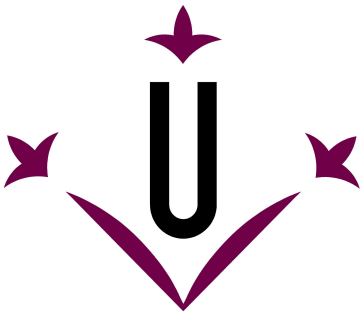
Escola Politècnica Superior

Grau en Enginyeria Informàtica

Automatic Reasoning and Learning



**Universitat de Lleida**

# Pràctica de Models Probabilístics:

## Indentificació de classes de vidres en l'escena del crim

**Data**: 25 de Juny, de 2018
**Professor**: Ramón Béjar Torres
**Nom:** Marc Melis Batalla **DNI:** 48257130W
**Nom:** Roger Truchero Visa **DNI:** 48056539V

---

# Objectius

L'objectiu del treball és realitzar aprenentatge basat en xarxes bayesianes, en lloc de basat en sistemes de regles. Es tracta del problema de classificació de diferents classes de vidres en funció de diferents propietats físiques dels mateixos. Aquest és el valor que té cada un dels 11 atributs que trobareu per cada instància (propietats d'un cristall correctament identificat) en el fitxer glass.data que es troba juntament amb la pràctica.

# Índex

# Índex de figures



# Índex de taules

# Part I
# Parseig de dades

Donat el problema ens hem trobat amb una sèrie de dades amb valors reals, però per tal de poder-los implementar dins d'una xarxa bayesiana necessitem discretitzar-los.

Per a fer-ho hem decidit utilitzar la llibreria *pandas*[1] de *python*. També hem afegit una primera línia al fitxer de les dades `glass.data.txt` amb l'identificador de columna donat per `glass.names.txt` per facilitar la lectura del document a *pandas* i no haver-ho d'afegir a posteriori i eliminem la columna d'identificadors perquè no ens aporta informació útil i *pandas* ja inclou els seus propis índexs.

| | Refractive_index | Na | Mg | Al | Si | K | Ca | Ba | Fe | Type |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 | 214.000000 |
| **mean** | 1.518365 | 13.407850 | 2.684533 | 1.444907 | 72.650935 | 0.497056 | 8.956963 | 0.175047 | 0.057009 | 2.780374 |
| **std** | 0.003037 | 0.816604 | 1.442408 | 0.499270 | 0.774546 | 0.652192 | 1.423153 | 0.497219 | 0.097439 | 2.103739 |
| **min** | 1.511150 | 10.730000 | 0.000000 | 0.290000 | 69.810000 | 0.000000 | 5.430000 | 0.000000 | 0.000000 | 1.000000 |
| **25%** | 1.516523 | 12.907500 | 2.115000 | 1.190000 | 72.280000 | 0.122500 | 8.240000 | 0.000000 | 0.000000 | 1.000000 |
| **50%** | 1.517680 | 13.300000 | 3.480000 | 1.360000 | 72.790000 | 0.555000 | 8.600000 | 0.000000 | 0.000000 | 2.000000 |
| **75%** | 1.519157 | 13.825000 | 3.600000 | 1.630000 | 73.087500 | 0.610000 | 9.172500 | 0.000000 | 0.100000 | 3.000000 |
| **max** | 1.533930 | 17.380000 | 4.490000 | 3.500000 | 75.410000 | 6.210000 | 16.190000 | 3.150000 | 0.510000 | 7.000000 |

Figura 1: Tendència central, dispersió i forma de la distribució del conjunt de dades

Per tal de discretitzar els valors hem utilitzat la funció `pandas.cut`, que retorna els valors en diferents intervals segons el número de particions que imposis per paràmetre. Un cop tenim els intervals obtenim un diccionari amb, per cada interval, el número natural que li correspon de 0 fins al número d'intervals − 1.

Un cop tenim els valors discretitzats afegim els camps necesaris pel format `.arff`. Afegim la relació i els atributs amb els seus rangs (tenin em compte que el camp `Type` no el discretitzarem ja que es un valor ja discret que va de 0 a 7. Finalment afegim les dades en format `csv` just a continuació i ja tenim el fitxer discretitzat i enllestit per passar-ho a *Weka*.

L'script que em utilitzat es trobat al fitxer `src/data_parsing.ipynb` com a una llibreta de *Jupyter Notebook*[2].

