IT UNIVERSITY OF COPENHAGEN

LSDA: Assignment 2

Author
Maciej Jalocha

Email address
macja@itu.dk

Code
https://github.itu.dk/Large-Scale-Data-Analysis-2025/a2

1 Specific DataFrame Queries - to get overview of data

1.1 Total number of reviews for all businesses.

```
business.count()
```

1.2 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 a DataFrame of (name,stars, review count)

```
business
.filter(
    (business.stars >= 5)
    &
    (business.review_count >= 500)
    )
.select(
    "name",
    "stars",
    "review_count"
)
```

1.3 Influencers who have written more than 1000 reviews. The output should be in the form of a SparkTable / DataFrame of user id.

```
influencers = users
.filter(users.review_count >= 1000)
.select("user_id")
```

1.4 The businesses names that have been reviewed by more than 5 influencer users.

```
business
.join(reviews,
    on="business_id", how="inner")
.join(influencers,
    on="user_id", how="inner")
.select("name", "user_id")
.groupBy('name')
.count()
.filter(col('count') >= 5)
.select('name')
```

1.5 An ordered list of users based on the average star counts they have given in all their reviews.

```
reviews
.join(users, on="user_id", how="right")
.select("stars", "user_id")
.groupBy('user_id')
.mean()
.sort(col('avg(stars)'),
    ascending=False)
```

2 Authenticity Study

In this study I am going to check if Alternative Hypothesis: ”Authentic dining experiences in South America and Asian restaurants are associated with low standards more often than in European restaurants” is more favourable than Null Hypothesis (i.e. no difference.).

To answer that question, I start with data exploration in order to better understand the data, but I treat it more as a standalone exploration rather than an introduction to hypothesis testing, so it is expected it will be detached from it. I talked with Zoi about it - I follow the assignment.

Then I will collect all reviews containing the ’authenticity’ language, then extract a variable denoting the cuisine to which the review is referring and a binary variable denoting the use of the low-standards language.

Code is at the bottom and is always related to the table. All tables were directly created in the PySpark and copied pasted from there.

2.1 Data Exploration

2.1.1 Data

In Table 1 I present tables from Yelp Dataset:

Column	Type
User	
user_id	int (PK)
average_stars	float
review_count	int
Review	
review_id	int (PK)
text	text
user_id	int (FK to users.user_id)
business_id	int (FK to business_id)
Business	
business_id	int (PK)
categories	text
city	text
review_count	int
state	text
stars	float
name	text

Table 1: Combined table schema for User, Review, and Business tables

business.stars denote average stars a business has received, and business.categories are comma-separated keywords related to a business. Let me note, that not all business table entries are restaurants (although the majority are). Review, business and user tables have sizes 6,990,280 and 150,346 and 1,987,897 respectively.

The knowledge of the fields is fundamental to formulate subsequent queries.

2.1.2 Authenticicy language prevalence

In order to detect 'authenticity language', I look if a column reviews.text contains one of the **substrings**: 'authentic' or 'legitimate'. **Limitations**: These are just two keywords, there might be other words suggesting use of 'authenticity' language. However, this study is actually not about 'language' but describing authentic experiences. A review like: 'We felt as if we were in China' speaks about authentic experiences but will likely not be captured by any keyword method. Lastly, there might be review not speaking about 'authentic' experiences. I didn't sample, though I suspect this effect shall be small.

substring	Count	Percentage
authentic	124634	0.018
legitimate	5066	0.0007
either	129503	0.019

Table 2: Authenticity language prevalence

Further exploration will be conducted by looking at reviews that contain at least one of these substrings in lowercase because distinction is not important for the hypothesis.

2.1.3 Authenticicy language across cuisines

Hypothesis testing requires cuisine grouping. Now we will look into use of authenticity language across cuisines.

Method:

In Table 3 we look at prevalence of 'authenticity' language across cuisines. The cuisines I chose were taken from Eater New York article i.e. most popular cuisines in New York City. cuisines were obtained by checking if the name of a cuisine appears in business.categories column. I briefly looked into other world cuisines - including noticeably into Spanish and Greek cuisines, which I thought were popular, but found their counts to be tiny compared to the cuisines from the article. I ended up not using them at all. **Limitations**: I'm aware that this is noisy though not to how big extent. I am aware that perhaps additional check for 'restaurant' keyword would help, but then again I am aware this could be introduce even further noise (is a food stand a restaurant?). From my perspective it is not very important for hypothesis testing, because - here is my assumption - hypothesis testing is going to be carried out on reviews using 'authenticity' language which are about a business with cuisine keyword in business.categories field and these two variables together shall be a good indicator it is a restaurant. I didn't check this thoroughly due to time constraints though.

