UNIVERSITÀ DEGLI STUDI
DI NAPOLI FEDERICO II

Università degli Studi di Napoli Federico II

Corso di Laurea Magistrale in Ingegneria Informatica

Elaborato in

BIG DATA ANALYTICS AND BUSINESS INTELIGENCE

e

BIG DATA ENGINEERING

HOMEWORK n.3

Candidati

Mattia Brescia M63001103

Luigi Garofalo M63001182

Prof: Giancarlo Sperlì

Anno accademico 2021/2022

Indice

Capitolo 1: Dataset characterization and Problem introduction 3

1.1 Introduzione al Problema 6

Capitolo 2: Processing Queries and algorithms based on Apache Spark 9

2.1 Calcolo delle statistiche 9

2.2 Soluzione proposta 12

Capitolo 1: Dataset characterization and Problem introduction

Il dataset di YELP è una collezione di attività, recensioni e dati degli utenti da utilizzare per scopi personali, educativi ed accademici. Il dataset è disponibile come file JSON dove sono presenti:

- 6,990,280 reviews
- 150,346 attività (businesses)
- 200,100 immagini
- 11 aree metropolitane
- 908,915 consigli
- 1,987,897 utenti
- >1.2 mln di attributi per le attività (parcheggio, ore etc)

Se si chiede ad uno statunitense cos'è Yelp molto probabilmente risponderà che non può farne a meno. Si danno giudizi e dalle recensioni su Yelp dipende infatti la scelta del locale dove andare a cena, del parrucchiere per il nuovo taglio di capelli, del dentista a cui affidarsi o del negozio in cui fare shopping, per citarne alcuni.

Yelp è simile a TripAdvisor ma recensisce qualunque cosa. Entrambe sono comunque piattaforme user generated content in cui cioè i contenuti sono prodotti dagli utenti che alimentano una comunità virtuale basata sul passaparola e sulla fiducia. Yelp batte nettamente TripAdvisor tanto da essere il sito di recensioni online più usato al mondo. Alla fine del 2016 Yelp vantava oltre 120 milioni di recensioni! Se si ha un'attività commerciale che si vuole far crescere non si può ignorare le sue potenzialità come strumento di marketing.

Come funziona YELP?

Si tratta di un servizio gratuito disponibile sia da fisso che da mobile attraverso un app che ospita recensioni, giudizi e commenti degli utenti sulle attività commerciali presenti sul territorio. Sfrutta quindi il marketing del passaparola (il cosidetto Wom) consentendo alle attività di espandersi ed avere più successo se lavorano bene ed offrono servizi validi e agli utenti di orientarsi più tranquillamente nella scelta tra le tante possibilità come se fossero giudati dal parere di un amico fidato. Ogni proprietario di un'attività commerciale (negozio, hotel, bar, centro di servizi etc.) può registrarla o "rivendicarla" per ottenere una scheda profilo dettagliata, completa e accattivante. Allo stesso tempo gli utenti non sono anonimi: ogni persona può creare una sua scheda con dati e foto e lasciare poi commenti sui luoghi in cui è stata che verranno contrassegnati a loro volta in base all'utilità. Yelp offre inoltre anche la possibilità di prenotare online gratuitamente il posto in ristoranti, hotel e servizi vari e modificare la prenotazione comodamente da pc o smartphone tramite app.

Dopo un'introduzione su YELP è opportuno analizzare la struttura del sottoinsieme del dataset utilizzato: ogni file del dataset è composto da un singolo tipo di oggetto, un oggetto JSON per riga.

Analizziamo il file **"business.json"**, il quale contiene le seguenti informazioni:

- address: string
- attributes: struct
    - AcceptsInsurance: string
    - AgesAllowed: string
    - Alcohol: string
    - Ambience: string
    - BYOB: string
    - BYOBCorkage: string
    - BestNights: string
    - BikeParking: string
    - BusinessAcceptsBitcoin: string
    - BusinessAcceptsCreditCards: string
    - BusinessParking: string
    - ByAppointmentOnly: string
    - Caters: string
    - CoatCheck: string
    - Corkage: string
    - DietaryRestrictions: string
    - DogsAllowed: string
    - DriveThru: string
    - GoodForDancing: string
    - GoodForKids: string
    - GoodForMeal: string
    - HairSpecializesIn: string
    - HappyHour: string

