Adatbánya Gyakorlat

Read CSV

- open file: url , path
- read csv
  - column operator: \t
  - decimal character :
  - use quotes ☑
  - parse numbers ☑
- rename (attributes): listába eredeti - új név
- numeric to polynomial:
  - attr filter type: single
  - attl: label
- set role
  - attr name: pl label
  - attr role: pl label
- store: tárolás helyének megadása + név

PCA

Dimenzió csökkentés - fő komponens analízis

- adathalmaz behúzása
- normalize: adatok normalizálása
  - subset
  - kiválasztjuk hogy melyik attr
  - range min 0.0, max 1.0
- split data: train és test adathalmaz
  - paritions 0.7, 0.3
  - stratified sampling
- PCA: reduction keep variance
  - treshold 0,95
  - train adathalmazt belehúzni (split 1)

> ahol már nagyobb a commulative var. mint 0,95 azokat az attr-kat kivesszük a modellből (az utolsó ahol elérjük az értéket)

- apply model: PCA és split data testje
  - felépített modell alkalmazására
  - PCA pre:mod, split2:unl

Döntési fa (szimpla predikció magyarázattal)

- adathalmaz behúzása
- split data: train-test 0.7-0.3
- multiply train: többször fog kelleni

- multipy test: ez is
- decision tree: train belehúzva
    - gain ratio, max depth: 10, confid: 0,1, min gain: 0.01, min leaf: 2
- apply model a 2 kimenetre (mod, exa)
- apply model2: az 1. apply mod kimenetele és a test adathalmaz

> a lab kimenet a test predikcióit tartalmazza
>
> confidence is van, a döntési fa modellje

- multiply model: a 2. apply mod kimenete
- explain prediction:
    - multipy mod
    - train
    - test
    - explain attr: 3
- EPOO: melyik attribútum járult hozzá erősebben a döntés meghozatalához
- test adathalmazra: zöld ha segítette a predikciót, piros ha ellentmondott velük
- ezt mutatja egy másik output 1-1 attr-al
- minden sor külön attr-ok fontossága
- attr-ok súlyozására

Döntési fa (precedencia értékeléssel)

- adathalmaz
- split data 70-30
- multiply
- decision tree: train behúzva
    - beállítások ua. gain ratioval
- apply model a dt 2 kimenetelére
- performance (binomial): az apply 1 kimenet
    - main: first
    - accuracy, classification error, AUC ✅

> Megadja az illesztés pontosságát, az eltalált osztályok %-os megoszlását

- apply mod az 1. mod kimenetelével és a test adathalmazzal
- performance(binom): ua. a 2. apply mod-re
- performance exa kimenetele:
    - sort by confidence descenfing
    - filter example range 1-10

> az első tíz legbiztosabb (true) sort adja vissza

- az apply mod 2 mod kimenete multiply + test adathalmaz
    - create lift chart
    - class: ture
    - type: frequency
    - number: 10

- false-ra is ugyan ilyen lift chart

> confidencia és prediktált/valós érték alapján diagrammot készít

Több döntési fa összehasonlítása (pontosság kiértékelése)

- adathalmaz behúzás
- multiply az egész adathalmazra
- 4 db split validation, 4 féle döntési fa
  - split: relative
  - ratio 0.7
  - type: automatic
- training oldal: decision tree
  - gini index ua. de max leaf 10
  - information gain ua.
  - gain ratio ua.
  - accuracy ua

> ave kimeneteket kihúzni outputnak egyel feljebb, megadja a fák pontosságát

- test oldal: apply modell, performance binomial, first 3 ✅
- kívül apply model az első split validation kimenetére (mod) + adathalmaz
- filter több dolog szerint custom_filter (spam=false, prediction=true), több kimenet
- exa sort conficence(true) descending
- original (ori) sort conficence(true) descending

Osztályozási módszerek

- adathalmaz
- normalize: all range 0.0-1.0
- multiply data egész adathalmaz
- 4 split validation
  - train: rule induction (szabály alapú)
    - minden az alapbeállításokon
  - test: apply model, performance binom first 3 ✅
  - train: naiv bayes
  - test ua.
  - train logistic regression (minden az alapon)
  - test: ua.
  - train neural network (alap)
  - test ua.
- output: mod és ave
- Compare Rocs: a 4 split validation ide is bele, ugyan úgy, másolás
  - roc outputba

