

Universitat de Lleida

Exercicis d'ARA

Bayesian Inference

Realitzat per:

Oriol Alàs Cercós, Marta Albets Mitjaneta, Sergi Simón Balcells

Data de lliurament:

8 de Juny de 2020

Universitat de Lleida
Escola Politècnica Superior
Grau en Enginyeria Informàtica
Aprenentatge i Raonament Automàtic

Professorat:
Ramón Béjar

Índex

1	Introducció	2
2	Parseig de dades	2
2.1	Linia de comandes	2
2.2	Weka i pandas	3
2.3	Partició de les dades	3
3	Models	4
3.1	Models de la classe 1	4
3.2	Model de la classe 2	5
3.3	Models de la classe 3	6
A	Gràfics	7

Índex de taules

1	Mètodes de partició utilitzats segons cada atribut	4
---	--------------------------------------------------------------	---

Índex de figures

1	Millor model de la classe 1	5
2	Millor model de la classe 2	5
3	Millor model de la classe 3	6
4	Scatter de l'atribut price (a) continu (b) discret	7
5	Scatter de l'atribut reviews (a) continu (b) discret	7
6	Scatter de l'atribut latitude (a) continu (b) discret	7
7	Scatter de l'atribut longitude (a) continu (b) discret	8

1 Introducció

En el present document, s'explicaran els passos seguits per al desenvolupament de les diferents parts de la pràctica de models probabilístics, utilitzant l'eina **weka**.

Els objectius principals que s'han volgut assolir, han sigut bàsicament tres: parsejar les dades donades de tal manera que weka pogués llegir-les, crear diferents models de tres tipus diferents:

1. Xarxa bayesiana obtinguda amb K2 amb el paràmetre $u = 3$, sent aquest el nombre màxim de parees per variable.
2. Naive Bayes sent la variable independent **overall_satisfaction** i amb $u = 0$.
3. Naive Bayes amb la mateixa variable independent que la classe 2, però sent $u = 3$.

D'aquests, s'havien de veure el millor de 10 exemples¹, i, finalment, l'explicació de com s'han escollit els conjunts discrets de valors per a transformar els atributs d'entrada en valors reals.

2 Parseig de dades

El parseig de les dades es realitza en dos fitxers continguts en **src**:

- **__main__.py** conte la funció **main** i funcions de parseig dels arguments passats per la línia d'ordres.
- **weka.py** conté funcions auxiliars per a categoritzar les dades i per a crear els texts que amb format **arff** per a weka.

2.1 Línia de comandes

Per a parsejar les dades passades per la línia d'ordres, donat que el projecte semblava requerir bastants arguments, s'ha utilitzat el mòdul estàndard d'**argparser**. Aquest ens permetia posar arguments opcionals per a ser utilitzats en el parseig. Així doncs, els diferents arguments que es poden passar són:

- **input-file**, **train-file**, **test-file** són els arguments requerits per l'enunciat de la pràctica. *Input file* és l'únic fitxer que ha d'estar creat amb anterioritat i ha de contenir un csv amb els valors que es volen parsejar.
- **pri**, **rev**, **lat**, **lon**: són diferents paràmetres que permeten canviar el nombre de particions en els arguments *price*, *reviews*, *latitude* i *longitude* respectivament. Per a canviar el valor, s'ha d'especificar un enter després del paràmetre, per exemple: **python3 -m src ... -pri 3**.

Aquests paràmetres han servit per a provar diferents valors en com partir les dades pel Weka. En la següent secció s'explicarà de quina manera s'ha realitzat i el perquè d'aquesta.

- **cpri**, **crev**, **clat**, **clon**: Dins de les maneres que es poden dividir en classificacions diferents dades contínues, hi ha dues formes que destaquen: fer la divisió per centils o realitzar-la amb rangs de valors de la mateixa mida. La diferència de realitzar-la per quantils és que tots els rangs tindran el mateix nombre de valors². Però, dividir-la així pot deixar en un mateix rang valors forans i valors normals, és a dir, si es divideix en dos grups la llista [2, 3, 4, 2000], 4 i 2000 aniran al mateix grup quan aparentment té poc sentit que ho siguin. Amb aquest paràmetre permetem en executar l'script, canviar la funció de tall desde d'un tall per rangs amb la mateixa mida, als creats a partir de quartils.
- **name**: permet canviar el nom de la relació amb què es guarda. S'ha d'especificar en el paràmetre: **python3 -m src ... -n notmydataset**.