    - HasTV: string
    - Music: string
    - NoiseLevel: string
    - Open24Hours: string
    - OutdoorSeating: string
    - RestaurantsAttire: string
    - RestaurantsCounterService: string
    - RestaurantsDelivery: string
    - RestaurantsGoodForGroups: string
    - RestaurantsPriceRange2: string
    - RestaurantsReservations: string
    - RestaurantsTableService: string
    - RestaurantsTakeOut: string
    - Smoking: string
    - WheelchairAccessible: string
    - WiFi: string
- business_id: string
- categories: string
- city: string
- hours: struct
    - Friday: string
    - Monday: string
    - Saturday: string
    - Sunday: string
    - Thursday: string
    - Tuesday: string
    - Wednesday: string
- is_open: long
- latitude: double
- longitude: double
- name: string
- postal_code: string
- review_count: long
- stars: double
- state: string

Un altro file da analizzare è il **"review.json"** il quale contiene informazioni relative alle recensioni lasciate dagli utenti tra cui la data, l'ID del locale, il numero di stelle ed il testo della recensione vera e propria:

- business_id: string
- cool: long
- date: string
- funny: long
- review_id: string
- stars: double
- text: string
- useful: long
- user_id: string

“User.json” contiene tutte le informazioni relative ai clienti, quindi non solo il nome e l’identificativo ma contiene anche informazioni relative al tipo ed il numero di complimenti che l’utente riceve, in generale :

- average_stars: double
- compliment_cool: long
- compliment_cute: long
- compliment_funny: long
- compliment_hot: long
- compliment_list: long
- compliment_more: long
- compliment_note: long
- compliment_photos: long
- compliment_plain: long
- compliment_profile: long
- compliment_writer: long
- cool: long
- elite: string
- fans: long
- friends: string
- funny: long
- name: string
- review_count: long
- useful: long
- user_id: string
- yelping_since: string

1.1 Introduzione al Problema

Un problema comune al giorno d’oggi è la monotonia. Spesso, la maggior parte delle persone si limita a frequentare sempre gli stessi locali per questioni di abitudine o perché magari si è indecisi su dove andare.
Nel momento in cui, però, una persona sceglie di uscire da questa routine molto frequentemente cerca “ispirazione” sui nuovi locali da visitare chiedendo ad un amico o magari cercando su Internet, guardando qualche storia su Instagram etc.
Di conseguenza, capiamo che nel momento in cui una persona sceglie di voler cambiare itinerario cerca di affidarsi al giudizio altrui, che può essere quello di un amico oppure quello di qualcun altro sul web che l’utente stesso reputa “affidabile” oppure interessante.

Sulla base di questi ragionamenti si è cercato di creare un algoritmo per suggerire ad un utente in cerca di un locale da visitare basato sui locali recensiti positivamente dagli amici dell' utente stesso oppure suggerendogli locali relativamente "buoni" che sono stati recensiti positivamente da utenti "affidabili".

In primo luogo sono state analizzate le caratteristiche degli utenti presenti all'interno del dataset. È saltato subito all'occhio che la maggior parte degli utenti del dataset non sono dei "buoni" utenti, dove per "buoni" si intendono utenti abbastanza attivi sulla piattaforma e che rispettano determinati parametri:

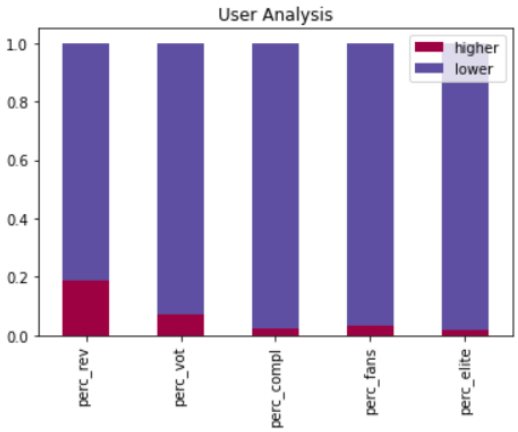
- Hanno un numero di recensioni effettuate abbastanza alto
- Hanno un numero di complimenti ricevuti dagli altri utenti abbastanza alto
- Sono utenti élite, e lo sono da abbastanza tempo
- Hanno un numero di voti inviati alle recensioni degli altri utenti abbastanza alto
- Hanno un numero di fans abbastanza alto

