École Polytechnique Fédérale de Lausanne

# Yelp Restaurants reviews

Bollero Francesco, Mariantoni Mattia, Mulet-Arabí Pau, Padovano Federica, Rossi Luca

November 21, 2021

# Contents

# 1 Introduction

For this project we decided to study a data-set showing data of reviews owned by Yelp. Yelp is an American company founded in 2004 by Russel Simmons and Jeremy Stoppelman. They created a website (Yelp.com) and an application (Yelp App) which publish crowd-sourcing reviews of businesses. This company has grown a lot in the past years, in December 2020 they counted 206.3 million reviews available on its business listing pages.

Figure 1: Yelp logo.

In particular we focused on restaurants reviews, so we chose three different databases that are taking in account users profiles of people using the Yelp App, business profiles of stores that have been reviewed and all the reviews. In the following image it is shown which are the most important attributes for each database, focusing on the fact that the Reviews data-set is linked to the other two thanks to the "User id" and "Business id" attributes.

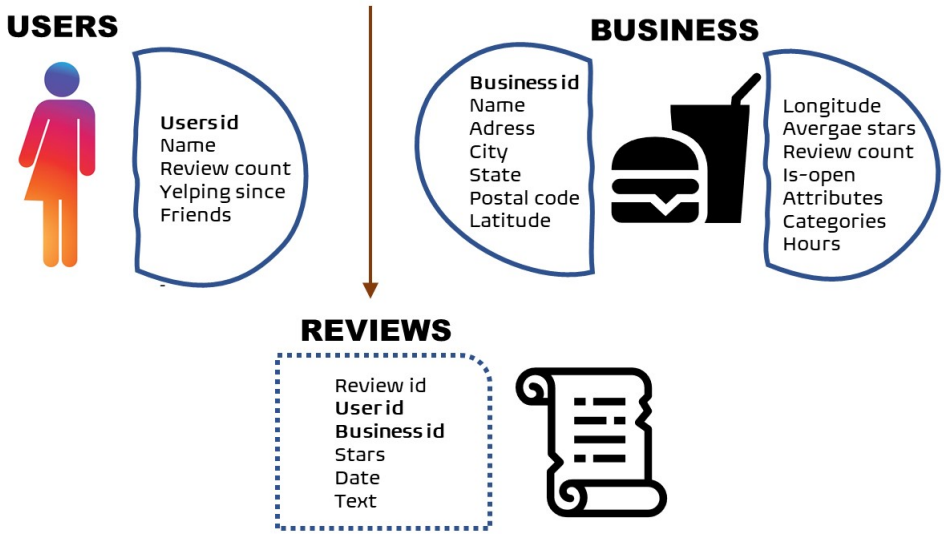



Figure 2: Data-set and their features.

In the first part of our work, we consider interesting studying the evolution in time and space of the app usage to have a better understanding of the dataset and to be able to draw meaningful conclusions for the following tasks. Secondly, we will focus on the ratings, indeed the ratings are the main indicators of the success of a restaurant. We will study how the ratings are related to the geographical position of the restaurants and how they are influenced by the period of the year in which they have been collected. We will

then try to understand which features are the most important for a better rate. Then, we will try to predict whether a restaurant is closed or not just looking at the information of the restaurant and at the reviews. In addition, we will create a recommender system based on common likes of the same restaurant. Finally we will cluster reviews by topics and with sentimental analysis we will try to predict the rating of each review just based on the text of the review and on her topic.

In particular, we decided to tackle the following questions:

1. How did the App usage evolve in time and space?
2. How are the number of reviews distributed among the cities? And the scores?
3. How are the ratings related to different periods of the year or different years ?
4. Is there a sort of discrimination against Mexican restaurants ?
5. Which features of a restaurant are important for a good review?
6. Can we predict whether a restaurant is closed or not based on the reviews and on the information of a restaurant?
7. People that go and love the same restaurants are friends? If not, can we suggest them new friends based on the fact that they share the same restaurants?
8. Can we cluster the reviews in different topics ?
9. Can we predict the reviews ratings analysing the review texts?

# 2 Description of the data-set

Yelp data set was compose by multiple 'json' files, to answer our questions we only used few of them:

- 'yelp academic dataset business.json'
- 'yelp academic dataset user.json'
- 'yelp academic dataset review.json'

These data set were then connected, we will explore the common variables in the sections below.

## 2.1 Review data-set

Because this data-set was extremely big (more than 8 million rows), Jupiter Notebook was not able to open it, consequently we had to find an alternative way to work with it. To do so, we made use of Pandas library which is extremely useful because it allowed us to open it five hundred thousands rows at the time (with the chunksize option in read json function) as pandas dataframe (DataFrame is a 2-dimensional labeled data structure with columns of potentially different types) and to convert them to 'cvs' files of five hundred thousands rows each, exception made for the last one. Each of these data set has the following variables:

- review id: index to identify each review.
- user id: index to identify the author of a review, it was also the connecting data to the user data-set.
- business id: index to identify the shop that was reviewed, it was also the connecting data to the business data-set.
- stars: the rate of the review, it could go from 1 (bad review) to 5 (good review).
- useful: this variable indicates the number of users who found the review useful.
- funny: this variable indicates the number of users who found the review funny.
- cool: this variable indicates the number of users who found the review cool.
- text: here we find the comment written by the user, which describes his experience in a certain shop.
- date: variable that indicates when the review was made. An example of a date is the following: '2014-10-11 03:34:02'. As can be seen we have also information about the hour of the review.

Once we had these smaller data-set we manage to open them one by one in Jupiter and drop the irrelevant columns, doing so we managed to handle the entire review data-set.

### 2.2 Business data-set

This data-set contains all the information about the business that are registered in Yelp app. It has 160.000 rows with both open and closed shops. It contains the following variables:

- business id:index to identify the shop that was reviewed, it was also the connecting data to the review data-set.
- name: name of the shop.
- address: address where the shop is located.
- city: city of the shop.
- state: it indicates the state where the city is located, we only find the abbreviation of the state name.
- postal code: indicates the postal code of the shop.
- latitude: indicates latitude of the shop.
- longitude: indicates longitude of the shop.
- stars: contains the medium rate of the shops, as the variable in Review data-set it can go from 1(bad) to 5(good), but this time it is a float for obvious reasons.
- review count: number of reviews that the shop received from the day it registered to Yelp app.
- is open: it contains a binary variables which is equal to 1 if the store is still open or 0 if the store is closed.
- attributes: it contains a dictionary with several adding features of the store (i.e. bike parking, wi-fi, restaurant takeaway,...). Unfortunately it was not useful because many shops don't have any attributes and additionally few shops have attributes in common.
- categories: contains all the categories of a certain store, which can be more than one.
- hours: contains a dictionary with all the opening days and hours per day of every shop.

### 2.3 User data-set

This data-set contains information about all the registered users of Yelp app. Here we find more that 2 millions users, and many features for each of them. We are going to report only the features that we used during our studies, because the data-set has 22 columns and we only used a couple of them. The following are the features we used for this project:

- user id: index to identify the author of a review, it was also the connecting data to the review data-set.
- review count: number of reviews written by each user.
- friends: contains all the ids of the user friends.

# 3 Evolution in space and time of app usage

In this section we are going to analyze the rapid increase of Yelp app, notice that in this section we will not focus on restaurants yet, but we will see the development of Yelp app in general just to have a further idea about the dimension of the data-set and its rapid usage increasing from 2004 (opening year of Yelp) until today. Furthermore we will find out in which states and cities we have a bigger amount of users, reviews and shops.

## 3.1 Coding libraries used

To make our studies we used pandas library to open 'csv' and 'json' files, pandas allows us to transform these types of data into dataframe which are easy to manipulate and to merge. In particular we made wide use of merge and group by functions, which make the code more fluent. For the plots we used both seaborn and matplotlib libraries, the first one to make the plots with colors adapted to colorblind people and the second one to do the subplots.

## 3.2 Evolution in time

Firstly, we focus on Yelp evolution in time. To have an idea of the dimension of the app we decided to do few bar plots to visualize the number of restaurants, users and reviews year by year registered in Yelp. The first number that must be notice is the number of reviews written in this period of 17 years. We have more than 8 millions reviews. To visualize the distribution in time of the reviews we made a bar plot, in this case we used only the year of the reviews, as a consequence we created a new column containing only year information.

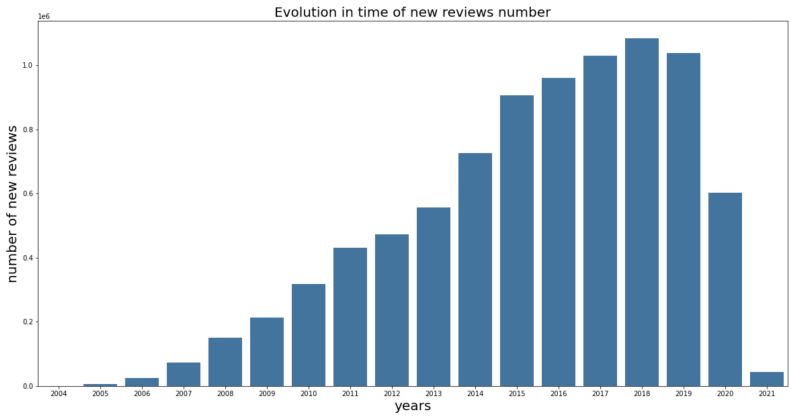



Figure 3: Bar plot with number of reviews posted on Yelp year by year.

As can be seen from the bar plot above Yelp had a rapid increase on its usage from 2004 to 2019 reaching its peak in 2018, in 2004 only 52 reviews were posted while in the last years (exception made for 2020 and 2021) more than a million reviews per year were posted.

Although its rapid and constant increase, we notice a really unexpected decrease in reviews number in the last two years, so the first thing that we have done is understand the causes of this event. For year 2021 the answer is obvious it is sufficient to see that the most recent review is of the 28th January 2021, which means that we do not have

all the data related to 2021. For 2020 the answer is not as obvious as the previous one, because we have all Yelp data related to that year, a decreasing in the number of reviews could be expected because the usage of Yelp app was already decreased from 2018 (year of maximum usage of Yelp) to 2019, but it was not significant. This sharp decrease can only be explained by the fact that the last year almost all the shops were closed because of Covid-19 pandemic and as a consequence the number of people going to restaurants, beauty shops, supermarkets... has fallen and as a consequence also the number of reviews.

We then pass to analyse the number of new active Yelp users year by year. To do so we merged the review data-set with user data-set, we used both the data-set for a simple reason, we didn't care about user registration, because having inactive users is not relevant for our studies but we care about the first users' review.

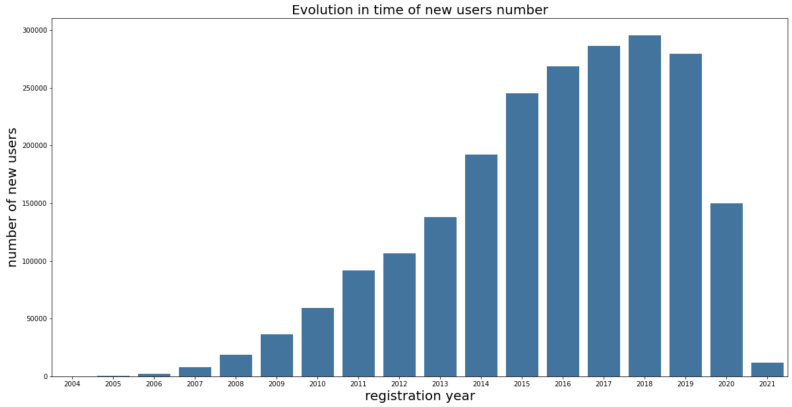



Figure 4: Bar plot with number of new Yelp users year by year.

As this bar plot shows the distribution of users registrations is similar to the previous one, it seems that the number of new users is related to the number of new reviews, consequently we can deduce that many users are not active and also that is not the number of reviews per user which is increasing but only the number of users, which cause an increase in the number of new reviews.

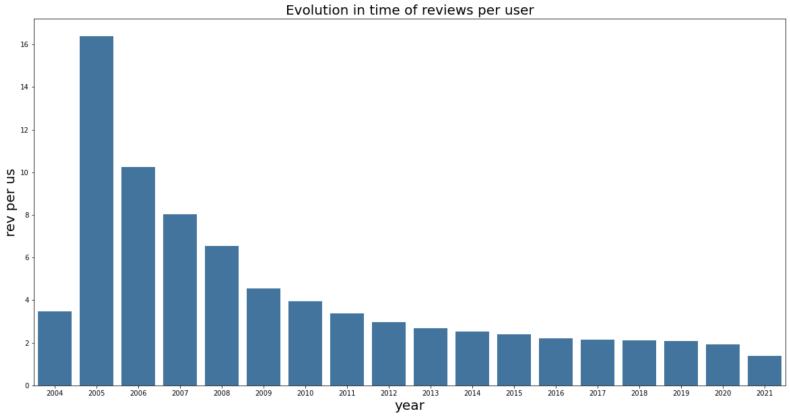



Figure 5: Bar plot with number of reviews per user year by year.

To confirm our analysis we made a rapid analysis on user data-set and we notice that only 700.000 wrote more than 10 reviews and less than a half of them wrote more than 30 reviews, which means that the biggest part of users probably used Yelp only few times (probably only when he register to Yelp app). The bar plot above confirms that the number of reviews per user decreased every year, which means that our conclusion about the increasing number of inactive users is true. Additionally we notice the same rapid decrease seen in 2020's reviews, probably caused by Covid-19 pandemic.

Finally we studied the number of new restaurants registered to Yelp every year. To do so we merged the previous data-set with the business data-set to have also the information about all the shops registered during the years in Yelp app.

As in the case of new users we take the date of the first business review as the date of registration of the business to Yelp app. In this case we could not choose because in the data-set there is not the date of registration for shops.

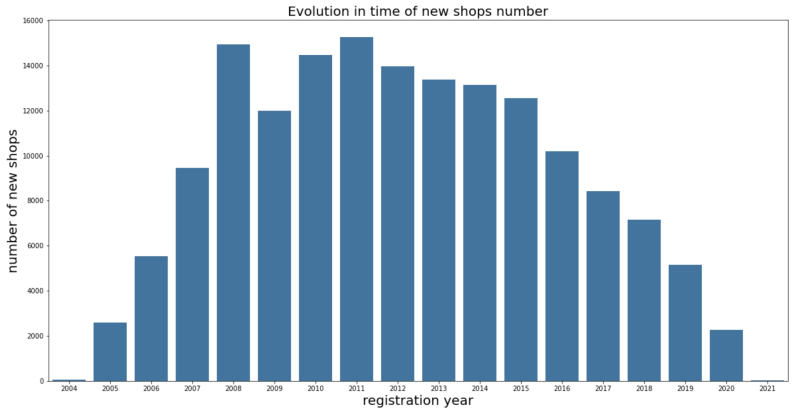



Figure 6: Bar plot with number of new shops year by year.

We immediately notice that the distribution of new restaurants is really different from the previous ones. But the explanation could be given by the fact that probably when the Yelp app opened it had low prices for new shops to enter, because it was not popular and it had no reason to make shops paying much money. While in the last years as we noticed from the previous bar plots it had more than 300.000 new users per year and more than a million reviews per year, which means that many people use it and consequently shops can take advantage of good advertising if they are part of the app. For this reason probably the prices to be part of the app has increased and many activities decided not to do so. Unfortunately we can not check our deduction because searching on the web these information are not available, but we can think that it's true by the fact that the number of new shops has started decreasing when the number of new users had started to increasing consistently.

### 3.3 Evolution in space

We now focus on the evolution in space of Yelp app. Due to the fact the the cities where at least one shop is registered are more than 800, we first decided to focus on the states where the various business are located, which are only 31. As can be seen from the bar plot below, there are several states where the number of shops registered to Yelp app is less then 10 and that the states with more than 100 shops are only 9 and 7 of them have more than 1000 shops, which means that Yelp is popular only in few American states. The most incredible thing that we noticed is that in those states (which are 22) we only find 741 reviews and 719 users, which means that in those states Yelp app has never really entered because there are really few users and less than 2 reviews per person.

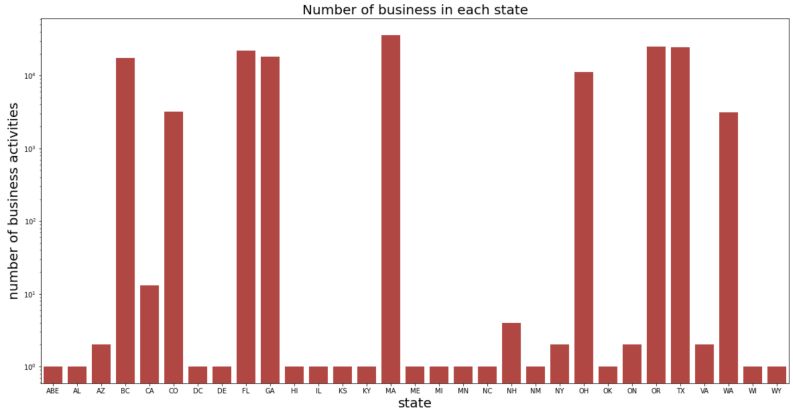



Figure 7: Bar plot with number of Yelp shops in the various states, the bar plot has logarithmic scale on y axis.

Once we have found the states where Yelp is used the most, we go more in the detail and we see how many open activities are there in the various states, because maybe we selected some states where the biggest part of the registered shops is closed and consequently Yelp app is not important any more in those states.

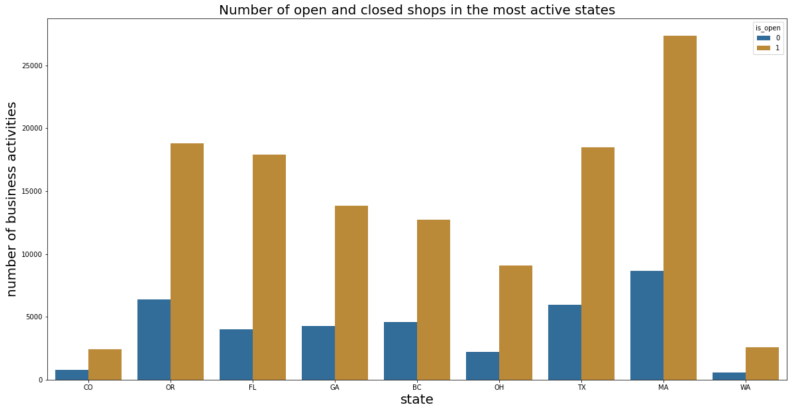



Figure 8: Bar plot with number of Yelp open and closed shops in few states. here the 0 indicates closed shops while 1 indicates open ones.

As we can see from the bar plot it is not our case, in fact the number of open activities is more than twice the number of closed ones in every state.

From this first analysis we have found the states where Yelp app was used and now we focus on the cities where is most used.

From those cities we selected the ones with more than 30000 users to have a clear bar plot of the users per city.

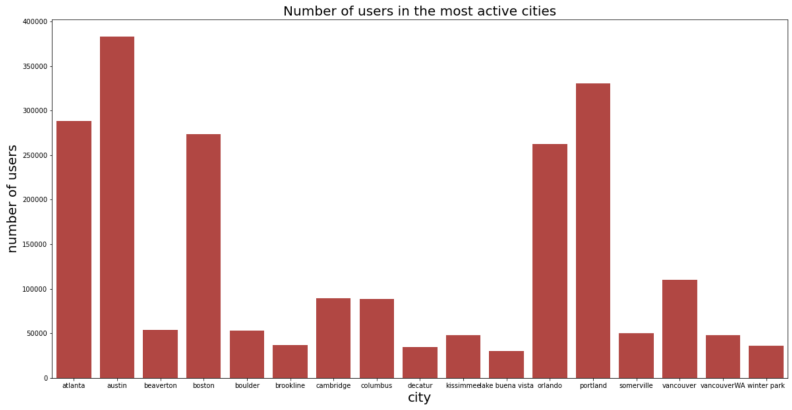



Figure 9: Bar plot with number of Yelp users per city.

As a result of the plot we found that the city with more users is Austin, but there are other cities with almost the same users' number like Atlanta and Portland. As a consequence we can not say that Austin is the city with the biggest usage of Yelp app. It can happen that most of the Austin's users are not active, in fact in the previous analysis we have seen that the number of inactive users is very high. To go more in the deep about it we plot the number of reviews per city.

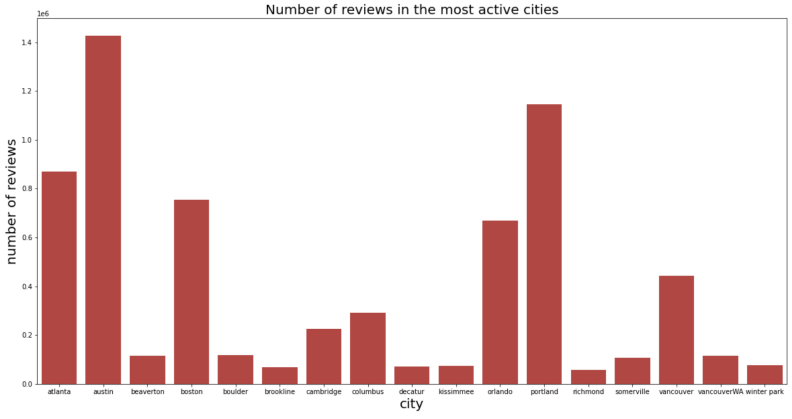



Figure 10: Bar plot with number of Yelp reviews per city.

