UC San Diego

DSC 102
Systems for Scalable Analytics

Rod Albuyeh

Topic 5: Model Building Systems

Chapter 8.1 and 8.3 of MLSys Book

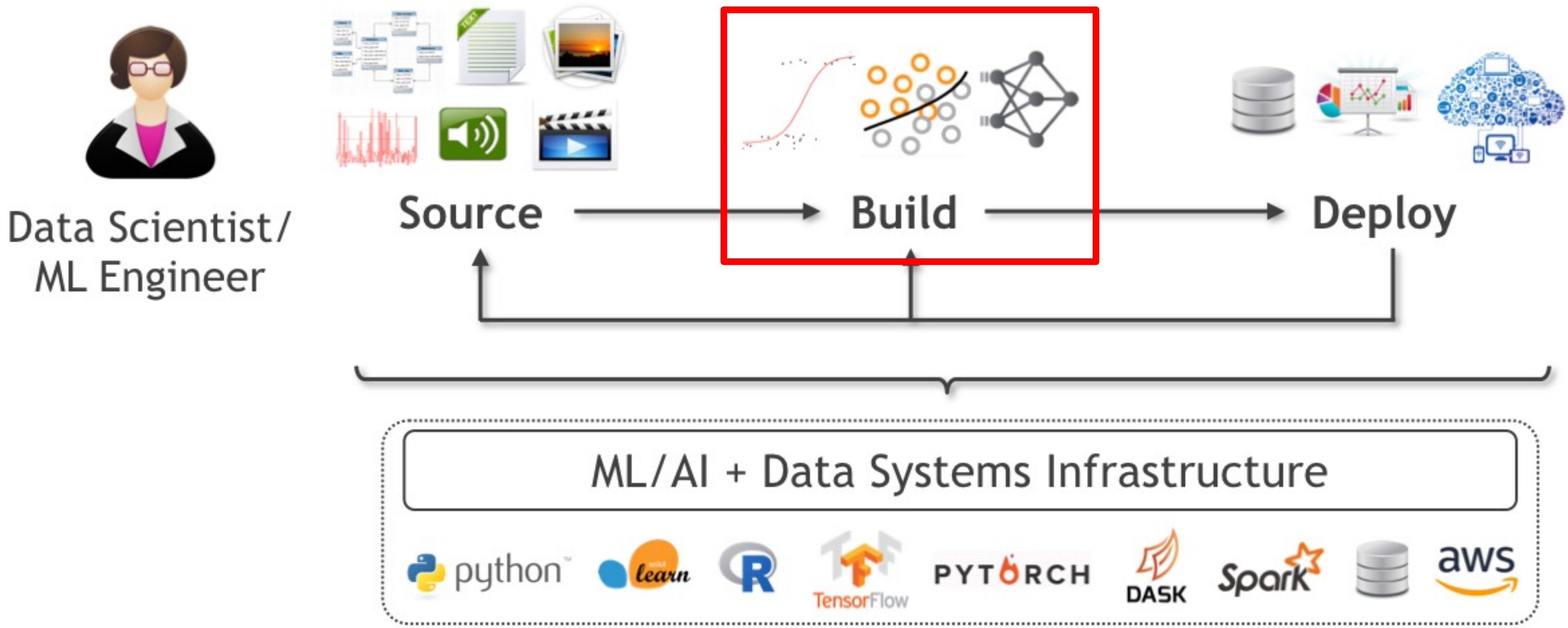
Admin

Reminder: 90%+ CAPE response rate for class yields 0.5% collective boost to final score.

Current response rate as of June 5th 7pm is 36.61%.

Help both your instructor and DSC 102 improve!

