Big Data Management

Assignement 1 - Yelp Reviews and Authenticity

Matteo Mormile | matmo@itu.dk | GitHub repository

10 October 2023

Contents

1 Introduction

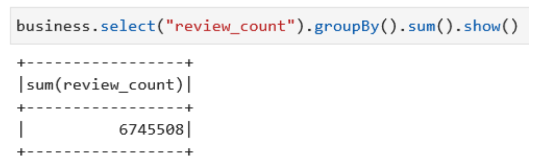
The report contains three different sections:

1. Creating queries to explore data.
2. Answering high-level questions and extracting insights from the data.
3. Performing feature engineering and machine learning for predictions.

2 Specific DataFrame Queries

Formulate the following queries using Spark DataFrames:

1. *Analyze business.json to find the total number of reviews for all businesses. The output should be in the form of a Spark Table/DataFrame with one value representing the count.*

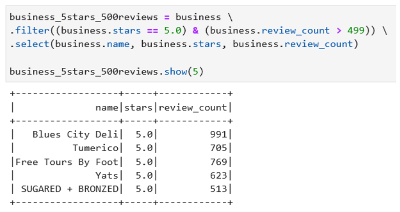
   

   ```
   business.select("review_count").groupBy().sum().show()
   ```

   ```
   +-----------------+
   |sum(review_count)|
   +-----------------+
   |          6745508|
   +-----------------+
   ```

   Starting from the business dataframe, we simply select the "review_count" column, we perform a groupby that then allows us to sum the values in the column.

2. *Analyze business.json to find all businesses that have received 5 stars and that have been reviewed by 500 or more users. The output should be in the form of DataFrame of (name, stars, review count).*

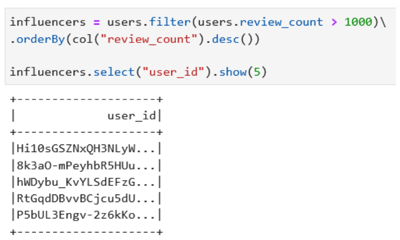
   

   ```
   business_5stars_500reviews = business \
   .filter((business.stars == 5.0) & (business.review_count > 499)) \
   .select(business.name, business.stars, business.review_count)

   business_5stars_500reviews.show(5)
   ```

   ```
   +------------------+-----+------------+
   |              name|stars|review_count|
   +------------------+-----+------------+
   |   Blues City Deli|  5.0|         991|
   |          Tumerico|  5.0|         705|
   |Free Tours By Foot|  5.0|         769|
   |              Yats|  5.0|         623|
   | SUGARED + BRONZED|  5.0|         513|
   +------------------+-----+------------+
   ```

   Starting from the business dataframe, we simply apply the filter function to select businesses that have received 5 stars and that have been reviewed by 500 or more. After we select the columns to keep in the new dataframe

3. *Analyze user.json to find the influencers who have written more than 1000 reviews. The output should be in the form of a Spark Table/DataFrame of user id.*

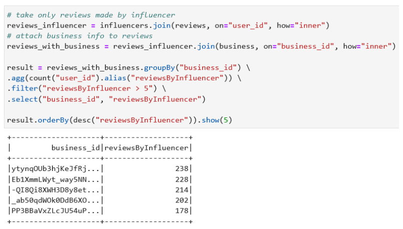
   

   ```
   influencers = users.filter(users.review_count > 1000)\
   .orderBy(col("review_count").desc())

   influencers.select("user_id").show(5)
   ```

   ```
   +--------------------+
   |             user_id|
   +--------------------+
   |Hi10sGSZNxQH3NLyW...|
   |8k3aO-mPeyhbR5HUu...|
   |hWDybu_KvYLSdEFzG...|
   |RtGqdDBvvBCjcu5dU...|
   |P5bUL3Engv-2z6kKo...|
   +--------------------+
   ```

   As the previous one we simply apply the filter on the user dataframe and the select to retrieve only the user_id column.

