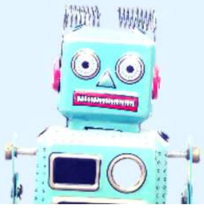
Dive Into

Machine and Deep Learning

For Tabular and Network Data

Gary Ang



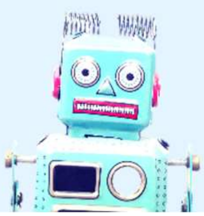
Overview

- Review of concepts
- Characteristics of tabular and network data
- From trees and forests to XGBoost
- From machine learning to deep learning
- From network analysis to Graph Neural Networks
- From supervised to unsupervised learning

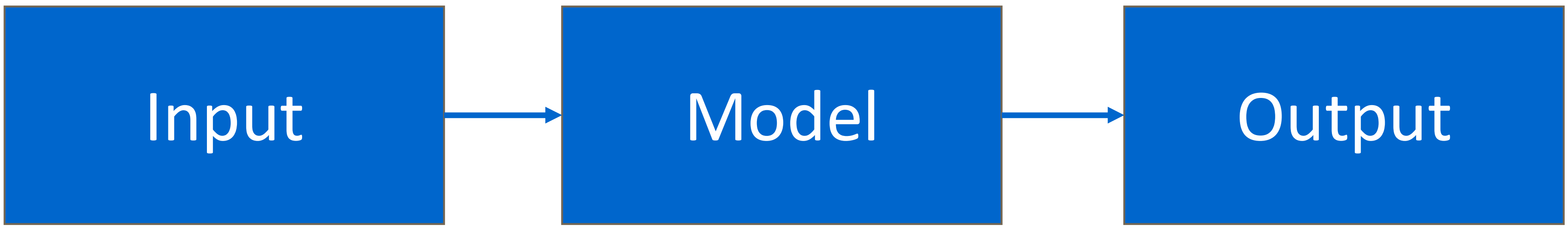
What we will focus on

- Intuition
- Mental models
- Patterns
- Concepts

Review of Concepts



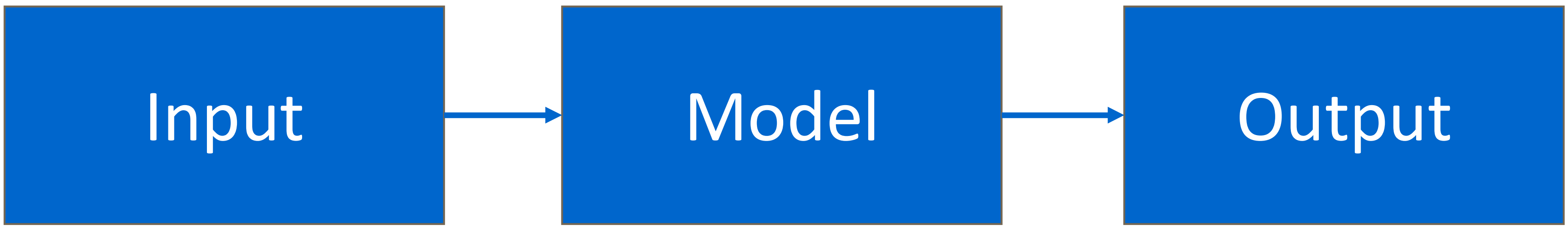
Let’s get into the flow



What kind of data do you use in your work area?

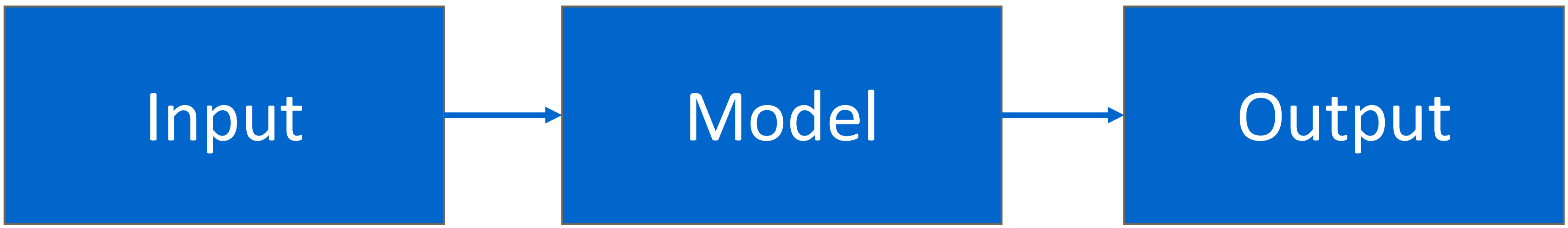
What are the outputs that you are interested in?

Let’s get into the flow



What kind of data do you use in your work area?

Let’s get into the flow



What are the outputs that you are interested in?

Let’s reminiscence

Back to secondary (or primary?) school

What does this equation describe?

$$\boxed{Y = AX + B}$$

Machine learning is an equation

There is no magic or sentient being working behind the scenes (at least for now)

The 'machine' in machine learning is clear.

What is being 'learnt'?

Which part of the equation represents the input?

Which part of the equation represents the output?

$$\boxed{Y = AX + B}$$

We learnt this in primary school, remember?

How do I arrive at the formulation?

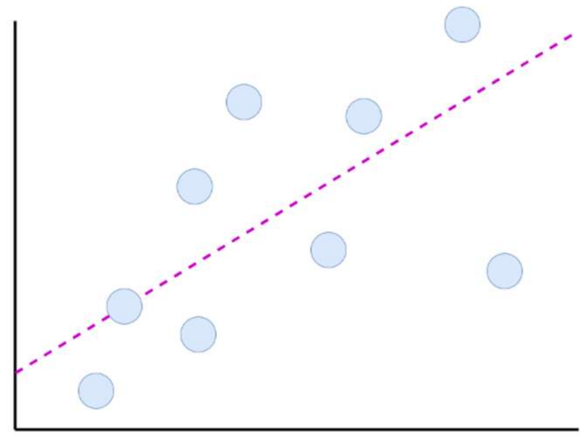
$$Y = AX + B$$

How good are the predictions, responses, targets, or dependent variables?

What are the characteristics of the inputs, features, or independent variables?

Are the coefficients, weights, or parameters statistically significant?

How do I find, learn, or train, these coefficients, weights, or parameters



Machine learning

$$Y = AX + B$$

Input → Model → Output

X A,B Y

Recollect this…

$$Y = AX + B$$

Input → Model → Output

3x1

Shape of A?

2x1

3x1 2x1 2x3

Supervised learning

$$Y = AX + B$$

Input → Model → Output

X A,B Y

Data Labels

[Cat, Dog]

Supervised learning

$$Y = AX + B$$

Input → Model → Output

X A,B Y

Data Labels

→ $1,500

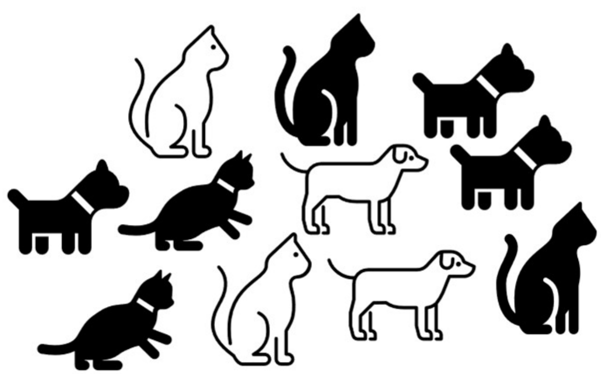
Unsupervised learning

$$Y = AX + B$$

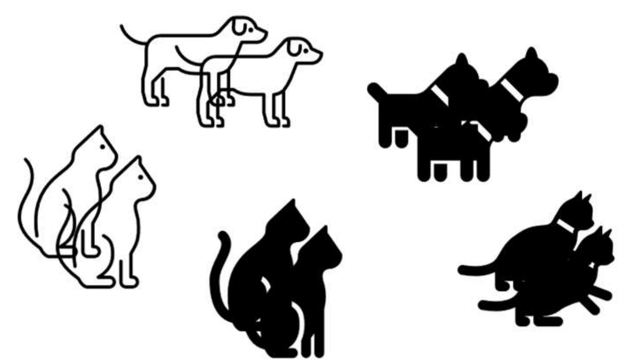
Input → Model → Output

X | A,B | Y

Data



Groupings



Common Models

Supervised Learning

- Decision trees
- Random Forest
- XGBoost
- K-nearest neighbors
- Linear discriminant analysis
- Linear regression, Logistic regression
- Support vector machines

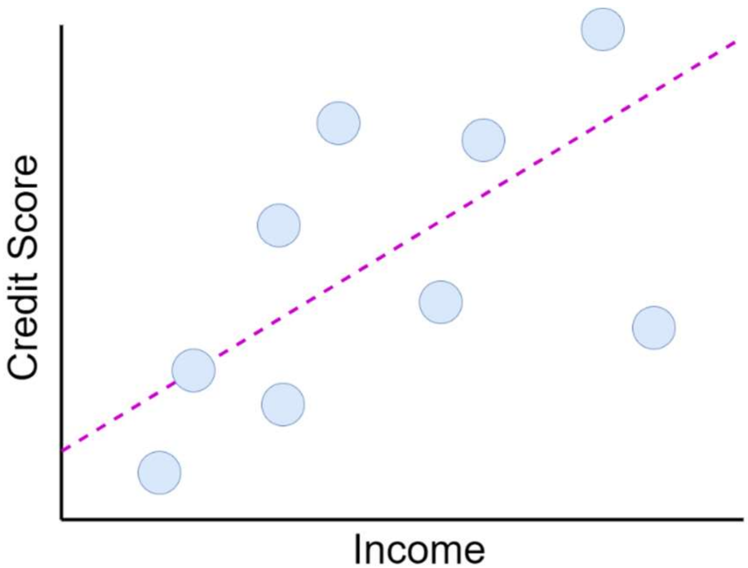
Unsupervised Learning

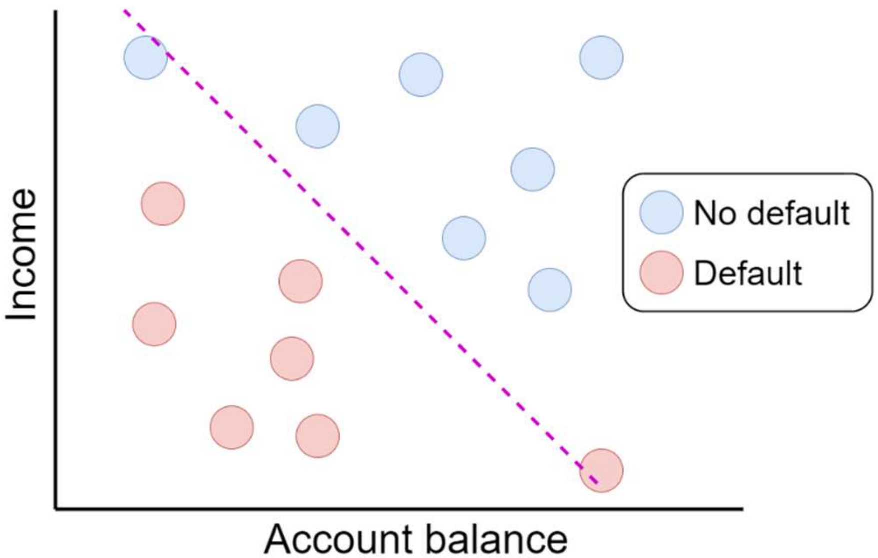
- Clustering - K-means
- Isolation Forest
- Dimensionality reduction - Principal component analysis
- Latent Dirichlet Allocation – Topic Modelling

Neural Networks

- Multilayer Perceptron/Dense Neural Network
- Convolutional Neural Network
- Transformer
- Recurrent Neural Network
- Graph Neural Network

Warm-Up



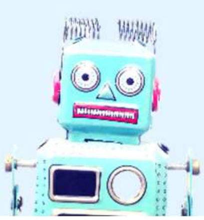


- What are the labels and inputs?
- Which figure shows a regression, which shows a classification?
- How would you describe the relationship between inputs and outputs?

Tabular and network datasets

- What are the differences between what you saw previously and such datasets?
- What are the key distinct differences between these data-types?
- Which of these are structured? Unstructured?

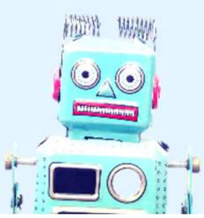
Core difference between ML and DL relates to feature engineering



Feature engineering

- Select features
  - *Recall k-best?*
- Transform non-linear to linear problem
  - *Real-world problems are usually not* $Y = AX + B$*!*
- Capture interactions
  - *Think about BMI and TDSR and constituents*
- Utilize unstructured inputs
  - *Think about networks vs. tabular datasets*

Tabular and network datasets



Tabular datasets

SAR	kycRiskScore	income	tenureMonths	creditScore	state	nbrPurchases90d	avgTxnSize90d	totalSpend90d	nbrDistinctMerch90d
0	3	110300	5	757	PA	10	153.8	1538	7
0	2	107800	6	715	NY	22	1.59	34.98	11
0	1	74000	13	751	MA	7	57.64	403.48	4
0	0	57700	1	659	NJ	14	29.52	413.28	7
0	1	59800	3	709	PA	54	115.77	6251.58	16
0	1	43500	11	717	CT	18	36.11	649.98	11
0	0	70200	9	720	ME	17	55.38	941.46	7
1	1	5900	1	772	MA	0	36.88	0	0
0	1	11400	43	727	NY	2	159.05	318.1	1
0	1	36700	12	735	PA	86	37.25	3203.5	41
0	0	43700	4	660	CT	19	6.49	123.31	14

- What might be interesting to predict?
  - See https://pathfinder.datarobot.com/use-case/reduce-false-positives-for-anti-money-laundering-aml?tab=tech for explanation on the columns
- What are the inputs used to predict the variable of interest?

Tabular datasets

SAR	kycRiskScore	income	tenureMonths	creditScore	state	nbrPurchases90d	avgTxnSize90d	totalSpend90d	nbrDistinctMerch90d
0	3	110300	5	757	PA	10	153.8	1538	7
0	2	107800	6	715	NY	22	1.59	34.98	11
0	1	74000	13	751	MA	7	57.64	403.48	4
0	0	57700	1	659	NJ	14	29.52	413.28	7
0	1	59800	3	709	PA	54	115.77	6251.58	16
0	1	43500	11	717	CT	18	36.11	649.98	11
0	0	70200	9	720	ME	17	55.38	941.46	7
1	1	5900	1	772	MA	0	36.88	0	0
0	1	11400	43	727	NY	2	159.05	318.1	1
0	1	36700	12	735	PA	86	37.25	3203.5	41
0	0	43700	4	660	CT	19	6.49	123.31	14

- Numerical vs. categorical
- Different ranges
- Missing values

Tabular datasets

SAR	kycRiskScore	income	tenureMonths	creditScore	state	nbrPurchases90d	avgTxnSize90d	totalSpend90d	nbrDistinctMerch90d
0	3	110300	5	757	PA	10	153.8	1538	7
0	2	107800	6	715	NY	22	1.59	34.98	11
0	1	74000	13	751	MA	7	57.64	403.48	4
0	0	57700	1	659	NJ	14	29.52	413.28	7
0	1	59800	3	709	PA	54	115.77	6251.58	16
0	1	43500	11	717	CT	18	36.11	649.98	11
0	0	70200	9	720	ME	17	55.38	941.46	7
1	1	5900	1	772	MA	0	36.88	0	0
0	1	11400	43	727	NY	2	159.05	318.1	1
0	1	36700	12	735	PA	86	37.25	3203.5	41
0	0	43700	4	660	CT	19	6.49	123.31	14

- Numerical vs. categorical
- Different ranges
- Missing values

Tabular datasets

SAR	kycRiskScore	income	tenureMonths	creditScore	state	nbrPurchases90d	avgTxnSize90d	totalSpend90d	nbrDistinctMerch90d
0	3	110300	5	757	PA	10	153.8	1538	7
0	2	107800	6	715	NY	22	1.59	34.98	11
0	1	74000	13	751	MA	7	57.64	403.48	4
0	0	57700	1	659	NJ	14	29.52	413.28	7
0	1	59800	3	709	PA	54	115.77	6251.58	16
0	1	43500	11	717	CT	18	36.11	649.98	11
0	0	70200	9	720	ME	17	55.38	941.46	7
1	1	5900	1	772	MA	0	36.88	0	0
0	1	11400	43	727	NY	2	159.05	318.1	1
0	1	36700	12	735	PA	86	37.25	3203.5	41
0	0	43700	4	660	CT	19	6.49	123.31	14

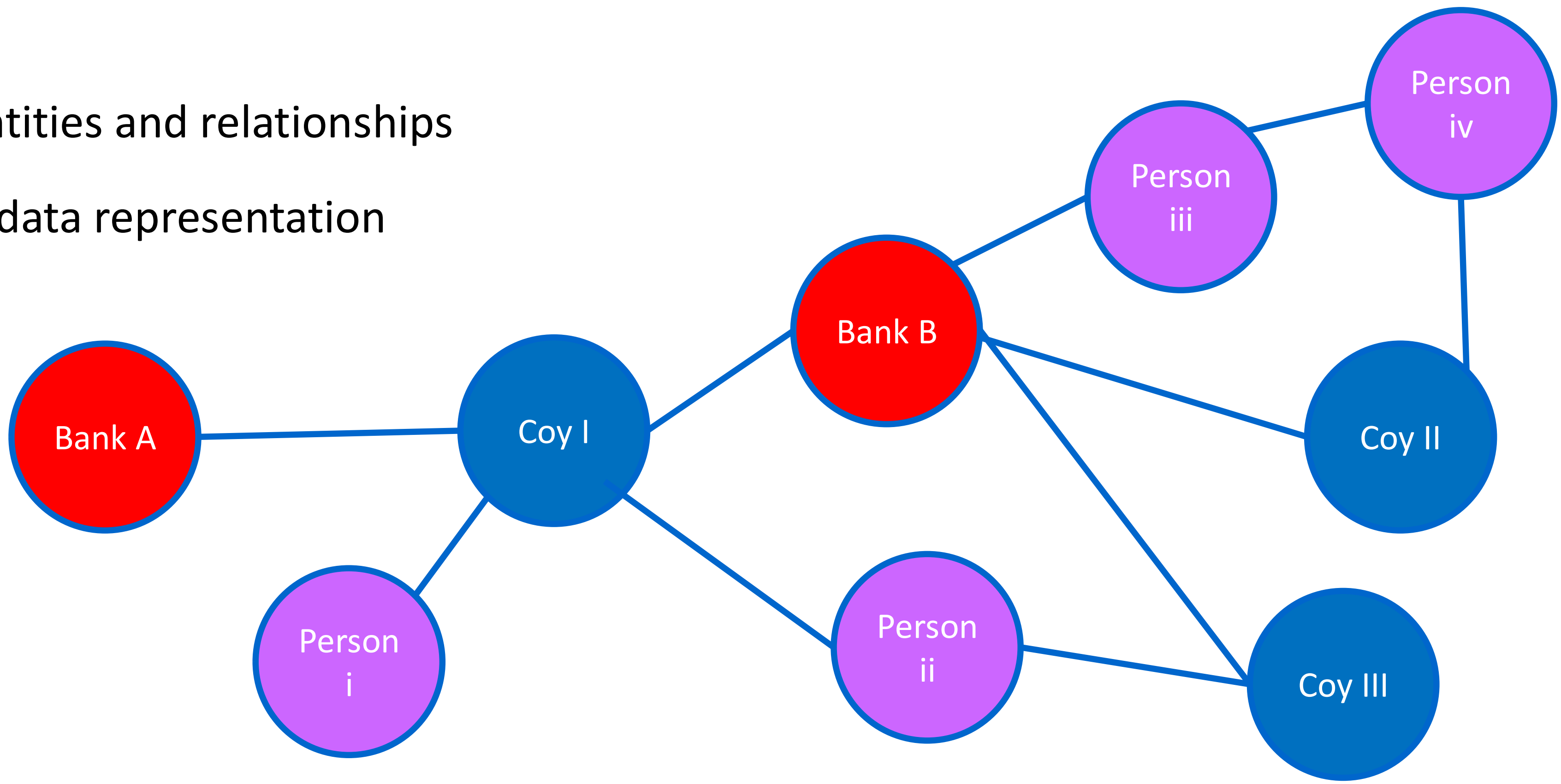
Tabular datasets

SAR	kycRiskScore	income	tenureMonths	creditScore	state	nbrPurchases90d	avgTxnSize90d	totalSpend90d	nbrDistinctMerch90d
0	3	110300	5	757	PA	10	153.8	1538	7
0	2	107800	6	715	NY	22	1.59	34.98	11
0	1	74000	13	751	MA	7	57.64	403.48	4
0	0	57700	1	659	NJ	14	29.52	413.28	7
0	1	59800	3	709	PA	54	115.77	6251.58	16
0	1	43500	11	717	CT	18	36.11	649.98	11
0	0	70200	9	720	ME	17	55.38	941.46	7
1	1	5900	1	772	MA	0	36.88	0	0
0	1	11400	43	727	NY	2	159.05	318.1	1
0	1	36700	12	735	PA	86	37.25	3203.5	41
0	0	43700	4	660	CT	19	6.49	123.31	14

- Assume 10,000 rows, machine or deep learning?
    - Why?