A l'hora de transformar-ho a `.arff` i descriure els atributs utilitzem els números de forma categòrica perquè d'aquesta manera aconseguim que un cristall amb 20% de sodi sigui completament diferent d'un de 30% (En cas que el nombre d'intervals sigui <10). Aquest comportament es podria canviar si féssim que els valors deixessin de ser categòrics per passar a ser numèrics i que la correlació entre dos valors, un cop discretitzats, siguin major com més pròxims es trobin.

[1] `https://pandas.pydata.org`
[2] `http://jupyter.org`

# Part II
# Aprenentatge de models amb K2

Abans de començar a apendre els models, haurem de carregar el fitxer *.arff* que conté les dades dels materials del crim.

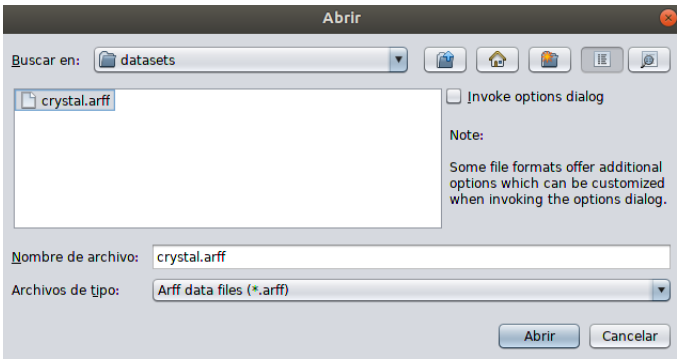



Figura 2: Selecció del fitxer de dades

Seguidament seleccionar el classificador BayesNet:

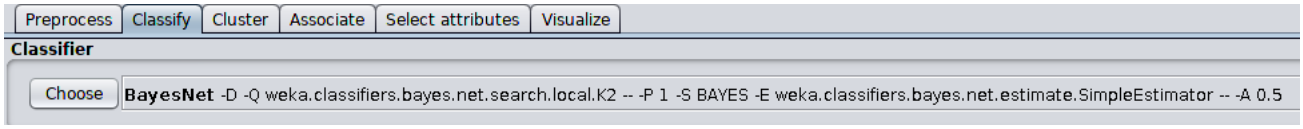



Figura 3: Classificador BayesNet

I finalment, seleccionar l'algorisme d'aprenentatge K2 per, posteriorment, modificar-ne els seus paràmetres d'execució i poder obtenir diferents tipus de models:

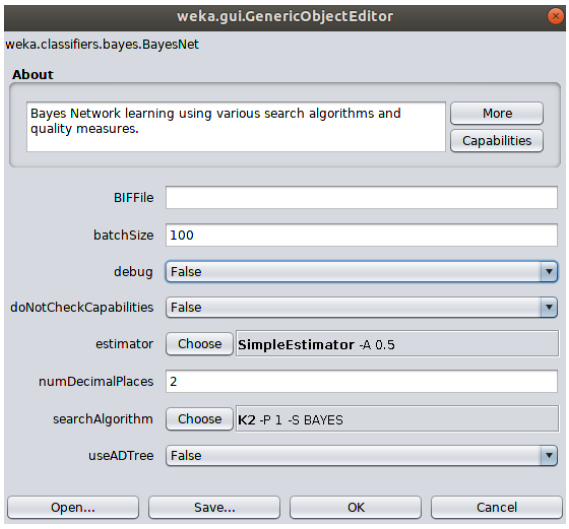



Figura 4: Algorisme de búsqueda K2

# 1 Model A

***Xarxes bayesianes obtingudes amb K2 a partir d'un model inicial buit (sense arestes inicials) i amb un ordre entre les variables escollit a l'atzar i amb un valor determinat pel nombre màxim de pares per variable (paràmetre O en l'algoritme K2).***

- Ajustem els paràmetres del K2 per satisfer les següents condicions:
  - **Model inicial sense arestes**: $initAsNaiveBayes \rightarrow False$
  - **Ordre aleatori de selecció de variables**: $randomOrder \rightarrow True$
  - **Màxim nombre de pares per variable**: $maxNrOfParents \rightarrow N$, $on\ N \epsilon \mathbb{N} - \{0\}$

