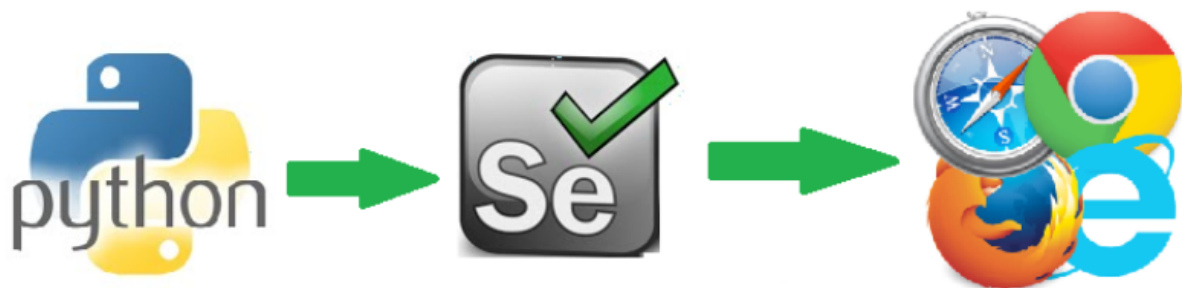


Analysis of TripAdvisor Data with the help of Selenium and Python

Hands-on tutorial on Web Scraping with Selenium, preprocess and analyze textual data to unveil new knowledge



This Python Selenium tutorial explains the process of installing and setting up Selenium, scraping textual data from TripAdvisor and analyzing the data in order to produce answers to some questions regarding basic visualizations, sentiment analysis and user profiling.

More specifically in this tutorial, we shall discuss :

- A. Installing and setting up Selenium
- B. Presentation of the TripAdvisor dataset requirements
- C. TripAdvisor dataset web scraping process
- D. Data preprocessing for textual TripAdvisor data (how to clean your data)
- E. Your final DataFrame format
- F. Answers to the questions of Section “Data Analysis” in Python
- G. Insightful conclusions

Install and setup Selenium for Python

By using Selenium with Python you can easily create automation tools that:

- Scrape data from the web
- Test sites or pages as per multiple predefined test-cases
- Get any repetitive online task done at scale

In all our scripts we would be using Selenium Webdriver. Basically what Selenium Webdriver does is control the browser on behalf of a user. The scripts that we write tell Selenium Webdriver what it needs to do on our behalf in order to scrape the information we want from the website.

The installation process of Selenium on Python is extremely straightforward.

If you have [pip](#) on your system, you can simply install or upgrade the Python bindings:







```
pip install -U selenium
```

Alternately, you can download the source distribution from [PyPI](#) (e.g. selenium-4.1.3.tar.gz), unarchive it, and run:

```
python setup.py install
```

As we mentioned previously, Selenium requires a driver to interact with the chosen browser. In this tutorial we would be using Chrome as the browser of our choice, which requires [ChromeDriver](#).

Index of /101.0.4951.15/

	Name	Last modified	Size	ETag
	Parent Directory		-	
	chromedriver_linux64.zip	2022-04-01 07:53:29	6.28MB	26b17ddda1bc1175fa3d3e78506a562b
	chromedriver_mac64.zip	2022-04-01 07:53:32	7.96MB	0c0e2932391fa49521b488e0ce4dfe1e
	chromedriver_mac64_m1.zip	2022-04-01 07:53:34	7.19MB	e18cf3711957ca4210802a065e334d67
	chromedriver_win32.zip	2022-04-01 07:53:37	6.05MB	fe5b321d4b97907c89dc164c6c99785c
	notes.txt	2022-04-01 07:53:42	0.00MB	446e4ff3aba3adbceda721a038451202

ChromeDriver comes with a variety of zip files based on the Operation System that you use. You can choose and download the appropriate zip file.

Next, save the zip file's path because we would need to specify in our script the specific path that the driver can be found.

```
PATH = "chromedriver.exe"
```

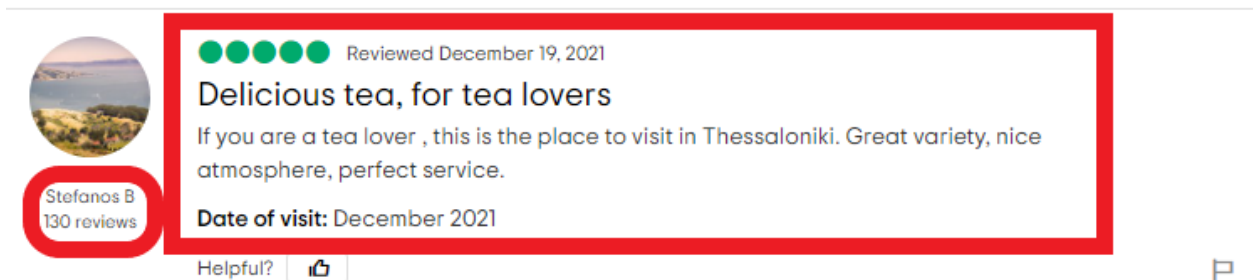
After we have successfully installed and set up the prerequisites, we are ready to import Selenium to our script and access a website using Chrome!

Presentation of the TripAdvisor dataset requirements

Tripadvisor, Inc. is an American online travel company that offers online hotel reservations, bookings, and reviews for transportation, lodging, travel experiences, and restaurants. For this tutorial -for educational purposes only- we are going to scrape TripAdvisor data and most specifically user's reviews for [Cafes & Bars in Thessaloniki](#).

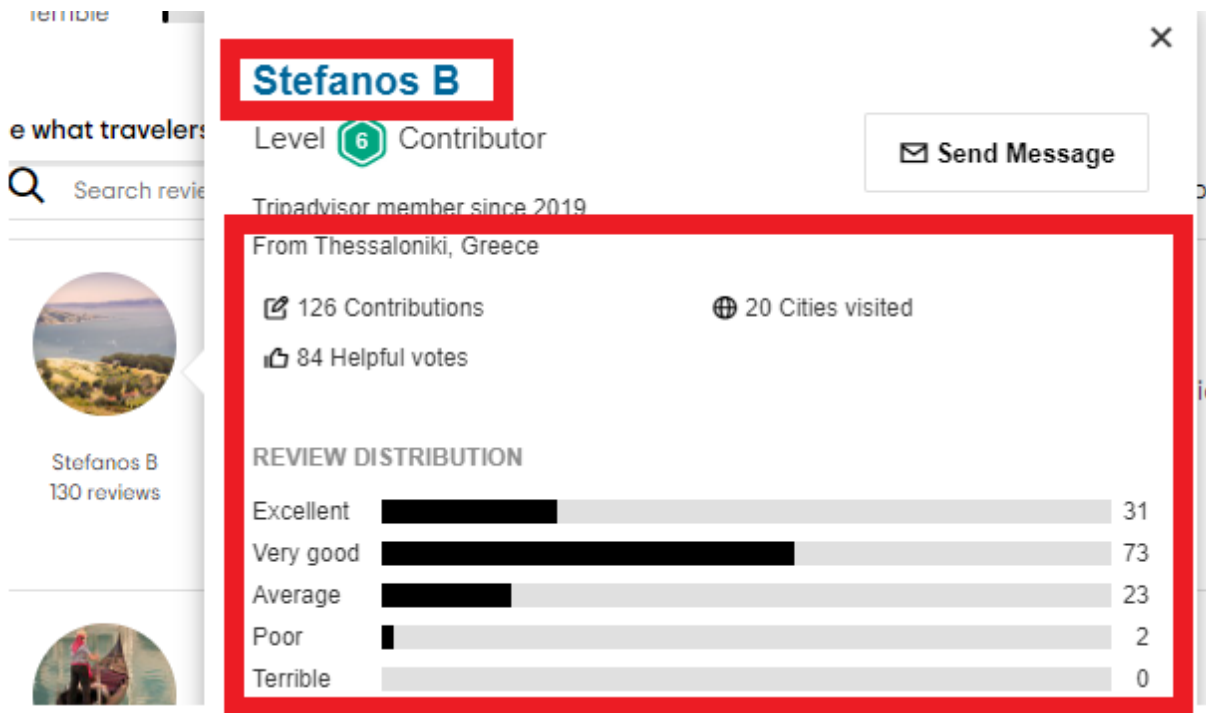
For each review, we are going to scrape the following data :