From this plot is clear that Austin is the city where Yelp app is most used, in fact there's a big difference between the number of reviews of Austin and the other two cities with many users. Additionally we can notice that these three cities are not in Massachusetts (state with the biggest number of shops), which is interesting because it could means that in that state there is more than one city where Yelp is widely used.

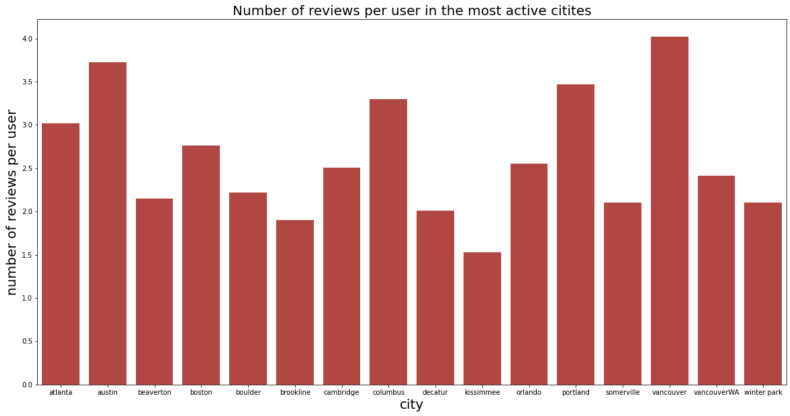



Figure 11: Bar plot with number of Yelp reviews per user in every city.

To understand whether Austin is actually the most active city, we made a bar plot with the number of reviews per user.

From the bar plot above we can notice two different things, the first one is that within the cities with more users Austin is the one with more active people on Yelp, which somehow confirm that is the most active city, but there are many cities (that we didn't notice before because with few users and reviews) that have a big number of reviews per user, for instance Vancouver and Columbus. We actually could expect this phenomenon from the previous analysis, because we already know that the percentage of inactive people is high and as a consequence city with more users could also have more inactive people, this is why Vancouver has more reviews per user than Austin.

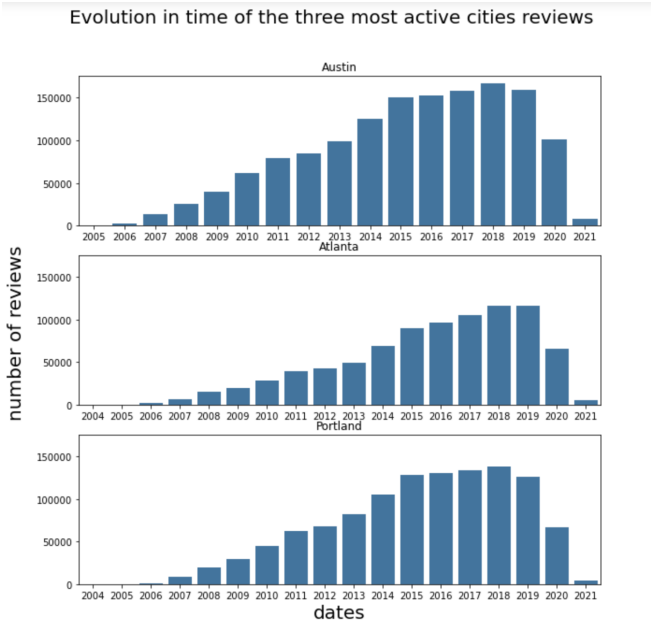



Figure 12: Bar plot with the distribution of reviews year by year for Austin, Atlanta and Portland respectively.

Finally we did a last analysis in the distribution of the number of reviews per year in the three most popular cities we detect above. We expected that these distributions should be similar to the general reviews distribution, because these are the cities with the biggest number of reviews. The bar plots confirms our intuition, in fact the number of new reviews per year follows almost the same development as the total amount of reviews per year, with a peak in 2018 and a rapid decrease in 2020.

### 3.4 Some notes

For this analysis we had to be aware at two factors, the first one is that few cities had two different rows (one in uppercase and the other not) to solve this problem we put all the names in lowercase, the other one was that there are different cities with the same names and for that reason we had to identify them with the name of the state. This is why in the plots we find Vancouver and VancouverWA.

# 4 Geographic analysis ratings

We wanted to do an analysis on geographic ratings of restaurants reviews, in order to have an overview of the data we're going to handle. To this end, we filtered the data-set '*yelp_academic_dataset_business.json*' removing all the activities not related to the restaurant business. This operation was carried out looking for the word 'Restaurants' between the feature *categories* for each business in the data-set.

## 4.1 Cities and States reviews

First of all, as mentioned in the first section, studying the data-set we discovered that there are many cities that have the same name but are allocated in different countries, for example Vancouver is a city in British Columbia as well as a city in Washington, close to Portland. For this reason for each city we always specify what country it belongs to.


Figure 13: Bar plot of the ten cities with the highest amount of business.

The graphic above shows the ten cities with highest amount of businesses reviewed in the Yelp App. For doing that, for each city we assembled together all the different *business_id* and then count them.


Figure 14: Bar plot of the ten cities with the highest amount of reviews.

This graphic represents the ten cities with the highest amount of reviews. For obtaining these results we summed for each city the *review_count* attribute of each *business_id* located there.


Figure 15: Map of the cities with the highest amount of reviews.

As we can notice the order it's not the same, in fact while in the first one Portland is the city with the biggest amount of business reviewed, in the second one we can notice that Austin even if it has less business reviewed than Portland, it has many more reviews for a total of 781596 reviews. On the other hand there are many cities that have only few reviews, in fact, as we can see in the table below, the minimum of city reviews is 5, that is a really low number considering the one Austin has. Other important factors are the standard deviation that is of the order of $10^4$ and the median that is 370 because they represents the great dispersion of the distribution of the total reviews, in fact the *std* is pretty big and the median is quite far from the mean.

| **Min** | **Max** | **Mean** | **Std** | **25%** | **50%** | **75%** |
|---|---|---|---|---|---|---|
| 5 | 781596 | 12576.03 | 69613.92 | 38.00 | 370.00 | 4644.00 |

Figure 16: Characteristics of the city reviews.

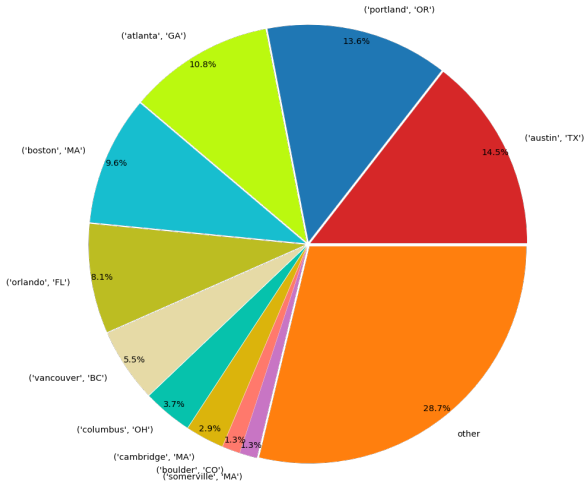



Figure 17: Pie chart of the percentage of reviews for the cities.

For this reason we decided to calculate for each city the percentage of reviews out of the total, and what we discovered and what it is shown in the pie chart above, is that those ten cities owned the 71.3% of the total amount of reviews contained in the Yelp data-set. The total of cities presented in the data-set are 429 but 419 of them represent only the 28.7% of the total of reviews.

We can do the same analysis we just did also for the States. As we can notice, the Yelp App is used exclusively in the North of America, but not in each country. In the above database it is shown all the States we can find in the data and their acronyms, fifteen on them are states of USA, while one of them (*British Columbia*) is a Canadian region.

| | | | |
|---|---|---|---|
| **CO** | Colorado | **OR** | Oregon |
| **BC** | British Columbia | **OH** | Ohio |
| **MA** | Massachusetts | **FL** | Florida |
| **TX** | Texas | **GA** | Georgia |
| **WA** | Washington | **KS** | Kansas |
| **MN** | Minnesota | **VA** | Virginia |
| **WY** | Wyoming | **KY** | Kentucky |
| **NH** | New Hampshire | **ABE** | Mississippi |

Figure 18: Acronyms of the states

The two graphics below represent respectively which are the states with the highest amount of business reviewed and which have the highest amount of reviews.

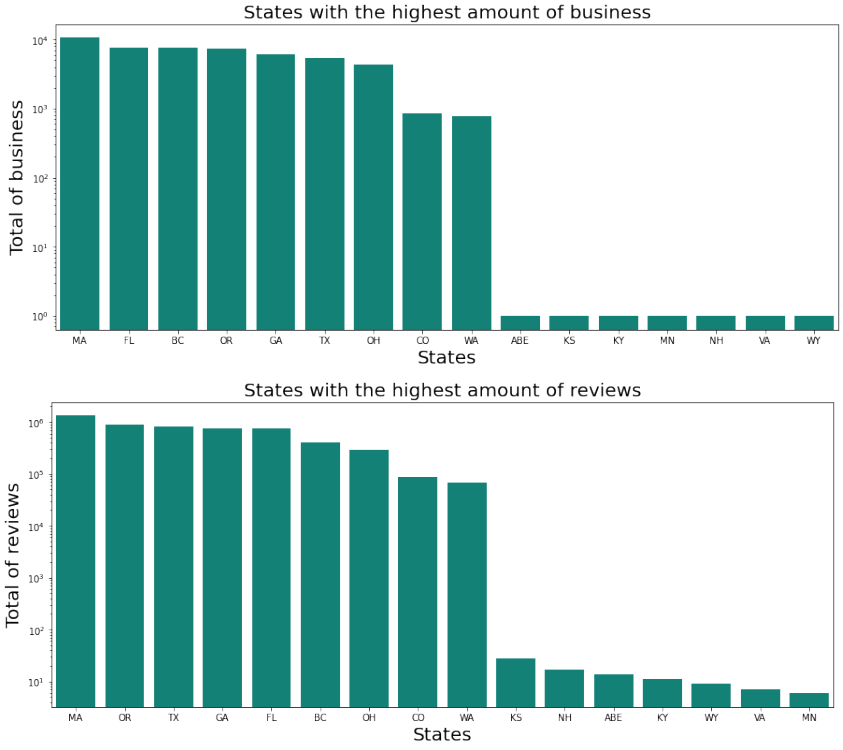



Figure 19: Distribution of business and reviews in States.

We can notice that in this case the State with the highest amount of business is

the same of the one with the biggest amount of reviews, and it is Massachusetts. This result is consistent with what we found earlier, because three cities of the ten cities with the highest amount of reviews are allocated in Massachusetts (Boston, Cambridge and Superville).

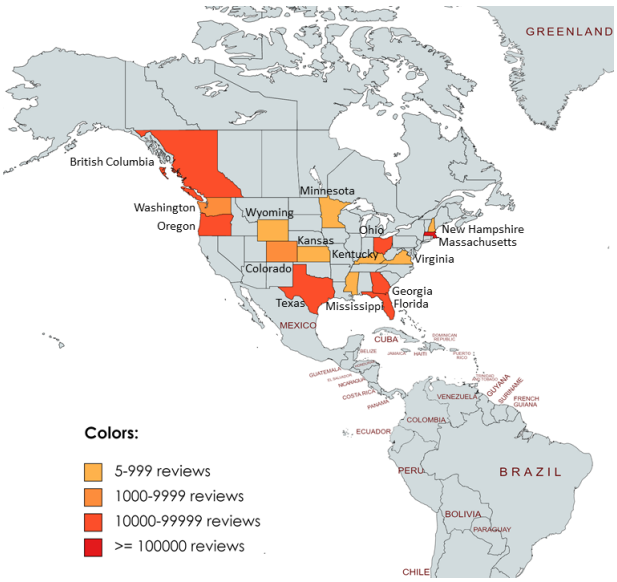



Figure 20: Distribution of business in States.

| **Min** | **Max** | **Mean** | **Std** |
|---|---|---|---|
| 6 | $1.321763e+06$ | $3.371948e+05$ | $4.316259e+05$ |
| **25%** | **50%** | **75%** | |
| $1.325000e+01$ | $7.737650e+04$ | $7.466590e+05$ | |

Figure 21: Characteristics of the State reviews.

In the map in *Figure* 21 we can visualise how those States are distributed in the North America surface. It is quite clear that they're not close to each other, so apparently there's not a specific side of North America where the Yelp App has been used the most for restaurants, but it's usage seems to be scattered in the surface. However, those states are not isolated, as we can see in the map many of them are closed to each other forming different groups, as for example British Columbia, Washington and Oregon. Furthermore, the State with the higher amount of reviews, Massachusetts, is also the smallest one. In the table in *Figure* 21 we show the main characteristics of the distribution of the reviews among the States. In this case as well it's interesting how there are some States as the first nine that have a big amount of reviews (over $10^4$ each), while the other seven have a really small one (under 50 each). The standard deviation is quite high and shows the dispersion of the data. In fact, this is really noticeable in the next pie chart (*Figure* 22) where the first nine State cover the entire surface. The ten cities with the highest amount of reviews are all allocated in those States.

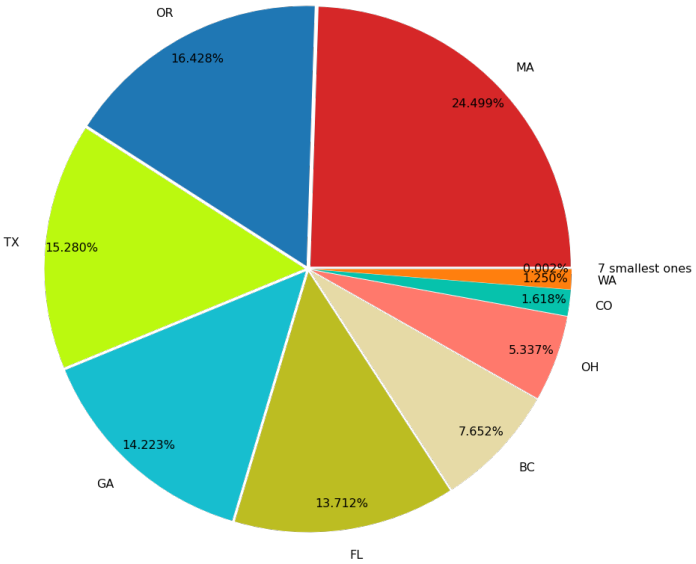



Figure 22: Percentage of reviews per States.

## 4.2 Test differences between rates in different places

An important question which might arise spontaneously is whether there is any correlation between the ratings received by a restaurant and the position of the restaurant.

First of all, we focused our attention on finding the cities which have a better average rating of the restaurants (in the city). Secondly, we have filtered again the data removing all the closed restaurant. Finally, grouping by cities and State, we looked at the cities with a greater average of the attribute *stars* which is a float number from 0 to 5 and represents the medium rating for each business in the data-set. In this way, we obtain the average rating for restaurants in each city in the data. The results are shown below.

| | | stars |
|---|---|---|
| city | state | |
| gotha | FL | 5.0 |
| northeast orlando | FL | 5.0 |
| union park | FL | 5.0 |
| st johns | OR | 5.0 |
| loughman | FL | 4.5 |
| ... | ... | ... |
| sheridan | WY | 1.5 |
| e.point | GA | 1.5 |
| clarkson | GA | 1.5 |
| englewood | OH | 1.5 |
| miami beach | FL | 1.0 |

Figure 23: Cities with the highest average scores.

At first sight it seems that there are a lot of cities with perfect score (five stars over five). However, we could imagine that many of this cities have a low number of restaurants and/or a low number of reviews. Thanks to the analysis done before the hypothesis is verified: there are a lot of cities with few reviews for the restaurants in the city. We observe the same by plotting the distribution of the review count below.

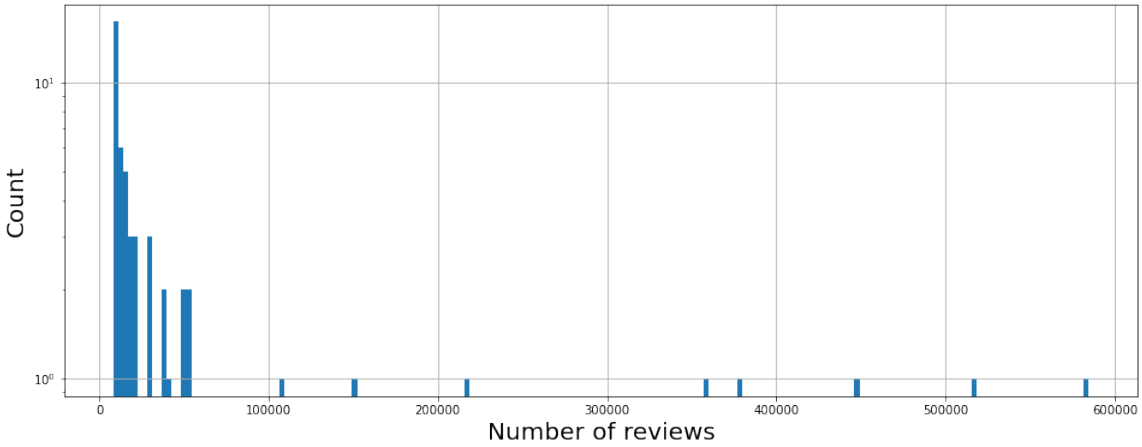



Figure 24: Distribution of review counts.

Since it is not meaningful to consider the average rating for a small number of reviewed restaurants, we decided to include in our analysis only the cities with a number of reviewed restaurants greater than 8000. Now, including only these cities and repeating the same procedure explained above (grouping by city and averaging the *stars* attribute) we obtain that the ten cities with the greatest rating average are the following.

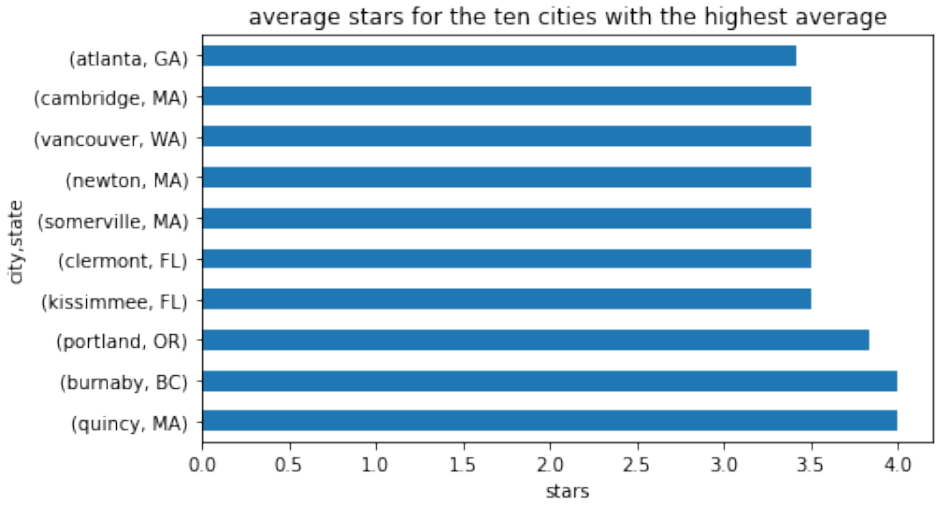



Figure 25: Average stars for the first ten cities with the highest average.

The city with the greatest average is Quincy with 4.00 as the average rating. A question that might arise at this moment is the following: 'Are all these ten cities close to each other ?'. A simple spatial plot shows that this is not the case therefore there seems to be no association between great rating and spatial position of the cities.

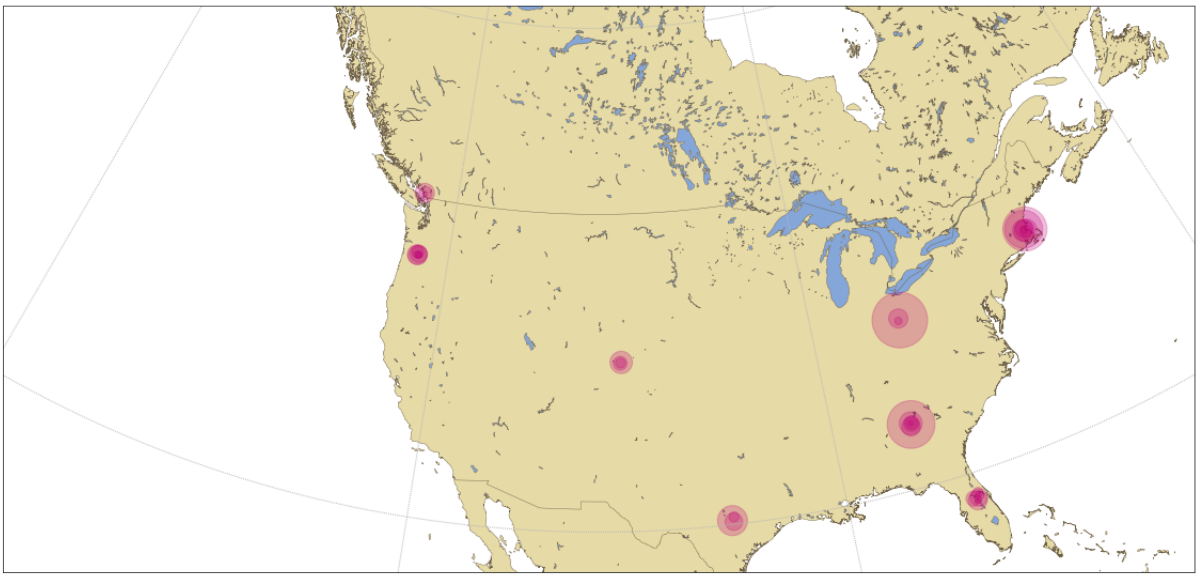


Figure 26: Spatial position of the cities with the highest reviews.

