UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Corso di Laurea Magistrale in Ingegneria Informatica

Dipartimento di Ingegneria Elettrica e delle tecnologie dell'informazione

Terzo Homework del corso di

BIG DATA ENGINEERING

Professore:
Giancarlo Sperlì

Candidati:
Benfenati Domenico M63001165
Carandente Vincenzo M63001229

ANNO ACCADEMICO 2021/2022
Secondo Semestre

Indice

Capitolo 1

Introduzione

1.1 Business Understanding

Tale elaborato ha l'obiettivo di processare un insieme di dati relativi alle recensioni contenute all'interno del open dataset della piattaforma **Yelp.com**.
Lo scopo delle elaborazioni effettuate è quello di mettere in evidenza informazioni esplicite, correlando dati provenienti da diverse sorgenti. Le informazioni prodotte possono essere quindi visualizzate facilmente da un tradizionale tool di Business Intelligence.
Il vantaggio di effettuare l'elaborazione a monte risiede nella possibilità di utilizzare uno strumento di calcolo distribuito, che consente di processare grandi moli di dati in maniera efficiente.
L'elaborazione è stata condotta mediante **Apache Spark**, noto framework impiegato per il calcolo distribuito, ampiamente utilizzato in applicazioni di Big Data. Nello specifico, è stato utilizzato **PySpark**, APIs Python che permette di scrivere applicazioni Spark.
Il modulo principalmente utilizzato è **Spark SQL**, il quale permette di processare dati strutturati in modo distribuito, fornendo un'astrazione di alto livello chiamata *DataFrame*.
La potenza di calcolo è stata infine fornita dalla piattaforma cloud **Kaggle**. Essa rappresenta un ambiente **web-based** che fornisce gestione automatica di cluster per l'esecuzione di calcolo distribuito ed un ambiente integrato di sviluppo di notebook Python.

1.2 Data Understanding

Nella sezione corrente si vogliono descrivere con maggiore precisione i dati a disposizione, i quali possono essere divisi in differenti categorie: **Business**, **Review**, **Users**, **Checkin**, **Tips**. È bene sottolineare che, dovendo implementare un database documentale, i dati utilizzati per popolarlo provengono da file in formato *.json*.

1.2.1 Business

La prima categoria di dati consiste in 150346 istanze, e include tutte le informazioni relative agli esercizi commerciali che sono inclusi all'interno delle classifiche Yelp.
In particolare, il file considerato nell'elaborazione è *yelp_academic_dataset_business.json* e contiene informazioni riguardo i vari esecizi commerciali quali posizione, città, valutazione in numero di stelle, numero di recensioni ricevute, e altre informazioni riguardo gli "attributi" del locale.

1.2.2 Review

La seconda categoria di dati contiene circa 6990280 istanze, e include tutte le informazioni relative alle recensioni sulla piattaforma Yelp. In particolare il file è *yelp_academic_dataset_review.json* e contiene l'identificativo, l'id dell'utente e dell'esercizio commerciale, la data, il numero di stelle attribuite, il testo della recensione e dei interi che specificano il numero di voti utili, divertenti e hot che altri utenti hanno attributo a tale recensione.

1.2.3 User

La terza categoria di dati conta circa 1987897 documenti, e include tutte le informazioni relative agli utenti presenti all'intenro della piattaforma Yelp.
In particolare il file è *yelp_academic_dataset_user.json* e contiene l'identificativo dell'utente, il nome, la data d'iscrizione, il numero di recensioni da esso effettuate, una lista di amici ad esso collegati, e una serie di valori interi indicanti il conteggio di differenti tipologie di complimenti ricevuti dall'utente.

1.2.4 Checkin

La quarta categoria di dati aggrega circa 131930 istanze, e include tutte le informazioni relative ai check-in degli utenti all'interno degli esercizi commerciali.
In particolare il file è *yelp_academic_dataset_checkin.json* e contiene l'identificativo del locale e una lista di date corrispondenti ai checkin effettuati in quell'esercizio. Tale categoria non è risultata utile ai fini dell'analisi, e si è quindi scelto di non considerare tali dati.

1.2.5 Tip

La quinta categoria di dati conta 908915 elementi, e include tutte le informazioni relative ai suggerimenti inseriti dagli utenti all'interno della piattaforma Yelp.
In particolare il file considerato nell'elaborazione è *yelp_academic_dataset_tip.json* e contiene informazioni circa l'utente che ha inserito il suggerimento, il locale a cui è destinato, e la data di inserimento, nonchè il testo in chiaro inserito dall'utente. Tale categoria non è risultata utile ai fini dell'analisi, e si è quindi scelto di non considerare tali dati.

Capitolo 2

Data Processing

Nel capitolo corrente sono riportate le operazioni di processing effettuate sui dati al fine di estrarne informazioni utili.

