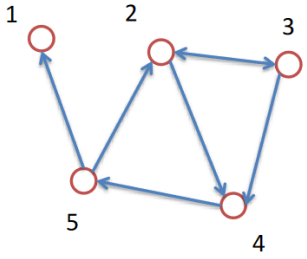
#Adjancy matrix
https://www.google.com/url?sa=i&url=https%3A%2F%2Fmathworld.wolfram.com%2FAdjacencyMatrix.html&psig=AOvVaw2QbLkuU5jRo7RRFouCmrPL&ust=1644271467056000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCNimwNiK7PUCFQAAAAAdAAAAABAD

Example – Adjacency Matrix



Indegree

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Outdegree

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Ο όρος NoSQL

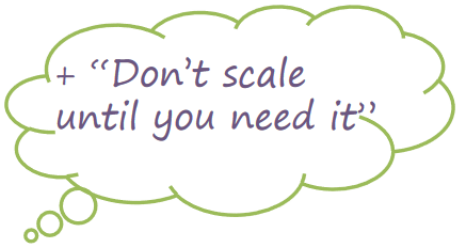
- **Μη-Σχεσιακά Συστήματα Διαχείρισης ΒΔ (ΣΔΒΔ)**
  - "SQL" = Παραδοσιακά σχεσιακά ΣΔΒΔ
- Πρόκειται συχνά για συστήματα **χωρίς σχήμα (schema-less)**, που αποφεύγουν τις συνενώσεις και είναι πολύ εύκολο να κλιμακωθούν
  - "NoSQL" = "No SQL" = Χωρίς χρήση παραδοσιακών σχεσιακών ΣΔΒΔ
- Ο όρος **NoSQL** επινοήθηκε το 1998 από τον Carlo Strozzi και επανεμφανίστηκε στις αρχές του 2009 με το συνέδριο no:sql(east)
- Ένας καλύτερος όρος θα ήταν "NoREL", αλλά…
  - "No SQL" ≠ "Don't use the SQL language"
  - **"NoSQL" = "Not Only SQL"**

Συστήματα NoSQL – Πλεονεκτήματα

- ☑ Εναλλακτική στα παραδοσιακά σχεσιακά ΣΔΒΔ
- ☑ Χωρίς σχήμα (**Schema-less**)
- ☑ Ταιριαστή επιλογή για πολλές **εφαρμογές Web 2.0**
- ☑ Τεράστιες αποθήκες δεδομένων
- ☑ Γρηγορότερη/φθηνότερη εγκατάσταση/στήσιμο
- ☑ Κάποιες υπηρεσίες είναι απλούστερο να υλοποιηθούν σε σχέση με τα ΣΔΒΔ
- ☑ Πολύ μεγάλη **επεκτασιμότητα** (δυνατότητα κλιμάκωσης)
- ☑ Χαλαρή συνοχή → **υψηλότερη απόδοση και διαθεσιμότητα**

Διαπανεπιστημιακό Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών

Συστήματα NoSQL – Μειονεκτήματα



- ☒ Οι υλοποιήσεις NoSQL συστημάτων είναι νεότερες
  … ενώ τα σχεσιακά ΣΔΒΔ και τα σχετικά εργαλεία είναι «ώριμα»
- ☒ Χαλαρή συνοχή δεδομένων → **λιγότερες εγγυήσεις**
- ☒ (Συχνά) **απουσία της έννοιας των συναλλαγών** (transactions)
- ☒ Απουσία declarative γλώσσας ερωτημάτων → **περισσότερος προγραμματισμός**

declarative programming: προγραμματίζουμε τη λογική παρά τον αλγόριθμο,
π.χ. SQL, HTML

Σχεσιακές & NoSQL ΒΔ vs. CAP

- Οι **σχεσιακές ΒΔ** δίνουν έμφαση στη **Συνοχή**, με αποτέλεσμα να πάσχουν είτε η Διαθεσιμότητα είτε η Ανοχή Κατακερματισμού