# 5 Rates and time

An important question which might arise spontaneously is whether there is any correlation between the ratings received by a restaurant and the moment in which the review was made by the user.

## 5.1 Checking if the time of the review influences the rating for restaurants

The second question that we want to address in this sub-section is whether there is a statistically significant difference between the ratings and the moment in which the user made the review for restaurant businesses. In order to access the date of each review of a given restaurant we had to use the data-set 'reviews' which contains each review of each business. The first step was to merge the filtered '*yelp_academic_dataset_business.json*' data-set with the review restaurant; in this way we had a single data-set containing for each review (of a given restaurant) all the information about the review (stars, date, user_Id,...) but also the information about the restaurant reviewed (city, latitude, longitude,...). Then, to reduce the computational cost and save time, we analyzed a sample of six thousands rows of the resulted data-set. We dropped every column of the data-set a part from the column 'star' which represents the rating of a single review and the column 'date' which is the date of the review. Then, we grouped by year and we plotted the average rating for each year. Here below we show how the plot looks like.

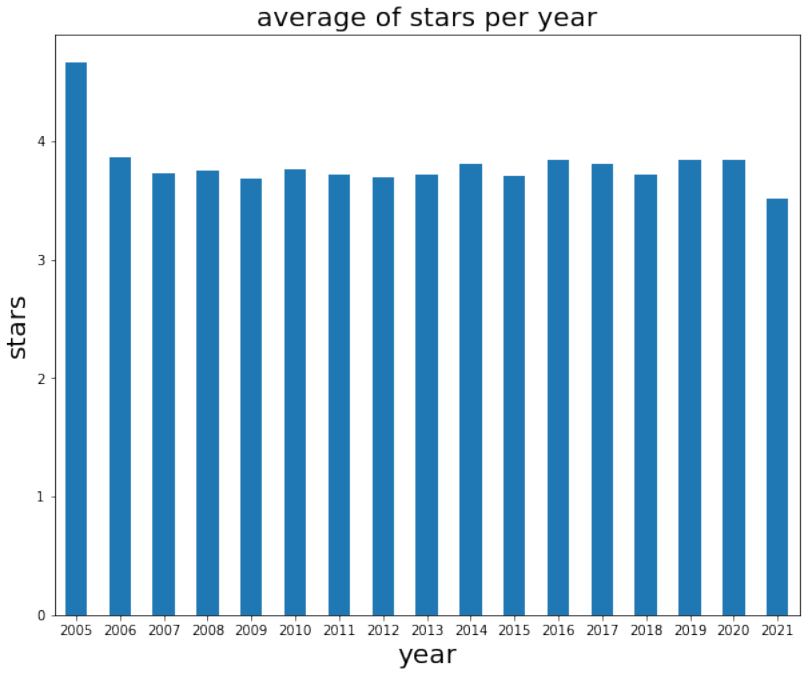



Figure 27: Average stars per year.

From the plot it seems that the year in which the reviews were the most positive was 2005. However, is this difference in mean significant ? To answer this question we used **bootstrap** to compute the confidence intervals for the mean for each year. Bootstrap is a popular and powerful method to compute confidence intervals for the mean of a population. The method consists in re-sampling from the population $N$ samples with

replacement (not to loose independence), computing the mean for each sample and taking as the confidence interval the values inside the interval $[a, b]$ where $a$ is the value of the lower(2.5 percent) quantile and $b$ is the value of the upper (97.5 percent) quantile. Adding now the confidence intervals to the plot we obtain the following result.

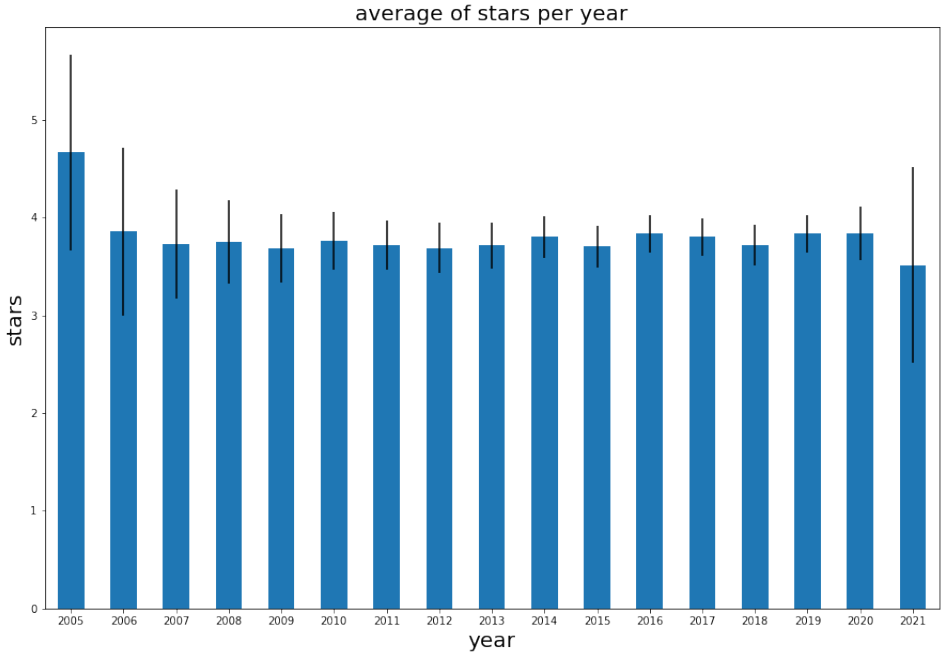



Figure 28: Average stars per year with bootstraps confidence intervals

From the plot it seems that there is not a significant difference between the means of different years for most of the pair of years since the confidence intervals are mostly overlapping. To deeply investigate, we consider a two sided T-test to compare the difference in mean between each pair of years $2005 - 2021$. Here below we report the values of the p-values corresponding to the t-test applied to each pair with null hypothesis that the two samples have the same mean.

| | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2005 | 1.00 | 0.15 | 0.13 | 0.17 | 0.18 | 0.19 | 0.16 | 0.17 | 0.18 | 0.23 | 0.24 | 0.28 | 0.29 | 0.27 | 0.31 | 0.33 | 0.21 |
| 2006 | 0.15 | 1.00 | 0.68 | 0.74 | 0.62 | 0.76 | 0.66 | 0.62 | 0.67 | 0.87 | 0.69 | 0.95 | 0.88 | 0.72 | 0.95 | 0.96 | 0.43 |
| 2007 | 0.13 | 0.68 | 1.00 | 0.90 | 0.82 | 0.86 | 0.94 | 0.82 | 0.93 | 0.68 | 0.90 | 0.58 | 0.72 | 0.96 | 0.60 | 0.61 | 0.43 |
| 2008 | 0.17 | 0.74 | 0.90 | 1.00 | 0.67 | 0.95 | 0.80 | 0.64 | 0.77 | 0.69 | 0.75 | 0.55 | 0.73 | 0.83 | 0.57 | 0.58 | 0.33 |
| 2009 | 0.18 | 0.62 | 0.82 | 0.67 | 1.00 | 0.55 | 0.78 | 0.98 | 0.82 | 0.30 | 0.87 | 0.20 | 0.34 | 0.79 | 0.22 | 0.25 | 0.47 |
| 2010 | 0.19 | 0.76 | 0.86 | 0.95 | 0.95 | 1.00 | 0.69 | 0.50 | 0.64 | 0.66 | 0.60 | 0.46 | 0.69 | 0.70 | 0.48 | 0.50 | 0.27 |
| 2011 | 0.16 | 0.66 | 0.94 | 0.80 | 0.80 | 0.82 | 1.00 | 0.75 | 0.95 | 0.34 | 0.89 | 0.19 | 0.37 | 1.00 | 0.21 | 0.24 | 0.34 |
| 2012 | 0.17 | 0.62 | 0.82 | 0.64 | 0.64 | 0.72 | 0.72 | 1.00 | 0.79 | 0.19 | 0.85 | 0.09 | 0.21 | 0.74 | 0.10 | 0.14 | 0.43 |
| 2013 | 0.18 | 0.67 | 0.93 | 0.77 | 0.82 | 0.64 | 0.95 | 0.79 | 1.00 | 0.27 | 0.93 | 0.13 | 0.29 | 0.94 | 0.14 | 0.19 | 0.37 |
| 2014 | 0.23 | 0.87 | 0.68 | 0.69 | 0.30 | 0.66 | 0.34 | 0.19 | 0.69 | 1.00 | 0.22 | 0.69 | 0.98 | 0.29 | 0.69 | 0.70 | 0.19 |
| 2015 | 0.24 | 0.69 | 0.90 | 0.75 | 0.87 | 0.60 | 0.89 | 0.85 | 0.93 | 0.22 | 1.00 | 0.086 | 0.21 | 0.86 | 0.088 | 0.14 | 0.43 |
| 2016 | 0.28 | 0.95 | 0.58 | 0.55 | 0.20 | 0.46 | 0.19 | 0.09 | 0.13 | 0.69 | 0.086 | 1.00 | 0.64 | 0.12 | 0.99 | 0.96 | 0.17 |
| 2017 | 0.29 | 0.88 | 0.72 | 0.73 | 0.34 | 0.69 | 0.37 | 0.21 | 0.29 | 0.98 | 0.21 | 0.64 | 1.00 | 0.27 | 0.64 | 0.68 | 0.24 |
| 2018 | 0.27 | 0.72 | 0.96 | 0.83 | 0.79 | 0.70 | 1.00 | 0.74 | 0.94 | 0.29 | 0.86 | 0.12 | 0.75 | 1.00 | 0.12 | 0.19 | 0.42 |
| 2019 | 0.31 | 0.95 | 0.60 | 0.57 | 0.22 | 0.48 | 0.21 | 0.10 | 0.14 | 0.69 | 0.088 | 0.99 | 0.64 | 0.12 | 1.00 | 0.98 | 0.19 |
| 2020 | 0.33 | 0.96 | 0.61 | 0.58 | 0.25 | 0.50 | 0.24 | 0.14 | 0.19 | 0.70 | 0.14 | 0.96 | 0.68 | 0.19 | 0.98 | 1.00 | 0.21 |
| 2021 | 0.21 | 0.43 | 0.43 | 0.33 | 0.47 | 0.27 | 0.34 | 0.43 | 0.37 | 0.19 | 0.43 | 0.17 | 0.24 | 0.42 | 0.19 | 0.21 | 1.00 |

Table 5.1: p-values for the T-test for each pair of years

As we can see from the table there are a few significant p-values if we consider a confidence level $\alpha = 0.1$ and no significant result for a confidence level $\alpha = 0.05$. It

seems that in 2016 the reviews were slightly more positive with respect to the ones of 2012 and 2015 and that in 2019 the reviews were slightly more positive than the ones of 2015. We repeat now the same reasoning for the months.

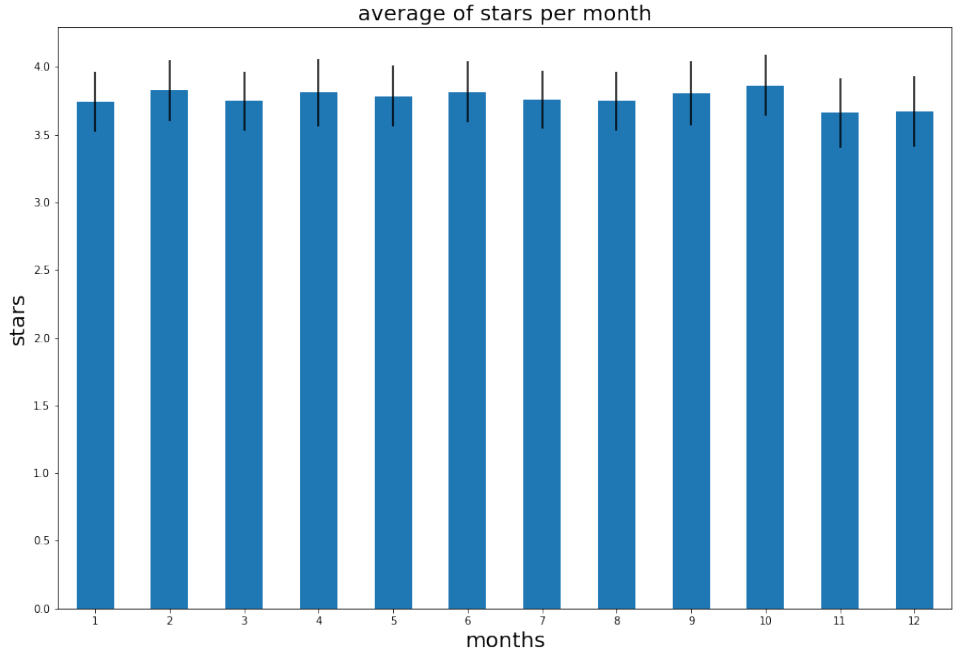



Figure 29: Average stars per month with confidence intervals

Even in this case the confidence intervals are mostly overlapping therefore we cannot conclude that there is a significant difference in means just by looking at the plot. Again, to further investigate, we repeat the T-test to discover any significant difference in mean. Here below is reported the table of the p-values for the t-test applied to each pair of months.

| | january | february | march | april | may | june | july | august | september | october | november | december |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| january | 1.00 | 0.28 | 0.96 | 0.41 | 0.59 | 0.36 | 0.83 | 0.95 | 0.45 | 0.14 | 0.34 | 0.38 |
| february | 0.89 | 1.00 | 0.32 | 0.83 | 0.61 | 0.87 | 0.39 | 0.32 | 0.78 | 0.67 | 0.054 | 0.063 |
| march | 0.96 | 0.32 | 1.00 | 0.45 | 0.64 | 0.40 | 0.87 | 1.00 | 0.48 | 0.16 | 0.32 | 0.35 |
| april | 0.41 | 0.83 | 0.45 | 1.00 | 0.77 | 0.96 | 0.54 | 0.45 | 0.95 | 0.54 | 0.096 | 0.11 |
| may | 0.59 | 0.61 | 0.64 | 0.77 | 1.00 | 0.72 | 0.75 | 0.64 | 0.82 | 0.36 | 0.16 | 0.19 |
| june | 0.36 | 0.87 | 0.40 | 0.96 | 0.72 | 1.00 | 0.49 | 0.40 | 0.91 | 0.56 | 0.074 | 0.087 |
| july | 0.83 | 0.39 | 0.87 | 0.54 | 0.75 | 0.49 | 1.00 | 0.88 | 0.58 | 0.20 | 0.25 | 0.29 |
| august | 0.95 | 0.32 | 1.00 | 0.45 | 0.64 | 0.40 | 0.88 | 1.00 | 0.48 | 0.16 | 0.32 | 0.36 |
| september | 0.45 | 0.78 | 0.48 | 0.95 | 0.82 | 0.91 | 0.58 | 0.48 | 1.00 | 0.50 | 0.11 | 0.12 |
| october | 0.14 | 0.67 | 0.16 | 0.54 | 0.36 | 0.56 | 0.20 | 0.16 | 0.50 | 1.00 | 0.022 | 0.026 |
| november | 0.34 | 0.054 | 0.32 | 0.096 | 0.16 | 0.074 | 0.25 | 0.32 | 0.11 | 0.022 | 1.00 | 0.93 |
| december | 0.38 | 0.063 | 0.35 | 0.11 | 0.19 | 0.087 | 0.29 | 0.36 | 0.12 | 0.026 | 0.66 | 1.00 |

Table 5.2: p-values for the T-test for each pair of months

From the values above we can conclude that there are few statistically significant differences considering a confidence level $\alpha = 0.1$ and only two significant differences with a confidence level $\alpha = 0.05$. Generally, november and december have a slightly statistically significant smaller average with respect to some of the other months (with respect to october for example).

Finally it could be interesting to repeat once again the same analysis with the reviews grouped by season instead of the reviews grouped by month. The result is shown below.

| | summer | autumn | winter | spring |
|---|---|---|---|---|
| summer | 1.00 | 0.89 | 0.88 | 0.59 |
| autumn | 0.89 | 1.00 | 0.99 | 0.52 |
| winter | 0.88 | 0.99 | 1.00 | 0.50 |
| spring | 0.59 | 0.52 | 0.50 | 1.00 |

Table 5.3: p-values for the t-test for each pair of seasons

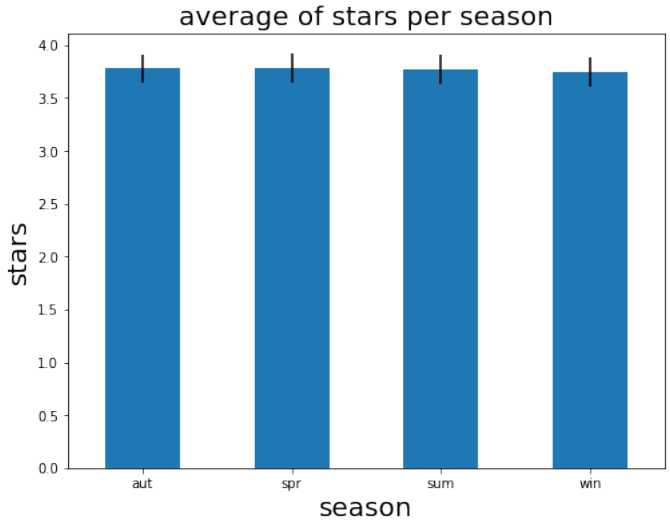



Figure 30: Average stars per season with confidence intervals

Even in this case the confidence intervals don't help us detecting any significant difference in mean. In addition, also the t-test cannot find any statistically significant difference in mean with the standard confidence levels used in statistics.

To summarize, we can conclude that it is hard to find periods in which the ratings are more positive on average. From our analysis it seems that the years 2016 and 2019 have a statistically significant better average than some of the other years and that november and december have a statistically smaller average than some of the other months. We cannot draw any meaningful conclusion regarding the seasons.

## 5.2 Checking if the time of the review influences the rating for general businesses

In the analysis above, we have considered only reviews concerning restaurants and we did not consider reviews concerning any other business. However, we could ask ourselves if the same differences in the ratings for different months/years/seasons observed in the previous analysis are the same that we would observe for the other businesses as well. To answer this question we repeat now the analysis for general business activities. First of all, we select a sample with the same size as before from the reviews file. Then we proceed by grouping the reviews by year, by calculating the average for each year and by using once again bootstrapping to compute the confidence intervals for the mean of each year. Once again we plot the result.

