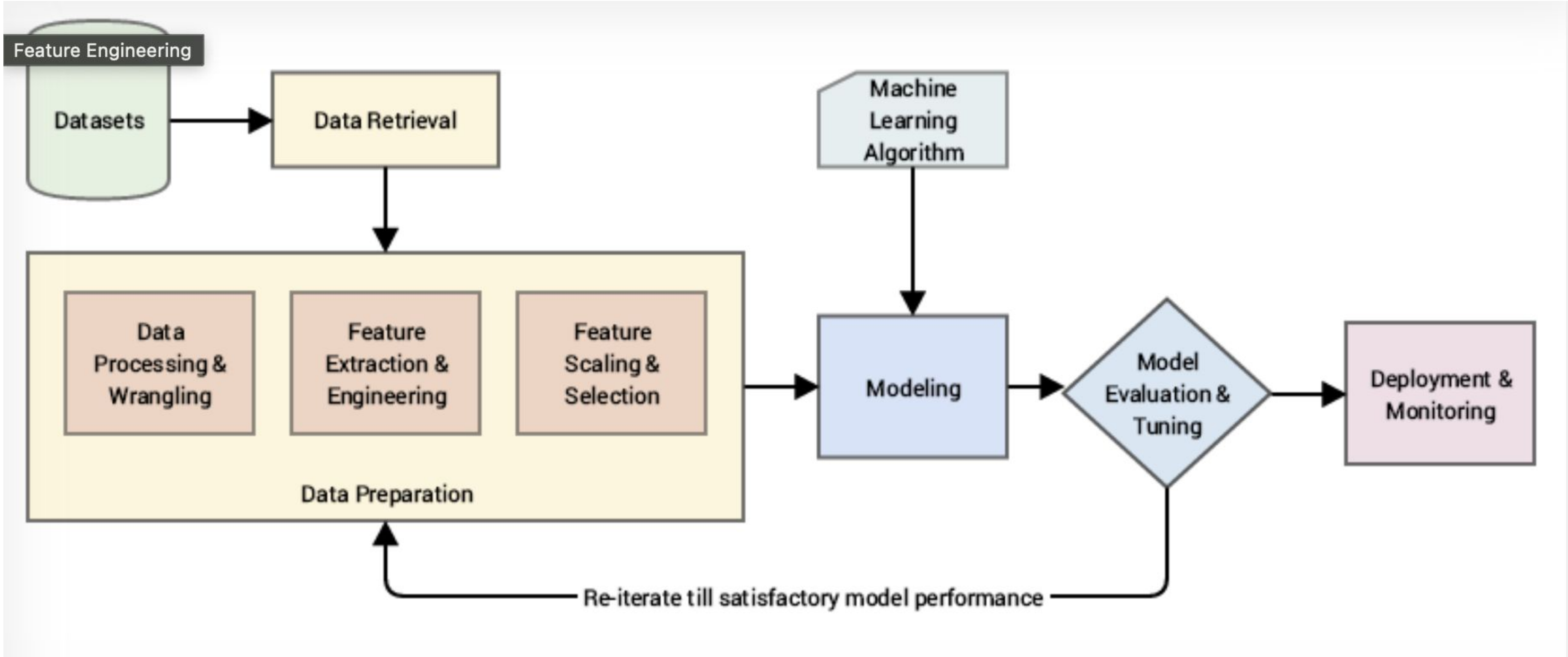
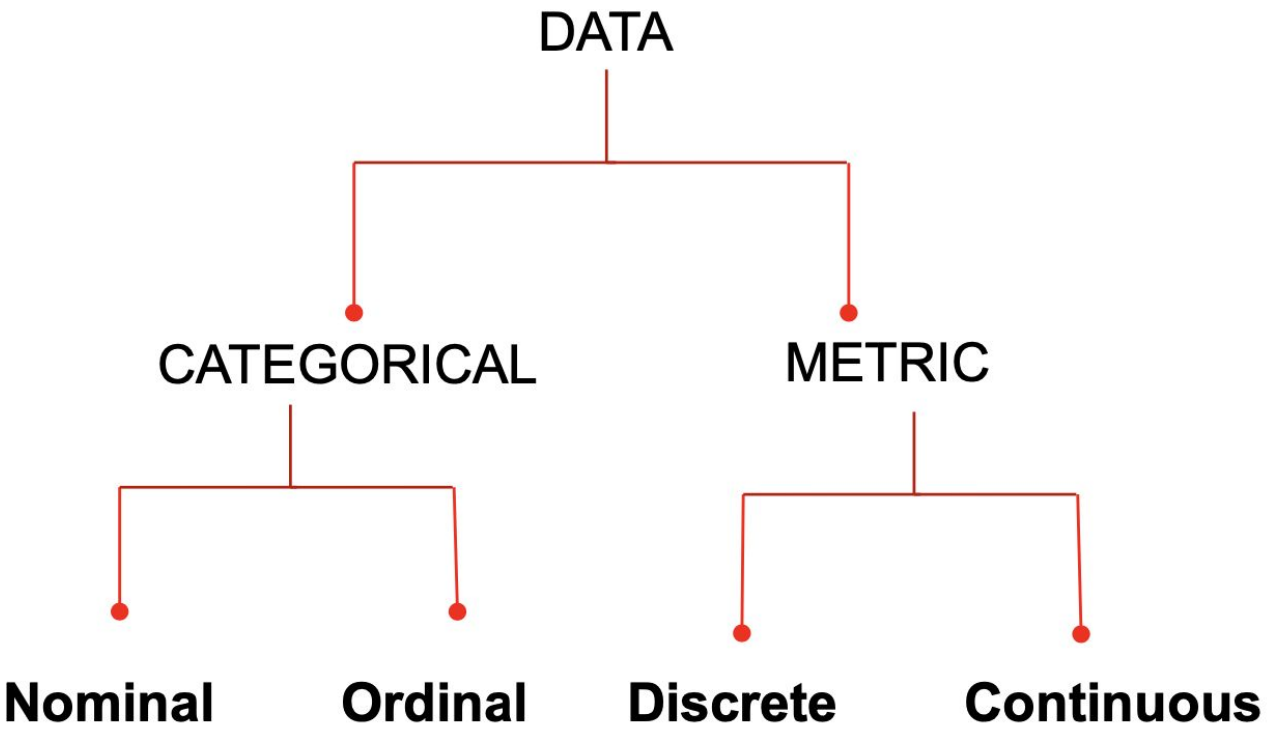
Feature Engineering

