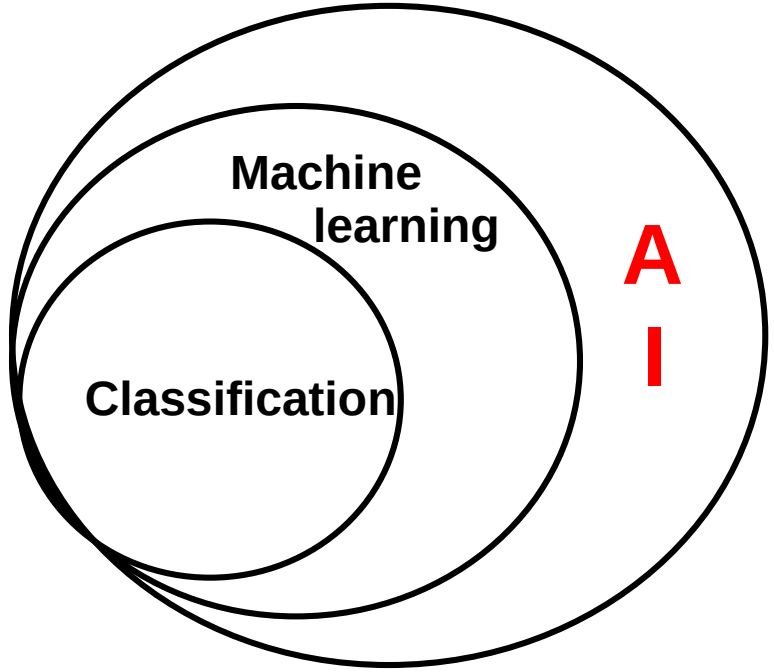
LAB for Classification

Lydia Y. Chen
IBM Research – Zurich Lab



How to Build Data Models

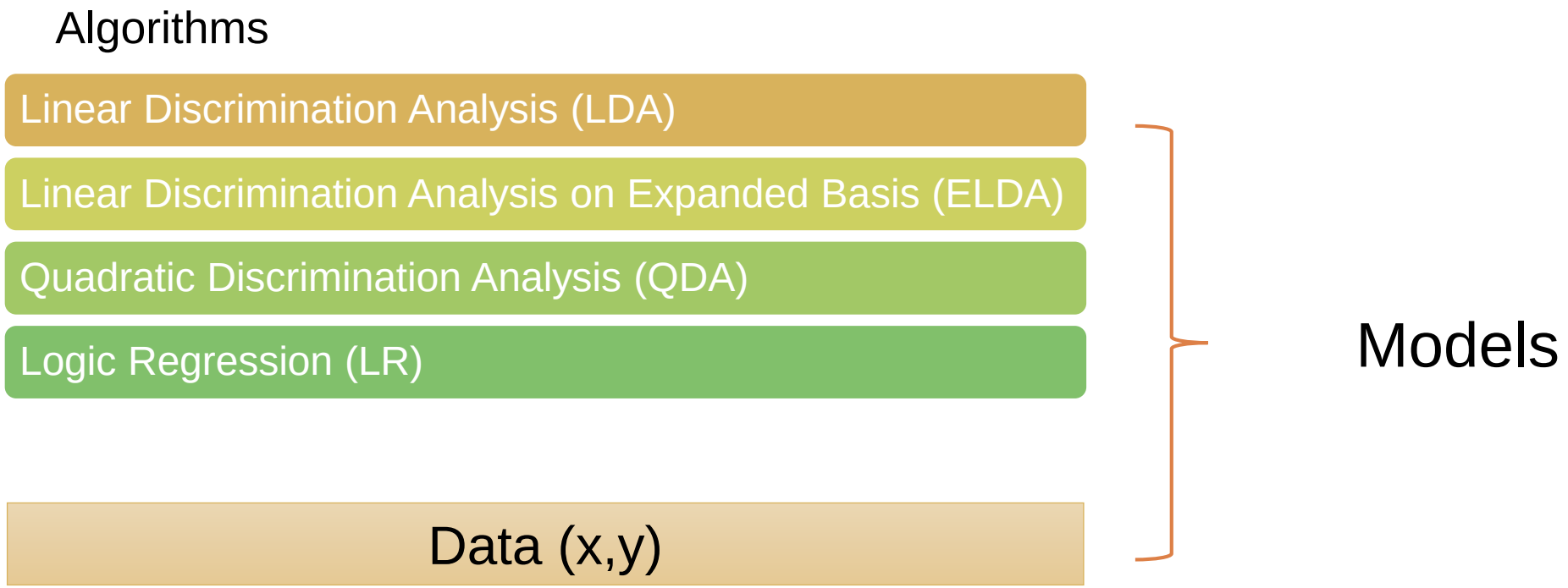
- Split data into: training and testing set
- Build the classifier from the training set
  - Clean the data: checking missing values
  - Be careful about the categorical data
  - Standardize the data range
- Inference on the testing data set
- Visualize the data and classification results