The Lifecycle of ML-based Analytics



Data acquisition
Data preparation

Feature Engineering
Training & Inference
Model Selection

Serving
Monitoring

Building Stage of ML Lifecycle

- Perform **model selection**, i.e., convert prepared ML-ready data to **prediction function(s)** and/or other analytics outputs
- What makes model building challenging/time-consuming?
  - **Heterogeneity** of data sources/formats/types
  - **Configuration complexity** of ML models
  - Large **scale** of data
  - **Long training runtimes** of some models
  - **Pareto optimization on criteria** for application
  - **Evolution** of data-generating process/application

Building Stage of ML Lifecycle

- Perform **model selection**, i.e., convert prepared ML-ready data to **prediction function(s)** and/or other analytics outputs
- Data scientist / ML engineer must steer 3 key activities that invoke ML **training** and **inference** as sub-routines:

1. **Feature Engineering (FE):**

   Appropriate signals representation for domain of prediction function.

2. **Algorithm/Architecture Selection (AS):**

   Choice of prediction functions class (incl. artificial neural networks (ANN) architecture) to use.

3. **Hyper-parameter Tuning (HT):**

   Model improvement (accuracy, etc.) by configuring ML "knobs".

https://dl.acm.org/doi/abs/10.1145/3399579.3399870

High Level Overview

- Perform **model selection**, i.e., convert prepared ML-ready data to **prediction function(s)** and/or other analytics outputs
- Data scientist / ML engineer must steer 3 key activities that invoke ML **training** and **inference** as sub-routines:

1. **Feature Engineering (FE):**

   Appropriate signals representation for domain of prediction function.

2. **Algorithm/Architecture Selection (AS):**

   Choice of prediction functions class (incl. artificial neural networks (ANN) architecture) to use.

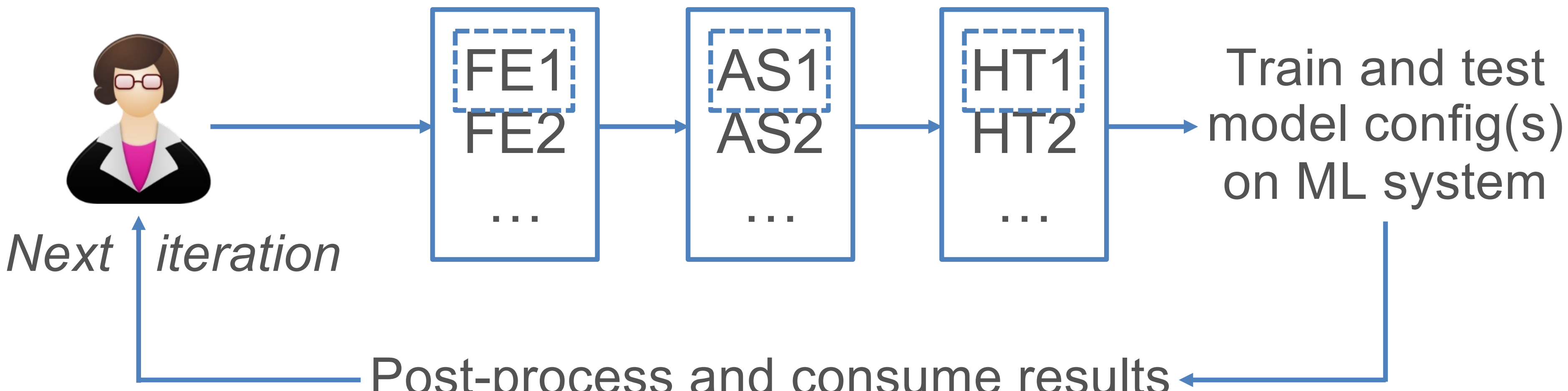
3. **Hyper-parameter Tuning (HT):**

   Model improvement (accuracy, etc.) by configuring ML “knobs”.