4. *Analyze review.json, business.json, and a view created from your answer to Q3 to find the businesses that have been reviewed by more than 5 influencer users.*

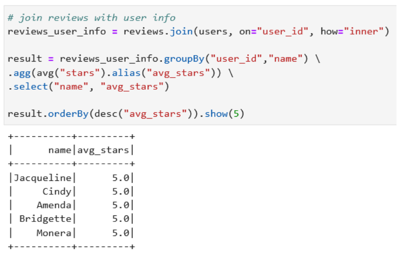


```
# take only reviews made by influencer
reviews_influencer = influencers.join(reviews, on="user_id", how="inner")
# attach business info to reviews
reviews_with_business = reviews_influencer.join(business, on="business_id", how="inner")

result = reviews_with_business.groupBy("business_id") \
.agg(count("user_id").alias("reviewsByInfluencer")) \
.filter("reviewsByInfluencer > 5") \
.select("business_id", "reviewsByInfluencer")

result.orderBy(desc("reviewsByInfluencer")).show(5)
```

```
+--------------------+-------------------+
|         business_id|reviewsByInfluencer|
+--------------------+-------------------+
|ytynqOUb3hjKeJfRj...|                238|
|Eb1XmmLWyt_way5NN...|                228|
|-QI8Qi8XWH3D8y8et...|                214|
|_ab50qdWOk0DdB6XO...|                202|
|PP3BBaVxZLcJU54uP...|                178|
+--------------------+-------------------+
```

Using the dataframe obtained in the previous step we perform the join with reviews so we get the reviews made by the influencers. By doing another join with the business df we also add to each reviews the info about the business they refer to. Then grouping by business_id and summing the user_id column we find the number of reviews made by influencers for each restaurant. By then filtering on the number of reviews we find the answer.

5. *Analyze review.json and user.json to find an ordered list of users based on the average star counts they have given in all their reviews.*



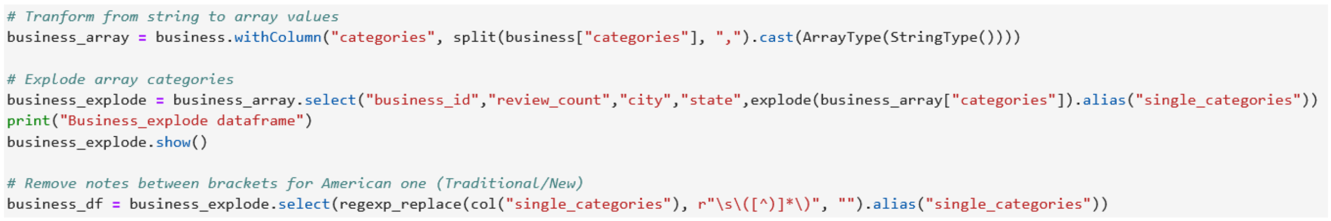
```
# join reviews with user info
reviews_user_info = reviews.join(users, on="user_id", how="inner")

result = reviews_user_info.groupBy("user_id","name") \
.agg(avg("stars").alias("avg_stars")) \
.select("name", "avg_stars")

result.orderBy(desc("avg_stars")).show(5)
```

```
+----------+---------+
|      name|avg_stars|
+----------+---------+
|Jacqueline|      5.0|
|     Cindy|      5.0|
|    Amenda|      5.0|
| Bridgette|      5.0|
|    Monera|      5.0|
+----------+---------+
```

First of all, we add to reviews dataframe the data about the users (we will display the name) who made the review. After we group by the user_id and calculate the new avg_star column by averaging the stars column present in reviews dataframe.

3 Authenticity Study

In order to properly work and analyze our data we will make and use some dataframes derived from the business one. In particular we will transform the categories column first into an array and then we will apply the explode command, which will create for each business new rows containing in the column "single_categories" the single values present before in the categories array. We will also use a regular expression to remove some superfluous notations within the category name, such as "new", "traditional". The code is the first block present in the Jupiter Notebook file of this section.