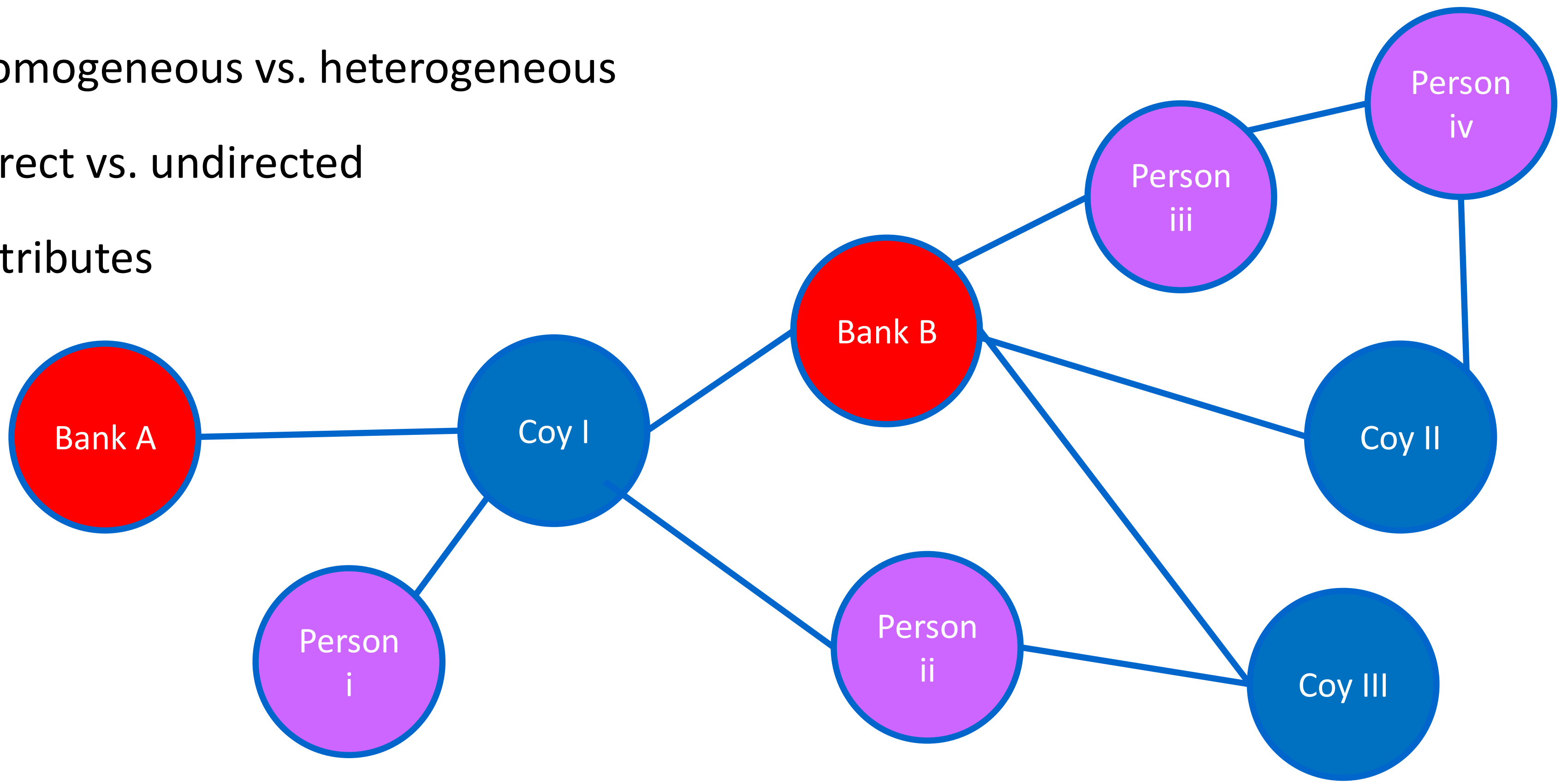
Network datasets

- Entities and relationships
- A data representation



Network datasets

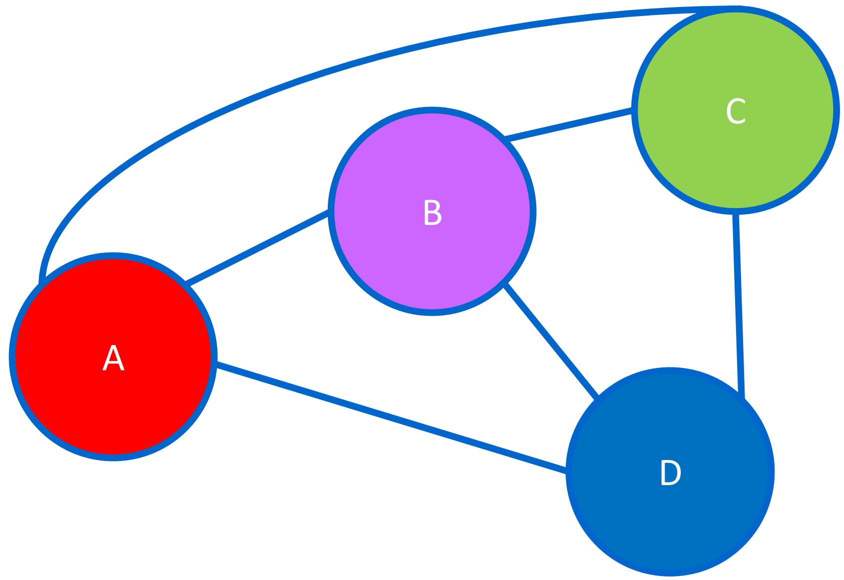
- Homogeneous vs. heterogeneous
- Direct vs. undirected
- Attributes



Network dataset representation

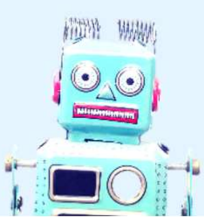
Let's try constructing:

- **Adjacency matrix**
- **Edgelist**
- Adjacency list



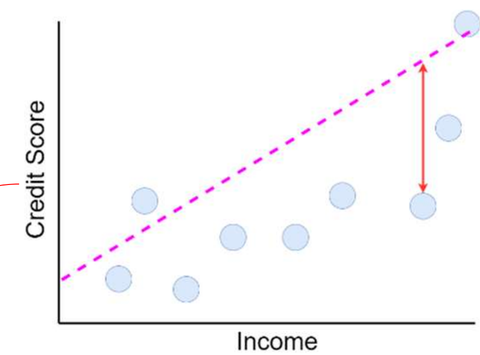
Machine Learning:

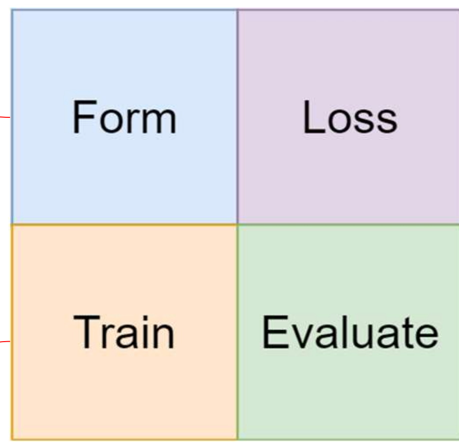
Trees and Forests to XGBoost



Framework: Linear Regression

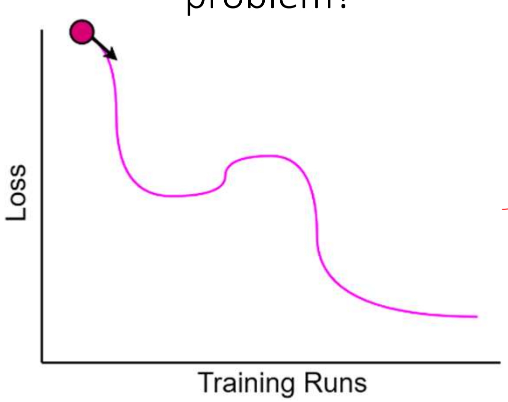
$$y = AX + B$$





Gradient descent, one of many ways to train/optimize.

Can you spot a common problem?



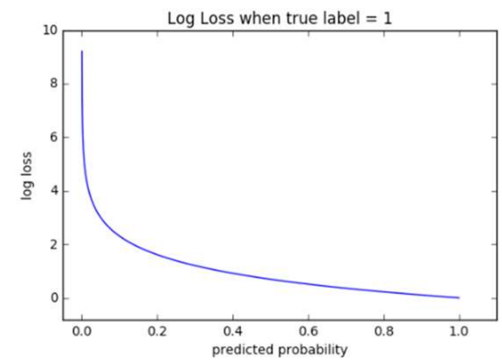
Root Mean Squared Error, Mean Abs. Error, Mean Abs. Percentage Error

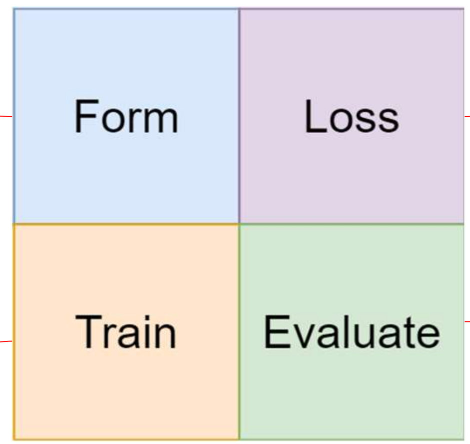
Framework: Logistic Regression

$$P = \frac{1}{1 + e^{-(AX+B)}}$$

Cross-Entropy

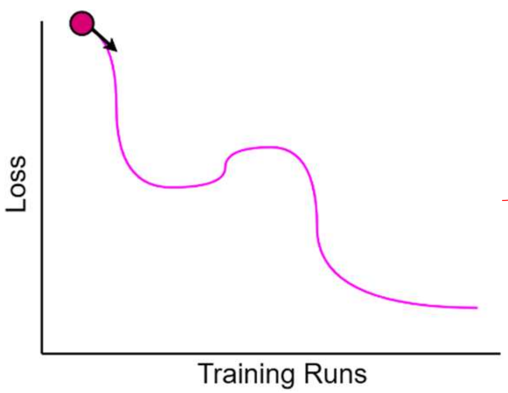
$$-Y log(P) - (1 - Y)\log(1 - P)$$





Gradient descent not the only way

Also Max. Likelihood Est.

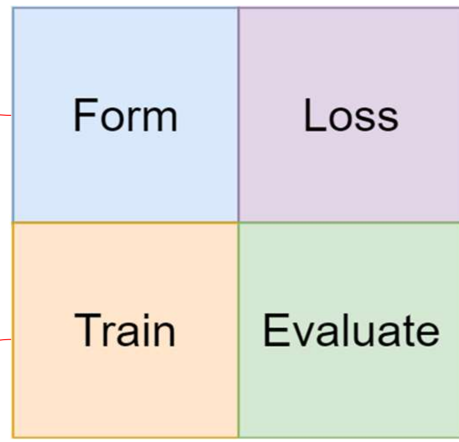


Accuracy, Recall, Precision, F1

Framework: Decision Tree

What is the form?

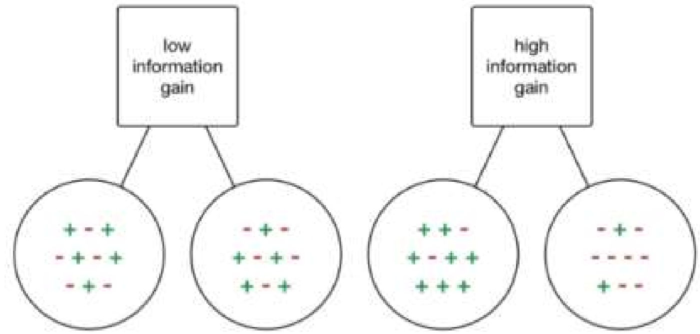
What would be a good criteria to split?



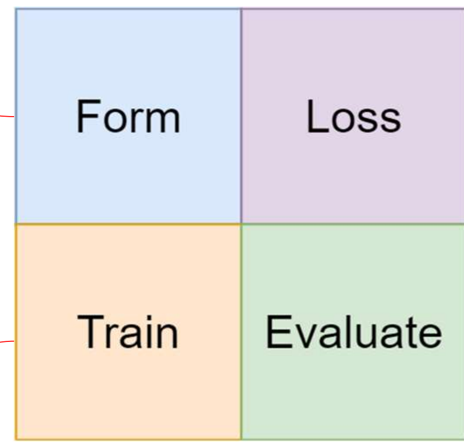
Given the splitting criteria, how could one build the tree?

Recall what we used for logistic regression

Framework: Decision Tree



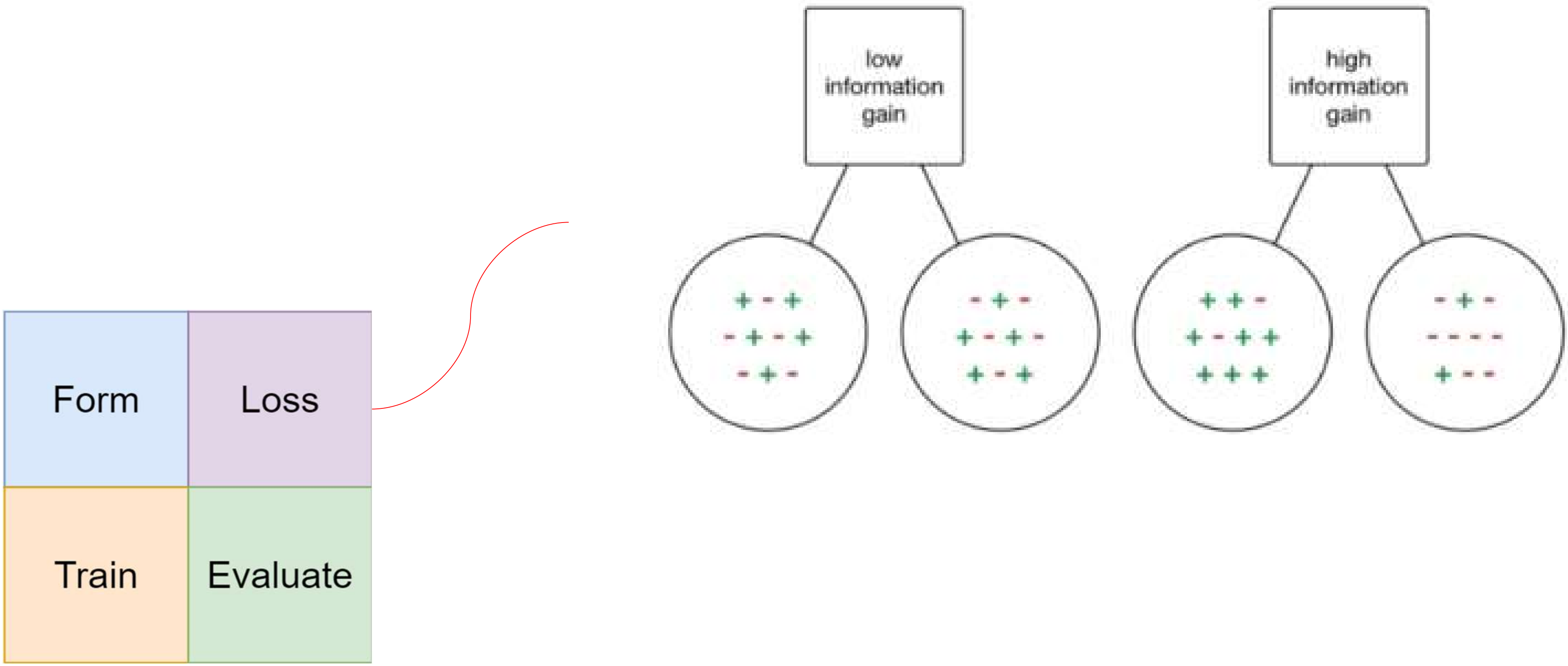
Binary Tree



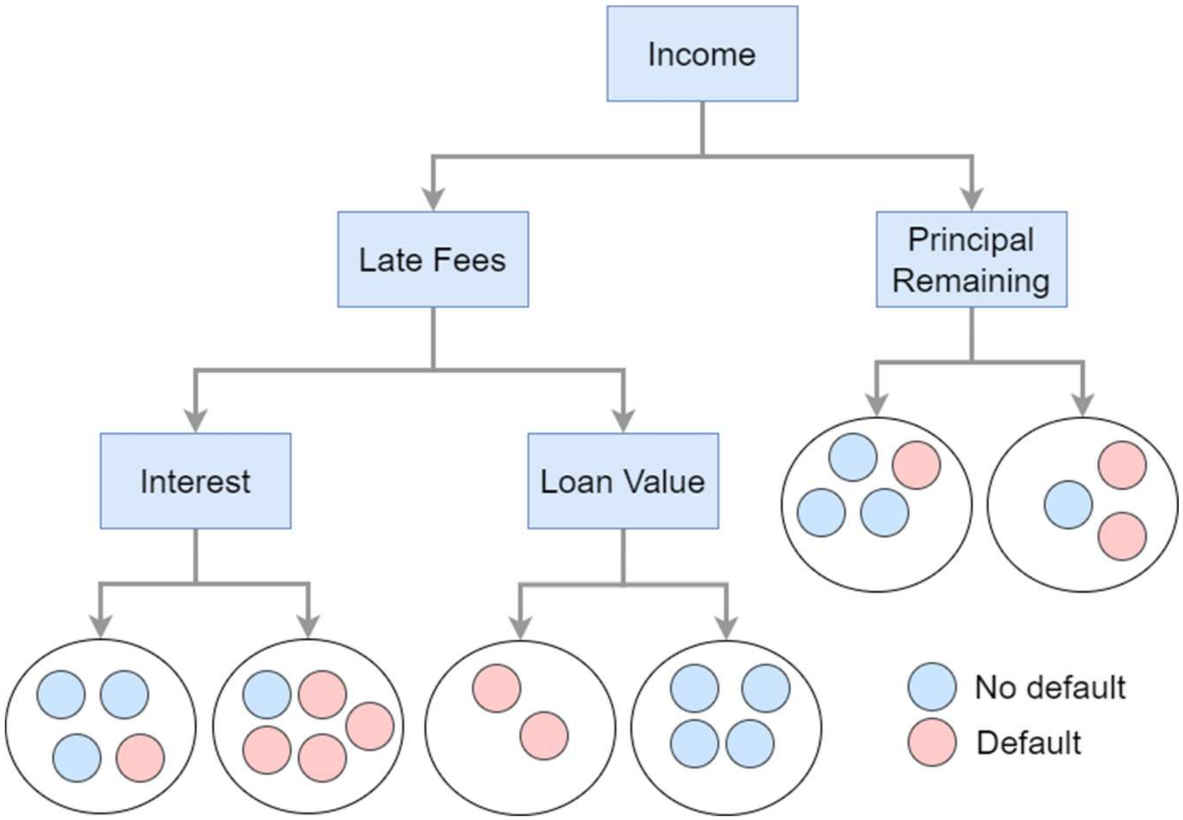
Greedy

Accuracy, Recall, Precision, F1

Framework: Decision Tree



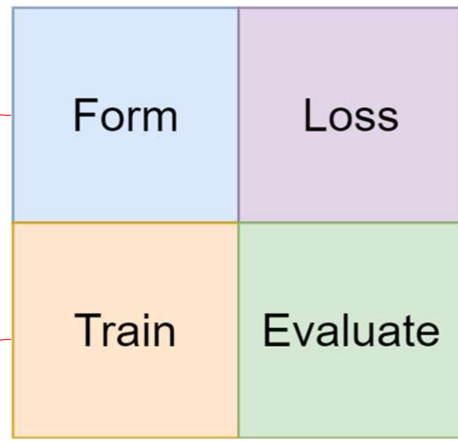
Decision Tree



Framework: Decision Tree

What would be a good criteria to split?

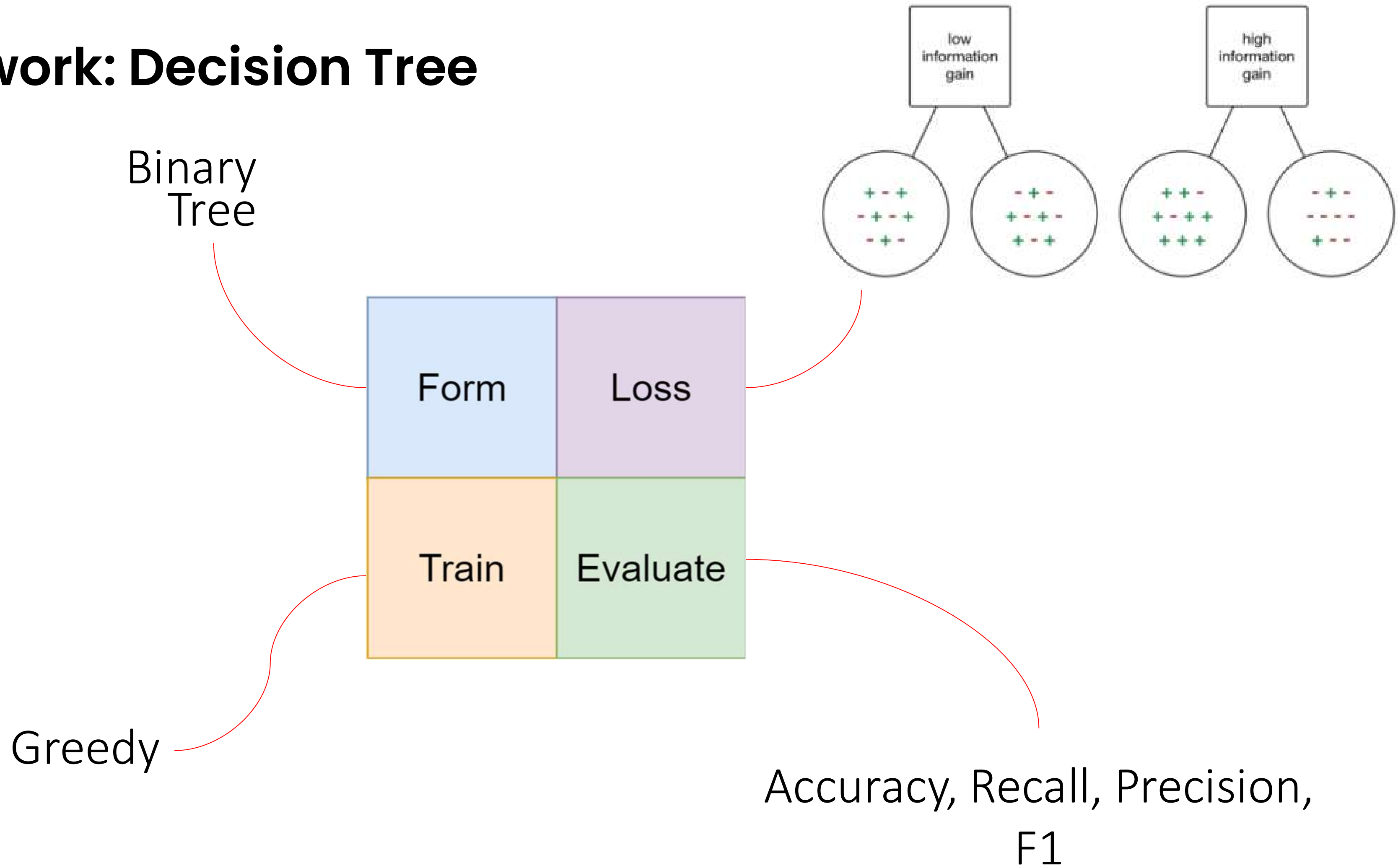
What is the form?



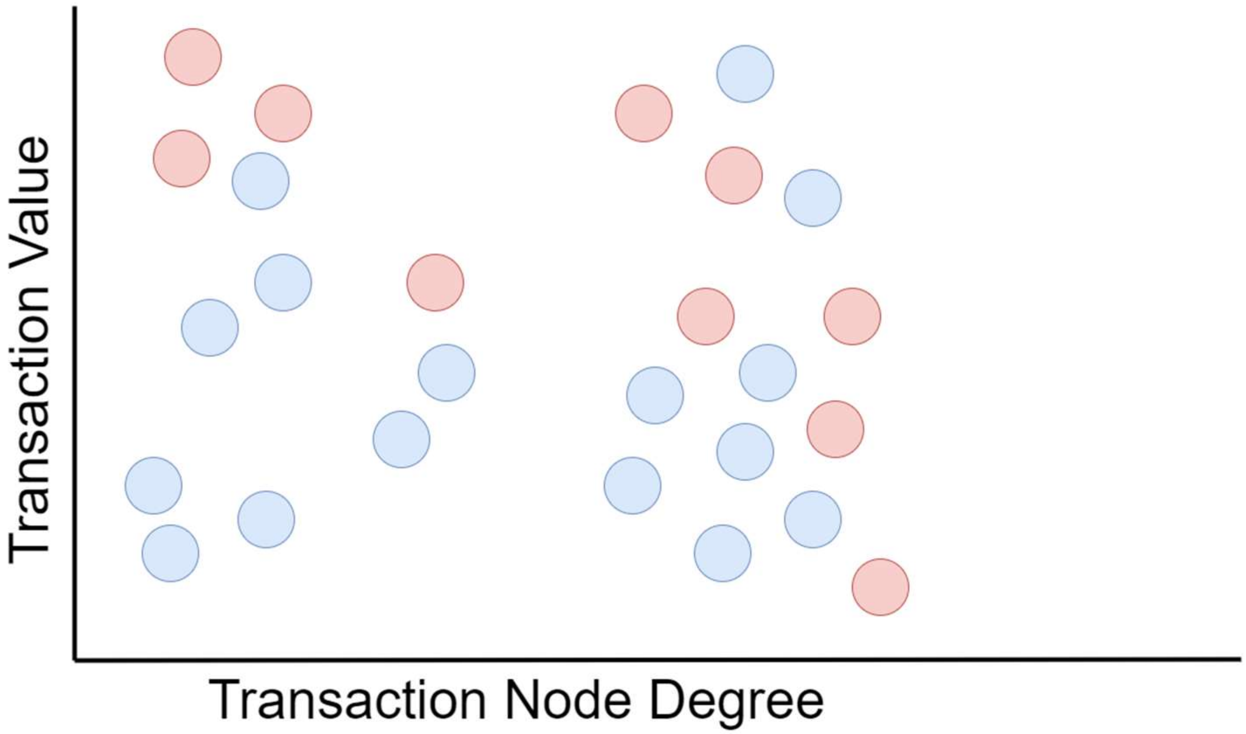
Given the splitting criteria, how could one build the tree?