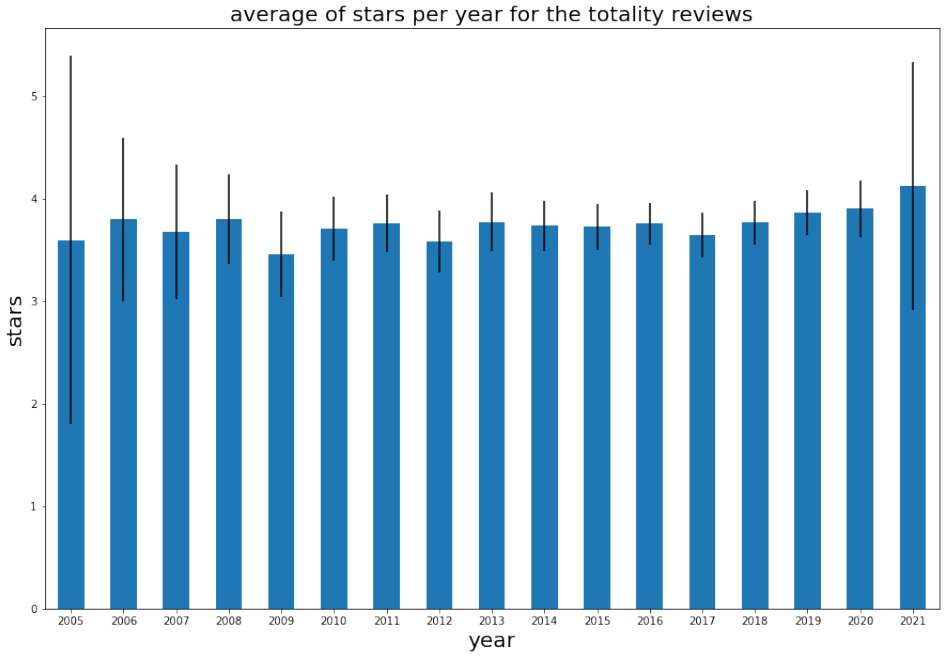



Figure 31: Average stars per year with confidence intervals for all businesses

We can also repeat the t-test for each pair of years. In the table below we will report the p-values for each pair of years.

| | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2005 | 1.00 | 0.71 | 0.89 | 0.69 | 0.82 | 0.84 | 0.78 | 0.99 | 0.77 | 0.82 | 0.84 | 0.81 | 0.95 | 0.80 | 0.69 | 0.64 | 0.49 |
| 2006 | 0.71 | 1.00 | 0.70 | 0.99 | 0.27 | 0.76 | 0.89 | 0.52 | 0.95 | 0.85 | 0.83 | 0.90 | 0.65 | 0.93 | 0.84 | 0.75 | 0.44 |
| 2007 | 0.89 | 0.70 | 1.00 | 0.54 | 0.30 | 0.87 | 0.68 | 0.67 | 0.63 | 0.77 | 0.81 | 0.71 | 0.87 | 0.68 | 0.38 | 0.29 | 0.19 |
| 2008 | 0.69 | 0.99 | 0.54 | 1.00 | 0.037 | 0.53 | 0.76 | 0.19 | 0.87 | 0.68 | 0.63 | 0.78 | 0.33 | 0.83 | 0.69 | 0.53 | 0.26 |
| 2009 | 0.82 | 0.27 | 0.30 | 0.037 | 1.00 | 0.055 | 0.018 | 0.35 | 0.014 | 0.028 | 0.036 | 0.021 | 0.17 | 0.017 | 0.0017 | 0.00093 | 0.026 |
| 2010 | 0.84 | 0.76 | 0.87 | 0.53 | 0.055 | 1.00 | 0.67 | 0.29 | 0.54 | 0.78 | 0.87 | 0.66 | 0.54 | 0.59 | 0.14 | 0.083 | 0.13 |
| 2011 | 0.78 | 0.89 | 0.68 | 0.76 | 0.018 | 1.00 | 0.67 | 1.00 | 0.11 | 0.85 | 0.75 | 1.00 | 0.24 | 0.92 | 0.26 | 0.15 | 0.18 |
| 2012 | 0.99 | 0.52 | 0.67 | 0.19 | 0.35 | 0.29 | 1.00 | 1.00 | 0.068 | 0.13 | 0.15 | 0.08 | 0.58 | 0.061 | 0.0043 | 0.0029 | 0.081 |
| 2013 | 0.77 | 0.95 | 0.63 | 0.87 | 0.014 | 0.54 | 0.11 | 0.068 | 1.00 | 0.68 | 0.58 | 0.82 | 0.14 | 0.92 | 0.32 | 0.19 | 0.24 |
| 2014 | 0.82 | 0.85 | 0.77 | 0.68 | 0.028 | 0.78 | 0.85 | 0.13 | 0.68 | 1.00 | 0.89 | 0.83 | 0.25 | 0.73 | 0.13 | 0.075 | 0.20 |
| 2015 | 0.84 | 0.83 | 0.81 | 0.63 | 0.036 | 0.87 | 0.75 | 0.15 | 0.58 | 0.89 | 1.00 | 0.71 | 0.29 | 0.61 | 0.087 | 0.052 | 0.19 |
| 2016 | 0.81 | 0.90 | 0.71 | 0.78 | 0.021 | 0.66 | 1.00 | 0.08 | 0.82 | 0.83 | 0.71 | 1.00 | 0.14 | 0.89 | 0.17 | 0.10 | 0.23 |
| 2017 | 0.95 | 0.65 | 0.87 | 0.33 | 0.17 | 0.54 | 0.24 | 0.58 | 0.14 | 0.25 | 0.29 | 0.14 | 1.00 | 0.10 | 0.0046 | 0.0041 | 0.13 |
| 2018 | 0.80 | 0.93 | 0.68 | 0.83 | 0.017 | 0.59 | 0.92 | 0.061 | 0.92 | 0.73 | 0.61 | 0.89 | 0.10 | 1.00 | 0.20 | 0.12 | 0.25 |
| 2019 | 0.69 | 0.84 | 0.38 | 0.69 | 0.0017 | 0.14 | 0.26 | 0.0043 | 0.32 | 0.13 | 0.087 | 0.17 | 0.0046 | 0.20 | 1.00 | 0.67 | 0.41 |
| 2020 | 0.64 | 0.75 | 0.29 | 0.53 | 0.00093 | 0.083 | 0.15 | 0.0029 | 0.19 | 0.075 | 0.052 | 0.10 | 0.0041 | 0.12 | 0.67 | 1.00 | 0.48 |
| 2021 | 0.49 | 0.44 | 0.19 | 0.26 | 0.026 | 0.13 | 0.18 | 0.081 | 0.24 | 0.20 | 0.19 | 0.23 | 0.13 | 0.25 | 0.41 | 0.48 | 1.00 |

Table 5.4: p-values for the T-test for each pair of years for reviews from all businesses

What we can notice from these p-values is that the year 2009 has a statistically significant smaller mean with respect to most of the years. This result seems totally independent of the result seen before for the restaurant reviews.

We repeat the same analysis by month. Here below we show the results.

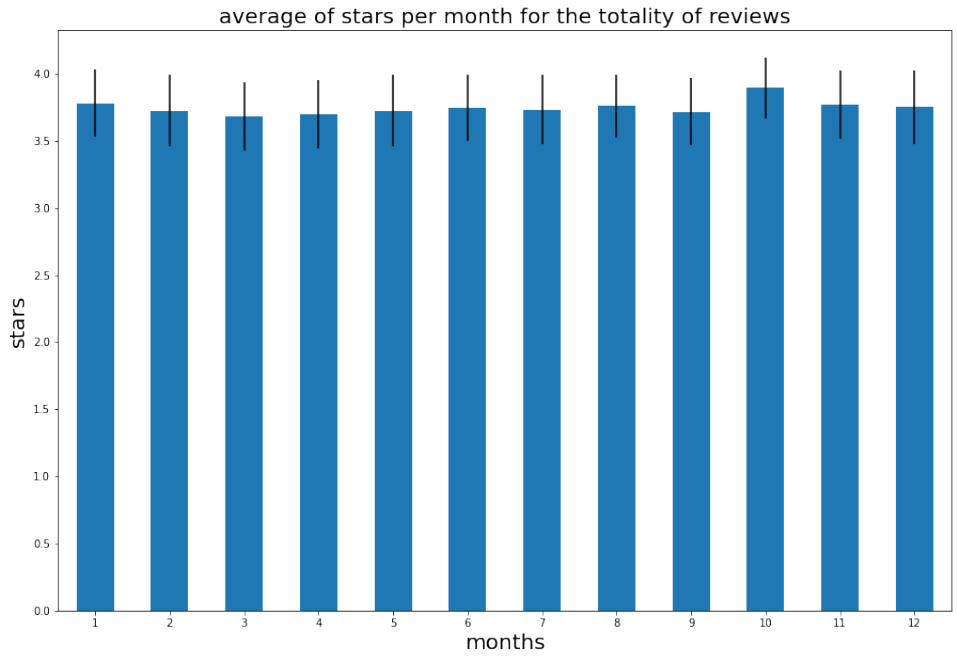



Figure 32: Average stars per month with confidence intervals for all businesses

| | january | february | march | april | may | june | july | august | september | october | november | december |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| january | 1.00 | 0.55 | 0.28 | 0.36 | 0.54 | 0.71 | 0.58 | 0.80 | 0.47 | 0.19 | 0.89 | 0.74 |
| february | 0.28 | 1.00 | 0.65 | 0.77 | 1.00 | 0.81 | 0.95 | 0.72 | 0.91 | 0.055 | 0.64 | 0.80 |
| march | 0.28 | 0.65 | 1.00 | 0.87 | 0.65 | 0.47 | 0.59 | 0.40 | 0.73 | 0.014 | 0.35 | 0.47 |
| april | 0.36 | 0.77 | 0.87 | 1.00 | 0.77 | 0.58 | 0.71 | 0.50 | 0.85 | 0.023 | 0.44 | 0.58 |
| may | 0.54 | 1.00 | 0.65 | 0.77 | 1.00 | 0.81 | 0.95 | 0.71 | 0.91 | 0.052 | 0.64 | 0.79 |
| june | 0.71 | 0.81 | 0.47 | 0.58 | 0.81 | 1.00 | 0.85 | 0.90 | 0.72 | 0.089 | 0.82 | 0.98 |
| july | 0.58 | 0.95 | 0.59 | 0.71 | 0.95 | 0.85 | 1.00 | 0.76 | 0.86 | 0.056 | 0.68 | 0.84 |
| august | 0.80 | 0.72 | 0.40 | 0.50 | 0.71 | 0.90 | 0.76 | 1.00 | 0.63 | 0.11 | 0.91 | 0.93 |
| september | 0.47 | 0.91 | 0.73 | 0.85 | 0.91 | 0.72 | 0.86 | 0.63 | 1.00 | 0.038 | 0.56 | 0.71 |
| october | 0.19 | 0.055 | 0.014 | 0.023 | 0.052 | 0.089 | 0.056 | 0.11 | 0.038 | 1.00 | 0.15 | 0.10 |
| november | 0.89 | 0.64 | 0.35 | 0.44 | 0.64 | 0.82 | 0.68 | 0.91 | 0.056 | 0.15 | 1.00 | 0.84 |
| december | 0.74 | 0.80 | 0.47 | 0.58 | 0.79 | 0.98 | 0.84 | 0.93 | 0.71 | 0.10 | 0.84 | 1.00 |

Table 5.5: p-values for the T-test for each pair of months

From the p-values we can conclude that october has a statistically significant bigger mean than the other months. Again this result does not relate with the previous analysis. Finally, we repeat the same analysis for the seasons again considering all the business and not only restaurants. Here below we display the results.

| | summer | autumn | winter | spring |
|---|---|---|---|---|
| summer | 1.00 | 0.33 | 0.90 | 0.40 |
| autumn | 0.33 | 1.00 | 0.42 | 0.071 |
| winter | 0.90 | 0.42 | 1.00 | 0.34 |
| spring | 0.40 | 0.071 | 0.34 | 1.00 |

Table 5.6: p-values for the t-test for each pair of seasons for all businesses

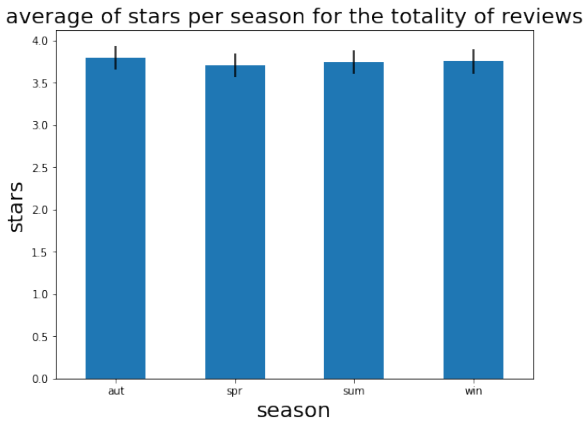



Figure 33: Average stars per season with confidence intervals for all businesses

In this case the only statistically significant result for $\alpha = 0.1$ is that autumn and spring have different means (spring has a smaller mean). Again this result does not relate to the previous analysis for the seasons and the restaurant reviews.

We can then conclude that there is not a period of time in which the reviews related to all businesses are generally higher but there are some meaningful fluctuations of the average of the rating depending on the business we consider. However, generally speaking it seems that during the autumn (in particular october) the reviews are generally slightly more positive.

# 6 Analysis of Mexican restaurants

Yelp is a compelling platform for selecting restaurants. It allows users to filter out based on ratings and look at what other people have said about the restaurant they want to try. Then it becomes much easier to find a restaurant that you may like. However, this rating system can become a double-edged sword as our decisions are affected by other people's ratings. A subset of users can introduce biases that are unrelated to gastronomical preference, like ethnicity. Then, because of the rating system, all the users somewhat endorse this ethnic bias implicitly as their choices depend on the biased group ratings. In this section, we want to analyze Mexican restaurants in detail to determine if there is any bias towards them and analyze what their customers complain about.

First of all, we show how Mexican restaurants are distributed across the United States. It can be critical to our study as it is well known that some regions of the US have stronger opposition to immigrants.

**Geographic analysis of Mexican restaurants**

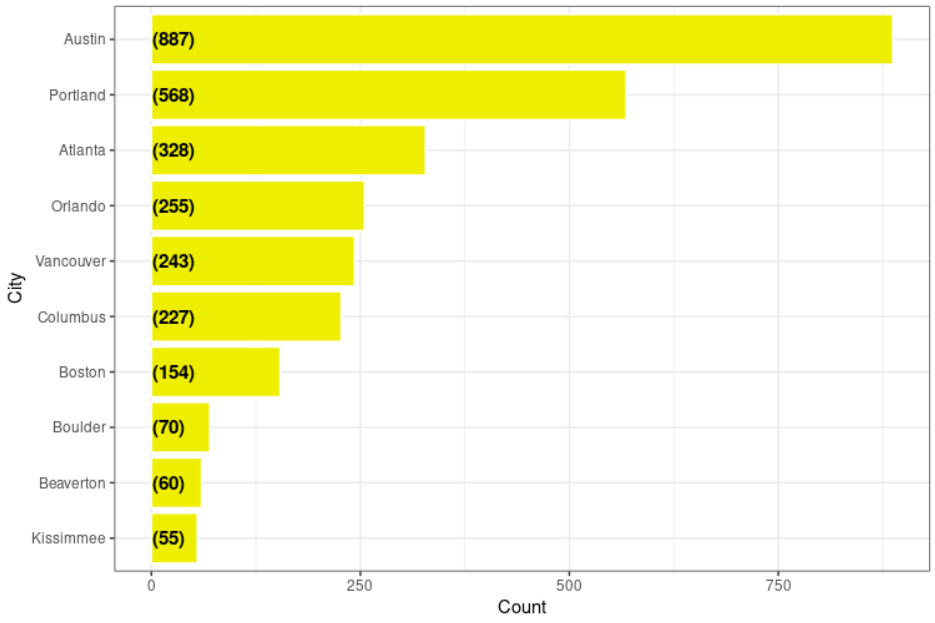



Figure 34: In this figure we see which cities have the most part of Mexican restaurants.

Now we want to test if there is a significant difference between the average stars of the Mexican restaurants compares to the rest. In practice we observe that the mean rating of the Mexican restaurants is lower but we want to check if it is statistically significant that they are actually different. We used t-test, as we thought that it would be robust for large sample (like ours) despite violating the normality assumption. To confirm our results we also used the Wilcoxon test that has no distributional assumptions.

$$\begin{cases} H_0: & \mu_{mex} = \mu_{general} \\ H_1: & \mu_{mex} \neq \mu_{general} \end{cases}$$

| **t-test:** $p - value < 2 \cdot 10^{-16}$ | **Wilcoxon:** $p - value < 2 \cdot 10^{-16}$ |
|---|---|

Now we compare Mexican restaurants to American restaurants driven by the fact that there must be no negative ethnic bias in the later as we are studying reviews in the US.

**American restaurants**

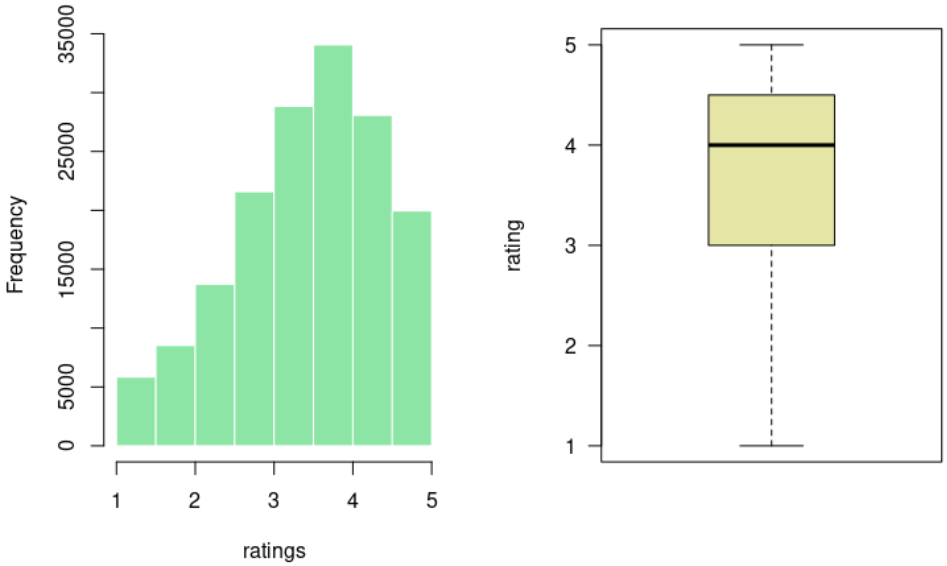



Figure 35: Histogram & boxplot for ratings of American restaurants

**Mexican restaurants**

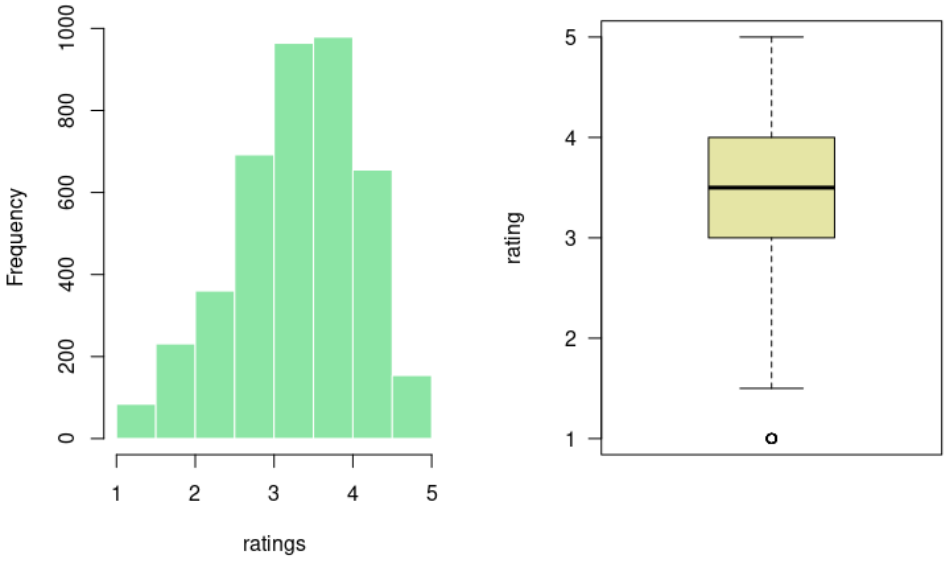



Figure 36: Histogram & boxplot for ratings of Mexican restaurants

We observe that despite the fact that American restaurants have a higher averaging ratings this may not be due to unjustified hate reviews. In fact the number of very bad reviews is proportionally much higher among American restaurants. This is compensated with a much heavier upper tail, that leads to a higher average.

Despite not having many bad reviews compared to American restaurants we still want to understand the bad reviews of Mexican restaurants. In order to do this we analyzed the text in the reviews of those restaurants that had the word "Mexican" in categories. We found that the most common words are the following:

**Most common words**


Figure 37: Word cloud with the most common words in Mexican restaurants reviews.

We were a bit surprised by the word truck, but it seems that there are many food trucks among Mexican restaurants. In the next plot we can get a better idea about which features are the most common among Mexican restaurants.

**Most common features**

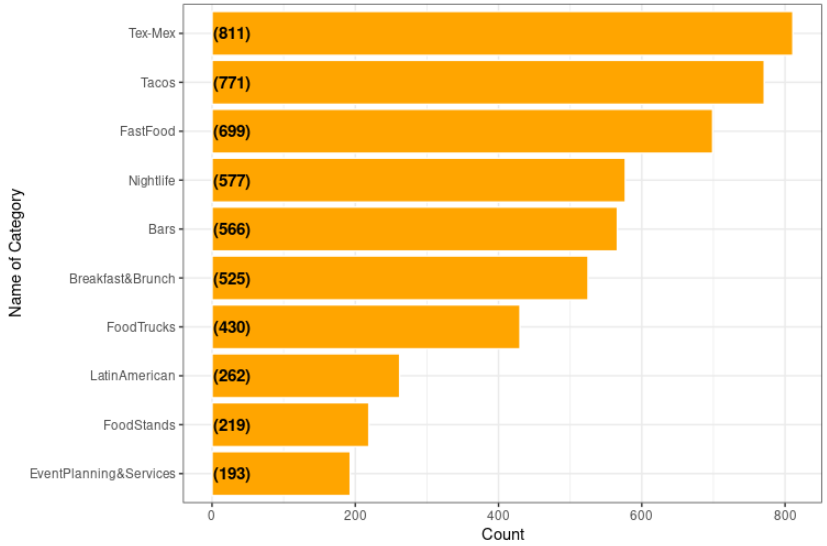



Figure 38: Histogram with most common shared categories among Mexican restaurants.

Finally we restrict our analysis to the text of those reviews with 2 stars or less.

**Common words in negative reviews**


Figure 39: Word cloud with the most common words in negative reviews of Mexican restaurants.

We observe that there are no hate words and that the complaints are the prevalent for restaurants in general. They can divided mainly in three groups:

- **Food:** taco, burrito, food, cheese.
- **Service:** waiting, people, service, rude.
- **Location:** place, drive, location.

| Word | Frequency |
|---|---|
| food | 10125 |
| order | 6348 |
| taco | 5800 |
| time | 4981 |
| service | 4226 |
| place | 4164 |
| location | 4132 |
| bell | 3267 |
| minutes | 2765 |
| chipotle | 2621 |

Table 6.1: This table shows the most common words in negative reviews and their frequency.

Two remarkable words that appear in the previous table are taco and bell. Taco Bell is an American-based chain of fast food specialized in Mexican meals. The slightly lower ratings in Mexican restaurants could be affected by Taco Bell restaurants. There is no reason to believe that the reviews in these restaurants are ethnically biased as they are no authentic Mexican restaurants but just an American chain. As a result we decided to compare Mexican restaurants with Americans again removing the fast food restaurants.

