Urban Informatics

Fall 2018

dr. federica bianco fb55@nyu.edu

@fedhere

CUSP

XII categorical distances
model diagnostics, OSMnx

Last Class!!!!!

Topics covered:

- Good practices with data: falsifiability, reproducibility
- Basic data retrieving and munging: APIs, Data formats
- SQL
- Basic statistics: distributions and their moments
- Hypothesis testing: *p*-value, statistical significance
- Statistical and Systematic errors
- Goodness of fit tests
- Visualizations
- Geospatial analysis
- Likelihood
- OLS
- Topics in (time) series analysis
- Clusters
- Decision and regression trees (CART)

Today:

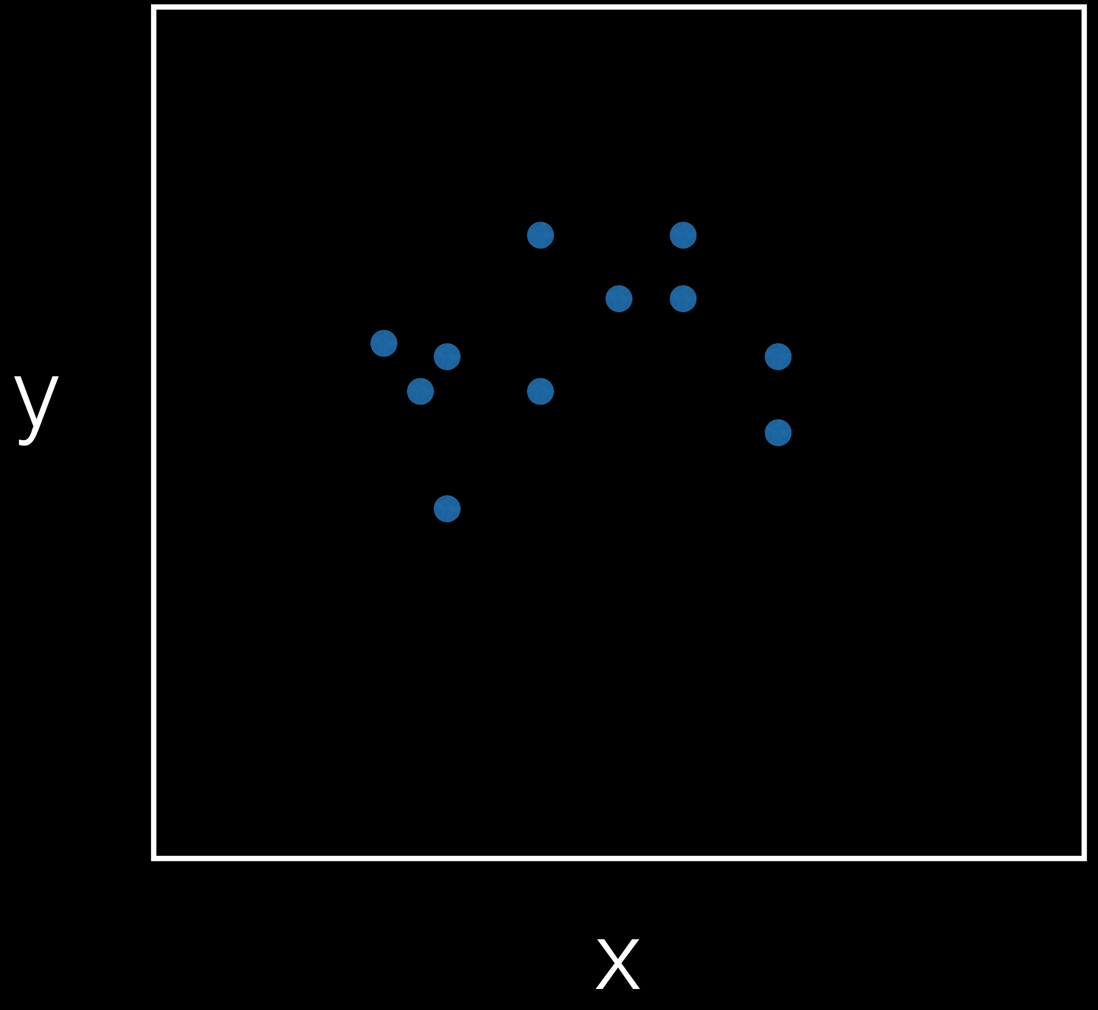
- categorical and mixed clustering
- model diagnostics (ROC, AUC)
- OSMnx
- tips on efficient coding

Distances

Partitioning methods: clustering

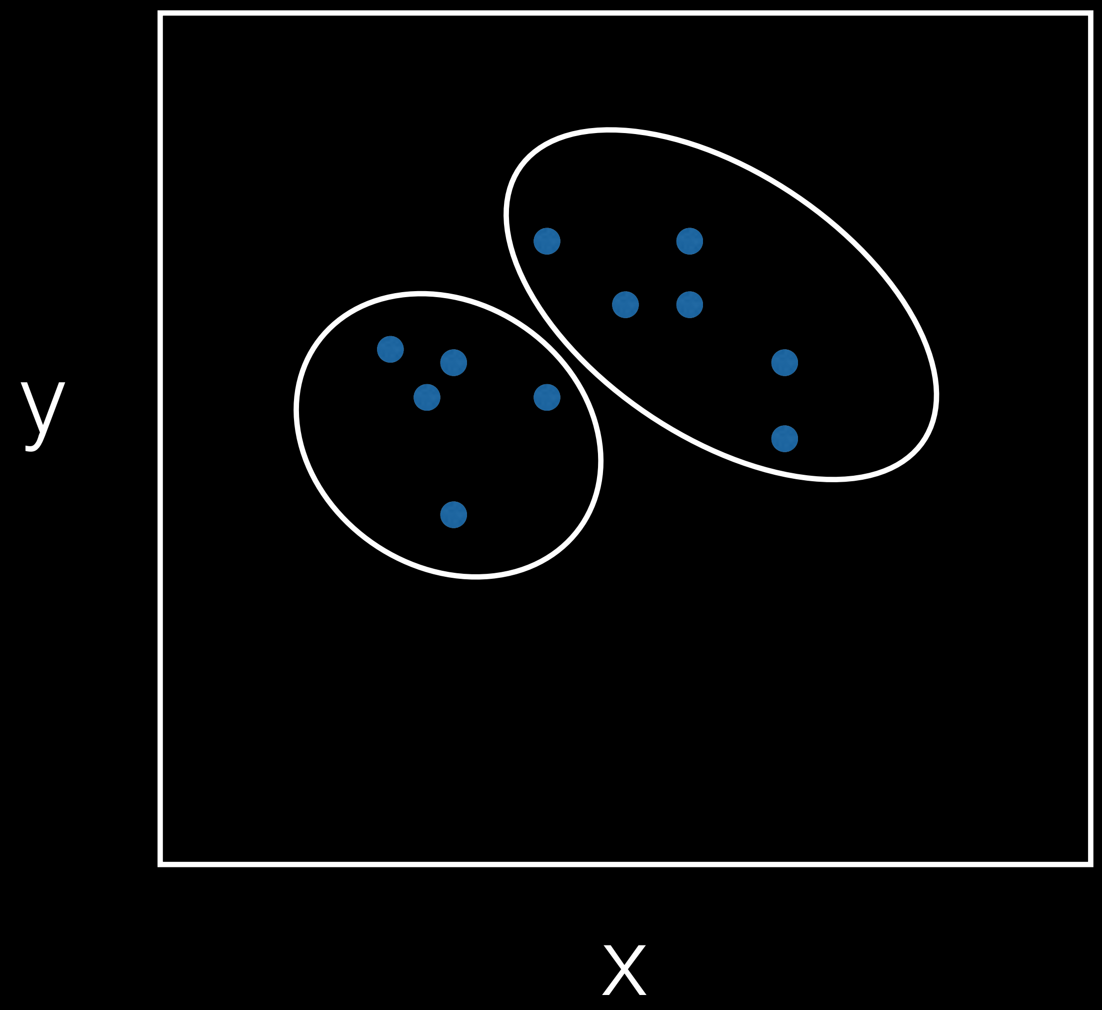
goal is to partition the space so that the observerd variables are separate in maximally homogeneous groups

observed:
(x, y)



