

DataScience for Development and Social Change, 2015

Science

Getting information from
your datasets. Truthfully.

Machine Learning

- ❖ Build a model that extract patterns from data
- ❖ Use those patterns to predict missing or future data
- ❖ Do it automatically

(part of artificial intelligence)

Example Uses

- ❖ Search engines: producing results tailored to you
- ❖ Spam filters: learning what should go into your spam folder (and not)
- ❖ Handwritten character recognition: recognizing “a”
- ❖ Gap-filling: estimating missing data values
- ❖ Prediction

Why Use Machine Learning?

- ❖ You don't have all the features you want in your data.
 - ❖ Some features don't exist
 - ❖ Some features have missing data
- ❖ You want to predict an outcome (eg. loan default) based on previous examples

Machine Learning Areas

- ❖ Classification: predict classes
- ❖ Regression: predict numerical values
- ❖ Clustering: predict group membership
- ❖ Dimension reduction: deciding which factors (e.g. age, sex, education, location, chickens) are most important in a prediction

Classification

Classification is the allocation of a piece of information to a category (e.g. male, female, other).

- ❖ Often need to automate this:
 - ❖ e.g. huge dataset, specialist knowledge (e.g. Tagalog, non-obvious connections), regularly-updated data etc.

Classification: Practice

- ❖ Which of these people are male or female?
 1. She previously worked at Kings College London.
 2. Bayani advises the UN on the use of drones.
 3. Kim is a leading scholar on text classification.
 4. His work on sand eels is renowned.
 5. Diwata Jones.
 6. He works closely with Sandra Smith and her work on fish.

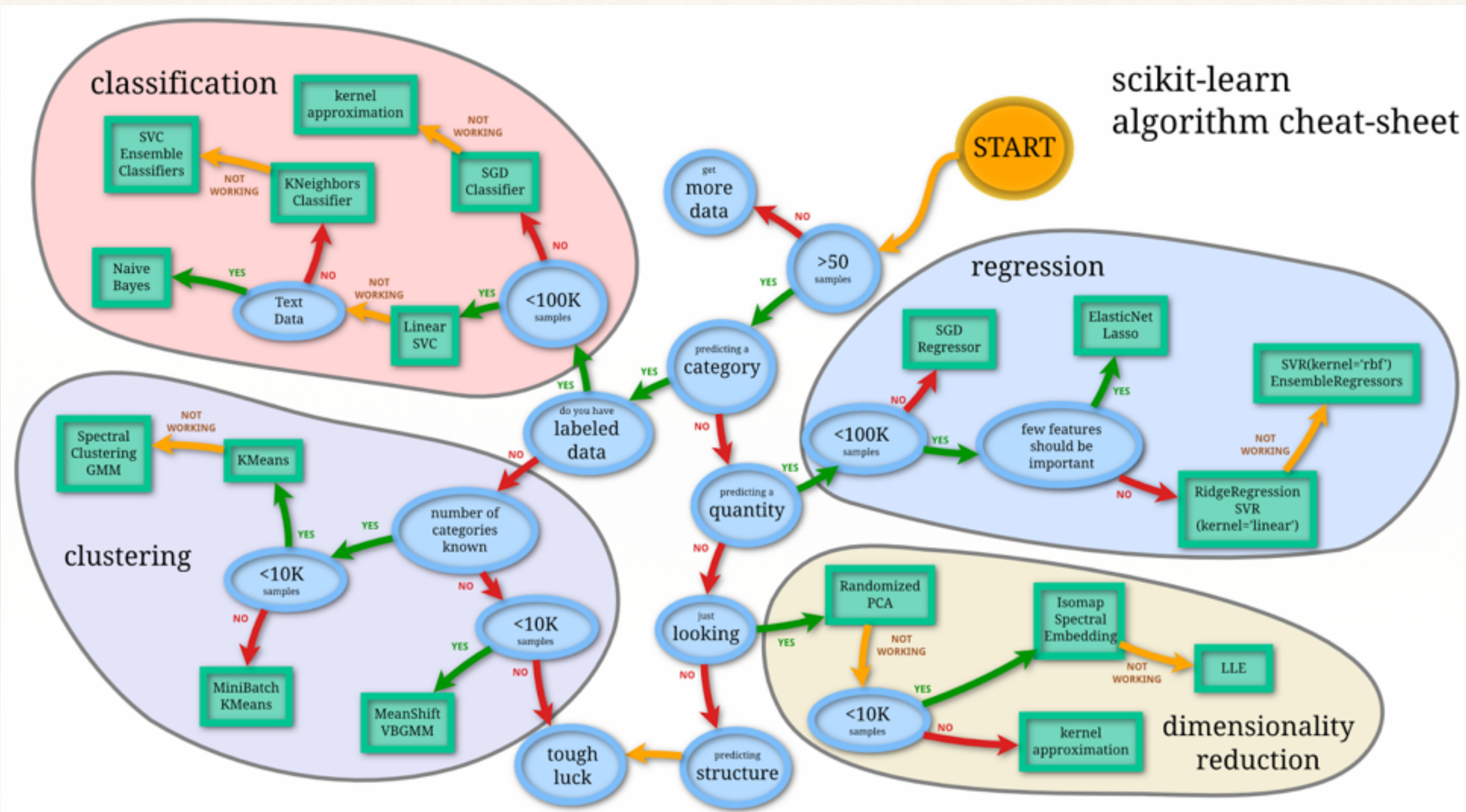
Classification: Practice

1. Female: “She” at start of description.
 2. Male: male first name (look in [babynames.ch](#))
 3. Unknown: “Kim” is male or female first name.
 4. Male: “His” at start of description.
 5. Female: female first name (look in [babynames.ch](#))
 6. Male: “He” at start of description, but note the “her” later in the text.
-
- ❖ Alternative gender classification methods include:
 - ❖ GenderAnalyzer: guesses gender of writer (has API)
 - ❖ Name endings: Names ending in a, e and i are more likely to be female, while names ending in k, o, r, s and t are more likely to be male

The Scikit.learn Library

- ❖ Python library
- ❖ Contains most common machine learning algorithms
- ❖ `import sklearn`

Scikit.Learn Algorithms



The Iris Dataset

Fisher's *Iris* Data

Sepal length ◆	Sepal width ◆	Petal length ◆	Petal width ◆	Species ◆
5.1	3.5	1.4	0.2	<i>I. setosa</i>
4.9	3.0	1.4	0.2	<i>I. setosa</i>
4.7	3.2	1.3	0.2	<i>I. setosa</i>
4.6	3.1	1.5	0.2	<i>I. setosa</i>
5.0	3.6	1.4	0.2	<i>I. setosa</i>
5.4	3.9	1.7	0.4	<i>I. setosa</i>
4.6	3.4	1.4	0.3	<i>I. setosa</i>
5.0	3.4	1.5	0.2	<i>I. setosa</i>



Reading in the Iris Dataset

- ❖ `import numpy as np`
- ❖ `from sklearn import datasets`
- ❖ `iris = datasets.load_iris()`
- ❖ `iris_X = iris.data`
- ❖ `iris_Y = iris.target`
- ❖ `print("Iris dataset: {}".format(iris))`
- ❖ `print("Xs: {}".format(iris_X))`
- ❖ `print("Ys: {}".format(iris_Y))`
- ❖ `print("Unique Y values: {}".format(np.unique(iris_Y)))`

Training and Test sets

- ❖ “Training set”: a set of pre-classified examples, for the classifier to learn from
- ❖ “Test set”: a smaller set of pre-classified examples, for the classifier to check its predictions

Split the Iris data into train and test

```
ntest=10
```

```
np.random.seed(0)
```

```
indices = np.random.permutation(len(iris_X))
```

```
iris_X_train = iris_X[indices[:-ntest]]
```

```
iris_Y_train = iris_Y[indices[:-ntest]]
```