2.1 Avvio e installazione di Spark

2.2 Lettura dei File

Dal momento che i dati non vengono aggiornati con una frequenza elevata, si è deciso di accedere ad essi direttamente tramite la repository all'interno di Kaggle, accessibile al seguente link.
Ciascun file è stato dunque importato all'interno di Spark tramite l'apposita funzione, come indicato di seguito.

```
1 #import della collection Review
2 yelp_review = spark.read.json('../input/yelp-dataset/yelp_academic_dataset_review.json')
3 # import della collection Business
4 yelp_business = spark.read.json('../input/yelp-dataset/yelp_academic_dataset_business.
      json')
5 # import della collection User
6 yelp_user = spark.read.json('../input/yelp-dataset/yelp_academic_dataset_user.json')
```

Tramite tali operazioni si sono costruiti degli oggetti Spark DataFrame, contenenti ogni campo presente all'interno del file *.json*.

2.3 Query effettuate

2.3.1 Top 10 Business per numero di recensioni

Si è voluto verificare quante fossero i business che mostrassero il maggior numero di recensioni con almeno tre stelle.

Si procede quindi ad effettuare un primo filtraggio delle recensioni, per numero di stelle superiore a 3, tramite i seguenti comandi:

```
rec_tre_stelle = yelp_review.filter('stars >3')
rec_group = rec_tre_stelle.groupby('business_id').count()
rec_sort = rec_group.sort('count',ascending=False)
```

Successivamente si va ad effettuare la join tra il dataframe delle reviews e quello dei business, da cui si è scelto di estrarre solo alcune colonne, quali nome, identificativo, e tipologie di categorie, tramite i seguenti comandi:

```
business = yelp_business.select('business_id','name','categories')
rec_join_buss = business.join(rec_sort,'business_id','inner')
rec_join_sort = rec_join_buss.sort('count', ascending=False)
top10_buss_per_rec = rec_join_sort.limit(10)
top10_buss_per_rec.show()
```

Si produce quindi il seguente output:

```
+--------------------+--------------------+--------------------+-----+
|         business_id|                name|          categories|count|
+--------------------+--------------------+--------------------+-----+
|_ab50qdWOk0DdB6XO...|   Acme Oyster House|Live/Raw Food, Se...| 5982|
|ac1AeYqs8Z4_e2X5M...|        Oceana Grill|Restaurants, Seaf...| 5865|
|GXFMD0Z4jEVZBCsbP...|Hattie B's Hot Ch...|American (Traditi...| 5425|
|ytynqOUb3hjKeJfRj...|Reading Terminal ...|Candy Stores, Sho...| 5380|
|oBNrLz4EDhiscSlbO...|Ruby Slipper - Ne...|Restaurants, Amer...| 4369|
|_C7QiQQc47AOEv4PE...|  Commander's Palace|French, Restauran...| 4008|
|GBTPC53ZrG1ZBY3DT...|                Luke|German, Restauran...| 3726|
|I_3LMZ_1m2mzR0oLI...|  Pappy's Smokehouse|Food, Restaurants...| 3575|
|6a4gLLFSgr-Q6CZXD...|              Cochon|Cajun/Creole, Sea...| 3567|
|gTC8IQ_i8zXytWSly...|      Cochon Butcher|Shopping, Butcher...| 3538|
+--------------------+--------------------+--------------------+-----+
```

2.3.2 Conteggio categoria con più business

Si vuole verificare quali siano le categorie più popolari all'interno del dataset. Si procede quindi a esplodere per ogni business le categorie ad esso associate, tramite uno split del campo "categories" della collection, usando il seguente set di comandi:

```
buss_cat_split = yelp_business.select('business_id','categories').withColumn("categories
    ", explode(split("categories", ",")))
buss_cat_split.show(10)
```

Si ottiene il seguente output:

```
+--------------------+--------------------+
|         business_id|          categories|
```

```
+--------------------+--------------------+
|Pns2l4eNsfO8kk83d...|             Doctors|
|Pns2l4eNsfO8kk83d...|Traditional Chine...|
|Pns2l4eNsfO8kk83d...|Naturopathic/Holi...|
|Pns2l4eNsfO8kk83d...|         Acupuncture|
|Pns2l4eNsfO8kk83d...|    Health & Medical|
|Pns2l4eNsfO8kk83d...|       Nutritionists|
|mpf3x-BjTdTEA3yCZ...|    Shipping Centers|
|mpf3x-BjTdTEA3yCZ...|      Local Services|
|mpf3x-BjTdTEA3yCZ...|            Notaries|
|mpf3x-BjTdTEA3yCZ...|     Mailbox Centers|
+--------------------+--------------------+
```

Successivamente si vanno a contare quante siano le categorie univoche presenti nel dataset, tramite il comando:

```
conteggio_categorie = yelp_business.select('business_id','categories').withColumn("
    categories", explode(split("categories", ", "))).select('categories').distinct().
    count()
print(f'Numero di categorie univoche: {conteggio_categorie}')
```

Tale esecuzione produce il seguente output:

```
Numero di categorie univoche: 1311
```

In seguito è necessario contare il numero di business presenti per ogni categoria univoca trovata, tramite i comandi:

```
buss_cat = yelp_business.select('business_id','categories').withColumn("categories",
    explode(split("categories", ", ")))
buss_cat.groupBy('categories').count().show()
```