1. Reviewer username
2. Business reviewed
3. Review date
4. Visit data
5. Review title
6. Review text
7. Review rating



In addition, to extract more data about the reviewer, for each review we are going to click on the reviewer's profile picture (automatically with Selenium) to extract the following information (whenever available) from the pop-up that appears :

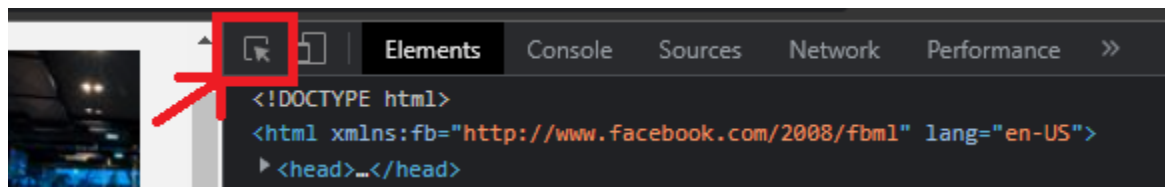
8. Reviewer's age
9. Reviewer's gender
10. Reviewer's location
11. Reviewer's review distribution



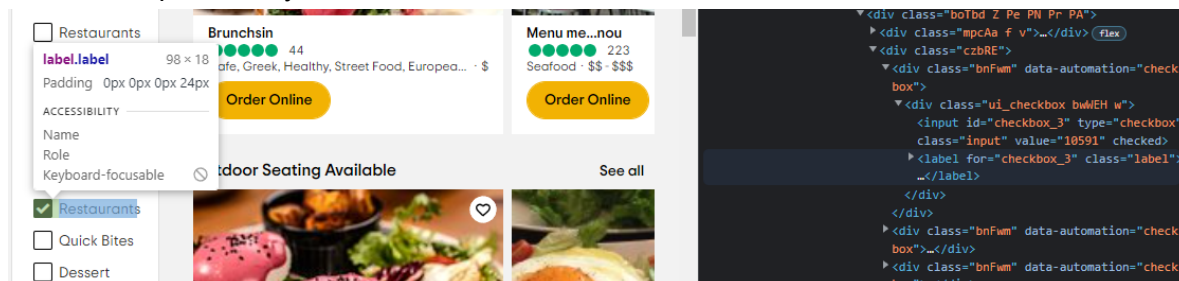
That's it! Using these collected data we are going to unveil new knowledge and produce answers to some questions regarding basic visualizations, sentiment analysis and user profiling.

TripAdvisor dataset web scraping process

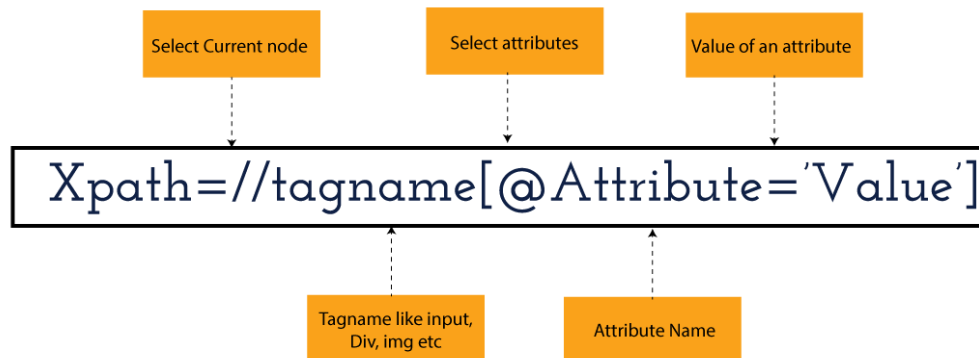
Before we start writing our Python script, we need to find a way to identify all the elements of the website we need to interact with or gather data from. To do so, open the [trip advisor website](https://www.tripadvisor.com) and open the inspector tool. After, we right click anywhere on the site and click "Inspect". On the newly open window we press the icon shown below and select the element we want to inspect.



As an example let's try to uncheck the Restaurants checkbox.



As we can see, the selected element has a parent element with id “checkbox_3”. We can select the element with Selenium given this id and perform the “click” action on it, in order to uncheck the checkbox. There are also other ways to identify an element apart from their id. We can also identify them by their class or by using an xpath. The xpath, is an XML path used to navigate through the HTML structure of the page in order to find one or more elements.



Both the class name and the xpath may include more than one element while the id is guaranteed to be unique.

Using this approach, we gather a list of all the elements we want to interact with.

Bellow are all the elements we are gathering for the purposes of this exercise:

Description	Type	Identification value	Use
Elements needed for the main restaurants page of TripAdvisor			
Cookies accept button	ID	onetrust-accept-btn-handler	Used to press the accept cookies button
Show more establishment types button	CLASS_NAME	fdmYH	Used to expand the list of establishment types on the main page
Restaurants establishment type checkbox	ID	checkbox_3	Used to check the restaurants checkbox state
Restaurants establishment type button	XPATH	<code>./label[@for='{RESTAURANTS_ESTABLISHMENT_TYPE_CHECKBOX[1]}']</code>	Used to click on the restaurants checkbox in order to change its state
Coffee and tea establishment type checkbox	ID	checkbox_6	Used to check the coffee and tea checkbox state
Coffee and tea establishment type button	XPATH	<code>./label[@for='{COFFEE_AND_TEA_ESTABLISHMENT_TYPE_CHECKBOX[1]}']</code>	Used to click on the coffee and tea checkbox in order to change its state
Bars establishment type checkbox	ID	checkbox_241	Used to check the bars checkbox state
Bars establishment type button	XPATH	<code>./label[@for='{BARS_ESTABLISHMENT_TYPE_CHECKBOX[1]}']</code>	Used to click on the bars checkbox in order to change its state
Number of establishments	XPATH	<code>./span[@class='ffdhf b']</code>	Used to gather the total number of establishments
Next page in establishments	XPATH	<code>./a[@class="nav next rndBtn ui_button primary taLnk"]</code>	Used to gather the next page button. After we gather all the establishment urls, we click this button if it exists else we stop the process
List item	XPATH	<code>./div[@data-test='&1_list_item']</code>	Used to gather one item from the establishments list