Come è possibile immaginare, non è facile definire questo "abbastanza" e per farlo si è scelto di considerare la media dei parametri sopra elencati. Analizzando il dataset però, la maggior parte degli utenti inoltre presenta quasi tutti valori nulli nei campi definiti precedentemente, di conseguenza considerare anche questi utenti nei conti avrebbe fatto abbassare la media, includendo negli utenti relativamente "buoni" anche quelli che effettivamente tanto "buoni" non sono.

I valori ottenuti sono stati quindi i seguenti:

Nome	Valore
N° medio di review fatte dagli utenti	7.1062173954398125
N° medio di voti inviati dagli utenti	23.395044779693364
N° medio di complimenti ricevuti dagli utenti	47.57004328029218
N° medio di fans degli utenti	108.79775254245776
N° medio di anni élite degli utenti	3.7343691747626044

Tramite queste statistiche, infatti, è stato possibile calcolare la percentuale di utenti presente all'interno del dataset che si trova al di sotto o al di sopra di queste medie, ed il risultato è stato riportato nel grafico seguente.



Da questo grafico si evince quindi che vi è un'alta percentuale di utenti che si trova al di sotto di tutte queste statistiche, maggiormente per le ultime 3.

Prima di proseguire nella descrizione della risoluzione del problema e delle tecniche implementate è opportuno focalizzarsi su come sono state calcolate queste statistiche.

Capitolo 2: Processing Queries and algorithms based on Apache Spark

2.1 Calcolo delle statistiche

Dopo aver caricato il dataset, si è deciso di ricavare le statistiche elencate nel capitolo precedente. Per quanto riguarda i campi “fans” e “review_count” essendo già presenti nel dataset è stato facile ricavarne la media tramite operazioni di aggregazione, mentre per quanto riguarda il numero di voti, di complimenti e di anni élite è stato necessario fare qualche operazione in più.

```
1.  #calcolo numero totale di utenti sulla piattaforma

2.  TotalUsers=userDS.count()

3.  #calcolo il numero medio di fans degli utenti

4.  avg_fans=userDS.select('fans').where(userDS['fans']!=0).agg({'fans':'avg'}).toPandas().to_numpy
    ().ravel()[0]

5.  #la percentuale di persone che hanno un numero di follower al di sotto della media:

6.  num_less_avg_fans=userDS.select("fans").where(userDS["fans"]<avg_fans).count()

7.  perc_inf_fans=num_less_avg_fans/TotalUsers

8.

9.  #calcolo il numero totale di complimenti ricevuti da ogni utente

10. compliments=userDS.withColumn("COMPLIMENTS",userDS["compliment_cool"]+userDS["compliment_funny"
    ]+userDS["compliment_hot"]+ userDS["compliment_more"]+
    userDS["compliment_profile"]+userDS["compliment_cute"]+
    userDS["compliment_list"]+userDS["compliment_note"]+userDS["compliment_plain"]+
    userDS["compliment_writer"]+userDS["compliment_photos"]).select("user_id","COMPLIMENTS")

11. #ne calcolo il valore medio

12. avg_compliments=compliments.select('COMPLIMENTS').where(compliments['COMPLIMENTS']!=0).agg({'CO
    MPLIMENTS':'avg'}).toPandas().to_numpy().ravel()[0]

13. #calcolo la percentuale di utenti che hanno un numero di complimenti al di sotto della media

14. num_less_avg_comp=compliments.select("user_id").where(compliments["COMPLIMENTS"]<avg_compliment
    s).count()

15. perc_inf_compl=num_less_avg_comp/TotalUsers
```

```
16.
17.
18.
19. #calcolo il numero totale di voti inviati dagli utenti
20. sent_votes=userDS.withColumn("SENT_VOTES",userDS["useful"]+userDS["funny"]+userDS["cool"]).sele
    ct("user_id","SENT_VOTES")
21. #ne calcolo il valore medio
22. avg_votes=sent_votes.select('SENT_VOTES').where(sent_votes['SENT_VOTES']!=0).agg({'SENT_VOTES':
    'avg'}).toPandas().to_numpy().ravel()[0]
23. #calcolo la percentuale di persone con un numero di voti minore della media
24. num_less_avg_votes=sent_votes.select("user_id").where(sent_votes["SENT_VOTES"]<avg_votes).count
    ()
25. perc_inf_vot=num_less_avg_votes/TotalUsers
26.
27. #calcolo il numero medio di review fatte dagli utenti
28. avg_review=userDS.select('review_count').where(userDS['review_count']!=0).agg({'review_count':'
    avg'}).toPandas().to_numpy().ravel()[0]
29. #calcolo la percentuale degli utenti che hanno un numero di recensioni al di sotto della media
30. num_less_avg_rev=userDS.select("user_id").where(userDS["review_count"]<avg_review).count()
31. perc_inf_rev=num_less_avg_rev/TotalUsers
32.
33. perc_inf=(perc_inf_rev,perc_inf_vot,perc_inf_compl,perc_inf_fans)
34. #è possibile notare che la maggior parte degli utenti si trovano al di sotto di queste medie
```