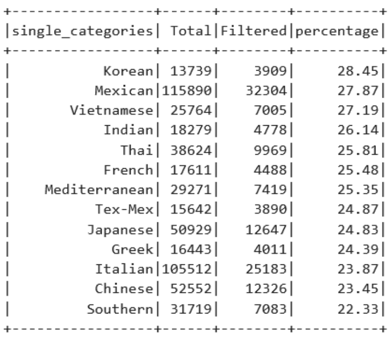


```
# Tranform from string to array values
business_array = business.withColumn("categories", split(business["categories"], ",").cast(ArrayType(StringType())))

# Explode array categories
business_explode = business_array.select("business_id","review_count","city","state",explode(business_array["categories"]).alias("single_categories"))
print("Business_explode dataframe")
business_explode.show()

# Remove notes between brackets for American one (Traditional/New)
business_df = business_explode.select(regexp_replace(col("single_categories"), r"\s\([^)]*\)", "").alias("single_categories"))
```

3.1 Data Exploration

In this section we want to analyze the Yelp database looking for some insight related to the use of the word *authenticity* and its synonyms, as done in the past and documented in this **Eater New York article**. To do this we will make use of some queries and analyze their result.

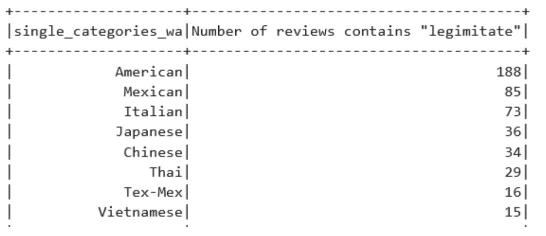
First we go to analyze how authentic language is used according to the different types of cuisines. By using the dataframe where businesses are already divided individually by category, we perform a join with the reviews that contains a word that invokes the term autencity (to do this we apply a filter on the reviews df). Having done this, we then go to perform a groupby on the individual categories and calculate the number of authentic reviews for each category. Finally we perform a filter to take into account only the categories that we are interested in and that are present in the list of cuisines.



```
+-----------------+------+--------+----------+
|single_categories| Total|Filtered|percentage|
+-----------------+------+--------+----------+
|           Korean| 13739|    3909|     28.45|
|          Mexican|115890|   32304|     27.87|
|       Vietnamese| 25764|    7005|     27.19|
|           Indian| 18279|    4778|     26.14|
|             Thai| 38624|    9969|     25.81|
|           French| 17611|    4488|     25.48|
|    Mediterranean| 29271|    7419|     25.35|
|          Tex-Mex| 15642|    3890|     24.87|
|         Japanese| 50929|   12647|     24.83|
|            Greek| 16443|    4011|     24.39|
|          Italian|105512|   25183|     23.87|
|          Chinese| 52552|   12326|     23.45|
|         Southern| 31719|    7083|     22.33|
+-----------------+------+--------+----------+
```

We can see that if we analyze cuisines according to the percentage between total reviews and those containing a synonym for authentic (we filter on those with at least 10000 total reviews) we observe among the top places, the main East Asian cuisines (Korean, Vietnamese, Chinese and Thai) synonymous that in these cuisines, the authenticity language is more used than in others where it is not given importance.