Machine Learning Model development Process



Fundamental Data types



Example 1 (revisited)

	A	B	C	D
1	Participan	Gender	Hair Color	Height (m)
2	1	Male	Black	1.80
3	2	Female	Red	1.57
4	3	Male	Black	1.73
5	4	Male	Black	1.70
6	5	Male	Black	1.81
7	6	Female	Brown	1.63
8	7	Male	Red	1.70
9	8	Female	Brown	1.55
10	9	Female	Red	1.62
11	10	Female	Black	1.60
12	11	Male	Brown	1.71
13	12	Male	Brown	1.65
14	13	Female	Brown	1.56

Attribute: Gender
Values: {Male, Female}

NONIMAL DATA

Nominal (CN):

- **<u>NO</u>** unit of measurement (such as second, day, kg, calories, etc.)
- it doesn't make sense to order the values (Male vs. Female?)
  - Ordering is completely arbitrary
- Can simply think of it as a <u>labelling process</u>
  - Blood type (O, A, B, A/B)
  - Name (David, Karen, John, …)
  - Gender
- Special case: If there are only two categories (e.g., Male/Female, True/False), we call it *dichotomous* (or Boolean attribute)

Ordinal (Categorical Variable)

	A	D	F
1	Sex	Seatbelt	
2	Male	2_Rarely	
3	Male	2_Rarely	
4	Female	3_Sometimes	
5	Female	5_Always	
6	Male	3_Sometimes	
7	Female	3_Sometimes	
8	Female	5_Always	
9	Female	5_Always	
10	Female	3_Sometimes	
11	Male	5_Always	
12	Male	2_Rarely	
13	Male	2_Rarely	
14	Male	2_Rarely	
15	Male	4_Mosttimes	

Attribute: Seatbelt
Values: {Always, Mosttimes, Sometimes, Rarely, Never}

ORDINAL DATA

Example 2 (revisited)

- **Ordinal (CO):**
  - NO unit of measurement (such as second, day, kg, calories, etc.)
  - But you can put it in order – **ordering matters**!
    - However, the difference between successive scales is not known.
    - Hence, they can't be placed on the number line.
    - E.g., feeling (Happy, Neutral, Sad) – you can't really known how different from Happy to Neutral, or from Neutral to Sad.
  - Basic arithmetic is not appropriate, eg: you shouldn't subtract, add, multiply, divide ordinal variable.

Discrete (Metric variable)

	A	B	C	D	E
1	ParticipantID	Gender	Age	Hair Color	Height (m)
2	1	Male	18	Black	1.8
3	2	Female	14	Red	1.57
4	3	Male	17	Black	1.73
5	4	Male	22	Black	1.7
6	5	Male	23	Black	1.81
7	6	Female	21	Brown	1.63
8	7	Male	23	Red	1.7
9	8	Female	21	Brown	1.55
10	9	Female	19	Red	1.62
11	10	Female	20	Black	1.6
12	11	Male	18	Brown	1.71
13	12	Male	25	Brown	1.65
14	13	Female	13	Brown	1.56

Attribute: **Age**
Values: 1,2,3,4,5,6,7,…..,125

DISCRETE DATA

Example 3 (revisited)

Continuous (Metric Variable)