Per quanto riguarda quindi il numero di complimenti, poiché all'interno del dataset vi erano vari attributi relativi ai complimenti, i quali descrivevano il numero di complimenti ricevuti divisi per categorie:

- Cool
- Funny
- More
- Profile
- Hot
- Cute
- List
- Note
- Plain
- Writer
- Photos

Si è scelto di creare una colonna denominata 'COMPLIMENTS' che contiene la somma di tutti questi complimenti, e successivamente è stato estratto il valore medio.

Un procedimento simile è stato fatto per il numero di voti inviati, aggiungendo la colonna 'SENT_VOTES' contenente la somma dei voti inviati dai vari utenti distinti in 3 tipologie:

- Useful
- Funny
- Cool

Per quanto riguarda il numero di anni élite, il discorso si fa più articolato: poiché il campo élite di ogni utente era rappresentato da una stringa contenente gli anni in cui l'utente è stato un'utente élite (es: '2008,2009,2010..') abbiamo dovuto effettuare delle operazioni per ottenere il numero di questi anni.
Innanzitutto, all'interno di queste stringhe vi era un problema: nel momento in cui si fa riferimento all'anno 2020 , nella lista viene presentato il valore '20,20' quindi è stato deciso di "rimpiazzare" questa parte di stringa con '2020' per evitare che il conteggio fosse effettuato in maniera sbagliata. Dopodiché la stringa viene spezzettata, ottenendo una lista del tipo [2020,2021,..] e viene calcolata la lunghezza di quest'ultima per ottenere il numero di anni élite.

```
1.  #Creo dataframe con il numero di anni elite per ogni utente
2.  rdd2=userDS.rdd.map(lambda x:[x['user_id'],
    len((x['elite']).replace("20,20","2020,").replace(","," ").split())])
3.
4.  schema = StructType([
5.       StructField('user_id', StringType(), True),
6.      StructField('EliteYears', IntegerType(), True)
7.  ])
8.  df = spark.createDataFrame(rdd2,schema)
9.
10. #calcolo dataset completo
11. Complete1=userDS.join(compliments,on=["user_id"],how='left')
12. Complete2=Complete1.join(df,on=["user_id"],how='left')
13. Complete=Complete2.join(sent_votes,on=["user_id"],how='left')
14.
15. #calcolo il numero medio di Eliteyears degli utenti
16. avg_elite=Complete.select('EliteYears').where(Complete['EliteYears']!=0).agg({'EliteYears':'avg
    '}).toPandas().to_numpy().ravel()[0]
17. #calcolo la percentuale degli utenti che hanno un numero di recensioni al di sotto della media
18.
```