Si produce il seguente output:

```
+--------------------+-----+
|          categories|count|
+--------------------+-----+
|      Dermatologists|  336|
|    Historical Tours|  136|
|      Paddleboarding|   98|
|    Hot Air Balloons|   12|
| Mobile Home Dealers|    8|
|             Beaches|  122|
|     Pet Photography|   22|
|       Skating Rinks|  111|
|       Data Recovery|  121|
|        Aerial Tours|   12|
|         Boat Repair|   78|
```

```
|              Fondue|   27|
|       Videographers|   73|
|   Pet Waste Removal|   10|
|Faith-based Crisi...|    1|
|         Hobby Shops|  552|
|            Day Spas| 1997|
|            Honduran|   24|
|               Reiki|  201|
|       Nephrologists|    2|
+--------------------+-----+
```

A valle di tale conteggio si è scelto di ordinare tale dataframe Spark per conteggio e selezionare solo le prime venti categorie per numero di locali. Fatto ciò si è trasferito il dataframe su **Pandas**, libreria nativa di Python che permette di elaborare strutture tabellari, anch'esse chiamate DataFrame, e di produrre alcuni grafici.

Le operazioni eseguite sono state le seguenti:

```
buss_cat_count = buss_cat.groupBy('categories').count()
top20_buss_cat_count = buss_cat_count.sort("count", ascending=False).limit(20)

top20_buss_cat_pandas = top20_buss_cat_count.toPandas().set_index('categories')

# grafico del numero di locali per ogni categoria
top20_buss_cat_pandas.plot.barh(figsize=(12,6)).invert_yaxis()
plt.title('Numero di Business per categoria')
plt.xlabel("Totale Business")
plt.ylabel("Categorie")
plt.show()
plt.savefig('num_business_per_cat.png')
```

Tale esecuzione ha prodotto il seguente grafico:

2.3.3 Monitoraggio recensioni elite

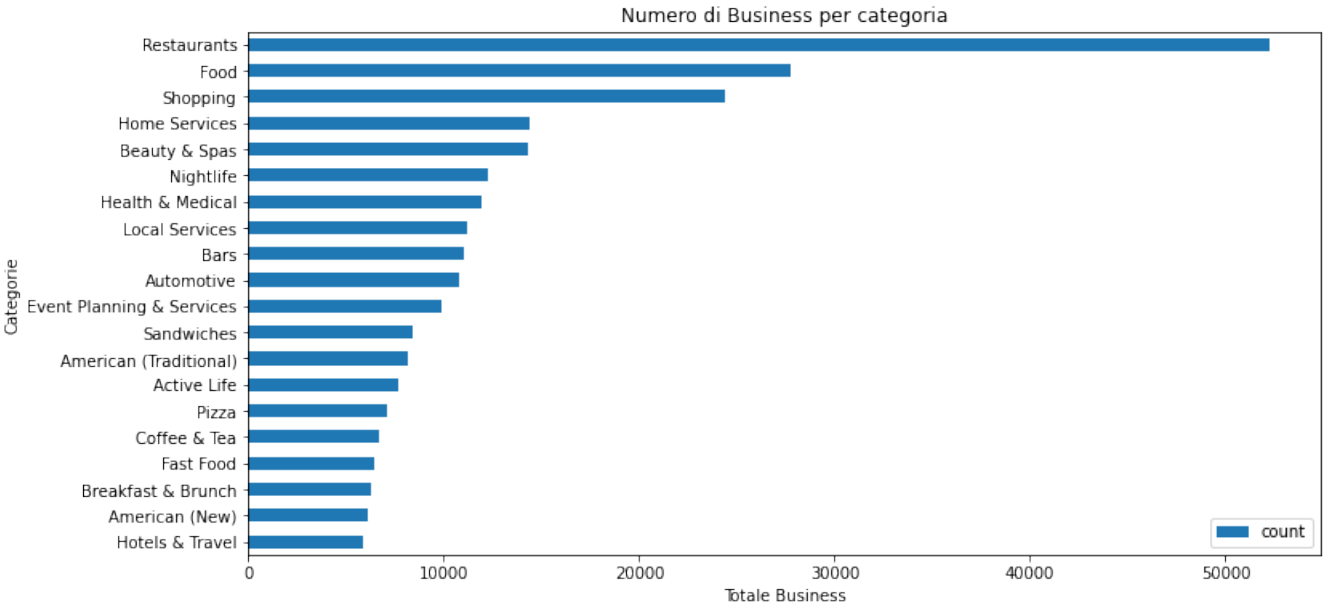
Si vuole verificare la percentuale di distacco delle recensioni degli utenti elite dalla media del locale, per verificare quanto tali recensioni siano estreme o concordi alla media del business.