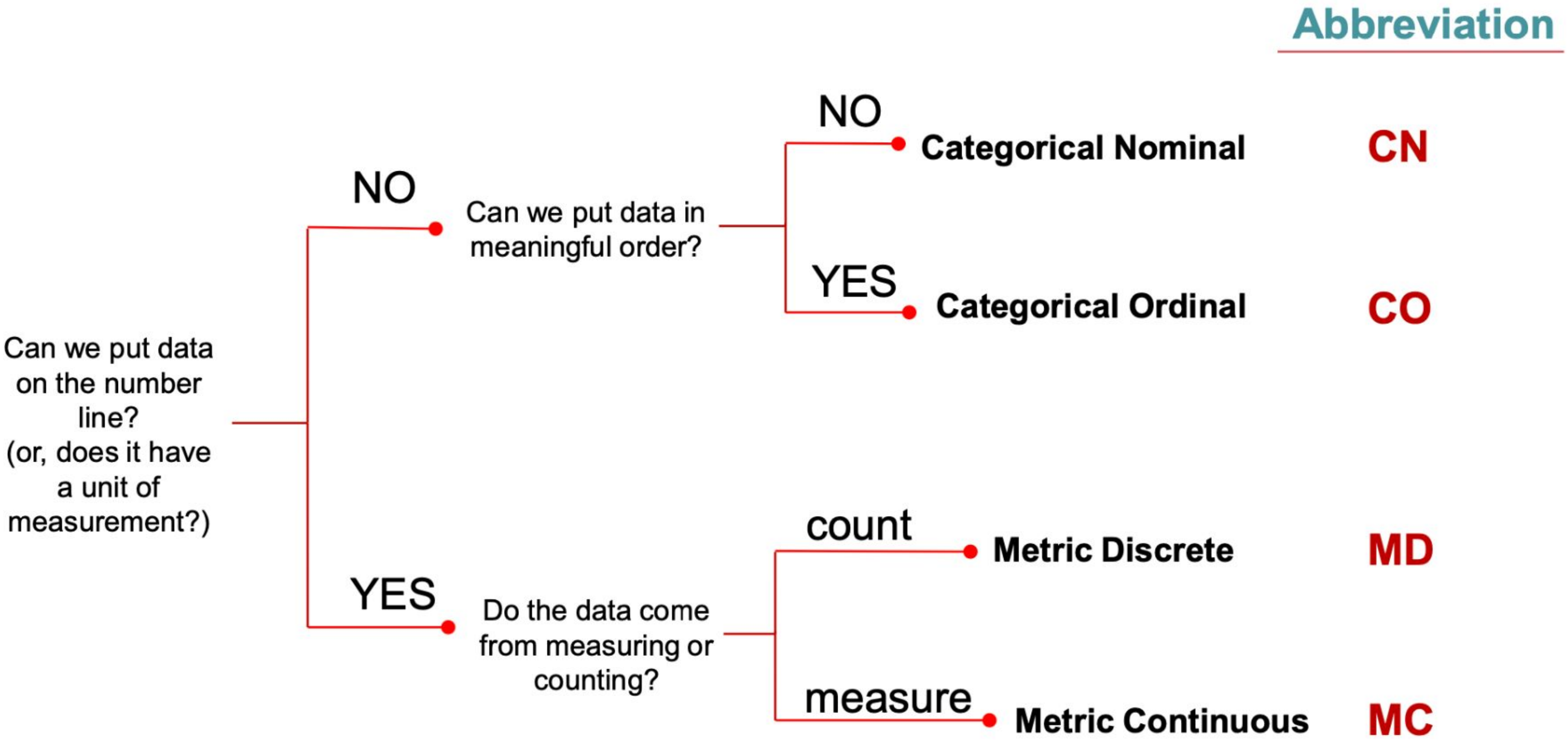
Example 4 (revisited)

	A	B	C	D
1	Participan	Gender	Hair Color	Height (m)
2	1	Male	Black	1.80
3	2	Female	Red	1.57
4	3	Male	Black	1.73
5	4	Male	Black	1.70
6	5	Male	Black	1.81
7	6	Female	Brown	1.63
8	7	Male	Red	1.70
9	8	Female	Brown	1.55
10	9	Female	Red	1.62
11	10	Female	Black	1.60
12	11	Male	Brown	1.71
13	12	Male	Brown	1.65
14	13	Female	Brown	1.56