LAB 1: Diabetes

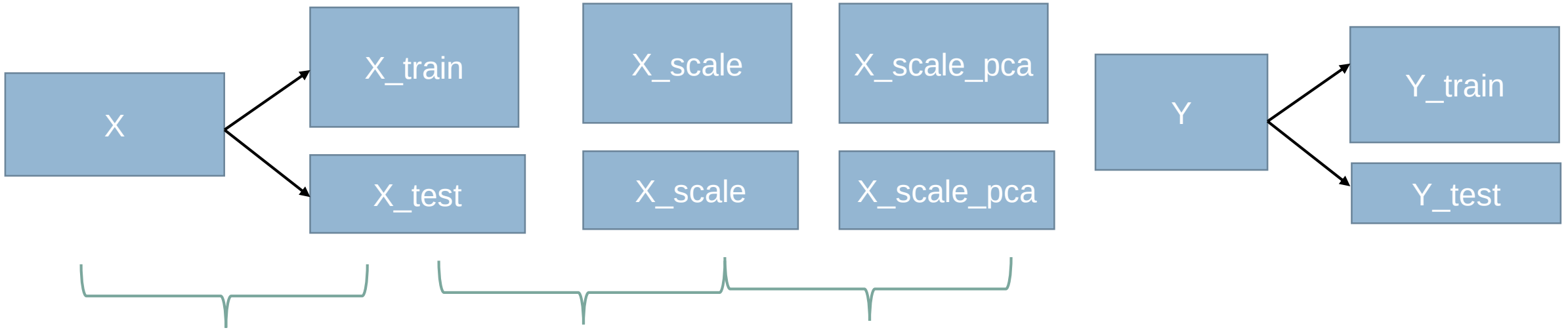
Data sets

- Data/diabetes.csv
- 768 instances
- 8 features(x) (no.1- no.8):
  1. Number of times pregnant
  2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
  3. Diastolic blood pressure (mm Hg)
  4. Triceps skin fold thickness (mm)
  5. 2-Hour serum insulin (mu U/ml) \
  6. Body mass index (weight in kg/(height in m)^2)
  7. Diabetes pedigree function
  8. Age (years)
- Response variable (y) (no.9): binary

Apply Classification Algorithms



1. Creating training and testing data

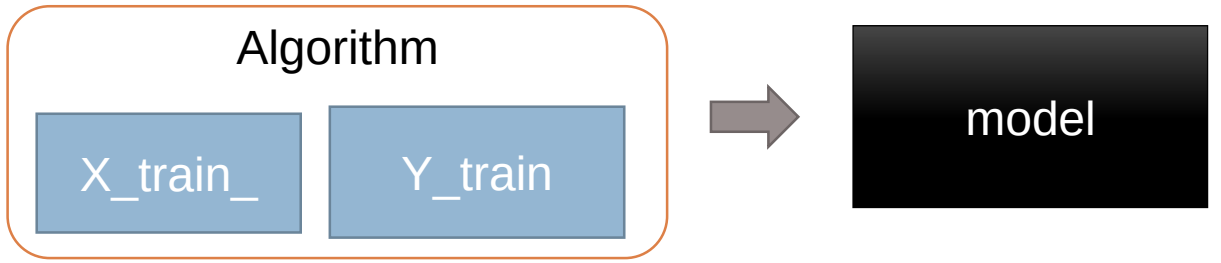


Splitting the data (e.g., 70-30)