Si procede quindi a rinominare alcune colonne all'interno delle collection, relative alle stelle, tramite i seguenti comandi:

```
yelp_business = yelp_business.withColumnRenamed('stars',"business_stars")
yelp_review = yelp_review.withColumnRenamed('stars',"review_stars")
```

Successivamente si effettua una join tra le collection, scartando gli utenti che non sono stati classificati come elite in alcun anno, tramite i comandi:

```
buss_join_rev = yelp_business.join(yelp_review, on=['business_id'], how='inner')
```




```
2
3 buss_join_rev_join_user = yelp_user.join(buss_join_rev, on=['user_id'], how='inner')
4
5 buss_join_rev_join_user.select('business_id','business_stars','review_stars','user_id','
    elite').sort('business_id','user_id').filter(buss_join_rev_join_user['elite'] != '')
    .show(10)
```

Si ottiene quindi il seguente output:

```
+--------------------+--------------+------------+--------------------+--------------------+
|         business_id|business_stars|review_stars|             user_id|               elite|
+--------------------+--------------+------------+--------------------+--------------------+
|---kPU91CF4Lq2-Wl...|           4.5|         4.0|i48cHEyRBl5g9_npY...|     2019,20,20,2021|
|---kPU91CF4Lq2-Wl...|           4.5|         5.0|qrCkKrEwQ-q9m1iWS...|2018,2019,20,20,2021|
|--0iUa4sNDFiZFrAd...|           3.0|         1.0|Dr-atSrDQURhT9GiH...|           2014,2015|
|--0iUa4sNDFiZFrAd...|           3.0|         4.0|j_orXueAYJBft-VUN...|2012,2013,2014,20...|
|--0iUa4sNDFiZFrAd...|           3.0|         3.0|mswloq-IjRr9yXLhB...|2011,2012,2013,20...|
|--0iUa4sNDFiZFrAd...|           3.0|         4.0|xbJlnyVkxuQq5jsG9...|2015,2016,2017,20...|
|--7PUidqRWpRSpXeb...|           2.0|         2.0|3YhG4h4Ok654iVfqd...|2013,2014,2015,20...|
|--7PUidqRWpRSpXeb...|           2.0|         1.0|5IY6Kb6BqNINbOoc1...|                2016|
|--7PUidqRWpRSpXeb...|           2.0|         1.0|Q5jOFJYhIsN8ouJ1r...|2009,2010,2011,20...|
|--7PUidqRWpRSpXeb...|           2.0|         4.0|r3QexFIhBXBT99can...|2011,2012,2013,20...|
+--------------------+--------------+------------+--------------------+--------------------+
```

Successivamente si vanno a determinare le percentuali di distacco tra la recensione dell'utente elite e le stelle medie presenti all'interno della collection business, tramite il calcolo della seguente formula:

$$DiffStars = \left| \frac{-(RevStars - BusStars)}{BusStars} \right| * 100$$

Il codice prodotto è il seguente:

```
buss_join_rev_join_user = buss_join_rev_join_user.withColumn('RevStar_BussStar_diff',(
    abs((buss_join_rev_join_user['review_stars']-buss_join_rev_join_user['business_stars
    '])*-1)/buss_join_rev_join_user['business_stars'])*100)
buss_join_rev_join_user = buss_join_rev_join_user.select('business_id','business_stars',
    'review_stars','RevStar_BussStar_diff','user_id','elite').sort('business_id').filter
    (buss_join_rev_join_user.elite!='')
buss_join_rev_join_user.show(10)
```

Si ottiene il seguente dataframe in output:

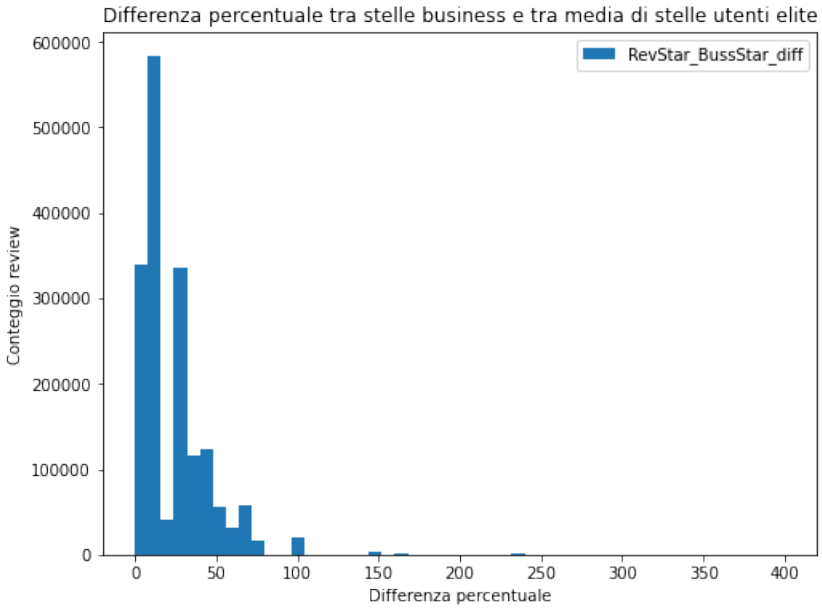
```
+--------------------+--------------+------------+--------------------+--------------------+--------------------+
|         business_id|business_stars|review_stars|RevStar_BussStar_diff|             user_id|               elite|
+--------------------+--------------+------------+--------------------+--------------------+--------------------+
|---kPU91CF4Lq2-Wl...|           4.5|         5.0|    11.11111111111111|qrCkKrEwQ-q9m1iWS...|2018,2019,20,20,2021|
|---kPU91CF4Lq2-Wl...|           4.5|         4.0|    11.11111111111111|i48cHEyRBl5g9_npY...|     2019,20,20,2021|
|--0iUa4sNDFiZFrAd...|           3.0|         4.0|    33.33333333333333|j_orXueAYJBft-VUN...|2012,2013,2014,20...|
|--0iUa4sNDFiZFrAd...|           3.0|         3.0|                 0.0|mswloq-IjRr9yXLhB...|2011,2012,2013,20...|
|--0iUa4sNDFiZFrAd...|           3.0|         4.0|    33.33333333333333|xbJlnyVkxuQq5jsG9...|2015,2016,2017,20...|
|--0iUa4sNDFiZFrAd...|           3.0|         1.0|    66.66666666666666|Dr-atSrDQURhT9GiH...|           2014,2015|
|--7PUidqRWpRSpXeb...|           2.0|         2.0|                 0.0|3YhG4h4Ok654iVfqd...|2013,2014,2015,20...|
|--7PUidqRWpRSpXeb...|           2.0|         4.0|               100.0|r3QexFIhBXBT99can...|2011,2012,2013,20...|
|--7PUidqRWpRSpXeb...|           2.0|         1.0|                50.0|5IY6Kb6BqNINb0oc1...|                2016|
|--7PUidqRWpRSpXeb...|           2.0|         1.0|                50.0|Q5jOFJYhIsN8ouJ1r...|2009,2010,2011,20...|
+--------------------+--------------+------------+--------------------+--------------------+--------------------+
```

Per visualizzare in maniera grafica tale andamento, si è scelto di portare a Pandas tale nuova colonna, per poi generarvi un grafico, tramite i comandi seguenti:

```
plot_bus_rev_user = buss_join_rev_join_user.select('RevStar_BussStar_diff').toPandas()

plot_bus_rev_user.plot.hist(bins=50, figsize=(8,6))
plt.xlabel('Differenza percentuale')
plt.ylabel('Conteggio review')
plt.title('Differenza percentuale tra stelle business e tra media di stelle utenti elite
    ')
plt.show()
plt.savefig('Diff_perc_busStars_EliteStars.png')
```

Si ottiene quindi il grafico in figura. Da tale andamento è già possibile andare a dedurre che la maggioranza delle recensioni degli utenti elite rispecchia fedelmente quella che è la valutazione del business, e che quindi è possibile fidarsi delle recensioni degli utenti elite.
A sostegno di tale ipotesi si è scelto di calcolare diverse percentuali per riguardo il numero di tali recensioni: in particolare, si vuole trovare la percentuale delle recensioni che si distacca dal valore del business di un certo range percentile. Tale analisi è fatta attraverso il seguente codice:




```
tot = buss_join_rev_join_user.count()

inf_25 = buss_join_rev_join_user.filter(buss_join_rev_join_user['RevStar_BussStar_diff']
     <= 25).count()
print(f'Differenza percentuale inferiore al 25%: {inf_25}, Percentuale: {(inf_25/tot)
    *100}%')

da_25_a_50 = buss_join_rev_join_user.filter(buss_join_rev_join_user['
    RevStar_BussStar_diff'] > 25).filter(buss_join_rev_join_user['RevStar_BussStar_diff'
    ] <= 50).count()
print(f'Differenza percentuale tra il 25% e il 50%: {da_25_a_50}, Percentuale: {(
    da_25_a_50/tot)*100}%')

da_50_a_75 = buss_join_rev_join_user.filter(buss_join_rev_join_user['
    RevStar_BussStar_diff'] > 50).filter(buss_join_rev_join_user['RevStar_BussStar_diff'
    ] <= 75).count()
print(f'Differenza percentuale tra il 50% e il 75%: {da_50_a_75}, Percentuale: {(
    da_50_a_75/tot)*100}%')

da_75_a_100 = buss_join_rev_join_user.filter(buss_join_rev_join_user['
    RevStar_BussStar_diff'] > 75).filter(buss_join_rev_join_user['RevStar_BussStar_diff'
    ] <= 100).count()
print(f'Differenza percentuale tra il 75% e il 100%: {da_75_a_100}, Percentuale: {(
    da_75_a_100/tot)*100}%')

sup_100 = buss_join_rev_join_user.filter(buss_join_rev_join_user['RevStar_BussStar_diff'
    ] > 100).count()
print(f'Differenza percentuale superiore al 100%: {sup_100}, Percentuale: {(sup_100/tot)
    *100}%')
```