Partitioning methods: clustering

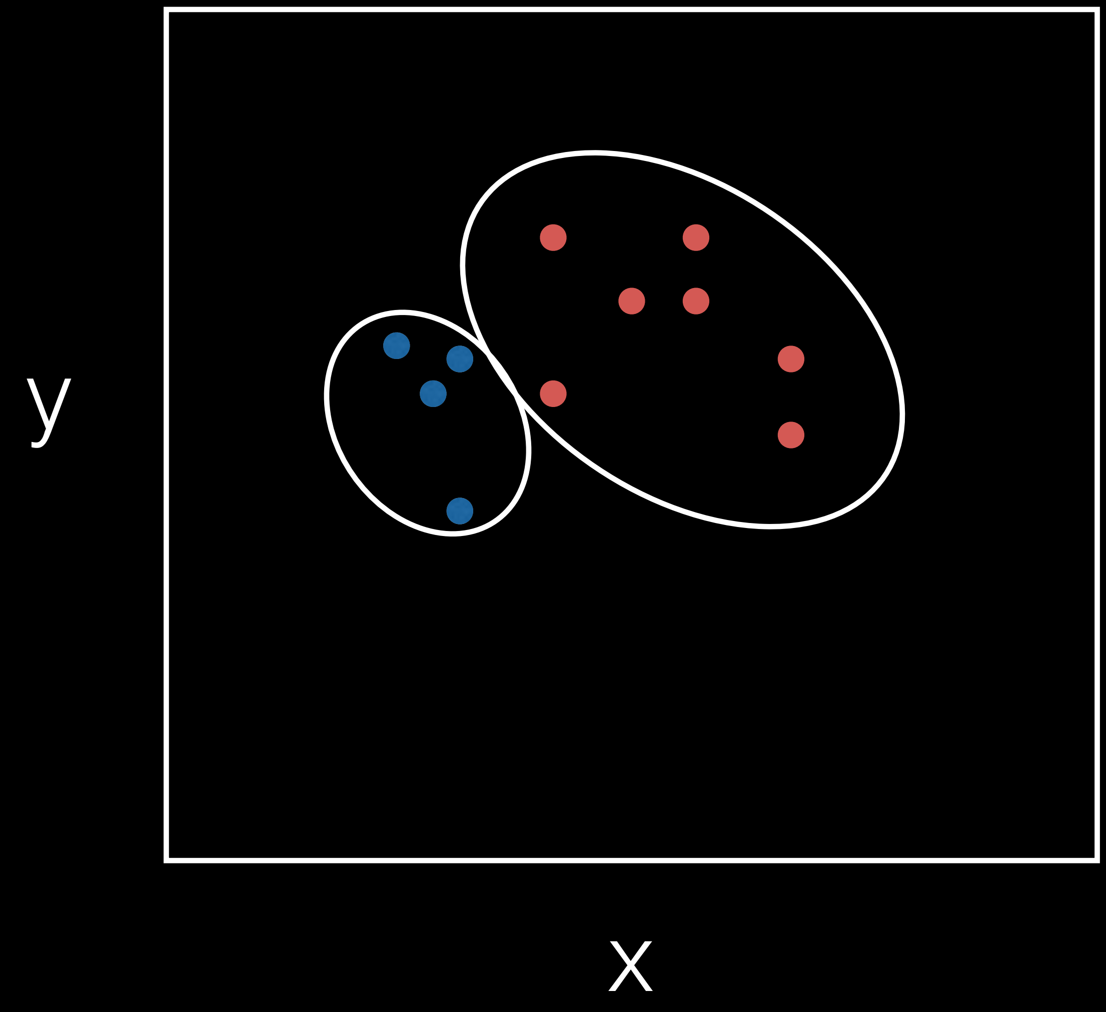
observed:
(x, y)



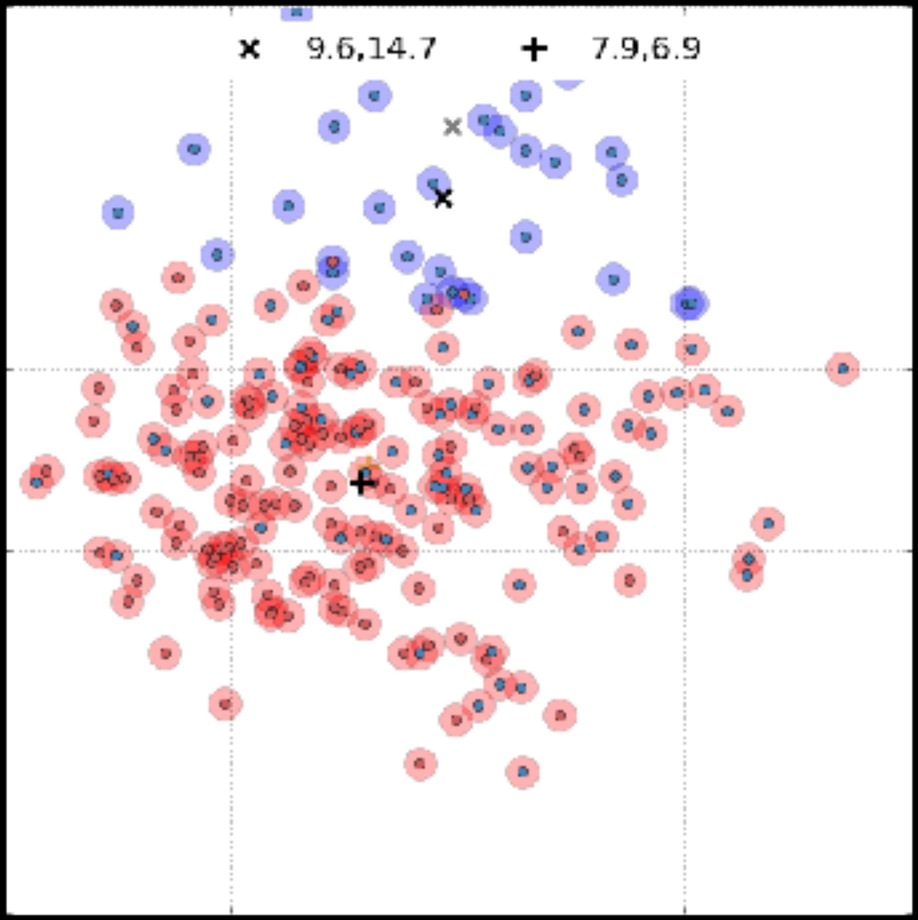
XII categorical distances
model diagnostics, OSMnx

Partitioning methods: clustering

observed:
(x, y, color)