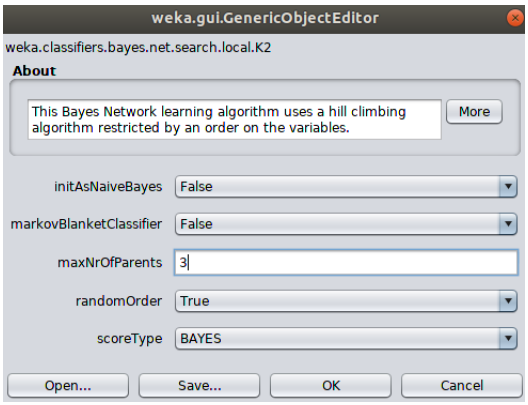



Figura 5: Paràmetres K2 model A

- Visualitzem el graf generat en l'execució:

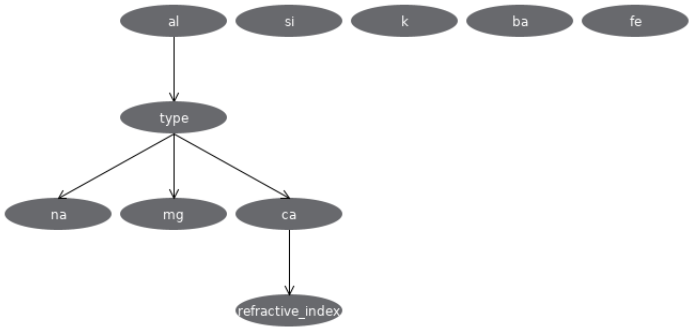



Figura 6: Graf aleatori generat amb els paràmetres del model A amb node arrel **si**

# 2 Model B

***Xarxa bayesiana naive (model únic), sent la variable de classe de vidre la variable independent (i pare de totes les altres). Per tant, el valor d'U en aquest cas haurà de ser 0.***

- Ajustem els paràmetres del K2 per satisfer les següents condicions:
  - **Type variable independent i pare de totes les altres**: *initAsNaiveBayes* → *True* i *(Nom)* → *Type*
  - **Màxim nombre de pares per variable**: *maxNrOfParents* → *0*

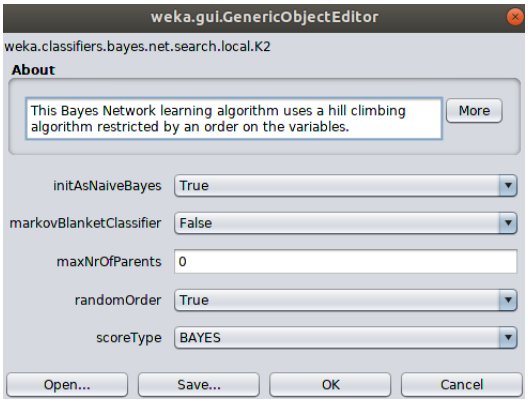



Figura 7: Paràmetres K2 model B

- Visualitzem el graf generat en l'execució:

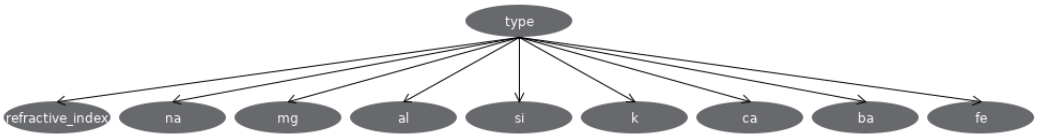



Figura 8: Graf aleatori generat amb els paràmetres del model B amb node arrel **type**

# 3 Model C

***Xarxes bayesianes obtingudes amb K2 a partir d'un model inicial que sigui la xarxa bayesiana naive del punt 2, però podent afegir arestes addicionals (i per tant U haurà de ser més gran que 0)***

- Ajustem els paràmetres del K2 per satisfer les següents condicions:
  - **Type variable independent i pare de totes les altres**: $initAsNaiveBayes \rightarrow True$ i $(Nom) \rightarrow Type$
  - **Màxim nombre de pares per variable**: $maxNrOfParents \rightarrow N$, *on* $N \epsilon \mathbb{N}$

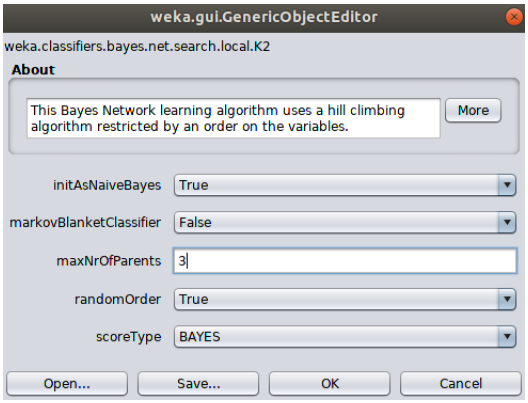



Figura 9: Paràmetres K2 model C

- Visualitzem el graf generat en l'execució:

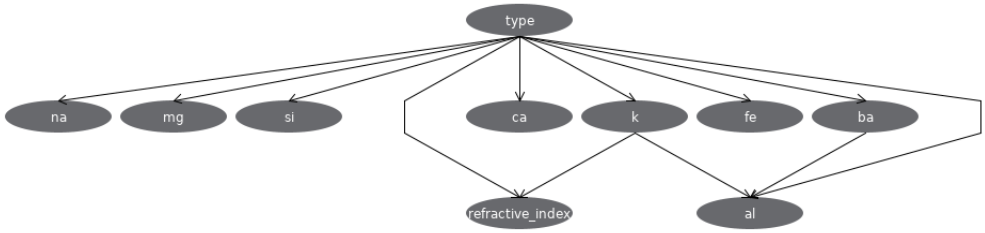



Figura 10: Graf aleatori generat amb els paràmetres del model C amb node arrel **type**

# Part III
# 10-fold Cross Validation

## 4 Generació train-folds

Per tal de separar el training set i el test set hem utilitzat una proporció (10% en aquest cas) per separar-los de forma aleatòria 10 vegades. D'aquesta forma simulem el *folding* de 10 però cada cop la part del test podria incloure mostres de testos anteriors. L'script utilitzat es troba en la llibreta del codi font amb l'implementació degudament comentada.

## 5 Evaluació test-folds

- Carreguem cada arxiu *crystal_train_i.arff* com a relació base.
- Adaptem els paràmetres d'execució del K2 segons toqui a cada model i configurem els test dataset que li volem passar. Aquest serà *crystal_test_i.arff*.

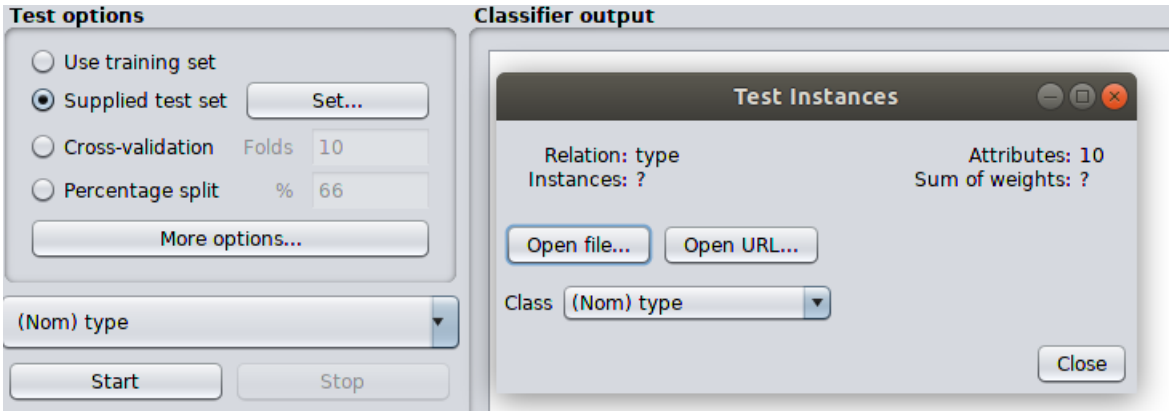



Figura 11: Configuració test sets Fold Cross Validation

- Un cop hem carregat el dataset $i$ de train i de test, fem 10 execucions d'aquest i ens quedem amb el millor.

# Part IV
# Selecció del millor model

## 6 Model A

| | Fold 0 | Fold 1 | Fold 2 | Fold 3 | Fold 4 |
|---|---|---|---|---|---|
| **Error %** | 61.904 | 52.381 | 18.134 | 47.619 | 64.248 |
| **LogScoreBayes** | -3552.58 | -3654.07 | -3578.69 | -3594.25 | -486.30 |

Taula 1: Errors de cada Fold pel model A - Part 1

| | Fold 5 | Fold 6 | Fold 7 | Fold 8 | Fold 9 |
|---|---|---|---|---|---|
| **Error %** | 38.095 | 57.142 | 47.619 | 57.142 | 52.381 |
| **LogScoreBayes** | -3614.40 | -3626.01 | -3664.74 | -3557.56 | -3637.68 |

Taula 2: Errors de cada Fold pel model A - Part 2

- Agafem la xarxa bayesiana que hem obtingut a partir del *Fold 2*, ja que és la que menys % d'error té , amb un **18.134%**, a l'hora de predir satisfactòriament.