¹En cas que es puguin tenir més de 10. En el cas de la classe 2, no es poden realitzar 10 exemples, ja que és cas únic.

²Realment, si la divisió per quartils no és exacta, un conjunt podria tenir menys valors, és a dir, si es divideix entre quatre conjunts un conjunt de 13 valors, llavors un d'aquests inevitablement tindrà un valor de més.

- **seed**: permet canviar la llavor en la qual s'agafen els valors per a fer els dos datasets (*train* i *test*). Si no s'especifica, s'agafa el valor per defecte dels últims 5 dígits d'un dels autors de la pràctica, com s'especificava a l'enunciat³.

Dins d'aquest codi s'utilitzen dos funcions:

- **ArgumentParser**: constructor de la classe. Ens permet crear un parser on es pot atribuir una descripció que s'utilitzarà amb el paràmetre *help* opcional.
- **add_argument**: ens permet afegir un argument. Si comença amb el caràcter '-', serà opcional, i es podrà especificar un nom verbós, com per exemple `-n` i `--name` realitzen la mateixa funció. Aquest mètode accepta diferents tipus de paràmetres per a canviar el seu comportament. A continuació s'expliquen els que s'han utilitzat en aquesta:
 - **metavar**: especifica una variable que s'utilitzarà per a mostrar el missatge d'ajuda per a paràmetres no opcionals.
 - **type**: ens permet especificar com s'ha de parsejar els valors per a poder ser utilitzats en el codi. En el nostre cas, només s'utilitza `str` i `int`.
 - **help**: proporciona un text d'ajuda quan es realitza la comanda `--help`.
 - **dest**: variable amb la qual serà guardada en el parseig.
 - **action**: especifica una acció per a realitzar. Només s'ha utilitzat per a guardar una constant `'store_const'`.
 - **const**: especificar la constant a guarda quan s'utilitza `'store_const'`. Amb conjunció amb el valor `default` i amb **action**, ens permet canviar la funció a utilitzar pel tall.

Per tal

2.2 Weka i pandas

Per a realitzar la funció de parseig s'ha utilitzat **pandas**. Per a fer-ho, s'ha usat la funció `read_csv`. A partir d'aquí, la manipulació de dades s'ha realitzat amb conjunció amb **numpy**. Així doncs, per a realitzar la discretització de les dades contínues, s'ha utilitzat les funcions⁴ `qcut` i `cut`. Aquestes dues funcions accepten els mateixos paràmetres, perquè el canvi es pot realitzar fluidament. A més a més per a fer el canvi d'una columna continua, es pot realitzar amb: `df[column_name] = cut_func(df[column], number_divisions, labels=False)`. *Labels* permet canviar el nom en el qual es categoritzen, passant dels rangs a noms donats com a llista. Per facilitar el parseig, s'ha passat el valor `False`, que canvia simplement numera les instàncies.

A més a més, s'ha realitzat la funció de `parse_weka`, que retorna una `str` representant la informació en format arff. S'ha retornat el valor en comptes de guardar-lo directament en el fitxer per separar la lògica d'on i que es guarda. La funció en si és una agrupació de `join` de les files i columnes per les diferents línies. La divisió dels datasets es realitza en la mateixa funció per assegurar que els dos sempre tindran el mateix nom a la relació i els atributs.

2.3 Partició de les dades

Per tal de passar els atributs amb valors continus a discrets, s'ha utilitzat la llibreria **pandas**, que ens permet llegir el `csv` d'una manera molt còmode i partir-lo de diferents maneres. Les funcions que s'han incorporat al nostre script de parseig són:

- **cut**. Divideix el dataset en diferents intervals utilitzant el rang de valors.
- **qcut**. Divideix el dataset mitjançant quantils, és a dir, tots els intervals tindran el mateix nombre de dades.