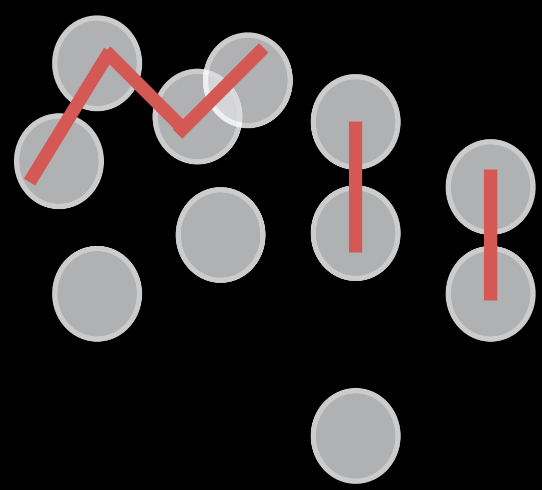
Crisp (or hard) clustering - K-means

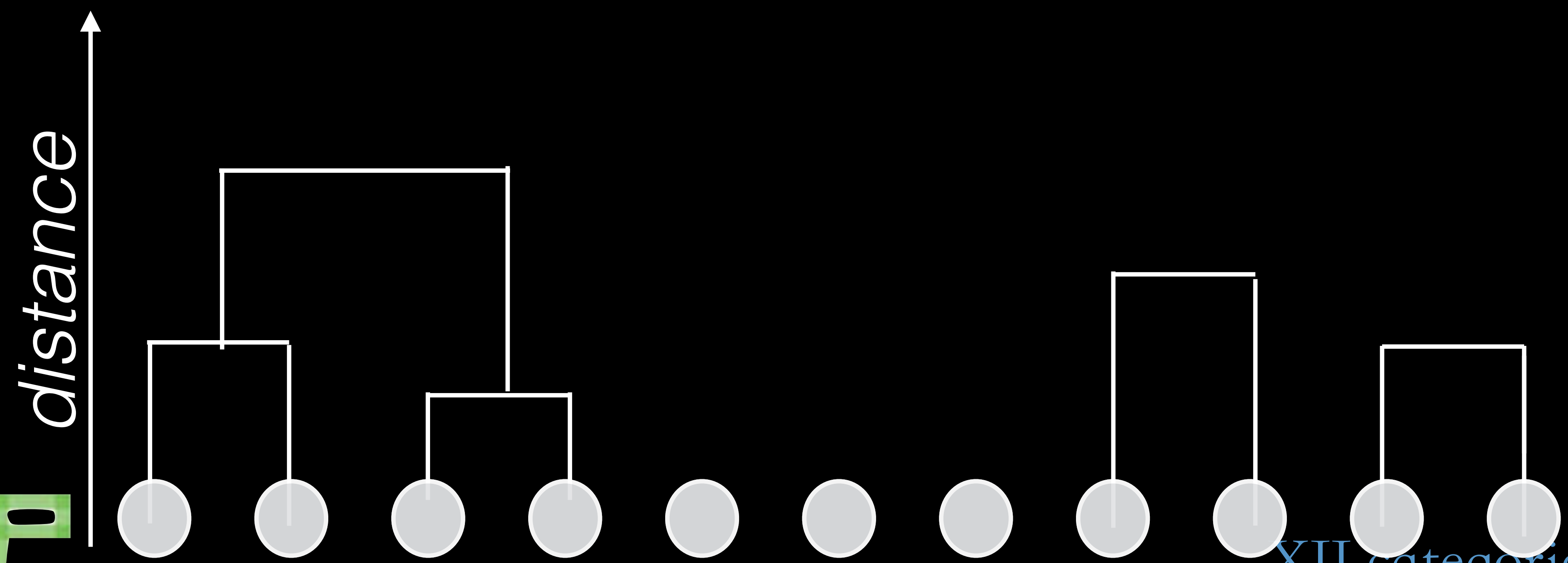


You guess the centers and assign points to clusters based on a predefined distance metric

hierarchical clustering

agglomerative
bottom-up





Summary and Key concepts

clustering is easy, but interpreting results is tricky

Distance metrics:

- Eucledian and other Minchowski metrics
- geospacial distances
- metrics for non continuous data

Partitioning methods: inexpensive, typically non deterministic

- Hard methods: *K-means, K-medoids*
- Soft (or fuzzy) methods: (i.e. probabilistic approach)
  - *Expectation Maximization Mixture models*

Hierarchical methods:

- divisive vs agglomerative, dendrograms

Distance Metrics Continuous variables

Minkowski family of distances

$$D(i,j) = \sqrt[p]{\sum_{k=1}^{N} |x_{ik} - x_{jk}|^p}$$

N features (dimensions)

Great Circle distances: $\phi_i, \lambda_i, \phi_j, \lambda_j$

geographical latitude and longitude

$$D(i,j) = R \arccos(\sin\phi_i \cdot \sin\phi_j + \cos\phi_i \cdot \cos\phi_j \cdot \cos(\Delta\lambda))$$

XII categorical distances
model diagnostics, OSMnx

Distance Metrics

Binary variables

contingency table

	1	0	*sum*
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

Distance Metrics

Binary variables

contingency table

	1	0	sum
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

e.g.: subway station
w ESCALATOR Y/N
w ELEVATOR Y/N

Distance Metrics

Binary variables

contingency table

	1	0	*sum*
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

e.g.: subway station
w ESCALATOR Y/N
w ELEVATOR Y/N

ELEVATOR

ESCALATOR	1	0	
1	7	3	
0	106	353	

Distance Metrics

Binary variables

contingency table

	1	0	sum
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

e.g.: subway station
w ESCALATOR Y/N
w ELEVATOR Y/N

ESCALATOR \ ELEVATOR	1	0	sum
1	7	3	10
0	106	353	459
sum	113	356	469

Distance Metrics

Binary variables

contingency table

	1	0	sum
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

e.g.: subway station
w ESCALATOR Y/N
w ELEVATOR Y/N

ELEVATOR

ESCALATOR	1	0	sum
1	7	3	10
0	106	353	459
sum	113	356	469

IF SYMMETRIC
(same chance to appear
i.e. roughly same total
Y and N)

$$D_{ij} = \frac{b+c}{a+b+c+d} = \frac{109}{469} = 0.23$$

Distance Metrics **Binary variables**

contingency table

	1	0	sum
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

e.g.: subway station
w ESCALATOR Y/N
w ELEVATOR Y/N

ELEVATOR

ESCALATOR	1	0	sum
1	7	3	10
0	106	353	459
sum	113	356	469

IF SYMMETRIC
(same chance to appear
i.e. roughly same total
Y and N)

$$D_{ij} = \frac{M_{i=0j=0} + M_{i=1j=1}}{M_{00}+M_{01}+M_{10}+M_{11}} = \frac{109}{469} = 0.23$$

XII categorical distances
model diagnostics, OSMnx

Distance Metrics

Binary variables

contingency table

	1	0	sum
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

e.g.: subway station
w ESCALATOR Y/N
w ELEVATOR Y/N

ELEVATOR

ESCALATOR

	1	0	sum
1	7	3	10
0	106	353	459
sum	113	356	469

IF ASYMMETRIC
(not same chance)

$$D_{ij} = \frac{b+c}{a+b+c} = \frac{109}{116} = 0.94$$

Distance Metrics

Binary variables

contingency table

	1	0	sum
1	a	b	$a+b$
0	c	d	$c+d$
sum	$a+c$	$b+d$	p

e.g.: subway station
w ESCALATOR Y/N
w ELEVATOR Y/N

ELEVATOR

ESCALATOR	1	0	sum
1	7	3	10
0	106	353	459
sum	113	356	469

IF ASYMMETRIC
(not same chance)

Jaccard similarity

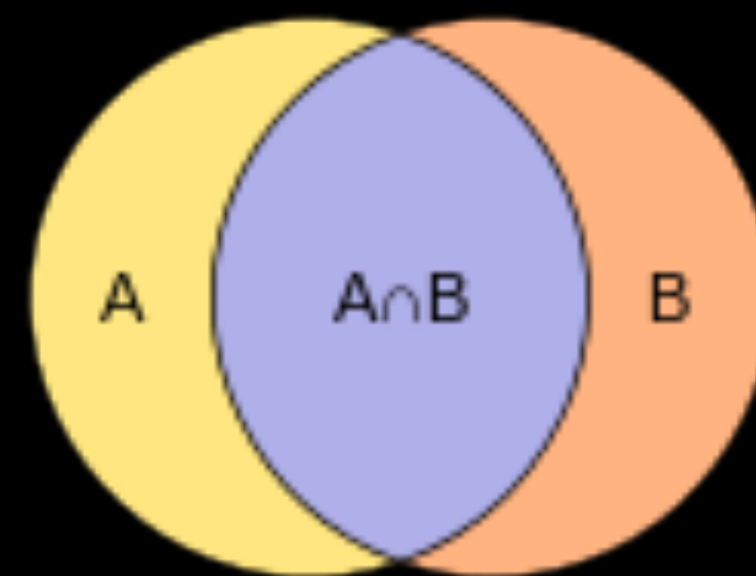
$$J_{ij} = \frac{a}{a+b+c} = \frac{7}{116} = 0.06$$

Distance Metrics Binary variables

Uses presence/absence data

Jaccard similarity coefficient S_j

$$S_j = \frac{a}{a+b+c}$$



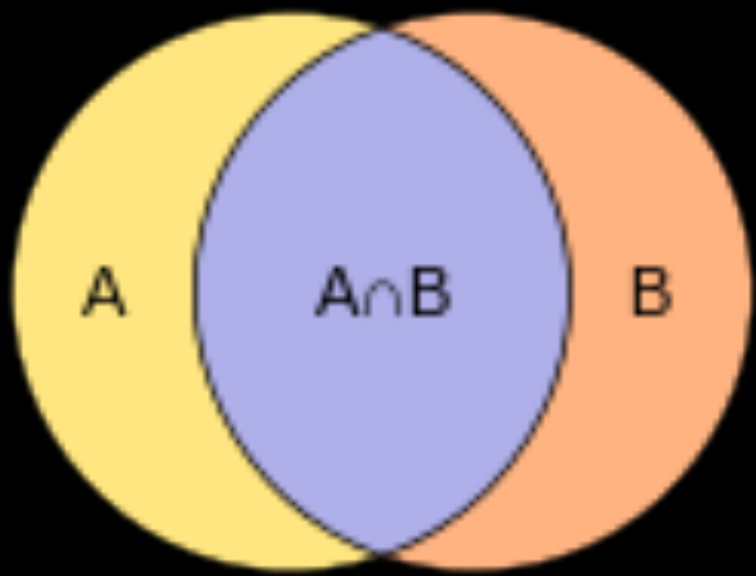
a = number of items in common,
b = number of items unique to the first set
c = number of items unique to the second set