> Az egyes osztályozó modszerek pontosságai hasonlíthatóak össze a segítségével

SVM (támaszvektor gép)

- adathalmaz
- normalize subset (3 capital)
- split data 70-30
- multiply train test
- SVM1 kernel : dot , C : 0.1
- SVM2 kernel: polynomial, C: 0.1
- 2 apply model a 2 SVM modeljére és a teszt adathalmazra
- 2 performance binom first 3 ✅

> két féle SVM kimenetének, pontosságának összehasonlítása

Ensemble módszerek (bagging, boosting)

- adathalmaz
- multipy
- 3 split validation
    - train bagging
    - test apply model performance binom first 3 ✅
    - train decision tree
    - test ua.
    - train: adaboost
    - test ua.
- output: mod és ave

> modellek és a velük elérhető pontosságok összehasonlításai

Klaszterezés

- adathalmaz
- multiply egész adathalmaz
- k-means clustering k = klaszterek száma
    - még két multiply
    - map clustering as labels:
        - első input k-means 2. output
        - második input k-means 1. output
    - cluster model visualizer
        - első input k-means 1. output
        - második input k-means 2. output
- loop parameters: input az adathalmaz
    - k-means: n=5
    - performance(cluster distance performance): fordított kötés
    - log: 1 perf. out
        - Number of cluster: Clustering(2) parameter k
        - Avg. within distance: Performance value avg_within_distance
        - Davies-Bouldin index Performance value DaviesBouldin

> megfelelő klaszterszám megtalálása az adathalmazra
>
> charts: x= cluster, y= avg distance
>
> ahol a nagy törés van az a jó klaszterszám

Egyéb klaszterezési algoritmusok

- adathalmaz, multiply
- k- means k=4 2.output out
- agglomerative clustering: simple
- agglomerative clustering: complete
- agglomerative clustering: average
- mind 3ra lapítás Flatten cluster k=4 2.output out

> output vizualizáción összehasonlíthatóak az egyes klaszterezések eredményei

- multiplyból detect outlier (5,2,1)
- filter examples outlier = false
- agglomerative clustering: complete
- flatten

DBSCAN

- adathalmaz
- DBSCAN: epsilon=2.5

> Megfelelő ε értékkel az elvárt klasztereket kapjuk
>
> az eredményt az output vizualizációjában játjuk (pontok távolságán alapul)

Társítási szabályok

- adathalmaz
- filter examples
  - túl gyakori a heineken, ront az eredményen ezért kiszűrjük
  - PRODUCT = heineken ; invert filter ✅
- multily, az egyik már kiküldve végső outputra
- subprocess (adatelőkészítés) az FP growth-hoz
  - aggregate: PRODUCT concatenation
    - group by CUSTOMER
  - Rename concat(PRODUCT) -> Products ; additional CUSTOMER -> Customer
  - set role Customer : id
  - Multiply, 2 kivezetés outputra a subprocessből, az egyik bekötve végső outputra is
- FP growth
  - list item in column ; min support 0.1, min number of itemset 50
- Create association rules ; input : FP growth fre (frequency) outputja
  - alap beállítások, mindkét output kivezetve végső outputra
- másik subprocess az apriorihoz
  - generate attributes ; TIME -> TIME+1

    - pivot
        - group by attributes CUSTOMER
        - column grouping attribute PRODUCT
        - aggregation attributes; TIME:sum
    - rename by replacing sum\(TIME\)_ -> semmire
    - replace missing values ; default: zero
    - numeric to binomial; single: CUSTOMER ; inverted ☑
    - set role CUSTOMER: id
    - Multiply, 2 kivezetés outputra a subprocessből, az egyik bekötve végső outputra is
- w-apriori
    - N: 20.0
    - M: 0.01
    - I ☑
    - V ☑
    - kivezetés végső outputra

Rendellenesség keresés

- adatbázis-normalize-multipy
- sample (relative 0.1)
    - detect outlier distance : neighbour változhat
- detect outlier densities (12, 0.9)
- sample, ua.
    - detect outlier LOF
- PCA fixed, 2 - multiply
- apply modellek: mod= PCA, url= outlier output 1

> eredmények szintén az output vizualizációjában hasonlíthatóak össze
>
> melyik módszer mit detektált kiugró értéknek

Generated PDF