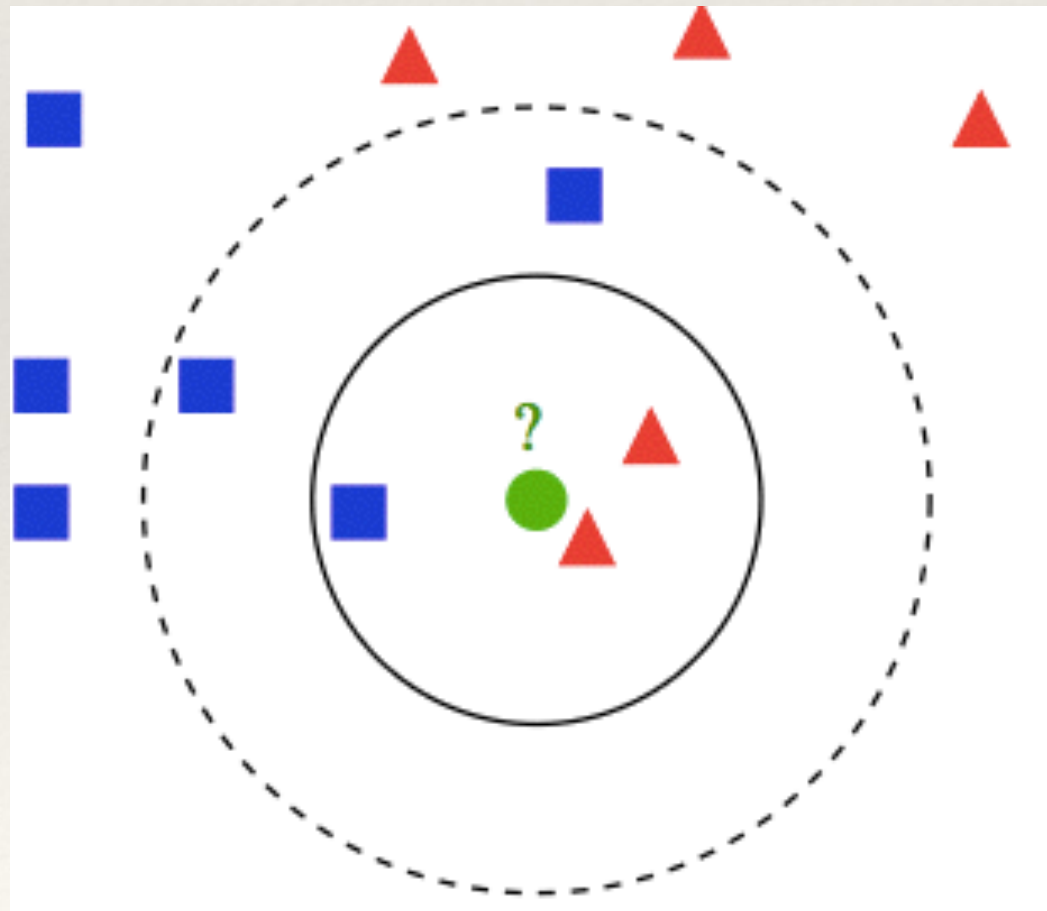
```
iris_X_test = iris_X[indices[-ntest:]]
```

```
iris_Y_test = iris_Y[indices[-ntest:]]
```

```
print("{} training points, {} test points".format(len(iris_X_train),  
len(iris_X_test)))
```

The kNN classifier

- ❖ “k Nearest Neighbours”
 - ❖ Look at the k closest points to an unknown point
 - ❖ The class of the unknown point is the class of most of those “neighbors”



kNN on the Iris Dataset

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier()
```

```
knn.fit(iris_X_train, iris_Y_train)
```

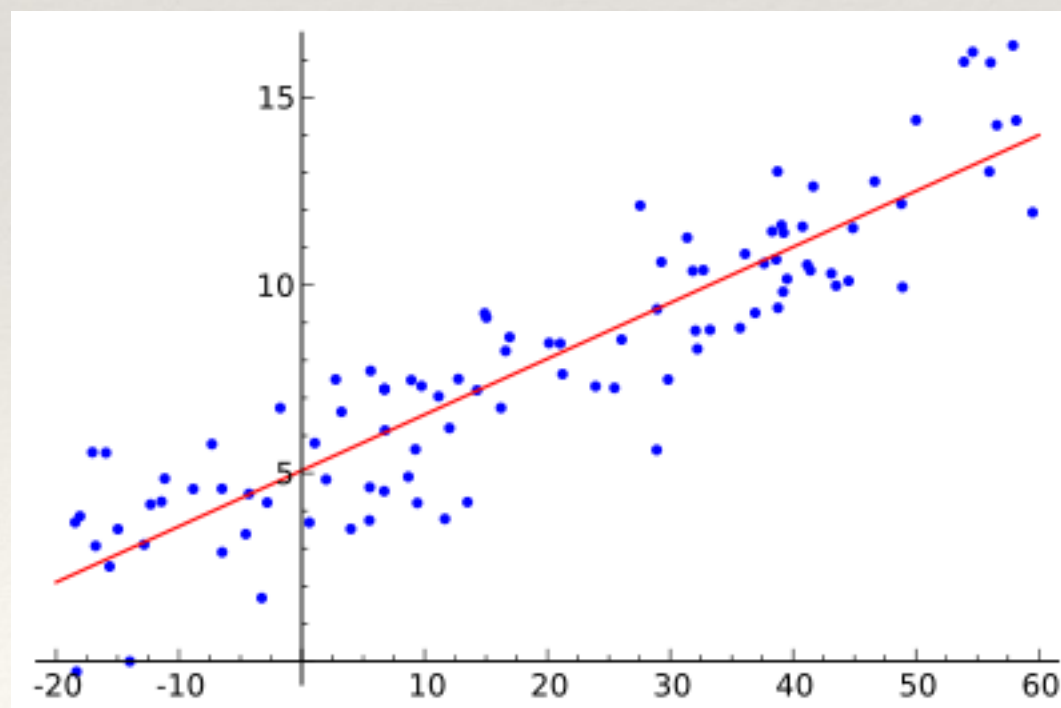
```
predicted_classes = knn.predict(iris_X_test)
```

```
print("kNN predicted classes: {}".format(predicted_classes))
```

```
print("Real classes: {}".format(iris_Y_test))
```

Regression

- ❖ Finds the best-fit line between points
- ❖ Because: you need to know the relationship between 2 or more features
- ❖ Needs:
 - ❖ data points, in 2 or more dimensions
 - ❖ “best-fit” equation, e.g. city-block metric, least-squares, edit distance etc



The Diabetes Dataset

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
from sklearn import datasets, linear_model
```

```
# Load the diabetes dataset
```

```
# 442 people: diabetes, age, sex, weight, blood pressure etc
```

```
diabetes = datasets.load_diabetes()
```

Diabetes Training and Test Sets

- ❖ `# Use only one feature`
- ❖ `diabetes_X = diabetes.data[:, np.newaxis]`
- ❖ `diabetes_X_temp = diabetes_X[:, :, 2]`
- ❖ `# Split the data into training / testing sets`
- ❖ `diabetes_X_train = diabetes_X_temp[:-20]`
- ❖ `diabetes_X_test = diabetes_X_temp[-20:]`
- ❖ `# Split the targets into training / testing sets`
- ❖ `diabetes_y_train = diabetes.target[:-20]`
- ❖ `diabetes_y_test = diabetes.target[-20:]`

Do the Linear Regression

```
#from sklearn import datasets, linear_model
```

```
# Create linear regression object
```

```
regr = linear_model.LinearRegression()
```

```
# Train the model using the training sets
```

```
regr.fit(diabetes_X_train, diabetes_y_train)
```

Look at your results

```
import matplotlib.pyplot as plt
```

```
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
```

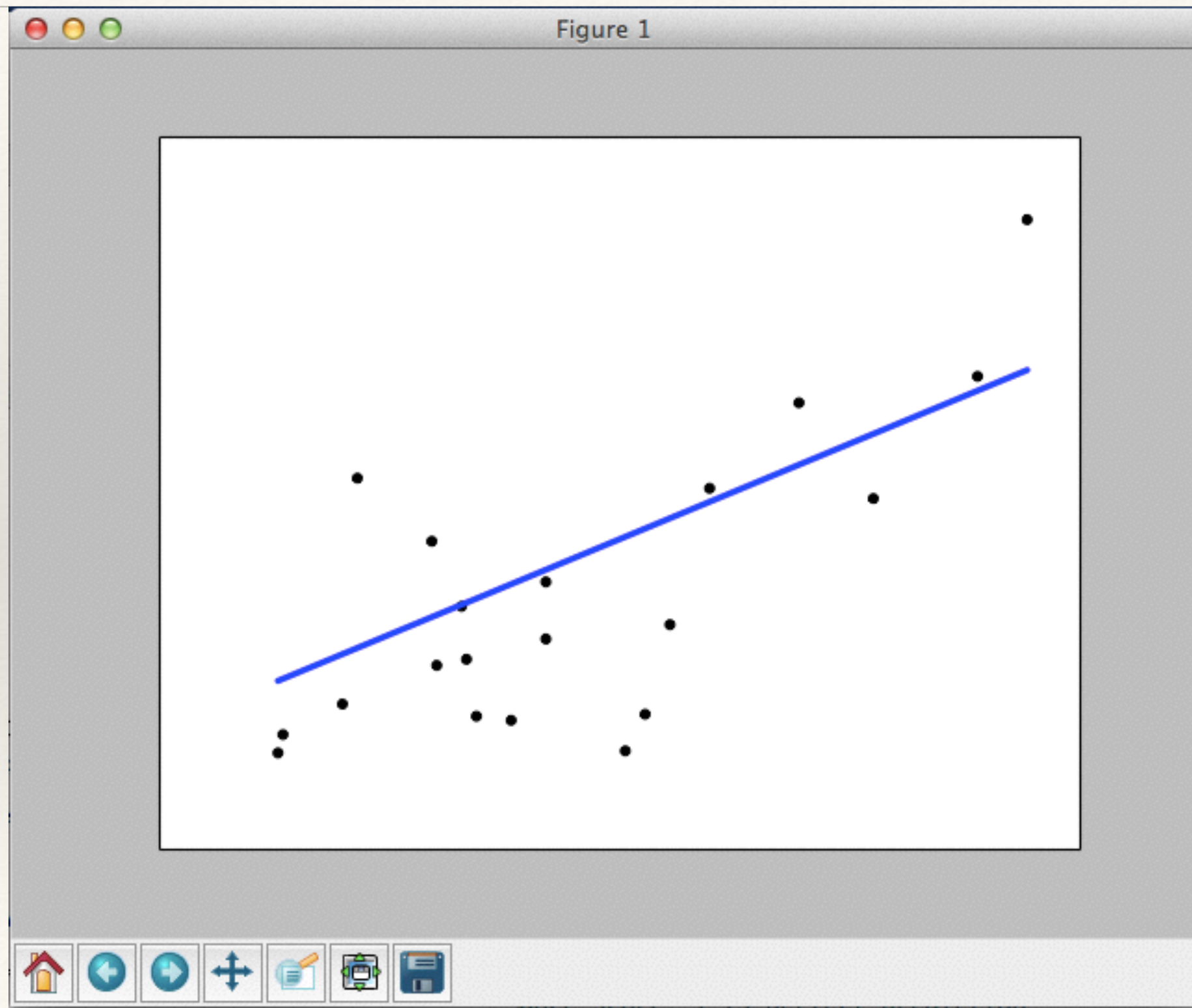
```
plt.plot(diabetes_X_test, regr.predict(diabetes_X_test),  
color='blue', linewidth=3)
```

```
plt.xticks(())
```

```
plt.yticks(())
```

```
plt.show()
```

Results



How Good was the Model?