Distance Metrics   Binary variables

Uses presence/absence data

Jaccard similarity coefficient S_j

$$S_j = \frac{A \bigcap B}{A \bigcup B}$$



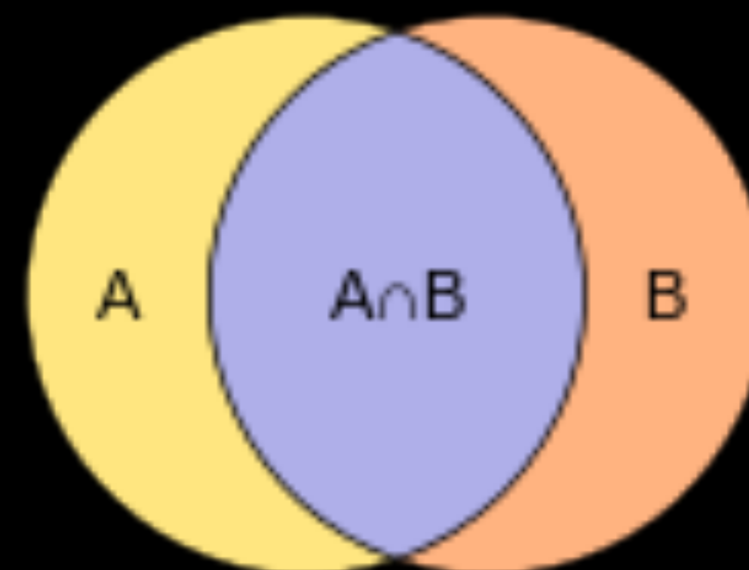
a = number of items in common,
b = number of items unique to the first set
c = number of items unique to the second set

Distance Metrics   Binary variables

Uses presence/absence data

Jaccard distance
$D_j = 1 - S_j$

$$S_j = \frac{A \cap B}{A \cup B}$$



a = number of items in common,
b = number of items unique to the first set
c = number of items unique to the second set

Distance Metrics Categorical Variables

Uses presence/absence data in two samples (non exclusive)

Simple similarity coefficient

Simple Matching Method

SMC

$$S_{ij} = \frac{p-m}{p}$$

p: number of variables

m: number of matches

jupyter

https://github.com/fedhere/UInotebooks/blob/master/cluster/categorical_clustering.ipynb

Distance Metrics   Ordinal variables

Uses ranks

map occurrences in a range 0-1

$r_{ij} = \{1 \ldots R_N\}$ -> $\mathbf{z_{ij} = \frac{r_{ij} - 1}{R_N - 1}}$

Distance Metrics MIXED variables

Hybrid dataset containing continuous, ordinal, categorical

weighted distance

$$D_w = \frac{\sum_{p=1}^{p} w_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{p=1}^{p} w_{ij}^{(f)}}$$

Distance Metrics vector Variables

Uses correlation coefficient!

A time series is a vector:
MTA rides/NYC establishments can be clustered w this distance
clustering time series + other features requires this

Pearson's correlation

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

Cosine similarity

$$\cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$

Model Diagnostics

Accuracy, Recall, Precision

Accuracy, Recall, Precision

$$LR = \frac{\text{False Negative}}{\text{True Negative}}$$

	H_0 is True	H_0 is False
H_0 is falsified	Type I error False Positive important message gets spammed	True Positive
H_0 is not falsified	True Negative	Type II error False negative Spam in your Inbox

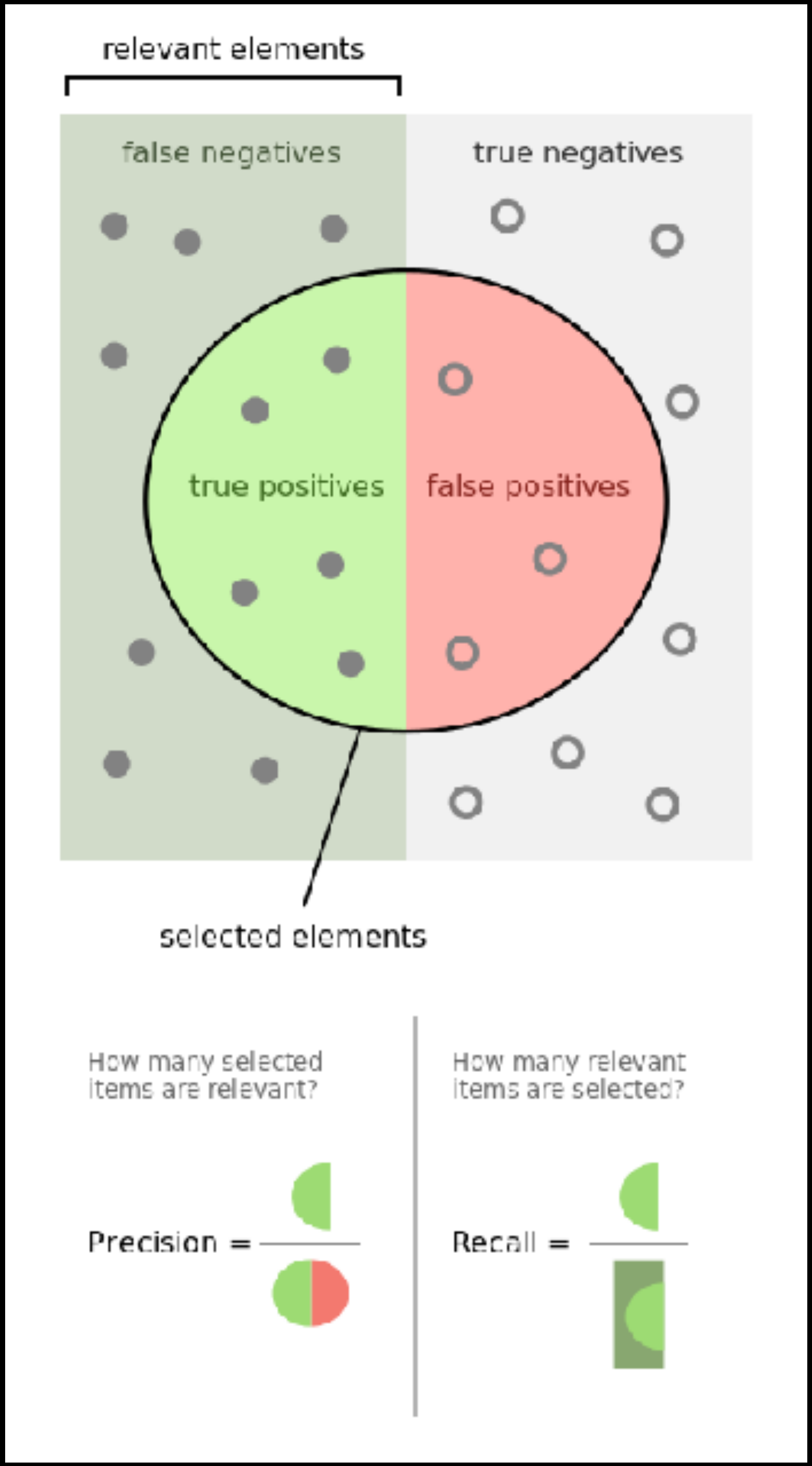
Accuracy, Recall, Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

TP = True positives
FP = False positives
TN = True negatives
FN = False negatives