Cuisine	Mentions	Total	Ratio
Mexican	31	432	0.07
Thai	6	93	0.06
Indian	5	76	0.07
Chinese	12	224	0.05
Japanese	4	177	0.03
Italian	11	432	0.03
Korean	2	31	0.07
French	1	83	0.02
Soul	1	64	0.02
Mediterranean	5	108	0.04

Table 3: Cuisine popularity with counts rounded to the nearest thousand. Total is total reviews, Mentions is count of reviews using 'authencitiy language'

Let me note, that we can spot that the historically poorer countries i.e. Mexico, Thai, Indian, Chinese and Korean appear to have higher use of 'authenticity' language rather than wealthier countries (Japan, Italy, France). Interestingly, Soul and Mediterranean cuisines does not follow that; perhaps it is related to the fact these cuisines have just become trendy, as the article suggests, and, secondly, are not tied to any country.

Upon reading Scientific American (which was mentioned in the main article) I came to realise that the potential difference could be be not between about Western/European and Non-Western/Asian/South American cuisines but the affluence of a nation. An example is Japan being affluent yet Asian. Apart from original hypothesis, I will check for that.

2.1.4 Influence of region on Authenticicy language

Now let us see counts for 'authenticity language' reviews across states and most reviewed cities in the dataset:

State	SAR	City	CAR
AB	0.06	Edmonton	0.06
AZ	0.04	Tucson	0.04
CA	0.05	Santa Barbara	0.05
DE	0.06	Wilmington	0.05
FL	0.05	Tampa	0.06
ID	0.05	Boise	0.05
IL	0.04	Edwardsville	0.03
IN	0.05	Indianapolis	0.05
LA	0.03	New Orleans	0.03
MO	0.04	Saint Louis	0.04
NJ	0.05	Cherry Hill	0.05
NV	0.05	Reno	0.04
PA	0.05	Philadelphia	0.04
TN	0.05	Nashville	0.04

Table 4: State and city authorization ratios. SAR - State Authenticity Ratio, same for a city.

The Table 4 presents authenticity ratios per state and a city with the most reviews in a state (ratios between counts of all reviews and 'authenticity language' using ones). It is interesting that we are seeing some very tiny cities and big cities together - but most likely it is because dataset is trimmed very much. After all we have only access to the minority of states in US and Canada and perhaps cities as well. Another interesting thing to see is that the CAR follows SAR - but it comes from the fact these cities contribute the most to the state counts. **Conclusion** The numbers suggests there is no difference between states nor cities i.e. location has no effect. (Author note: the hardest query, took 4 hours :) but I used rollup!)

2.2 Hypothesis testing

During data exploration I obtained two required variables for each of the reviews i.e. cuisine and use of 'authencitiy language'. I still need to define 'low-standards language' though. Let me lay out the setup and the plan:

2.2.1 Setup

- I'm merging the cuisines into European and **S**outh **A**merican or **As**ian cuisines (SAAS). I.e. Italian, French and Mediterranean are European, rest is not. Two populations.
- I'm filtering reviews only to the ones that contain authenticity language, and then I add additional variable denoting whether 'low-standards' language is used or not. One variable - 'authentic+lowstandards' vs 'authentic+no-lowstandards'. I.e. all tested reviews contain the 'authenticity language' and very whether these contain 'low-standards' language.

This setup therefore contains two populations and one single dichotomous variable. First I perform chi-square test as a test for independence (could be diff. of proportions with pooled SE as well, but I use the most standard method), then I do the directional test using chi-square to assess which probability could be higher (Euroeapn or SAAS) and then compute diff. of proportions and corresponding double sided 95% CI (unpooled SE) to assess the specific difference. I refer you to Statistical Data Analysis for the Life Science, 2nd Edition, specifically pages 335, 338 and 340, 'middle' paragraph in each. **Question:** In the book I saw that they computed CI for difference of proportions and stated it could act as a significance test, but they computed unpooled SE instead of pooled while doing so (page 323). Why is that? This is coursebook for stats.

Method of obtaining 'low-standards (LSL) language'

Again, I am keyword matching. I check for the containment of: "dirty", "kitsch", "cheap", "rude", "simple" in reviews.text. The same limitations as for obtaining "use of 'authenticity language' variable" apply therefore I will not repeat myself here.