The coefficients

```
print('Coefficients: \n', regr.coef_)
```

The mean square error

```
print("Residual sum of squares: %.2f"
```

```
      % np.mean((regr.predict(diabetes_X_test) - diabetes_y_test) ** 2))
```

Explained variance score: 1 is perfect prediction

```
print('Variance score: %.2f' % regr.score(diabetes_X_test,  
diabetes_y_test))
```

Logistic Regression

- ❖ Because: you want to predict the class of a datapoint
- ❖ Uses the logistic function ('S' shaped curve) to separate datasets.
- ❖ Examples: predict whether person will be diabetic, given their BMI
- ❖ <http://blog.yhathq.com/posts/logistic-regression-and-python.html>

Unsupervised Learning

- ❖ Supervised learning:
 - ❖ We 'teach' the computer how to do a task
 - ❖ E.g. we tag male/female names until the machine can do this reliably (i.e. low number of misclassifications) for itself
- ❖ Unsupervised learning:
 - ❖ The computer learns for itself, without teaching
 - ❖ E.g. the computer separates a dataset into classes of closely-related data points, without being told what or where these are

Clustering

- ❖ Divide a dataset up into n related “clusters” or classes
 - ❖ **Because:** you want to know if groups exist in your data (so you can investigate further / use characteristics of those groups to advantage)
 - ❖ **Issue:** knowing “ n ”
 - ❖ **Algorithms:** k-means (and many many others)
 - ❖ **Python module:** sklearn.cluster

k-Means Algorithm

- ❖ Divides a dataset into k groups, where the datapoints in each group are closer to each other, than to datapoints in other groups
- ❖ Issue: you have to choose “ k ”

k-Means with the Iris Dataset

```
from sklearn import cluster, datasets
```

```
iris = datasets.load_iris()
```

```
k_means = cluster.KMeans(3)
```

```
k_means.fit(iris.data)
```

```
print("Generated labels: {}".format(k_means.labels_))
```

```
print("Real labels: {}".format(iris.target))
```

Dimension Reduction

- ❖ For classification to work, you need a feature set that's useful, e.g.
 - ❖ First names
 - ❖ First name endings
 - ❖ Personal pronouns in text
 - ❖ Not: department (e.g. Engineering)
- ❖ Dimension reduction helps you choose that feature list

Network Analysis

- What is a network?
- What features does a network have?
- What analysis is possible with those features?
- How do we explain that analysis?

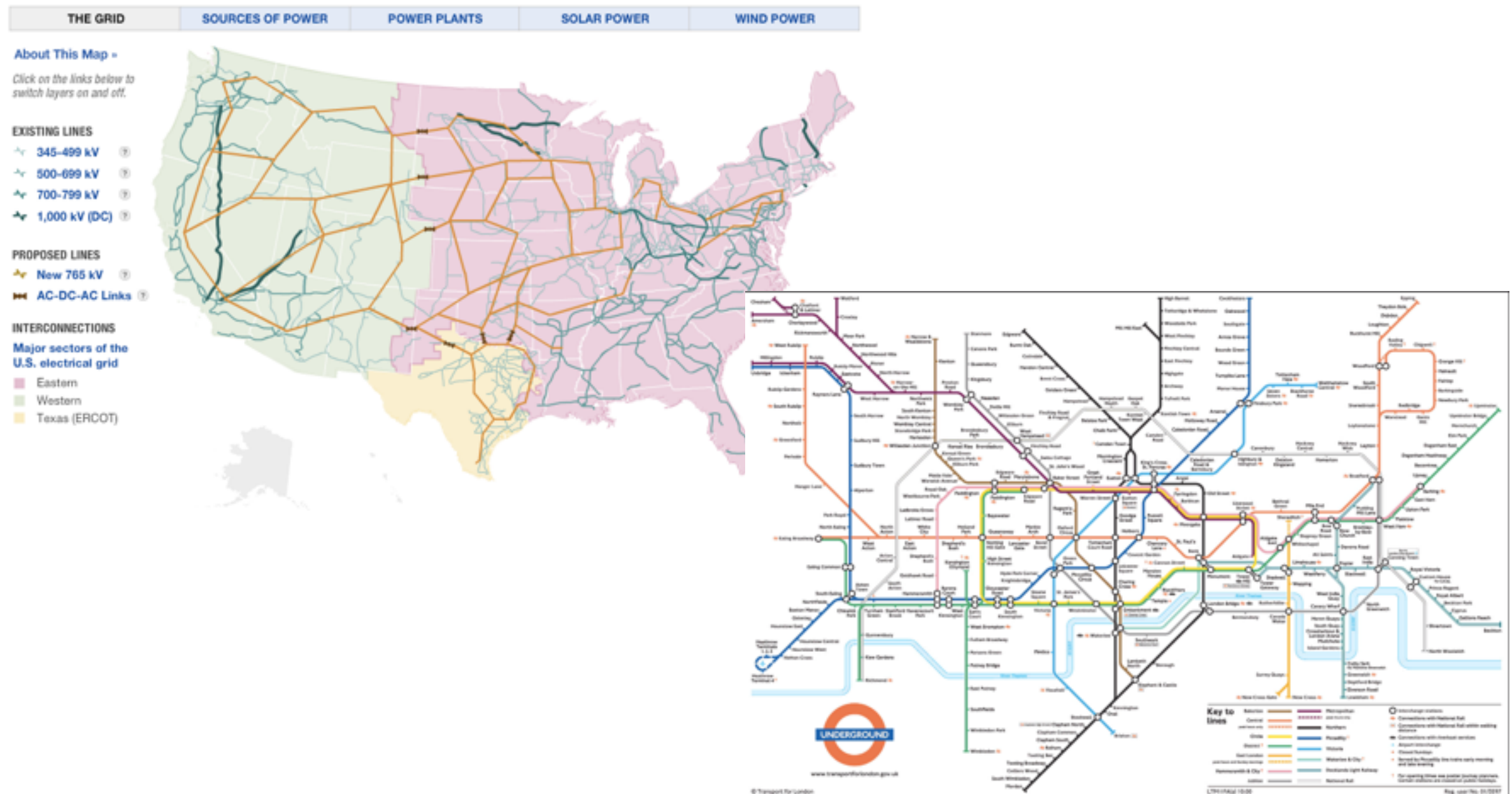
“Network”

“A group of interconnected people or things”
(Oxford English Dictionary)