³Hem agafat els dígits 04011 de Sergi Simón, no són els últims 5 per un error.

⁴L'explicació de les funcions es pot trobar a la secció 2.3.

A més a més, es va pensar en un primer moment utilitzar `KMeans` per tal d'agrupar la longitud i la latitud en un sol atribut. No obstant això, els atributs no han de tenir correlació directa. Weka ja ho realitzarà per nosaltres mitjançant models probabilístics. Per tant, es va declinar utilitzar aquesta opció.

Per tal d'arribar a la conclusió de quin mètode era el millor per tal de categoritzar els atributs, s'ha utilitzat la llibreria `matplotlib` per a visualitzar les dades de manera gràfica. S'ha realitzat un gràfic⁵ per cada atribut a categoritzar sobre l'*overall_satisfaction* i després s'ha realitzat altre cop el gràfic havent-lo categoritzat. Aquest procediment s'ha repetit fins a trobar una partició que fos més representativa.

El mètode de cada atribut realitzat es pot veure a la Taula 1. Com es pot observar, el millor mètode per tal de preservar el comportament de la variable continua és utilitzant `cut`.

Atribut	Mètode	Num. Divisions
price	cut	15
reviews	cut	10
latitude	cut	20
longitude	cut	20

Taula 1: Mètodes de partició utilitzats segons cada atribut

3 Models

En aquest apartat, es pot veure els valors, tant de la puntuació $\log(UPSM)$ com el percentatge d'error de la classificació, obtinguts en els diferents models de cadascuna de les classes de models.

3.1 Models de la classe 1

1. **log(UPSM):** -92608.96126554035 **Error de classificació:** 48.4167 %
2. **log(UPSM):** -92139.67866809308 **Error de classificació:** 48.4167 %
3. **log(UPSM):** -92388.27489566772 **Error de classificació:** 48.4167 %
4. **log(UPSM):** -92227.05224681877 **Error de classificació:** 48.4484 %
5. **log(UPSM):** -92066.4712487785 **Error de classificació:** 48.4167 %
6. **log(UPSM):** -92302.15263672064 **Error de classificació:** 48.4167 %
7. **log(UPSM):** -92180.71383973274 **Error de classificació:** 48.4167 %
8. **log(UPSM):** -92586.42609497187 **Error de classificació:** 48.4167 %
9. **log(UPSM):** -92245.67223271335 **Error de classificació:** 48.4167 %
10. **log(UPSM):** -93188.71009303001 **Error de classificació:** 48.4167 %

Per a poder seleccionar el millor model dels anteriors, ens hem de fixar sobretot en quin d'ells ens classifica correctament un major nombre d'instàncies i, per tant, vaga la redundància, quin model té el menor percentatge d'error de classificació.

Com es pot observar, tots els models tenen el mateix percentatge d'error excepte un, el qual es pot descartar directament, ja que és més alt que la resta. Entre els altres models, no es sorprenent que el percentatge no variï, ja que es pot observar en les dades tan inicials com després de separar-les en els conjunts d'aprenentatge i de prova, la tendència d'un 50% de casos en els quals el valor d'*overall_satisfaction* és de 4.5.

⁵Els gràfics poden trobar a la secció A de l'apèndix.

Per tant, veient aquestes dades, s'arriba a la conclusió que es troben en una situació d'empat". Una manera de resoldre'l, i la qual s'ha usat, és comparar els valors del $\log(UPSM)$ de cada model.

Per concloure, es considera que, el millor model d'entre els que tenen el millor percentatge en aquesta classe, serà aquell que tingui un *UPSM score* més elevat, és a dir, el model que el seu $\log(UPSM)$ sigui més alt⁶. Per tant, considerarem com a millor model de la classe 1, el classificat amb el nombre 5, del qual es pot observar la seva representació gràfica a la Figura 1.