Attribute: Height
Values: 1.8, 1.57, …

CONTINUOUS DATA

Classification Tree for Attribute Type

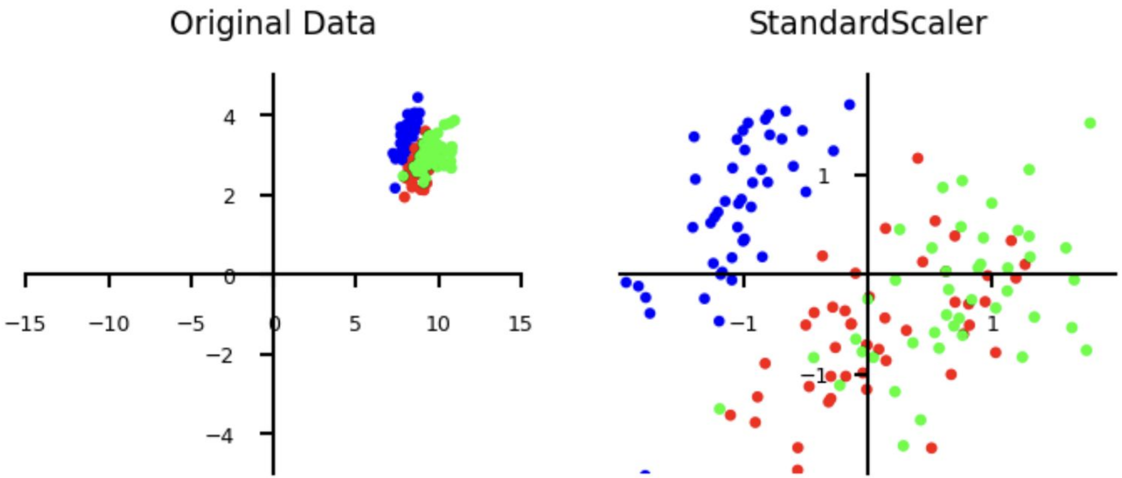


Data transformations

- Machine learning models make a lot of assumptions about the data
- In reality, these assumptions are often violated
- We build *pipelines* that *transform* the data before feeding it to the learners
    - Scaling (or other numeric transformations)
    - Encoding (convert categorical features into numerical ones)
    - Automatic feature selection
    - Feature engineering (e.g. binning, polynomial features,…)
    - Handling missing data
    - Handling imbalanced data
    - Dimensionality reduction (e.g. PCA)
    - Learned embeddings (e.g. for text)
- Seek the best combinations of transformations and learning methods
    - Often done empirically, using cross-validation
    - Make sure that there is no data leakage during this process!

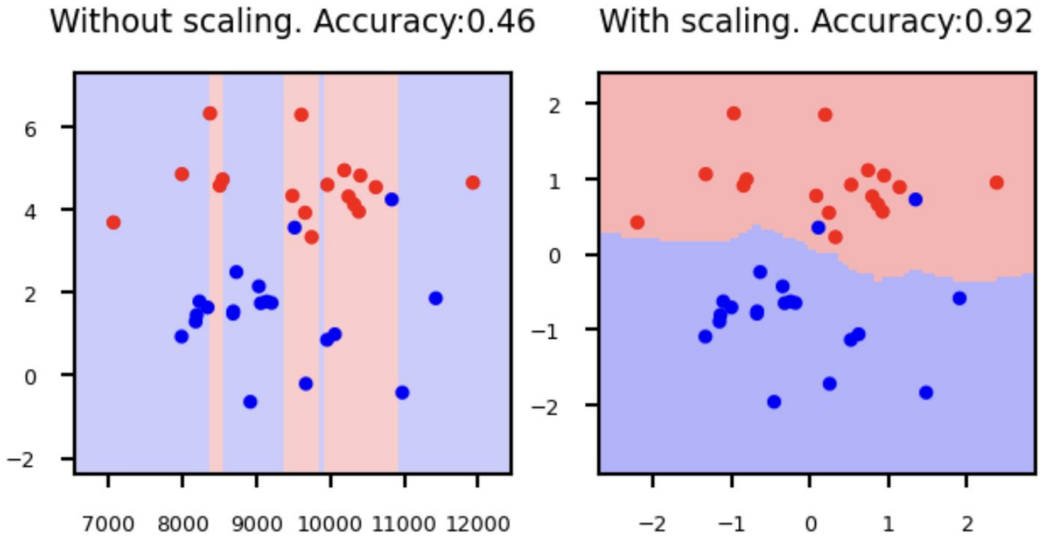
Scaling

- Use when different numeric features have different scales (different range of values)
    - Features with much higher values may overpower the others
- Goal: bring them all within the same range
- Different methods exist



Why do we need scaling?

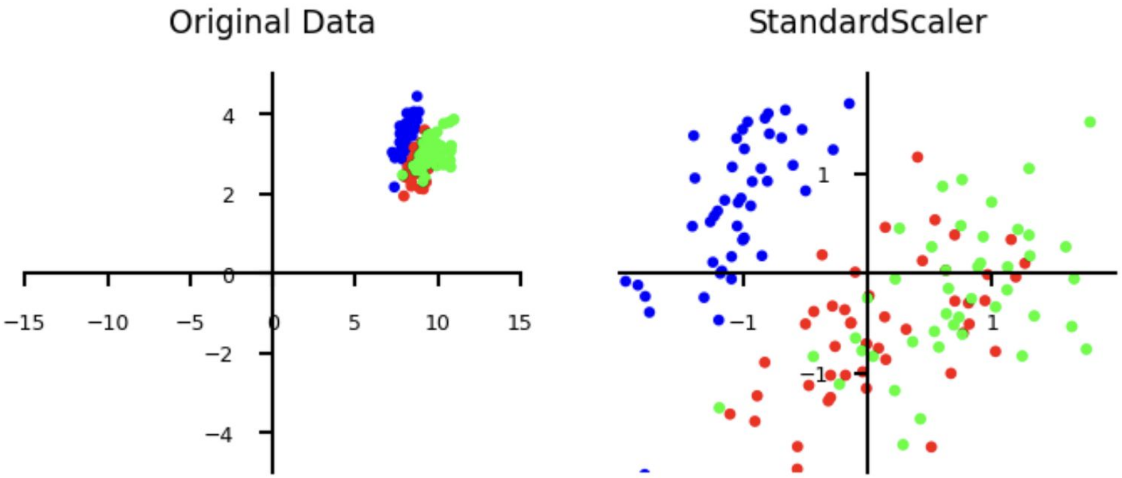
- KNN: Distances depend mainly on feature with larger values
- SVMs: (kernelized) dot products are also based on distances
- Linear model: Feature scale affects regularization
    - Weights have similar scales, more interpretable



Standard scaling (standardization)

- Generally most useful, assumes data is more or less normally distributed
- Per feature, subtract the mean value μ, scale by standard deviation σ
- New feature has $\mu = 0$ and $\sigma = 1$, values can still be arbitrarily large