List item url	XPATH	./a[@class="bHGqj Cj b"]	Used to gather the establishment URL from one item of the establishments list
Elements needed for each establishment page of TripAdvisor (gathered from the main page above)			
Review table	ID	taplc_location_reviews_list_resp_rr_resp_0	Used to gather the table with all the reviews. We wait for this element to appear before we proceed in order to be sure that the reviews are loaded
Reviews count	CLASS_NAME	reviews_header_count	Used to gather the total number of reviews
Review container	XPATH	./div[@class="review-container"]	Used to gather the container with all the reviews
Expand review button	XPATH	//span[@class="taLnk ulBlueLinks"]	The button that expands the reviews that are too long
English reviews radio button	ID	filters_detail_language_filterLang_en	Used to press the English reviews radio button filter
Review title	XPATH	./span[@class="noQuotes"]	Gather the title of one review
Review date	XPATH	./span[contains(@class, 'ratingDate')]	Gather the date of one review
Review rating	XPATH	./span[contains(@class, 'ui_bubble_rating bubble_')]	Gather the rating of one review
Review text	XPATH	./p[@class="partial_entry"]	Gather the text of one review
Date of visit	XPATH	./div[@class="prw_rup prw_reviews_stay_date_hsx"]	Gather the date of visit of one review
Next page in reviews	XPATH	./a[@class="nav next ui_button primary"]	The next page button of the reviews. After we gather all the needed data we press this button if it exists else we terminate the data gathering process for this establishment
Reviewer image	XPATH	./div[@class="prw_rup prw_reviews_member_info_resp"]	Used to press on the reviewers image to open more information related to the user
Reviewer pop up container	CLASS_NAME	ui_overlay	Used to gather the container that contains the user info that are revealed by pressing the user image
Reviewer name	XPATH	./h3[@class="username reviewsEnhancements"]	Gather the name of the reviewer
Reviewer age and town	XPATH	./ul[@class="memberdescriptionReviewEnhancements"]	Gather the age and town of the user. These information are in the same text element and they are gathered together. We split the string after to gather the info separately. This is not included on all users.
Reviewer profile	XPATH	./a[contains(@href, "/Profile/")]	Used to gather the user handle. It is a unique identifier for the user.
Reviewer enhancements	XPATH	./ul[@class="countsReviewEnhancements"]/li/span[@class="badgeTextReviewEnhancements"]	Used to gather extra information about the user. Not included on all users.
Reviewer distribution	XPATH	./span[@class="rowCountReviewEnhancements rowCellReviewEnhancements"]	Used to gather the user's review distribution
Point of interest name	CLASS_NAME	fHibz	The name of the establishment
Reviewer popup close button	XPATH	./div[@class="ui_close_x"]	The button clicked to close the reviewer information popup.
Loading spinner	CLASS_NAME	cssLoadingSpinner	Gather the loading spinner. It is used to understand if the page is still loading or not.

Loading container	ID	taplc_hotels_loading_box_rr_resp_0	Gather the loading container. It is used to understand if the page is still loading or not.
-------------------	----	------------------------------------	---

All the gathered information is stored on a mongo database for later analysis.

Final DataFrame

The final DataFrame has the following format :

```
final_df = pd.DataFrame(df, columns = ['sex', 'contributions', 'cities_visited', 'age', 'photo', 'helpful_votes', 'distribution', 'polarity', 'punc_count', 'capitals', 'num_exclamation_marks', 'num_question_marks', 'num_punctuation', 'num_emojis', 'Sentiment'])
```

final_df															Python
✓	0.8s														
	sex	contributions	cities_visited	age	helpful_votes	photo	distribution	polarity	punc_count	capitals	num_exclamation_marks	num_question_marks	num_punctuation	num_emojis	Sentiment
0	woman	5	141	35-49	1	1	('Excellent': 3, 'Very Good': 1, 'Average': 0...	0.399500	12	8	0	0	12	0	positive
1	man	61	24	35-49	42	179	('Excellent': 35, 'Very Good': 20, 'Average': ...	0.433333	10	7	0	0	10	0	positive
2	woman	87	114	35-49	56	114	('Excellent': 52, 'Very Good': 18, 'Average': ...	0.466667	4	4	1	0	4	0	positive
3	man	88	131	25-34	59	18	('Excellent': 22, 'Very Good': 51, 'Average': ...	0.350000	3	2	0	0	3	0	positive
4	man	24	52	35-49	5	5	('Excellent': 22, 'Very Good': 1, 'Average': 0...	0.512500	2	2	0	0	2	0	positive
...
304	man	63	111	35-49	39	11	('Excellent': 21, 'Very Good': 34, 'Average': ...	0.390741	5	4	0	0	5	0	positive
305	woman	157	125	50-64	82	123	('Excellent': 334, 'Very Good': 781, 'Average': ...	0.775000	2	3	0	0	2	0	positive
306	man	9	2	18-24	1	2	('Excellent': 4, 'Very Good': 3, 'Average': 2...	0.408333	6	7	2	0	6	0	positive
307	woman	905	204	35-49	82	123	('Excellent': 544, 'Very Good': 189, 'Average': ...	0.387857	8	3	0	0	8	0	positive
308	man	186	380	25-34	321	204	('Excellent': 87, 'Very Good': 59, 'Average': ...	-0.187500	9	4	1	0	9	0	positive

309 rows × 15 columns

In the following section we will explain in detail the created columns like the polarity, punc_count, capitals, num_exclamation_marks, num_question_marks, num_punctuation, num_emojis and sentiment.

Answers to the questions of Data Analysis

Basic Visualizations

Question 1.

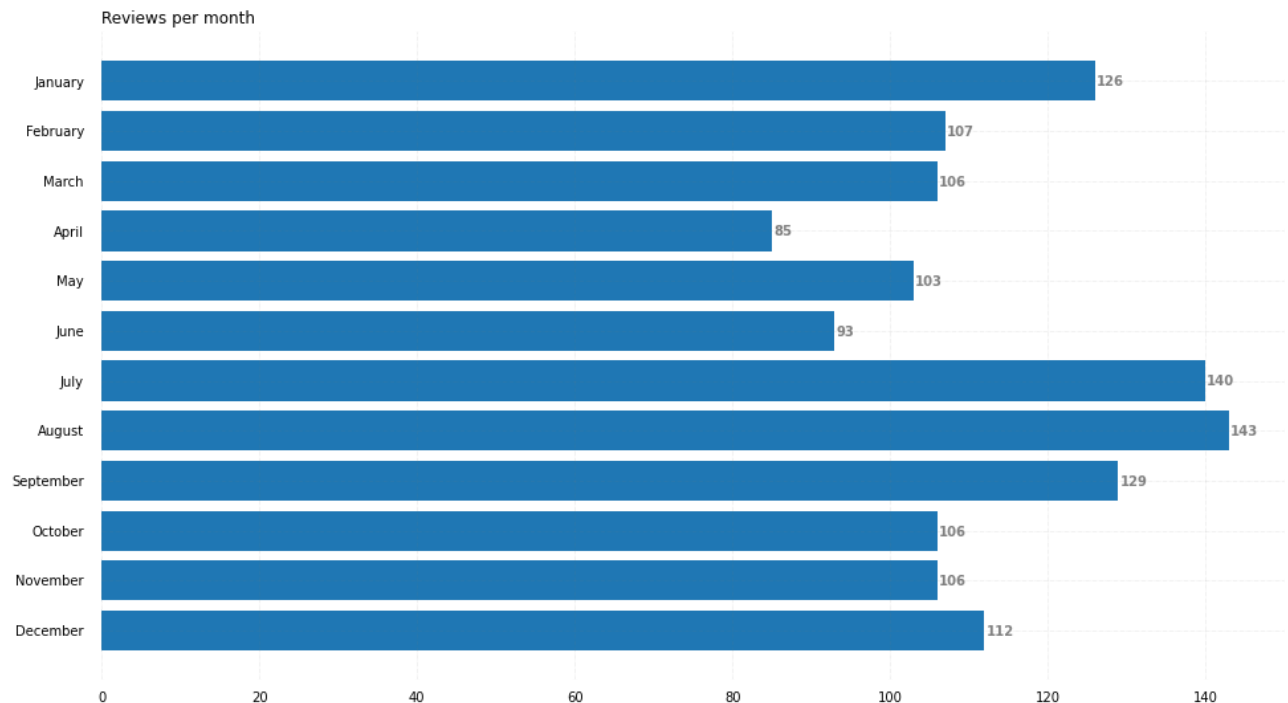
Visualize the number of monthly reviews over time. Which was the month with the most reviews? Is there any seasonality in the volume of reviews?