Figura 1: Millor model de la classe 1

3.2 Model de la classe 2

1. **log(UPSM):** -117515.15847723951 **Error de classificació:** 48.4484 %

En el cas de la segona classe, només hi trobem un model possible, per tant, serà aquest és el que es considera millor. El fet que només n'hi hagi un de possible és degut al fet que aquest model és el corresponent a la xarxa bayesiana *naïve*, tenint com a variable de classe **overall_satisfaction**, el qual és un model únic.

A la figura 2 es pot veure la representació gràfica d'aquest model.

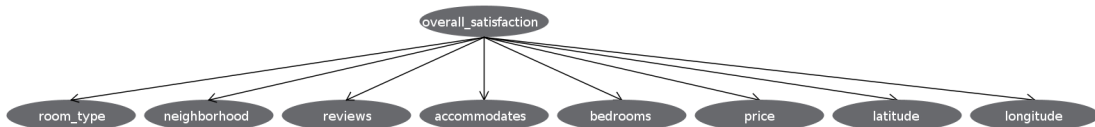


Figura 2: Millor model de la classe 2

⁶Com que els valors es troben en negatiu, el valor més gran és el que té el valor absolut més petit.

3.3 Models de la classe 3

1. **log(UPSM):** -96800.35169816265 **Error de classificació:** 48.3534 %
2. **log(UPSM):** -95829.88423245584 **Error de classificació:** 48.86 %
3. **log(UPSM):** -95565.62218051209 **Error de classificació:** 48.8284 %
4. **log(UPSM):** -95859.94046082163 **Error de classificació:** 48.4167 %
5. **log(UPSM):** -97231.01625115846 **Error de classificació:** 48.7334 %
6. **log(UPSM):** -96894.65682950999 **Error de classificació:** 48.3217 %
7. **log(UPSM):** -95822.6410503832 **Error de classificació:** 48.67 %
8. **log(UPSM):** -97094.31982431604 **Error de classificació:** 48.1951 %
9. **log(UPSM):** -95604.5241813889 **Error de classificació:** 48.575 %
10. **log(UPSM):** -96108.4925816474 **Error de classificació:** 48.3534 %

Per tal de poder escollir el millor model d'aquesta classe, s'ha fet de la mateixa manera que en la classe 1. Tot i això, en aquest cas sí que s'ha pogut veure més diversitat de percentatges d'error en els diferents models. Per aquest motiu, no ha estat necessari comparar els valors de l' *UPSM*.

Els percentatges són bastant semblants entre si, ja que el fet d'encertar algunes variables més que un altre model, no implica un gran canvi en el percentatge d'encerts, a causa de la gran quantitat d'instàncies que tenim en el test. Malgrat tot, hi ha un model que ha encertat algunes instàncies més que la resta. Per tant, considerem com a millor model de la classe 3, classificat amb el nombre 8, del qual es pot observar la seva representació gràfica a la Figura 3.