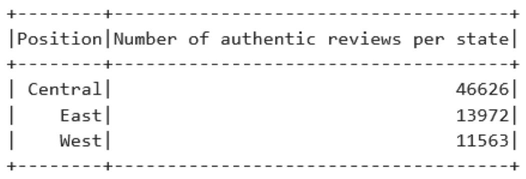
Another type of analysis we can do is to directly look at the number of reviews for each cuisine that contain the *legitimate* string. To do this we use the business_cuisine_wa dataframe which contains the list of all the cuisines present in the dataframe with the removal of any notes in brackets. Starting from the review dataframe we filter those reviews that contain the word legitimate, we then join with the business_cuisine_wa dataframe to then be able to group according the different cuisines. During clustering we count the number of reviews found. Also in this case we observe how the East Asian countries are among the top positions. We also find countries like Italy and Mexico where cuisine is a distinctive and characteristic element of that nation.



```
+--------------------+--------------------------------------+
|single_categories_wa|Number of reviews contains "legimitate"|
+--------------------+--------------------------------------+
|            American|                                   188|
|             Mexican|                                    85|
|             Italian|                                    73|
|            Japanese|                                    36|
|             Chinese|                                    34|
|                Thai|                                    29|
|             Tex-Mex|                                    16|
|          Vietnamese|                                    15|
```

Having a large number of reviews from America, I decided to analyse how these are distributed among the American states, particularly those in the East, Centre and North.

Starting from the dataframe that contains the list of reviews with a synonym of the word authentic inside the text, I then joined with the business_america dataframe which contains the list of all the businesses (id and position) that have American cuisine as their categories. Helping me with a specifically created dataframe, which contains the relative geographical position for each nation, I finally grouped by the position and obtained the following results.



```
+--------+-----------------------------------+
|Position|Number of authentic reviews per state|
+--------+-----------------------------------+
| Central|                              46626|
|    East|                              13972|
|    West|                              11563|
+--------+-----------------------------------+
```

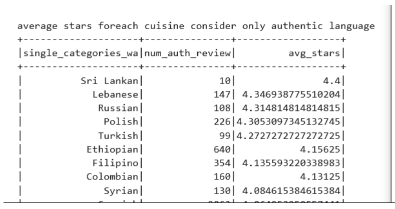
As we can observe from the result, it seems that authentic language is more frequently used in the central nations of America rather than on the borders. This could be because, as the border cities are more visited in America, they might have lost some of their American authenticity. In contrast, in states like Texas, Colorado, and Kansas, it is known that there is a more rural population with less globalized cuisine.

3.2 Hypothesis Testing

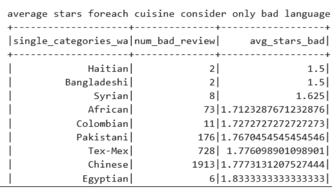
We want to try to test the hypothesis proposed by the article which says that the authenticity language is used to describe different characteristics for different cuisines (and by extension, makes it harder for non-white restaurant owners to enter the higher-end restaurant market).

I have decided to conduct the research in two different ways. In the first method, I will analyze individual reviews containing synonyms for *authentic* and reviews containing "bad" words. Then, I will calculate the average of these reviews, group them by cuisine, and perform a weighted average (based on the number of reviews for each cuisine), grouping everything by continents. In the second case, I will analyze the average of reviews containing both synonyms for "authentic" and negative words, still grouping by cuisine and continent.

For the first part, since the evaluation between reviews containing synonyms for "authentic" and those containing bad words is the same, I will focus only on the first one in the description below.
Starting with the dataframe containing reviews with synonyms for "authentic," I performed a join with the "business_cuisine_wa" df, which contains a list of all business IDs that fall into the categories of the most important cuisines worldwide. Afterward, I grouped the data by cuisine and calculated the average stars of the reviews.