https://dl.acm.org/doi/abs/10.1145/3399579.3399870

Model Selection Process

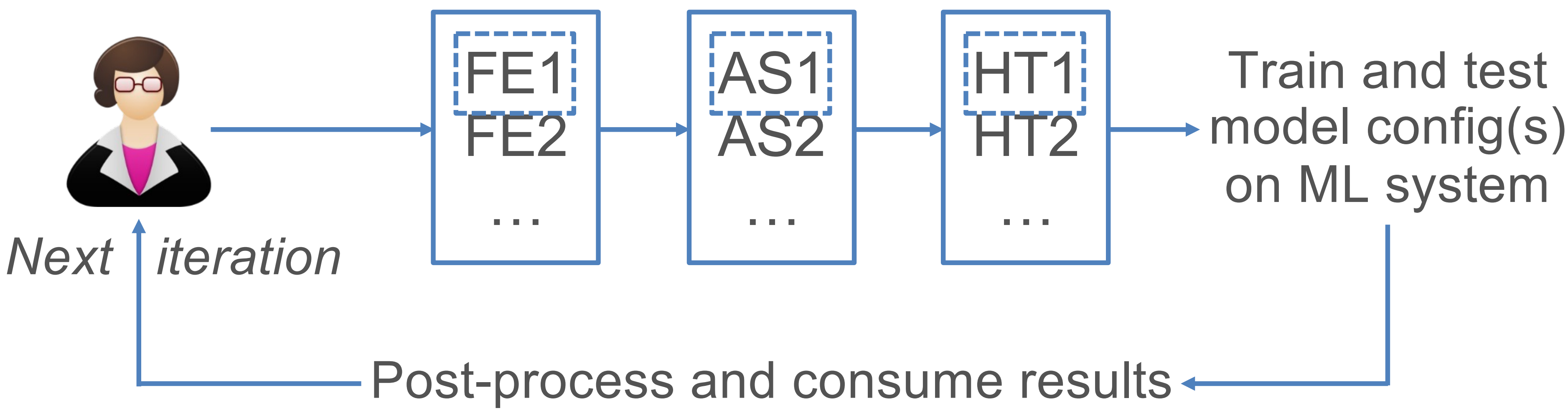
- Model selection is usually an *iterative exploratory* process with human making decisions on FE, AS, and/or HT
- Increasingly, automation of some or all parts possible: **AutoML**



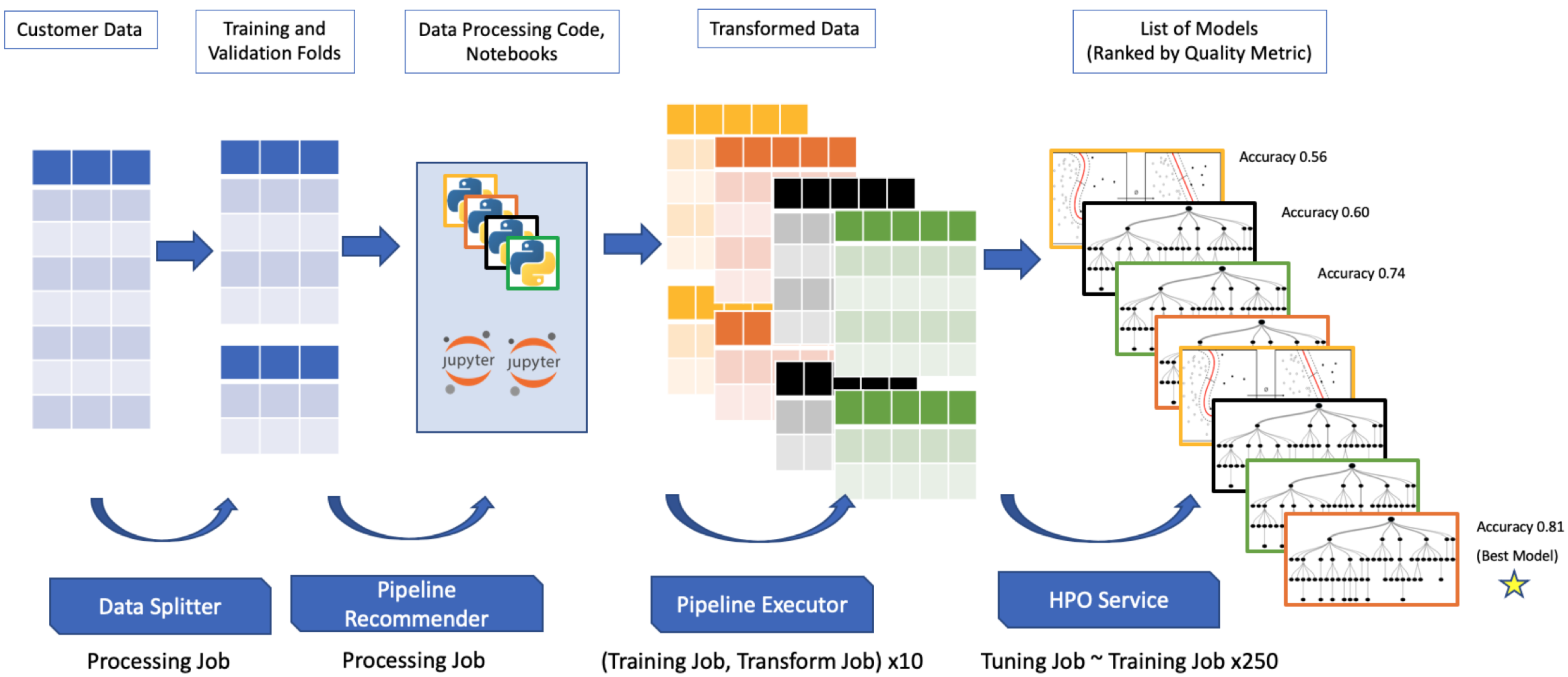
https://dl.acm.org/doi/abs/10.1145/3399579.3399870