- Οι **NoSQL ΒΔ** δίνουν έμφαση στη **Διαθεσιμότητα** και την **Ανοχή Κατακερματισμού**
  - ***«Συνοχή εν τέλει» (Eventual Consistency)***
    - Οι αναζητήσεις στο Google δεν χρειάζεται να περιλαμβάνουν έγγραφα που δημιουργήθηκαν τα τελευταία δευτερόλεπτα
    - Το News Feed του Facebook δεν χρειάζεται να περιλαμβάνει updates που έγιναν τα τελευταία δευτερόλεπτα

Σχεσιακά ΣΔΒΔ Vs. NoSQL

Σχεσιακά ΣΔΒΔ

- Ισχυρή συνοχή
- Μεγάλα σύνολα δεδομένων
- Δυνατότητα κλιμάκωσης
- Καλή διαθεσιμότητα

Συστήματα NoSQL

- Συνοχή «εν τέλει»
- ΤΕΡΑΣΤΙΑ σύνολα δεδομένων
- Εύκολη κλιμάκωση
- Πολύ υψηλή διαθεσιμότητα

p = 1 0 0 0 0 0 0 0 0 0

q = 0 0 0 0 0 0 1 0 0 1

M_{01} = 2 (the number of attributes where p was 0 and q was 1)
M_{10} = 1 (the number of attributes where p was 1 and q was 0)
M_{00} = 7 (the number of attributes where p was 0 and q was 0)
M_{11} = 0 (the number of attributes where p was 1 and q was 1)

$$J = (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

- Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

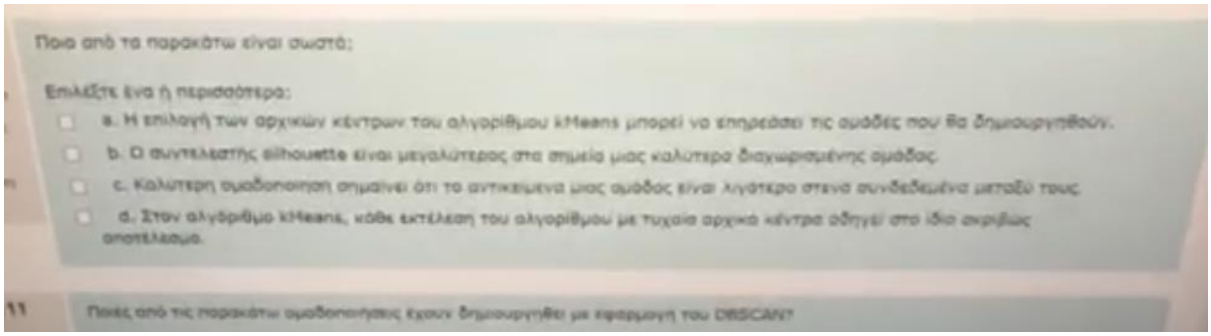
C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

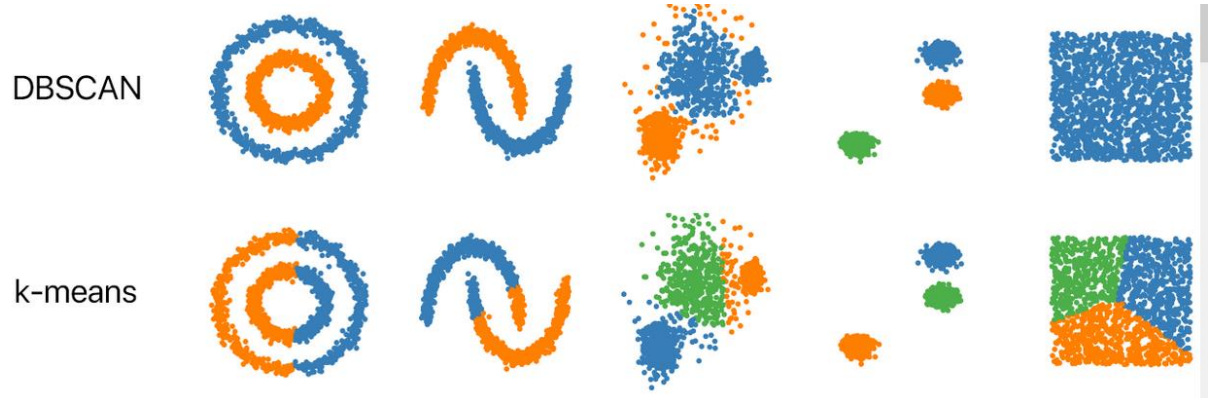
C1	**1**
C2	**5**

$P(C1) = 1/6 \qquad P(C2) = 5/6$

$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$



(a και ισως b)