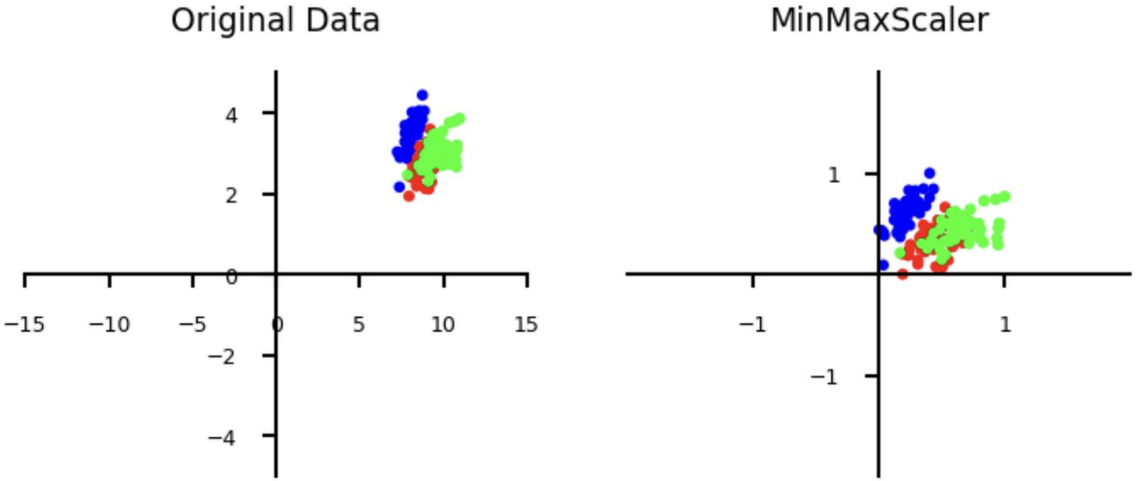
$$\mathbf{x}_{new} = \frac{\mathbf{x} - \mu}{\sigma}$$



Min-max scaling

- Scales all features between a given min and max value (e.g. 0 and 1)
- Makes sense if min/max values have meaning in your data
- Sensitive to outliers

$$\mathbf{x}_{new} = \frac{\mathbf{x} - x_{min}}{x_{max} - x_{min}} \cdot (max - min) + min$$



Categorical feature encoding

- Many algorithms can only handle numeric features, so we need to encode the categorical ones

	boro	salary	vegan
0	Manhattan	103	0
1	Queens	89	0
2	Manhattan	142	0
3	Brooklyn	54	1
4	Brooklyn	63	1
5	Bronx	219	0

Ordinal encoding

- Simply assigns an integer value to each category in the order they are encountered
- Only really useful if there exist a natural order in categories
    - Model will consider one category to be 'higher' or 'closer' to another

	boro	boro_ordinal	salary
0	Manhattan	2	103
1	Queens	3	89
2	Manhattan	2	142
3	Brooklyn	1	54
4	Brooklyn	1	63
5	Bronx	0	219

One-hot encoding (dummy encoding)

- Simply adds a new 0/1 feature for every category, having 1 (hot) if the sample has that category
- Can explode if a feature has lots of values, causing issues with high dimensionality
- What if test set contains a new category not seen in training data?
    - Either ignore it (just use all 0's in row), or handle manually (e.g. resample)

	boro	boro_Bronx	boro_Brooklyn	boro_Manhattan	boro_Queens	salary
0	Manhattan	0	0	1	0	103
1	Queens	0	0	0	1	89
2	Manhattan	0	0	1	0	142
3	Brooklyn	0	1	0	0	54
4	Brooklyn	0	1	0	0	63
5	Bronx	1	0	0	0	219

Goals of Feature Engineering

- Convert ‘context’ -> input to learning algorithm.
- Expose the structure of the concept to the learning algorithm.
- Work well with the structure of the model the algorithm will create.
- Balance number of features, complexity of concept, complexity of model, amount of data.

Sample from SMS Spam

- SMS Message (arbitrary text) -> 5 dimensional array of binary features

- 1 if message is longer than 40 chars, 0 otherwise
- 1 if message contains a digit, 0 otherwise
- 1 if message contains word ‘call’, 0 otherwise
- 1 if message contains word ‘to’, 0 otherwise
- 1 if message contains word ‘your’, 0 otherwise

“SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info”

Long?	HasDigit?	ContainsWord(Call)	ContainsWord(to)	ContainsWord(your)

Basic Feature Types

Binary Features

- ContainsWord(call)?
- IsLongSMSMessage?
- Contains(*#)?
- ContainsPunctuation?