Cont:

Cuisine	Authentic+LSL	Authentic+!LSL
European	1187	15705
SAAS	4976	56472

Table 5: 2x2 Contigency table. Counts for two categories: rows ("Cuisine" vs. "Top") and columns (Authentic and Dirty vs. Authentic not Dirty).

The test for independence gives p-value of 5e-06 and Chi-square 20.82. For European restaurant we get probability of 0.070 and for SAAS restaurant 0.081, so SAAS probability is higher. So, the directional test to check that the probability for SAAS is higher is simply dividing the previous p-value by half (book, page 342) i.e 2.5e-06 which is still significant.

Given we rejected H_0, now I compute CI, with unpooled SE, to obtain the difference of proportions. 95% CI is (0.0063, 0.0151). It is close to zero, but p-value is extremely tiny.

The above test support the hypothesis that authenticity language in reviews of SAAS cuisines is used to describe low standards.

Japan: is it affected by authenticity trap?

I also checked if the result remained if Japan was in the group of 'European' countries. I read that Japan despite being Asian might have escaped the 'authenticity trap'. It seems to be the case, because p-value still remains significant. However, it dropped drastically. P-value of the directional test that poor (SAAS without Japan) cuisines has higher probability than affluent cuisines (Europe with Japan) dropped to 9.7e-04 i.e. 50-fold. In my view this supports the argument that Japanese is not affected by the 'authenticity' trap, but this is not a direct nor rigours check. A country-cuisine level should be incorporated.

2.3 Model training

I used SVM due to its known superior performance on high dimensional data. Unfortunately, MLLib constrained me to use SVM only with a) Linear Kernel (though I wanted to test different ones) b) only as a classifier (though I think regressor could do better) - though as my metrics I use RMSE which is more appropriate I think. I perform stratified split to train and test (no validation) in proportion 0.7:0.3 (with random seed), then train SVM with maximum iterations of 10 and regularisation parameter 0.1 (punishment for crossing boundaries of soft SVM). See the code 3 (the last, not 3) During training I had also to wrangle with two issues: a) a known bug in 3.3.1 with SVM such that indexing of labels must start from zero. Therefore I perform mapping for stars x -¿ x-1. b) some bug that worker didn't have installed numpy. I found out that it wass possible to install additional dependencies to workers by passing conda .yml with numpy in ucloud.

Limitation: I based my hyper paramaters on the default hyper parameters from the example in the documentation. For example, by default maxIter is 100, but I set it to 10 to redue training time to keep it max 10 minutes. Had I had more time it would be obvious for me to increase this parameter to at least default value.

For the features I used a) two binary features being true when the review's text contained authenticity language and low-standards language respecitvely (flags) b) HashingTF with 2**14 feats - taken from exercises (text features) c) state, one-hot encoded. (state).

Results: The conclusion is strong that text

Features	RMSE
text, flags	0.973
text, state	0.975
flags, state text	0.973
flags	1.8
all	0.975

Table 6: LinearSVM results

features are sufficiently extracting relevant information, and that location is not helpful.

2.4 Other Limitations

- picking only the most popular NY is noisy. This could have been picked out from the data just by counting reviews one cuisine variable is obtained. But the best case would be to find all possible cuisines. I made a mistake by dropping Greek and Spanish cuisines.

3 Code

If I were to put important code in the report it would have been unreadable.

Table 2

```
c = lower(col('text'))
n_authentic = reviews\
.filter(c.contains('legitimate'))\
.count() #change with other conditions
total = reviews.count()
percentage_authentic \
= n_authentic/total
```

Code for 3

```
condition =\
c.contains('legitimate') \
| c.contains('authentic')
reviews_cuisine_text = business\
.filter(
    col('cuisine')!='non-restaurant')\
.join(reviews, on="business_id",
    how="inner")\
.select('cuisine', 'text', 'review_id')\
.cache()

c=col('text')
review_mentions_authentic=\
reviews_cuisine_text\
.withColumn('mentions_authentic',
    condition)\
.cache()

review_mentions_authentic\
.filter(
    col('mentions_authentic') == True)\
.groupBy('cuisine')\
.count()\
.withColumnRenamed(
    'count', 'mentions_count')\
.join(
    reviews_cuisine_text\
    .groupBy('cuisine')\
    .count()\
    .withColumnRenamed(
    'count', 'total_count'),
    on='cuisine')\
.withColumn('ratio', round(
    col('mentions_count')/
    col('total_count'), 2)).show()
```