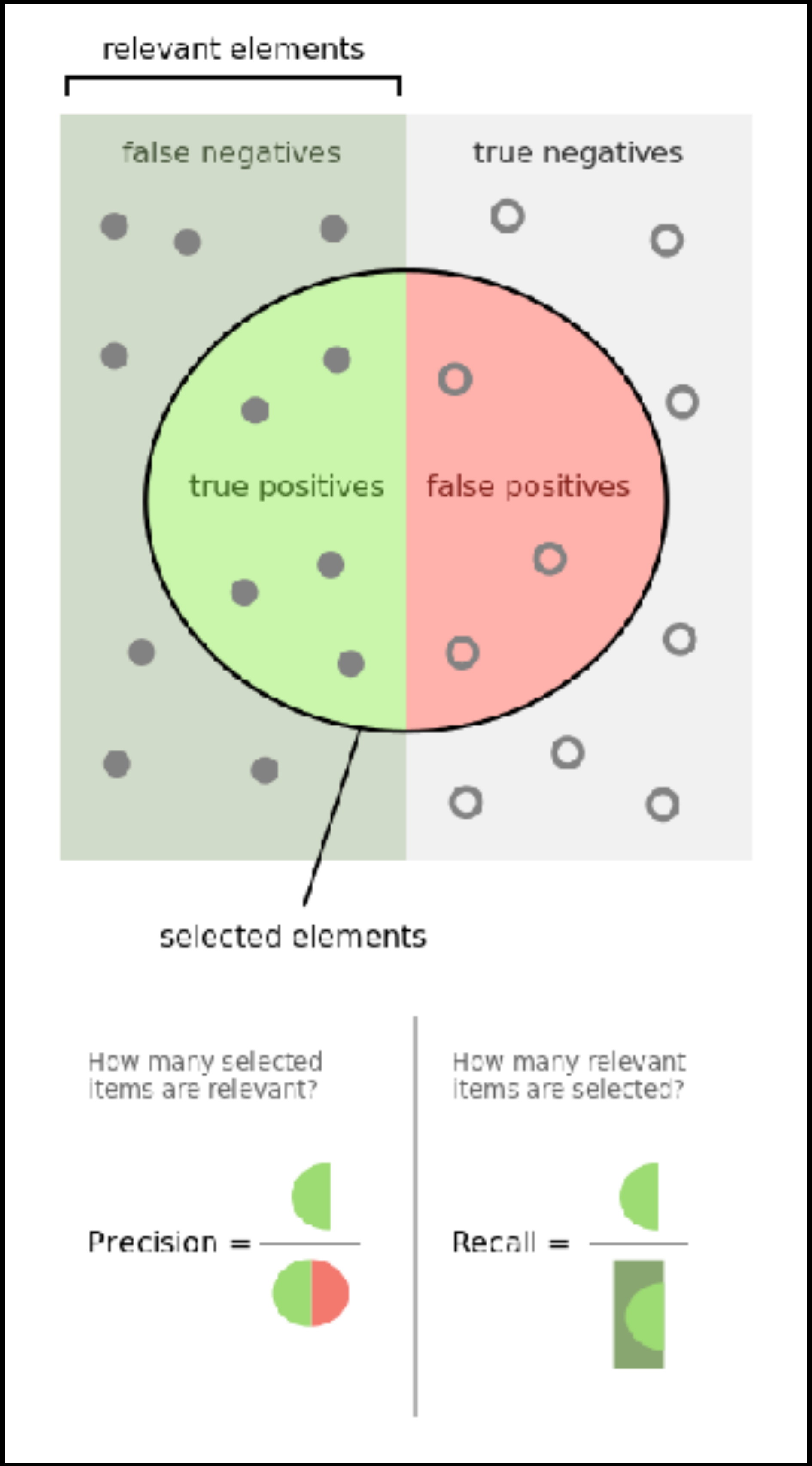
Accuracy, Recall, Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