What would you like to measure?

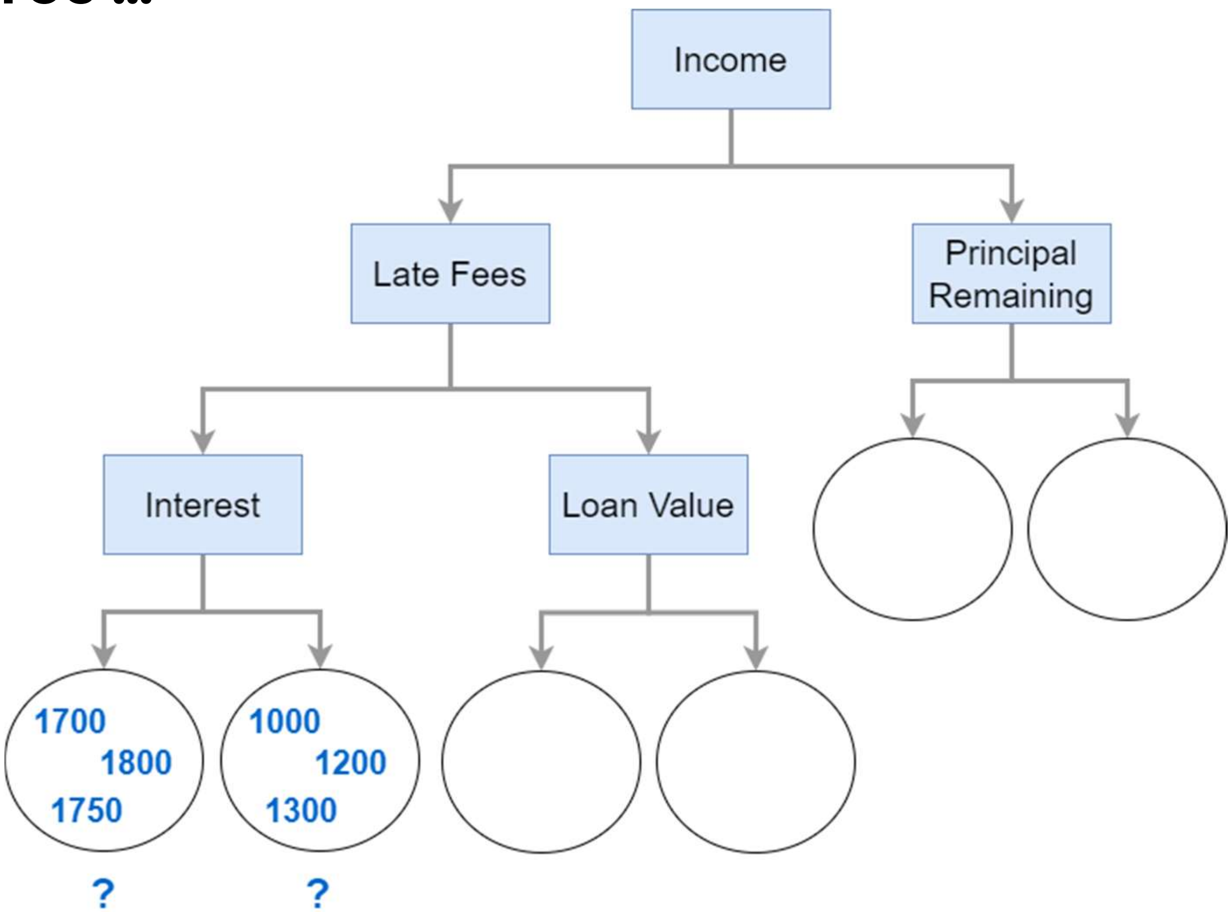
Framework: Decision Tree



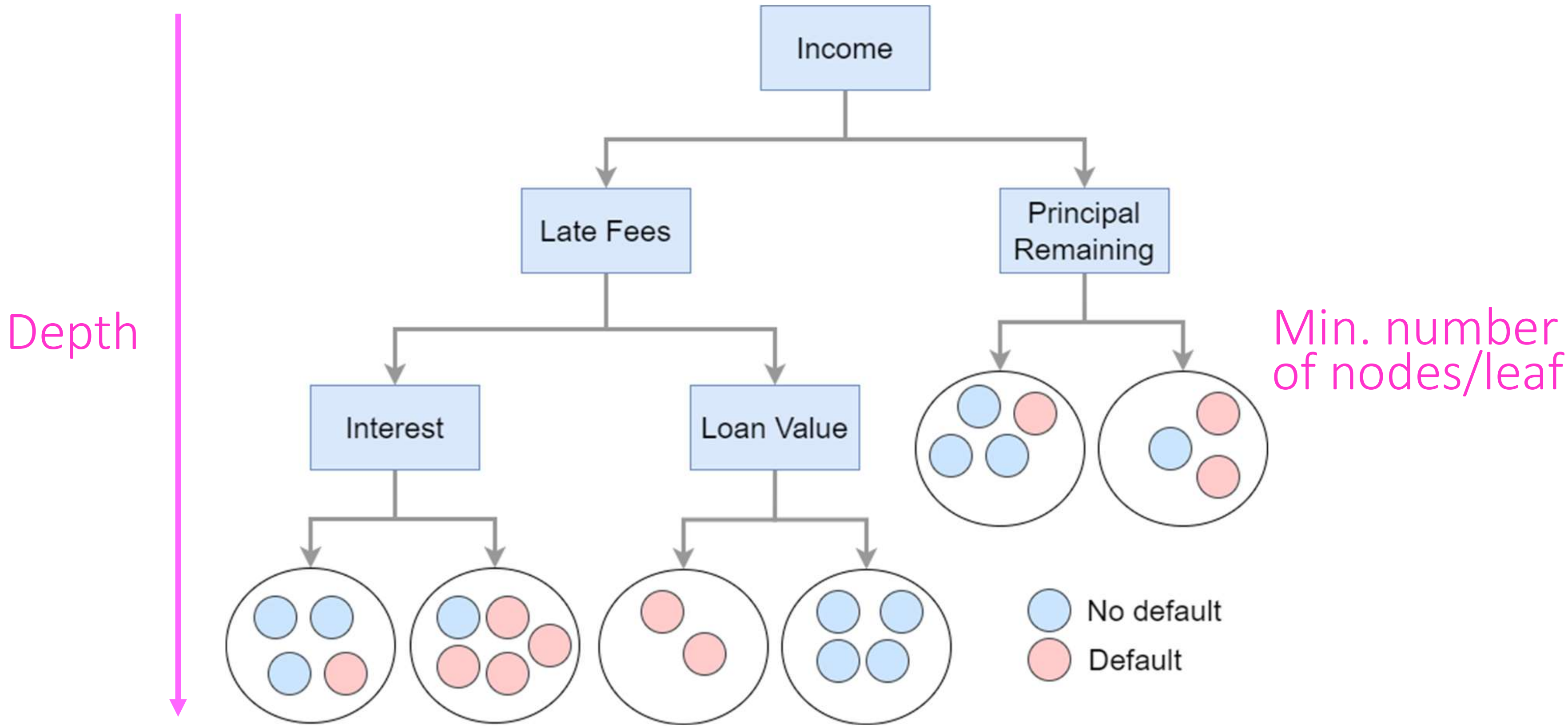
Decision Trees – Simple Example



Now think about how this would work for regression on credit scores …

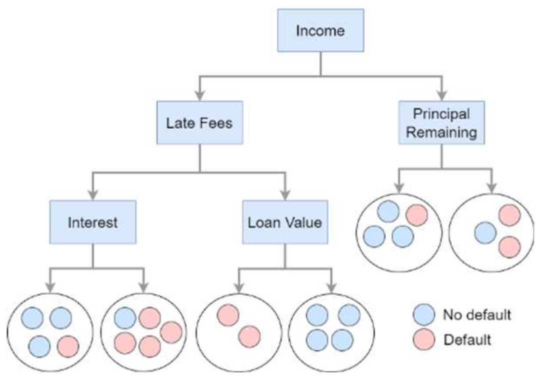


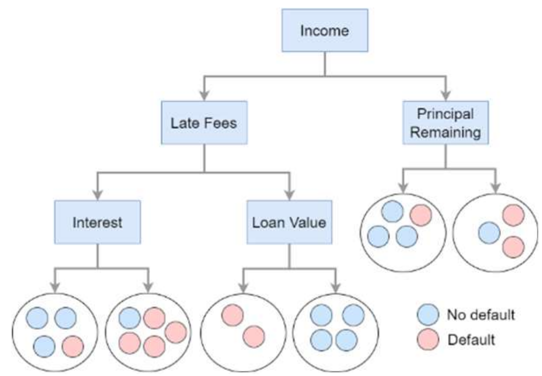
Hyperparameters

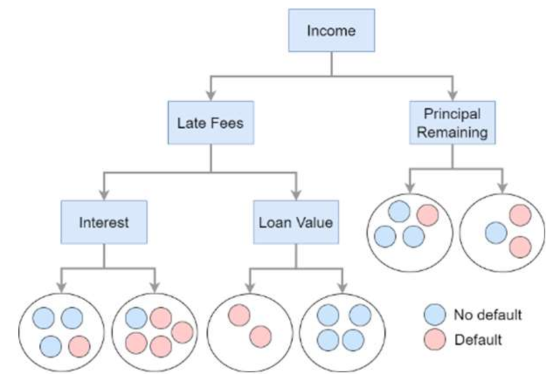



```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)
```

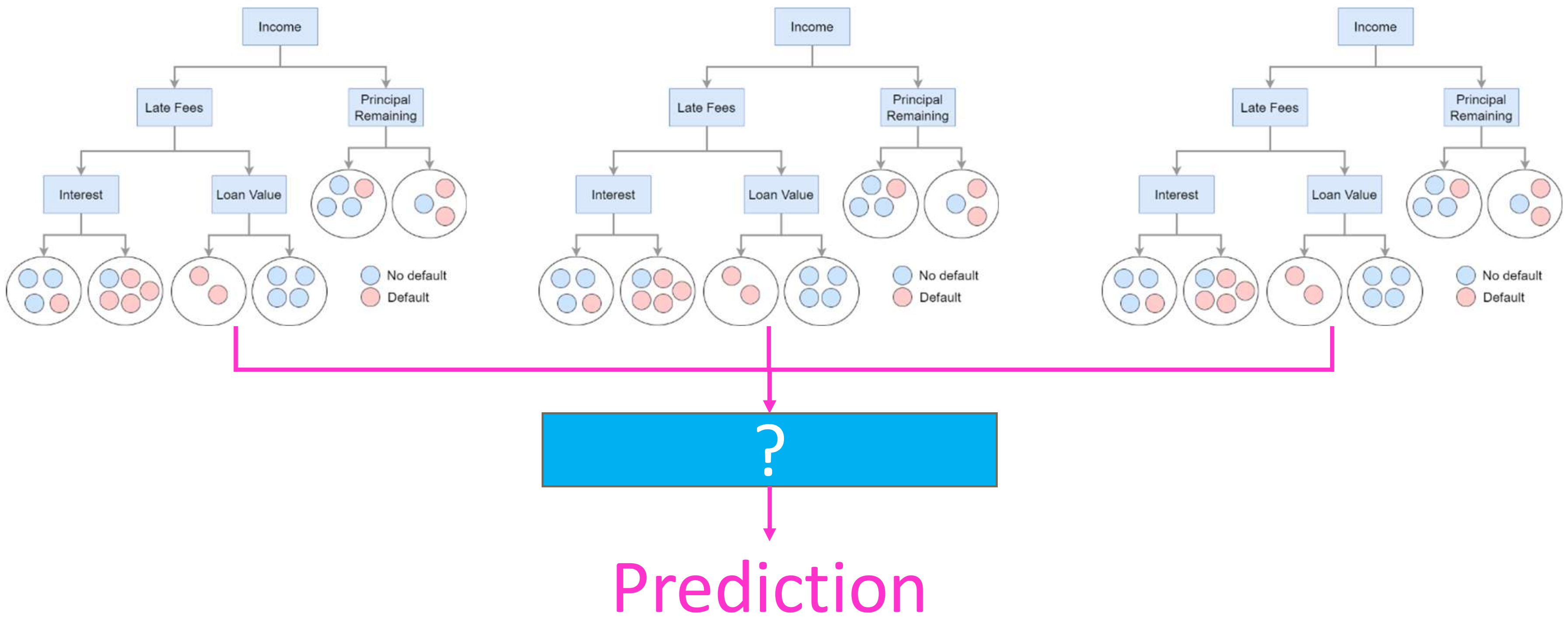
Why stop at one decision tree?



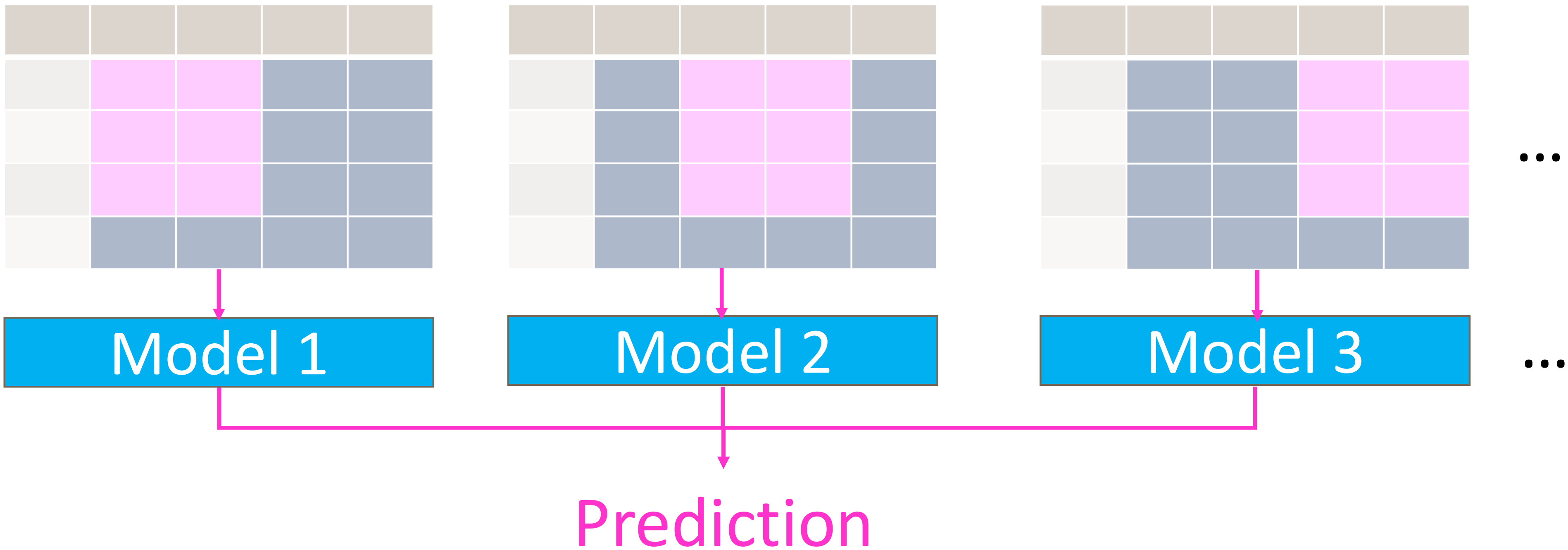




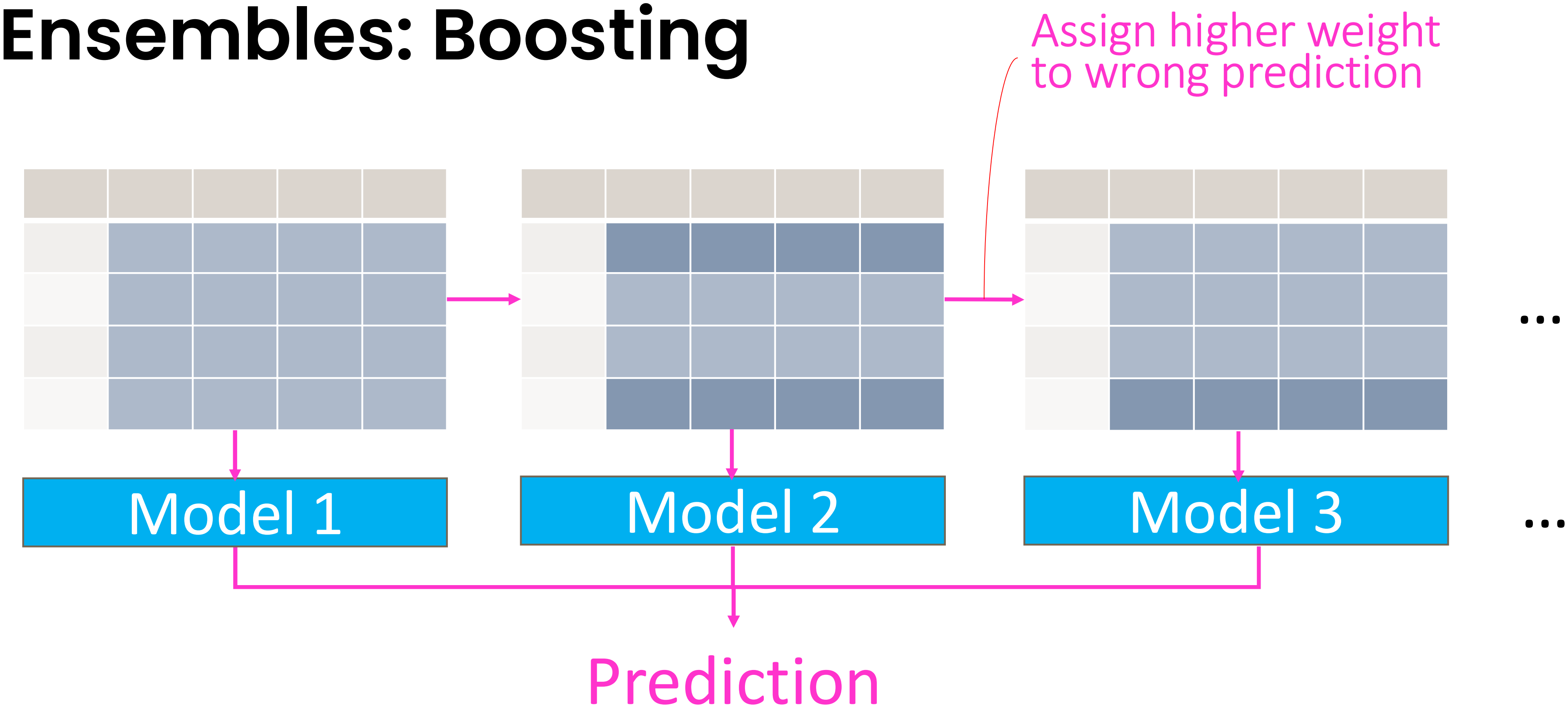
Ensembles


Income
Late Fees
Principal Remaining
Interest
Loan Value
No default
Default
Income
Late Fees
Principal Remaining
Interest
Loan Value
No default
Default
Income
Late Fees
Principal Remaining
Interest
Loan Value
No default
Default
?
Prediction

Ensembles: Bagging



Ensembles: Boosting



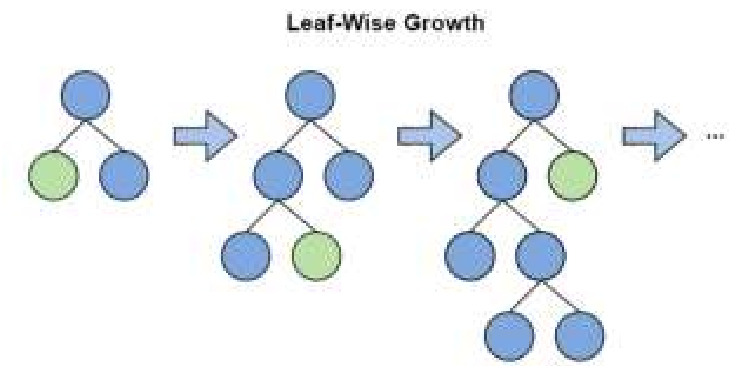
XGBoost

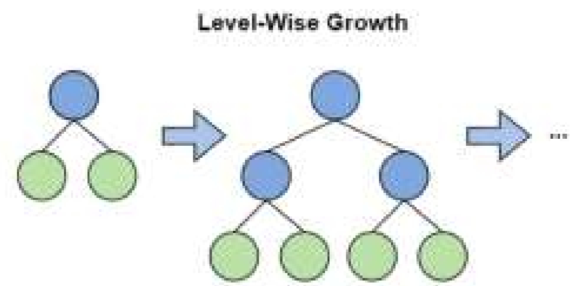
XGBoost is basically based on the idea of boosting, but with some additional math and optimization

For the curious, more details available at https://xgboost.readthedocs.io/en/stable/tutorials/model.html

XGBoost vs. LightGBM

LightGBM grows leaf-wise (horizontally) while XGBoost grows level-wise (vertically)





LightGBM

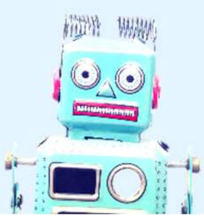
XGBoost

For the curious, more details available at https://towardsdatascience.com/catboost-vs-lightgbm-vs-xgboost-c80f40662924