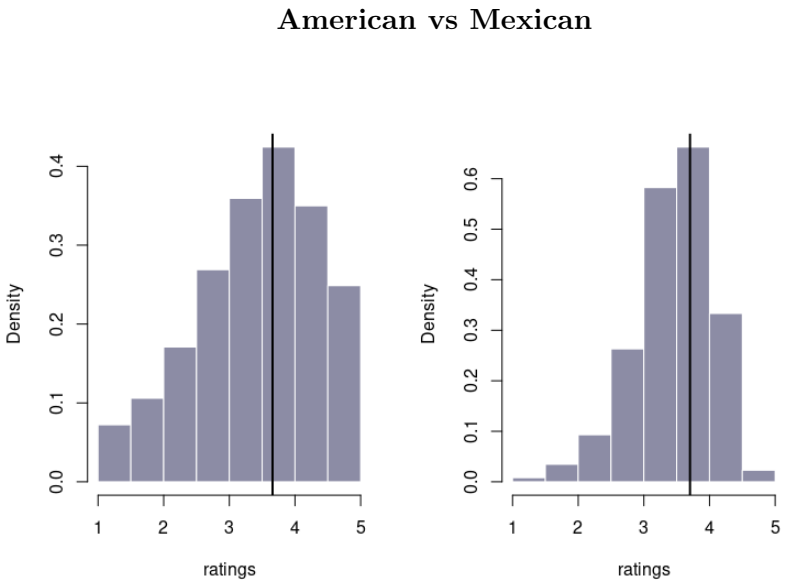



Figure 40: Comparison of rating ditribution between American (left) and Mexican (right) restaurants without fast food chains.

We observe that now the mean of the two groups are very similar, (Americans: 3.65, Mexicans: 3.7). The main difference is that the tails of the Americans are much heavier than for the Mexican restaurants. Therefore, from the observed data we can say that Mexican restaurants are less likely to receive very bad reviews. Hence there is no reason to believe that there is any ethnic bias in the reviews towards them.

# 7 Feature importance

The goal of this section is to analyze the effect of different factors in the restaurant rating. We will use both, statistical analysis and visualization

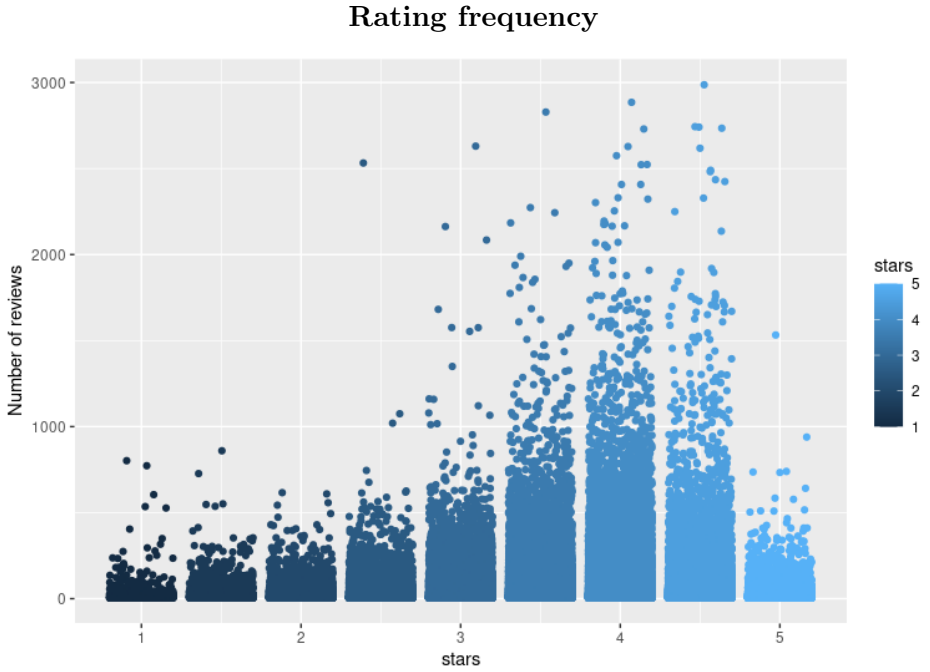



Figure 41: In this plot each restaurant is represented by a dot where x is the average rating and y is the number of reviews.

We observe that the restaurants with more reviews tend to be located in the bins of more than 3 stars. However the converse statement is not necessarily true from the previous plot. There are many restaurant with high ratings and less than 500 reviews.

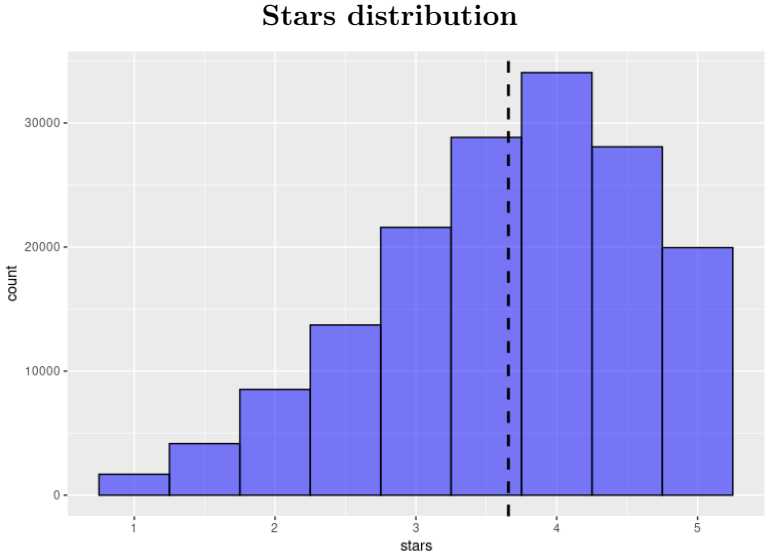



Figure 42: Histogram of the different ratings and their frequencies with 0.5 steps as in the dataset. The dotted line is the mean rating which is 3.657.


Figure 43: Boxplots of the stars for different groups of restaurants classified by the price range. The price range is increasing (e.g 1 is the cheapest and 4 the most expensive).

We do not observe differences between the first three groups. The last one has a similar mean and third quantile as the others but a much lower first quantile.

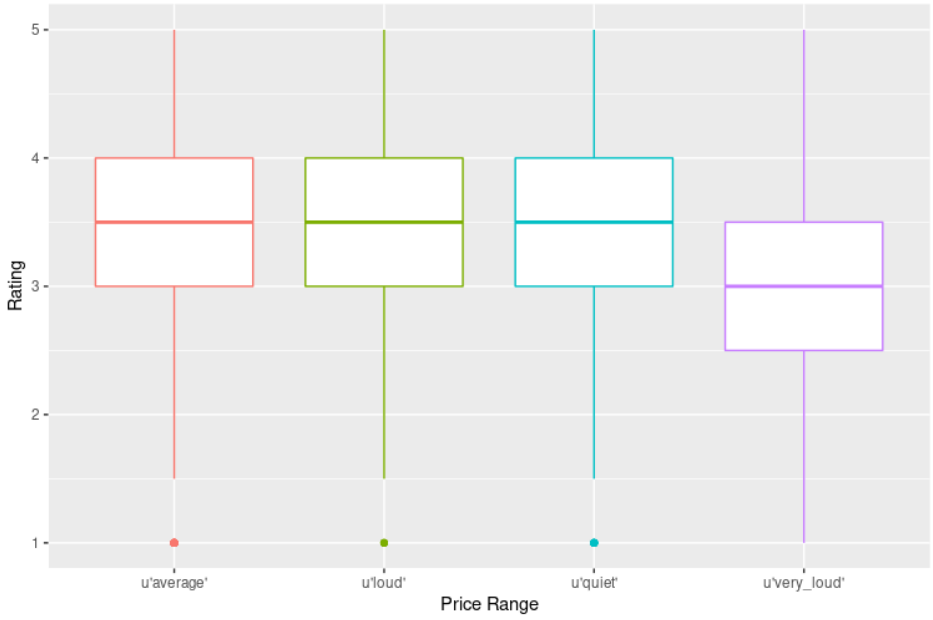
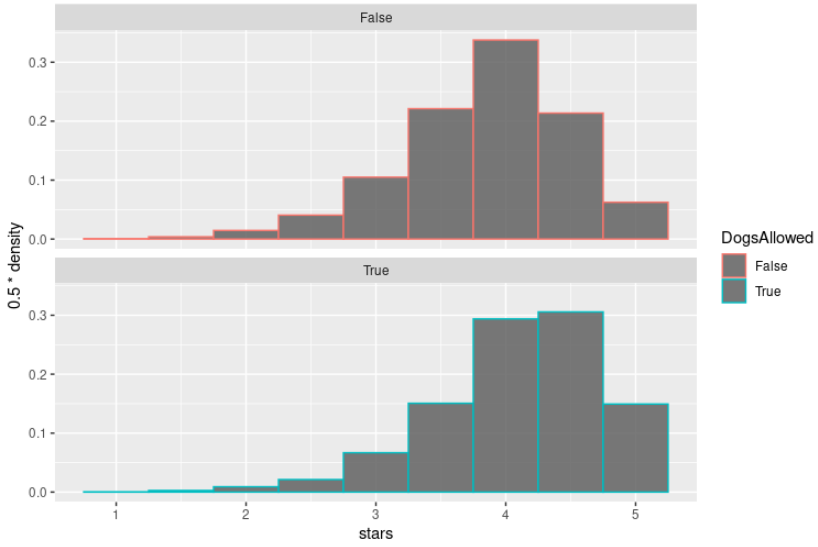



Figure 44: Boxplots of the stars for different restaurants divided by the noise.

We cannot observe differences between the first three groups. However the last one (very loud) is much more different. In fact the whole IQR is moved down by one quantile.

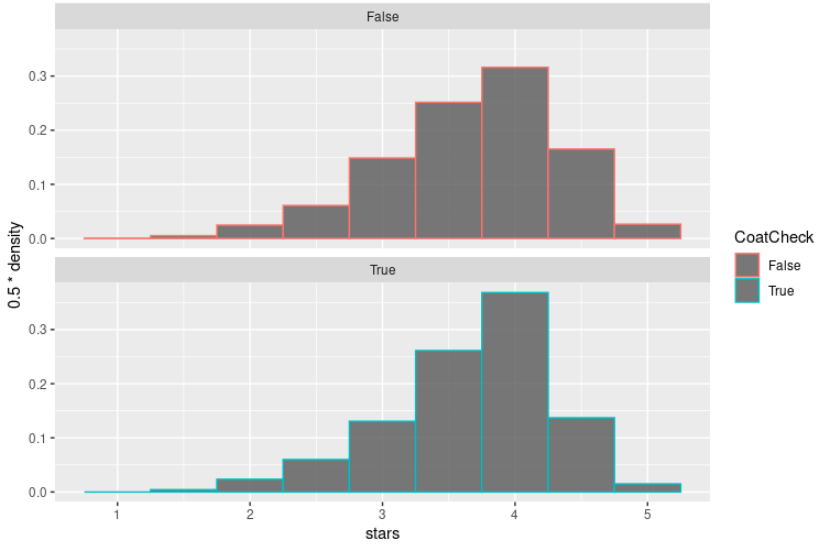
We tested if there was a significant difference between the average stars of the restaurants for different attributes. First we used t-test, as we thought that it would be robust despite having discrete variables. This statement is supported by the fact that for large sample (like ours) the mean of the discrete variable will become almost continuous and normally distributed. Nevertheless to be completely sure we also used the Wilcoxon test that has no distributional assumptions. We performed this tests for all the attributes (see github). Here we show some of examples of both significant and not significant attributes.

**Dogs: allowed vs forbidden**




**t-test:** $p - value < 2 \cdot 10^{-12}$ **Wilcoxon:** $p - value < 2 \cdot 10^{-14}$

**Coat check**




**t-test:** $p - value = 0.902$ **Wilcoxon:** $p - value = 0.873$

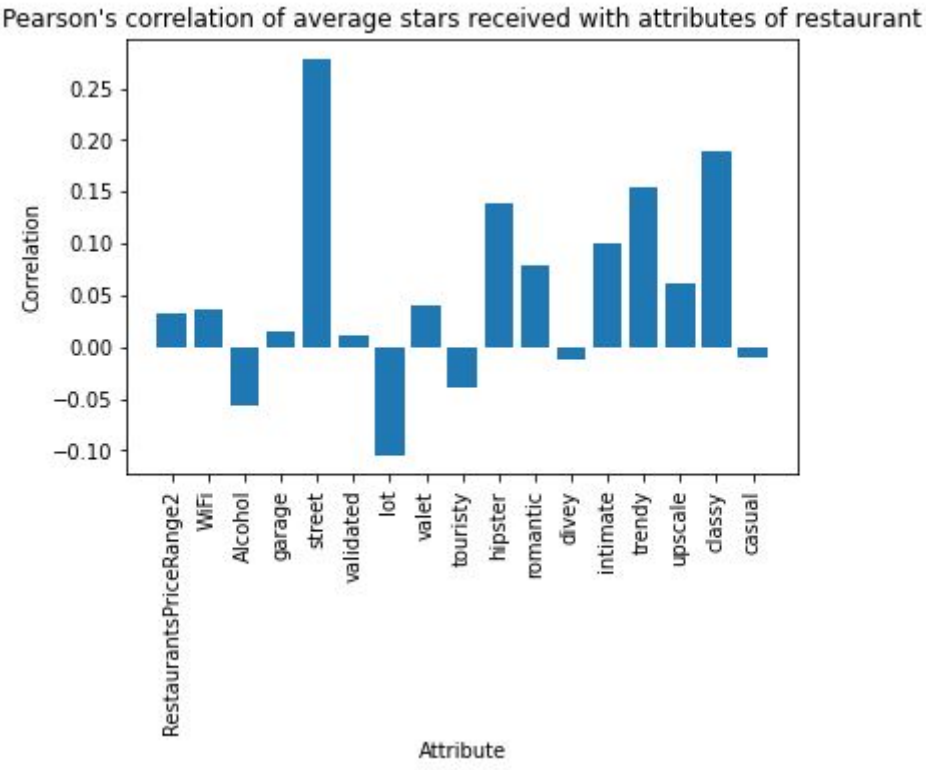
## 7.1 Correlation analysis

### 7.1.1 Cleaning

Our data contained a column named "attributes" which contained, in the form of a dictionary, all the available attributes for a particular restaurant. For every restaurant, the average stars obtained were available under the column "stars". To extract which features were most relevant for a good review, we had to first clean the attributes related column, as a first step we removed the restaurants for which no attributes were available. Then since the type of attributes available was dependent on the restaurant itself, we decide to fetch the most frequently present attributes between all the available restaurants and then select only the restaurants that had **ALL** the aforementioned attributes to proceed for our analysis. Of course this procedure reduced the dimension of our data, but we still had $\sim 2600$ rows available, other cleaning was necessary, such as transform sub-attributes from dictionary form to multiple columns and then transform all categorical variables into numeric ones.

### 7.1.2 Pearson's correlation

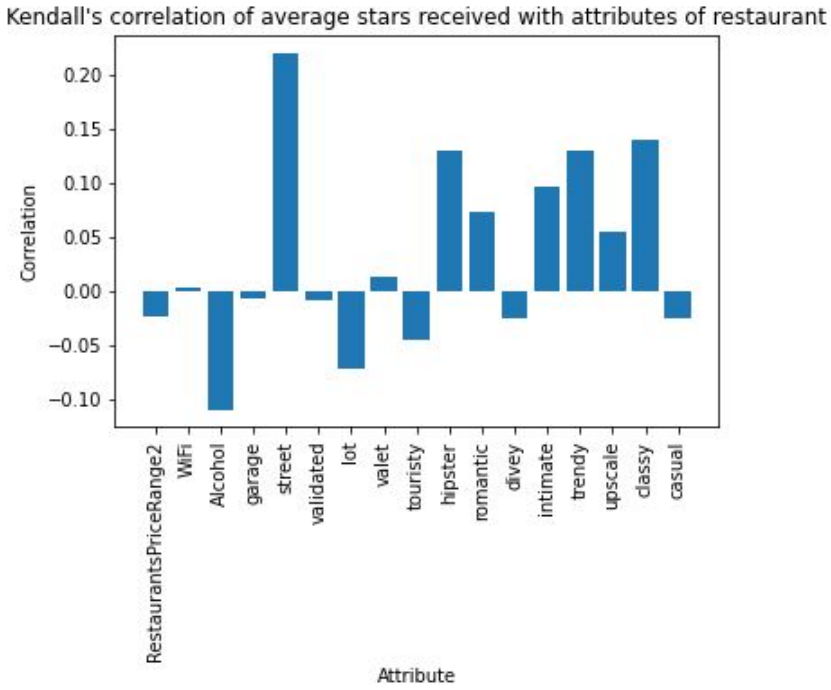
Pearson's correlation[1] is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations; thus it is essentially a normalised measurement of the covariance, such that the result always has a value $\in [-1, 1]$. For the pearson's correlation method, the most

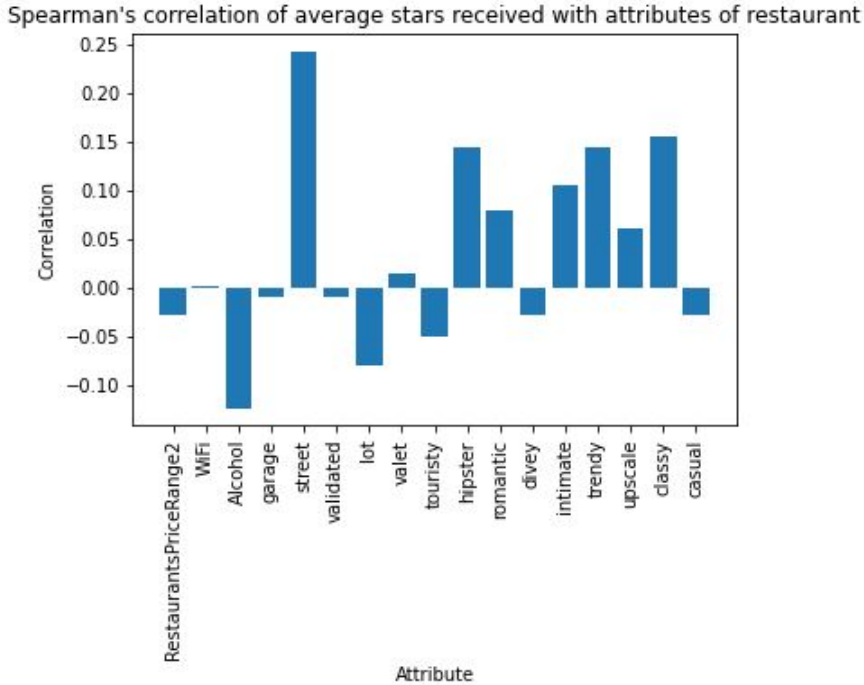



relevant attributes for a restaurant that positively correlate with an higher average star review are: street parking availability, classy environment, trendy environment.

### 7.1.3 Alternative correlations

Kendall and spearman correlation are special cases of rank correlation coefficients [2], they are scalar measures of dependence that depend only on the copula of a bivariate distribution of the data and not on the marginal distributions. Their standard empirical estimators may be calculated by considering the ranks of the data only, hence the name. Following those methods, the most relevant attributes for a restaurant that posi-

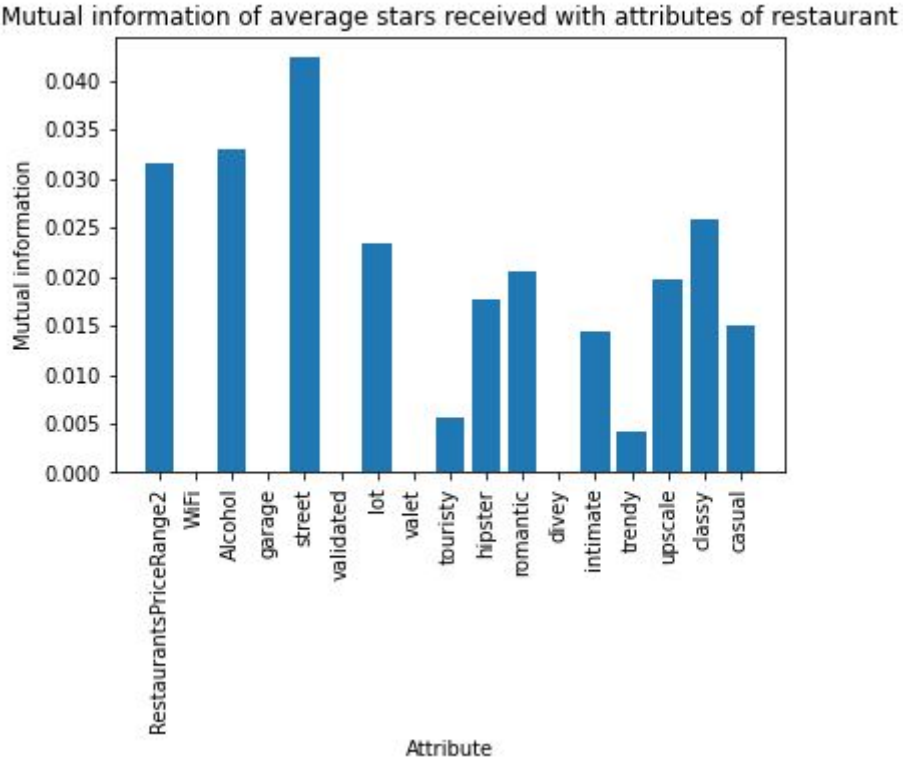



tively correlate with an higher average star review are: street parking availability, classy environment, touristy environment.