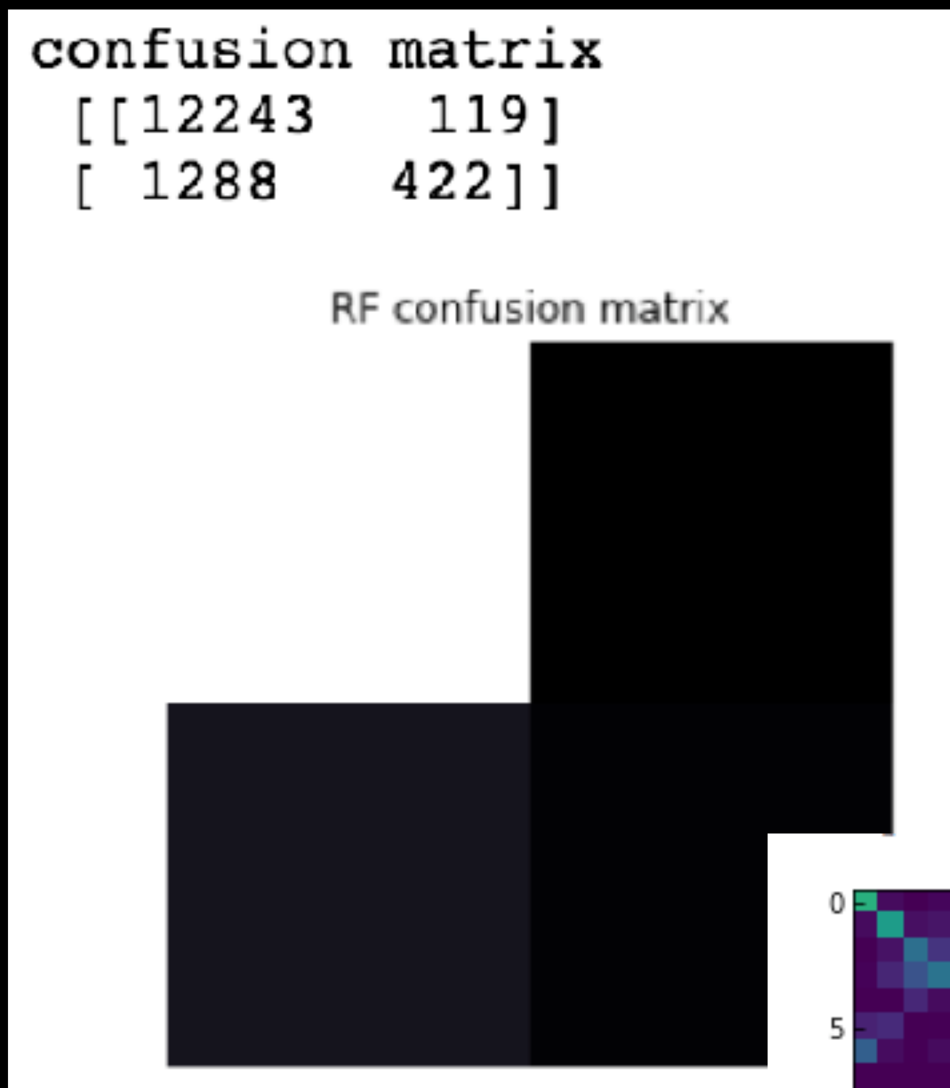
$$\text{Recall} = \frac{TP}{TP + FN}$$

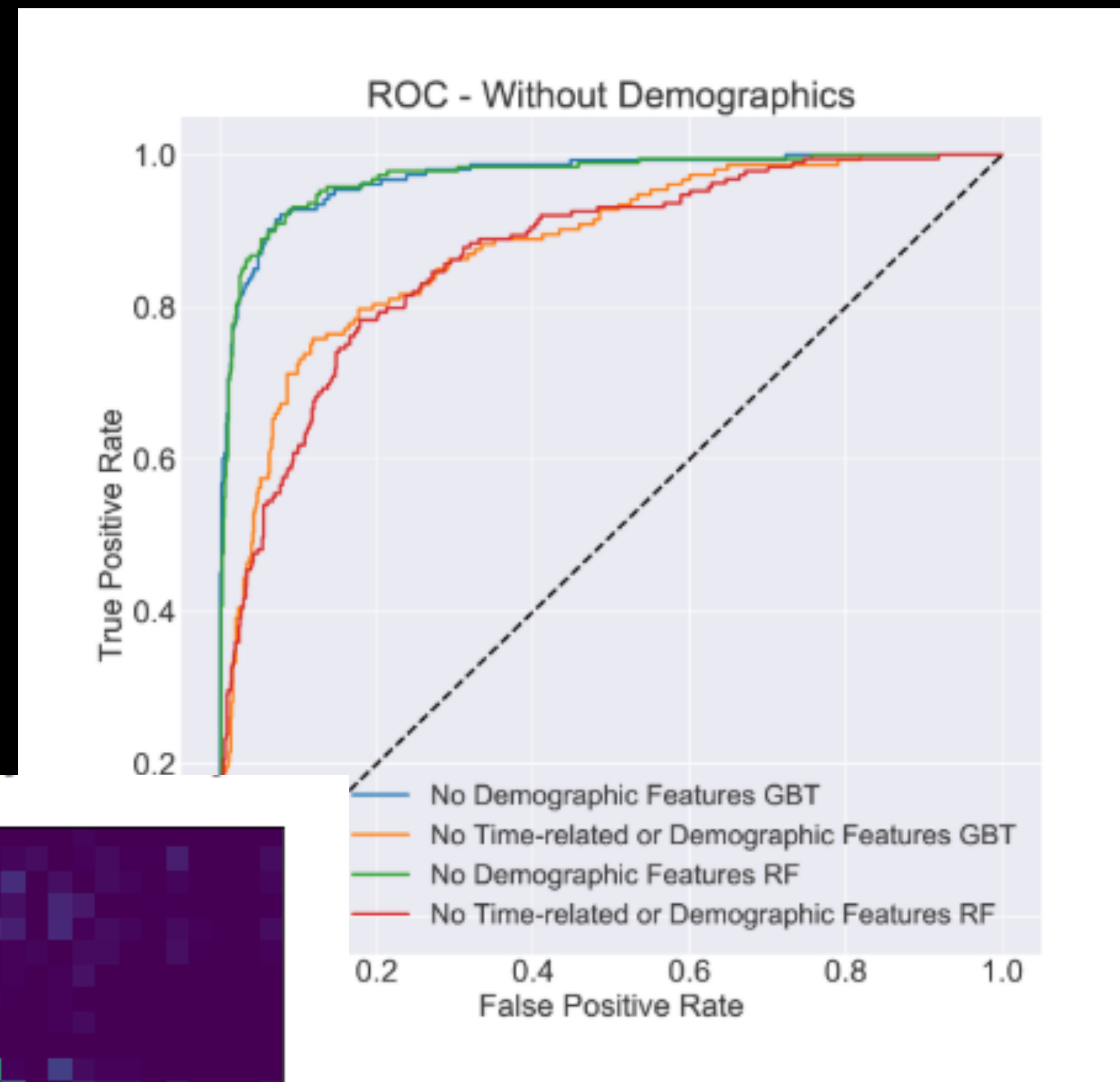
$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

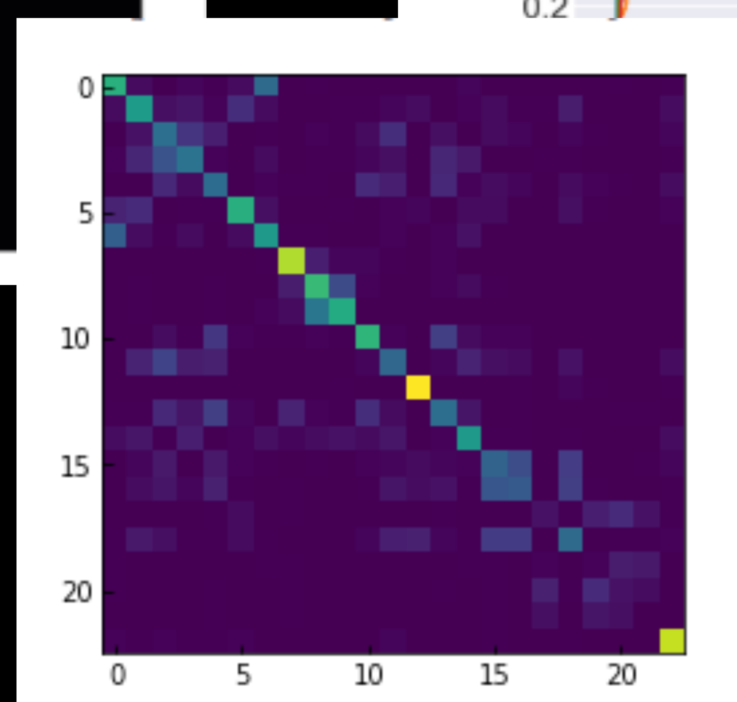
TP = True positives
FP = False positives
TN = True negatives
FN = False negatives



ROC AUC and confusion matrices







XII categorical distances
model diagnostics, OSMnx

Data Preprocessing

Full on whitening

input data, PLUTO, manhattan (42,000x15)

	Borough	Block	Lot	CD	CT2010	CB2010	SchoolDist	Council	ZipCode	FireComp	...	TaxMap	EDesigNum	APPBBL	APPDate	PLUTOMapID	Ver
0	MN	1545	52	108	138	4000	02	5	10028	E022	...	10515	None	0.000000e+00	None	1	
1	MN	723	7501	104	93	6000	02	3	10001	E003	...	10302	None	1.007230e+09	11/30/2006	1	
2	MN	1680	48	111	170	5000	04	8	10029	E091	...	10605	None	0.000000e+00	None	1	
3	MN	1385	32	108	130	2003	02	4	10021	E039	...	10508	None	0.000000e+00	None	1	
4	MN	1197	27	107	169	5000	03	6	10024	E074	...	10408	None	0.000000e+00	None	1	

axis 0 ——>

axis 1 ——>

axis 0 : observations
axis 1 : features

Full on whitening

input data, PLUTO, manhattan (42,000x15)

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\ \mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)] & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix}.$$

matrix of expected values of data

Full on whitening

input data, PLUTO, manhattan (42,000x15)

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & 0 & \cdots & 0 \\ 0 & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix}.$$

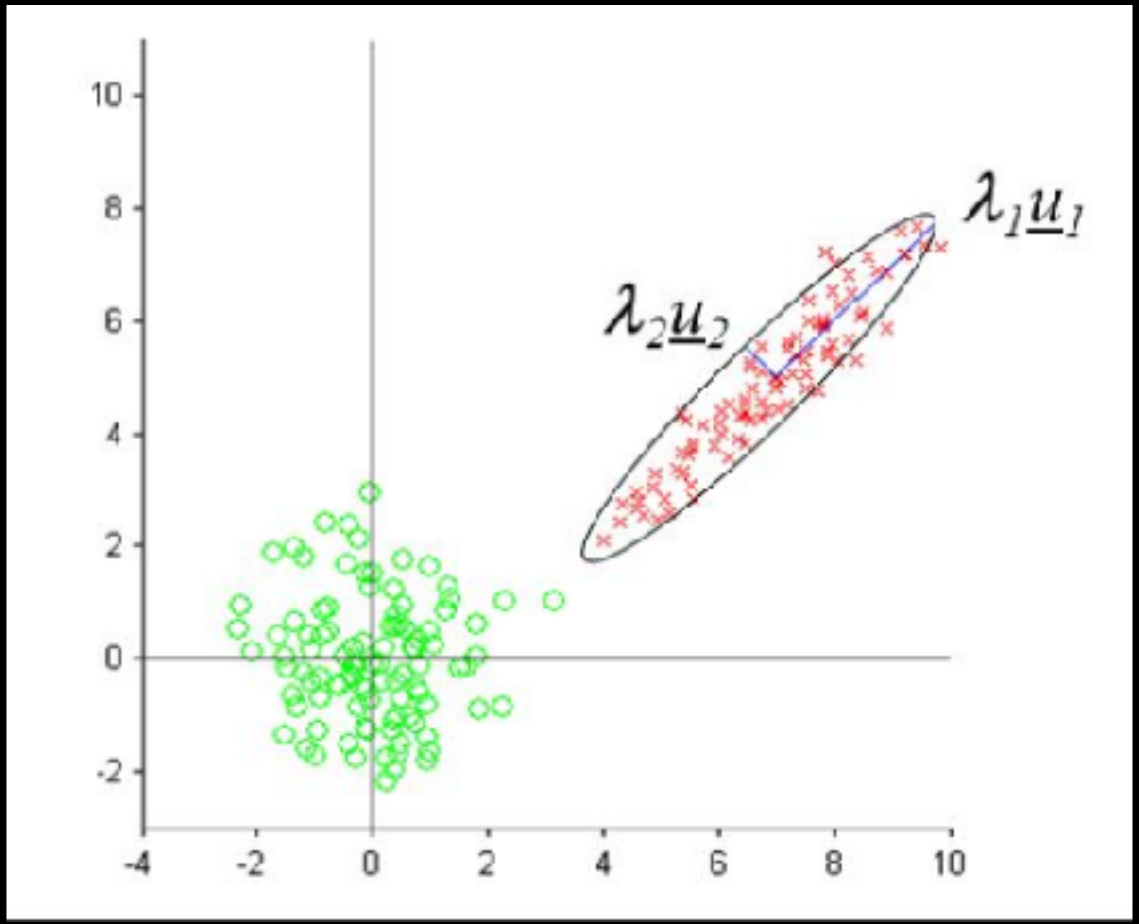
no covariance

Full on whitening

find the matrix W that diagonalized Σ

covariance

$\Sigma\, W$ = diagonal matrix



no covariance

Full on whitening

find the matrix W that diagonalized Σ

```
from zca import ZCA
import numpy as np
X = np.random.random((10000, 15)) # data array
trf = ZCA().fit(X)
X_whitened = trf.transform(X)
X_reconstructed =
trf.inverse_transform(X_whitened)
assert(np.allclose(X, X_reconstructed)) # True
```