In order to answer to this question we had to install the following:

```
pip install wordcloud
pip install emoji
pip install gensim
```

At first we are going to get the sum of monthly reviews and visualize a barplot.

```
myquery = { "date": { "$regex": f"^{month}" } }
count = collection.count_documents(myquery)
```



As we can see, the month with the most reviews is August followed by July and September which all are months that most people have their vacations.

To continue, we are making a table with how many reviews we have monthly over the period of 10 years and visualize that with the help of a multiline plot.

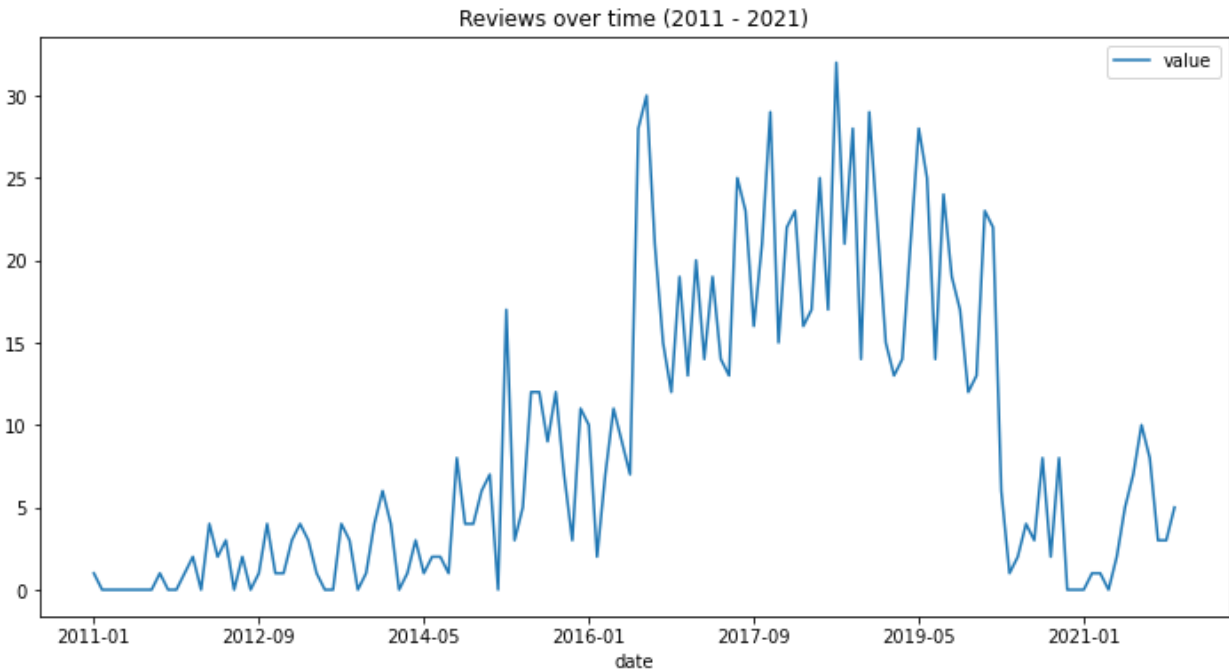
```
table = []
for item in collection.find():
    date = item["date"]
    year = date.split('-', 1)[1]
    month = (date.split('-', 1)[0]).split(' ', 1)[0]
    day = (date.split('-', 1)[0]).split(' ', 1)[1]
    line = year+'-'+get_month(month)
    table.append(line)

table = sorted(table)
timeseries = []

#2011 to 2021
y = 2011
while y<2022:
    for m in range(1,13):
        ym = str(y)+'-'+str(m).zfill(2)
        count = table.count(ym)
        itis = [ym, count]
        timeseries.append(itis)
    y = y+1

df = pd.DataFrame(timeseries, columns=['date', 'value'])
print(df.head())
df.set_index(pd.to_datetime(df['date']), inplace=True)
ax = df.plot(x='date', y='value', figsize=(12,6), title='Reviews over
time (2011 - 2021)')
```

In the plot (next page) we can see how the trend of tripadvisor is increasing over the years (note that it was created in 2007) and how the review numbers are affected by the coronavirus as we have a huge decrease in reviews at the end of 2019.