Standardizing the data, keeping every future in the similar range

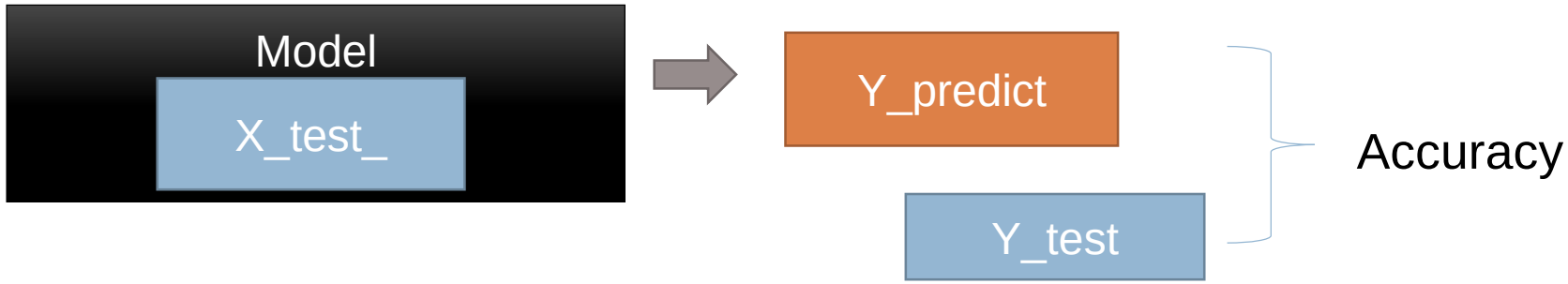
Get the principle components (reduce the dimension)

2. Build the model on training set using a specific algorithm and training data



Algorithm: lr, lda, qda, elda, svm, dt, rf
Input: x_train, x_train_scale, x_train_pca

3.Check the accuracy on the testing set



Input: x_test, x_test_scale, x_test_pca

Basic structure of functions

- model= modelfunctionname().fit(X_train, Y_train)

- Y_model=model.predict(X_test)
- metrics.accuracy_score(Y_test,Y_model)
- accuracy_score(Y_test,Y_model)*100))
- compute_and_plot_cm(Y_test, Y_model, data_class_labels, title="model")
- plot_2d(X_test, Y_model, data_class_labels, title="model")

