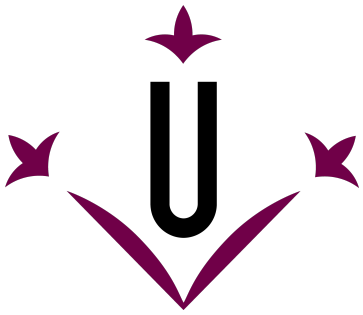

Universitat de Lleida

Grau en Enginyeria Informàtica

Pràctica 2: Models probabilístics

Assignatura d'Aprenentatge i Raonament Automàtic

Jordi Farrera Palou
Joel Ampurdanés Bonjoch

20 de juny de 2022

Índex

Índex de figures

1 Introducció

En aquesta pràctica, se'ns demana realitzar 3 models probabilístics utilitzant weka:

1. Xarxes bayesianes obtingudes amb K2 a partir d'un model inicial buit amb un ordre entre les variables escollides a l'atzar i amb un valor màxim de nombre de pares per variable igual a 3.

2. Xarxa bayesiana naive, on la variable `overall_satisfaction` és la variable independent. Per tant, el valor màxim de pares per variable en aquest cas, serà igual a 0.

3. Xarxes bayesianes obtingudes amb K2 a partir d'un model inicial que sigui la xarxa bayesiana naive del punt 2, però amb un valor del número màxim de pares per variable igual a 3.

Per a poder realitzar aquests 3 models, primer s'ha de crear un script de python per parsejar les dades i generar els fitxers *train.arff* i *test.arff*. La comanda per executar aquest script és:

```
python3 transform_to_arff.py barca.csv <train-arff-file> <test-arff-file>
```

Per a executar el programa es necessari tenir **python3** al sistema. Si es té, s'ha de realitzar les següents comandes al terminal:

```
python3 -m venv venv
source venv/bin/activate
python3 -m pip install -r requirements.txt
```

2 Parseig de dades

L'script que realitza el parseig de les dades és: **transform_to_arff.py**

2.1 Línia de comandes

Per parsejar les dades passades per la línia de comandes, s'ha utilitzat el mòdul de python `argparser`. Aquest ens permet posar arguments opcionals per a ser utilitzats en el parseig. Per tant els diferents arguments que es poden passar són:

- `dataset`: argument en el qual s'indica la ruta cap al fitxer .csv que conté les dades a parsejar.

- `train`: s'indica el nom a posar pel fitxer de train incloent l'extensió .arff, és a dir, <nom-fitxer>.arff.

- `test`: s'indica el nom a posar pel fitxer de test incloent l'extensió .arff, és a dir, <nom-fitxer>.arff.

- `seed`: permet canviar la llavor en la qual s'agafen els valors per a fer els dos datasets train i test. Si aquest camp no s'especifica, s'agafa com a valor per defecte els últims cinc dígits del DNI d'un dels autors de la pràctica.

- `percentage`: permet canviar el percentatge de les dades que aniran al train i per tant, les del test també. Aquest percentatge s'indica de la següent manera, on si es vol un 85% l'argument a passar ha de ser 0.85. Si aquest camp no s'especifica, per defecte el percentatge serà 75%.

Dins d'aquest codi s'utilitzen dos funcions:

- `ArgumentParser`: constructor de la classe.

- `add_argument`: ens permet afegir un argument. Aquest mètode accepta diferents tipus de paràmetres per a canviar el seu comportament, els quins hem utilitzat són els següents:

  - `help`: proporciona un text d'ajuda quan es realitza la comanda `--help`.
  - `nargs`: posant `nargs` com a ? (`nargs='?'`), ens permet que si indiquem l'argument en la línia de comandes ens agafarà aquest com a únic, i a més, si aquest no està indicat s'agafarà el valor `default` com a argument.
  - `default`: el valor o l'argument que s'agafaria si aquest no ha estat indicat per la línia de comandes.
  - `type`: ens permet especificar com s'han de parsejar els valors per a poder ser utilitzats en el codi.

2.2 Pandas

Per poder realitzar el parseig de les dades hem utilitzat `pandas`. En primer lloc la funció `read_csv`, la qual ens permet llegir el dataset.