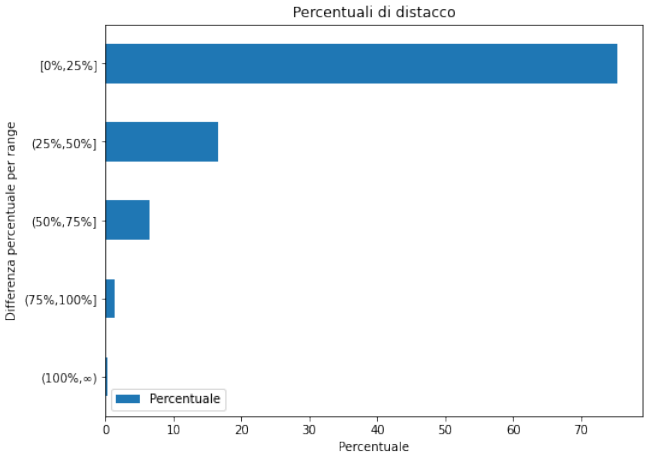
A valle di tale esecuzione, è stato prodotto il seguente output:

```
Differenza percentuale inferiore al 25\%: 1299274, Percentuale: 75.29151199136793\%
Differenza percentuale tra il 25\% e il 50\%: 285261, Percentuale: 16.53056399356072\%
Differenza percentuale tra il 50\% e il 75\%: 110822, Percentuale: 6.422014095492849\%
Differenza percentuale tra il 75\% e il 100\%: 24332, Percentuale: 1.4100128762477848\%
Differenza percentuale superiore al 100\%: 5969, Percentuale: 0.3458970433307179\%
```

Per comodità di visualizzazione, si è scelto di accorpare tali dati all'interno di un dataframe ed effettuare un grafico a barre.

```
df_percentuali = spark.createDataFrame(
    [
        ('[0%,25%]', (inf_25/tot)*100),
        ('(25%,50%]', (da_25_a_50/tot)*100),
        ('(50%,75%]', (da_50_a_75/tot)*100),
        ('(75%,100%]', (da_75_a_100/tot)*100),
        ('ϖ(100%,)', (sup_100/tot)*100),
    ],
    ['Range', 'Percentuale']
)
perc_pandas = df_percentuali.toPandas()
perc_pandas = perc_pandas.set_index('Range')
perc_pandas.plot.barh(figsize=(8,6)).invert_yaxis()
plt.title('Percentuali di distacco')
plt.xlabel('Percentuale')
plt.ylabel('Differenza percentuale per range')
plt.xticks()
plt.yticks()
plt.show()
plt.savefig('perc_distacco_tot.png')
```

Tale codice produce il seguente grafico.



2.4 Algoritmo predittivo per gli utenti

2.4.1 Versione Base

Si è deciso di addestrare un modello di Machine Learning, tramite la libreria **SparkML**, per prevedere la bontà di un utente rispetto ad alcuni parametri trovati all'interno della collection degli utenti.

Come prima approssimazione, quello di cui si necessita è delle etichette manualmente inserite per ogni utente, visto che non si dispone di tale informazione. Si è deciso quindi di creare manualmente una regola ad-hoc per l'identificazione di tali etichette, che in prima analisi comprendeva la verifica del numero di recensioni dell'utente, che gli altri utenti hanno trovato "useul", confrontato con il numero totale di recensioni che sono state fatte dall'utente.

Tale regola è stata prodotta attraverso il seguente codice:

```
user_class = (F.when(F.col('useful')/F.col('review_count') > 5, "good").otherwise("bad")
    )
UserLabeled = yelp_user.withColumn("goodness", user_class)
UserLabeled = UserLabeled.drop('yelping_since', 'user_id', 'name', 'friends', 'elite', '
    compliment_funny')

UserLabeled.show(5)
```

Viene così prodotto il seguente output:

```
+-------------+------------+------+------+--------+
|average_stars|review_count|useful|  name|goodness|
+-------------+------------+------+------+--------+
|         3.91|         585|  7217|Walker|    good|
|         3.74|        4333| 43091|Daniel|    good|
|         3.32|         665|  2086| Steph|     bad|
|         4.27|         224|   512|  Gwen|     bad|
|         3.54|          79|    29| Karen|     bad|
+-------------+------------+------+------+--------+
```

Si è quindi scelto di utilizzare tale classificazione manuale, per addestrare un modello di ML tramite framework Spark. In particolare, avendo costruito tale dataset assegnando una label binaria, si è deciso di utilizzare una **SVM (Support Vector Machine)**, modello di machine learning che rende le migliori performance in presenza di problemi di classificazione binaria. Si è quindi provveduto in primis a generare un vettore di features, tramite l'apposito comando Spark, e a trasformare la classe in un valore intero, sfruttando il codice seguente:

```
from pyspark.ml.feature import VectorAssembler, StringIndexer

columns = UserLabeled.columns
columns.remove('goodness')
```

```
ass = VectorAssembler(inputCols=columns, outputCol="features")
df = ass.transform(UserLabeled)

stringIndexer = StringIndexer(inputCol="goodness", outputCol="label")
si_model = stringIndexer.fit(df)
df = si_model.transform(df)
df.show(5)
```

Successivamente è necessario definire il modello, e avviarne il processo di training.