LAB 2: Google Trace

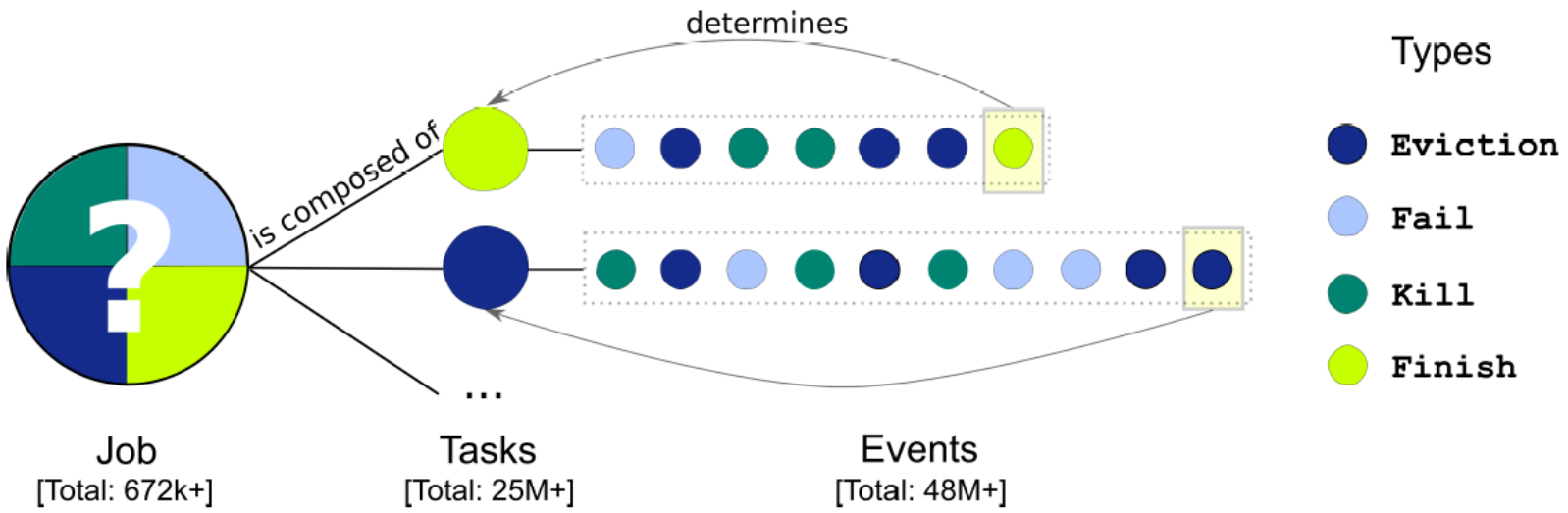
Big data from performance logs

Datacenters “***manually***” and
“***digitally***“ monitor performance

Look into performance big data

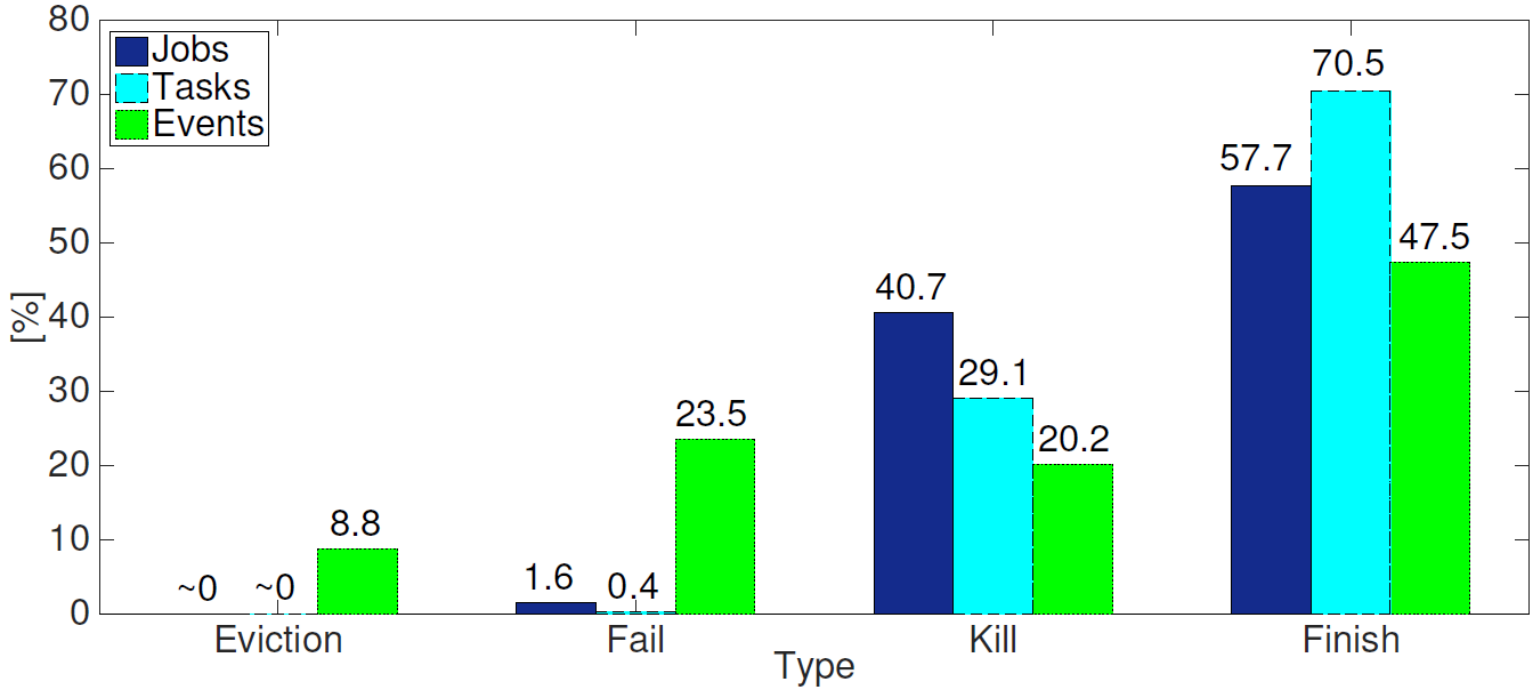
Can we derive actionable insights from performance big data?