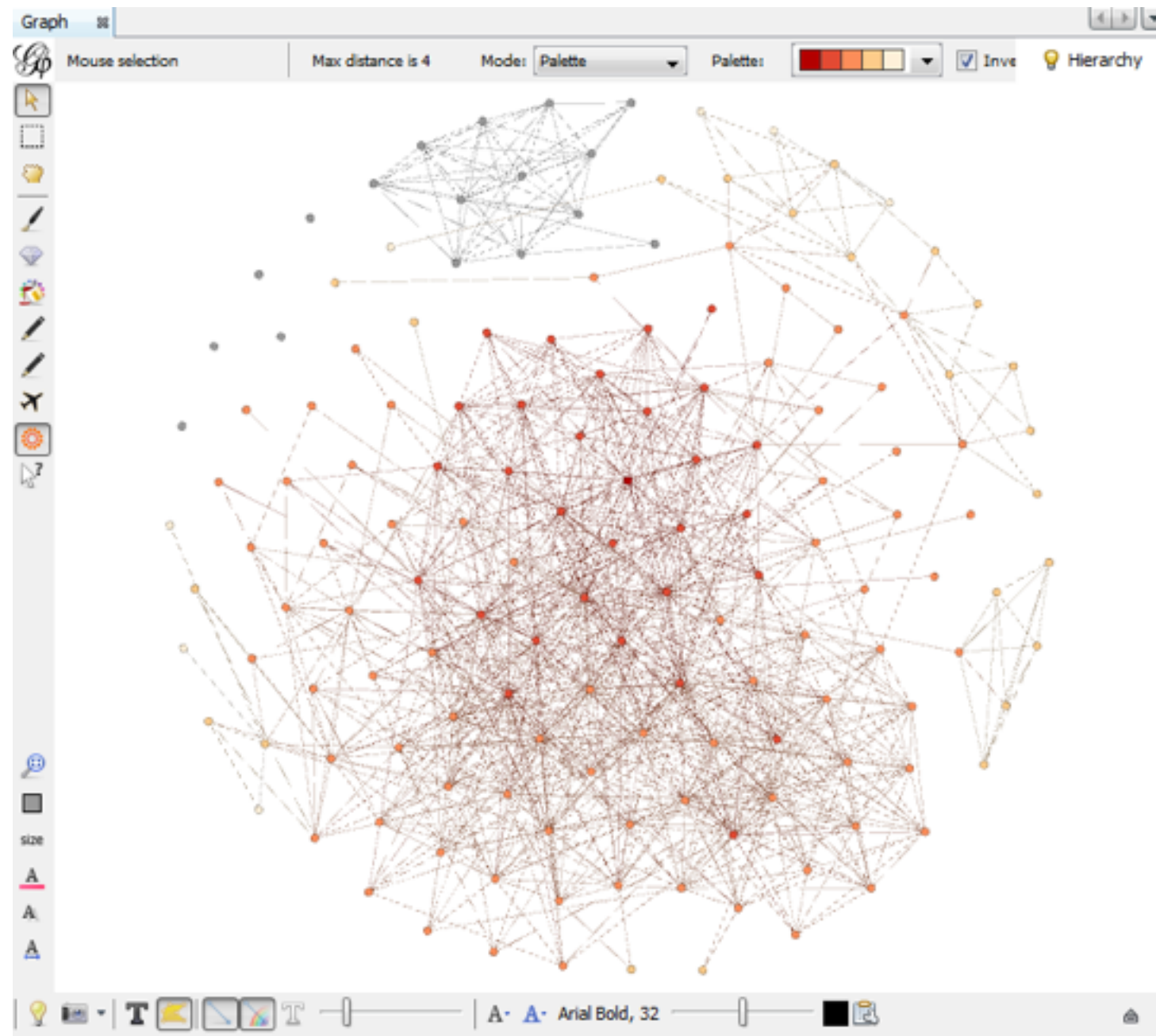
Use networks to understand, use and explain
relationships

Infrastructure Networks




NPR: Visualising the US Power Grid
Transport for London: London Underground Map

Social Networks



(Sara's Facebook friends, in Gephi)

Songs

 **Spotify** | DEVELOPER My Apps Tools Support News

[Home](#)
[News](#)
[Developer Showcase](#)
[My Applications](#)
[Web API](#)
[User Guide](#)
[Beginner's Tutorial](#)
[API Endpoint Reference](#)

Get an Artist's Related Artists

Get Spotify catalog information about artists similar to a given artist. Similarity is based on analysis of the Spotify community's **listening history**.

Endpoint

```
GET https://api.spotify.com/v1/artists/{id}/related-artists
```

Request Parameters

Path element	Value
id	The Spotify ID for the artist.

[Spotify API reference](#)

Words

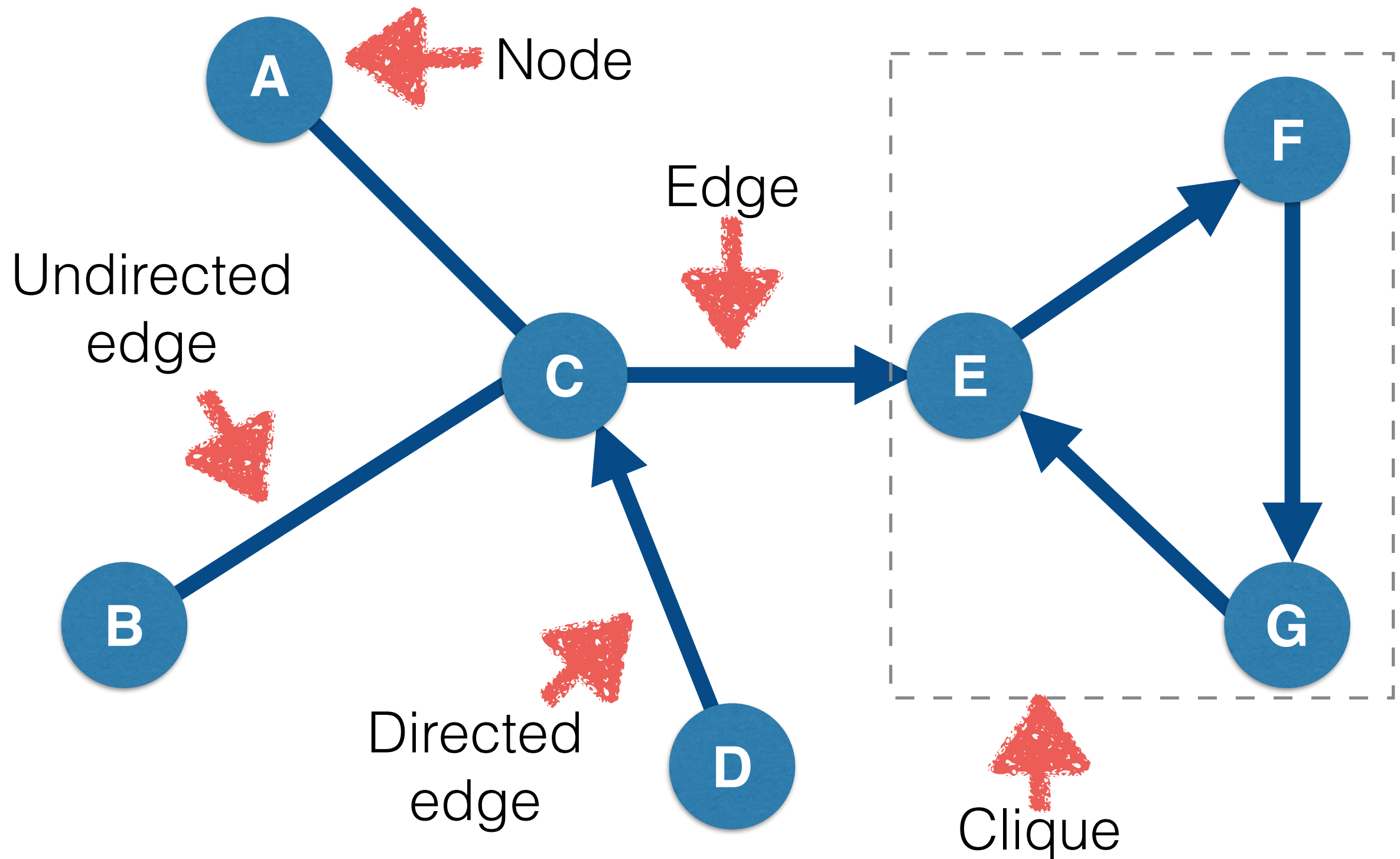
	bank	career	education	employment	example	experience	Florida	home	human	insurance	job	jobs	monster.com	office	opening	opportunities
bank	--															
career		--														
education		1	--													
employment	3		2	--												
example			35		--											
experience			42		35	--										
Florida				9			--									
home				7				--								
human	1								--							
insurance	1		2	6		2				--						
job	28	2	4	4			4	1	2	3	--					
jobs	1	1	4	16			6				17	--				
monster.com											1		--			
office				8				2			3			--		
opening											30	30			--	
opportunities		3		20						1	24	15			30	--

([Wise blogpost](#) on word co-occurrence matrices)

