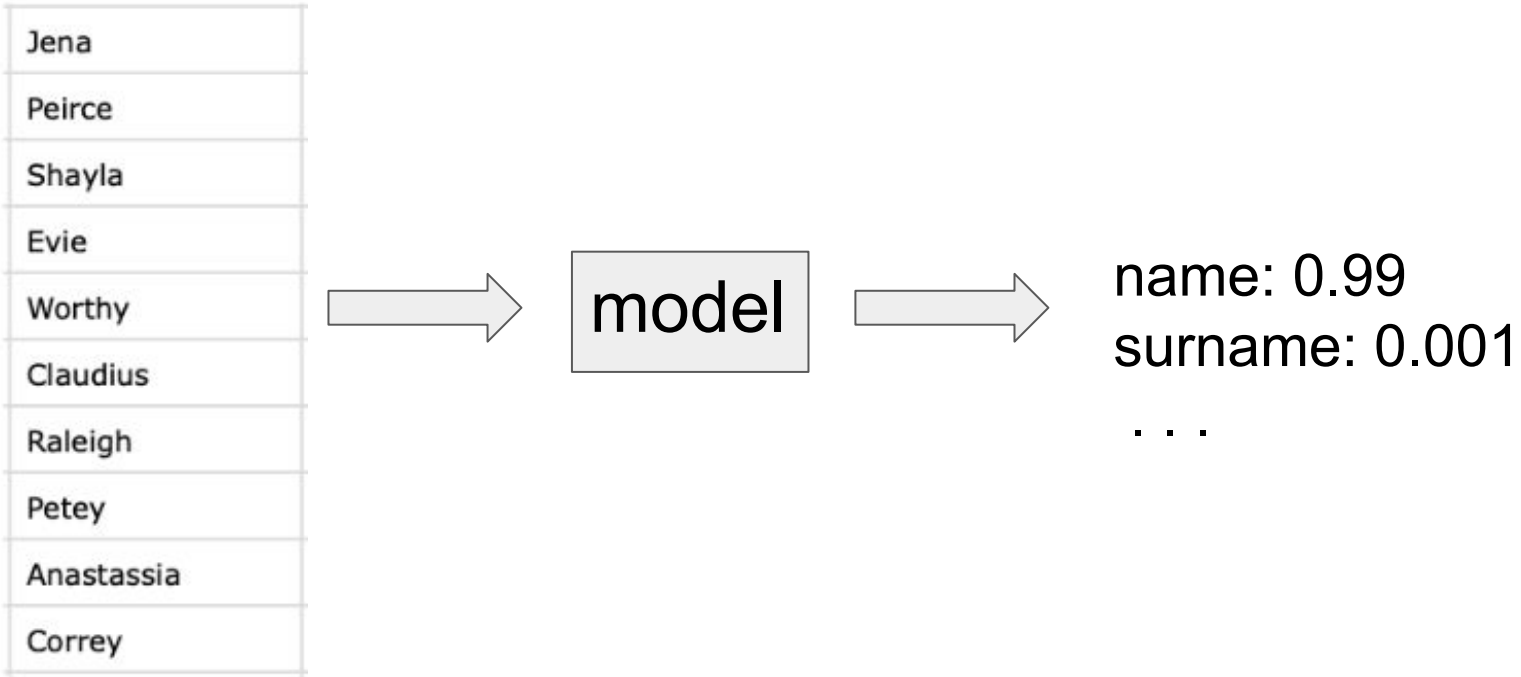
what we want? (another example)

Washington
Hollywood
Fort Wayne
Cincinnati
New York City
Fort Wayne
Portland
Wilmington
Anaheim
Arlington



city: 0.9
country: 0.09
name: 0.01
age: 0.01
. . .

what we want? (last example)



what we want? (more general)