Scaling - independent features

X = preprocessing.scale(X)

axis 0 ——>

	Borough	Block	Lot	CD	CT2010	CB2010	SchoolDist	Council	ZipCode	FireComp	...	TaxMap	EDesigNum	APPBBL	APPDate	PLUTOMapID	Ver
0	MN	1545	52	108	138	4000	02	5	10028	E022	...	10515	None	0.000000e+00	None	1	
1	MN	723	7501	104	93	6000	02	3	10001	E003	...	10302	None	1.007230e+09	11/30/2006	1	
2	MN	1680	48	111	170	5000	04	8	10029	E091	...	10605	None	0.000000e+00	None	1	
3	MN	1385	32	108	130	2003	02	4	10021	E039	...	10508	None	0.000000e+00	None	1	
4	MN	1197	27	107	169	5000	03	6	10024	E074	...	10408	None	0.000000e+00	None	1	

axis 0 ——> mean = 0, stdev = 1

```
X = preprocessing.scale(X, axis=0)
```

Last executed 2018-12-12 09:35:39 in 46ms

```
X.mean(axis=0)
```

Last executed 2018-12-12 09:35:40 in 13ms

```
array([ 3.85590369e-16, -6.93196168e-17, -5.90549813e-16, -5.95882091e-16,
       -8.49165306e-16, -1.57568821e-15, -8.00508267e-16,  5.55890004e-16,
       -5.16564452e-16,  1.09378357e-15,  3.46598084e-16,  2.31954102e-16,
        2.78611537e-16, -2.51283611e-16,  8.66495210e-18,  3.03939858e-16,
       -3.66594127e-17, -9.27149875e-16, -6.39873386e-16,  2.93275302e-17,
        9.19817992e-17,  6.33208038e-18, -1.99960433e-17,  9.55144336e-16,
       -2.20623011e-16,  6.93196168e-17, -9.46479383e-17,  2.26621824e-16,
        6.93196168e-17,  2.32953905e-16])
```

HW11

XII categorical distances

nodel diagnostics, OSMnx

Scaling - independent features

X = preprocessing.scale(X)

axis 0 ——>

	Borough	Block	Lot	CD	CT2010	CB2010	SchoolDist	Council	ZipCode	FireComp	...	TaxMap	EDesigNum	APPBBL	APPDate	PLUTOMapID	Ver
0	MN	1545	52	108	138	4000	02	5	10028	E022	...	10515	None	0.000000e+00	None	1	
1	MN	723	7501	104	93	6000	02	3	10001	E003	...	10302	None	1.007230e+09	11/30/2006	1	
2	MN	1680	48	111	170	5000	04	8	10029	E091	...	10605	None	0.000000e+00	None	1	
3	MN	1385	32	108	130	2003	02	4	10021	E039	...	10508	None	0.000000e+00	None	1	
4	MN	1197	27	107	169	5000	03	6	10024	E074	...	10408	None	0.000000e+00	None	1	

axis 0 ——> mean = 0, stdev = 1

```
X = preprocessing.scale(X, axis=0)
```

Last executed 2018-12-12 09:35:39 in 46ms

```
X.std(axis=0)
```

Last executed 2018-12-12 09:36:28 in 19ms

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

HW11

Scaling - feature vectors: *e.g. time series*

X = preprocessing.scale(X, axis=1)

	Borough	Block	Lot	CD	CT2010	CB2010	SchoolDist	Council	ZipCode	FireComp	...	TaxMap	EDesigNum	APPBBL	APPDate	PLUTOMapID	Ver
0	MN	1545	52	108	138	4000	02	5	10028	E022	...	10515	None	0.000000e+00	None	1	
1	MN	723	7501	104	93	6000	02	3	10001	E003	...	10302	None	1.007230e+09	11/30/2006	1	
2	MN	1680	48	111	170	5000	04	8	10029	E091	...	10605	None	0.000000e+00	None	1	
3	MN	1385	32	108	130	2003	02	4	10021	E039	...	10508	None	0.000000e+00	None	1	
4	MN	1197	27	107	169	5000	03	6	10024	E074	...	10408	None	0.000000e+00	None	1	

axis 1 ——> mean = 0, stdev = 1

Build one that uses as input features the following engineered features :

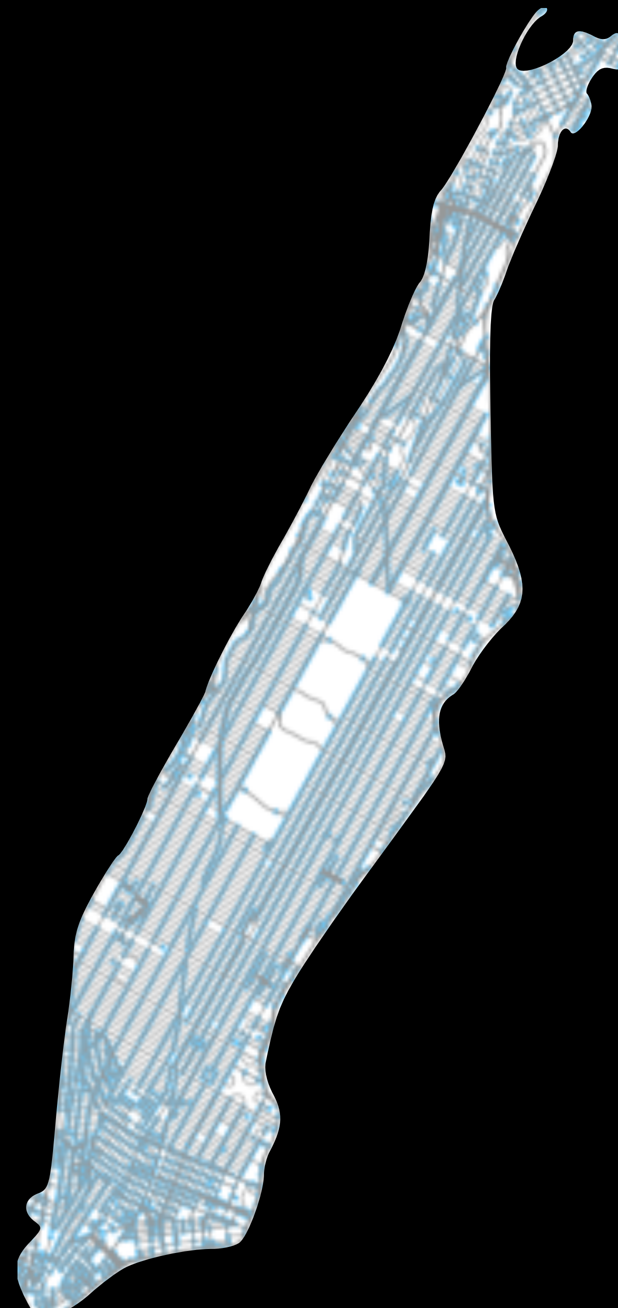
- the time series mean divided by the mean of all time series for that station
- the time series standard deviation by the standard deviation of all time series for that station
- the slope and intercept of a line fit to the standardized time series