user_id	fans	friends	name	review_count	yelping_since	COMPLIMENTS	EliteYears	SENT_VOTES
04ZoQVzOSLkSmnFYj...	2	fN6yGf17ZVHh6vMgL...	Aditi	73	2013-07-08 21:35:12	36	2	68
2Qs9PUgkLkufo_NUE...	0	eGwVrwaot_mH34Y1_...	Tony	1	2015-12-08 03:02:34	0	0	0
6ZOsQRBvZjAzcDgo0...	0	DOkMjn8tVj9Nm6tIR...	Chesley	1	2014-08-23 00:39:10	0	0	0
8HKx7p15nsuaoVWER...	0	zSI5ymu_U85dRAZIP...	Sarah	5	2016-12-26 22:24:49	1	0	4
8Jao7KkVZq3HTPVXW...	0	_2x1BfGK24JqJNdGD...	Carrie	43	2015-02-25 08:20:10	3	2	128
8NdmQrpt6m5zNEi2C...	1	cuuniqnZRlS_EPosx...	Clare	18	2016-03-31 22:58:07	1	0	35
8tt9XbWt4s5ZhG6Po...	0	BRq3qXQ02CpZUcVVz...	Kathy	10	2017-04-20 21:29:42	0	0	10
Ab8IxfvDofE7thVMD...	0	None	P	8	2012-02-05 22:03:03	1	0	10
BDOCu_rbF75vMK0st...	0	None	Tiffany	40	2016-11-27 15:03:55	0	0	15
C9Cvyl5a-_v99OzJG...	0	None	Antwoine	6	2015-04-12 18:15:30	0	0	12

Figura 1: Dataset Post-processing

2.2 Soluzione proposta

Arrivati a questo punto è stato possibile ottenere l'insieme degli utenti relativamente "affidabili", nonché quelli che avevano tutti questi parametri al di sopra della media.

```
1.  goodUsers=Complete.select("user_id").where((Complete['fans']>=avg_fans) &
    (Complete['review_count']>=avg_review)

2.                                          & (Complete['COMPLIMENTS']>=avg_compliments) &
    (Complete['SENT_VOTES']>=avg_votes)

3.                                            &( Complete['EliteYears']>=avg_elite))

4.
```

user_id
E_9WOCYIGHtT2nbcY...
FlXBpK_YZxLo27jcM...
jO44Apni7iJZVVK4H...
SbrDl9B69acb3PEiT...
sZdy2AQD0gn7RMAJc...
oQwFLohuwcJYMd0Wh...
4ZfcCa4m5RWvO4EFz...
dioXbYFdMCyE7zCK9...
qCNZXu0nA1m9_qQDS...
a_TVmbLLYVk91lNlR...

Figura 2:Good Users

Successivamente è stata estratta la lista dei locali recensiti positivamente (con almeno 3 stelle) dai goodUsers.

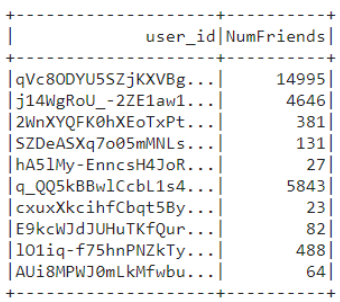
```
1.  #Ottengo la lista dei locali recensiti dai GoodUsers

2.  localss=goodUsers.join(reviewDS,on=['user_id'],how='left').select('user_id','business_id','star
    s')

3.  #Ottengo la lista del locali recensiti dai goodUsers con >3 stelle

4.  localss3stars=localss.select('user_id','business_id').where(localss['stars']>3)

5.
```

user_id	business_id
--_H9j6ggxvqhh9nP...	j0tHKpa2I2pj3uuxC...
--_H9j6ggxvqhh9nP...	nDD_6elPOfDe0eRk9...
--_H9j6ggxvqhh9nP...	lgrrHTUt96Ic_pczI...
--_H9j6ggxvqhh9nP...	ubkCYzHu9ZnGi3vOQ...
--_H9j6ggxvqhh9nP...	5BWh2ViME8lAm1G1S...
--_H9j6ggxvqhh9nP...	POfVSVLxZ-6sYCBkQ...
--_H9j6ggxvqhh9nP...	SXAQjjOCIuSJtthvW...
--_H9j6ggxvqhh9nP...	NzIvofRG2K0JFzvsa...
--_H9j6ggxvqhh9nP...	_M9C-YOf5uFnCMe6_...
--_H9j6ggxvqhh9nP...	YYhtoCEpgBgTAQ-ND...