La fase di addestramento passa per la definizione di un dataset di addestramento e di uno per la verifica della bontà dell'addestramento, tramite i seguenti comandi:

```
from pyspark.ml.classification import LinearSVC
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

train = df.sampleBy("label", fractions={0: 0.7, 1: 0.7}, seed=10)
test = df.subtract(train)

lsvc = LinearSVC(maxIter=10, regParam=0.1)

lsvcModel = lsvc.fit(train)
```

Successivamente è necessario valutare la bontà di tale addestramento tramite un test set locale, calcolandovi una metrica specifica tra le etichette pre-impostate e quelle predette, che si è scelto essere l'accuracy. Per fare ciò sono stati dati in pasto al modello i dati di test, con i quali il modello ha estratto una predizione, sulla quale si è calcolata la metrica desiderata, tramite i seguenti comandi:

```
pred = lsvcModel.transform(test)

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="
    prediction", metricName="accuracy")
acc = evaluator.evaluate(pred)
print("Accuracy: %g" % (acc))
```

L'output ottenuto è il seguente:

```
Accuracy: 0.982835
```

2.4.2 Versione Aggiornata

Data l'accuratezza potrebbe sembrare che tale modello sia prossimo ad essere perfetto, ma la bontà dei risultati ottenuti è da confrontare con la semplicità della regola costruita per l'assegnazione dell'etichetta.

Infatti, avendo un etichetta dipendendente da un subset di features che è ridotto, è ovvio aspettarsi che un modello di machine learning possa identificare correttamente il pattern utilizzato per la costruizione dell'etichetta.

La bontà della classificazione sta quindi tutta nella semplicità della regola che l'algoritmo deve identificare per effettuare la sua predizione.

Si è quindi cercato di identificare la bontà degli utenti andando a considerare un numero di parametri superiore per l'assegnazione dell'etichetta, ed una regola differente da un semplice confronto.

Si tiene quindi ora conto del numero di complimenti ricevuti dall'utente, in relazione al numero di recensioni effettuate dallo stesso, andando a mediare la somma di tali complimenti per tale valore, tramite le seguenti righe di codice:

```
user_filtrati = yelp_user.select('compliment_cool', 'compliment_cute', 'compliment_hot',
     'compliment_note', 'compliment_writer', 'compliment_profile', 'review_count')

UserLabeledV2 = user_filtrati.withColumn('conto', (F.col('compliment_cool')+ F.col('
    compliment_cute')+F.col('compliment_hot')+F.col('compliment_note') + F.col('
    compliment_writer')+ F.col('compliment_profile'))/F.col('review_count'))
UserLabeledV2.show(10)
```

L'output visualizzato, di cui se ne riporta una versione con alcuni nomi delle colonne alterati per motivi di visualizzazione all'interno del documento, è quindi il seguente:

```
+--------+--------+-------+--------+----------------+-----------------+------------+--------------------+
|compcool|compcute|comphot|compnote|compliment_writer|compliment_profile|review_count|               conto|
+--------+--------+-------+--------+----------------+-----------------+------------+--------------------+
|     467|      56|    250|     232|             239|               55|         585|  2.2205128205128206|
|    3131|     157|   1145|    1847|            1521|              184|        4333|  1.8428340641587815|
|     119|      17|     89|      66|              35|               10|         665|  0.5052631578947369|
|      26|       6|     24|      12|              10|                1|         224| 0.35267857142857145|
|       0|       0|      1|       1|               0|                0|          79| 0.02531645569620253|
|    2543|     361|   1713|    1212|             815|              191|        1221|   5.597870597870598|
|       0|       0|      0|       0|               0|                0|          12|                 0.0|
|      12|       0|      4|       8|               5|                2|         358| 0.08659217877094973|
|       5|       3|      2|       3|               3|                0|          40|                 0.4|
|       3|       0|      0|       1|               0|                0|         109| 0.03669724770642202|
|       0|       0|      0|       0|               0|                0|           4|                 0.0|
+--------+--------+-------+--------+----------------+-----------------+------------+--------------------+
```

Per cercare di bilanciare il numero di etichette, si è andato a verificare quale fosse il numero medio di tale punteggi assegnati agli utenti, tramite i comandi:

```
df_check_value = UserLabeledV2.filter(F.col('conto') > 0.0).groupBy().avg('conto')
df_check_value.show(10)
```

Si è prodotto il seguente output:

```
+------------------+
|        avg(conto)|
+------------------+
|0.2213330822844841|
+------------------+
```

Si è scelto quindi di utilizzare come soglia un valore prossimo a quello medio per generare le etichette; si è quindi prodotto il seguente codice:

```
compliment_class = (F.when((F.col('compliment_cool')+ F.col('compliment_cute')+F.col('
    compliment_hot')+F.col('compliment_note') + F.col('compliment_writer')+ F.col('
    compliment_profile'))/F.col('review_count') > 0.3, "Trust").otherwise("Untrust"))

user_filtrati = user_filtrati.withColumn("complimentClass", compliment_class)
user_filtrati.show(10)
```