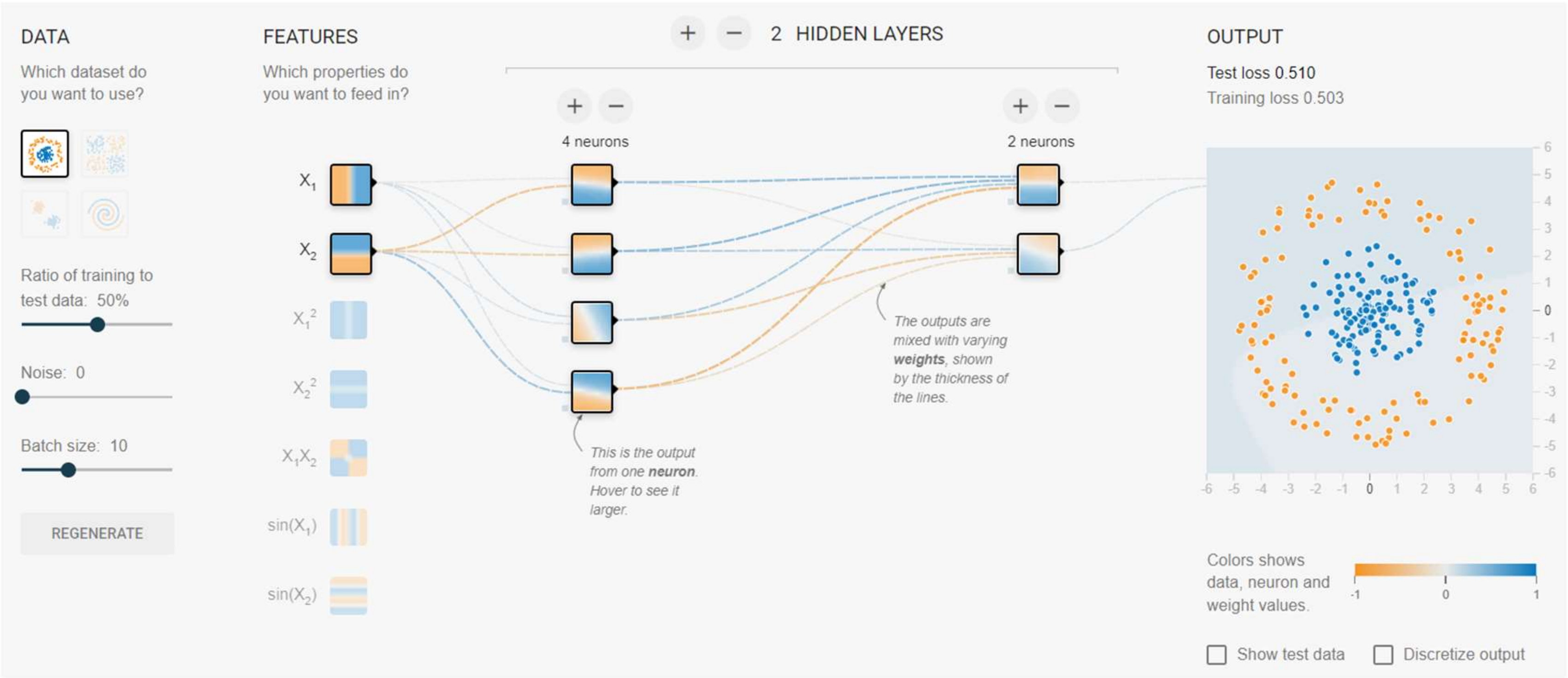
Libraries

- **Scikit Learn - https://scikit-learn.org/stable/**
  - Most machine learning libraries can be found in this library
- **XGBoost - https://xgboost.readthedocs.io**
  - Very popular go to, should work in most cases for tabular datasets
- **LightGBM - https://lightgbm.readthedocs.io**
  - Can be faster than XGBoost
- **CatBoost - https://catboost.ai/**
  - Works well for categorical datasets
- Some deep learning models for tabular datasets but not clear that better than machine learning models across tasks
  - Example, TabNet - https://github.com/google-research/google-research/tree/master/tabnet

From Machine Learning to Deep Learning



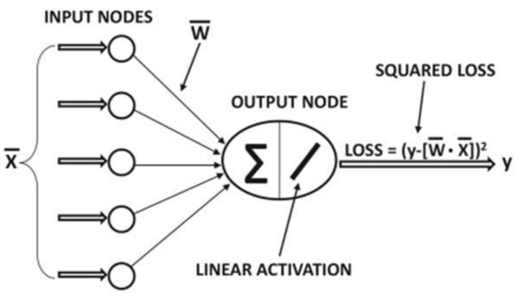
Neural Networks



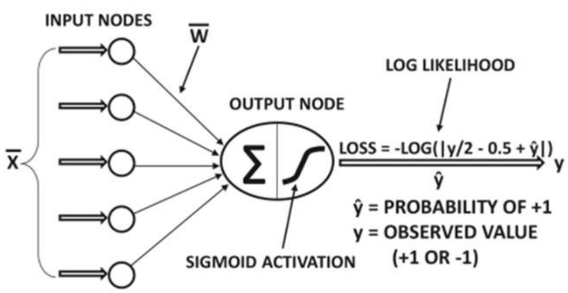
https://playground.tensorflow.org/

Framework: Neural Networks

Linear Regression



Logistic Regression



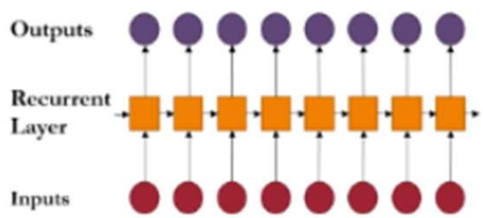
Gradient descent

Form	Loss
Train	Evaluate

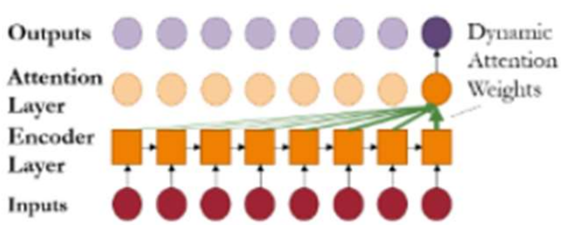
Cross-Entropy Loss, Squared Error Loss

Accuracy, Recall, Precision, F1, Root Mean Squared Error, Mean Abs. Error, Mean Abs. Percentage Error

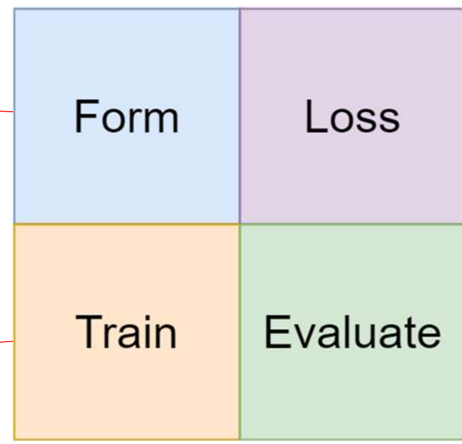
Framework: Neural Networks



(b) RNN Model.



(c) Attention-based Model.

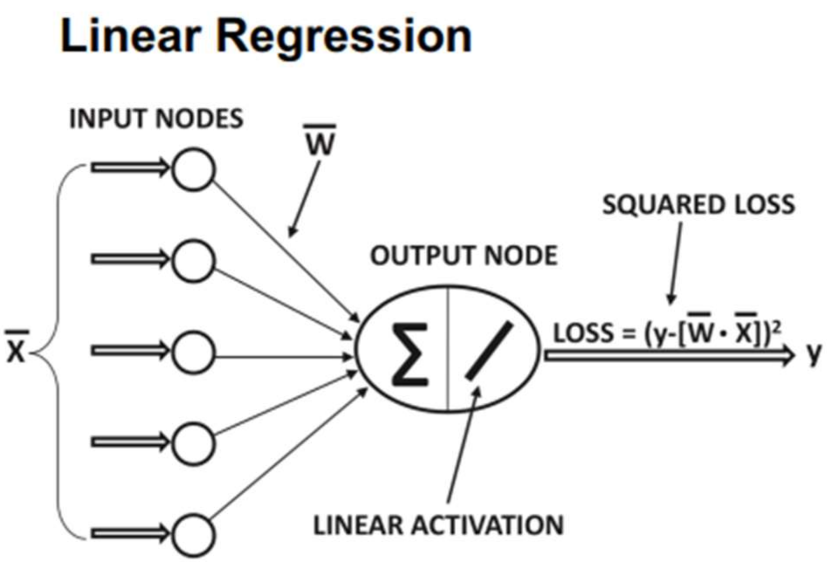


Cross-Entropy Loss, Squared Error Loss

Gradient descent

Accuracy, Recall, Precision, F1, Root Mean Squared Error, Mean Abs. Error, Mean Abs. Percentage Error

Neural Networks



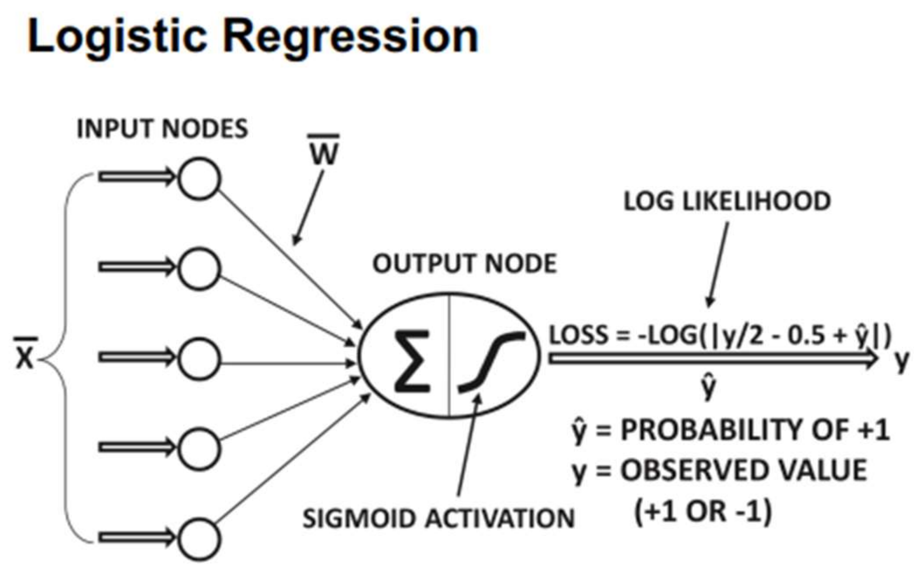


Figure from Neural Networks and Deep Learning, Charu Aggarwal

Neural Networks

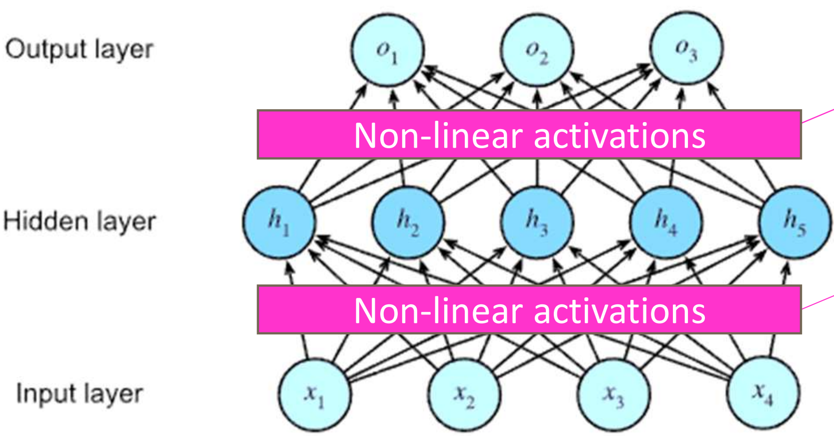


Figure from Dive into Deep Learning:
https://d2l.ai/chapter_multilayer-perceptrons/mlp.html

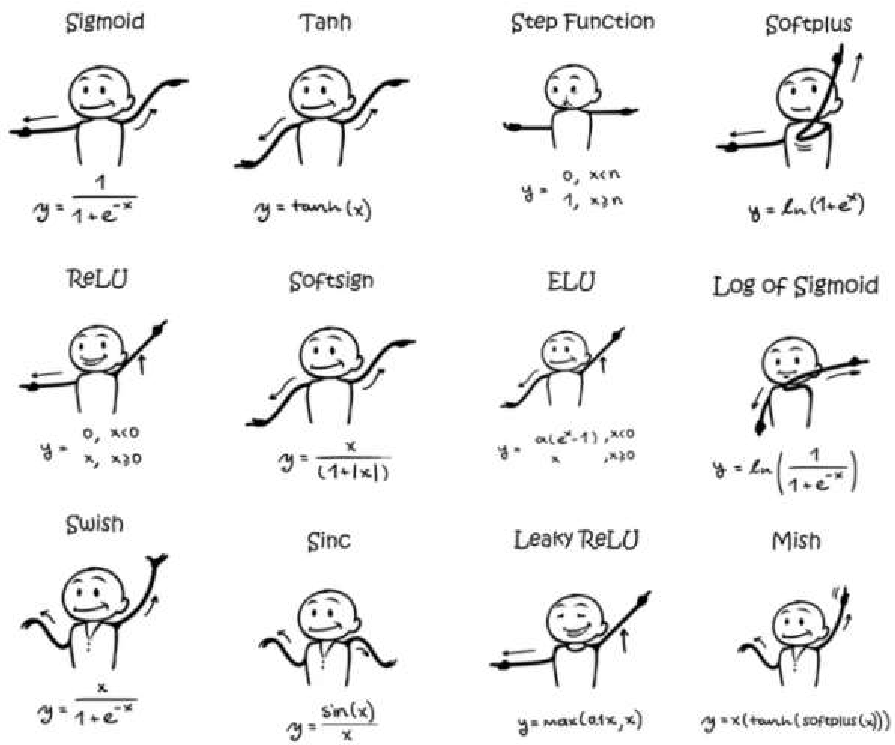
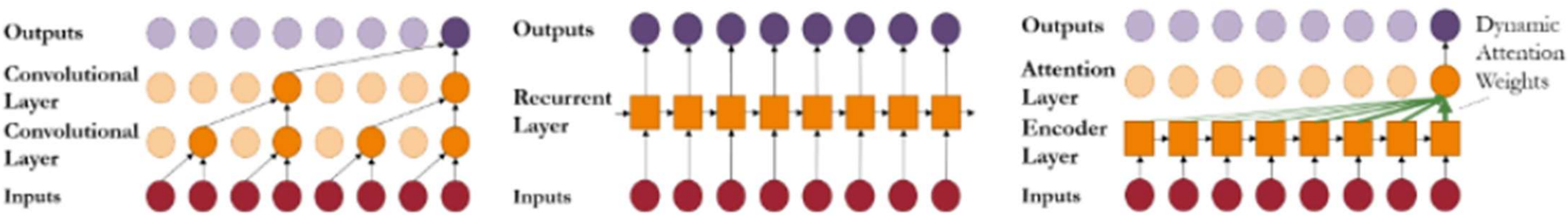


Figure from https://medium.com/analytics-vidhya/activation-functions-in-neural-network-55d1afb5397a

Neural Networks

We will go into CNN and RNNs more when in the next class when we look at multimodal datasets

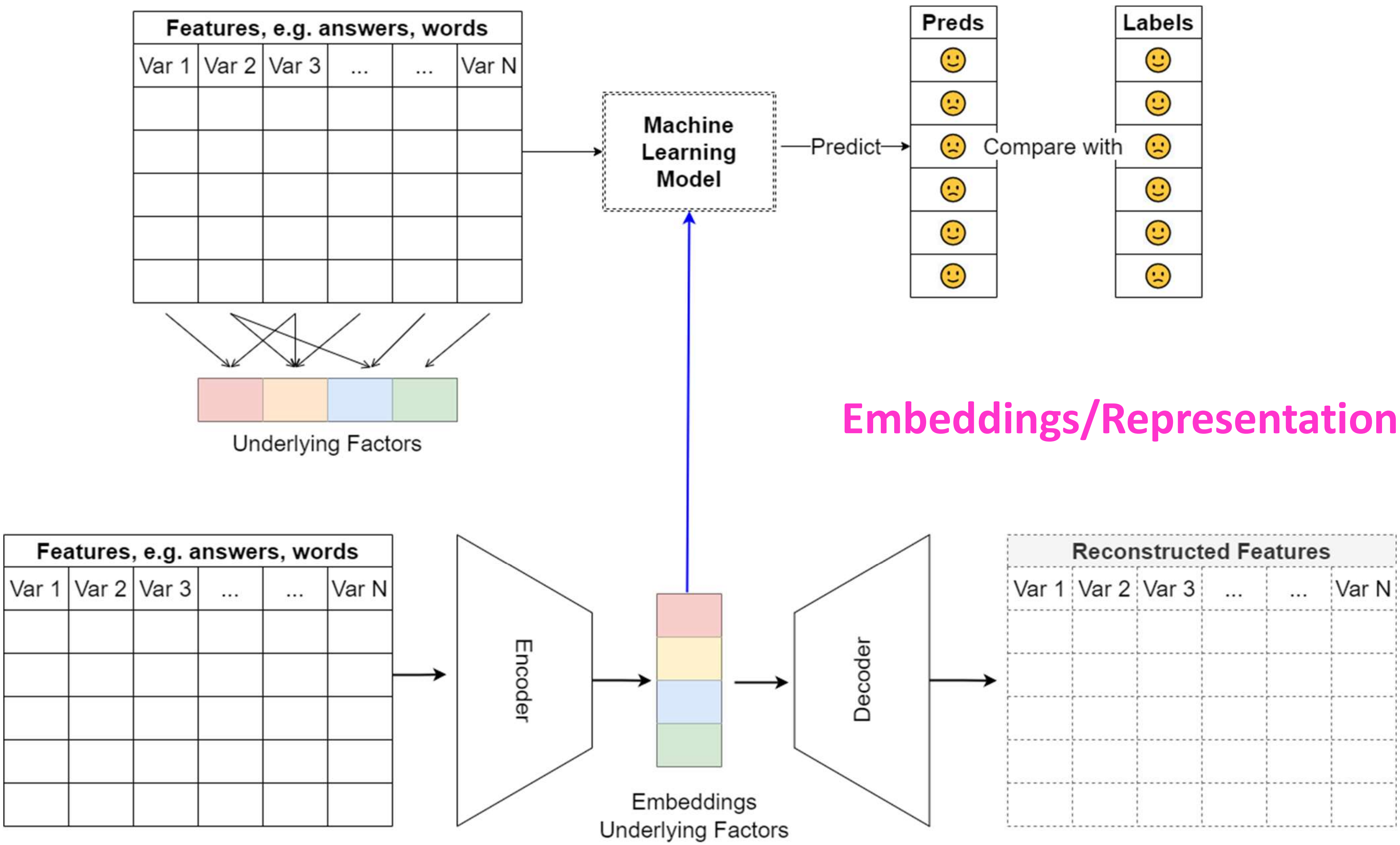


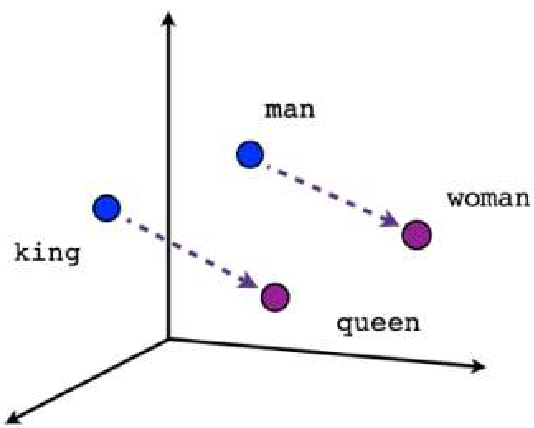
(a) CNN Model. (b) RNN Model. (c) Attention-based Model.

Figure 1: Incorporating temporal information using different encoder architectures.

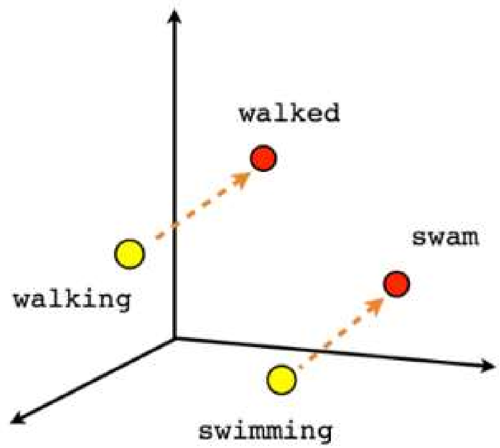
From Time Series Forecasting With Deep Learning: A Survey, Lim et al., 2020

Embeddings/Representations

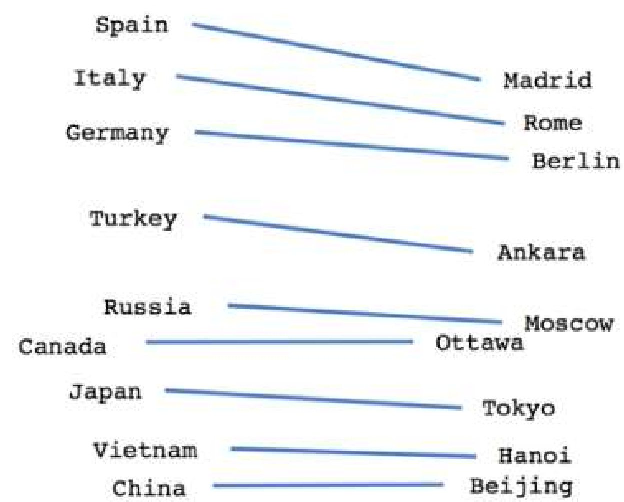




Male-Female



Verb tense

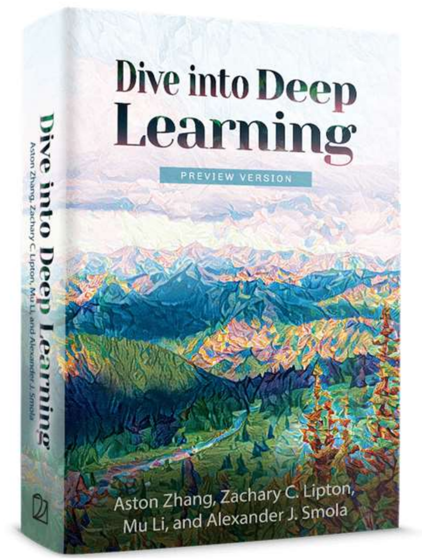


Country-Capital

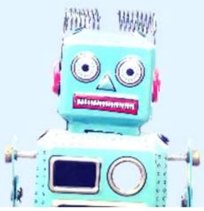
https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8

Libraries/Resources

- **One of the best books with hands-on : https://d2l.ai**
- **Tensorflow**
  - Google’s framework, good for production
- **Keras**
  - Wrapper around Tensorflow
- **Pytorch**
  - Meta’s framework, good for research, getting better for production
- **Pytorch Lightning**
  - Wrapper around Pytorch

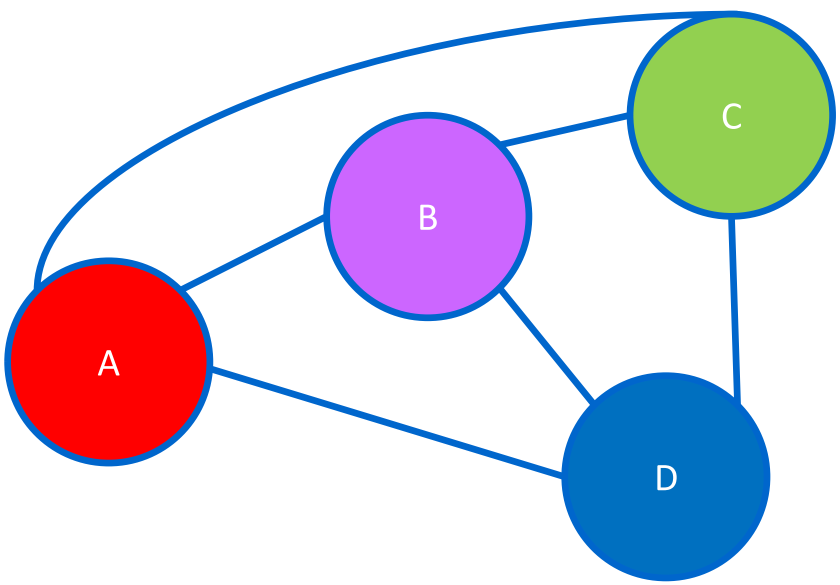


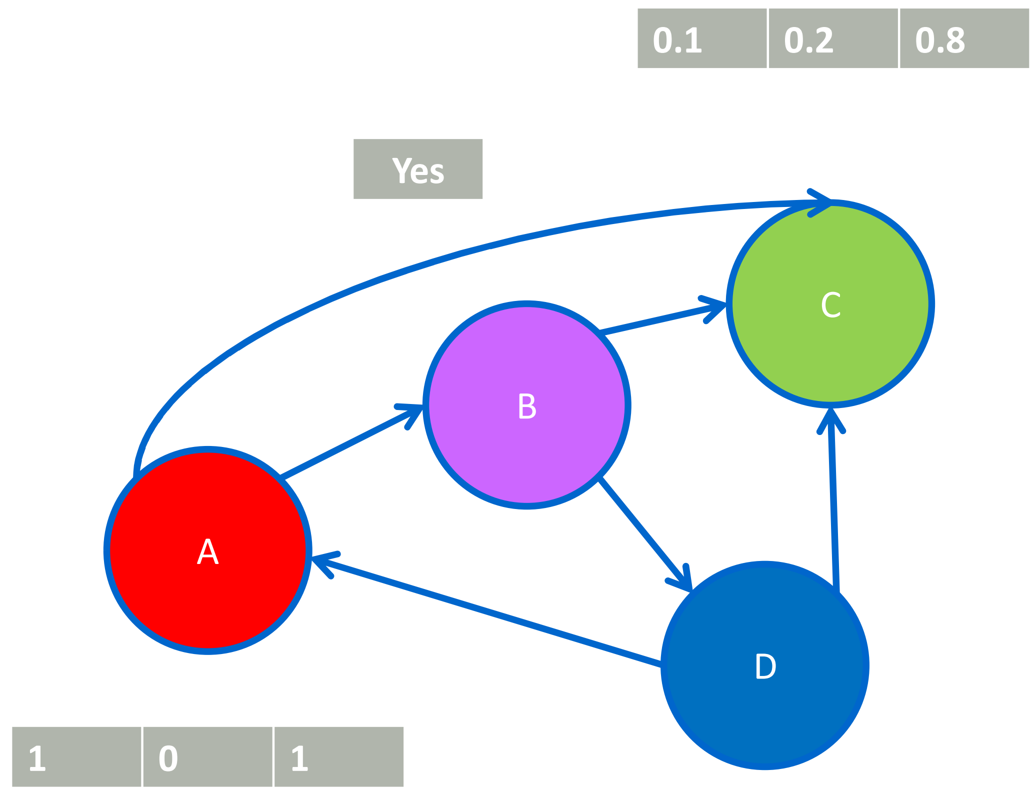
From Network Analysis to Graph Neural Networks



Networks or graphs 101

- Node and edges
- Node, edge and graph attributes
- Directed, undirected





I see graphs everywhere

- Provide some examples of graphs

Give some examples of network/graph tasks

- Node tasks
- Edge tasks
- Graph tasks

Key Network Statistics

- **Degree centrality**
  - How many other nodes are you connected to?
- **Betweenness centrality**
  - How many paths between nodes go through you?
- **Closeness centrality**
  - Which node can reach the most nodes in a network?
- **Eigenvector centrality**
  - How important are nodes connected to you?