Model Selection Process

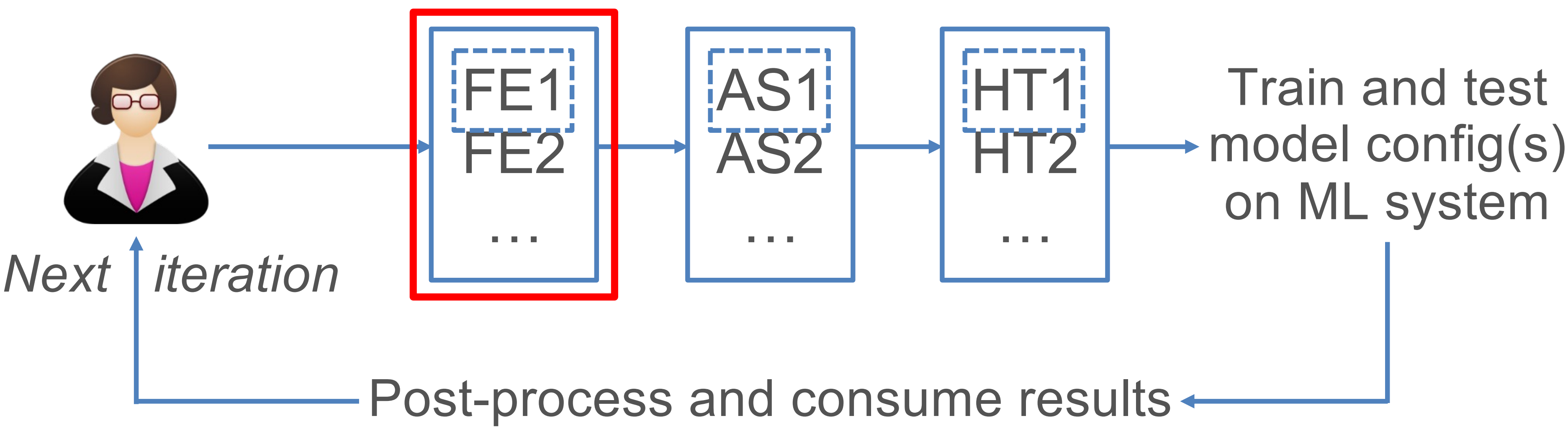
- Decisions on FE, AS, HT guided by many constraints/metrics: prediction accuracy, data/feature types, interpretability, tool availability, scalability, runtimes, fairness, legal issues, etc.
- Decisions are typically application-specific and dataset-specific; recall Pareto surfaces and tradeoffs