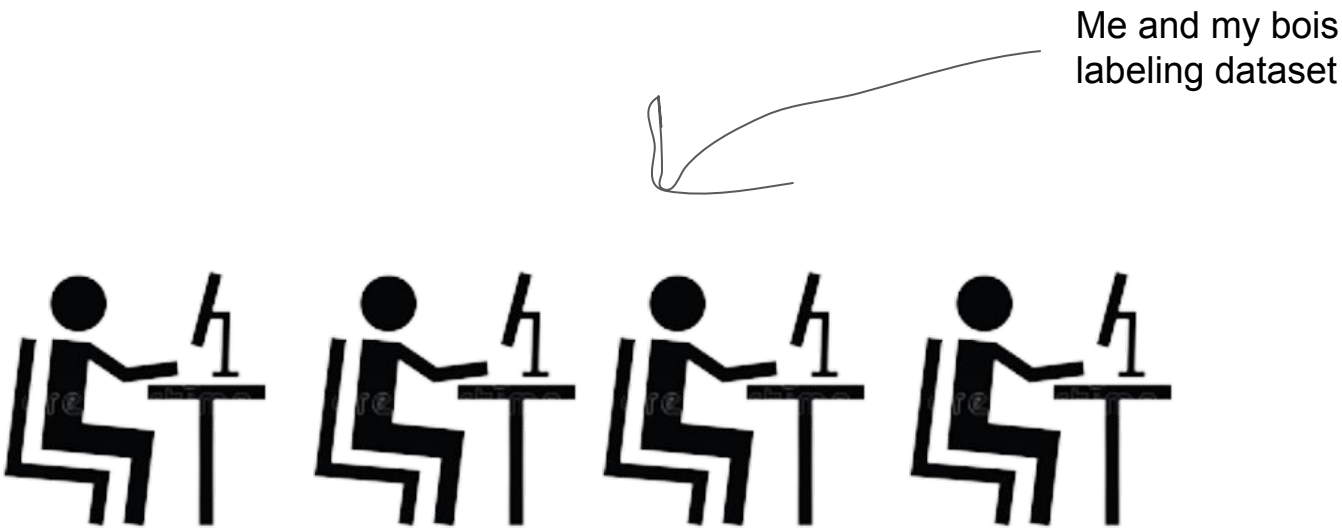
input: array of strings  output: what do those strings mean?

what we want? (more philosophical)

syntax  semantics

data

Initially, there was an attempt to create training dataset from scratch ..



all categories

address
album
birth Place
city
code
component
creator
depth
education
format
industry
location
notes
origin
position
rank
requirement
sex
symbol
weight

affiliate
area
brand
class
collection
continent
credit
description
elevation
gender
isbn
manufacturer
operator
owner
product
ranking
result
species
team
year

affiliation
artist
capacity
classification
command
country
currency
director
family
genre
jockey
name
order
person
publisher
region
sales
state
team Name

age
birth Date
category
club
company
county
day
duration
file Size
grades
language
nationality
organisation
plays
range
religion
service
status
type