### 7.1.4 Mutual information

The mutual information of two random variables is a measure of the mutual dependence between the two variables, it quantifies the "amount of information" obtained about one random variable by observing the other random variable. The concept of mutual information is intimately linked to that of entropy of a random variable, a fundamental notion in information theory that quantifies the expected "amount of information" held in a random variable.[3] For this method, the most relevant attributes for a restaurant that




positively correlate with an higher average star review are: street parking availability, alcohol availability, cost of the restaurant (high price paid).

# 8 Predict whether a restaurant is closed or not

The aim of this section is to predict whether, with all the characteristics of the restaurants that we have in Yelp app, we can build a model which could predict the closure of a restaurant. In this case we used only the business data-set and the reviews data-set, the other one had not interesting variables to construct an effective model in our opinion.

## 8.1 Preprocess of data

We started to explore the variables we have in the business data-set. From this data-set we had to transform the written variables into numeric variables. Both the 'attributes' and the 'hours' variables where dictionaries of strings. We first concentrated on the 'attributes' column of the dataframe, which at first sight could be a very useful variable because it contains a big number of characteristics attributed to the shops. Unfortunately from a more careful analysis we found that the data contained in this column were not complete, in fact we found too many missing data and also many restaurants with different attributes, which give no possibility to use them for our analysis. We did also a rapid check to find whether there could be some common attributes, but we didn't have positive results and consequently we decided not to use this variable. We then concentrate on the opening days variable, this variable was made by dictionaries containing strings with the opening days and hours per day. Obviously this variable appear us too important to achieve our goal, so we can not get rid of it. Consequently we implemented a function to convert this dictionaries of strings into a float which indicates the number of opening hours per week of a certain shop. Before applying this function to our data-set we decided to create a new dataframe containing only the restaurants (the only shops we were interested in) just to be sure not to do useless operations. We then decided to concentrate only on a city, to do so we searched the city with the biggest number of restaurants, which is Portland, then before deciding to keep it we check that the number of opening restaurants was similar to the closed one, once we checked that Portland respects this request we get rid of all the restaurants out of this city.

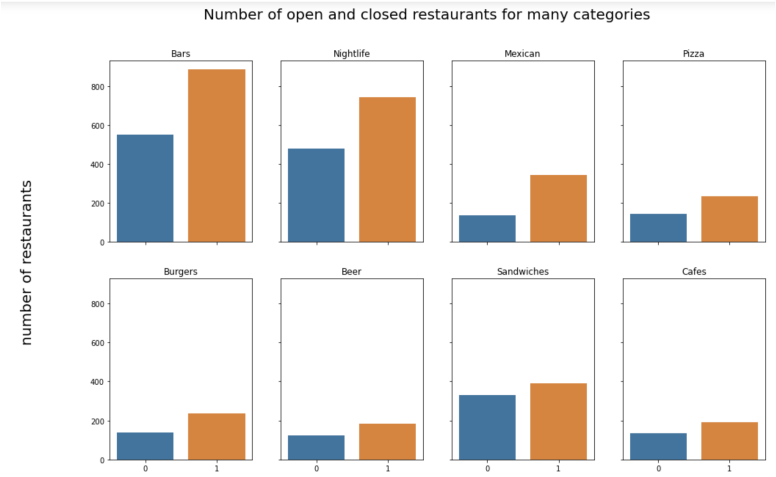



Figure 45: Countplot with the number of open and closed restaurants for the most common categories. Where 0 indicates closed restaurants and 1 the open ones.

Once obtained a fitter dataframe we started working on the categories. Once again this variable was difficult to handle because every restaurant had a different number of

categories. But at the same time we believed that the category of a restaurant could be important to reach our goal. To solve this problem we decided to find the most common categories. From this analysis we have found that the categories in the count-plot above were the most common.

Because, as we can observe from the count plot, the number of restaurants with those categories was not irrelevant we decided to create eight new columns (one for each category) containing 1 where a restaurant has that category and 0 otherwise. Once we obtained those information from that variable, we removed the column of the categories. Once we have finished to clean the business data-set we decided to concentrate on the reviews data-set. We decided to use also this data-set because, although we already had the rating of each restaurant in the other data-set, that data in the business data-set contained only the medium stars attributed to a certain restaurant which could be useless for our analysis. In fact if we consider a restaurant that was popular in 2010 and that few years later changed administration and started to fail till the closure it may happen that the medium rate is high. To avoid this possible event we merged the previous dataframe with the reviews' one. Once merged we sorted the reviews basing on the date they were posted and we made the mean of the 10 newest reviews for each restaurant and we created a new variable with this mean. We then did the same with the newest 20 reviews. Once added these two columns we had our dataframe was ready to find a model to predict the closure of a restaurant.

## 8.2 Models

Now that we collected and preprocessed the data, we can build a model to predict whether a restaurant is actually closed or not. In order to do it, we will use two different models: Logistic Regression and Support Vector Machine. The first is a classical and popular model in Machine Learning which is interpretable but often lack on performance since its decision boundaries are linear in the predictors. On the other hand the second is way less interpretable but has often better performances. Before starting we follow the standard procedure in Machine Learning of dividing the data in training and test data. We will train our model on the training set and we will evaluate its final performance on the test set.

### 8.2.1 Logistic Regression

Logistic Regression is a popular machine learning model used for binary classification (class labels are 0,1) tasks which assumes that the probability that $y$ (the class label) is one given the observed values of $x$ (features) and $w$ (a parameter that has to be found) is:

$$P(y = 1|x, w) = \sigma(x^T w + w_0)$$

Where $\sigma(x) = \frac{e^x}{1+e^x}$ is the sigmoid function. The model will then predict 1 if $P(y = 1|x, w) \geq \frac{1}{2}$, it will predict 0 otherwise. Under the assumption that the input $x$ does not depend on the parameter $w$, we have that the likelihood is:

$$P(y, x|w) = P(x|w)P(y|x, w) = P(x)P(y|x, w)$$

Where we have used the independence of $x$ and $w$. In order to maximize the likelihood with respect to $w$ it is sufficient to maximize $P(y|x,w)$. We have:

$$P(y|x,w) = \prod_{i=1}^{n} P(y_i|x_i,w) = \prod_{i:y_i=1} P(y_i=1|x_i,w) \prod_{i:y_i=0} P(y_i=0|x_i,w) =$$
$$= \prod_{i=1}^{n} \sigma(x_i^T w)^{y_i}(1-\sigma(x_i^T w))^{1-y_i}$$

Computing now the negative of the logarithm of the likelihood $P(y|x,w)$ we have after a small computation that

$$L(w) = -\log(p(y|x,w)) = \sum_{i=1}^{n} -y_i x_i^T w + \log(1+e^{x_i^T w})$$

Therefore, from the maximum likelihood principle, we have that the parameter $w^*$ which minimizes the negative log likelihood is the best parameters for our predictions: $w^* = \text{argmin}L(w)$. It could be easy to show that the negative log likelihood $L(w)$ is a convex function, so its minima could be find simply using the gradient descent algorithm or one of his variations (stochastic gradient descent or batch stochastic gradient descent). The logistic regression algorithm will then work as follow:

- We split the data in training and test data. We normally use a 80-20 percent split which means that 20 percent of the data will be used for the test set and 80 percent for the training.
- We split again both the training and test set in $x$ (the features) and $y$ (the labels).
- We apply gradient descent (or a variation of it) to the negative log likelihood for the training set to find the best hyperparameter $w^*$.
- We use $w^*$ to make the prediction for the test set. The model will then predict 1 if $P(y=1|x,w) \geq \frac{1}{2}$, it will predict 0 otherwise
- We evaluate the performance on the test set comparing the predictions and the true labels.

We applied the above explained machine learning model to our dataset using the library statsmodel. Here below is the summary of the results.

```
Optimization terminated successfully.
         Current function value: 0.628217
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                is_open   No. Observations:                 3918
Model:                          Logit   Df Residuals:                     3904
Method:                           MLE   Df Model:                           13
Date:                Sun, 14 Nov 2021   Pseudo R-squ.:                 0.07615
Time:                        17:41:37   Log-Likelihood:                -2461.4
converged:                       True   LL-Null:                       -2664.2
Covariance Type:            nonrobust   LLR p-value:                 1.363e-78
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept         -2.6880      0.269     -9.977      0.000      -3.216      -2.160
mean_of_recent_10  0.4894      0.114      4.312      0.000       0.267       0.712
mean_of_recent_20 -0.0043      0.158     -0.027      0.978      -0.315       0.306
Cafes              0.0246      0.140      0.175      0.861      -0.250       0.300
Sandwiches        -0.2027      0.095     -2.125      0.034      -0.390      -0.016
Bars               0.2852      0.142      2.002      0.045       0.006       0.564
Beer              -0.1031      0.147     -0.702      0.483      -0.391       0.185
Pizza              0.2492      0.125      1.992      0.046       0.004       0.494
Mexican            0.5890      0.124      4.737      0.000       0.345       0.833
Burgers            0.2201      0.134      1.647      0.099      -0.042       0.482
Nightlife         -0.3201      0.153     -2.091      0.037      -0.620      -0.020
stars             -0.0637      0.112     -0.570      0.569      -0.283       0.155
review_count       0.0026      0.000     10.262      0.000       0.002       0.003
nhours             0.0170      0.001     11.751      0.000       0.014       0.020
------------------------------------------------------------------------------
```

Figure 46: summary table for logistic regression model

From the table summary above we can have important information about the features which contribute the most to well-predict the output. Indeed, for each feature $x_i$ we can see the coefficient $w_i$ associated to it (which is a strong indicator of which features are the most important for the model), their confidence intervals but also the p-values associated to the null-hypothesis that the coefficient $w_i$ is equal to zero. From the table it is easy to see that:

- mean_of_recent_10 is a positive predictor to maintain a restaurant open since the confidence interval for the value of the weights is above 0 for both the extremes.
- Sandwiches have higher risk of closing (the coefficient is negative).
- Bars are slightly negative predictor for the closing of a restaurant(positive coefficient which means that is positive predictor to remain open).
- Pizza is a slightly negative predictor for the closing of a restaurant.
- Mexican is a negative predictor for the closing of a restaurant.
- Nightlife is a slightly positive predictor for the closing of a restaurant.
- review_count is a negative predictor for the closing of a restaurant.
- nhours is a positive predictor for the closing of a restaurants.

As we have seen, the model has an high interpretability. The accuracy for the test set is 0.64 which cannot considered a bad one since the task is quite complex from the data that we have.

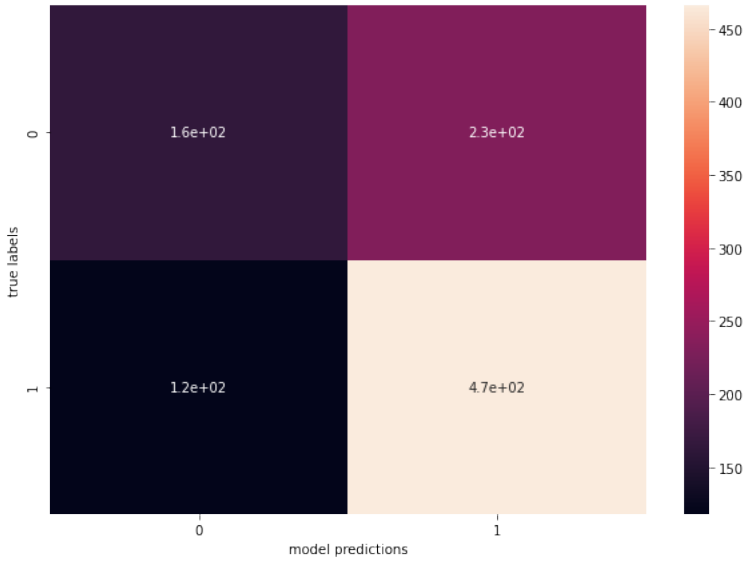



Figure 47: Confusion matrix for logistic regression model.

The confusion matrix for the logistic regression model is displayed below.

### 8.2.2 Support Vector Machine

The Support Vector Machine is another popular classification model. The idea behind the basic version of the algorithm is to find the best separating hyperplane ($y = w^T x$ so we just need to find $w$) where the best means that the margin, defined as the distance between the closest point and the decision boundary, is maximized.

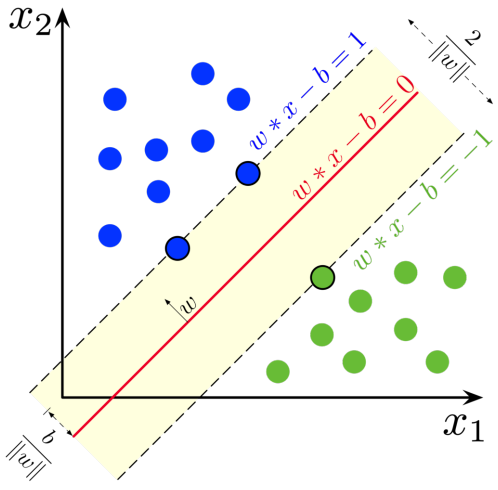



Figure 48: Support Vector Machine.

It can be shown that the margin is equal to $||\frac{1}{w}||$. So, for linearly separable dataset

the minimization problem has the following formulation:

$$\min_{w} ||w|| \text{ such that } y_i x_i^T w \geq 1$$

Indeed the minimization of $||w||$ is actually a maximization of the margin and the constraints simply imply that each point of the dataset is correctly classified and lie outside the margin: if $y_i = 1$ then $x_i^T w \geq 1$ and if $y_i = -1$ then $x_i^T w \leq -1$. However this problem (called **Hard Margin Support Vector Machine** in literature) has no solution if the dataset is not linearly separable. Indeed in this case we cannot find a linear classifier able to separate the two classes without committing any misclassification error. Then the problem has a different formulation called **Soft Margin Support Vector Machine** which is the following:

$$\min_{w,\epsilon} \lambda||w||^2 + \sum_{i=1}^{n} \epsilon_i \text{ subject to } y_i x_i^T w \geq 1 - \epsilon_i$$

We will explain briefly why the problem has this formulation.

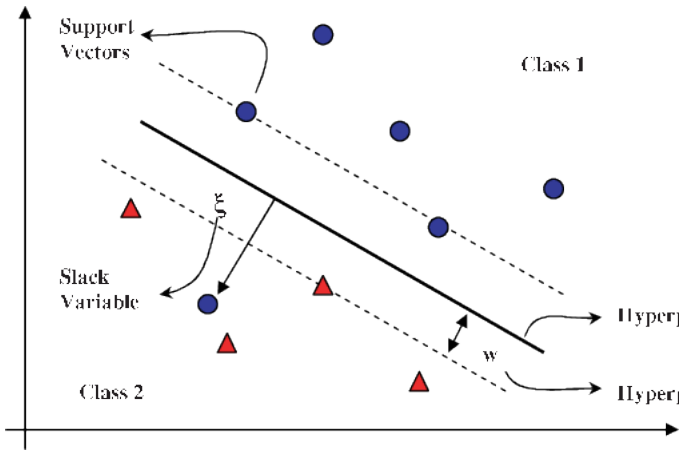



Figure 49: Soft margin Support Vector Machine

In this case we introduce some slacks variables $\epsilon_i$ to be less restrictive in asking that each point is correctly classified. Indeed with the constraint $y_i x_i^T w \geq 1 - \epsilon_i$ we can have that some points actually lie inside the margin but minimizing $\sum_{i=1}^{n} \epsilon_i$ we minimize these errors. In addiction to that, we still maximize the margin minimizing $||w||$. The two optimization problems stated above can be solved efficiently passing at the dual formulation and using quadratic solvers since the duality theorem holds and the dual is a quadratic problem. An evolution of the Support Vector Machine algorithm is **Kernel Support Vector Machine**. In this case we augment the feature space introducing derived features and then we apply the classical Support Vector Machine algorithm to the new features to find the best separating hyperplane. All this process can be simply carried out using a different formulation of the minimization problem which introduces Kernels.

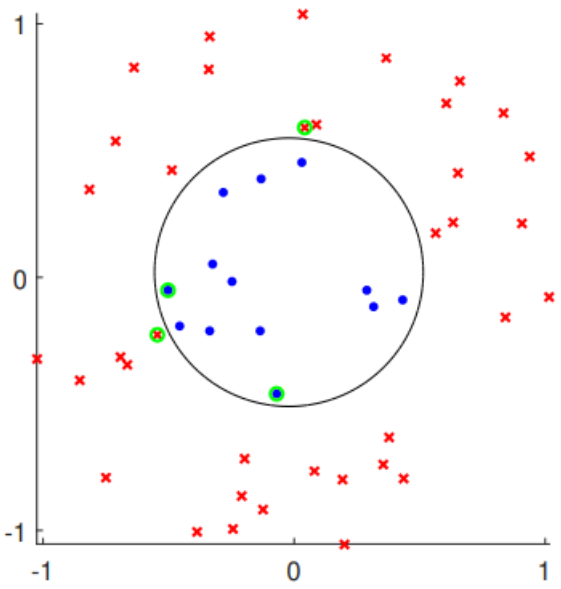



Figure 50: non linear separable dataset

For example, when having a non linear decision boundary as the one represented in the figure above, we could simply add a new feature $z = x^2 + y^2$ (where $x, y$ are the initial features) to transform the dataset in the following which is now separable by a plane.

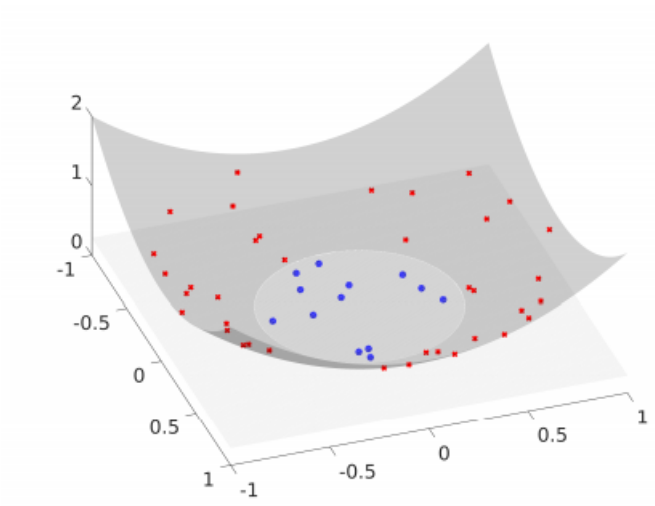



Figure 51: paraboloid after feature expansion

We applied Kernel Support Vector Machine to predict whether a restaurant was closed or not based on the other features. We used **Cross-validation** to select the best hyperparameter for the model such as the regularization coefficient $\lambda$ in the formula above and the type of Kernel used to augment the feature space. We will explain now how cross validation works. There are essentially two ways of tuning hyperparameters:

- The first way is to divide the data initially in training data, validation data and test data. The train set is used to train our model, the hyperparameters are chosen

such that they minimize the loss of the validation set and finally the test set is used to evaluate predictions.

- Another way could be to split the data just in training and test data. We could now divide the train set in $k$ folds, $k-1$ of them will be used as training set and the last one will be used as the validation set. For each different split of the folds, we train on the $k-1$ training fold and we evaluate the performance on the last fold. At the end we average the loss and we select the best hyperparameters such that the average loss for the k-validation-folds is minimized.

After having tuned the hyperparameters of the model we use the model to make the predictions. The accuracy in this case was 0.68 on the test set which shows the superiority of the Support Vector Machine model. Here below is reported the confusion matrix for the predictions.

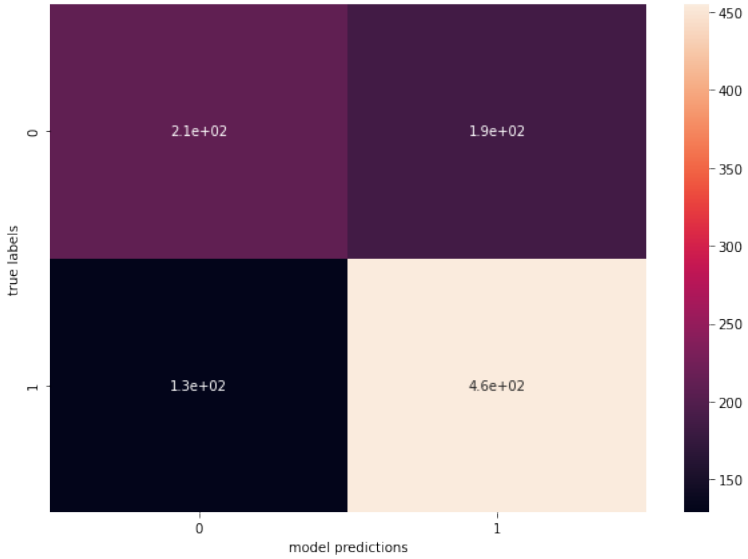



Figure 52: Confusion matrix for support vector machine.

In addition, we will plot the ROC curve for this last model. The receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The plot has on his x axis the false positive rate and on the y label the true positive rate. The perfect classifier should be the horizontal line passing through the point (0,1) while the random classifier is represented by the line $y = x$. Here is how the ROC curve for our classifier looks like.

