UNIVERSITAT ROVIRA i VIRGILI

Sistemes d'Informació en les Organitzacions

Filtratge col·laboratiu: Prediccions de puntuacions dels restaurants

Grup 4

Aleix Sancho Pujals

Francesc Ferré Tarrés

Víctor López Romero

UNIVERSITAT ROVIRA i VIRGILI

Índex

1. Introducció

Després d'analitzar tots els resultats obtinguts en la primera part de la pràctica, vam procedir a provar diferents procediments per tal d'aconseguir una bona predicció de dades. Primerament creiem que era necessari una divisió de les dades per a disminuir el conjunt de dades, ja que inicialment fer prediccions amb 4M de puntuacions és molt complicat que la memòria del sistema ho suporti.

Una vegada distribuïdes havíem d'entrenar cada grup amb un conjunt d'entrenament i un de test i finalment fer les prediccions de totes les dades que falten. Òbviament per fer les divisions en grups és quan faríem servir totes les mètriques que vam analitzar en la primera part de la pràctica.

2. Mètodes emprats

Inicialment vam fer ús de diferents tècniques que consistien a repartir els usuaris en funció dels quals van puntuar més restaurants (en una escala de 10). Per tant teníem els restaurants repartits entre els quals van anar de 0-10, 11-20, 21-30, 31-40, 41-50, 51-60, 61-70, 71-80, 81-90, 91-100. Llavors, un cop realitzat això, el que vam fer va ser utilitzar les tècniques presents en el paquet de python anomenat scikit-learn, destinat a predir valors desconeguts (https://scikit-learn.org/stable/modules/impute.html). Aquestes tècniques es basaven amb la predicció de dades a partir del veïnatge. Els resultats d'aquestes prediccions, van ser nefastes, obteníem un MAE molt elevat (anava entre 4.48 a 6.68).

Vist que el resultat no va ser com esperàvem, vam creure que possiblement seria culpa de la divisió de dades, per tant vam mantenir la llibreria de sklearn però aquesta vegada per utilitzar la funció k-means i fer clustering. Els clústers els vam fer a partir de diferents dades: la mitjana de puntuacions d'usuaris, la mitjana de puntuacions de restaurants, la mediana, la desviació estàndard, etc. Totes aquestes distribucions de dades van donar un resultat similar a l'anterior i per tant vam obviar els resultats.

Llavors, veient aquests mal resultats, vam decidir cercar altres formes i vam veure un altre paquet de python, que es basa amb algorismes d'entrenament i que posteriorment, un cop entrenat l'algorisme, realitza prediccions. Aquest paquet s'anomena Surprise (https://surprise.readthedocs.io/en/stable/getting_started.html).

Aquest és el que ens ha permès arribar segons en el rànquing venint del final de la taula, amb un MAE de 3.3862. El paquet Surprise de Python conté diversos algorismes de predicció, per tal d'escollir el millor algorisme vam estar un temps prudencial provant cadascun dels algorismes amb una funció que aquest paquet presenta anomenada cross_validate. Amb aquesta funció es fa com una prova per saber si va bé o malament cadascun dels algorismes. Per escollir l'algorisme bo, hem mirat aquell que ha donat més bon RMAE (segons les seves prediccions). En el nostre cas, ha estat l'algorisme SVD. I ja a partir d'aquí simplement hem recollit la seva predicció, a través de la funció predict. Per fer les prediccions amb els diferents algorismes hem decidit que li passem aquells usuaris que han estat en més restaurants (més de 90) i aquells restaurants que han estat visitats més de 68000 vegades, per tal que les dades siguin les més fiables possibles.

3. Instruccions sobre el codi

L'execució en un ordinador no funciona del tot bé perquè dóna errors sobre la memòria. Per tant, hem hagut de realitzar aquesta pràctica amb l'ús del poder d'execució del cloud. En aquest cas, hem fet ús del python que hi ha a Anacondas i IDE Spyder. El que hem fet és muntar totes les dades amb l'estructura correcta per tal que l'algorisme funcioni correctament. Inicialment tenim les dades amb una estructura de dataframe de la llibreria pandas amb una estructura com aquesta:

```
        userID  restaurantID  rating
index
1            1             1   -7.82
1            1             2    8.79
1            1             3   -9.66
1            1             4   -8.16
1            1             5   -7.52
...        ...           ...     ...
73421    73421            65    1.36
73421    73421            66    7.18
73421    73421            69    0.49
73421    73421            72    5.87
73421    73421            82    6.65

[4096360 rows x 3 columns]
```

Aquí tenim totes les puntuacions que s'han fet en tot el sistema, les puntuacions nul·les (99) no hi són.

Després el que hem fet és adaptar-ho a l'estructura del surprise, primer cal crear un objecte Reader:

```
reader = Reader(rating_scale=(-10, 10))
```

On el rating_scale és l'escala de puntuacions que en el nostre cas oscil·la entre -10 i 10.