Centrality

Degree

`degree_centrality`(G)	Compute the degree centrality for nodes.
`in_degree_centrality`(G)	Compute the in-degree centrality for nodes.
`out_degree_centrality`(G)	Compute the out-degree centrality for nodes.

Eigenvector

`eigenvector_centrality`(G[, max_iter, tol, ...])	Compute the eigenvector centrality for the graph `G`.
`eigenvector_centrality_numpy`(G[, weight, ...])	Compute the eigenvector centrality for the graph G.
`katz_centrality`(G[, alpha, beta, max_iter, ...])	Compute the Katz centrality for the nodes of the graph G.
`katz_centrality_numpy`(G[, alpha, beta, ...])	Compute the Katz centrality for the graph G.

Network Statistics Computation

- Let's just understand degree and betweenness centrality
- Can easily find out more online for other network statistics

$$C_D(i) = \sum_{j=1}^{N} \boxed{A_{ij}}$$

A - Adjacency Matrix

$$C_B(i) = \sum_{j}^{n} \sum_{k>j}^{n} \boxed{\frac{g_{jk}(i)}{g_{jk}}}$$

Fraction of paths between node j and node k that pass through node i

Network Statistics for ML

SAR	Account	Balance	Degree	Betweenness
1	A	11	10000	0.8
0	B	20,021	100	0.1
1	C	1,123	10000	0.7
0	D	300,123	10	0.2
…	…	…	…	…

- Let’s see how deep learning can deal with network information without such statistics later …

Network Eigenvalue Centrality

- Not all neighbors are equal!
- A node nearer more important nodes is more important than a node near less important nodes
- Important for understanding Graph Neural Networks
- Let's focus on a specific variant of Eigenvalue centrality – PageRank
  - Google started with this!

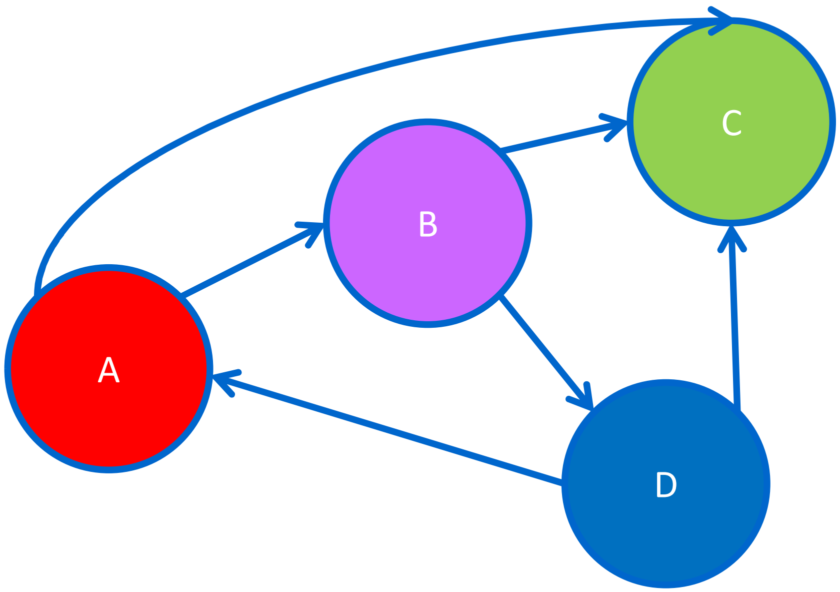
PageRank

- Assign all nodes an equal value – 1/n
- Update:
  - Each node divides this value equally across number of out-going edges, and passes these equal shares to the nodes it points to
  - If no out-going edges, passes to itself (self-loop)
- Do for k steps

PageRank

Initial value is 1/x?

What is node A's PageRank after 1 step?



Many topics in network analysis

- **Node centrality**
- Community detection
- Homophily (Birds of a feather flock together)
- Signed networks
- Homogeneous vs. heterogeneous networks
- And others

Community Detection

- **Node-Centric Community**
  - Each node in a group satisfies certain properties
- **Group-Centric Community**
  - Consider the connections within a group as a whole. The group has to satisfy certain properties without zooming into node-level
- **Network-Centric Community**
  - Partition the whole network into several disjoint sets
- **Hierarchy-Centric Community**
  - Construct a hierarchical structure of communities

Community Detection

Communities

Functions for computing and measuring community structure.

The functions in this class are not imported into the top-level `networkx` namespace. You can access these functions by importing the `networkx.algorithms.community` module, then accessing the functions as attributes of `community`. For example:

```
>>> from networkx.algorithms import community
>>> G = nx.barbell_graph(5, 1)
>>> communities_generator = community.girvan_newman(G)
>>> top_level_communities = next(communities_generator)
>>> next_level_communities = next(communities_generator)
>>> sorted(map(sorted, next_level_communities))
[[0, 1, 2, 3, 4], [5], [6, 7, 8, 9, 10]]
```

Bipartitions

Functions for computing the Kernighan–Lin bipartition algorithm.

`kernighan_lin_bisection`(G[, partition, ...])	Partition a graph into two blocks using the Kernighan–Lin algorithm.

K-Clique

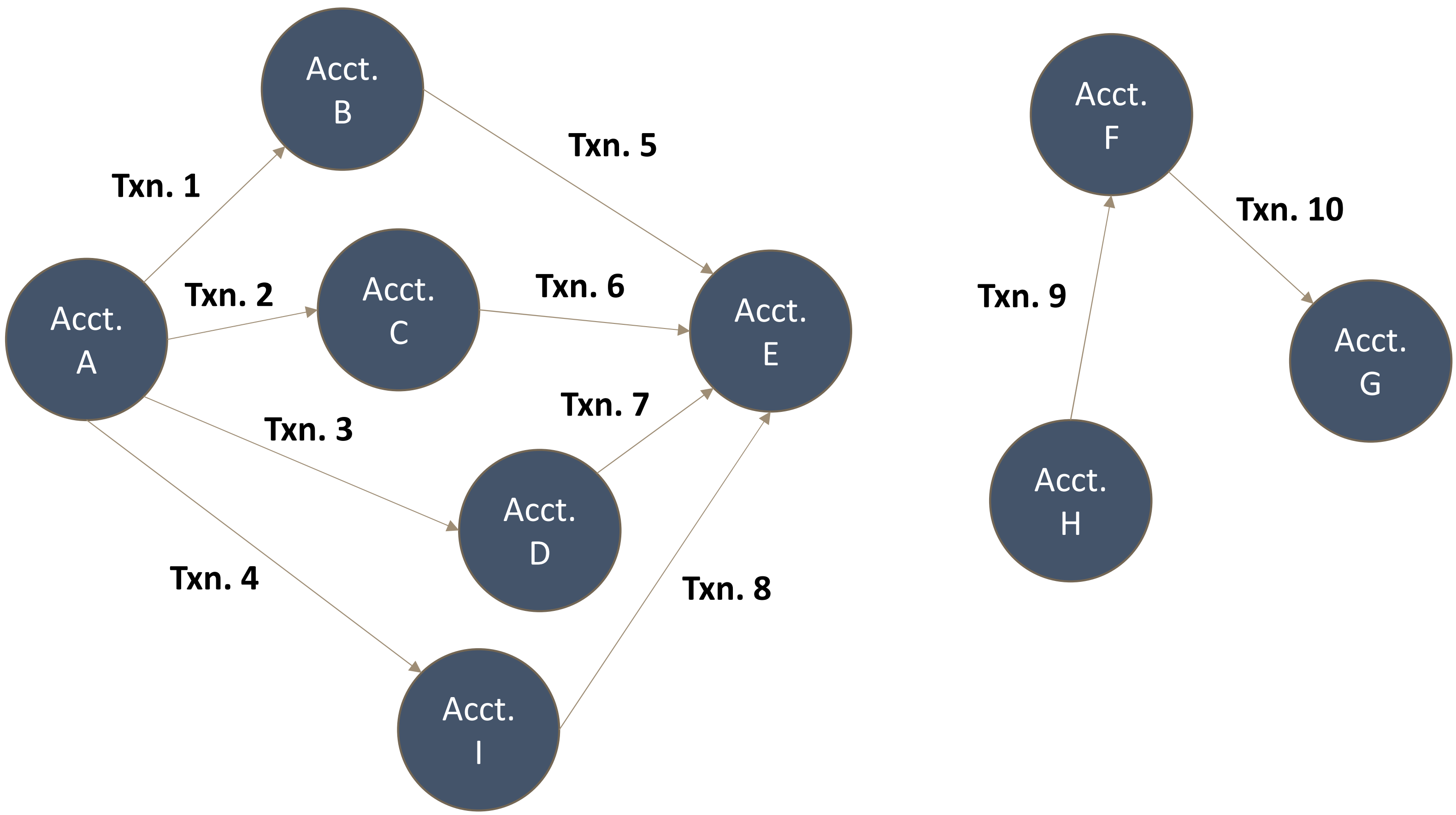
`k_clique_communities`(G, k[, cliques])	Find k-clique communities in graph using the percolation method.

Modularity-based communities

Let's go back to our tabular dataset

SAR	kycRiskScore	income	tenureMonths	creditScore	state	nbrPurchases90d	avgTxnSize90d	totalSpend90d	nbrDistinctMerch90d
0	3	110300	5	757	PA	10	153.8	1538	7
0	2	107800	6	715	NY	22	1.59	34.98	11
0	1	74000	13	751	MA	7	57.64	403.48	4
0	0	57700	1	659	NJ	14	29.52	413.28	7
0	1	59800	3	709	PA	54	115.77	6251.58	16
0	1	43500	11	717	CT	18	36.11	649.98	11
0	0	70200	9	720	ME	17	55.38	941.46	7
1	1	5900	1	772	MA	0	36.88	0	0
0	1	11400	43	727	NY	2	159.05	318.1	1
0	1	36700	12	735	PA	86	37.25	3203.5	41
0	0	43700	4	660	CT	19	6.49	123.31	14

How to represent network data in a table?



Isn't the adj. matrix a table?

- Calculating the maximum possible number of edges for a graph with n nodes. How?
- Each node is connected to $n-1$ edges -> $n \times (n-1)$
- Every edge counted in this way connects two nodes
- **So total number of edges is?**

- **Issue: Do you think most graphs or networks in the real world are that dense?**

Network Statistics

- Use key network structures as input features
- Represented as network statistics
- Many, many possible statistics, as we have seen
  - Node degree centrality based on different centrality measures
  - Node community feature based on community detection

Network Analytics Libraries

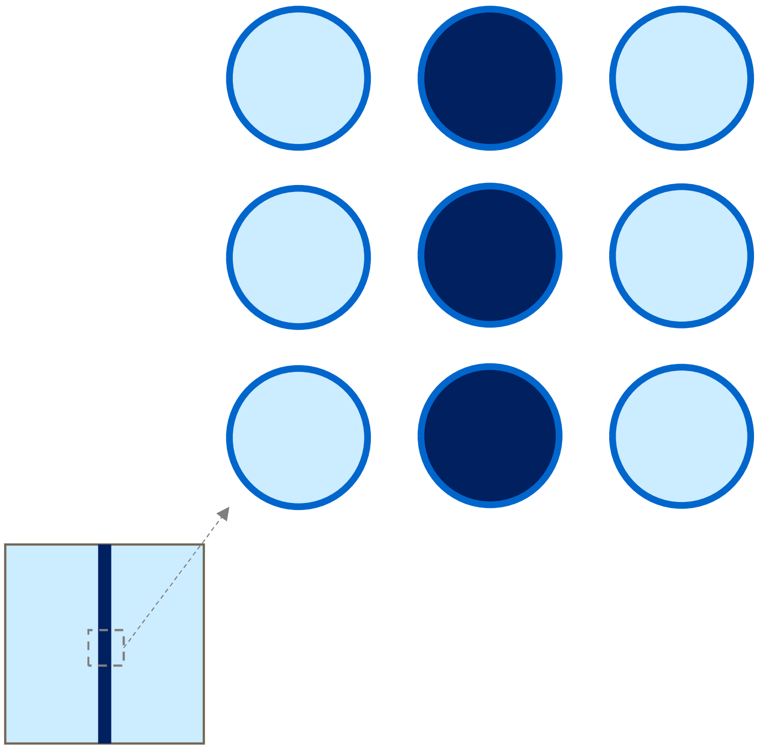
- **NetworkX - https://networkx.github.io/**
  - Popular, easy to use and comprehensive but slow
- **iGraph - https://igraph.org/**
  - Not too bad, but less comprehensive
- **SNAP - https://snap.stanford.edu/snap/quick.html**
  - Not as easy to use
- **cuGraph - https://docs.rapids.ai/api/cugraph/stable/basics/cugraph_intro.html**
  - Super-fast, but need Linux environment and GPU

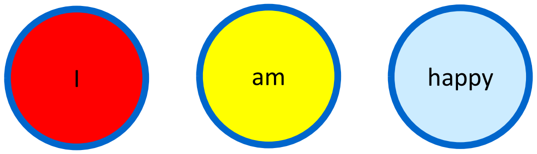
Can we just use the networks directly?

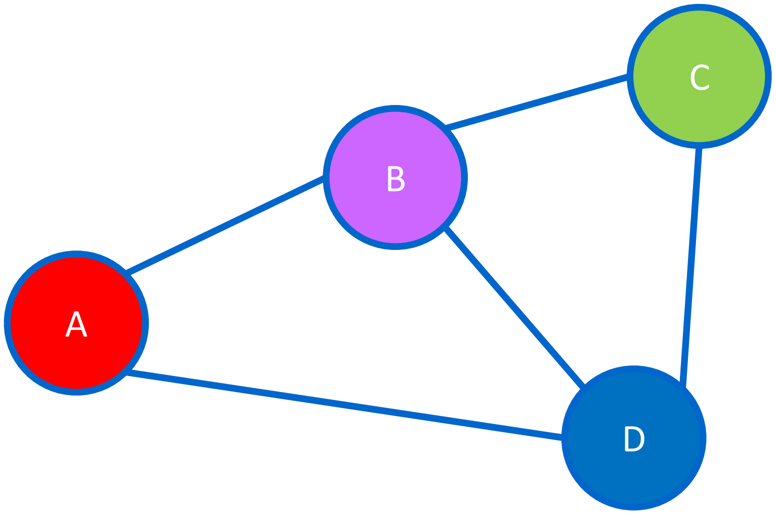
Regular (Euclidean) vs. Irregular (Non-Euclidean)

Neural Networks for Graphs

- What is the key difference when it comes to graphs vs. images or text

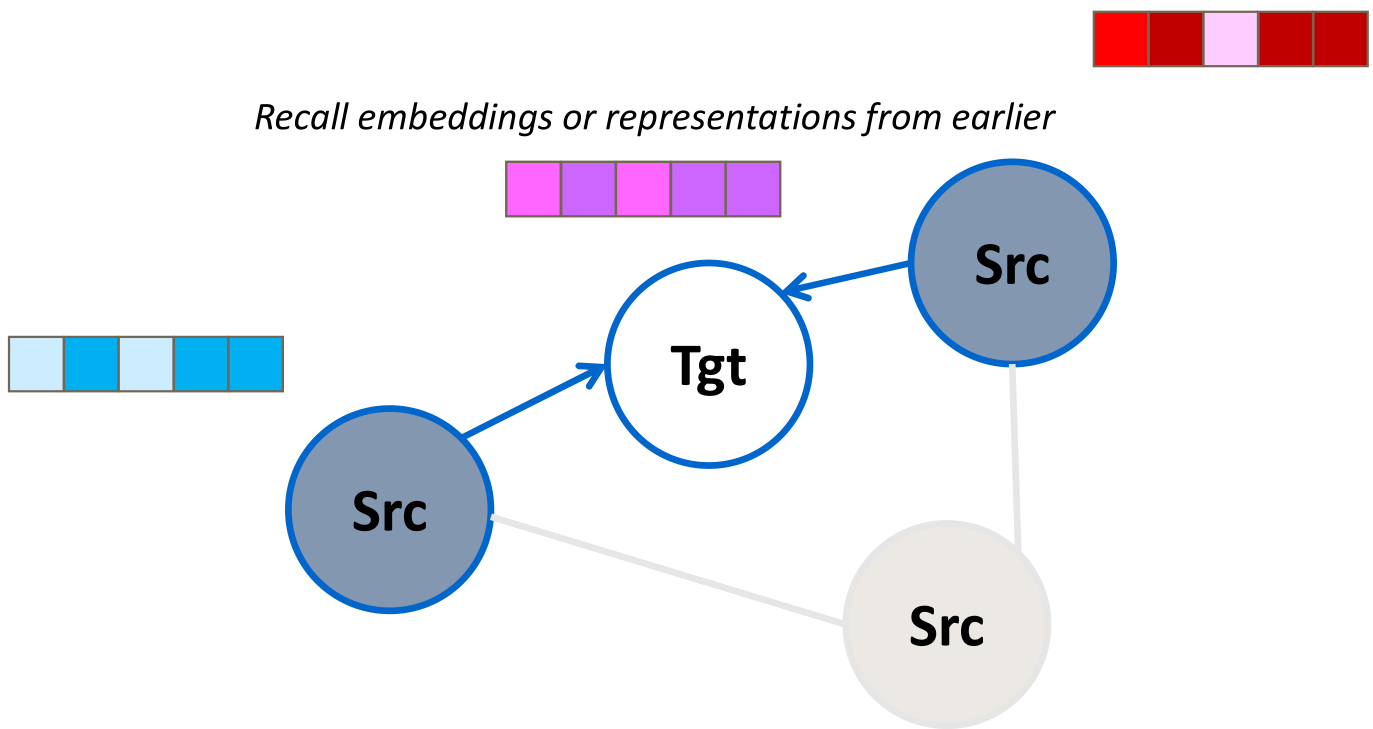






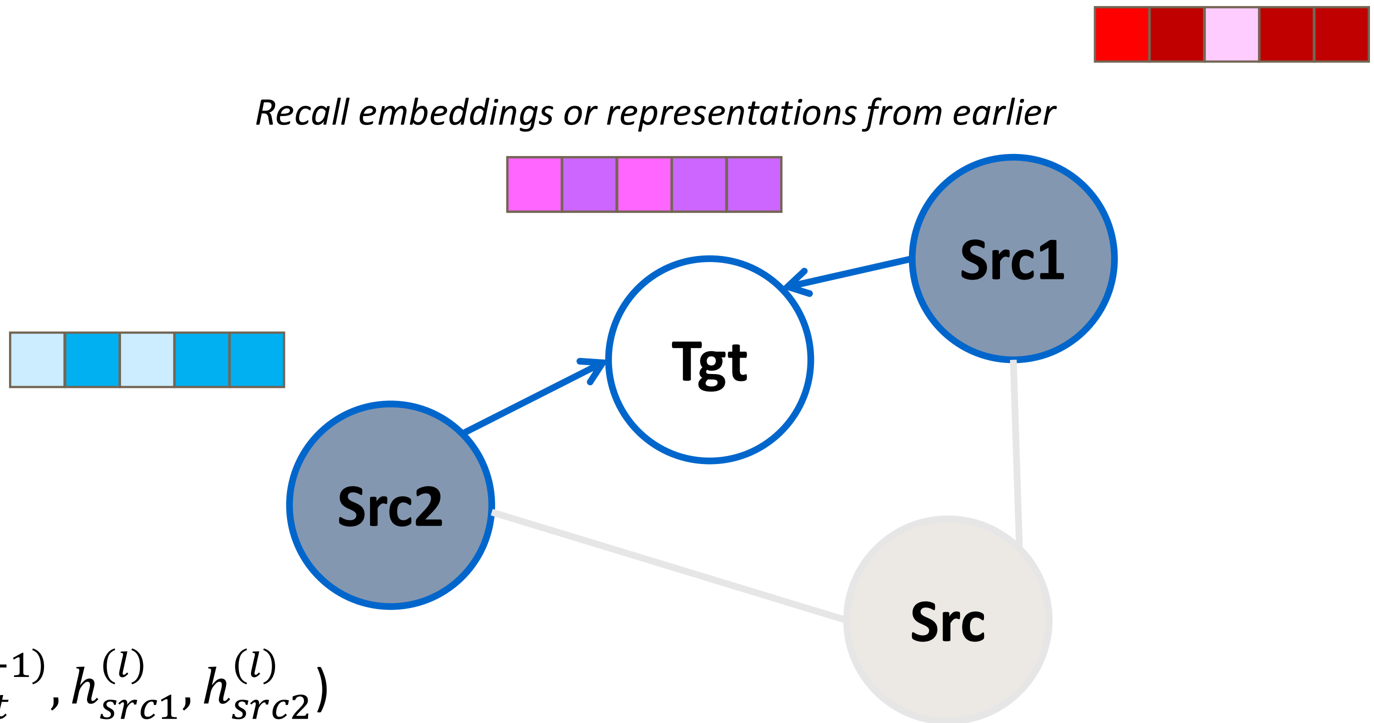
Basic Idea of a Graph Neural Network

Recall embeddings or representations from earlier



Basic Idea of a Graph Neural Network

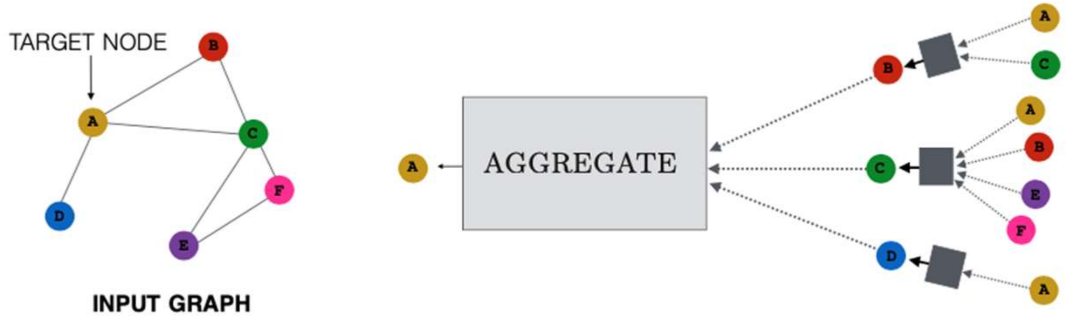
- A graph neural network (GNN) is basically learning a function f that generates the embedding of a node based on its neighbours and edges (and only its neighbours and edges)



$$h_{tgt}^{l} = f(h_{tgt}^{(l-1)}, h_{src1}^{(l)}, h_{src2}^{(l)})$$

Basic Idea of a Graph Neural Network

- Graph Convolutional Networks (GCN) (Kipf and Welling, 2016), GraphSAGE (Hamilton et al., 2017), Graph Attention Networks (Velickovic et al., 2018) (& many others)
- Such models capture network structures – k-layers capture k-hops



$$h_u^{(k+1)} = \underbrace{UPDATE^{(k)}}_{\text{Updating state of target node}} \left(h_u^{(k)}, \underbrace{AGGREGATE^{(k)}\left(\left\{h_v^{(k)}, \forall v \in N(u)\right\}\right)}_{\text{Composing messages from neighbors}}\right)$$

$m_{N(u)}^{(k)}$

Fig. from Gilmer et al., Neural Message Passing for Quantum Chemistry, PMLR 2017

Resources

- **A good introduction - https://distill.pub/2021/gnn-intro/**
- **Deep Graph Library (DGL) - https://docs.dgl.ai/**
- **PyTorch Geometric (PyG) - https://pytorch-geometric.readthedocs.io/en/latest/**
  - Both DGL and PyG are pretty good, but require comfort with deep learning
- **StellarGraph - https://stellargraph.readthedocs.io/**
  - Less commonly used

- We will go through DGL in the exercises later

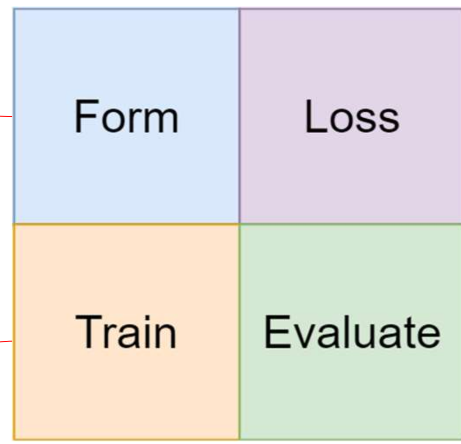
Recap

- Review of concepts
- Characteristics of tabular and network data
- From trees and forests to XGBoost
- From machine learning to deep learning
- From network analysis to Graph Neural Networks
- **From supervised to unsupervised learning**

Framework: Decision Tree

What is the form?