A study on Google trace



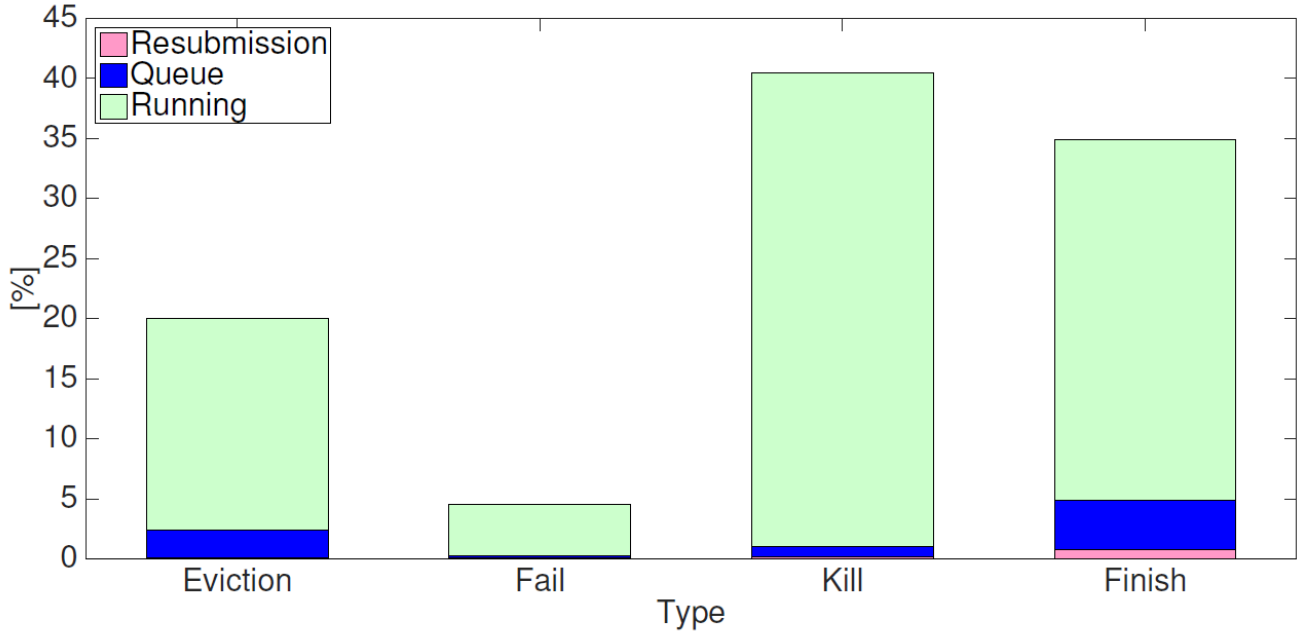
- Heterogeneous system and applications (multiple priorities)
- Multi-layer structure: job, task and event

Statistics on unsuccessful executions



- Many unsuccessful executions: job (42.3%), task (29.5%), and events (52.5%)

The impact of unsuccessful events: wasted time



Resubmission: termination to sub. Queue: sub. to scheduling. Running: scheduling to termination

- More than 65% of time is wasted
- Events fail after a *short* while but are evicted after a *long* while

Relevant Features

Type	group	Feature List	
Static	Event (5)	Task priority Task requested CPU/RAM/DISK	Task scheduling class
	Job (10)	Job size AVG/STD task priority	Job scheduling class AVG/STD task requested CPU/RAM/DISK
	System(36)	AVG/STD lower/same/higher priority arrivals AVG/STD lower/same/higher priority throughput AVG/STD lower/same/higher priority n. of tasks	AVG/STD lower/same/higher priority variation of arrivals AVG/STD lower/same/higher priority variation of throughput AVG/STD lower/same/higher priority variation of n. of tasks
Historical	Previous event type (4)	Number of previous finish events Number of previous fail events	Number of previous eviction events Number of previous kill events
	Previous event data (15)	Previous event priority Previous event requested CPU/RAM/DISK Previous event queue/running time Previous event machine concurrency	Previous event scheduling class Previous event type Previous event machine CPU/RAM capacity Previous event machine CPU/RAM reservation/utilization

Key features:

- Static feature (known upon arrival): system load, priority
- Historical feature: previous event

Putting it together: machine learning

- Linear Discrimination Analysis (LDA)
- Linear Discrimination Analysis on Expanded Basis (ELDA)
- Quadratic Discrimination Analysis (QDA)
- Logic Regression (LR)
- Decision Tree (DT)
- Support Vector Machine (SVM)
- Neural Networks

Questions:

- How to predict event (more difficult), task, and job type?
- How to minimize the waste of resources/time?

Data sets

- Data/job.csv
- 1541720 instances
- 28 features (x) (no.1- no.28)
- Response variable (y): 4 classes