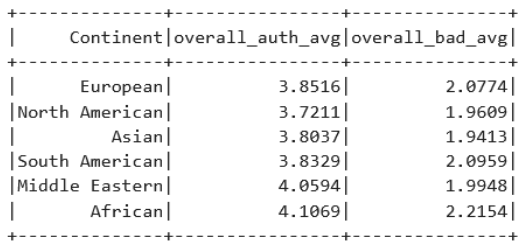
```
average stars foreach cuisine consider only authentic language
+--------------------+---------------+------------------+
|single_categories_wa|num_auth_review|         avg_stars|
+--------------------+---------------+------------------+
|          Sri Lankan|             10|               4.4|
|            Lebanese|            147| 4.346938775510204|
|             Russian|            108| 4.314814814814815|
|              Polish|            226|4.3053097345132745|
|             Turkish|             99|4.2727272727272725|
|           Ethiopian|            640|           4.15625|
|            Filipino|            354| 4.135593220338983|
|           Colombian|            160|           4.13125|
|              Syrian|            130| 4.084615384615384|
```



```
average stars foreach cuisine consider only bad language
+--------------------+--------------+------------------+
|single_categories_wa|num_bad_review|     avg_stars_bad|
+--------------------+--------------+------------------+
|             Haitian|             2|               1.5|
|          Bangladeshi|             2|               1.5|
|              Syrian|             8|             1.625|
|             African|            73|1.7123287671232876|
|           Colombian|            11|1.7272727272727273|
|           Pakistani|           176|1.7670454545454546|
|             Tex-Mex|           728| 1.776098901098901|
|             Chinese|          1913|1.7773131207527444|
|            Egyptian|             6|1.8333333333333333|
```

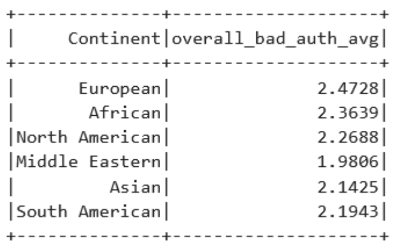
Using a dataframe that maps cuisines to continents, I then performed another join, followed by a group by operation to obtain the values for the weighted average, which will be calculated after by performing the division and saving it in a new column.

By following the same process for reviews containing negative words, we obtain a column as follows, representing the average ratings for reviews containing authentic and negative words, divided by continents.



```
+--------------+----------------+---------------+
|     Continent|overall_auth_avg|overall_bad_avg|
+--------------+----------------+---------------+
|      European|          3.8516|         2.0774|
|North American|          3.7211|         1.9609|
|         Asian|          3.8037|         1.9413|
|South American|          3.8329|         2.0959|
|Middle Eastern|          4.0594|         1.9948|
|       African|          4.1069|         2.2154|
+--------------+----------------+---------------+
```

For the second part, I simply performed a join between the reviews containing synonyms for "authentic" and those containing negative words (a double filter on the content would have been more efficient), and then I grouped them first by cuisine and then by continent.



```
+--------------+--------------------+
|     Continent|overall_bad_auth_avg|
+--------------+--------------------+
|      European|              2.4728|
|       African|              2.3639|
|North American|              2.2688|
|Middle Eastern|              1.9806|
|         Asian|              2.1425|
|South American|              2.1943|
+--------------+--------------------+
```

From the first table, we can observe that:

- The reviews containing negative words lead to a lower average rating in Asian countries compared to the rest of the world.
- As highlighted in the article, reviews containing synonyms for the word 'authentic' correspond to higher average ratings in European and South American regions compared to the Asian continent.

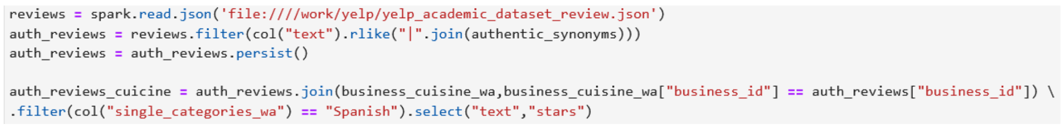
On the other hand, the second table confirms that if we search for reviews containing both authentic language and negative words, these reviews will have a greater negative impact on Asian cuisines.