data

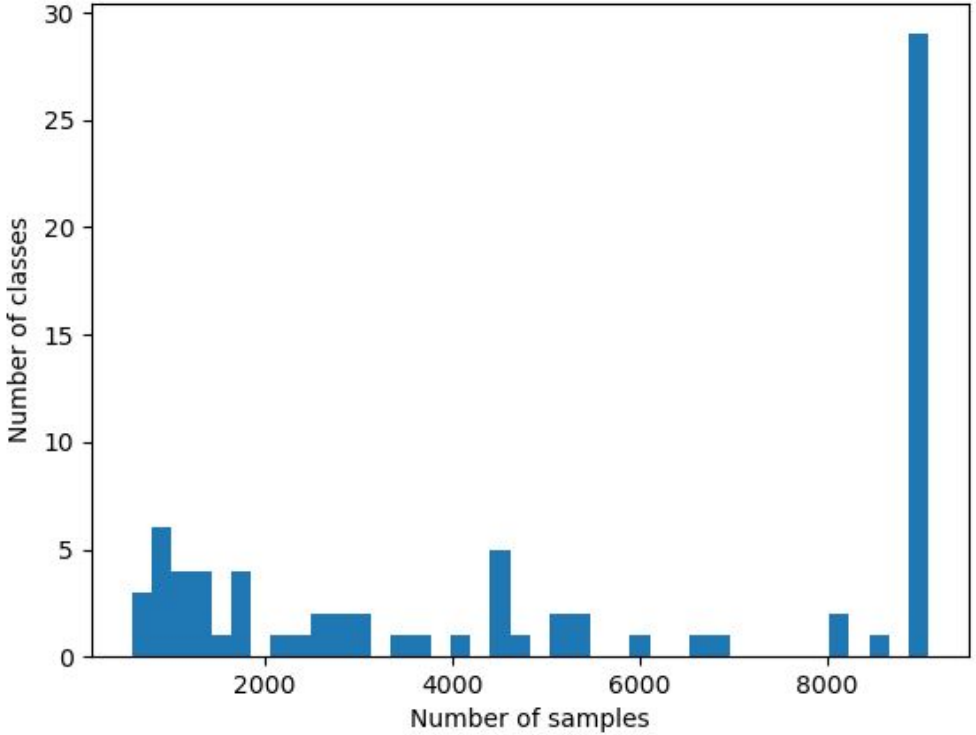
so we will use parsed VizNet data ☆

VizNet is **a centralized and large-scale repository of data as used in practice, compiled from the web, open data repositories, and online visualization platforms**. VizNet enables data scientists and visualization researchers to aggregate data, enumerate visual encodings, and crowdsource effectiveness evaluations. ☆

data, how we parse it?

PAT_CITY	PAT_STATE	PAT_ADDRESS	PAT_ZIP
Fort Lauderdale	FL	2315 Esch Park	33355
Sioux Falls	SD	28055 Westend Trail	57198
Stockton	CA	29013 Magdeline Court	95210
Albuquerque	NM	52243 Orin Hill	87195
Corona	CA	40 Bonner Avenue	92878
Mobile	AL	9410 Oxford Plaza	36622
Milwaukee	WI	0 Manufacturers Plaza	53220
Montgomery	AL	4 Stone Corner Road	36125
Alexandria	VA	5 Carey Alley	22333
Montgomery	AL	372 Jenna Street	36114
Houston	TX	56409 Delladonna Plaza	77255

distribution of number of samples for each category



how we fit data into model?

```
3:49
4:01
0:00
0:00
3:16
3:28
0:00
3:58
4:28
3:32
3:28
0:00
3:02
3:22
3:33
```

array of strings

```
length_min      0.0
length_max      39.0
length_mean     25.53846153846154
length_std      11.321289574652532
numeric_min     0.0
numeric_max     0.0
numeric_mean    0.0
numeric_std     0.0
alphabetic_min  0.0
alphabetic_max  0.9230769230769231
  . . .
```

vector of features

model

what features we extract?

- lengths of elements
- percentage of alphabet letters
- percentage of digits
- how often each character occurs
- how often each character occurs on fixed position
- percentage of empty (nan or None) elements
- uniqueness

We then take a bunch of stats about each of those: mean, min, max, standard deviation, skewness, etc ☆

what models did we try?