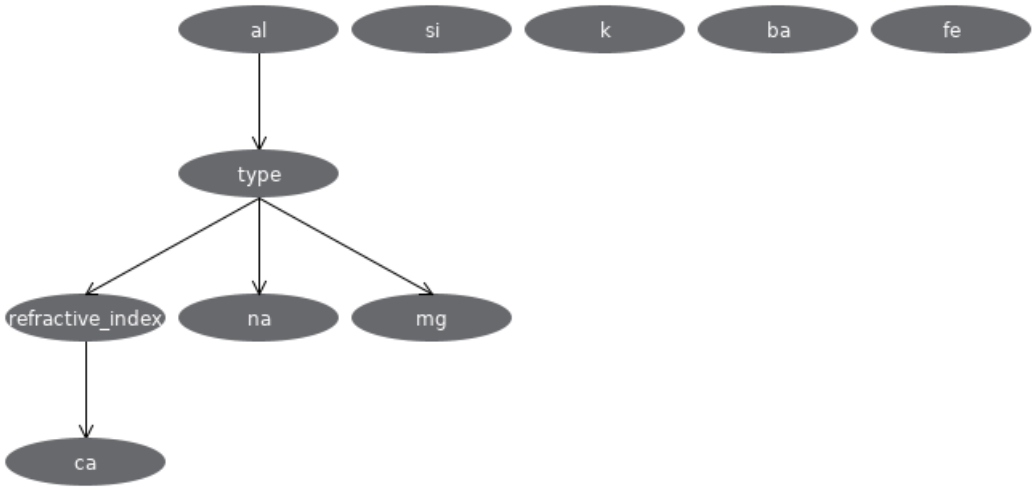



Figura 12: Graf de la millor xarxa bayesiana obtinguda pel model A

# 7 Model C

| | Fold 0 | Fold 1 | Fold 2 | Fold 3 | Fold 4 |
|---|---|---|---|---|---|
| **Error %** | 90.476 | 61.904 | 38.079 | 57.142 | 90.476 |
| **LogScoreBayes** | -3738.98 | -3705.42 | -3717.93 | -3728.72 | -3698.30 |

Taula 3: Errors de cada Fold pel model C - Part 1

| | Fold 5 | Fold 6 | Fold 7 | Fold 8 | Fold 9 |
|---|---|---|---|---|---|
| **Error %** | 71.428 | 61.904 | 76.190 | 61.904 | 80.952 |
| **LogScoreBayes** | -3735.10 | -3711.46 | -3707.39 | -3722.13 | -3732.31 |

Taula 4: Errors de cada Fold pel model C - Part 2

- Agafem la xarxa bayesiana que hem obtingut a partir del *Fold 2*, ja que és la que menys % d'error té , amb un **38.079%**, a l'hora de predir satisfactòriament.

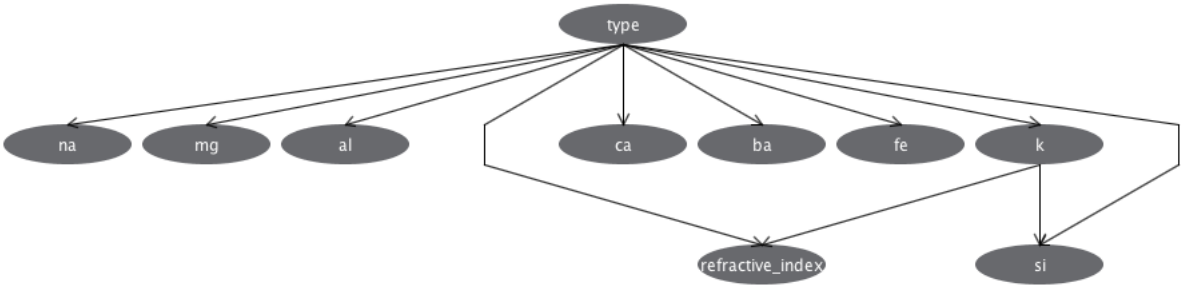



Figura 13: Graf de la millor xarxa bayesiana obtinguda pel model C