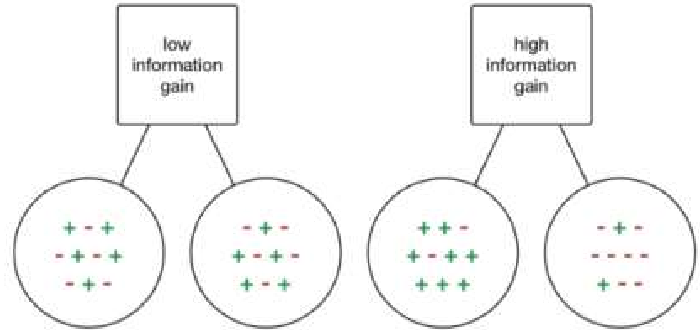
What would be a good criteria to split?



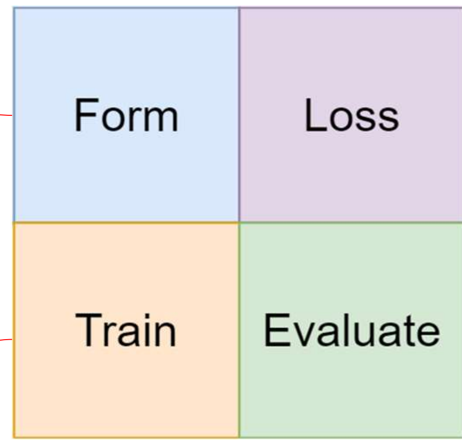
Given the splitting criteria, how could one build the tree?

Recall what we used for logistic regression

Framework: Decision Tree



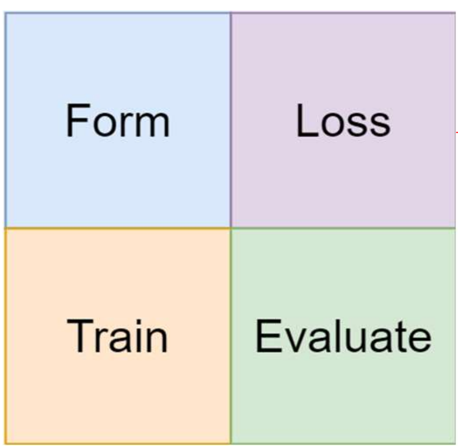
Binary Tree

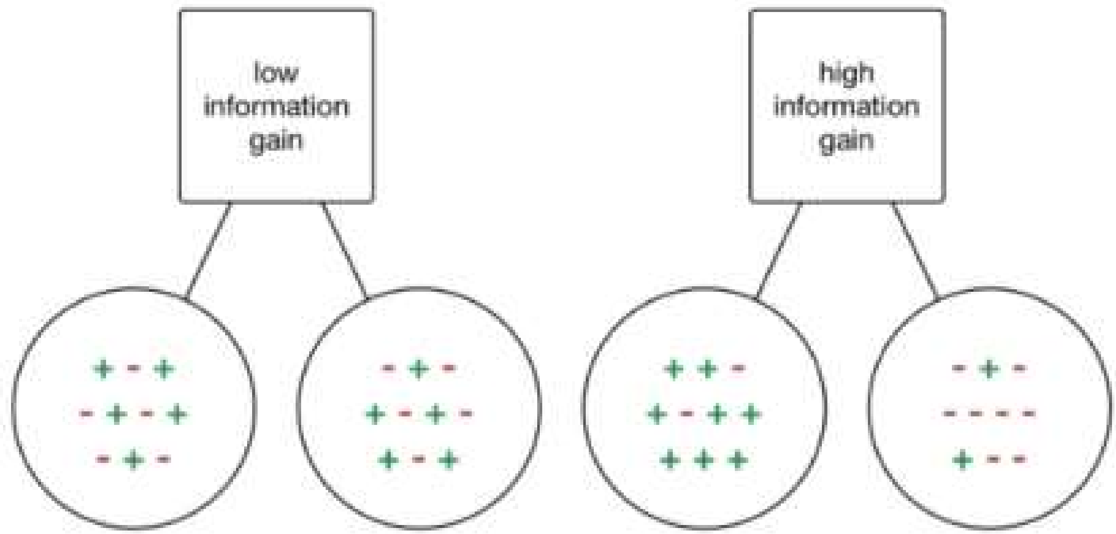


Greedy

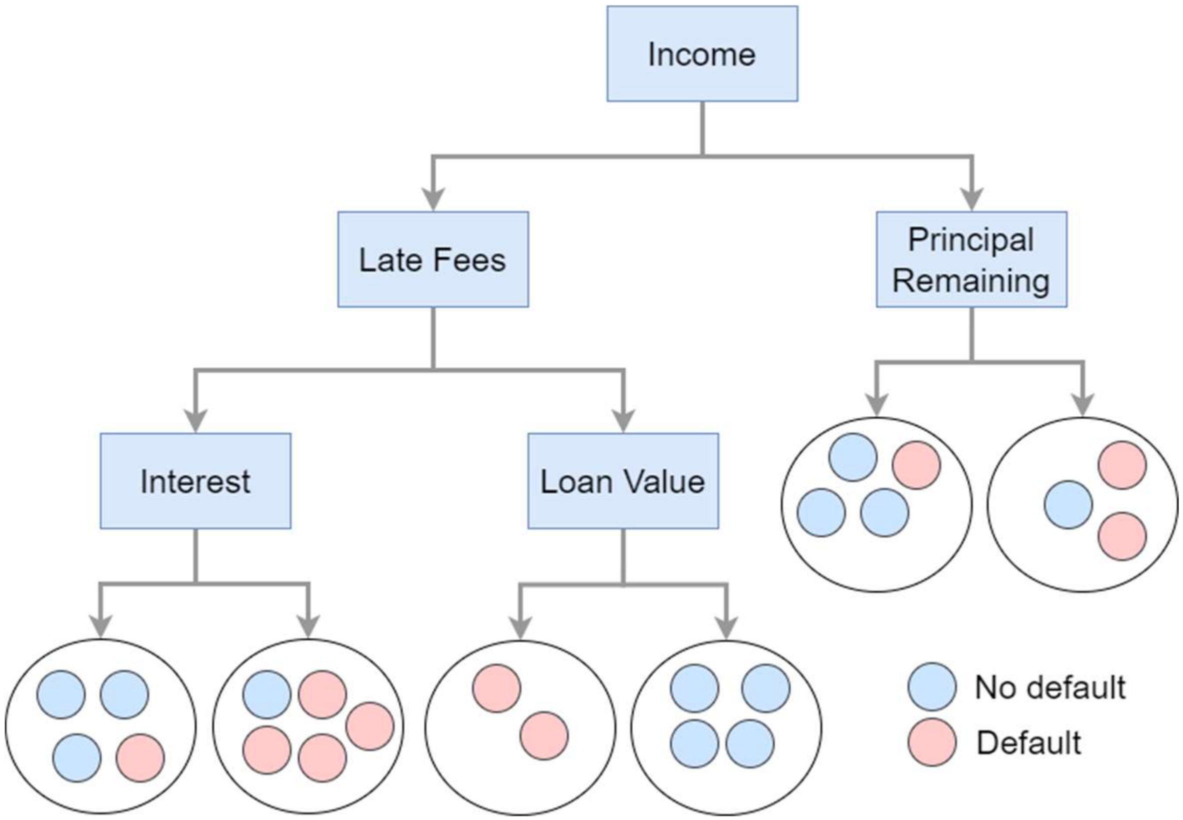
Accuracy, Recall, Precision, F1

Framework: Decision Tree

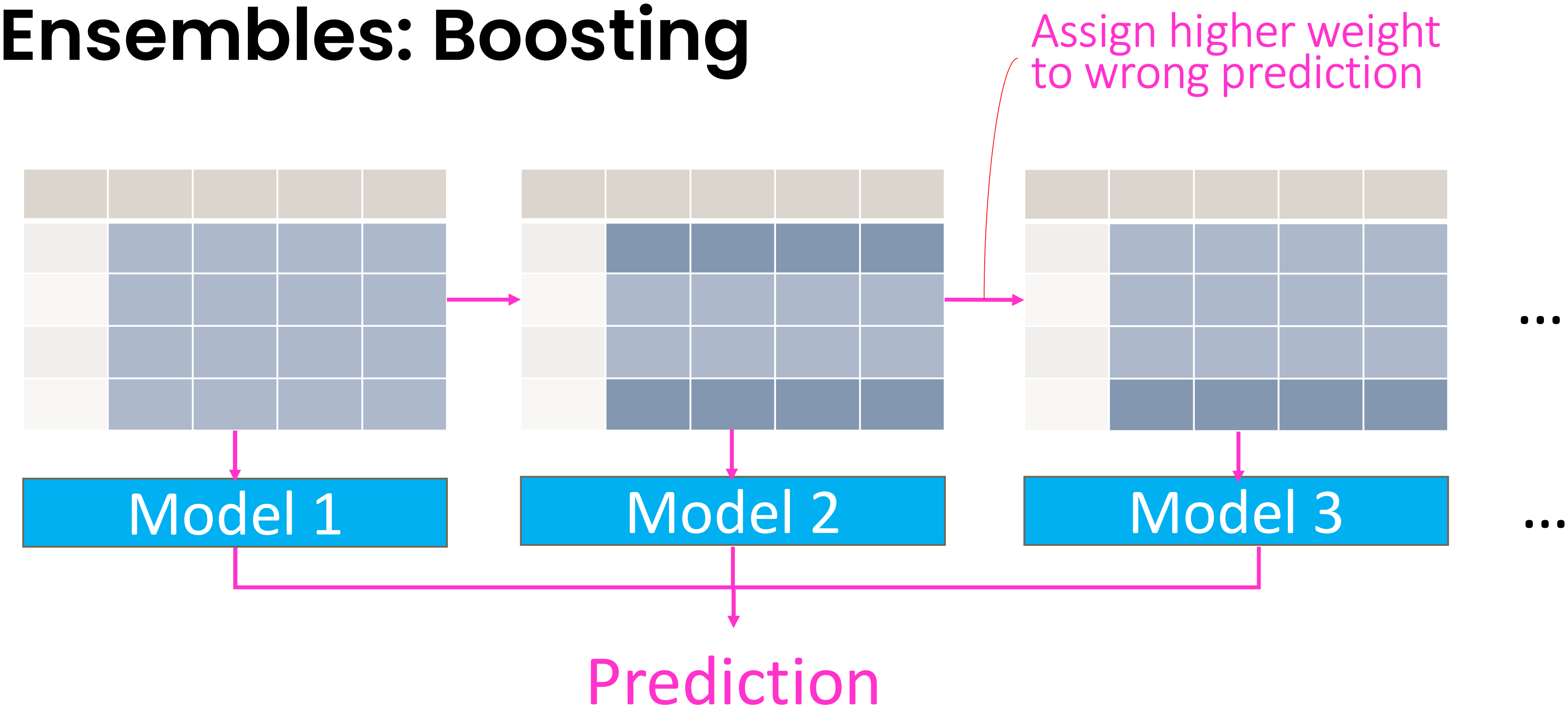




Decision Tree



Ensembles: Boosting



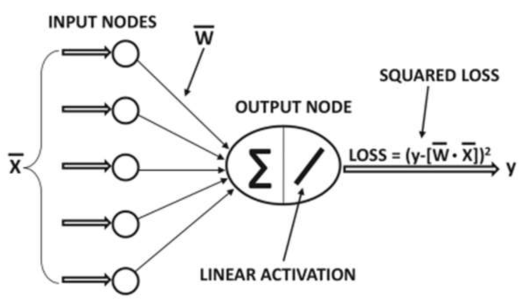
XGBoost

XGBoost is basically based on the idea of boosting, but with some additional math and optimization

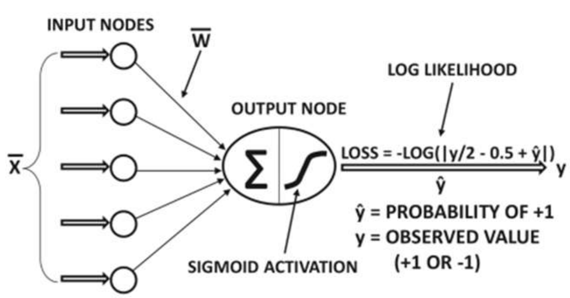
For the curious, more details available at https://xgboost.readthedocs.io/en/stable/tutorials/model.html

Framework: Neural Networks

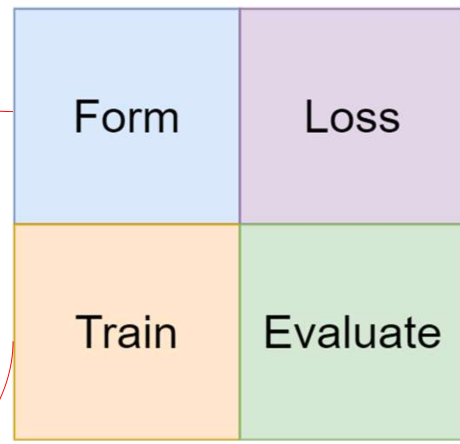
Linear Regression



Logistic Regression



Cross-Entropy Loss, Squared Error Loss

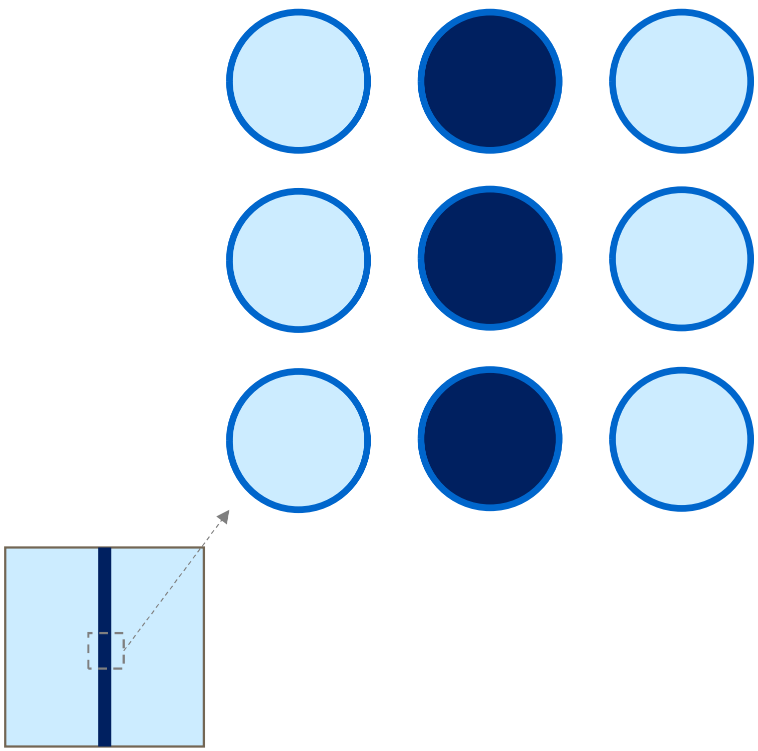


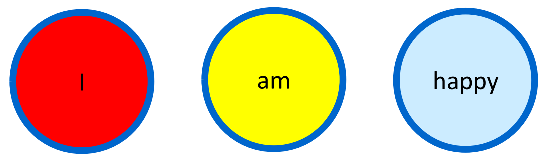
Gradient descent

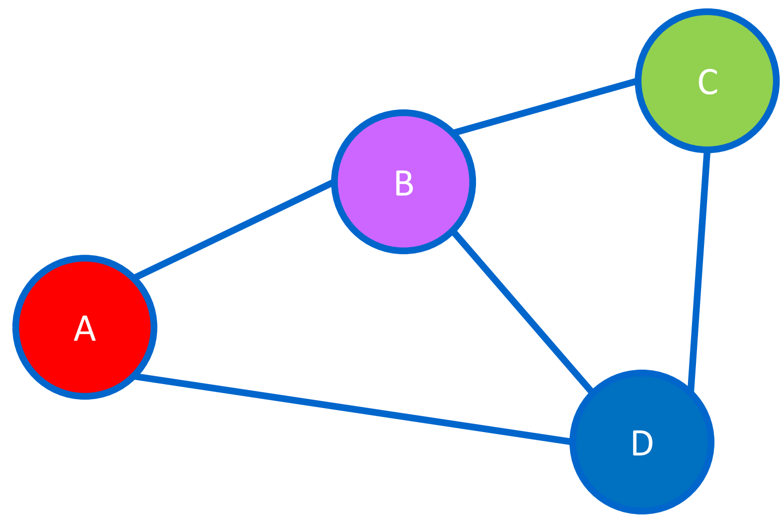
Accuracy, Recall, Precision, F1, Root Mean Squared Error, Mean Abs. Error, Mean Abs. Percentage Error

Neural Networks for Graphs

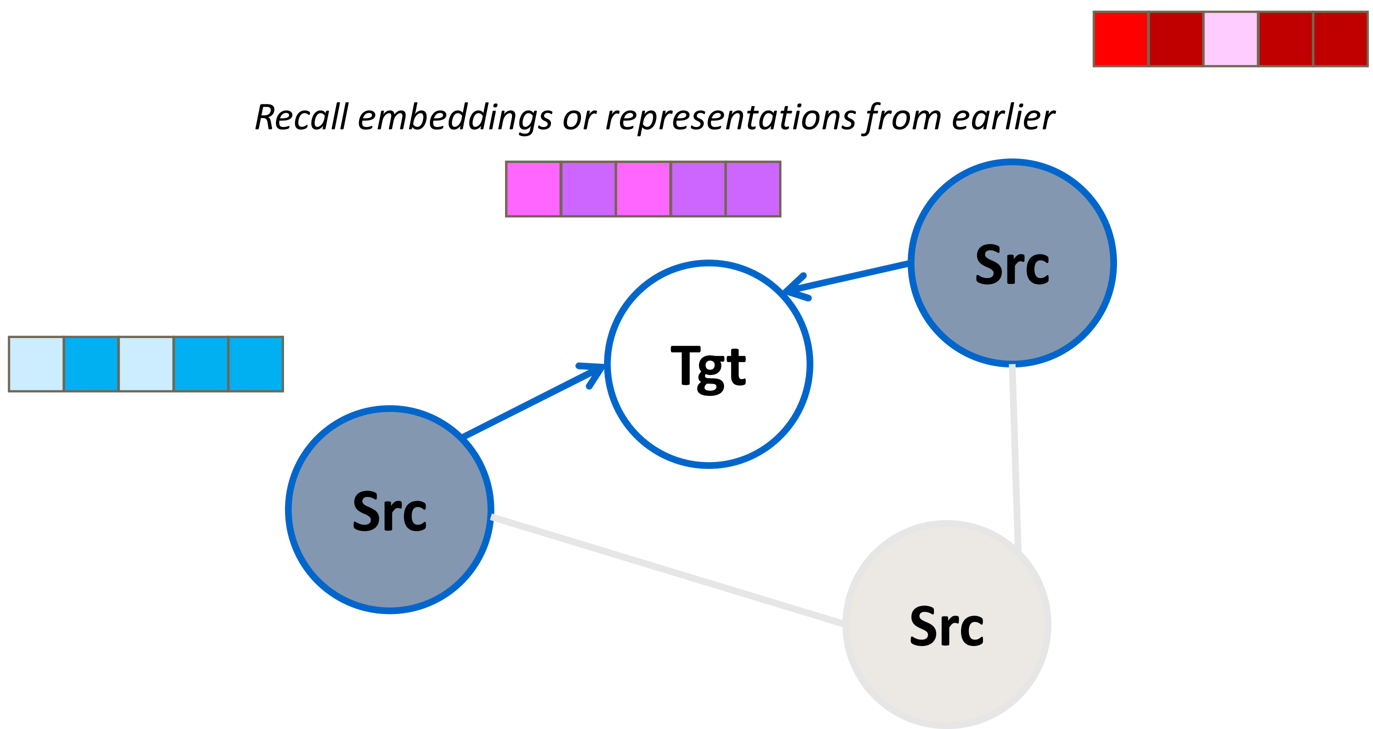
- What is the key difference when it comes to graphs vs. images or text