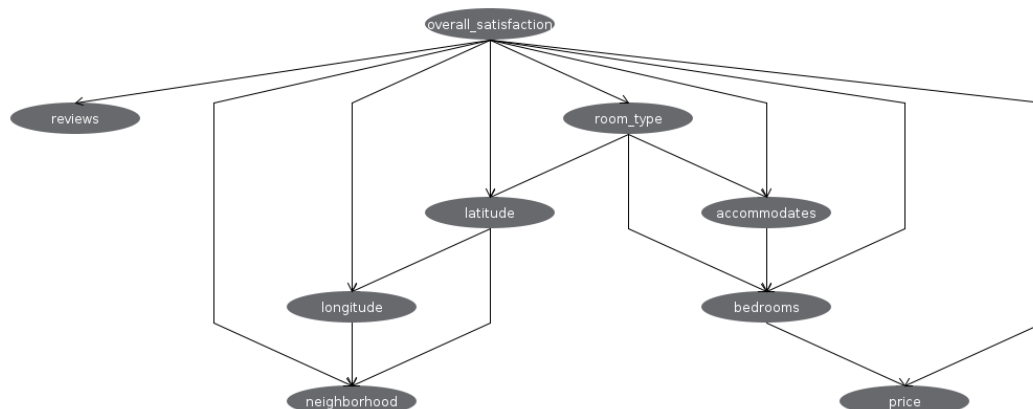


Figura 3: Millor model de la classe 3

A Gràfics

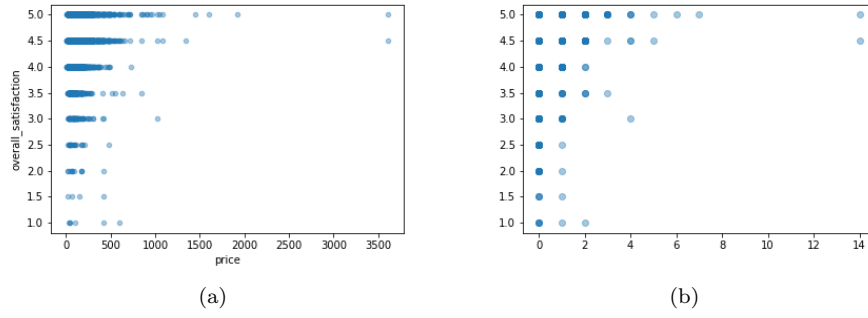


Figura 4: Scatter de l'atribut price (a) continu (b) discret

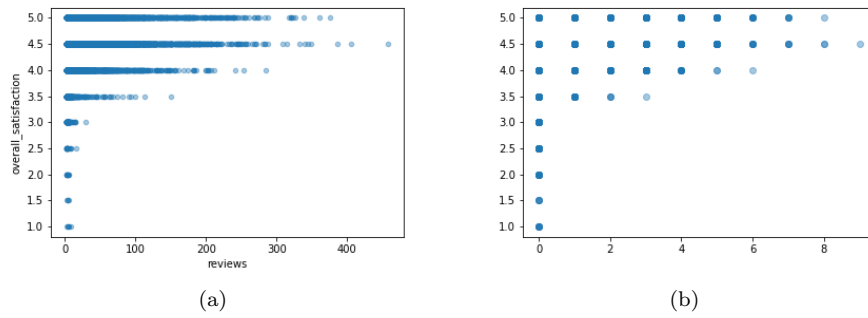


Figura 5: Scatter de l'atribut reviews (a) continu (b) discret

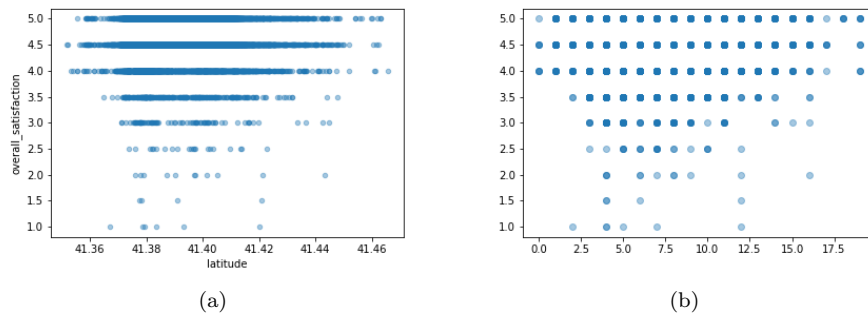
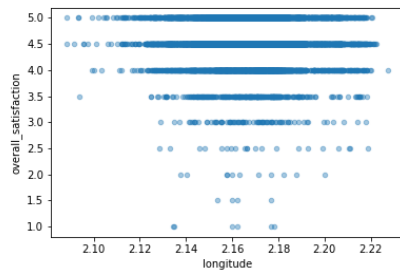
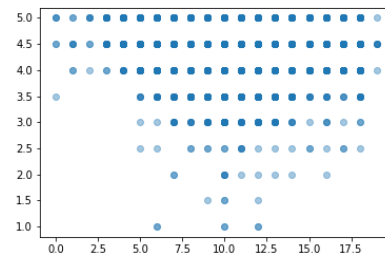


Figura 6: Scatter de l'atribut latitude (a) continu (b) discret



(a)



(b)

Figura 7: Scatter de l'atribut longitude (a) continu (b) discret