Next, in order to find if there is seasonality in how many reviews we have during 10 years, we are going to create a multi line plot where each line is a different year and in x-axis we have the months.

```
table2 = []
for item in collection.find():
    date = item["date"]
    year = date.split(', ', 1)[1]
    month = (date.split(', ', 1)[0]).split(' ', 1)[0]
    day = (date.split(', ', 1)[0]).split(' ', 1)[1]
    line = year+get_month(month)+day.zfill(2)
    if int(year) > 2010 and int(year)!=2022:
        table2.append(line)

table2 = sorted(table2)
timeseries2 = []
for item in table2:
    value = table2.count(item)
    timeseries2.append([item, value])

df = pd.DataFrame(timeseries2, columns=['Date', 'Value'])
df.set_index(pd.to_datetime(df['Date']), inplace=True)
df = pd.DataFrame(timeseries2, columns=['Date', 'Value'])
```

```

# Fill the dates that are missing
df.set_index(pd.to_datetime(df['Date']), inplace=True)
df = df.resample('D').sum().fillna(0)

# Prepare data and group by month
df['year'] = [d.year for d in df.index]
df['month_n'] = [d.month for d in df.index]
df['month'] = [d.strftime('%b') for d in df.index]
years = df['year'].unique()
df =
df.groupby(['year', 'month_n', 'month'], as_index=False).agg({'Value': 'sum'})

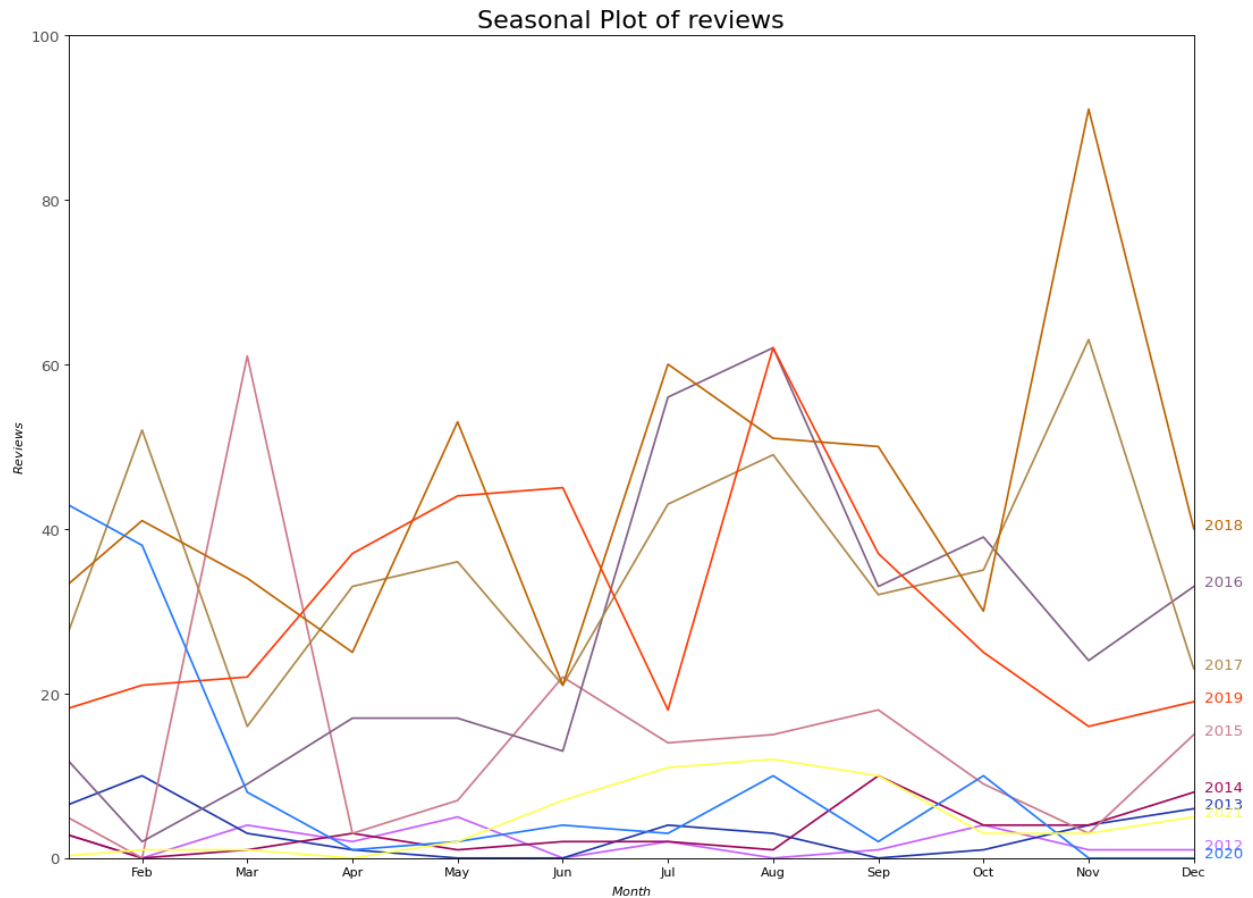
# Prepare Colors
np.random.seed(100)
mycolors = np.random.choice(list(mpl.colors.XKCD_COLORS.keys()),
len(years), replace=False)

# Draw Plot
plt.figure(figsize=(16,12), dpi= 80)
for i, y in enumerate(years):
    if i > 0:
        plt.plot('month', 'Value', data=df.loc[df.year==y, :],
color=mycolors[i], label=y)
        plt.text(df.loc[df.year==y, :].shape[0]-.9, df.loc[df.year==y,
'Value'][-1:].values[0], y, fontsize=12, color=mycolors[i])

# Decoration
plt.gca().set(xlim=(0.3, 11), ylim=(0, 100), ylabel='$Reviews$',
xlabel='$Month$')
plt.yticks(fontsize=12, alpha=.7)
plt.title("Seasonal Plot of reviews", fontsize=20)
plt.show()

```

So, in the next plot, as we already guessed from the monthly barplot, even if the most reviews were in November of 2017 and 2018 that is the case for only these two years and actually most of the years have a peak in review numbers in the month of August.



Question 2.

Visualize the most common words, bi-grams, and tri-grams across all reviews through a bar chart or word cloud. Also, visualize the most common words, bi-grams, and tri-grams in 5-star versus 1-star reviews.

First we are going to get all the texts from our database and place it in three strings, one for all the text, one for 1-star reviews and one for 5-star reviews.

```
for item in collection.find():
    text_all = text_all+' '+item['text']
    if item['review_rating'] == '10':
        text_10 = text_10+' '+item['text']
    elif item['review_rating'] == '50':
        text_50 = text_50+' '+item['text']
```