Basic Idea of a Graph Neural Network



From supervised to unsupervised learning

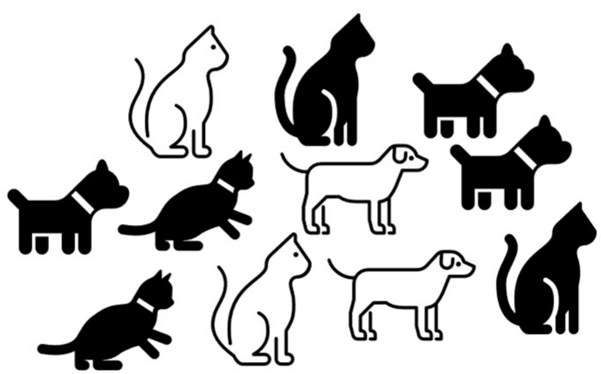
Unsupervised learning

$$Y = AX + B$$

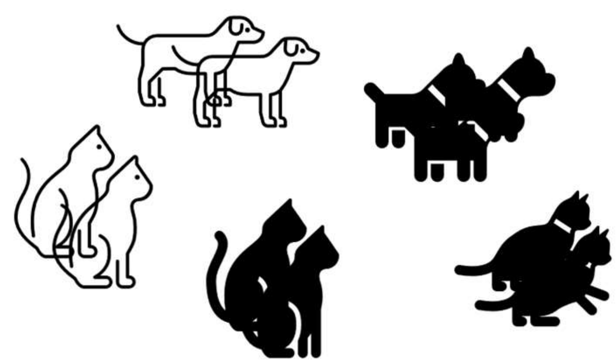
Input → Model → Output

X A,B Y

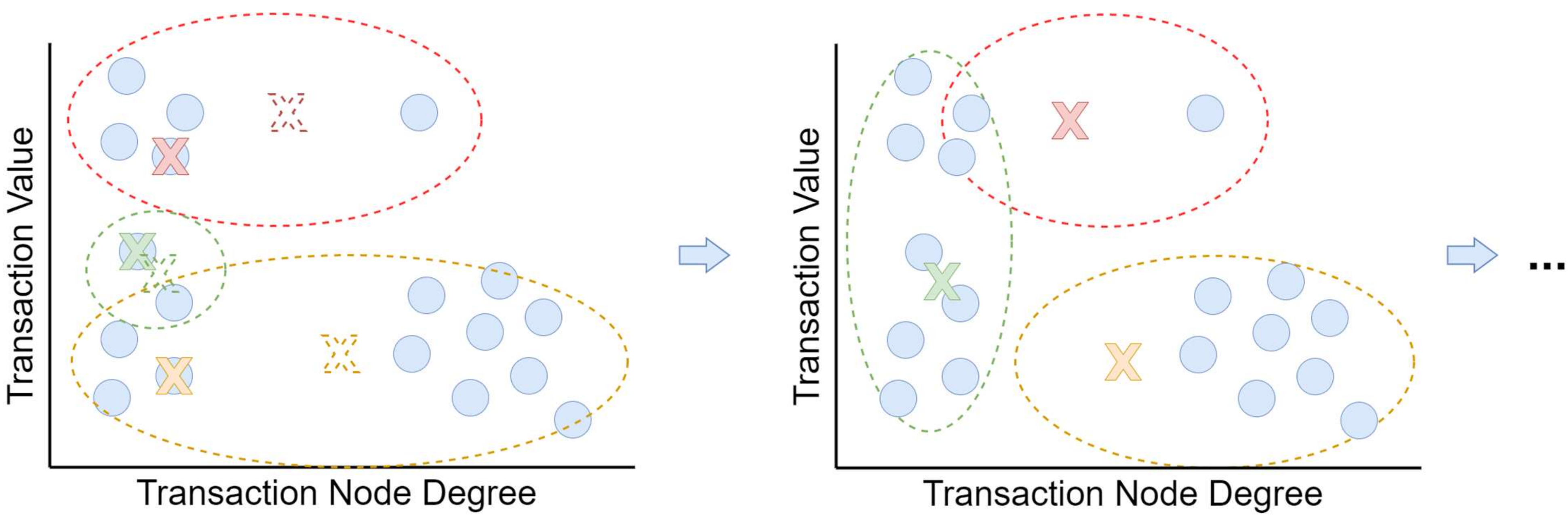
Data



Groupings

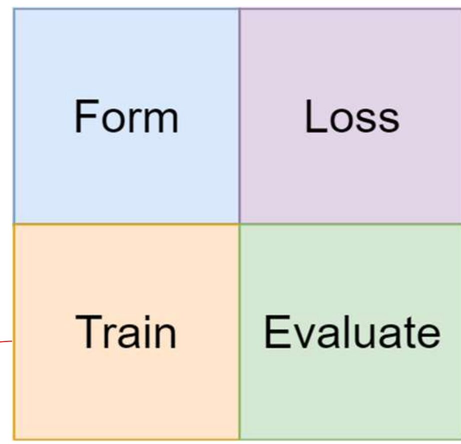


Clustering: K-Means



Framework: K-Means

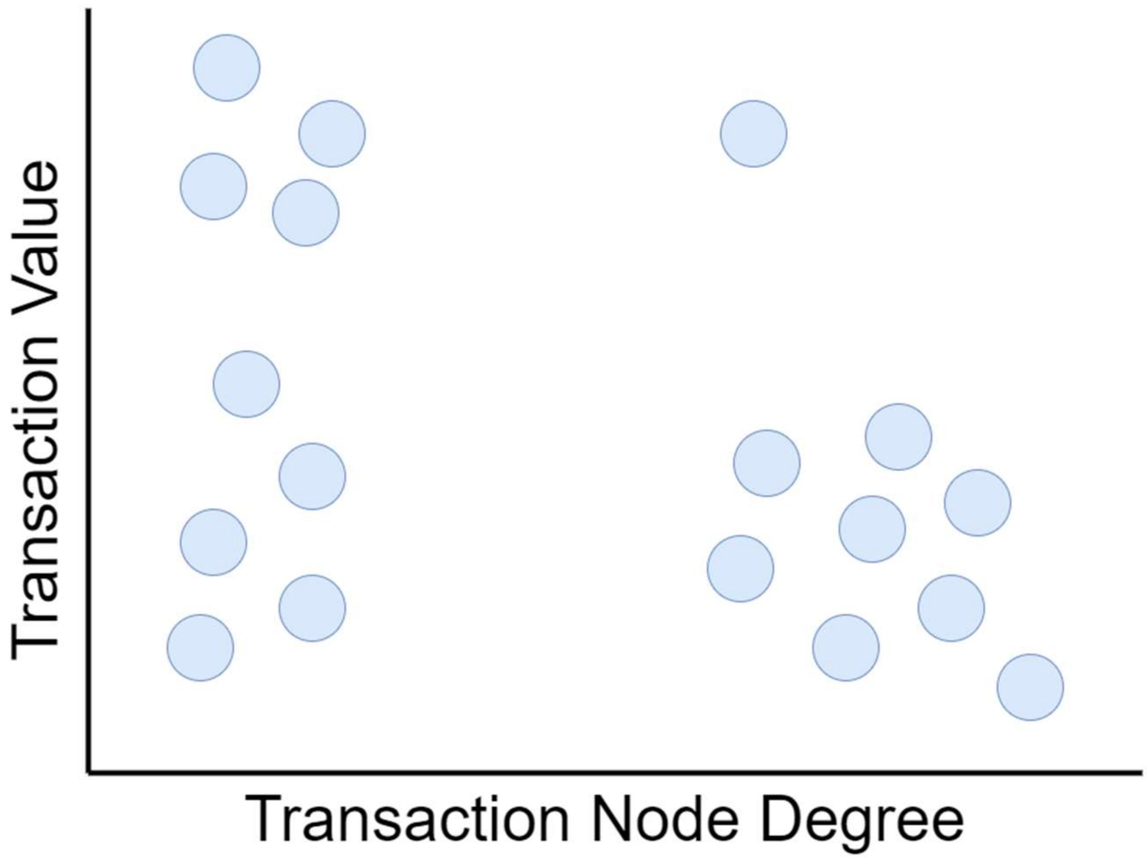
What is the loss?



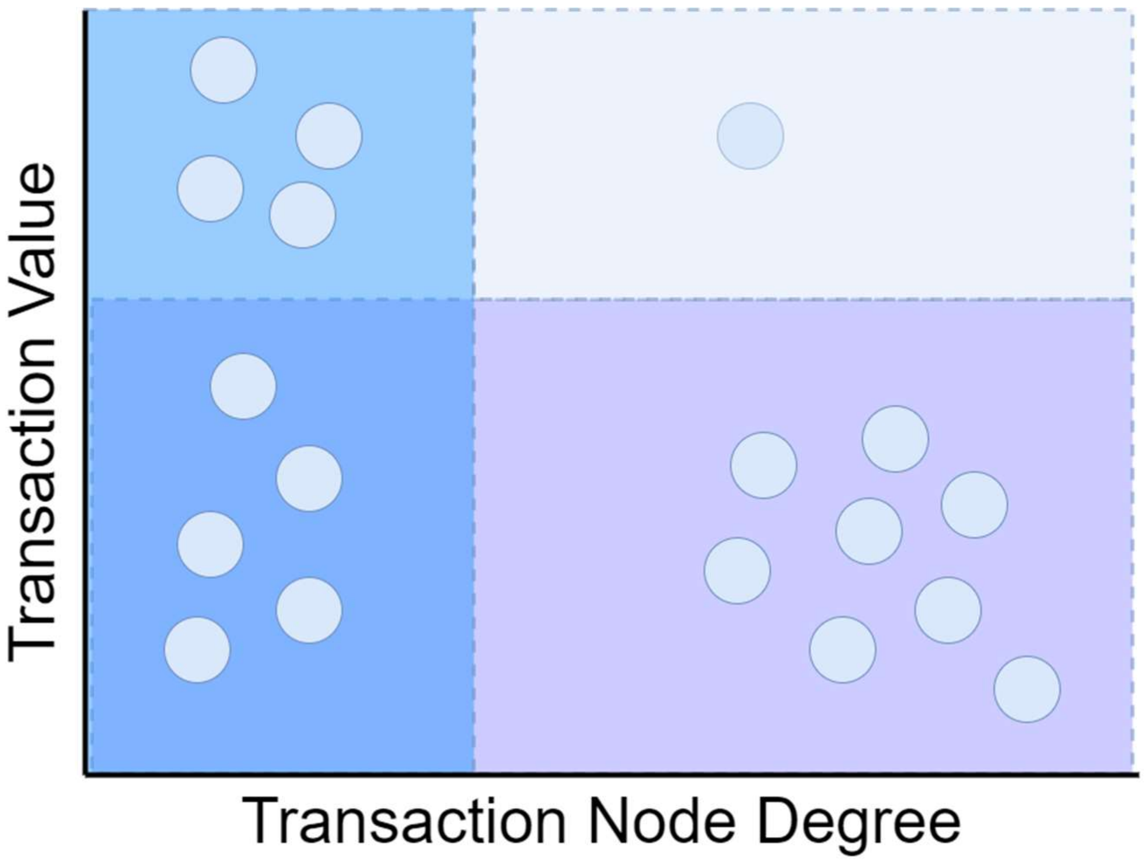
How do you train the model?

How do you evaluate?

Spot the anomaly

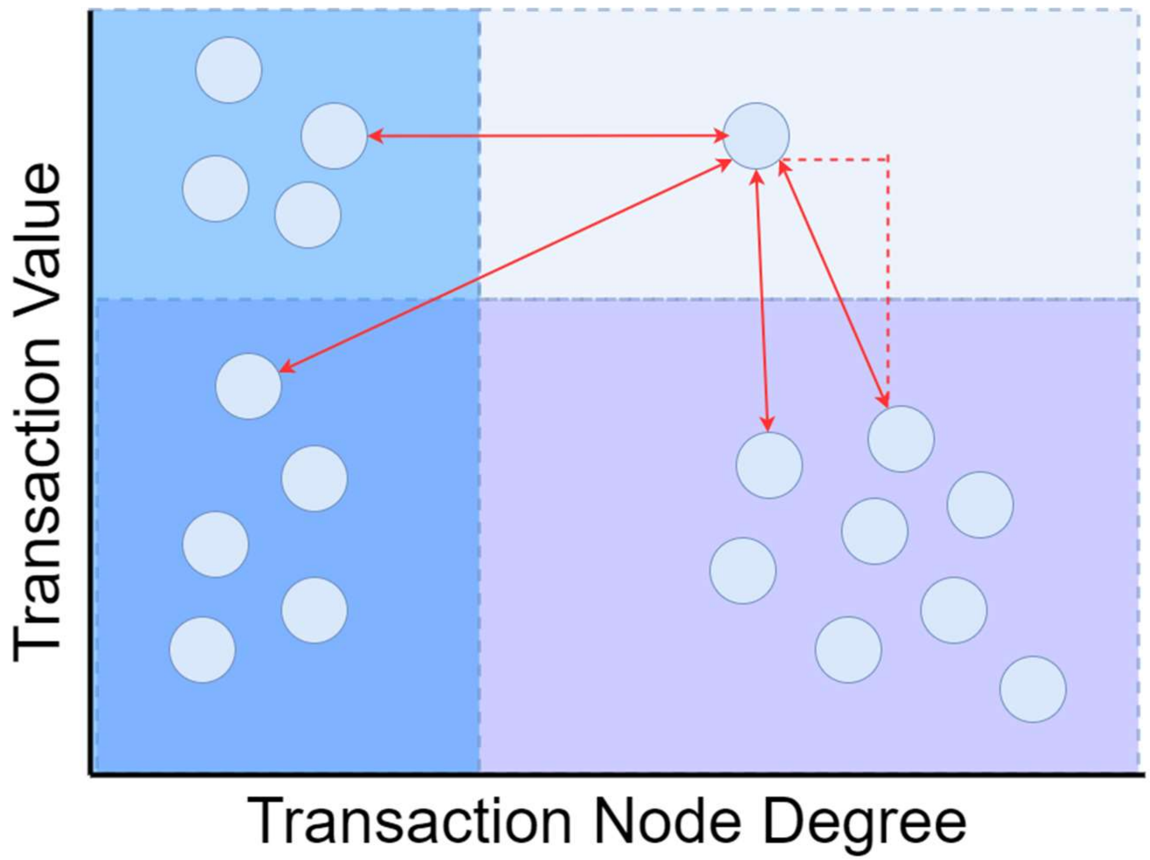


We need an objective measure



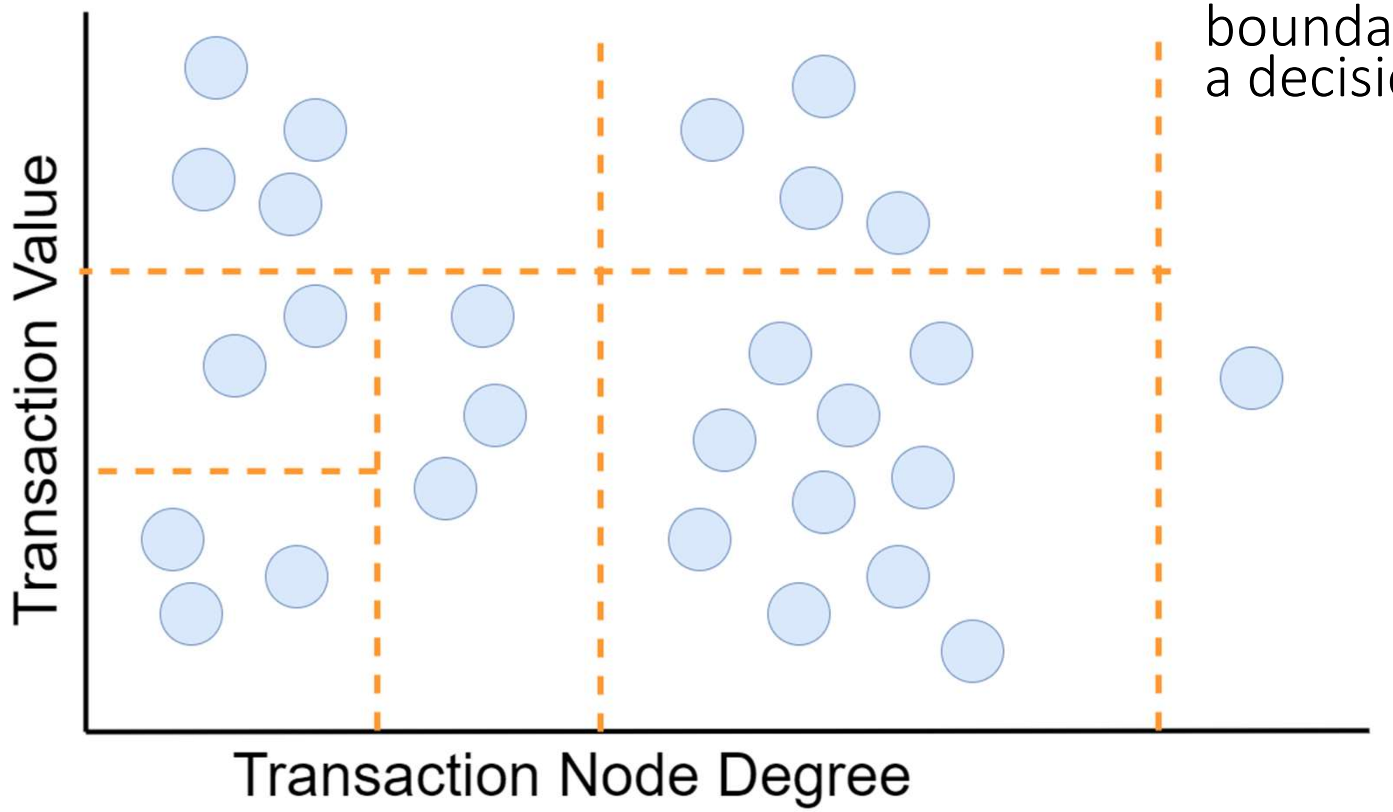
Recall K-means clustering

We need an objective measure

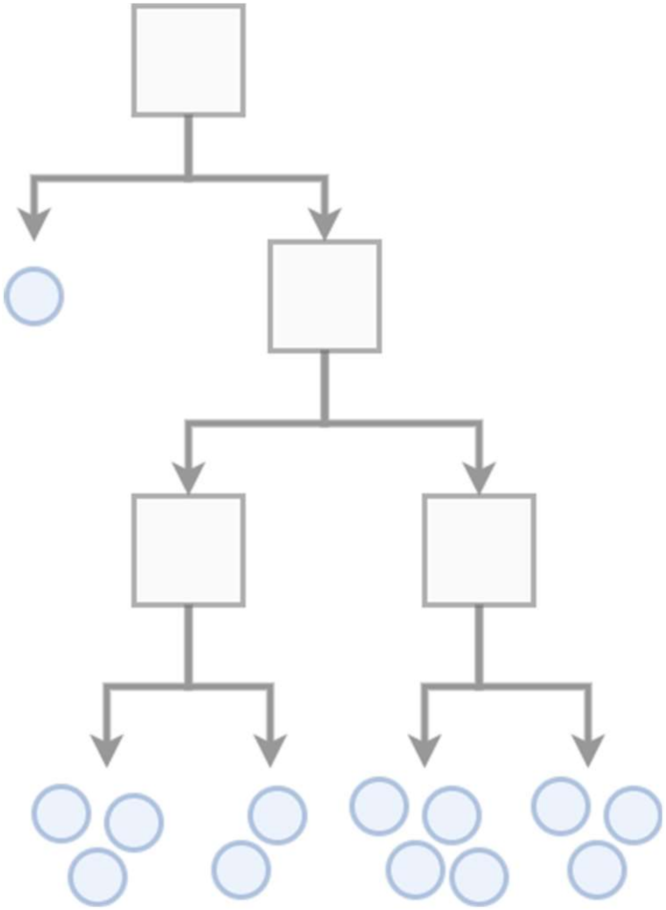


Isolation Forest

How would this decision boundary look in a decision tree?



Isolation Forest

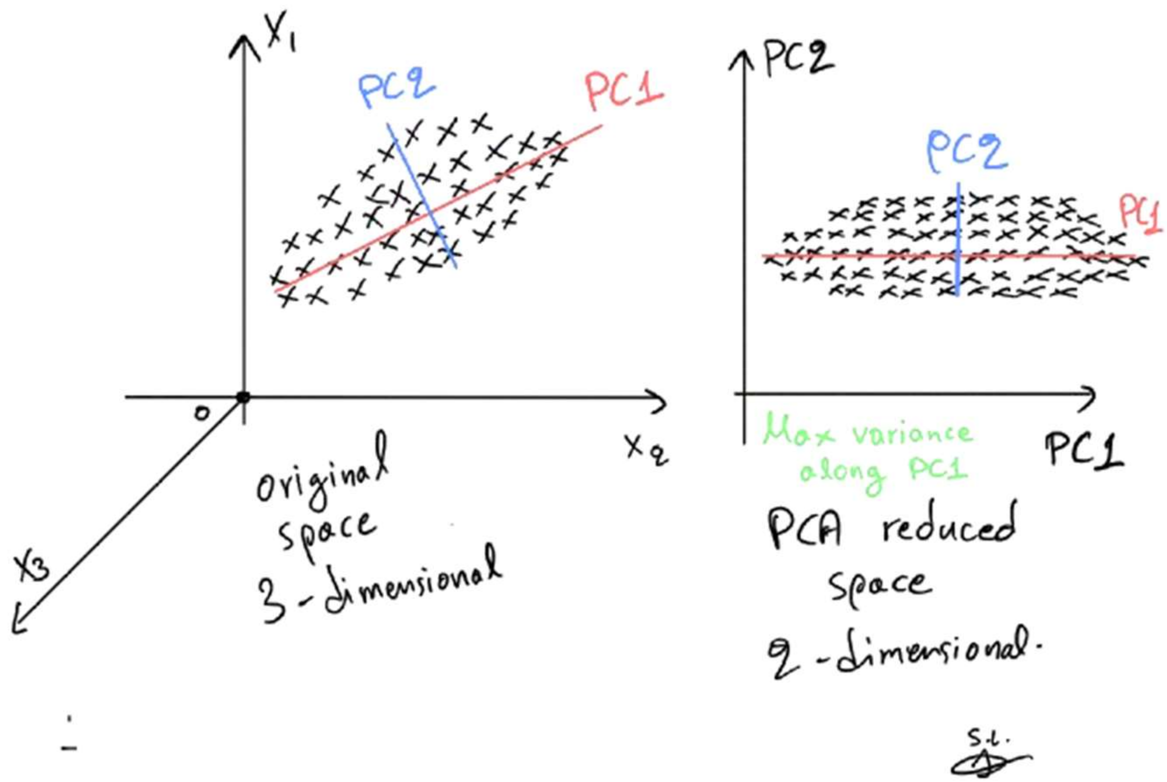


Which path leads to an anomalous instance?

What is an anomaly in AML?

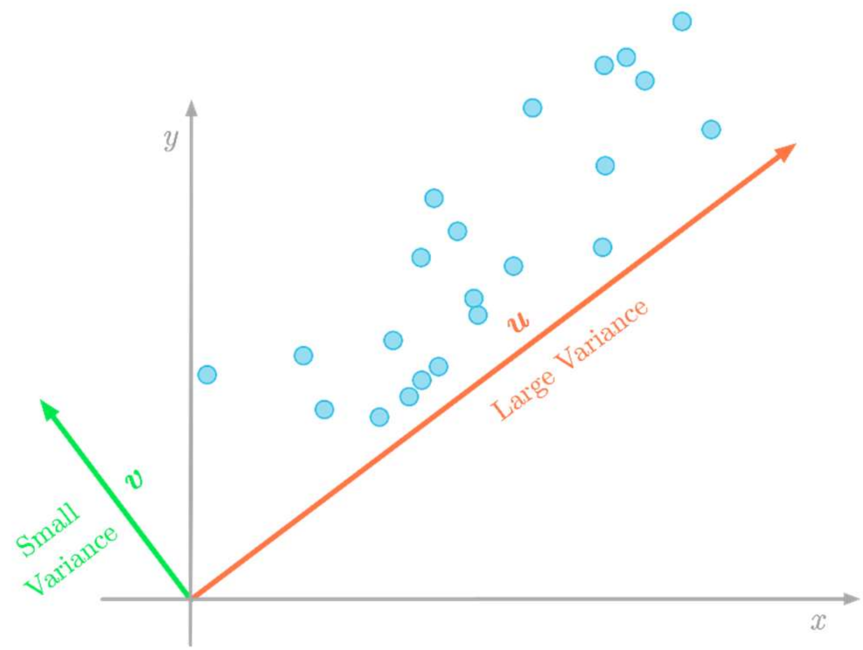
- Is it always clear?
- Does an anomaly stay still?
- Can you spot an anomaly with supervised learning?