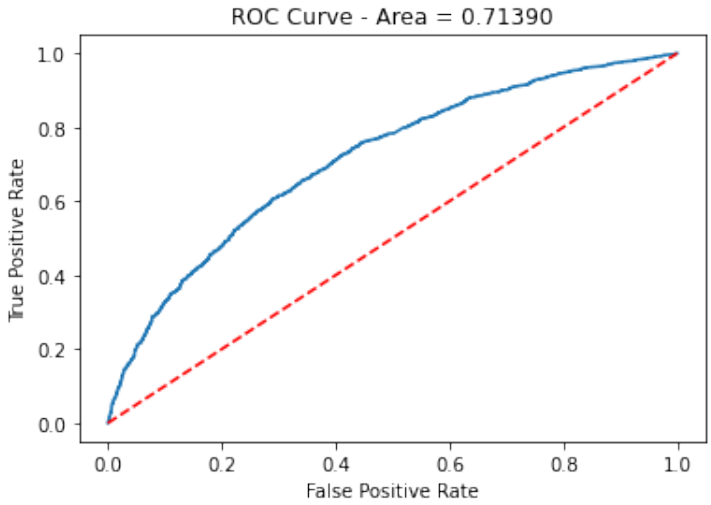



Figure 53: ROC curve for support vector machine.

The area under the curve is above 0.7 so the classifier works sufficiently well with respect to his task.

# 9 Recommender system

In this section our aim will be to construct models with the purpose of suggesting new friends or new restaurants to the users basing on their previous experiences on Yelp app.

## 9.1 Introductory reasoning

We first reason on how we could construct a model to suggest new friends and to suggest new restaurants to the users. To answer our first question we want to find out users that have in common a significant amount of high graded restaurants. In our analysis the correlation between two people is given by the number of restaurants with a grade between four and five. We considered $user\_id$ as nodes that are linked by edges. The nodes $i$ and $j$ are linked by an edge $V(i,j)$ in this way:

$$w_{i,j} = \#\{\text{Restaurants with grades} = 5 \text{ that users } i \text{ and } j \text{ have in common}\}$$
$$V(i,j) = w_{i,j} \Longleftrightarrow w_{i,j} \geq 1$$

Figure 54: Graph nodes and edges of users.

To visualize our idea we decided to first implement a program using pandas data-frame library. First of all we selected a sample of the data-set containing all the restaurants of Vancouver in British Columbia, because friendships are more likely to be between people of the same city and users that have many visited restaurants in common tend to live in the same town. Then we also selected all the reviews graded more or equal to five stars, because we decided to consider as "good reviews" only the highest evaluations. After that we created a new data-frame containing in each row the person's $id$, the $id$ of another user, the amount of liked restaurants the two have in common. At this point we took in consideration only couple of users that have in common at least seven restaurants. We decided that seven could be a reasonable extreme value for the number of common restaurant to define a similar taste between two people, in fact we think that if this number is between one and six it is too low to assert a similarity.

The histogram in *Figure 55* represents the number of potentials couples (edges) we found with our analysis. At the end of this first analysis we counted all the couples remained after all those cleanings, because those are the one that are more likely to be or to become friends. To investigate a little more we also decided to study if the couples we selected with our criteria are truly friends on Yelp App. We are able to do so because of the *Friends* attribute in the Users data-set that states for each $user\_id$ all the $user\_id$ of his friends. So we counted how many of the couples left in the last

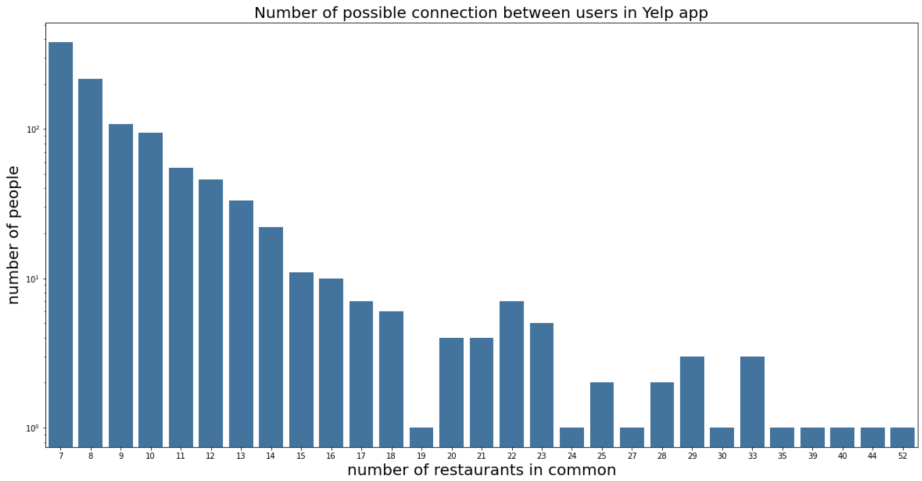



Figure 55: Distribution of the number of common liked restaurants.

data-frame are actually friends and we discover that they represent only the 17% of the entire amount of couples.

After this first part we discover that only a small part of users that liked the same restaurants are actually friends for this reason we think the might be interesting to take advantage of those information to create a recommendation system in order to suggest users both new restaurants and new friends depending on their tastes.

We first built an algorithm to suggest users new restaurants based on other users experiences. Because we think it is not appropriate to use *Friends* attribute in order to make suggestions, but it is better to take recommendations from people that have more similar tastes and similar experiences. For doing that we started from the network of *user_id* we've just created.

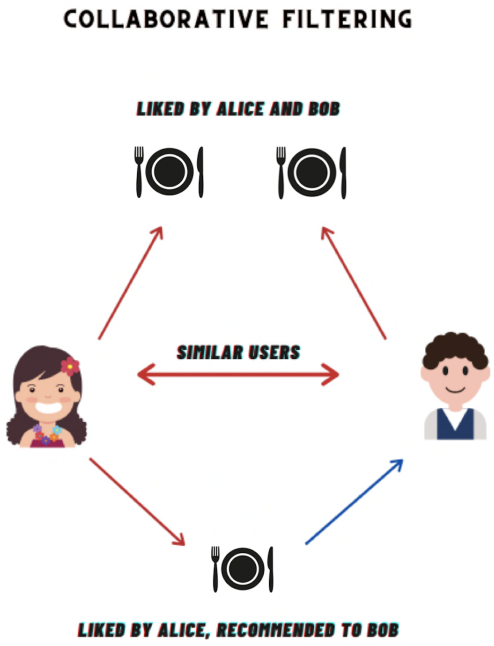



Figure 56: The idea of restaurant suggestions model.

The idea of our algorithm is to take a user and a list with all the users that we selected in the previous task. We then sort them basing on the number of common restaurants (rated 5 stars) with the first user and selected only the most related people. Once we have these users (one or two) we put in a list all the restaurants that these users liked the most, so with 5 stars rating. This list represents the all the possible new restaurants where the first user can go eating. *Figure 56* well express how our algorithm works. Unfortunately these algorithm does not give a method to rate the restaurant that we are suggesting, actually the problem that we've found was that some people gave 5 stars to more than 100 restaurants and consequently the users who were most correlated to this type of people obtained many suggestions, which could be useless for two reasons. The first one is that if a user received too many suggestions it could not watch them at all and the second one is that they are not ranked so the user could not know which restaurant could be the best.

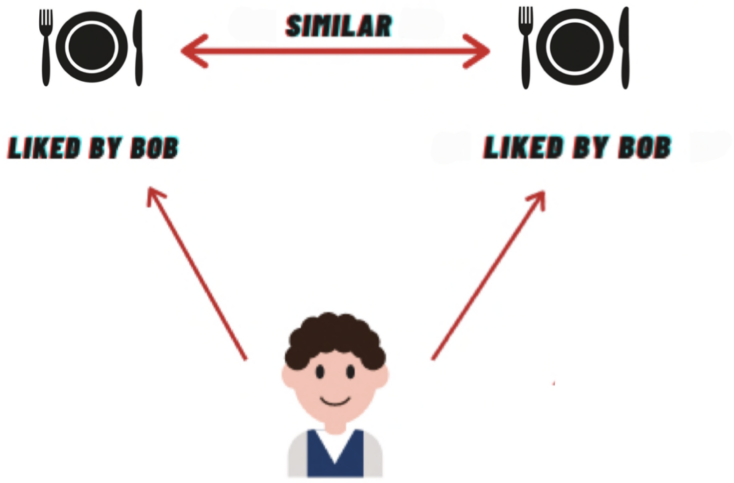



Figure 57: The idea of restaurant competitors model.

In the third part of this analysis we decided also to find possible competitors to the restaurants registered on Yelp. To do so we apply almost the same idea we applied before. This time we considered the restaurants as nodes of a graph and we connected them with edges whose weight is the number of users that liked both the restaurant and consequently gave a high rate. Once obtained all these restaurants we only kept the edges with a number sufficiently high. Once obtained this new graph if two nodes are connected then it means that they could be competitors. The idea of the algorithm is well described in *Figure 57*.

## 9.2 Theoretical model

In this section we present in detail the methodology that we have used to construct our recommender system in practice. There are many different methodologies to build a recommendation system like deep learning, local averaging techniques or Bayesian networks. The model we have used is based in low rank approximation techniques. Our decision is mainly driven by the fact that in this kind of model is quite simple to transform a user-business recommendation into a user-user or a business-business. This adaptability will be very useful for us as we want to do the three:

- Recommend new restaurants to our users
- Tell our businesses who are their competitors
- Suggest new friends based on users taste similarities

The usage of low rank factorization techniques have been proven to be very successful to build recommendation systems. In fact, the famous Netflix Prize for the best collaborative filtering algorithm was awarded to a team of scientists that implemented low rank approximation techniques, besting Netflix own algorithm by more than 10%.

Our goal is to encode the ideas previously exposed into a model. We want the system to assign ratings to hidden user-business pairs based on information about visible user-business pairs. We assume that these dependencies will be somewhat linear. This makes perfect sense: we expect that if two users agreed on the rating for visible data, then they will also agree for the hidden data and the level of agreement will be the same. However we want our similarities to be more complex than simple based on nodes that are at distance 2. To see why, we can think about an example with 3 users in which user 2 has visited some restaurants in common with user 1 and user 3 but the last two users have no restaurants in common. In the case of considering only distance 2 models like the one exposed before we would not be able to say nothing about the similarities of user 1 and user 3 although we have information to do it because we know how are they related to user 2. Therefore it is clear that we have to build a model that considers dependencies beyond simple first order connections. Another motivation for the liner techniques is that the loss of information is zero, in the sense that the similarities have to be transitive. In our model we will put no restriction about how many users we need to explain the hidden ratings of a particular user. However we do not want all the users to contribute equally, because in some of them the relationship may be very strong while in others could be almost irrellevant. Finally we want all this aspects to be captured automatically, and that is what low rank approximation techniques do indeed.

The setting is the following. We have $m$ users, $n$ businesses and some matches (ratings) between them. Now we want to estimate the rating that any user would give to any business. In order to do so we build the following matrix

$$R := \begin{cases} r_{ij} & \text{rating of user (i) to business (j)} \\ 0 & \text{when there is no rating available} \end{cases}$$

We assume that there are multiple similarities between users so that we may be able to approximate $R$ with a low rank version of it $\tilde{R}$.

$$rank(\tilde{R}) = k \quad << \quad rank(R)$$

Then, we can factorize $\tilde{R}$ as:

$$\tilde{R} = PQ^T \qquad P \in \mathbb{R}^{m \times k} \quad Q \in \mathbb{R}^{n \times k}$$

and the problem becomes the following:

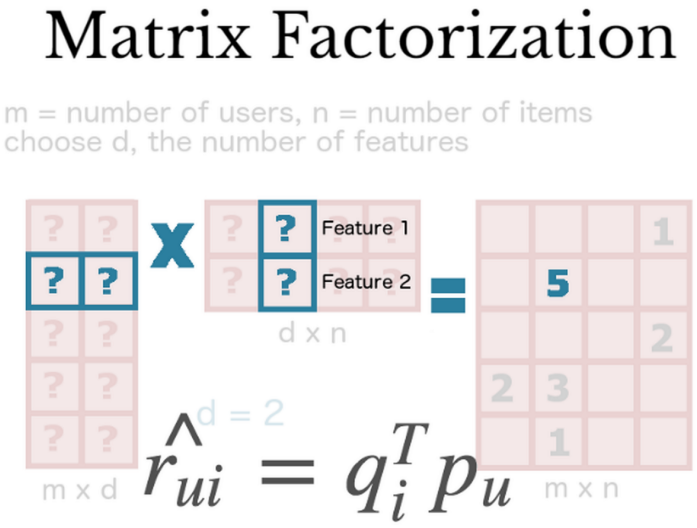
$$min \left[ \left\| R - PQ^T \right\|^2 + \lambda(\|Q\|| + \|P\|) \right]$$

restricted to $P \in \mathbb{R}^{m \times k} \quad Q \in \mathbb{R}^{n \times k}$ where for $\|\cdot\|$ we consider the Frobenius norm. In order to find out the regularization parameter $\lambda$ and the rank k we performed cross validation. It could seem strange that we are doing regularization and assuming low rank at the same time, but the motivation is clear. On the one hand the regularization with the Frobenius norm leads to a matrix solution that is not too sparse with many zeros and it is what we need to recover as many ratings as possible. On the other hand the rank assumption follows from the fact that we are trying to find multiple similarites (linear dependencies).

Finally the estimated predictions will look like this:

$$\hat{r}_{ij} = \sum_{i=1}^{k} q_{ik}^T p_{kj}$$

so that each rating is the contribution of k latent factors. The model can be summarized by the following picture.



## 9.3 Practical results

In this part we apply the matrix factorization model exposed in the previous section to build a complete recommendation system that performs 3 tasks. First of all we want to mention some aspects of the model fitting. We removed 25% of the visible ratings and then minimized $\left\|R - PQ^T\right\|^2 + \lambda(\|Q\|| + \|P\|)$ with the remaining ratings. We used cross validation with 3 folds to find the optimal rank and regularization hyperparameter. Finally we evaluated the model in the ratings removed when fitting. We got the following performance in the test set:

$$\boxed{MSE(\tilde{R}) = 0.9072}$$

It is important to note that this is not the mean square error between $R$ and $\tilde{R}$ but only between previously removed ratings $r_{ij}$ and the corresponding estimates $\tilde{r}_{ij}$

### 9.3.1 Restaurant recommendation

By factorizing the Rating matrix with a low rank approximation we recover the estimated rating for every pair of *user-restaurant*. Then once fixed a user we want to give him/her personalized recommendations. Since we have recovered a low rank version of the rating matrix, we can simply fix the row corresponding to a user and sort it. Hence, the recommendation of our system will be the restaurants high highest estimated rating among those that the users have not previously tried

$$\boxed{\textbf{User: } RtGqdDBvvBCjcu5dUqwfzA}$$

**Results for selected user**

| | business_id | rating |
|---|---|---|
| 1 | dQjv55_rr1mdF18H6baRxg | 4.208295 |
| 2 | x9RA_NPjcvQ8-EU28ppBqA | 4.196035 |
| 3 | -rEA553gskIEHZteI5RExQ | 4.188387 |
| 4 | aGEaKx1jbVyQxeMumKYMHQ | 4.091798 |
| 5 | p1FElOKJUkFOpXx4dvTwEg | 4.069034 |

Table 9.1: Restaurant recommendation for the selected user based on estimated ratings.

### 9.3.2 Friend recommendation

Once we have recovered an approximation of the rating matrix it is very easy to adapt it for different recommendations. In particular to suggest new friends to our users what we did is to look at the correlations of the rows in the rating matrix. These correlations tell us how similar is the taste of two users. We do not expect these correlations to be an absolute approximation to the real similarity between users. In fact we know that our users will probably be more correlated in our estimations than in reality, due to the fact that our model is a low rank approximation. Therefore, by assumption there should be many high correlated users. However, although these correlations are not an absolute measure of taste similarity between users they can be very good in relative terms. In other words if our model says that user 1 is more similar in terms of taste to user 2 than

to user 3 it is quite likely that it will be also true in reality as far as the hidden ratings are not too different to the visible ones.

**User:** $ei7wfryXlvZ6OM9NK27cPQ$

| | User_id | Correlations |
|---|---|---|
| 1 | MIPPioT-rUJQwTu3_LjGpA | 0.939570 |
| 2 | GERRIWMslcP4-O6mLErPSQ | 0.939338 |
| 3 | z3FEHWGWSR5Z9ZVARQroHw | 0.936907 |
| 4 | ydbq7bGaMO_9M1GRO7VnAg | 0.936231 |
| 5 | yWzgaaze8dn-lss14MiZJw | 0.934996 |

Table 9.2: Friends recommendation for the selected user based on correlation between the ratings vectors.

### 9.3.3 Identifying competitors

Finally we want to tell each restaurant which are his potential competitors. In order to do so, we follow a similar approach as in the previous section, looking at the correlations between the columns of the rating matrix. In this case we expect these to be much lower than in the friends recommendation because the number of restaurants is very small compared to the number of users. Hence there should be less linear dependencies between the columns in the approximated rating matrix.

**Business:** $niUrhHoR9leK0lr5moyySQ$

| | Business_id | Correlations |
|---|---|---|
| 1 | j0qVvKZEBdlkQHqyYE1iTw | 0.793876 |
| 2 | vLbjVhNd73054UVdvOeg7Q | 0.792549 |
| 3 | aGEaKx1jbVyQxeMumKYMHQ | 0.784495 |
| 4 | y1BLVJUWINOSMLeDaXqXBQ | 0.783197 |
| 5 | EXfEzv1sg3smvXJo36hSmQ | 0.782686 |

Table 9.3: Possible competitors for a selected restaurant based on the users that should like both of them.

# 10 Topic analysis

The goal of this section is to understand what users want to express when they write a review. We want to find a set of topics that can describe the different thoughts in the yelp reviews. This is motivated by:

- **Summarize the reviews:** If we can transform the reviews into the set of topics they are talking about we will be able to describe several paragraphs with just a couple of words. This can have many applications in huge platforms like yelp. On the one hand it can save a lot of time to users when they surf the reviews as they can focus in the topic that they are interested about. On the other hand it can help businesses to extract in a clear and concise way people's thoughts about their restaurant. Is the service good? What about the drinks?, etc
- **Feature extraction:** The output of our model will be a vector for each review that will tell us the proportion of each topic in the text. This can be used together with the ratings to better understand the sentiment of each topic in a particular restaurant. This would allow to identify which topics are positive or negative for each restaurant.

It is straightforward to know what a user is talking about when expressing their opinion about a particular business. For example, in the following sentence:

*"Great place to hang out after work: the prices are decent, and the ambience is fun"*

We could say that there are two topics in this sentence: Price  ambiance. It is effortless to do this manually, but at first sight, there is no clear way to do it automatically for a vast amount of data. A naive approach could be to create by hand several topics like Price or ambiance and assign the review to each topic depending if a keyword appears or not. However, this wouldn't work because different people would express the same sentence with other words. For example:

*"I recommend it to haunt there: it is pretty cheap, and the atmosphere is cool."*

Of course, we can solve this by adding several words to each topic. However, this kind of manual approach is suboptimal. First of all, the fact that we select the topics and their words can lead to ignoring some hidden topics or relevant terms in a particular category. Moreover, not all the words in the different topics are equally important. Hence, we do not want to classify the reviews by simply doing a frequency counting of the words in different categories but also taking into account each word's importance in the corresponding category. In order to consider all these aspects in our model, we decided to use Latent Dirichlet Allocation (LDA).

Latent Dirichlet Allocation is a generative probabilistic model developed in population genetics by K.K Pritchard, M.Stephens, and P.Donnelly. It was later applied to Text mining by David Blei, Andrew Ng, and Michael I. Jordan. Here we present the particular version that we have used, but there are multiple variations. The main idea is that some hidden topics can describe the reviews and that each of these topics is characterized with some words. However, we have to neither set the topics nor the words in them manually. The model assumes a Bayesian prior and then computes a posterior from the available data. We interpret each review as a probability distribution of different topics and the topics as distributions of words. Formally we define the model in the following way:

- A **word** is the basic unit of the discrete data, defined to be an item from a vocabulary indexed by $\{1, ..., V\}$. We represent the words as vectors with one hot encoding (e.g one component equal to 1 and the rest equal to zero
- A **review** is a sequence of $N$ words denoted by $\mathbf{w} = (w_1, w_2, ..., w_N)$.
- Our input **data** is a collection of $M$ reviews denoted by $\mathcal{D} = \{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_M}\}$.