Next we are going to get the bi-grams and tri-grams from the three texts using the following function and then we are going to visualize them into a barplot like we did in the first question.

```

def get_grams(text):
    new_text =
gensim.parsing.preprocessing.remove_stopwords(remove_emoji(text.translate(s
tr.maketrans('', '', string.punctuation)).lower()))
    words = new_text.split()

    words_zip = zip(words, words[1:])
    words_zip2 = zip(words, words[1:], words[2:])
    two_grams_list = [item for item in words_zip]
    three_grams_list = [item for item in words_zip2]

    count_freq = {}
    count_freq2 = {}
    for item in two_grams_list:
        if item in count_freq:
            count_freq[item] +=1
        else:
            count_freq[item] = 1

    for item in three_grams_list:
        if item in count_freq2:
            count_freq2[item] +=1
        else:
            count_freq2[item] = 1

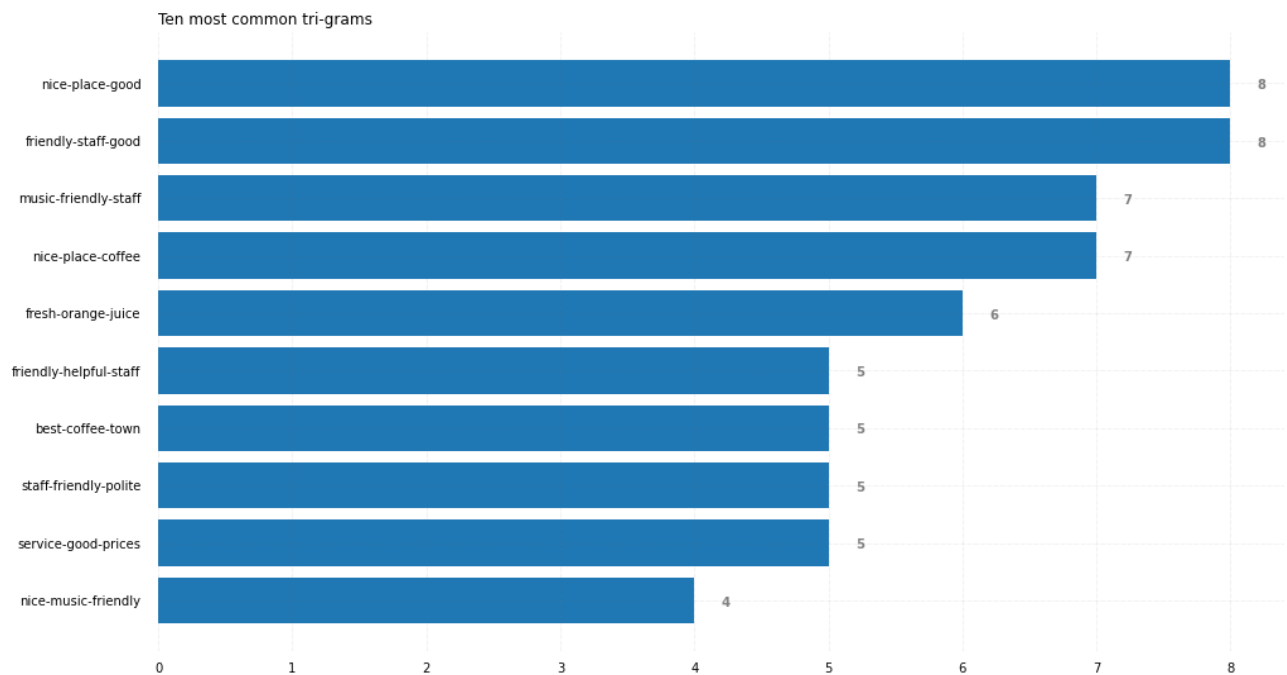
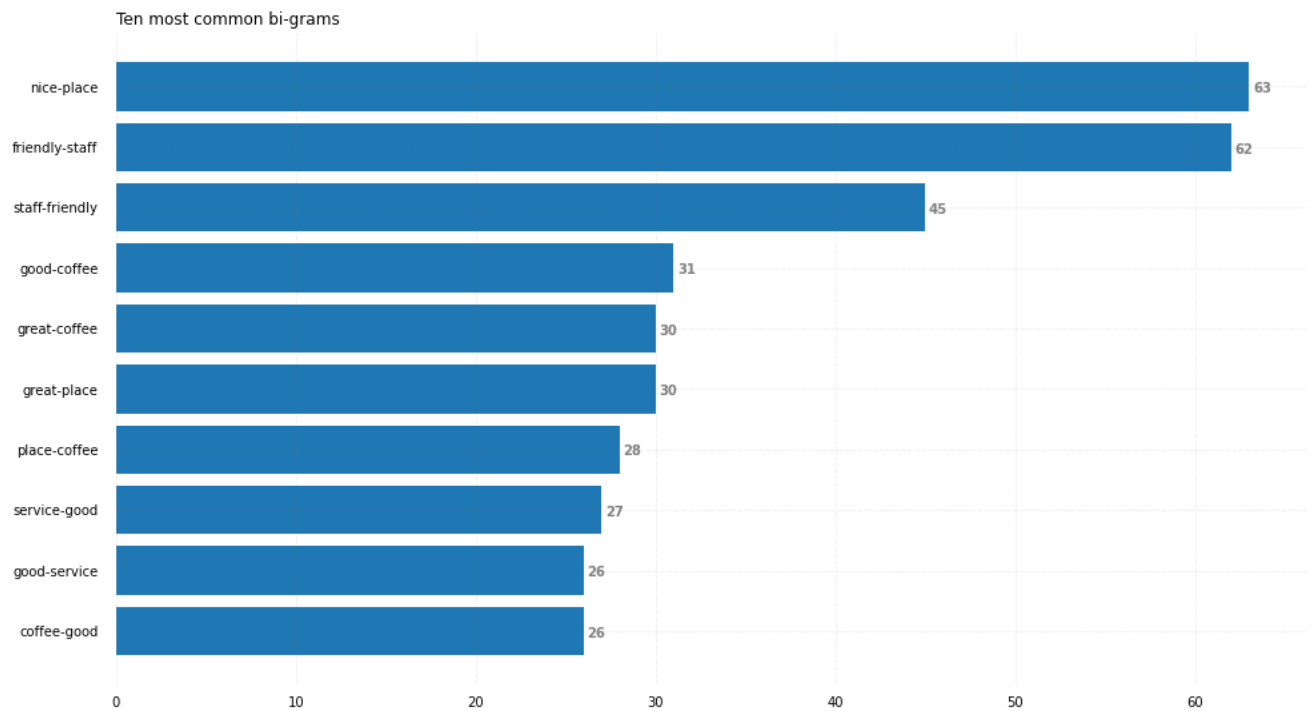
    sorted_two_grams = sorted(count_freq.items(), key=lambda item: item[1],
reverse = True)
    sorted_three_grams = sorted(count_freq2.items(), key=lambda item:
item[1], reverse = True)
    return sorted_two_grams, sorted_three_grams

```

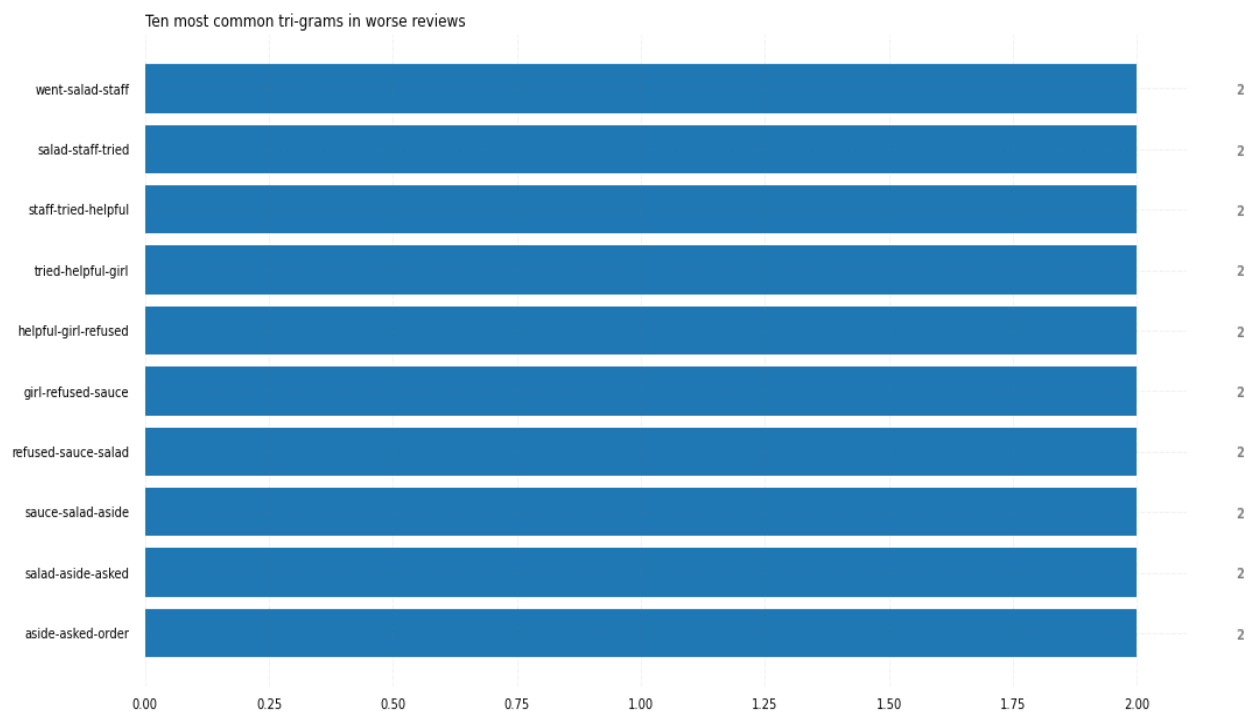
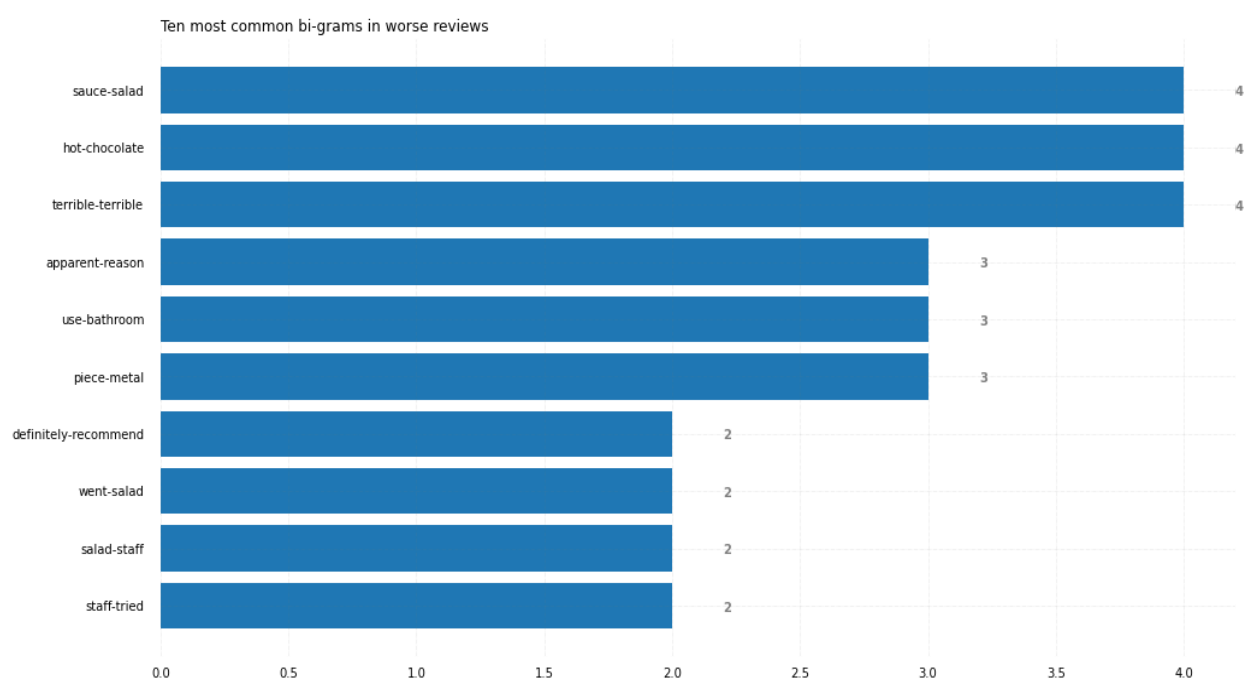
Result:

In the bi-grams and tri-grams for every text nice-place and nice-place-good are the winners. Most of our reviews were 5-star reviews and so this is

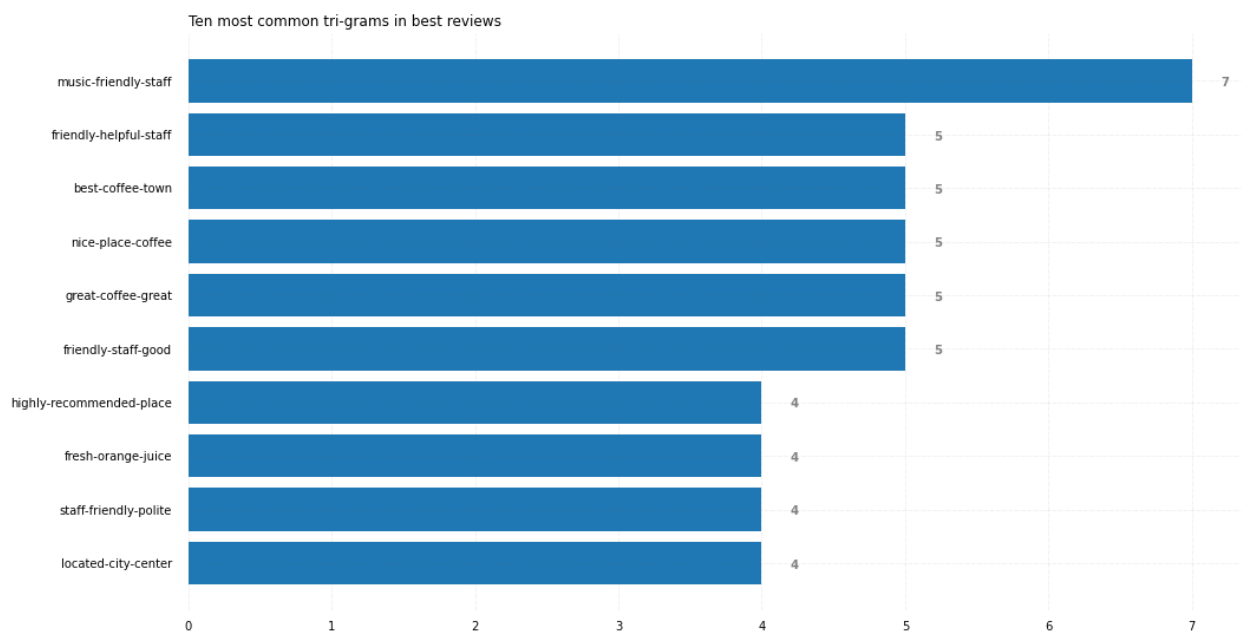
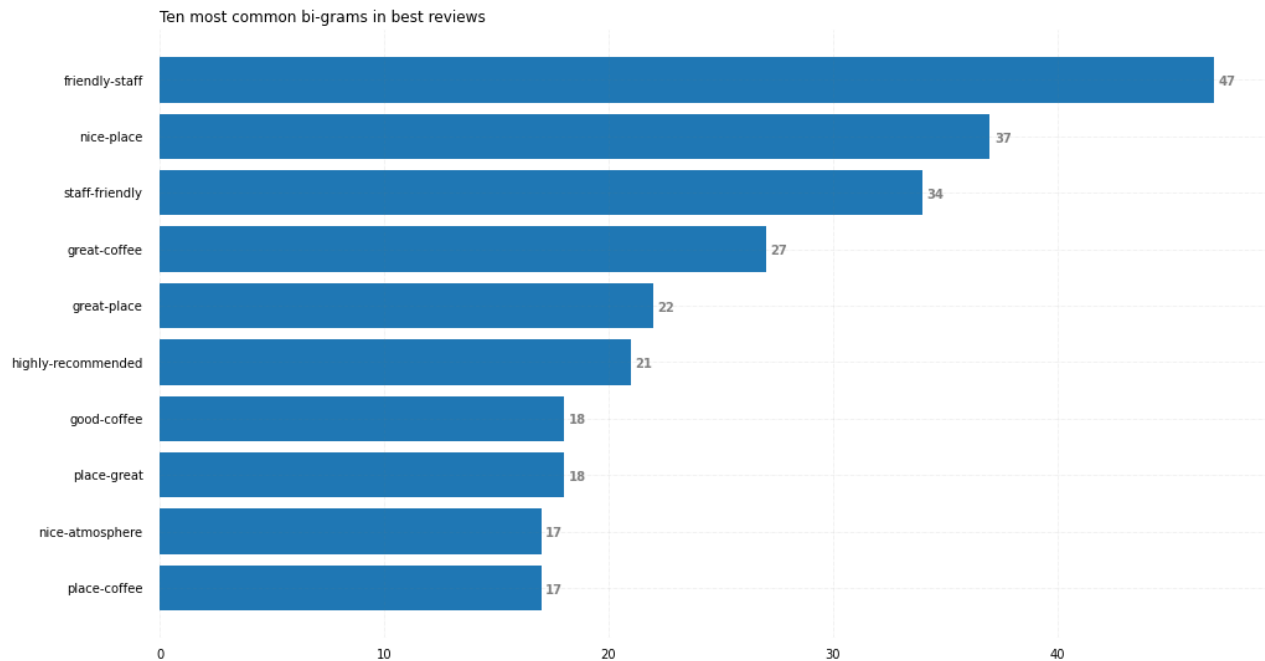
Here we can see the ten most common bi-grams and tri-grams in every text.