Categorical Features

- FirstWordPOS ->
  { Verb, Noun, Other }
- MessageLength ->
  { Short, Medium, Long, VeryLong }
- TokenType ->
  { Number, URL, Word, Phone#, Unknown }
- GrammarAnalysis ->
  - { Fragment, SimpleSentence, ComplexSentence }

Numeric Features

- CountOfWord(call)
- MessageLength
- FirstNumberInMessage
- WritingGradeLevel

Feature Engineering for Text

- Tokenizing
- Bag of Words
- N-grams
- TF-IDF
- Embeddings

Tokenizing

- Breaking text into words

  “Nah, I don't think he goes to usf” ->

  [‘Nah,’ ‘I’, ‘don't’, ‘think’, ‘he’, ‘goes’, ‘to’, ‘usf’]

- Dealing with punctuation

  “Nah,” ->

  [‘Nah,’] or [‘Nah’, ‘,’] or [‘Nah’]

  “don't” ->

  [‘don't’] or [‘don’, ‘'’, ‘t’] or [‘don’, ‘t’] or [‘do’, ‘n't’]

- Normalizing

  “Nah,” ->

  [‘Nah,’] or [‘nah,’]

  “1452” ->

  [‘1452’] or [<number>]

Some tips for deciding

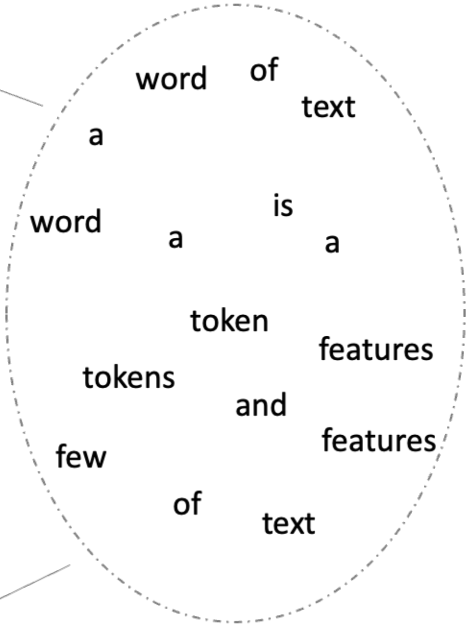
- If you have lots of data / optimization…
  - Keep as much information as possible
  - Let the learning algorithm figure out what is important and what isn’t
- If you don’t have much data / optimization...
  - Reduce the number of features you maintain
  - Normalize away irrelevant things
- Focus on things relevant to the concept…
  - Explore data / use your intuition
  - Overfitting / underfitting ← much more later

Bag of Words

m1: A word of text.
m2: A word is a token.
m3: Tokens and features.
m4: Few features of text.

Training data

Bag of words



Tokens

One feature per unique token

x_1	a
x_2	word
x_3	of
x_4	text
x_5	is
x_6	token
x_7	tokens
x_8	and
x_9	features
x_{10}	few

Features

Bag of Words: Example

test1: Some features for a text example.

Out of vocabulary

m1: A word of text.
m2: A word is a token.
m3: Tokens and features.
m4: Few features of text.

x_1	a
x_2	word
x_3	of
x_4	text
x_5	is
x_6	token
x_7	tokens
x_8	and
x_9	features
x_{10}	few

Selected Features

	m1	m2	m3	m4
x_1	1	1	0	0
x_2	1	1	0	0
x_3	1	0	0	1
x_4	1	0	0	1
x_5	0	1	0	0
x_6	0	1	0	0
x_7	0	0	1	0
x_8	0	0	1	0
x_9	0	0	1	1
x_{10}	0	0	0	1

Training X

	test1
x_1	1
x_2	0
x_3	0
x_4	1
x_5	0
x_6	0
x_7	0
x_8	0
x_9	1
x_{10}	0

Test X

Use bag of words when you have a lot of data, can use many features

N-Grams: Tokens

- Instead of using single tokens as features, use series of N tokens
- “down the bank” vs “from the bank”

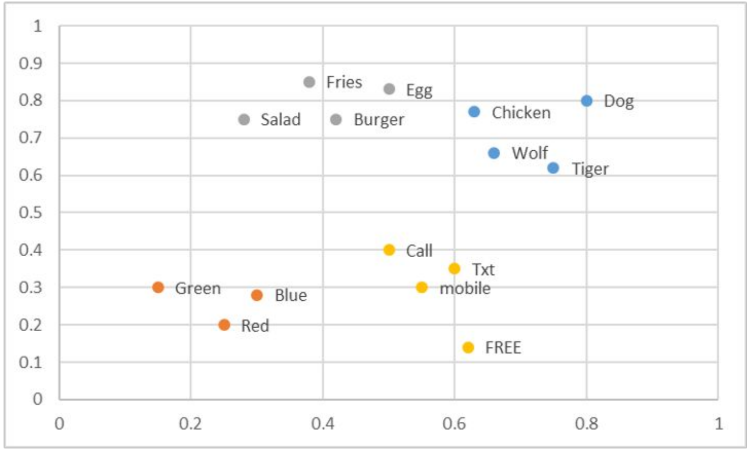
Message 1: “Nah I don't think he goes to usf”
Message 2: “Text FA to 87121 to receive entry”

Message 2:

Nah I	I don’t	don’t think	think he	he goes	goes to	to usf	…	Text FA	FA to	87121 to	To receive	receive entry
0	0	0	0	0	0	0	…	1	1	1	1	1