Network Analysis

Use networks to understand, use and explain
relationships

Network Features

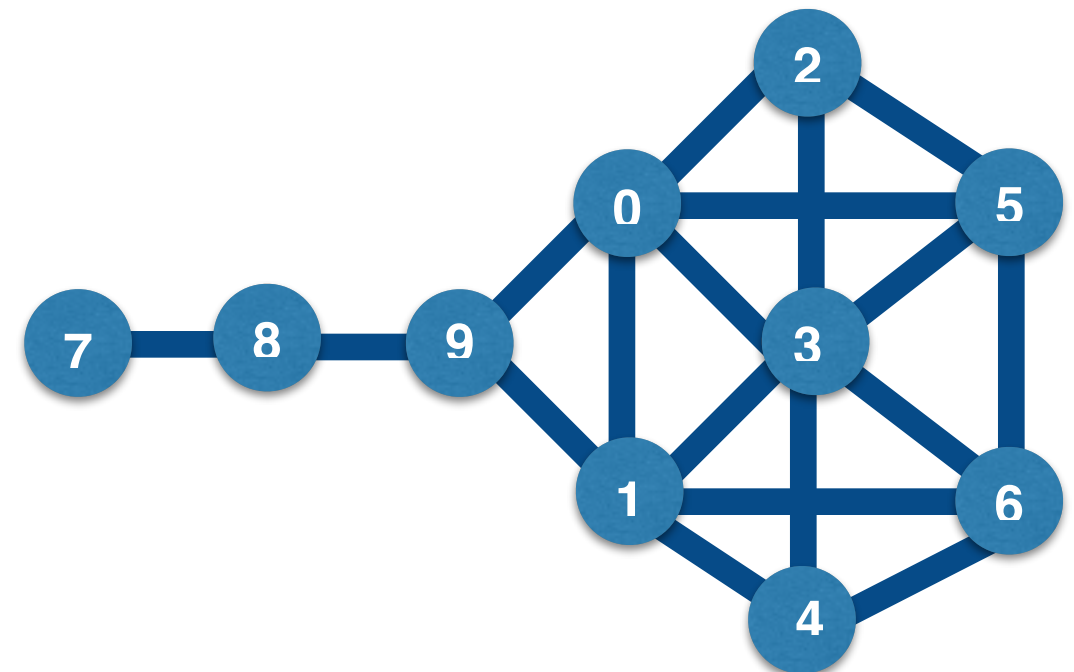


Network Representations

- **Diagram**

- **Adjacency matrix**

```
[ [ 0, 1, 1, 1, 0, 0, 0, 0, 0, 1],  
  [ 1, 0, 0, 1, 1, 0, 1, 0, 0, 1],  
  [ 1, 0, 0, 1, 0, 1, 0, 0, 0, 0],  
  [ 1, 1, 1, 0, 1, 1, 1, 0, 0, 0],  
  [ 0, 1, 0, 1, 0, 0, 1, 0, 0, 0],  
  [ 0, 0, 1, 1, 0, 0, 1, 0, 0, 0],  
  [ 0, 1, 0, 1, 1, 1, 0, 0, 0, 0],  
  [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],  
  [ 0, 0, 0, 0, 0, 0, 0, 1, 0, 1],  
  [ 1, 1, 0, 0, 0, 0, 0, 0, 1, 0]]
```



- **Adjacency list**

```
{0: [1, 2, 3, 9], 1: [0, 9, 3, 4, 6], 2: [0, 3, 5], 3: [0, 1, 2, 4, 5, 6],  
  4: [1, 3, 6], 5: [2, 3, 6], 6: [1, 3, 4, 5], 7: [8], 8: [9, 7], 9: [8, 1, 0]}
```

- **Edge list**

```
{(0,1), (0,2), (0,3), (0,9), (1,3), (1,4), (1,6), (1,9), (2,3), (2,5), (3,4),  
  (3,5), (3,6), (4,6), (5,6), (7,8), (8,9)}
```

- **Maths**

$G = (V, E, e)$

The NetworkX Library

- Python network analysis library

```
import networkx as nx
```

```
edgelist = {(0,1),(0,2),(0,3),(0,9),(1,3),(1,4),(1,6),  
(1,9),(2,3),(2,5),(3,4),(3,5),(3,6),(4,6),(5,6),(7,8),  
(8,9)}
```

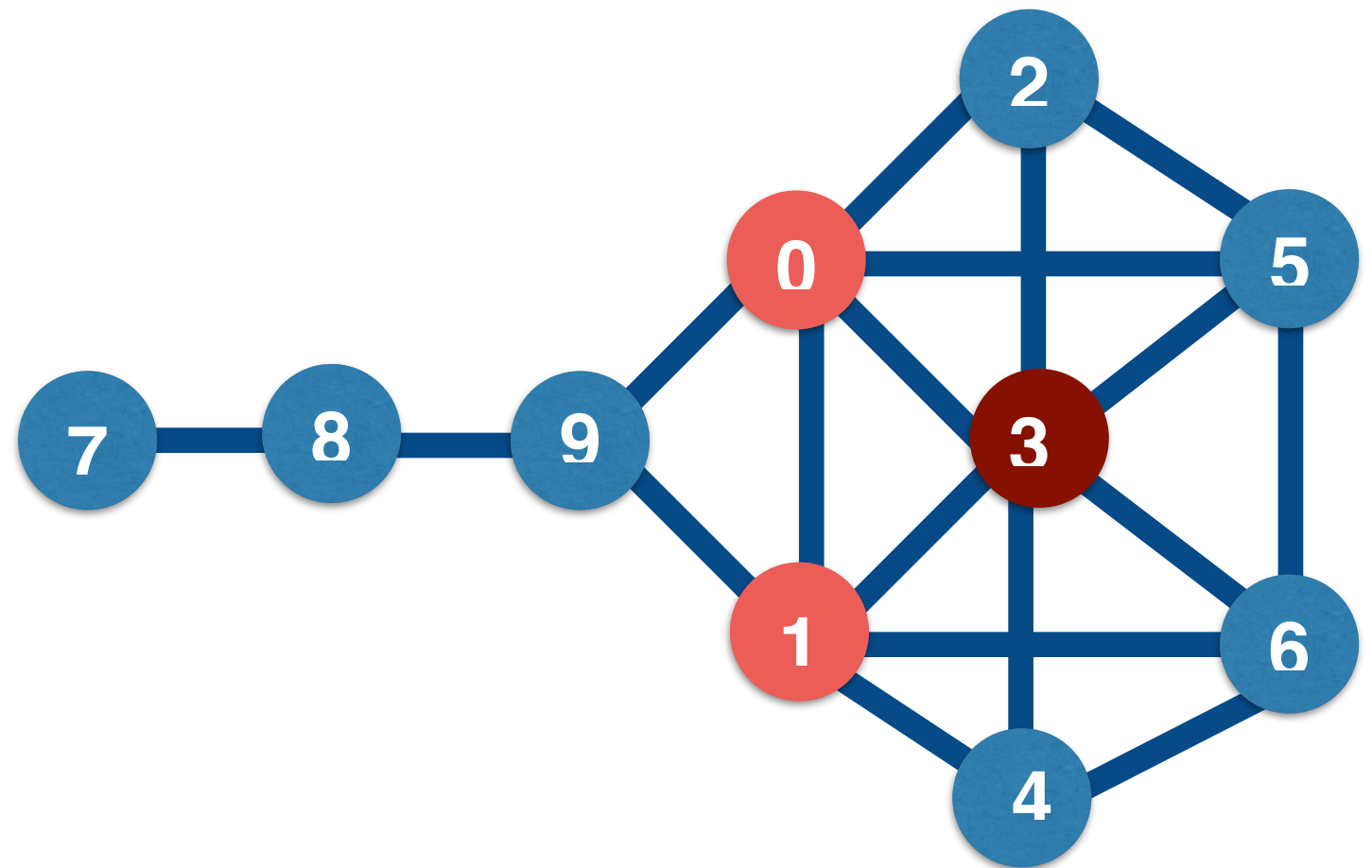
```
G = nx.Graph()  
for edge in edgelist:  
    G.add_edge(edge[0], edge[1])
```

Node Centrality

- Finding the most “important”/“influential” nodes
 - i.e. how “central” is a node to the network

Degree centrality: “who has lots of friends?”