Figura 3:Good Locals

Dopo questo procedimento, è stato creato un dataset contenente l'identificativo dell'utente ed il numero di amici di quest'ultimo seguendo una logica simile a quella utilizzata per il calcolo degli anni élite, poiché il campo friends presentava una stringa con tutti gli identificativi degli utenti amici dell'utente. Di conseguenza questa lista è stata spezzettata e calcolandone la lunghezza è stato possibile ottenere il numero di amici posseduti da ogni utente.

```
1.  rdd3=userDS.rdd.map(lambda x:[x['user_id'], len((x['friends']).replace(","," ").split())])
2.
3.  schema = StructType([
4.       StructField('user_id', StringType(), True),
5.      StructField('NumFriends', IntegerType(), True)
6.  ])
7.  df = spark.createDataFrame(rdd3,schema)
8.
```



user_id	NumFriends
qVc8ODYU5SZjKXVBg...	14995
j14WgRoU_-2ZE1aw1...	4646
2WnXYQFK0hXEoTxPt...	381
SZDeASXq7o05mMNLs...	131
hA5lMy-EnncsH4JoR...	27
q_QQ5kBBwlCcbL1s4...	5843
cxuxXkcihfCbqt5By...	23
E9kcWJdJUHuTKfQur...	82
lO1iq-f75hnPNZkTy...	488
AUi8MPWJ0mLkMfwbu...	64

Figura 4:Number of Friends

In seguito è stato estratto un'utente in maniera randomica all'interno del dataset con lo scopo di suggerire a quest'ultimo un insieme di locali seguendo l'algoritmo precedentemente descritto.

```
1.  Users=userDS.select('user_id')
2.  person=Users.rdd.takeSample(False,1,seed=0)
3.  person_id=person[0][0]
4.
```

```
[Row(user_id='yHdOV4jFu8frBB5z4hkPRQ')]
```

Ottenuto l'identificativo dell'utente è stato calcolato il numero di amici che quest'ultimo possiede. Viene creato quindi un dataset contenente i locali recensiti positivamente dai goodUsers, e da quest'ultimo vengono eliminati i locali che l'utente stesso ha recensito negativamente con lo scopo di non suggerirgli ulteriormente un locale che magari non ha gradito.
La soluzione proposta, a questo punto, distingue 2 casi:

- Il caso in cui l'utente non abbia amici
- Il caso in cui l'utente abbia almeno 1 amico

Nel 1° caso, poiché l'utente non dispone di una lista di amici da cui attingere per ottenere una lista di locali probabilmente affini ai interessi, viene suggerito all'utente un insieme di locali recensiti positivamente dai goodUsers e che contemporaneamente non sono stati recensiti negativamente dall'utente stesso.

```
1.  NumeroAmici=df.select('NumFriends').where(df['user_id']==person_id).toPandas().to_numpy().ravel
    ()[0]

2.  NomiLocali=businessDS.select('name','business_id','stars')

3.  Locali_suggeriti=localss3stars.select('business_id').distinct()

4.   #devo eliminare i locali recensiti dall'utente in maniera negativa

5.  NotGoodLocals=reviewDS.select('user_id','business_id').where((reviewDS['user_id']==person_id)
    &(reviewDS['stars']<3))

6.  Locali_suggeriti=Locali_suggeriti.join(NomiLocali,on=['business_id'],how='left').join(NotGoodLo
    cals,on=['business_id'],how='leftanti')

7.  if(NumeroAmici==0):

8.      print('Poichè l utente non ha amici , gli verranno suggeriti locali in funzione degli
    utenti migliori')

9.      print('.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.')

10.     Locali_suggeriti.show()

11.     Mappa=Locali_suggeriti.limit(10)

12.
```

Nel 2° Caso, innanzitutto si identificano tutti gli identificativi degli amici dell'utente e vengono salvati all'interno di una lista. In seguito, vengono individuati tutti i locali recensiti positivamente (con più di 3 stelle) dagli amici dell'utente, andando ad eliminare tramite una left-anti join tutti i locali che l'utente non ha gradito.
A questo punto si è ragionato su di un'altra problematica: anche se l'utente ha una lista di amici non vuota, non è detto che quest'ultimi abbiano recensito almeno 10 locali con più di 3 stelle. Per ovviare a questo problema è stato calcolato il numero di locali recensiti dagli amici dell'utente, se questo numero è minore di 10 allora ne viene calcolata la differenza.