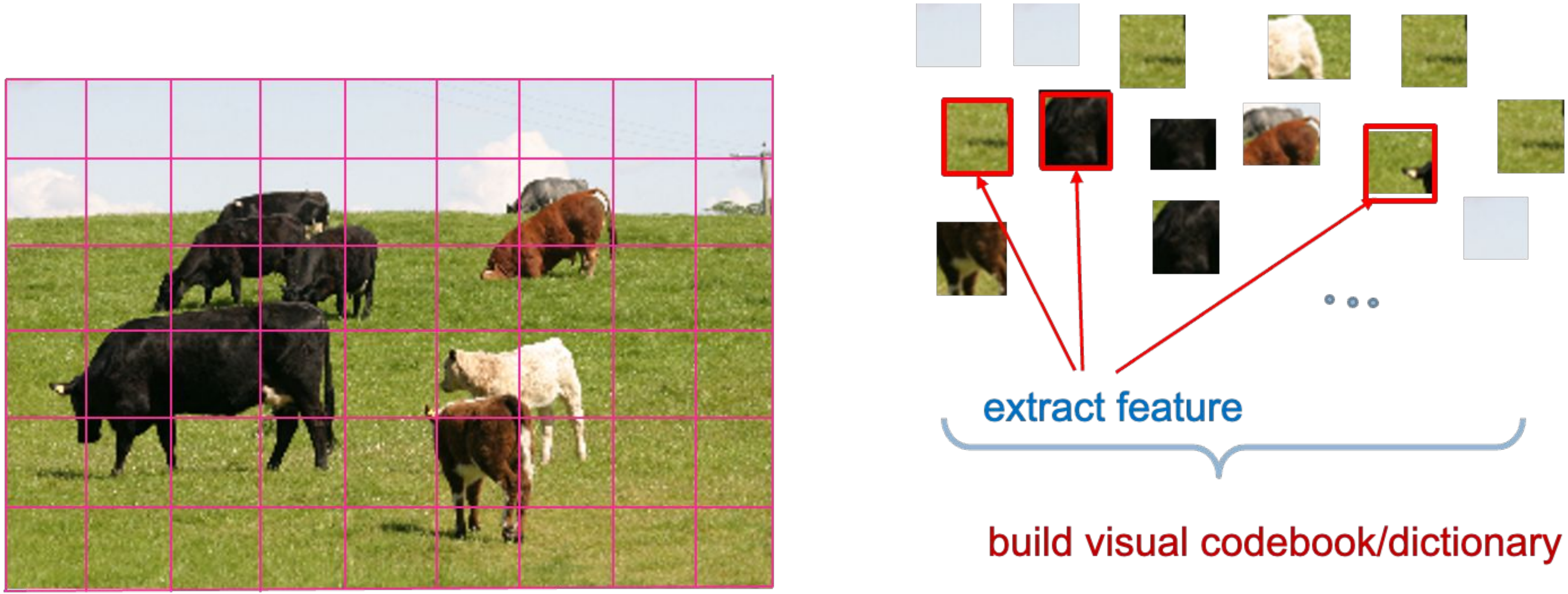
Use when you have a LOT of data, can use MANY features