Random forest

20 trees

Accuracy: 0.8277

Neural network

3 layers
1024 neurons on each layer
dropout 0.1

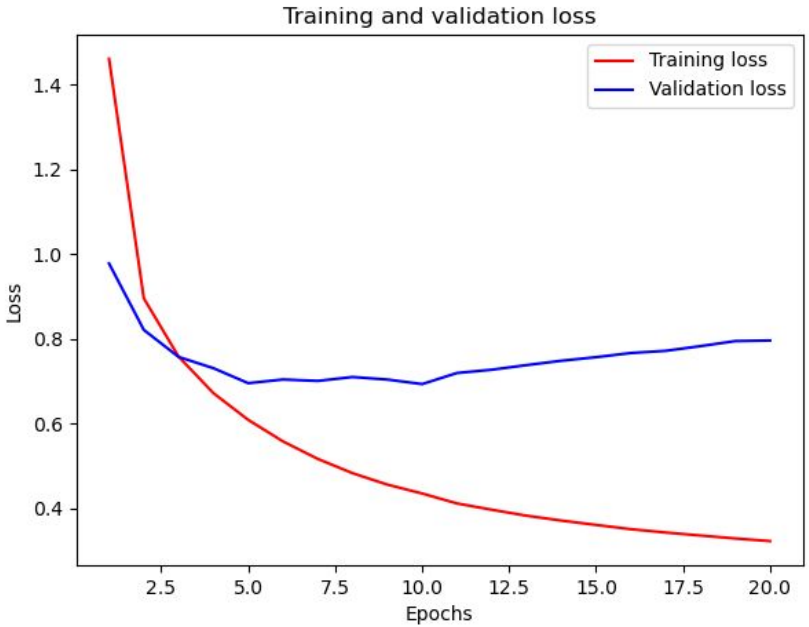
Accuracy: ~0.84

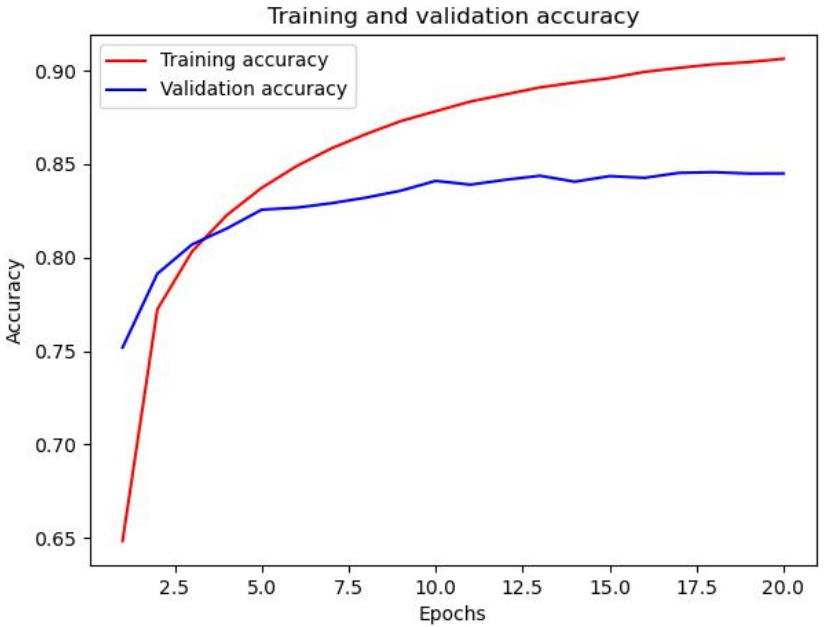
XGBoost

100 trees
depth 10

Accuracy: ~0.5 :(

learning curves for neural network



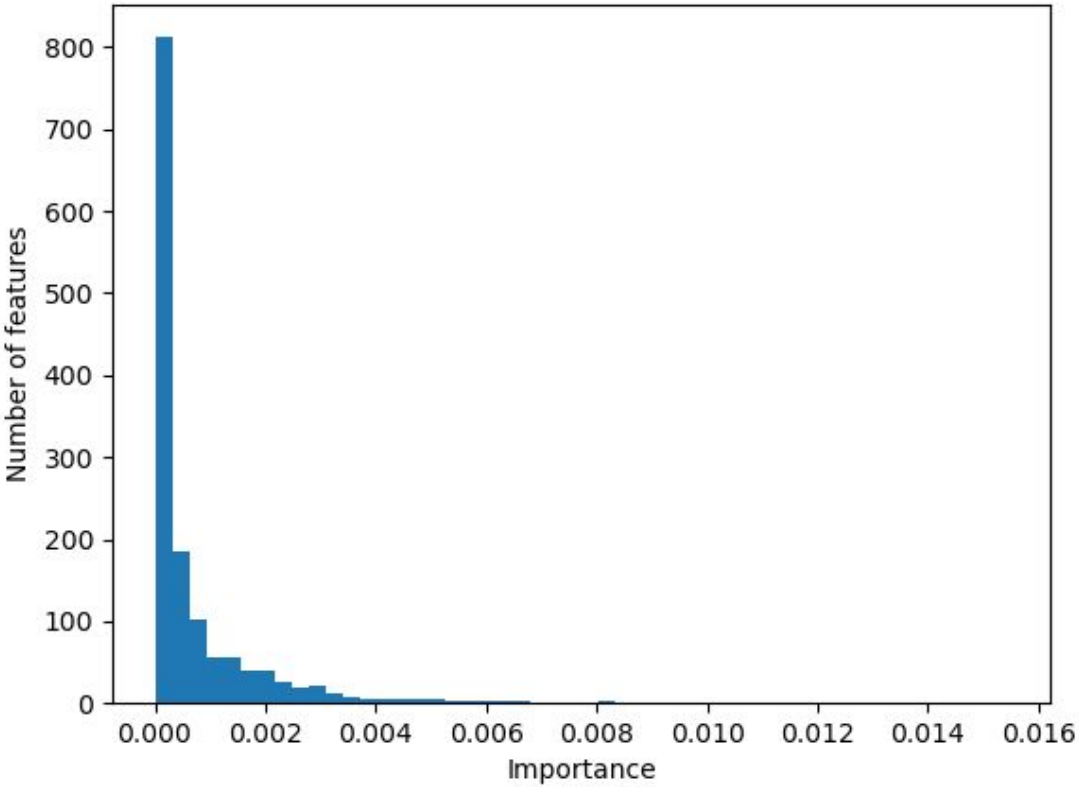


what are the most important features?

```
length_mean                    : 0.0154
 _occurrence_mean☆             : 0.0103
length_max                     : 0.0098
uniqueness                     : 0.0094
uppercase_mean                 : 0.0092
uppercase_min                  : 0.0082
length_min                     : 0.0081
alphabetic_mean                : 0.0080
 _occurrence_max☆              : 0.0079
numeric_max                    : 0.0074
```