3	0.666
0	0.555
1	0.555
5	0.444
6	0.444
2	0.333
4	0.333
9	0.333
8	0.222
7	0.111

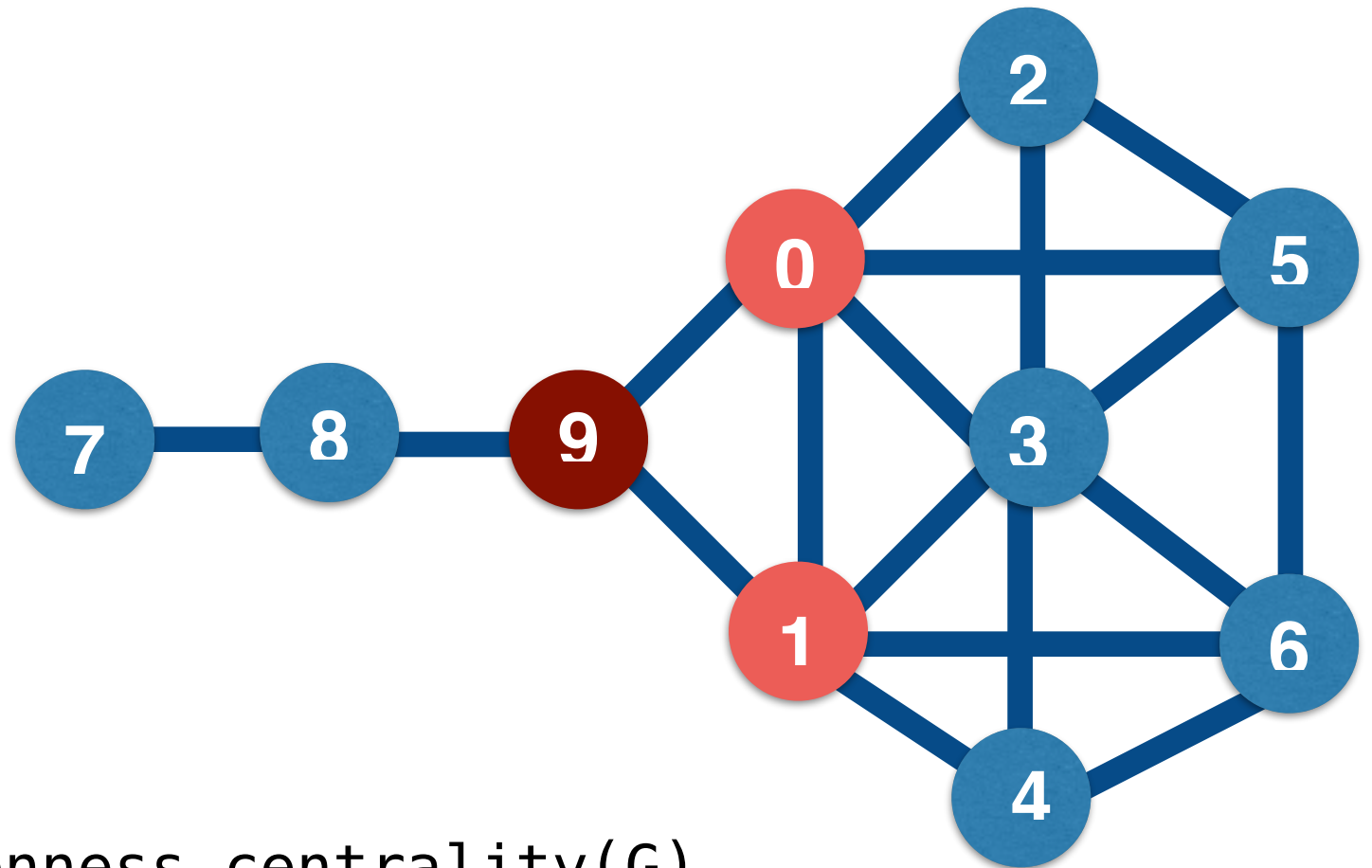


`nx.degree centrality(G)`

= number of edges directly connected to n

Betweenness centrality: “who are the bridges”?

9	0.38
0	0.23
1	0.23
8	0.22
3	0.10
5	0.02
6	0.02
2	0.00
4	0.00
7	0.00

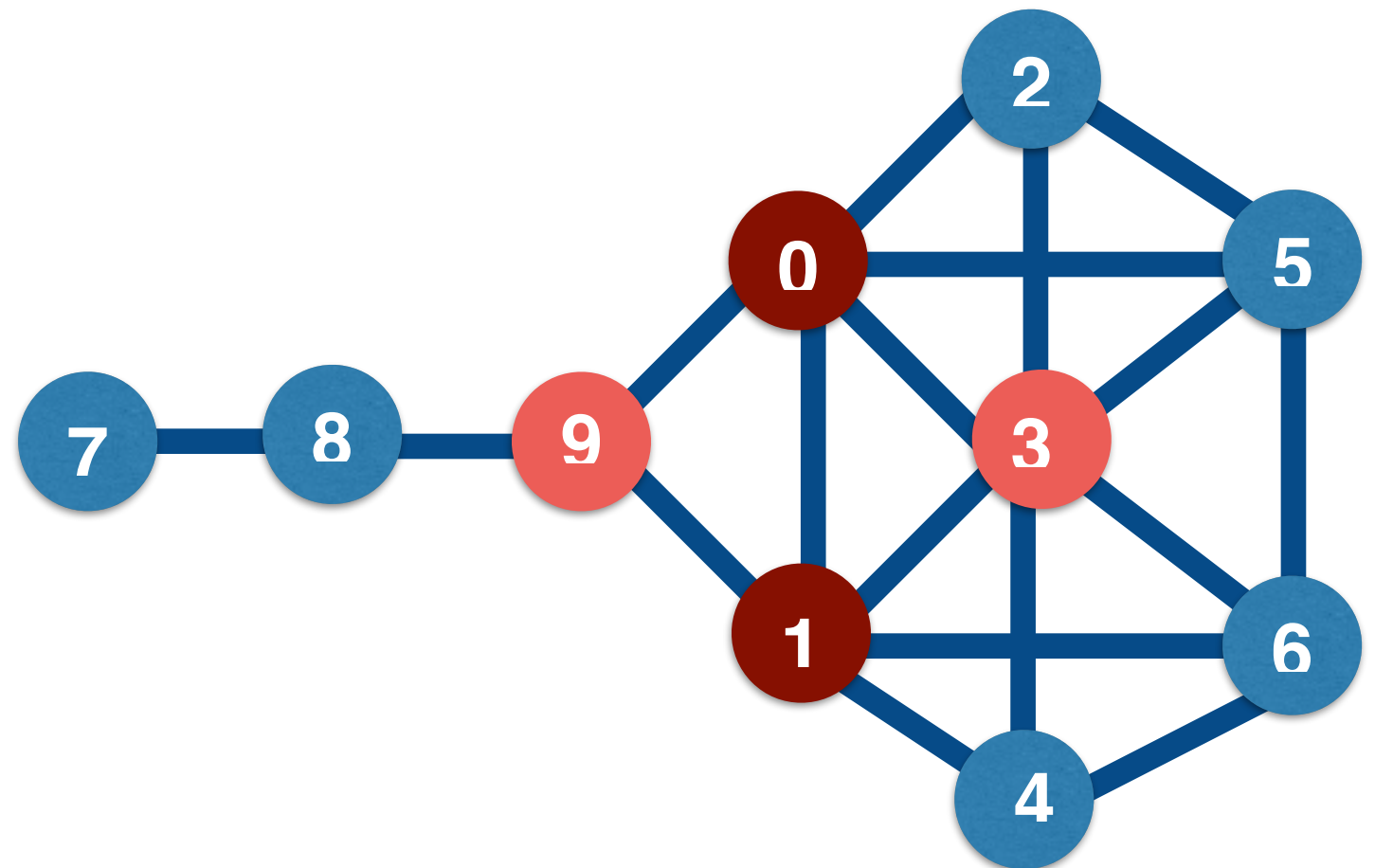


`nx.betweenness centrality(G)`

= (number of shortest paths including n / total number of shortest paths) / number of pairs of nodes

Closeness centrality: “who are the hubs”?

0	0.64
1	0.64
3	0.60
9	0.60
5	0.52
6	0.52
2	0.50
4	0.50
8	0.42
7	0.31

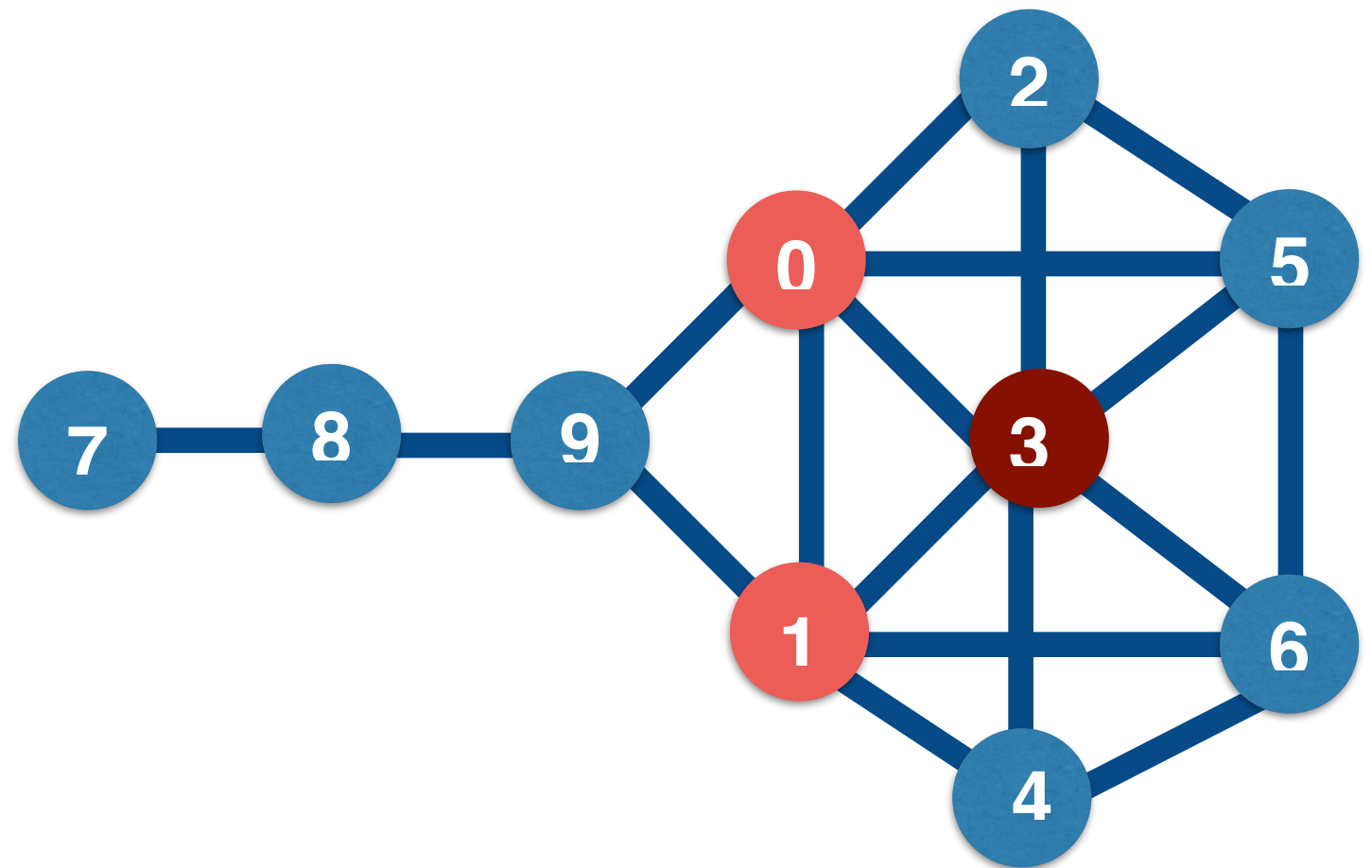


`nx.closeness centrality(G)`

= $\text{sum}(\text{distance to each other node}) / (\text{number of nodes} - 1)$

Eigenvalue centrality “who has most network influence”?