Per tractar les dades que tenim al dataset, hem canviat els noms de la columna `room_type` utilitzant el mètode `replace` per tenir-los tots amb el mateix format, ja que hi ha diferents tipus d'habitacions que el seu format és separat en guions i d'altres que es separat en espais.

També hem mapejat els valors de la columna `overall_satisfaction` on inicialment els valors d'aquesta columna eren [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5] i al mapejar-los queda de la següent manera: [1, 2, 3, 4, 5, 6, 7, 8, 9].

En les columnes `accomodates` i `bedrooms` hem canviat el tipus dels valors que hi han en aquestes columnes a string ja que volem representar aquests valors en el format .arff.

Les columnes `review`, `price`, `latitude` i `longitude`, per passar els seus valors continus a discrets s'ha utilitzat la funció `cut` de la llibreria `pandas`, la qual ens permet dividir el dataset en diferents intervals utilitzant el rang de valors. El nº de divisions que s'ha

escollit per aquestes columnes és de 15 ja que s'ha provat diferents nº de divisions com per exemple, 5 o 20 i al posar els arxius train i test a Weka ens donava una *accuracy* menor comparat amb 15 divisions que ens dóna major. Després d'aplicar el mètode `cut` canviem el tipus dels valors de les columnes a string ja que volem representar aquests valors en el format .arff.

3 Models

En aquest apartat, es pot veure els valors, tant la puntuació `log(UPSM)` com el percentatge d'error de la classificació.

3.1 Models de la classe 1

Les 10 execucions que s'ha realitzat, ens ha donat els següents resultats:

1. **log(UPSM)**: -89814.4077531957 **Error classificació**: 50.1583 %
2. **log(UPSM)**: -90397.4380825399 **Error classificació**: 50.19 %
3. **log(UPSM)**: -90415.35102713174 **Error classificació**: 50.1583 %
4. **log(UPSM)**: -90389.28857008628 **Error classificació**: 50.1583 %
5. **log(UPSM)**: -89867.8124939057 **Error classificació**: 50.1583 %
6. **log(UPSM)**: -90211.85119256149 **Error classificació**: 50.1583 %
7. **log(UPSM)**: -90211.85119256149 **Error classificació**: 50.1583 %
8. **log(UPSM)**: -90450.80679248775 **Error classificació**: 50.1583 %
9. **log(UPSM)**: -90401.35562920943 **Error classificació**: 50.1583 %
10. **log(UPSM)**: -89702.37746226412 **Error classificació**: 50.1583 %

Per a poder seleccionar el millor model dels anteriors hem d'observar quin d'ells té el menor error de classificació, és a dir, el quin més encerts té.

Com s'observa, casi tots els models excepte el 2 tenen el mateix error de classificació, i perquè l'error menor és el més repetit per a seleccionar el millor model s'ha d'agafar el quin tingui la puntuació `log(UPSM)` major. Per tant, el millor model és el 10è.

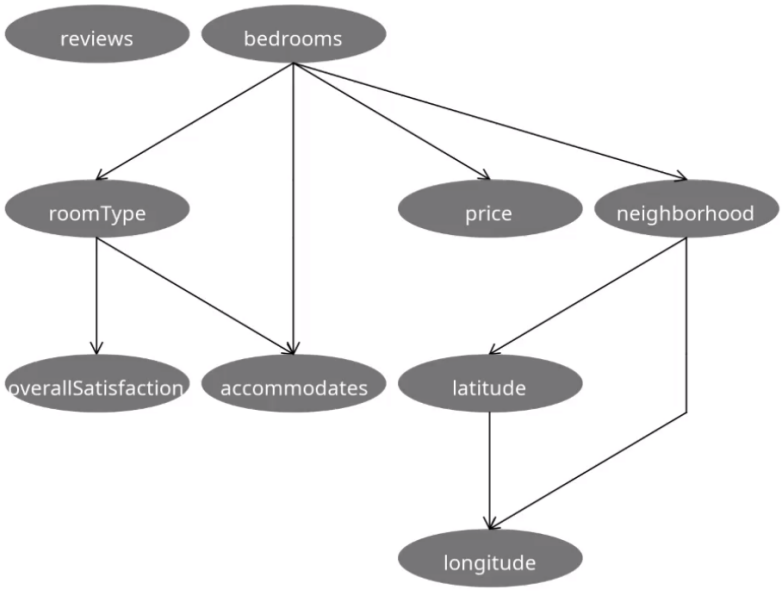

Figura 1: Millor model de la classe 1

3.2 Model de la classe 2

L'execució que s'ha realitzat, ja que només hi ha un model possible, és la següent:

1. **log(UPSM)**: -114260.90629096203 **Error classificació**: 50.57 %

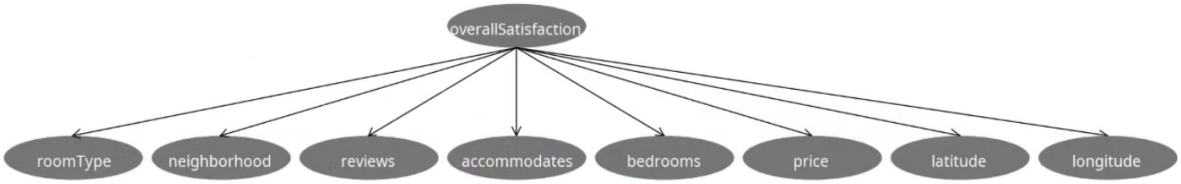

Figura 2: Model de la classe 2

3.3 Models de la classe 3

Les 10 execucions que s'ha realitzat, ens ha donat els següents resultats:

1. **log(UPSM)**: -92145.77402274412 **Error classificació**: 50.2217 %
2. **log(UPSM)**: -93769.8023647269 **Error classificació**: 49.8417 %

3. **log(UPSM)**: -92947.0699716769 **Error classificació**: 49.9367 %
4. **log(UPSM)**: -92383.57112958582 **Error classificació**: 50.19 %
5. **log(UPSM)**: -93769.8023647269 **Error classificació**: 49.8417 %
6. **log(UPSM)**: -93355.04636895073 **Error classificació**: 50.0317 %
7. **log(UPSM)**: -92773.38194075429 **Error classificació**: 50 %
8. **log(UPSM)**: -93769.8023647269 **Error classificació**: 49.8417 %
9. **log(UPSM)**: -92339.30633873143 **Error classificació**: 50.0633 %
10. **log(UPSM)**: -92563.94627690384 **Error classificació**: 50.19 %

Per escollir el millor model de la classe 3, al igual que a la classe 1, s'ha agafat el model amb menor error de classificació.

Com s'observa, tenim que els models 2, 5 i 8 tenen el mateix error de classificació i també la mateixa puntuació `log(UPSM)`, i per tant, el millor model es qualsevol d'aquests tres.

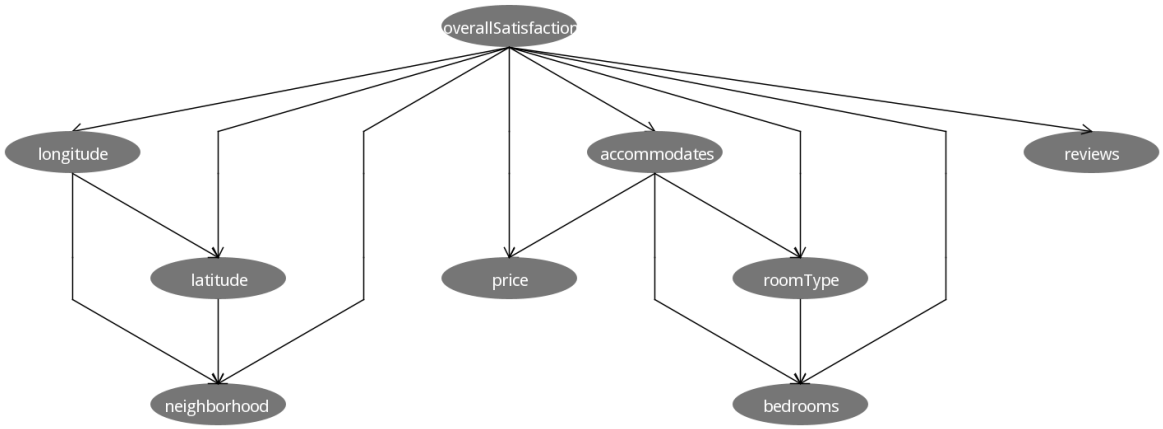

Figura 3: Millor model de la classe 3