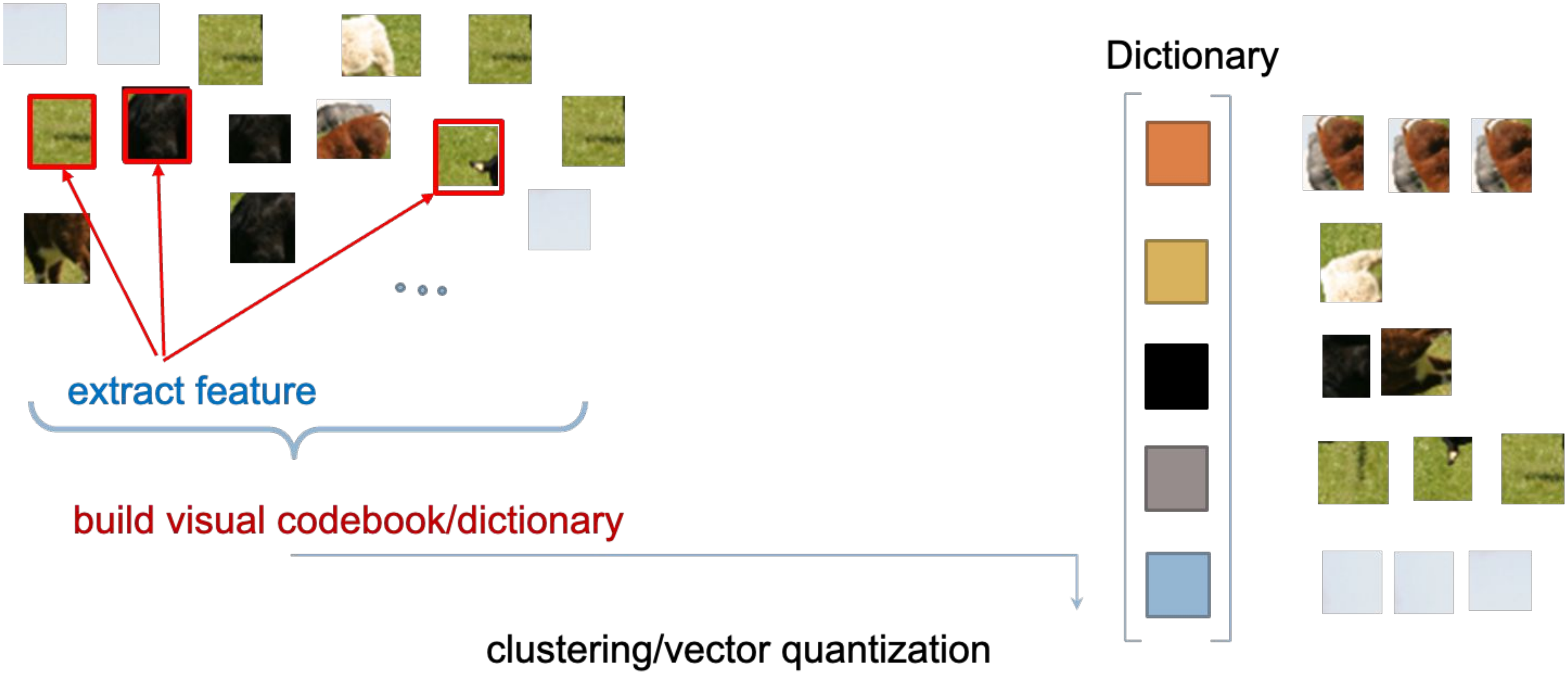
Embeddings -- Word2Vec and FastText

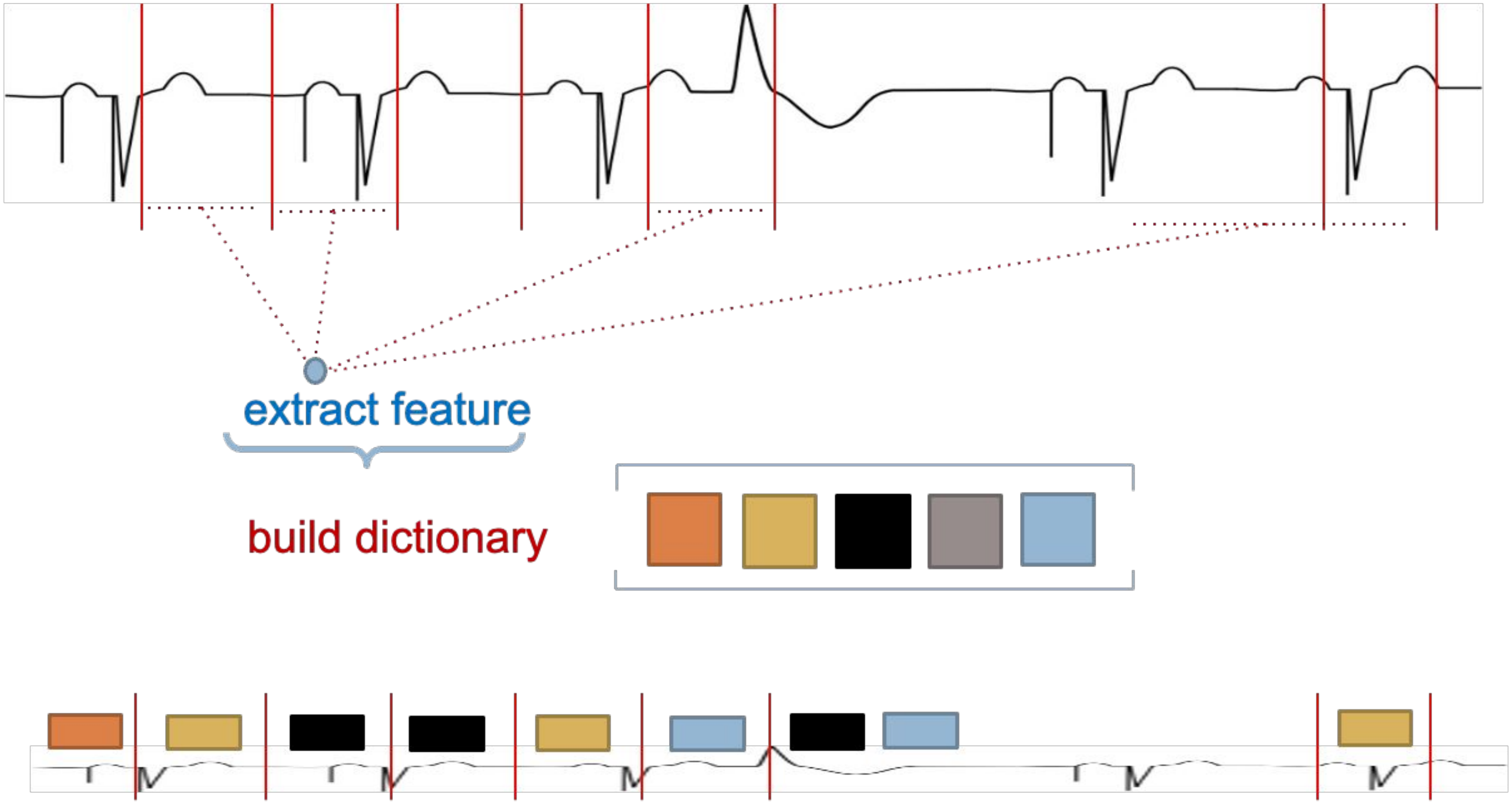


- Word -> Coordinate in N dimension
- Regions of space contain similar concepts
- Creating Features Options:
    - Average vector across words
    - Count in specific regions
- Commonly used with neural networks

Replaces words with their ‘meanings’ – sparse -> dense representation


extract feature
build visual codebook/dictionary


extract feature
build visual codebook/dictionary
clustering/vector quantization
Dictionary



$$x = (1, 2, 3, 3, 2, 4, 3, 4, 1, 2, \dots)$$