3	0.48
0	0.39
1	0.39
5	0.35
6	0.35
2	0.28
4	0.28
9	0.19
8	0.04
7	0.01



```
nx.eigenvector_centrality(G)
```


Network properties

- Characteristic path length: average shortest distance between all pairs of nodes
- Clustering coefficient: how likely a network is to contain highly-connected groups
- Degree distribution: histogram of node degrees

Community Detection

“Are there groups in this network?”

“What can I do with that information?”

Disconnected Networks

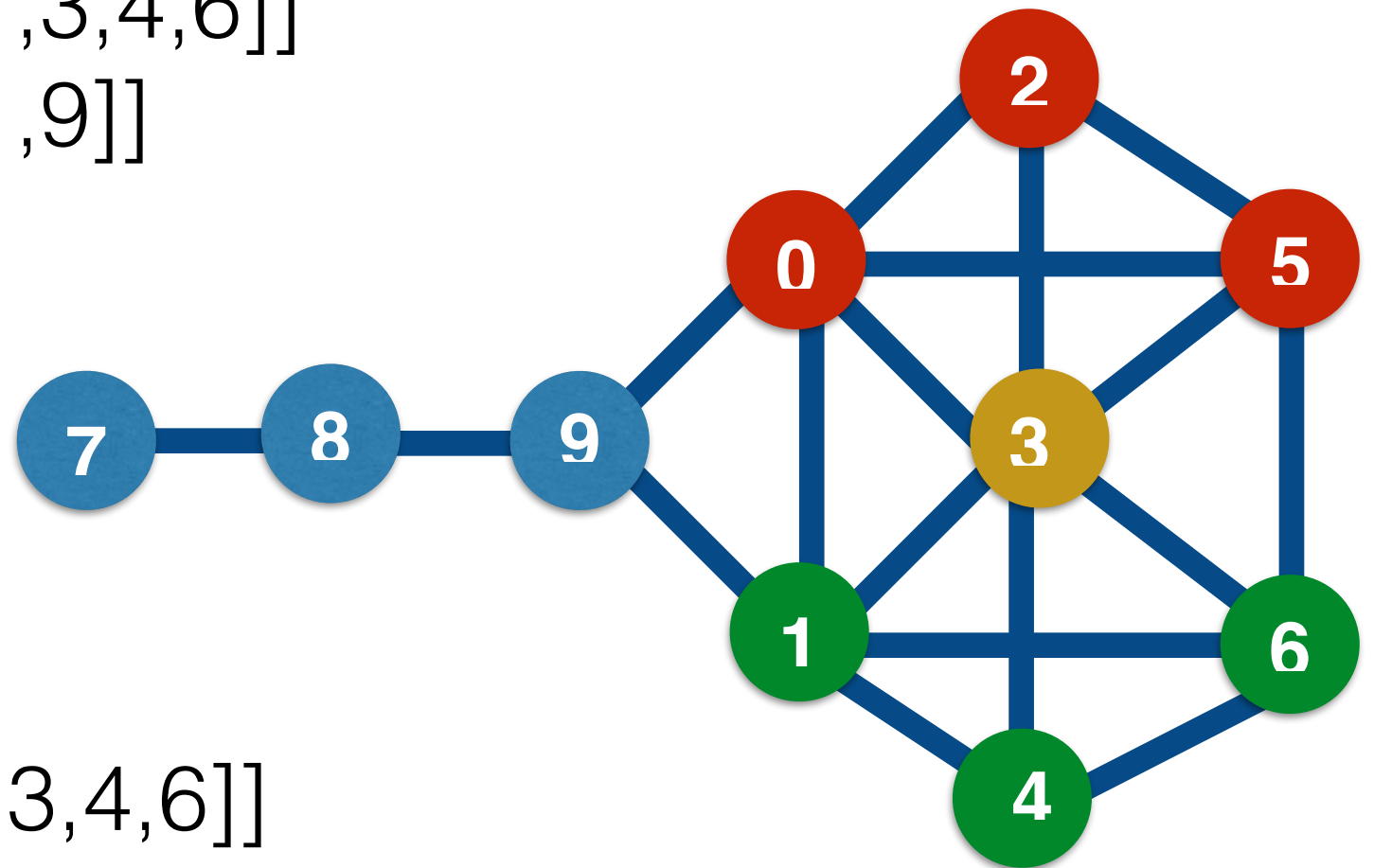
- Not all nodes are connected to each other
- Connected component = every node in the component can be reached from every other node
- Giant component = connected component that covers most of the network

Cliques and K-Cores

4-cliques: $[[0,2,3,5],[1,3,4,6]]$

3-cliques: $[[0,1,3],[0,1,9]]$

2-cliques: $[[7,8],[8,9]]$



3-cores: $[[0,2,3,5], [1,3,4,6]]$

2-core: $[0,1,2,3,4,5,6,9]$

```
nx.find_cliques(G)
nx.k_clique_communities(G, 3)
```

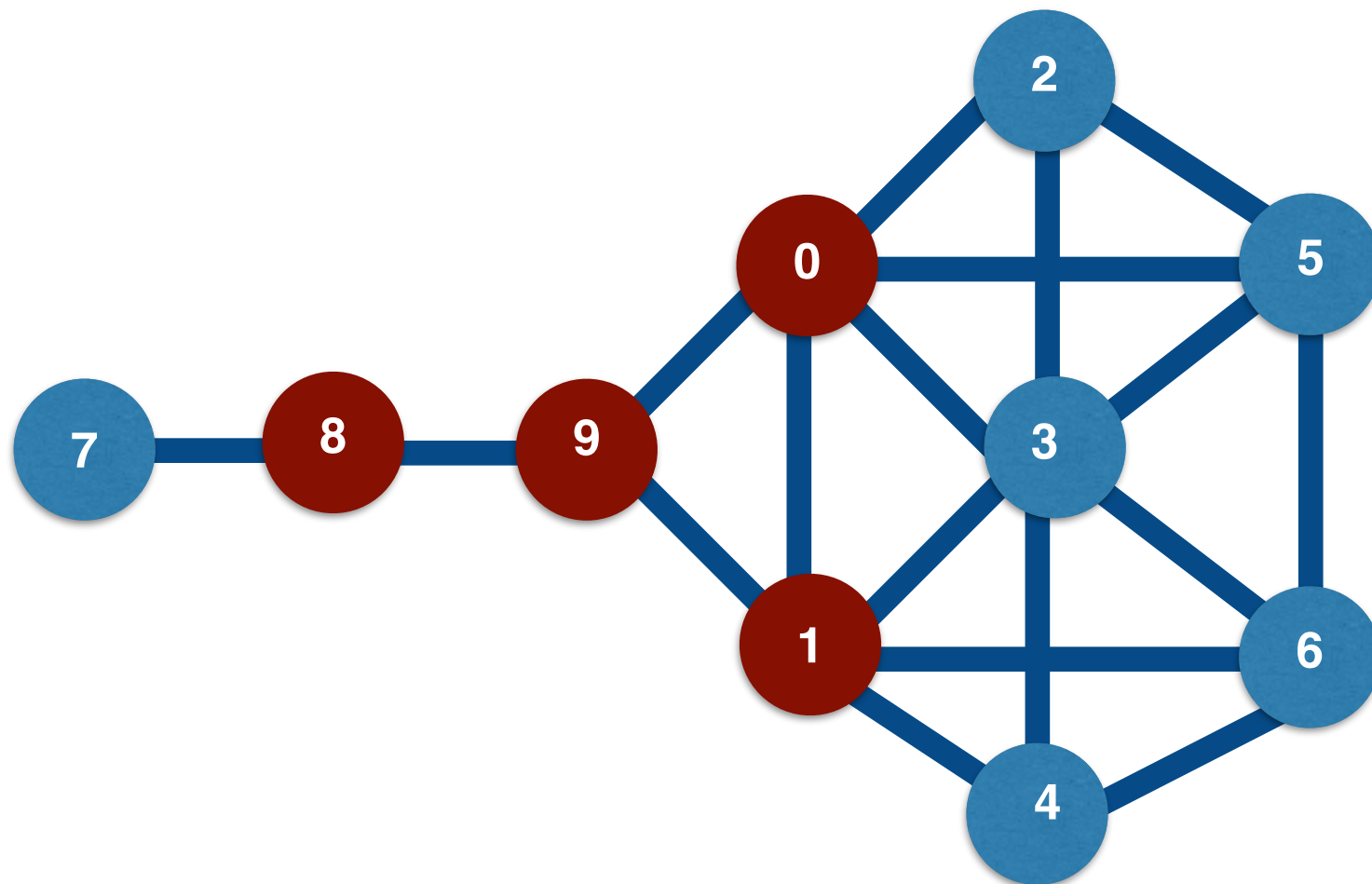

Other Clique methods

- N-clique: every node in the clique is connected to all other nodes by a path of length n or less
- P-clique: each node is connected to at least $p\%$ of the other nodes in the group.