Now we want to represent the reviews as random mixtures over latent variables, where each latent has also a distribution over words. In order to infer those distributions we make the following assumptions:

1. Number of words in a review: $N \sim Poisson(\lambda)$
2. Distribution of topics in a review: $\theta_i \sim Dir(\alpha)$ with fixed dimension
3. For each of the N words in a review:
   (a) The topic distribution in the word is $z_k \sim Multinomial(\theta)$
   (b) The probability of the word appearing in a topic: $p(w|z_k) \sim Multinomial(\beta)$

For now we consider that $\alpha$ and $\beta$ are fixed. Then the prior joint distribution of the topic mixture, the set of topics and words is given by:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{10.1}$$

and marginalizing over $\mathbf{z}, \theta$ we can obtain the distribution of the data:

$$p(\mathcal{D}|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left( \prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta) p(w_n|z_{dn}, \beta) \right) d\theta_d \tag{10.2}$$

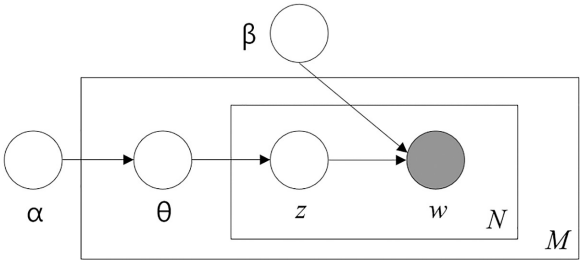



Figure 58: Graphical model representation of LDA.

The LDA model can be interpreted as a probabilistic graphical model In the figure below it becomes clear the hierarchical nature of the model. The parameters $\alpha$ and $\beta$ are in the highest level, as they are common in all the data. Then the variables $\theta$ are review-level variables and finally $z$ and $w$ are word-level variables.

Now we would like to estimate the parameters $\alpha, \beta$ based on the observed reviews. A common approach is to compute the posterior distribution

$$p(\theta, z|\mathcal{D}, \alpha, \beta)$$

using (11.1) up to an intractable normalizing constant. Then we can apply Variational inference or MCMC to approximate the posterior. However in the package that we used (Gensim) this is done with a mix of Variational inference and the EM algorithm. Although we have not implemented it we summarize what the function we used is doing.

The goal is to maximize

$$p(\mathcal{D}|\alpha, \beta)$$

which can be done with the EM algorithm. However since it is intractable, the idea is to instead obtain a lower bound of the log likelihood with the variational distribution and then maximize it.

**Expectation**

1. $(\gamma^*, \psi^*) = \underset{\gamma,\psi}{argmin} KL\left(q(\theta, z|\gamma, \psi)||p(\theta, z|w, \alpha, \beta)\right)$
   where $q(\theta, z|\gamma, \psi) = q(\theta|\gamma)\prod_{n=1}^{N} q(z|\theta)$
2. $\log p(w|\alpha, \beta) = L(\gamma, \psi; \alpha, \beta) + KL\left(q(\theta, z|\gamma, \psi)||p(\theta, z|w, \alpha, \beta)\right)$

$$L(\gamma, \psi; \alpha, \beta) = \mathbb{E}[\log p(\theta, z, w|\alpha, \beta)] - \mathbb{E}[\log q(\theta, z)]$$

**Maximization**

3. Maximize $L(\gamma^*, \psi^*; \alpha, \beta)$ with respect to the true parameters $\alpha, \beta$

## 10.1 Preprocessing

Before applying Latent Dirichlet allocation to our dataset, we had to perform some data cleaning. First of all, the stopwords, like prepositions, pronouns, articles, etc. For doing that we employed the **nltk.corpus** package that provides functions that can be used to read corpus files in a variety of formats. In our case we adopted the **stopwords.words('english')** function that create a list of all the high-frequency words as connectors ('the','and', 'also'...), verbs ('do','did','was'...), pronouns ('it','I','me'...) and many others that don't give relevant information about the sentence meaning and for this reason we want to filter them out of a document before further processing.

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i | you've | himself | they | that | been | a | while | through | in | here | few | own | just | re | doesn | ma | shouldn't |
| me | you'll | she | them | that'll | being | an | of | during | out | there | more | same | don | ve | doesn't | mightn | wasn |
| my | you'd | she's | their | these | have | the | at | before | on | when | most | so | don't | y | hadn | mightn't | wasn't |
| myself | your | her | theirs | those | has | and | by | after | off | where | other | than | should | ain | hadn't | mustn | weren |
| we | yours | hers | themselves | am | had | but | for | above | over | why | some | too | should've | aren | hasn | mustn't | weren't |
| our | yourself | herself | what | is | having | if | with | below | under | how | such | very | now | aren't | hasn't | needn | won |
| ours | yourselves | it | which | are | do | or | about | to | again | all | no | s | d | couldn | haven | needn't | won't |
| ourselves | he | it's | who | was | does | because | against | from | further | any | nor | t | ll | couldn't | haven't | shan | wouldn |
| you | him | its | whom | were | did | as | between | up | then | both | not | can | m | didn | isn | shan't | wouldn't |

Figure 59: Stopwords.

So we first select a small sample of 10000 rows from the data-set *Users* containing all the reviews text we need. The next step is to remove from those ten thousands texts all the *stopwords* that are useless for the topic analysis. For each text we used the **string.split()** function that split the string into a list where each word is a list item. From this list we selected all the words that, in their lowercase form, are not in the *stopwords* list. The **".join()** is a string method that returns a string by joining all the elements of a list, separated by a string separator. Trough this function we unify all those selected words in a single string. At this point that we have a string without all the *stopwords* we need to remove the punctuation and for doing that we employed the **sentence.translate(str.maketrans( '', '', string.punctuation)** function. After we created the new cleaned string for each review text we piled them in a single list that will then become a new column, *new_text*, of the data-frame.

| | |
|---|---|
| **Old text** | The greens and beans were unique and great! They were a tad spicy. The fries were awesome! The green tea was refreshing, too. I didn't care too much for the cole slaw or the smoked wings (very small) with the Kitchen sauce, but the hubs loved them. Service was nice and fairly quick considering the high volume of customers they received on Saturday night. We will be back. |
| **New text** | greens beans unique great tad spicy fries awesome green tea refreshing too care much cole slaw smoked wings very small kitchen sauce hubs loved them service nice fairly quick considering high volume customers received saturday night back |

Figure 60: Example of the cleaned review text.

## 10.2 LDA application

At this point we have a pre-processed data-set and we only need to apply the LDA method we explained before. For doing that we applied *gensim* library that is designed to process raw, unstructured digital texts using unsupervised machine learning algorithms. But means of this library we're able to estimate the Latent Dirichlet Allocation model parameters based on a training corpus, represented by the tokenized reviews *new_text*, and for a selected number of topics. We decided to take in consideration a number of ten topics. The topics we obtained employing this method are explained in the table below. As we can see for each topic we find a list of words and how much they are relevant for this specific topic. In this case the words seem to be quite general and there's not a broken division between the different topics.

After those results, for each review text we calculate the topic distribution through the function *get_document_topics*(*text*) and put these values in ten new different columns each of which represents one specific topic. In *Figure* 62 it is shown how worked the LDA model on the example of *Figure* 60 and the resulted new columns.

The images in *Figure* 63 represents the ten different topics and how are distributed the words in topic number ten. Particularly, in the left side of the graphic it is shown how interact all the ten topics. Many of them are intersected with other topics to the point that they seem overlapping. Despite that there are topics number six,eight,nine and ten that have no intersections.The size of the circumferences diameter represent the marginal topic distribution. In the right side we have the distribution of the words composing topic number ten. The red bars for each word represent the estimated term frequency within the selected topic, while the blue bar is the overall term frequency.

| N° | **Topics** |
|---|---|
| 1 | 0.013*"place" + 0.011*"food" + 0.007*"good" + 0.007*"great" + '0.006*"service" + 0.006*"time" + 0.005*"like" + 0.005*"would" + 0.005*"get" + 0.005*"also" |
| 2 | 0.012*"food" + 0.009*"place" + 0.009*"great" + 0.009*"good" + 0.008*"get" + 0.007*"service" + 0.007*"time" + 0.006*"one" + 0.006*"go" + 0.005*"would" |
| 3 | 0.008*"good" + 0.007*"one" + 0.006*"like" + 0.006*"time" + 0.005*"food" + 0.005*"service" + 0.005*"great" + 0.005*"would" + 0.005*"get" + 0.004*"also" |
| 4 | 0.008*"good" + 0.007*"go" + 0.006*"great" + 0.006*"like" + 0.006*"food" + 0.006*"get" + 0.005*"well" + 0.005*"one" + 0.005*"time" + 0.005*"really" |
| 5 | 0.008*"great" + 0.007*"food" + 0.006*"time" + 0.006*"place" + 0.006*"would" + 0.005*"us" + 0.005*"like" + 0.005*"one" + 0.005*"even" + 0.005*"good" |
| 6 | 0.008*"great" + 0.008*"good" + 0.008*"back" + 0.007*"food" + 0.007*"one" + 0.006*"service" + 0.006*"time" + 0.006*"place" + 0.006*"im" + 0.005*"would" |
| 7 | 0.008*"like" + 0.006*"great" + 0.006*"back" + 0.006*"one" + 0.005*"time" + 0.005*"would" + 0.005*"get" + 0.005*"food" + 0.005*"place" + 0.004*"it" |
| 8 | 0.009*"place" + 0.007*"great" + 0.006*"good" + 0.006*"food" + 0.006*"like" + 0.005*"go" + 0.005*"service" + 0.005*"time" + 0.005*"one" + 0.005*"got" |
| 9 | 0.010*"place" + 0.009*"good" + 0.008*"food" + 0.007*"great" + 0.007*"one" + 0.006*"like" + 0.006*"get" + 0.006*"service" + 0.006*"time" + 0.005*"back" |
| 10 | 0.016*"good" + 0.011*"food" + 0.009*"like" + 0.007*"really" + 0.006*"great" + 0.006*"place" + 0.006*"would" + 0.005*"back" + 0.005*"get" + 0.005*"service" |

Figure 61: Topics result.

| new_text | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| greens beans unique great tad spicy fries awes... | 0.254551 | 0.088733 | 0.406717 | 0.982307 | 0.949959 | 0.952568 | 0.98197 | 0.967767 | 0.967816 | 0.867801 |

Figure 62: Words removed in each sentences.

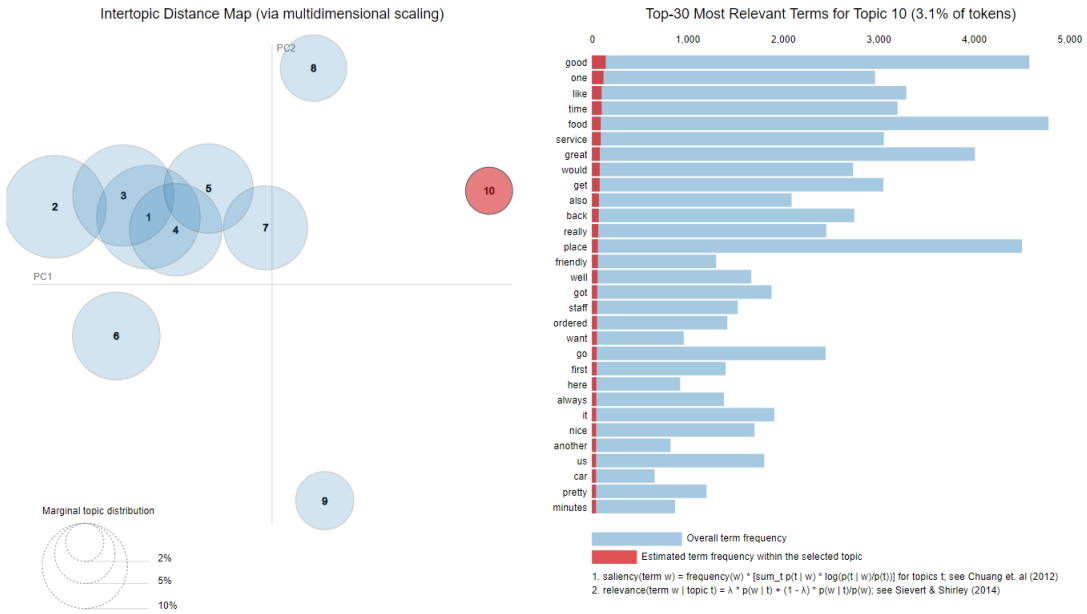



Figure 63: Analysis on topic number 10.

# 11 Text analysis to predict ratings

## 11.1 Preprocessing

In the previous section we have applied an unsupervised machine learning model (LDA) to cluster reviews by topic. Can we now build a machine learning model which predicts the score of the reviews using the features characterizing the topic of each review ? To this end, we consider the dataset to which we have previously applied LDA method. Here below we show how a sample of the dataset look like.

| | stars | text | new_text | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **372339** | 1 | Blue Host has been on a downward slide for ove... | blue host downward slide year theyve obviously... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **144885** | 2 | I was there a few years back. I appreciated th... | years back appreciated video showed us explain... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **107653** | 2 | Good place for nachos and drinks, that's about... | good place nachos drinks thats it | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **379595** | 5 | First - great service. My wife is pregnant an... | first great service wife pregnant server poin... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **245044** | 4 | Friendly staff and is seemingly packed with us... | friendly staff seemingly packed usuals food gr... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **355071** | 5 | Halinas staff is always professional and effic... | halinas staff always professional efficient iv... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **296070** | 4 | This place may not impress people from the out... | place may impress people outside cheesesteaks ... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **379786** | 4 | Great place for a night out with friends. Good... | great place night friends good food good servi... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **436895** | 3 | Delish but king crabs had Lil meat and not wor... | delish king crabs lil meat worth 50 dollars 2 ... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |
| **383777** | 5 | This is the best Thai Food on the North end. T... | best thai food north end place clean food wond... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 |

Figure 64: Dataset after LDA.

We notice that clustering the reviews by topic allows us to have more information about the score of the reviews, in particular, we expect reviews in the same cluster having a similar score since the words used in the reviews should be similar in the same cluster. Therefore, we will use them as the features of a supervised machine learning model whose aim is to predict the score of the reviews. To be sure that the model has enough features to learn from the data we will also apply sentiment analysis to the text of our reviews since the it can help us decipher the mood and emotions of general public and gather insightful information regarding the context. To this end we have used TextBlob: a python library for Natural Language Processing (NLP) which actively used Natural Language ToolKit (NLTK) to achieve its tasks. For lexicon-based approaches, a sentiment is defined by its semantic orientation and the intensity of each word in the sentence. This requires a pre-defined dictionary classifying negative and positive words. Generally, a text message will be represented by bag of words. After assigning individual scores to all the words, final sentiment is calculated by some pooling operation like taking an average of all the sentiments. TextBlob returns polarity and subjectivity of a sentence. Polarity lies between [-1,1], -1 defines a negative sentiment and 1 defines a positive sentiment (negation words reverse the polarity). It also returns subjectivity which lies between [0,1]. Subjectivity quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains personal opinion rather than factual information. We used Textblob to retrieve the polarity of each review and we added this feature to the dataframe. Here below we show a sample of the dataframe after having applied Textblob sentiment analysis.

| | stars | text | new_text | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | polarity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **372339** | 1 | Blue Host has been on a downward slide for ove... | blue host downward slide year theyve obviously... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | -0.200000 |
| **144885** | 2 | I was there a few years back. I appreciated th... | years back appreciated video showed us explain... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | 0.022511 |
| **107653** | 2 | Good place for nachos and drinks, that's about... | good place nachos drinks thats it | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | 0.700000 |
| **379595** | 5 | First - great service. My wife is pregnant an... | first great service wife pregnant server poin... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | 0.506250 |
| **245044** | 4 | Friendly staff and is seemingly packed with us... | friendly staff seemingly packed usuals food gr... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | 0.587500 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **355071** | 5 | Halinas staff is always professional and effic... | halinas staff always professional efficient iv... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | 0.375000 |
| **296070** | 4 | This place may not impress people from the out... | place may impress people outside cheesesteaks ... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | 0.312500 |
| **379786** | 4 | Great place for a night out with friends. Good... | great place night friends good food good servi... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | 0.546875 |
| **436895** | 3 | Delish but king crabs had Lil meat and not wor... | delish king crabs lil meat worth 50 dollars 2 ... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | -0.075000 |
| **383777** | 5 | This is the best Thai Food on the North end. T... | best thai food north end place clean food wond... | 0.53235 | 0.532248 | 0.930688 | 0.954971 | 0.143865 | 0.438007 | 0.394262 | 0.440094 | 0.381254 | 0.082891 | 0.562037 |

Figure 65: Dataset after sentiment analysis.

We can now use the polarity of each reviews and the topic features to build a supervised machine learning model to predict the review scores.

## 11.2 Methods

Once we transformed the reviews into numerical variables with topic and sentiment analysis we could easily apply any regression technique to try to learn the ratings from the associated reviews. We decided to use Random Forest driven by their indifference to the structure of the inputs and their training speed. These factors were characteristic in our problem. On the one hand we have no prior intuition about which can be the structure of a function that maps reviews to ratings so that it could be unreasonable to select a model that restricts the hypothesis space to a particular class of functions. On the other hand we had many reviews so that it was crucial to use a model that could be trained without taking too much time. In this part of the project we used the python library Scikit-Learn that already implements Random forest. Now we will explain briefly how exactly this model is implemented in Scikit-Learn.

The most basic entities of the Random Forest are decision trees. Decision trees are trained as follows: given a set of input variables $(a_1, ..., a_m)$ we find the optimal pair $(a_k, t_k)$ such that when we split the data in two groups according to $a_k \leq t_k$ it leads to the minimum loss. In our case the loss is given by the mean square error where the predictor is the average of the points that lie in the same subdivision as the input given. Once the data has been split in two groups, it continues recursively splitting the subsets using the same logic. The algorithm stops when the tree reaches the maximum depth which is set as a hyperparameter. This is very fast, the training complexity is $\mathcal{O}(m \times n \log(n))$. It is important to point our that at each step we are only checking the loss for the next level and not if the selected variable will lead to the optimal subdivision several levels down. In that case the algorithm would be very expensive $\mathcal{O}(\exp(n))$.

The idea behind Random Forest is to aggregate many trees in order to achieve as high accuracy as the best decision tree but with much less variance. But how can we do this efficiently? The answer is bootstrap aggregation. We create B bootstrap training sets and then for each of them train a decision tree $f_b$. Then the final predictor is

$f = \frac{1}{B}\sum_{i=1}^{B} f_b$. Another aspect of random forest is that to achieve faster training speed and decrease variance we only train the individual decision trees weakly. The weakly training consists in the following variation: at each step of the decision tree algorithm we only consider the optimal division among a subset of k features out of the m features. The subset of features used are sampled at random at each iteration. Therefore each split depends on random variables which leads to a lower correlation between the different predictors.

These are the hyperparameters that we used for training:

- **Max-depth:** it is the number of subdivisions that we create in each decision tree.
- **Max-features:** the number of features k that are sampled at each iteration of the decision trees to select the optimal split
- **Min.samples.leaf:** We won't split a subset if the generated leafs do not have at least this number of points.
- **Min.samples.split:** it is very similar to the previous parameter. In this case it is the minimum number of samples required to split an internal node.
- **N.estimators:** it is the number of bootstraped datasets used to build the final predictor.

We found the optimal hyperparameters via randomized Grid search which performs better than standard Grid Search as we have read in the paper in the bibliography.

# 12 Conclusion

In this we analyzed the big dataset cantaining all the information about yelp app including the reviews. First of all we wanted to get to know better the app. As a result we decided to analyze how it is used across different locations and the growth that it has experienced since it was released in 2004. We noticed that they users are mostly concentrated in some cities of several states in the US. Also, we became aware that the app is becoming more popular day after day with the exception of the Covid times. Then we decided to deep diver into how the restaurants' ratings has evolved in time. We focused on restaurants because they represent the majority of businesses present in the yelp app. We found that the ratings were homogeneous in time across different years. Then we analyzed the monthly variation to detect possible seasonality trends but we didn't find statistically significant differences between ratings for different months although we observed slightly lower ratings in November and December.

After our first analysis we became aware of how powerful the yelp app can be to easily select a restaurant that matches our preferences based on other users' ratings. However we also understood that this can be dangerous because small groups can introduce biases that are then be implicitly supported by the majority that chooses the restaurants based on previous ratings. Motivated by this fact we decided to analyze Mexican restaurants to determine if they are negatively affected by an ethnic bias as they tend to be discriminated in the US but we found that it is not the case in yelp reviews. Once we discovered that there were no ethnic biases in yelp reviews we decided to analyze which attributes of the restaurants are relevant to have high ratings. We discovered that price does not affect the ratings except in very expensive restaurants. We also found that the most positive highly correlated variables were: street parking, classy environment and trendy. Among the negatively correlated variables the one with the highest absolute correlation was if the restaurant serves alcohol. Finally, to conclude the analysis of the restaurants we decided to quantify how likely is for a restaurant to close in the near future. We revealed that the most important features to remain open were the average rating of the last 10 reviews, if the restaurant was Mexican and if it was a bar. We also found that the businesses related to nightlife are the most likely to close.

In the last part of our project we decided to build two models that could improve the yelp app. In the first place we implemented a recommendation system to achieve 3 tasks. On the first hand it recommends restaurants to users based on previous preferences so that they could reach a higher degree of satisfaction. On the other hand it suggests friends to strengthen the links between different app users. Finally it also helps restaurants to identify competitors so that they can improve their service. The second model that we build was driven by the fact that the app has been growing a lot and now contains a vast amount of reviews. As a result, we wanted to help people understand what people are saying in their reviews without reading all of them. The model we implemented can find a set of topics that summarize all the reviews. Then, for each review it is able to tell which is the weight of each topic in that review. Therefore we can easily transform each review into a numerical vector that summarizes it. We leveraged this aspect to predict the ratings of the reviews together with sentiment analysis.

## References

[1] Pearson Wikipedia.

[2] Rank based correlation coefficient Wikipedia.

[3] Mutual.

[4] Random search.