Table 4

```
condition =\
col('text').contains('legitimate') |\
col('text').contains('authentic')
restaurant_reviews_on_business =\
business.filter(
    col('cuisine')!='non-restaurant')\
.join(reviews,
    on="business_id", how="inner")\
.select('cuisine', 'text', 'state', 'city')\
.cache()
window = Window.partitionBy("state")\
.orderBy(col("count").desc())

totals = restaurant_reviews_on_business\
.rollup('state', 'city')\
.count()\
.filter(col('state').isNotNull())\
.withColumn('rank',
    row_number().over(window))\
.filter(col('rank') <= 2)\
.select('state', 'city', 'count')

city_df = totals.filter(col('city')\
    .isNotNull())\
.select('state', 'city',
    col('count').alias('city_count'))
state_df = totals.filter(col('city')\
.isNull()).select('state',
    col('count').alias('state_count'))
```

```
totals = city_df\
    .join(state_df, on='state')

auth_totals = restaurant_reviews_on_business\
.filter(condition)\
.rollup('state', 'city')\
.count()\
.filter(col('state').isNotNull())\
.select('state', 'city', 'count')

auth_city_df = auth_totals.filter(col('city')\
.isNotNull()).select('state', 'city',
    col('count').alias('auth_city_count'))
auth_state_df = auth_totals.filter(col('city')\
.isNull()).select('state',
    col('count').alias('auth_state_count'))
auth_totals =\
auth_city_df.join(auth_state_df, on='state')

combined = totals.join(auth_totals,
    on=['state', 'city'], how='left')
combined = combined\
    .withColumn('city_auth_ratio', round(
    col('auth_city_count')/col('city_count'),2))\
    .withColumn('state_auth_ratio', round(
    col('auth_state_count')/col('state_count'),2))\
    .select('state', 'state_auth_ratio',
        'city', 'city_auth_ratio')
```

Code for training the model

```
c=col('text')
data = business\
.filter(col('cuisine')!='non-restaurant')\
.select('business_id', 'state')\
.join(reviews, on="business_id", how="inner")\
.withColumn('mentions_dirty',
    low_standards_language_condition.cast('integer'))\
.withColumn('mentions_authentic',
    authenticity_language_condition.cast('integer'))\
.select('text', 'stars', 'state',
    'mentions_dirty', 'mentions_authentic', 'review_id
.cache()

def stratified_train_test_split(df, frac, label,
    join_on, seed=42):
    """ stratfied split of a dataframe in train and te
    fractions =\
        df.select(label)\
        .distinct()\
        .withColumn("fraction", f.lit(frac))\
        .rdd\
        .collectAsMap()
    df_frac =\
        df.stat.sampleBy(label, fractions, seed)
    df_remaining = \
        df.join(df_frac, on=join_on, how="left_anti")
    return df_frac, df_remaining
```

```
reviews_sam = data#.sample(withReplacement=False, fraction=1/2)
reviews_sam = reviews_sam\
    .withColumn('stars',
        zero_center_stars(reviews_sam['stars']))
# reviews_sam.count()

raw_train, raw_test =\
    stratified_train_test_split(reviews_sam, 0.7, label='stars', join_on='review_id')
# text
tokenizer =\
    Tokenizer(inputCol="text", outputCol="words")
hashingtf =\
    HashingTF(numFeatures=2**14,inputCol=tokenizer.getOutputCol(), outputCol="textFeatures")

feats = {
    'featuresCol': 'allFeatures',
    'labelCol': 'stars'
}
lsvc = LinearSVC(maxIter=10, regParam=0.1, **feats)
ovr = OneVsRest(classifier=lsvc, **feats)
revaluator = RegressionEvaluator(
    predictionCol='prediction', labelCol='stars')
state_indexer = StringIndexer(
    inputCol="state", outputCol="stateIndex", handleInvalid="keep")
state_encoder = OneHotEncoder(
    inputCols=["stateIndex"], outputCols=["stateVec"])

assembler = VectorAssembler(
    inputCols=["textFeatures", "stateVec", "mentions_dirty", "mentions_authentic"],
    outputCol="allFeatures")


pipeline = Pipeline(
    stages=[tokenizer, hashingtf, state_indexer, state_encoder, assembler,ovr])

trained_model_pipeline = pipeline.fit(raw_train)
test_preds = trained_model_pipeline.transform(raw_test)
revaluator.evaluate(test_preds, {revaluator.metricName: "rmse"})
```