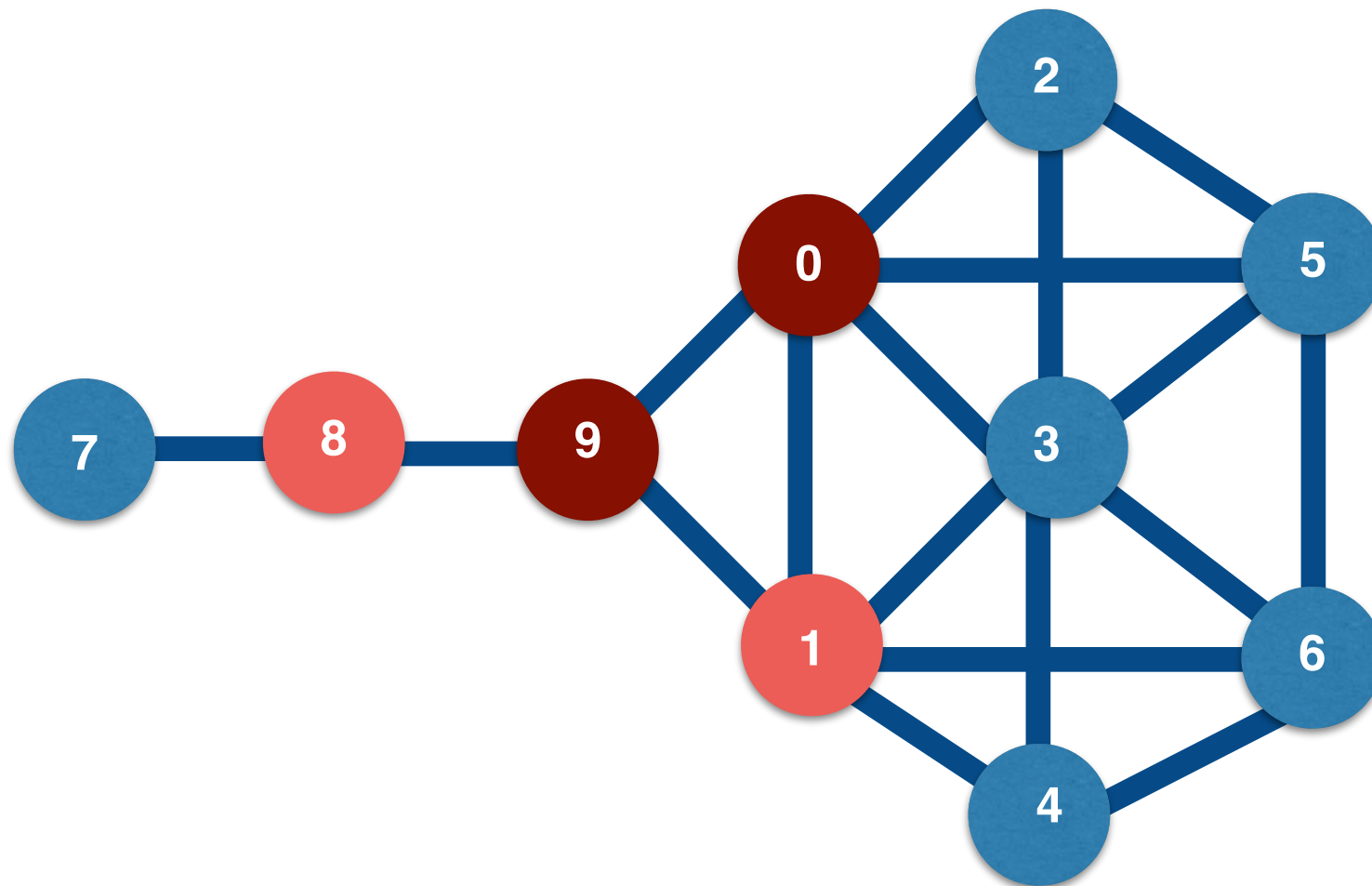
Network Effects

Predict how information or states (e.g. political opinion or rumours) are most likely to move across a network

Diffusion (Simple contagion)



Complex contagion



Describing Networks



Network diagram



Edge bundling

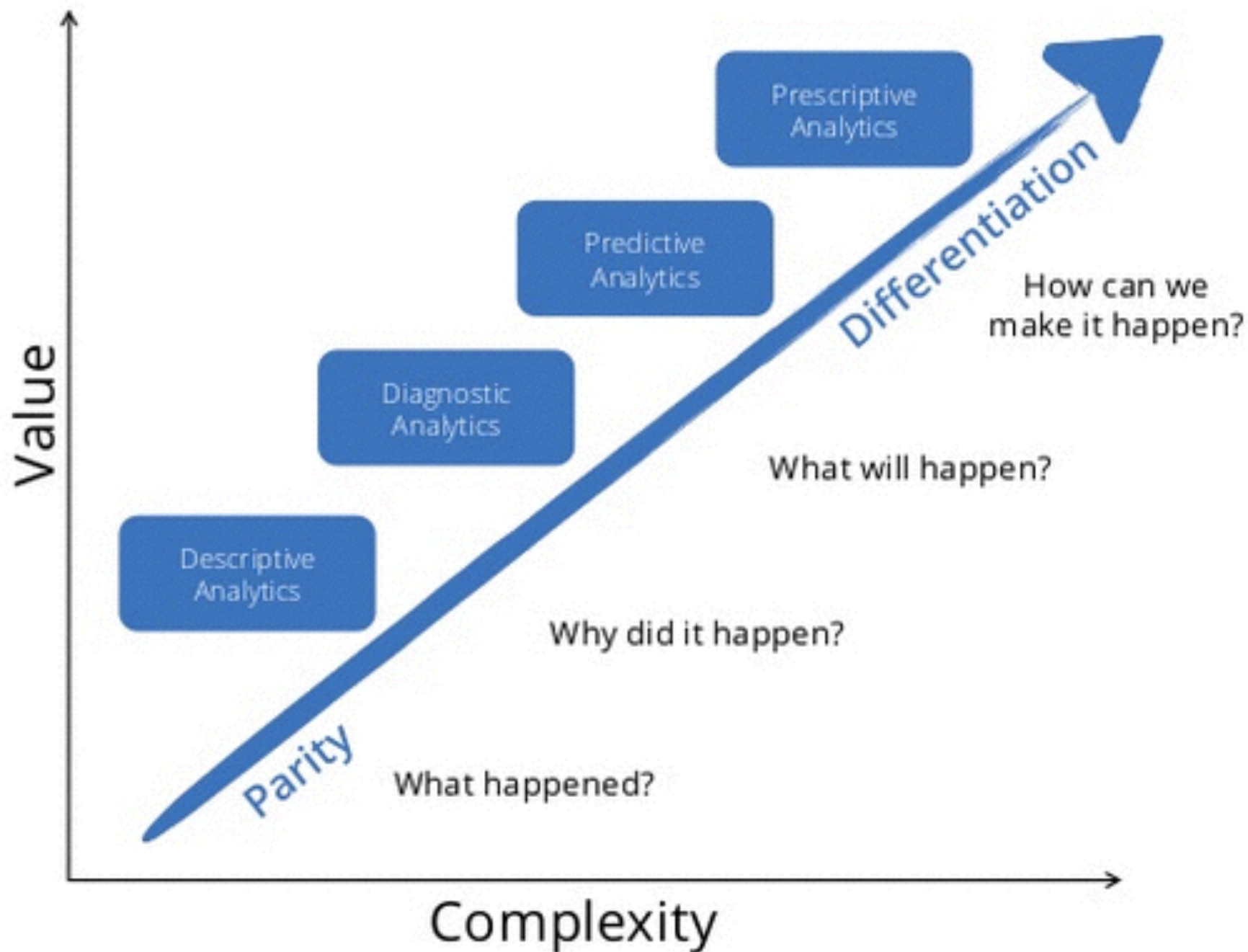
bl.ocks.org/mbostock/4062045
<http://bost.ocks.org/mike/uberdata/>
<http://bl.ocks.org/mbostock/7607999>

Network Analysis Tools

- Python libraries:
 - NetworkX
 - iGraph
 - graph-tool
 - Matplotlib (visualisation)
 - Pygraphviz (visualisation)
 - Mayavi (3d visualisation)
- Standalone tools:
 - SNAP
 - GUESS
 - NetMiner (free for students)
 - Gephi (visualisation)
 - GraphViz (visualisation)
 - NodeXL (excel add-on)

Longer list: http://en.wikipedia.org/wiki/Social_network_analysis_software

Moving Beyond Prediction



(Image: Ken Collier, Thoughtworks)

Continuing your Science journey

- ❖ <http://scikit-learn.org/stable/index.html>
- ❖ “An empirical classification of supervised learning algorithms”