AutoML-based Selection



Feature Engineering



Feature Engineering

- Converting prepared data into a *feature vector representation* for ML training and inference
  - Aka feature extraction, representation extraction, etc.
- Umbrella term for many tasks dep. on type of ML model trained:
  1. Recoding and value conversions
  2. Joins and/or aggregates
  3. Feature interactions
  4. Feature selection
  5. Dimensionality reduction
  6. Temporal feature extraction
  7. Textual feature extraction and embeddings
  8. Learned feature extraction in deep learning

1. Recoding and value conversions

- Common on relational/tabular data
- Typically needs some *global column stats* + code to *reconvert* each tuple (example's feature values)

UserID	State	Date	Upvotes	Comment	Label
143	CA	4/3/19	1539	"This restaurant is overrated"	-
337	NY	11/7/19	5020	"Not too bad!"	+
98	WI	2/8/20	402	"Pretty rad"	+
…	…	…	…	…	…

Example:

Decision trees can use categorical features directly but GLMs support only numeric features; need numerical vector such as **one-hot encoded, weight of evidence / target encoding, integer encoding, embedding (via additional DL model), etc.** How to implement and scale these?

1. Recoding and value conversions

- Common on relational/tabular data
- Typically needs some *global column stats* + code to *reconvert* each tuple (example's feature values)

UserID	State	Date	Upvotes	Comment	Label
143	CA	4/3/19	1539	"This restaurant is overrated"	-
337	NY	11/7/19	5020	"Not too bad!"	+
98	WI	2/8/20	402	"Pretty rad"	+
…	…	…	…	…	…

Example:

GLMs and ANNs need **standardization** (either mean/stdev or min/max based) and **decorrelation**

Scaling global stats: How to scale mean/stdev/max/min?

Reconversion: Tuple-level function to modify number using stats. How to scale?

1. Recoding and value conversions

- Common on relational/tabular data
- Typically needs some *global column stats* + code to *reconvert* each tuple (example's feature values)

UserID	State	Date	Upvotes	Comment	Label
143	CA	4/3/19	1539	"This restaurant is overrated"	-
337	NY	11/7/19	5020	"Not too bad!"	+
98	WI	2/8/20	402	"Pretty rad"	+
…	…	…	…	…	…

Example:

Some models like Bayesian Networks or Markov Logic Networks benefit from (or even need) **binning/discretization** of numerics

Scaling global stats: How to scale histogram computations?

Reconversion: Tuple-level function to convert number to bin ID

2. Joins and Aggregates

- Common on relational/tabular data
- Most real-world relational datasets are multi-table; require key-foreign key joins, aggregation-and-key-key-joins, etc.

UserID	Age	Name
304	40	…
23	25	…
143	33	…
…	…	…

UserID	State	Date	Upvotes	Comment	Label
143	CA	…	…	…	-
337	NY	…	…	…	+
143	CA	…	…	…	+
…	…	…	…	…	…

Example:

Join tables on UserID; concatenate user's info. as extra features!

What kind of join is this? How to scale this computation?

2. Joins and Aggregates

- Common on relational/tabular data
- Most real-world relational datasets are multi-table; require key-foreign key joins, aggregation-and-key-key-joins, etc.

UserID	State	Date	Upvotes	Comment	Label
143	CA	…	…	…	-
337	NY	…	…	…	+
143	CA	…	…	…	+
…	…	…	…	…	…

Example:

Join table with itself on UserID to count #reviews and avg #upvotes for each user in a new temp. table and join that to get more features!

What kind of computation is this? How to scale it?

3. Polynomials and Feature Interactions

- Sometimes used on relational/tabular data, especially for high-bias models like GLMs
- Pairwise is common; ternary is not unheard of

F1	F2	F3	Label
3	2	…	-
4	20	…	+
5	10	…	+
…	…	…	…

F1	F2	F3	F11	F12	F13	F22	F23	F33	Label
3	2	…	9	6	…	4	…	…	-
4	20	…	16	80	…	400	…	…	+
5	10	…	25	50	…	100	…	…	+
…	…	…	…	…	…	…	…	…	…

- No global stats, just a tuple-level function
- Popularity of this has reduced due to GBMs popularity for tabular data, which encode nonlinearities and interactions as part of the learning process.