τις παρακάτω εντολές:

```
> ConfusionMatrix(pred, ytest)
> Precision(ytest, pred)
> Recall(ytest, pred)
> F1_Score(ytest, pred)
```

Στις παραπάνω εντολές μπορούμε να επιλέξουμε την κλάση για τις οποίες υπολογίζονται οι μετρικές δίνοντάς τη ως τρίτη παράμετρο:

```
> Precision(ytest, pred, "class1")
```

```
pred = predict(model, xtest, type="class")
```

Μπορούμε επίσης να κάνουμε τους υπολογισμούς χρησιμοποιώντας την R. Αρχικά υπολογίζουμε την εντροπία για το σύνολο των δεδομένων:

```
> freq = prop.table(table(weather[, c(4)]))

> Entropy_All = - freq["No"] * log2(freq["No"]) - freq["Yes"] * log2(freq["Yes"])
```

Κατόπιν, για το Outlook κατασκευάζουμε τους παρακάτω πίνακες συχνοτήτων:

```
> absfreq = table(weather[, c(1, 4)])

> freq = prop.table(absfreq, 1)

> freqSum = rowSums(prop.table(absfreq))
```

Υπολογίζουμε την εντροπία για το Sunny και το Rainy:

```
> Entropy_Sunny = - freq["Sunny", "No"] * log2(freq["Sunny", "No"]) - freq["Sunny", "Yes"] * log2(freq["Sunny", "Yes"])
```

```
> Entropy_Rainy = - freq["Rainy", "No"] * log2(freq["Rainy", "No"]) - freq["Rainy", "Yes"] * log2(freq["Rainy", "Yes"])
```

Το συνολικό κέρδος πληροφορίας για το Outlook υπολογίζεται με την εντολή:

```
> GAIN_Outlook = Entropy_All - freqSum["Sunny"] * Entropy_Sunny - freqSum["Rainy"] * Entropy_Rainy
```

```
model = kmeans(kdata, centers = kdata[1:3,])
```

1.3 Hierarchical Clustering στην R

Για ομαδοποίηση με χρήση hierarchical clustering στην R, υπολογίζουμε αρχικά τον πίνακα αποστάσεων για τα δεδομένα:

```
> d = dist(data)
```

Στη συνέχεια εφαρμόζουμε hierarchical clustering μ την εντολή:

```
> hc = hclust(d, method = "single")
```

Με την παράμετρο method επιλέγουμε τον τρόπο ορισμού της απόστασης μεταξύ δύο clusters. Μπορούμε να επιλέξουμε την απόσταση μεταξύ των δύο κοντινότερων σημείων των clusters (`single`), την απόσταση μεταξύ των δύο μακρινότερων σημείων των clusters (`complete`), τη μέση απόσταση μεταξύ όλων των σημείων των clusters (`average`), την απόσταση μεταξύ των centroids των clusters (`centroid`), τη μείωση του τετραγωνικού σφάλματος που θα προκύψει αν συνενωθούν τα clusters (`ward.D2`) κ.α.

Μπορούμε να σχεδιάσουμε το δενδρόγραμμα με την εντολή:

```
> plot(hc)
```

Μπορούμε επίσης να εμφανίσουμε την κατανομή των σημείων σε clusters:

```
> clusters = cutree(hc, k = 4)
```

Όπου η παράμετρος k δηλώνει τον αριθμό των clusters.

Για να δούμε το διαχωρισμό των clusters στο δενδρόγραμμα (αφού έχουμε σχεδιάσει το δενδρόγραμμα με την εντολή `plot(hc)`) μπορούμε επιπλέον να εκτελέσουμε την εντολή:

```
> rect.hclust(hc, k = 4)
```