In conclusione, all'utente verranno suggeriti un numero di locali pari al numero di locali recensiti dagli amici dell'utente facendo riferimento ai locali che gli amici hanno gradito, la restante parte verrà rimpiazzata con i locali recensiti positivamente dai goodUsers. In alternativa se il numero di locali recensiti dagli amici è almeno 10 allora verranno presentati 10 locali all'utente che sono stati recensiti positivamente dai suoi amici.

```
1.  if(NumeroAmici>0):
2.      print('Lista dei locali consigliati')
3.      NomiLocali=businessDS.select('name','business_id','stars')
4.      frnds=userDS.select('user_id','friends').where(userDS['user_id']==person_id)
5.      frnds=frnds.join(df,on='user_id',how='left')
6.      h=frnds.select('friends').toPandas().to_numpy().ravel()[0]
7.      listaAmici = h.replace(",", "") \
8.      .split()
9.      #vedoi locali ben recensiti dagli amici dell utente
10.
    localiBenRecensiti=reviewDS.select('business_id','user_id').filter(reviewDS['user_id'].isin(lis
    taAmici)).filter(reviewDS['stars']>3)
11.
    localss=localiBenRecensiti.join(NomiLocali,on='business_id',how='left').join(NotGoodLocals,on=[
    'business_id'],how='leftanti')
12.     localss=localss.select('business_id','name','stars').where(localss['stars']>3)
13.     numLoc=localss.count()
14.     if(numLoc<10): #se sono meno di 10, il rimanente glielo suggerisco tramite i locali
    recensiti bene dai goodUsers
15.         numAdd=10-numLoc
16.         print(f'il numero di locali recensiti dagli amici è {numLoc} \n')
17.         print(f'Di conseguenza verranno proposti {numAdd} locali scelti dalla lista dei
    goodUsers \n')
18.         #localss3stars=localss3stars.select('business_id','name','stars') #rimuovo la colonna
    del user_id dai localss che non mi serve
19.         temp=Locali_suggeriti.limit(numAdd)
20.         final=temp.union(localss)
21.         final.show()
22.         Mappa=final
23.     else:
24.
25.         localss.limit(10).show()
```

```
26.         Mappa=localss.limit(10)

27.
```

Output:

```
il numero di locali recensiti dagli amici è 9

Di conseguenza verranno proposti 1 locali scelti dalla lista dei goodUsers
+--------------------+--------------------+-----+
|         business_id|                name|stars|
+--------------------+--------------------+-----+
|L-VNs3YquPGKVsXl2...|Peacemaker Lobste...|  4.0|
|Qsf6aYZHrP5SLioZX...|KO Modern Korean ...|  4.0|
|ctHjyadbDQAtUFfkc...|               Zahav|  4.5|
|PSo_C1Sfa13JHjzVN...|Indian Walk Veter...|  5.0|
|27TyycLr6a5ju7ROX...|         Osaka Sushi|  4.0|
|GsOc2dLx08yIdTyGV...|Trattoria Rosa Bi...|  4.0|
|3ldp8B52AOzNSFnPM...|           Fette Sau|  4.0|
|MsYHQD2-2kbiQmKbI...| Jahan Kabob & Grill|  4.0|
|pABQ5benw5OkuARYH...| Mill Street Cantina|  4.0|
|Ek9iI8uJ7U3IDzVWr...|Giumarello's Rest...|  4.0|
+--------------------+--------------------+-----+
```

Figura 5:Output

Per concludere, vengono raccolte le informazioni di latitudine e longitudine dei locali precedentemente suggeriti all'utente con lo scopo di fornirgli una vera e propria mappa, implementata grazie alla libreria python "folium".

```
1.  locations=Mappa.select('business_id').join(businessDS,on='business_id',how='left').select('name
    ','latitude','longitude')

2.  names=locations.select('name').toPandas().to_numpy().ravel()

3.  latitudes=locations.select('latitude').toPandas().to_numpy().ravel()

4.  longitudes=locations.select('longitude').toPandas().to_numpy().ravel()

5.  map = folium.Map(location=[latitudes.mean(), longitudes.mean()], zoom_start=14,
    control_scale=True)

6.  for i in range(len(latitudes)):

7.      folium.Marker([latitudes[i], longitudes[i]], popup=names[i]).add_to(map)

8.  map

9.
```