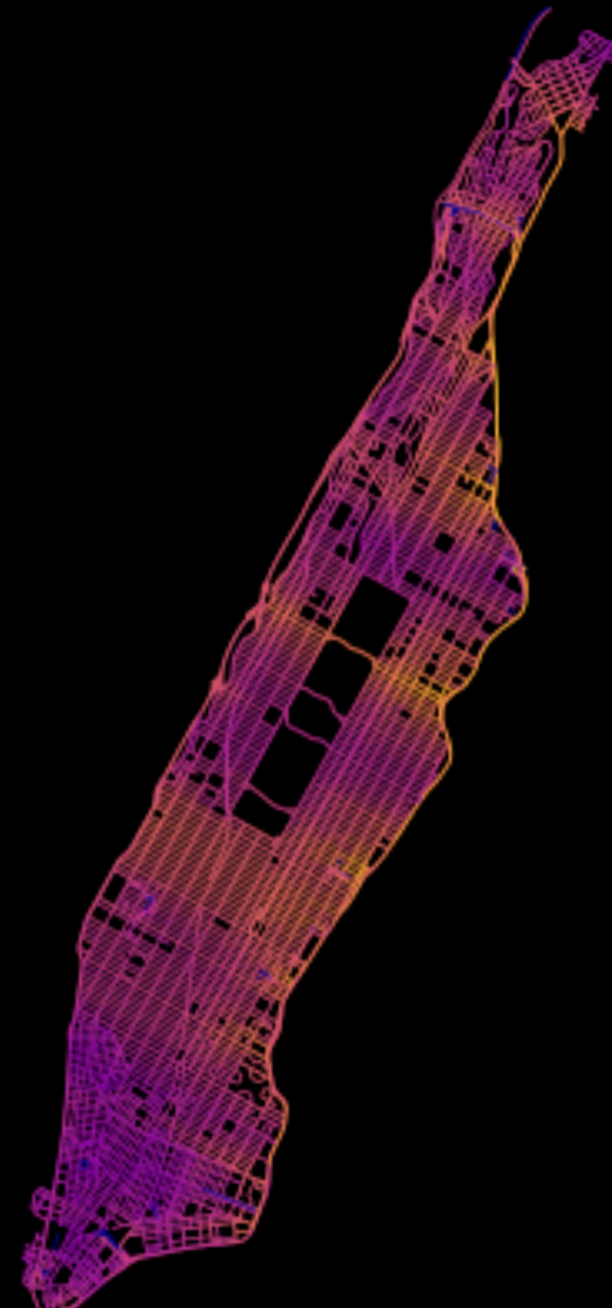
```
(time_series - time_series.mean())/time_series.std()
```

you will have to remove time series containing NaN because the random forest sklearn implementation does not work with NaNs. An easy way to do that is to remove all time series whose standard deviation is NaN

super important missing topic:
pruning!
when is my tree overfitting?

OSMnx





https://github.com/gboeing/osmnx

https://github.com/gboeing/osmnx-examples/

OSMnx

is your code optimized:

check CPU AND MEMORY usage

vectorize (slice and avoid for loops)

avoid storing information you do not need in memory

use local variables

remove all redundant calculations from inside loops

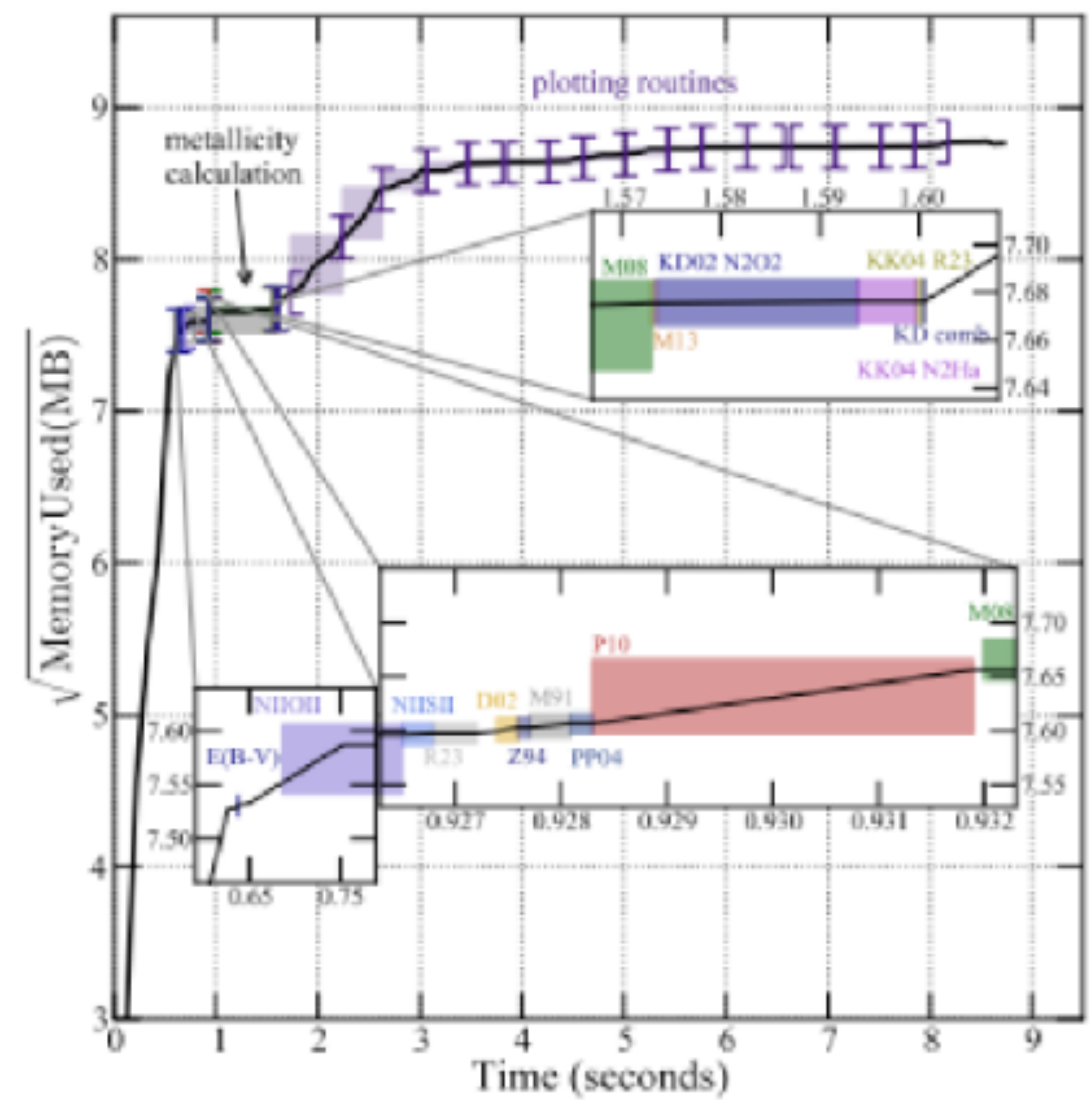


FIG. 6.— Memory usage: we plot the square root of the memory usage in Megabytes as a function of time for running our code (using N=2,000 and all default metallicity scales except the D13 *pyqz* ones) on a single set of measured emission lines (Table 2, host galaxy of SN 2008D). The square root is plotted, instead of the natural value, to enhance visibility. Three inserts show the regions where most of the metallicity scales are calculated, zoomed in, since the run time of the code is dominated by plotting routines, including the calculation of the bin size with Knuth's rule. Each function call is represented by an opening and closing bracket in the main plot, and by a shaded rectangle in the zoomed-in insets. The calculation of $N2O2$, which requires 0.25 seconds, is split be-

model diagnostics, OSMnx

Reading:

An excellent use of viz for data exploration and transition to inferential analysis

https://blog.data.gov.sg/how-we-caught-the-circle-line-rogue-train-with-data-79405c86ab6a#.iz1r655xo

Lee Shangqian, Daniel Sim & Clarence Ng

Distance measures for clustering:

http://sfb649.wiwi.hu-berlin.de/fedc_homepage/xplore/tutorials/mvahtmlnode79.html

Decision trees:

http://what-when-how.com/artificial-intelligence/decision-tree-applications-for-data-modelling-artificial-intelligence/

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4380222/

Efficient python coding:

https://wiki.python.org/moin/PythonSpeed/PerformanceTips