Seguidament guardem la nostra estructura de dades en l'objecte Reader:

```
all_data = Dataset.load_from_df(all_ratings[['userID', 'restaurantID', 'rating']], reader)
```

Un cop tenim això, ja podem començar a treballar amb les dades. Per assegurar-nos d'entrenar bé el sistema, el que fem és dividir tot el grup de dades (sabem que totes les dades són correctes) en dos grups, un grup d'entrenament i un grup de testing. El primer grup, el d'entrenament, serà amb el qual l'algorisme de machine learning treballarà per tal d'aconseguit trobar patrons i buscar una relació de similitud entre usuaris, puntuacions i restaurants. Un cop ha aconseguit trobar un patró vàlid, provarem aquest algorisme amb el conjunt de test per saber si les prediccions s'acosten als valors que tenim i així poder aproximar el MAE que obtindrem amb les dades que voldrem predir posteriorment, tot aquest procés es fa amb les següents línies de codi:

```
trainset, testset = train_test_split(all_data, test_size=0.25)

algo = SVD()

predictions = algo.fit(trainset).test(testset)


mae = accuracy.mae(predictions)
mae
```

En el nostre cas vam fer diverses proves i vam decidir que la repartició de dades entre el conjunt d'entrenament i el conjunt de test és d'un 75% de les dades pel conjunt d'entrenament i d'un 25% de les dades pel conjunt de test. Un cop dit això, cal apuntar que el repartiment de les dades és aleatori per tant, en cada execució el MAE és molt probable que variï lleugerament.

Per fer les prediccions hem utilitzat l'algorisme de prediccions SVD. Aquest algorisme és equivalent a la factorització de matrius probabilístiques. És un model de factor latent que implica l'estimació de paràmetres per descendència de gradients estocàstics.

Finalment el que quedava per acabar l'algorisme és predir les dades que ens demanen en el fitxer de mostra i guardar-les amb el format adequat.

```
for uid, iid in sample_list.itertuples(index=False):
    pred = algo.predict(int(uid), int(iid), r_ui=0, verbose=True)
    pred_dict = {'userID' : uid, 'restaurantID' : iid, 'rating' : format(float(pred.est), '.4f')}
    pred_list.append(pred_dict)
```

Aquí es veu que es crida la funció predict per totes les parelles d'id d'usuari i id de restaurant.

4. Possibles millores del codi

Tot i haver obtingut un bon resultat respecte a la resta de puntuació, creiem que la línia a seguir per a millorar aquest resultat seria implementar un algorisme on ens permeti incloure més d'una variable independent per tal d'oferir més informació a l'algorisme, aquestes variables podrien ser la mitjana de cada usuari i/o restaurant, la desviació estàndard, la volatilitat de cada usuari i/o restaurant, etc. El paquet Surprise de Python, que és el que utilitzem en la solució, neix per a resoldre un problema similar on hi ha pel·lícules (ítems), usuaris que veuen les pel·lícules i puntuacions per cada una d'aquestes. El problema que resol és el que tenim nosaltres entre mans, és a dir, predir la puntuació dels usuaris que no han vist la pel·lícula en qüestió (enllaç al dataset de les pel·lícules). És per això que l'algorisme en el moment de fer l'entrenament, no permet cap altre paràmetre perquè està pensat per resoldre els problemes d'aquesta índole (tres columnes on la primera és l'usuari, la segona és l'ítem i la tercera la puntuació).

Cal dir que també hem pensat a dividir les dades segons la seva similitud basada en la distància del cosinus, però ens ha resultat impossible poder aconseguir aquest resultat perquè la memòria de l'ordinador no era capaç de processar totes les dades.

Finalment, després de comentar aquestes tècniques podem resumir que els nostres dos possibles camins que hem intentat resoldre per millorar el resultat ha estat:

- Utilitzar el mateix tipus de procediment però incloent més variables independents, és a dir dividir les dades en entrenament i test, però a l'hora de predir fer-ho amb més informació, amb més columnes de dades.
- Abans de fer l'algorisme de predicció i de dividir les dades en entrenament i test, agrupar els usuaris o restaurant en diferents grups basant-nos en la distància del cosinus.

5. Conclusions

En realitzar aquesta pràctica ens hem adonat de la importància d'escollir un bon algorisme de predicció i la complicació que té. En el nostre cas en tenir relativament poques dades hem pogut fer una petita prova amb diferents algorismes (agafant un conjunt de dades reduïdes) per tal de decantar-nos per l'algorisme que hem utilitzat al final. Per tant, això ens porta que per facilitar la tria de l'algorisme, prèviament s'ha de fer una bona anàlisi de les dades. Aquest estudi previ ens pot facilitar molt les coses a posteriori, ja que podem començar a trobar possibles tendències i començar a veure quin seria el camí a seguir, a més que també podem decidir quines dades ens poden servir com variables independents i quines no.