Si è ottenuto il seguente output:

```
+--------+--------+-------+--------+----------------+-----------------+------------+---------------+
|compcool|compcute|comphot|compnote|compliment_writer|compliment_profile|review_count|complimentClass|
+--------+--------+-------+--------+----------------+-----------------+------------+---------------+
|     467|      56|    250|     232|             239|               55|         585|          Trust|
|    3131|     157|   1145|    1847|            1521|              184|        4333|          Trust|
|     119|      17|     89|      66|              35|               10|         665|          Trust|
|      26|       6|     24|      12|              10|                1|         224|          Trust|
|       0|       0|      1|       1|               0|                0|          79|        Untrust|
|    2543|     361|   1713|    1212|             815|              191|        1221|          Trust|
|       0|       0|      0|       0|               0|                0|          12|        Untrust|
|      12|       0|      4|       8|               5|                2|         358|        Untrust|
|       5|       3|      2|       3|               3|                0|          40|          Trust|
|       3|       0|      0|       1|               0|                0|         109|        Untrust|
|       0|       0|      0|       0|               0|                0|           4|        Untrust|
+--------+--------+-------+--------+----------------+-----------------+------------+---------------+
```

Si è quindi provveduto ad effettuare le stesse operazioni fatte per la versione precedente, quindi identificando il vettore delle features, e creando il modello sulla base dell'etichetta convertita in intero, tramite i seguenti comandi:

```
columns2 = user_filtrati.columns
columns2.remove('complimentClass')
ass2 = VectorAssembler(inputCols=columns2, outputCol="features")
df2 = ass2.transform(user_filtrati)
```

```
5
6  stringIndexer2 = StringIndexer(inputCol="complimentClass", outputCol="label")
7  si_model2 = stringIndexer2.fit(df2)
8  df2 = si_model2.transform(df2)
9  df2.show(5)
10
11 train2 = df2.sampleBy("label", fractions={0: 0.7, 1: 0.7}, seed=10)
12 test2 = df2.subtract(train2)
13
14 lsvc2 = LinearSVC(maxIter=30, regParam=0.00001)
15
16 lsvcModel2 = lsvc2.fit(train2)
17
18
19 pred2 = lsvcModel2.transform(test2)
20
21 evaluator2 = MulticlassClassificationEvaluator(labelCol="label", predictionCol="
       prediction", metricName="accuracy")
22 acc2 = evaluator2.evaluate(pred2)
23 print("Accuracy: %g" % (acc2))
```

Con tale versione del modello, si è prodotto il seguente output:

```
Accuracy: 0.940241
```

Si conclude quindi che il modello ha delle performance comunque ottime, ma le prestationi sono leggermente decadute vista la regola di labeling più complessa.
Successivamente si è deciso di visualizzare alcuni utenti con le relative etichette di entrambi i modelli, tramite il seguente codice:

```
1  Trust_df = pred2.select('prediction').withColumnRenamed('prediction', 'Trustness')
2  Goodness_df = pred.select('prediction').withColumnRenamed('prediction', 'Goodness')
3
4  good_df_p = Goodness_df.toPandas()
5  trust_df_p = Trust_df.toPandas()
6  user_pandas = yelp_user.select('name','user_id', 'review_count').toPandas()
7
8  concat_df = pd.concat([user_pandas, good_df_p, trust_df_p], axis=1)
9
10 gdn_map = {0.0:'good', 1.0:'bad'}
11 tst_map = {0.0:'trust', 1.0:'untrust'}
12
13 concat_df = concat_df.replace({'Goodness':gdn_map, 'Trustness':tst_map})
14 concat_df = concat_df.dropna()
15 concat_df
```

Dall'esecuzione di questa porzione di codice si è infine prodotto il dataframe con le etichette predette, riportato di seguito.

	name	user_id	review_count	Goodness	Trustness
0	Walker	qVc8ODYU5SZjKXVBgXdI7w	585	good	trust
1	Daniel	j14WgRoU_-2ZE1aw1dXrJg	4333	good	trust
2	Steph	2WnXYQFK0hXEoTxPtV2zvg	665	good	trust
3	Gwen	SZDeASXq7o05mMNLshsdIA	224	good	trust
4	Karen	hA5IMy-EnncsH4JoR-hFGQ	79	good	trust
...	...	...	...	...	...
36458	Liz	h6_AMck7djAmqw5AGZ8rMg	64	good	untrust
36459	Sean	qbRwNuTqVKmVwLgZtTIXDg	107	good	untrust
36460	J	0Fhqp1nd4wjjMtmC367tAQ	3	good	untrust
36461	Rachel	KtUNBiZXQHfyTQLqYQIeXg	108	good	trust
36462	Gloria	hdyRQXc6hPFh5D79LwSLdA	152	good	untrust