☆ nothing in the beginning means ‘ ‘, space character

features importances distribution



Not good, obviously.
Nearly 80% of features are useless :(

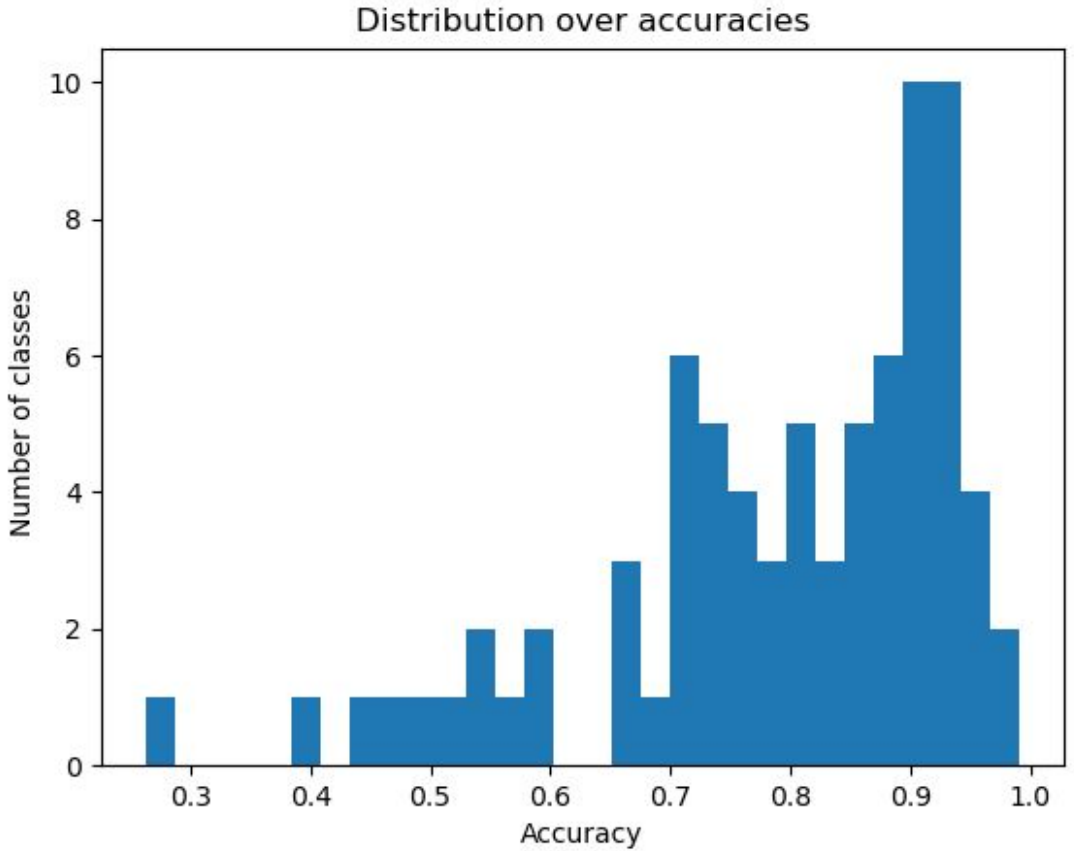
classes that were classified good

grades	0.991
isbn	0.986
birth Date	0.964
elevation	0.954
symbol	0.952
industry	0.950
age	0.937
year	0.936
duration	0.932
affiliation	0.927
format	0.926
sex	0.926

classes that were classified bad

publisher	0.659
affiliate	0.656
capacity	0.651
nationality	0.599
team Name	0.589
credit	0.557
manufacturer	0.540
name	0.538
operator	0.509
person	0.493
sales	0.468
brand	0.452
ranking	0.405
director	0.262

distribution of accuracies among all classes



how the result looks like

```
Values    :  'Central Missouri', 'unattached', 'unattached', 'Kansas Sta . . .
Predicted :  {'affiliation': 0.3, 'country': 0.2, 'category': 0.2}
Truth     :  affiliation

Values    :  95, 100, 95, 89, 84, 91, 88, 94, 75, 78, 90, 84, 90, 76, 93 . . .
Predicted :  {'rank': 0.3, 'plays': 0.3, 'education': 0.2}
Truth     :  weight

Values    :  'Katie Crews', 'Christian Hiraldo', 'Alex Estrada', 'Fredy  . . .
Predicted :  {'jockey': 0.9, 'owner': 0.1, 'year': 0.0}
Truth     :  jockey

Values    :  'Christian', 'Non-Christian', 'Unreported', 'Jewish', 'Athe . . .
Predicted :  {'type': 0.2, 'language': 0.1, 'name': 0.1}
Truth     :  religion

Values    :  'AAF-McQuay Canada Inc.', 'AAF-McQuay Canada Inc.', 'Abilit . . .
Predicted :  {'company': 0.3, 'album': 0.2, 'description': 0.1}
Truth     :  company
```

what I learned from doing this project

- Preparing and preprocessing data is 90% of the work
- Making a decent dataset by yourself is hard
- Extracting and training sometimes takes hours so always do a backup
- Plots and graphs can really help
- Don’t reinvent the wheel when it’s possible

let’s give it a test!

github repo with all stuff:

https://github.com/rureirureirurei/cat