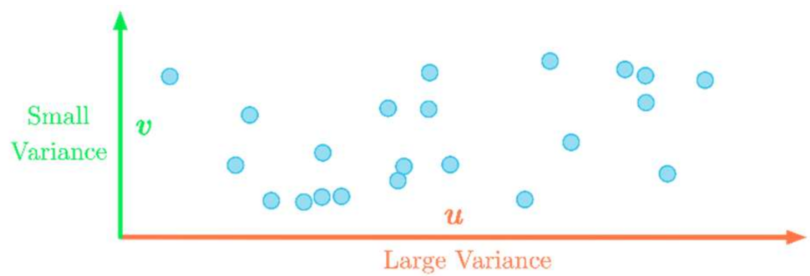
Dimensionality Reduction



From https://towardsdatascience.com/pca-clearly-explained-how-when-why-to-use-it-and-feature-importance-a-guide-in-python-7c274582c37e

Principal Components Analysis





Conclusion

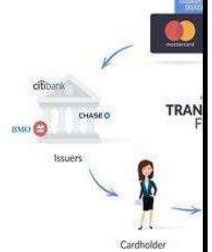
- Tabular and network data are two very different types of data
- XGBoost is essentially a combination of many trees that performs well on tabular data
- Deep learning enables us to learn important features
- GNNs enables us to learn important network features instead of using network statistics
- Even without labels, we can do a lot with unsupervised learning

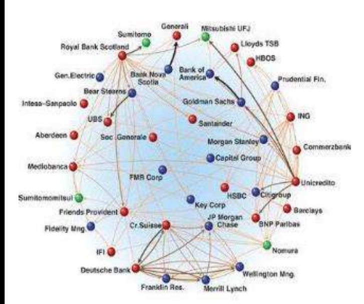
Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs - Apps

Fraud & Abuse

Detect malicious accounts, fraudulent financial transactions, fa… fraudulent insu…

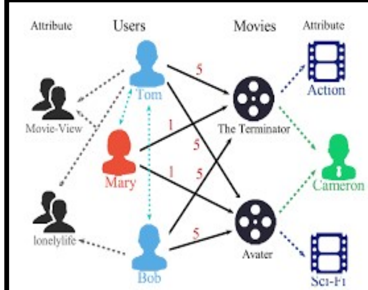




Financial transactions network

Recommendations

Products, media, articles, experiences, jobs, courses, spouses



Collaborative knowledge graph

Marketing

Who should get a discount? Who are the influencers? Who are the risk of churning?

Social network

Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

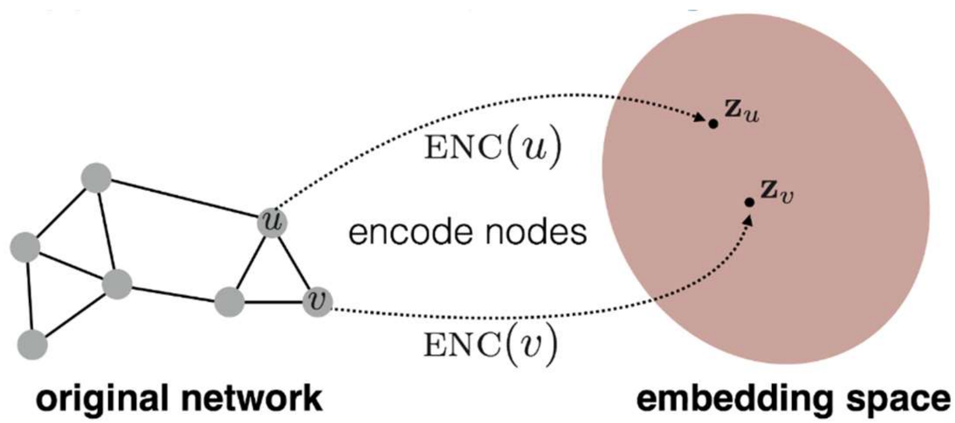
Quick Review of GNNs – Tasks

- Node classification
  - Detect malicious accounts
  - Target right customers
- Link prediction
  - Recommendations
  - Predict missing relations in a knowledge graph
- Graph classification
  - Predict the property of a chemical compound

Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs – Node Embeddings

- Embed nodes to a low-dimension space so that these embeddings capture the essential task-specific information and use them to train off-the-self classifiers.
  - For example, node similarities in the embedding space approximate similarities in the original graph.



Representation Learning on Networks, snap.stanford.edu/proj/embeddings-www, WWW 2018

Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs – Approaches

- Generate embeddings by manual feature engineering
  - Requires domain expertise and sig. effort
- Automatically generate embeddings using unsupervised dimensionality reduction approaches, e.g., PCA
  - Cannot do end-to-end learning
- GNNs help us address these disadvantages

Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

Graph neural networks are based on ***message-passing***

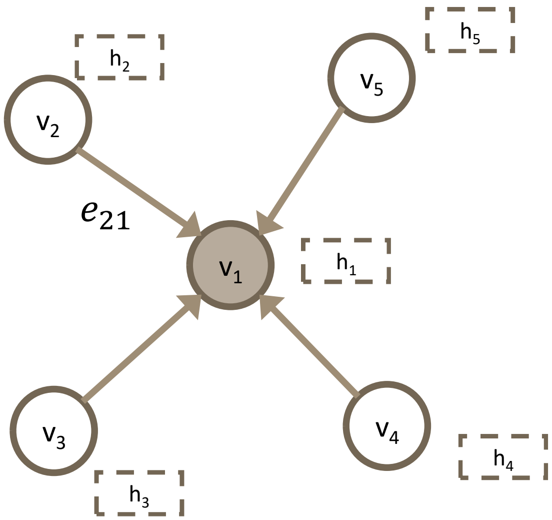
Reduce/Aggregate

Message

$$m_v^{(l)} = \boxed{\sum_{w \in N(v)}} \boxed{M^{(l)}(h_v^{(l-1)}, h_w^{(l-1)}, e_{vw})}$$

$$h_v^{(l)} = \boxed{U^{(l)}(h_v^{(l-1)}, m_v^{(l)})}$$

Update

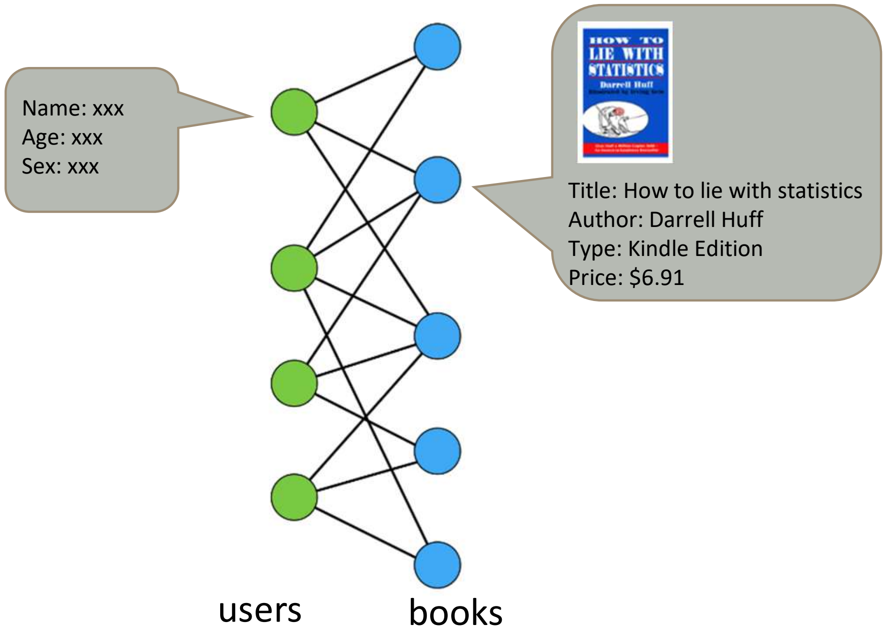


Neural Message Passing for Quantum Chemistry

Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

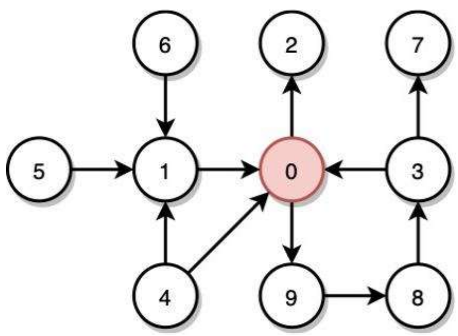
GNNs compute node embeddings using both the structure of the graph and the features of the nodes and edges.

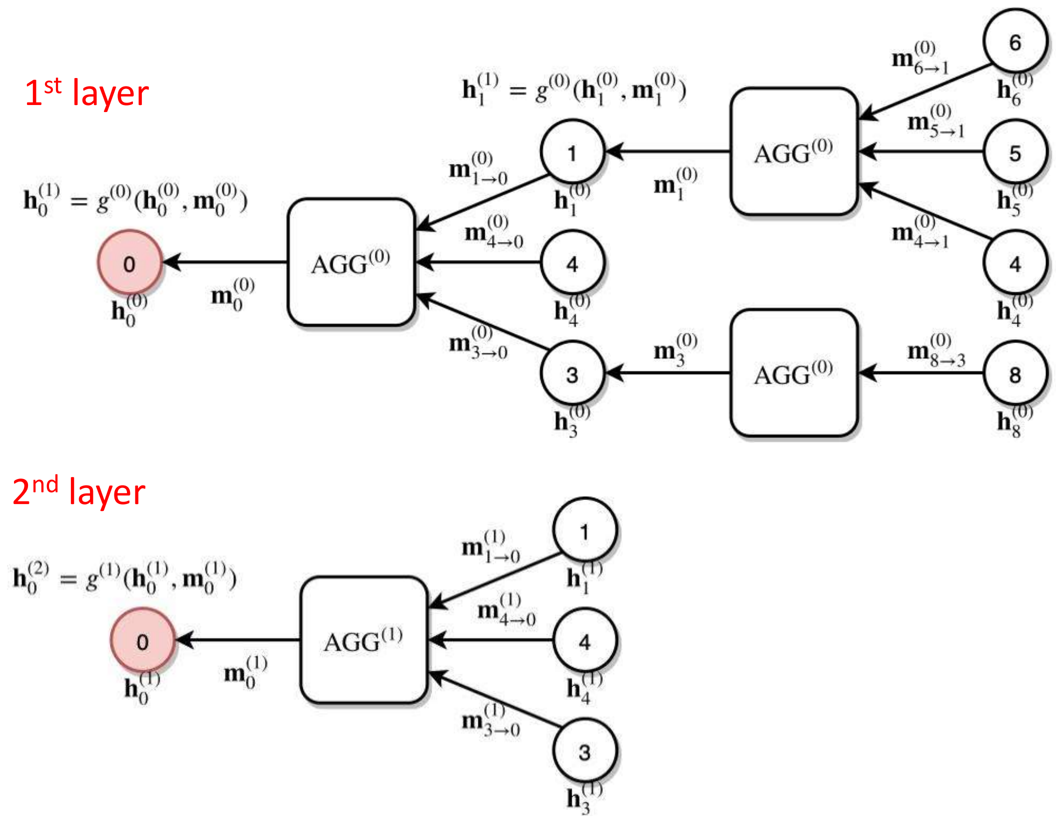


Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

Multiple GNN layers can be stacked together.

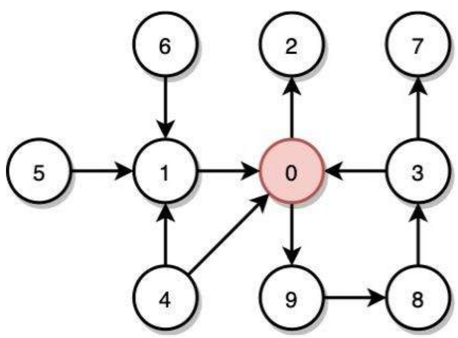


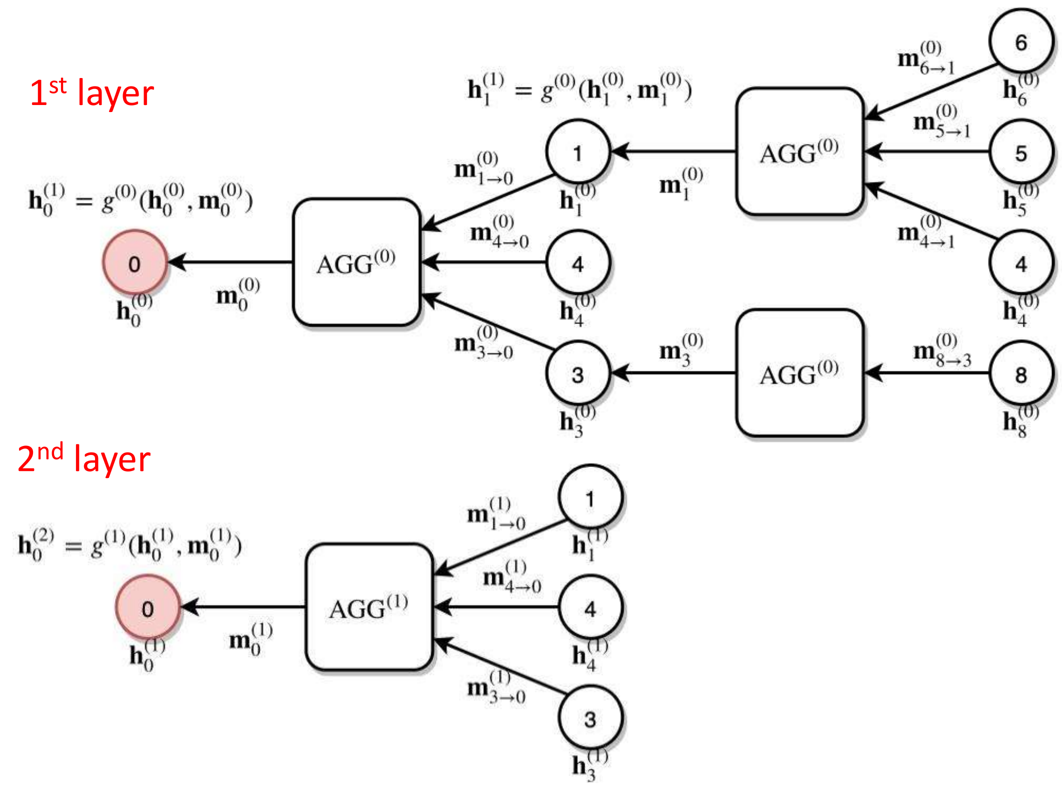


Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

GNNs can *capture* distant information in a non-linear fashion.

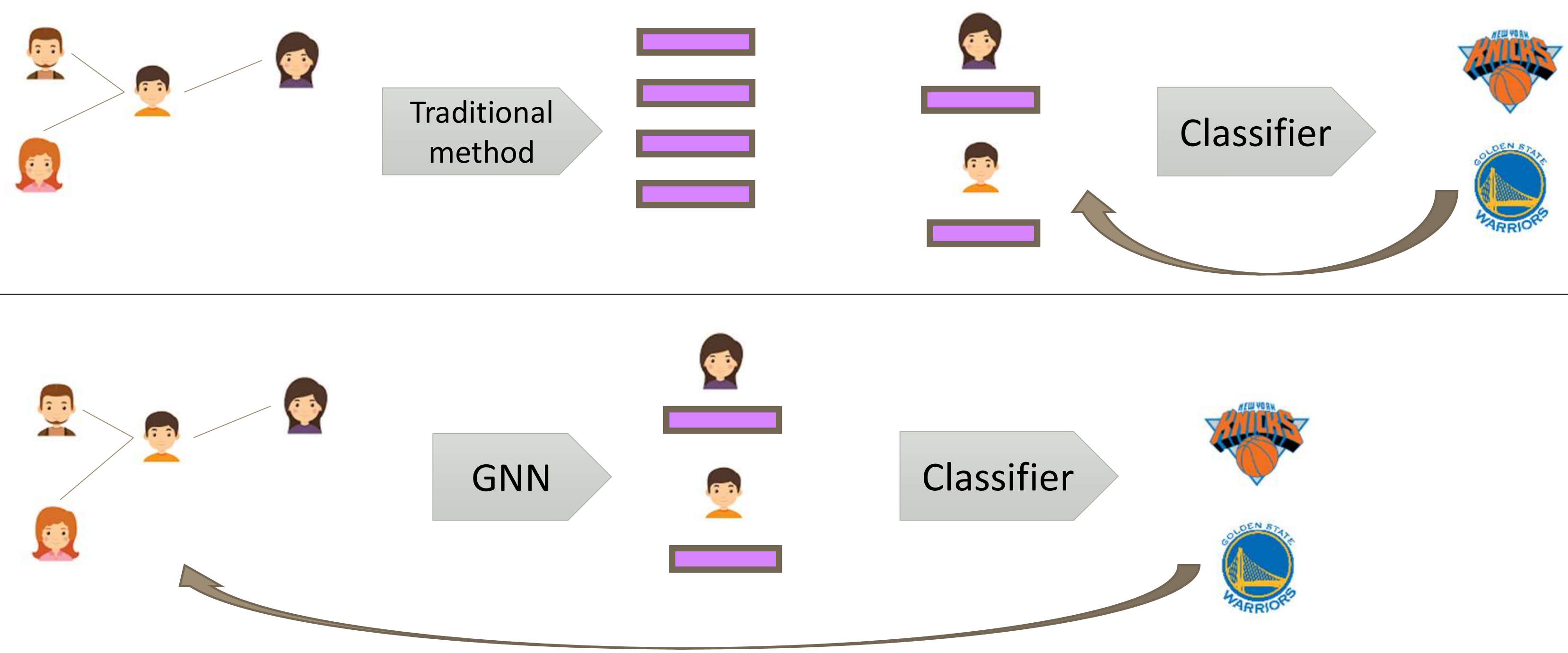




Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

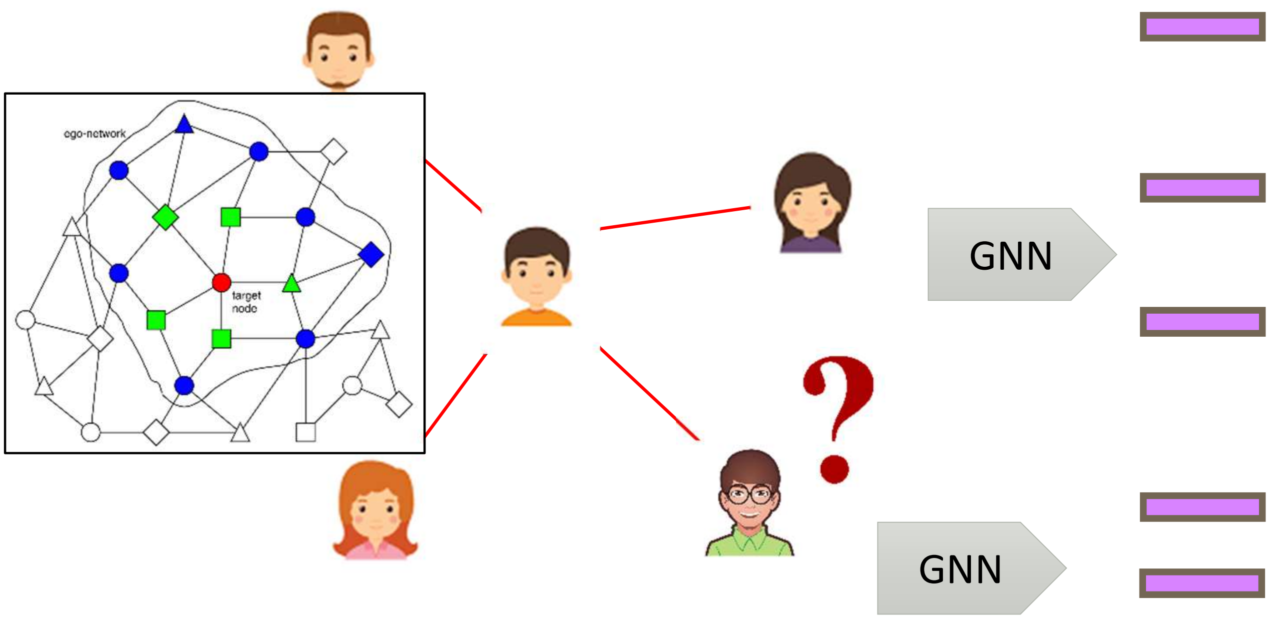
GNNs and the downstream classification/regression models can be trained in an **end-to-end** fashion.



Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

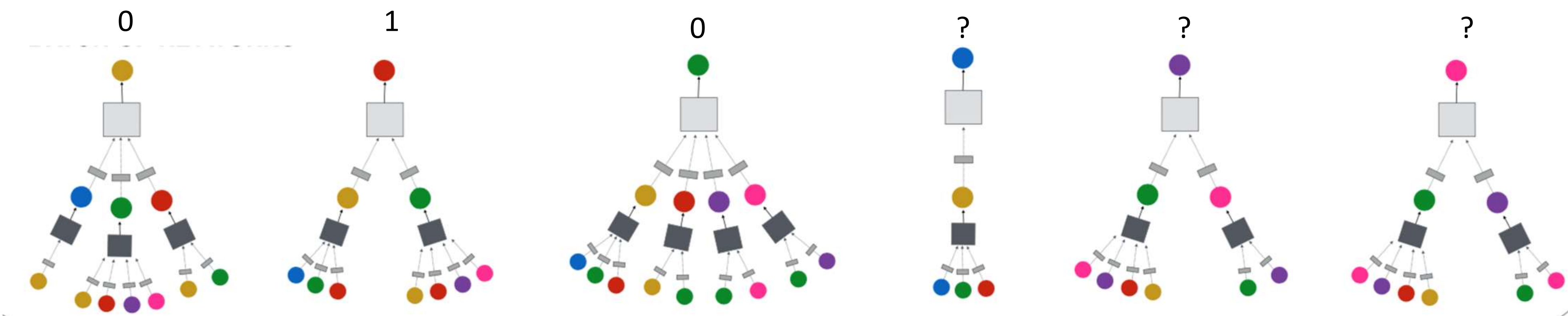
GNNs are ***inductive*** because they learn the same neural networks on all the nodes and edges.



Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

- Node classification is trained in the semi-supervised setting.



Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

- We train a link prediction model with connectivity of nodes as the training signal.
  - Positive edges are trained against a few negative edges

Graph convolution to encode node embeddings

Graph convolution to encode node embeddings

Credits: Adapted from https://github.com/dglai/KDD20-Hands-on-Tutorial

Quick Review of GNNs

- Graph readout to compute graph embeddings.
- Train a graph classifier on the graph embedding.

$$g = readout\left(h_1^{(l)}, h_2^{(l)}, \dots, h_n^{(l)}\right)$$

$$\text{loss} = \text{CrossEntryLoss}(f(g), label)$$