4 Rating Prediction

We now want to build an ML model that predicts the rating of a certain restaurant given a user review.
To do this, we will use two different predictors: the Decision Tree algorithm and the Naive Bayes algorithm. We will compare the accuracy achieved with these two algorithms and attempt to provide meaningful interpretations of the results. The main steps we will perform on our dataset will be:

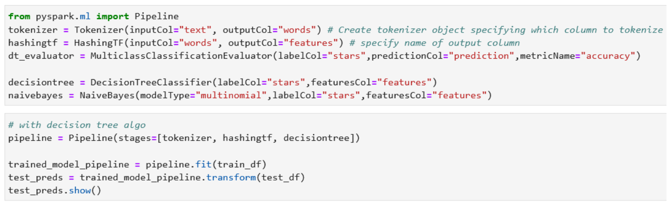
- splitting the dataset into training and test sets
- extracting features from the reviews
- training your model on the training set
- and evaluating your model on the test set

Before proceeding with the division between train and data set, given the size of the review dataset, I have decided to reduce the prediction to the reviews that contain the word "authentic" (as referenced in the project above) and belong to the Spanish cuisine category. To do this, starting from the review DataFrame, I will filter based on the text content and category type, selecting only the columns that are truly needed for creating my model.



```
reviews = spark.read.json('file:////work/yelp/yelp_academic_dataset_review.json')
auth_reviews = reviews.filter(col("text").rlike("|".join(authentic_synonyms)))
auth_reviews = auth_reviews.persist()

auth_reviews_cuicine = auth_reviews.join(business_cuisine_wa,business_cuisine_wa["business_id"] == auth_reviews["business_id"]) \
.filter(col("single_categories_wa") == "Spanish").select("text","stars")
```

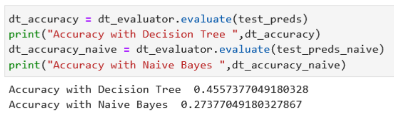
After that, I can split the dataset into training and test sets. However, before training my model with the training data, I need to transform the data to obtain suitable values for the predictor model. Specifically, using the Tokenizer object, I will convert the text string of the review into an array of words. Then, through the hashingTF object, I will convert each word into a distinct number that identifies it. This process is the same for both types of predictors.
To avoid performing these transformations sequentially, we can use the Pipeline object, which will automatically apply them to the DataFrame before instructing the model.



```
from pyspark.ml import Pipeline
tokenizer = Tokenizer(inputCol="text", outputCol="words") # Create tokenizer object specifying which column to tokenize
hashingtf = HashingTF(inputCol="words", outputCol="features") # specify name of output column
dt_evaluator = MulticlassClassificationEvaluator(labelCol="stars",predictionCol="prediction",metricName="accuracy")

decisiontree = DecisionTreeClassifier(labelCol="stars",featuresCol="features")
naivebayes = NaiveBayes(modelType="multinomial",labelCol="stars",featuresCol="features")
```

```
# with decision tree algo
pipeline = Pipeline(stages=[tokenizer, hashingtf, decisiontree])

trained_model_pipeline = pipeline.fit(train_df)
test_preds = trained_model_pipeline.transform(test_df)
test_preds.show()
```

I repeat the same operations, changing the type of algorithm used within the pipeline object and incorporating Naive Bayes. Afterward, by using a Multiclass Classification Evaluator object (multiple because the prediction type is not binary, but is the number of stars given in the review, from 1 to 5), I can calculate the accuracy of the two algorithms.



```
dt_accuracy = dt_evaluator.evaluate(test_preds)
print("Accuracy with Decision Tree ",dt_accuracy)
dt_accuracy_naive = dt_evaluator.evaluate(test_preds_naive)
print("Accuracy with Naive Bayes ",dt_accuracy_naive)
```

```
Accuracy with Decision Tree  0.4557377049180328
Accuracy with Naive Bayes  0.27377049180327867
```

As we can see, the accuracy of the Decision Tree algorithm is significantly higher. The higher accuracy of the Decision Tree algorithm in this specific case could be due to several factors, including its ability to model complex relationships in the data and adaptability to the training data.