4. Feature Selection

- Often used on high dimensional relational/tabular data
- **Basic Idea:** Instead of using whole feature set, use a subset

UserID	State	Date	Upvotes	Comment	Label
…	…	…	…	…	…

State	Upvotes	Comment	Label
…	…	…	…

Upvotes	Comment	Label
…	…	…

…

- Formulated as a *discrete optimization* problem
  - NP-Hard in #features in general
  - Many heuristics exist in ML/data mining; typically rely on some *information theoretic criteria*
  - Typically scaled as "outer loops" over training/inference
- Some ML users also prefer human-in-the-loop approach

5. Dimensionality Reduction

- Often used on relational/structured/tabular data
- **Basic Idea:** Transforms features to a different latent space
- **Examples:** Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Linear Discriminant Analysis (LDA), Matrix factorization

UserID	State	Date	Upvotes	Comment	Label
…	…	…	…	…	…

F1	F2	F3	Label
0.3	4.2	-29.2	…

Q: How is this different from "feature selection"?

- Feat. sel. *preserves* semantics of each feature but dim. red. typically does not—combines features in "nonsensical" ways
- Scaling this is non-trivial! Similar to scaling individual ML training algorithms (later)

6. Temporal Feature Extraction

- Many relational/tabular data have time/date
- Per-example reconversion to extract numerics/categoricals
- Sometimes global stats needed to calibrate time
- Complex temporal features studied in *time series mining*

UserID	State	Date	Upvotes	Comment	Label
143	CA	4/3/19	1539	"This restaurant is overrated"	-
337	NY	11/7/19	5020	"Not too bad!"	+
98	WI	2/8/20	402	"Pretty rad"	+
…	…	…	…	…	…

Example:

Most classifiers cannot use Date directly; extract month (categorical), year (categorical?), day? (categorical), etc.

Reconversion: Tuple-level function (many-to-one) to extract numbers/categories

7. Textual Feature Extraction

- Many relational/tabular data have text columns; in NLP, whole example is often just text
- Most classifiers cannot process text/strings directly
- Extracting numerics from text studied in *text mining*

…	Comment	Label
…	"This restaurant is sucks"	-
…	"Good good!"	+
…	"Pretty rad"	+
…	…	…

…	sucks	good	…	Label
…	1	0	…	-
…	0	2	…	+
…	0	0	…	+
…	…	…	…	…

Example:

Bag-of-words features: count number of times each word in a given *vocabulary* arises; need to know vocabulary first

Scaling global stats: How to get vocabulary?

Reconversion: Tuple-level function to count words; look up index

7. Textual Feature Extraction

- **Knowledge Base-based:** Domain-specific knowledge bases like entity dictionaries (e.g., celebrity or chemical names) help extract domain-specific features
- **Embedding-based:**
  - Numeric vector for a text token; popular in NLP
  - Offline training of function from string to numeric vector in self-supervised way on large text corpus (e.g., Wikipedia); embedding dimensionality is a hyper-parameter
  - *Pre-trained* word embeddings (Word2Vec and GloVe) and sentence embeddings (Doc2Vec) available off-the-shelf; to scale, just use a tuple-level conversion function

Word2Vec: https://en.wikipedia.org/wiki/Word2vec
GloVe: https://nlp.stanford.edu/projects/glove
Doc2Vec: https://arxiv.org/abs/1405.4053

8. Learned Feature Extraction in DL

- A big win of Deep Learning (DL) is no manual feature engineering on *unstructured* data
  - DL is *not* common on structured/tabular data, but growing in popularity. See: https://arxiv.org/pdf/2110.01889.pdf
- DL is very *versatile*: almost *any data type* as input and/or output:
  - Convolutional NNs (CNNs) over image tensors
  - Recurrent NNs (RNNs) and Transformers over text
  - Graph NNs (GNNs) over graph-structured data
- Neural architecture specifies how to extract and transform features internally with weights that are learned
- **Software 2.0:** Buzzword for such "learned feature extraction" programs vs old hand-crafted feature engineering

https://medium.com/@karpathy/software-2-0-a64152b37c35