Here we can see the ten most common bi-grams and tri-grams in the 1-star reviews.



Here we can see the ten most common bi-grams and tri-grams in the 5-star reviews.



To continue, we are going to create 3 word clouds. But first, what is a wordcloud? A word cloud is a visual representation object for text processing, which shows the most frequent word with

For the word clouds we are going to use the same texts as before, but this time we are going to remove any punctuation, emoji and stopwords.


```

for word in words:
    k=k+1
    print(k, '/', len(words))
    row = [word]
    for year in range(2011, 2022):
        yearly_reviews_query = { "date": {"$regex":
f"{str(year)}", "$options": "$i"}}
        yearly_reviews =
collection.count_documents(yearly_reviews_query)
        myquery = {"$and": [{ "text": {"$regex":
f"{word}", "$options": "$i"}}, { "date": {"$regex": f"{str(year)}",
"$options": "$i"}}]}
        percentage = (collection.count_documents(myquery) /
yearly_reviews)*100
        row.append(percentage)
    c = 0
    for i in row[1:12]:
        c = c + i
    if c >= 100:
        print(row)
        words_table.append(row)

return(words_table)

```

This function (where “text” is a string which contains every text from our dataset from which we have removed punctuation, emojis and stop words) gives us a table of unique words and how many times they appeared in the reviews throughout the years. Then we are going to get the last year count minus the first year count and we are going to print a plot for the 10 fastest growing and a plot for the 10 shrinking words according to frequency.

For example for the fastest growing words we have:

```

# Plotting fastest growing words
X = ['2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018',
'2019', '2021']

# Initialize the subplot function using number of rows and columns
figure, axis = plt.subplots(5, 2, figsize=(20, 40))

i = 0

```

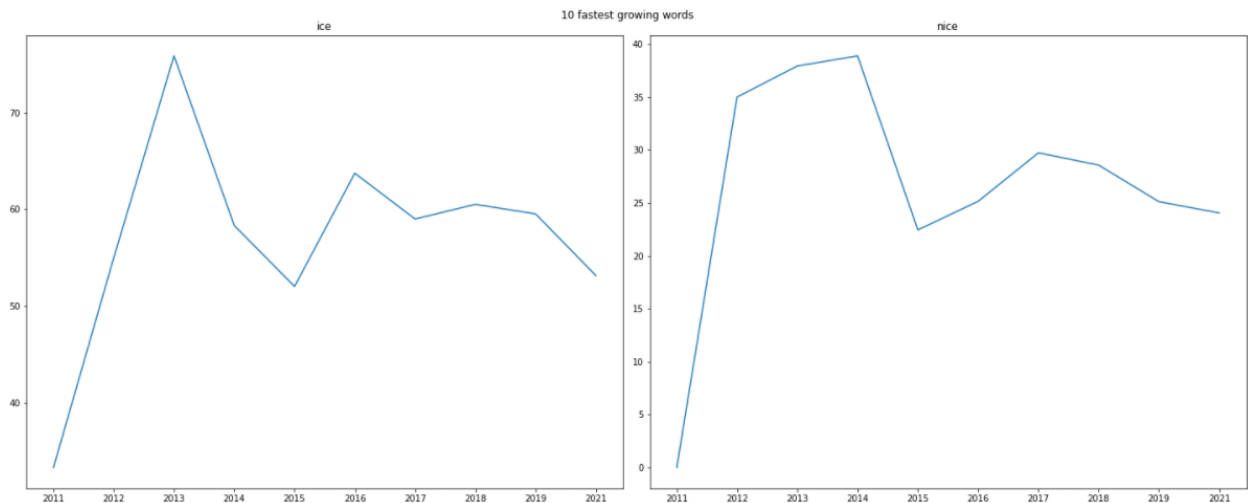
```

for row in fastest_growing_table[:5]:
    Y = row[1:11]
    axis[i,0].plot(X,Y)
    axis[i,0].set_title(row[0])
    i = i+1

i=0
for row in fastest_growing_table[6:12]:
    Y = row[1:11]
    axis[i,1].plot(X,Y)
    axis[i,1].set_title(row[0])
    i = i+1

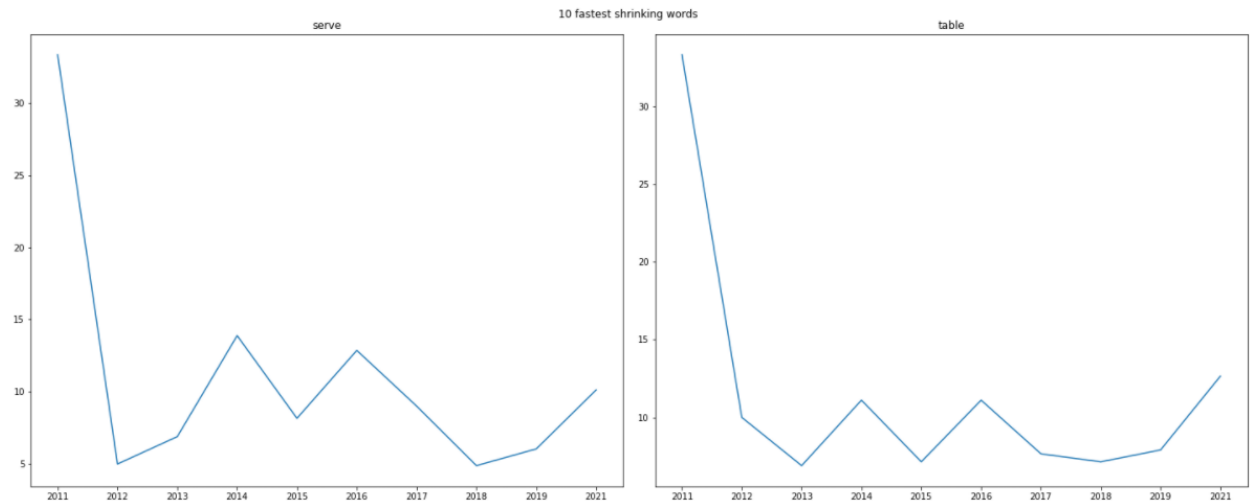
figure.suptitle("10 fastest growing words")
figure.tight_layout()
figure.subplots_adjust(top=0.97)
# Combine all the operations and display
plt.show()

```



Where on y axis we have the frequency of appearance.

And for the fastest shrinking words we have:



(The whole plots can be seen in the github repository)

Question 4.

Explore and visualize emerging topics from all the user reviews across time, similar to here. Which topics do you identify? Do they make sense to you?

In this case the texts should be preprocessed as in the previous steps with removing punctuation, emojis and stop words.

Then we are going to create a dataframe where we have our text and processed text and also the year this text was used.

First we are going to create a corpus and a dictionary with the words from our texts (df_4 is our dataframe).

```
data = df_4.text_processed.values.tolist()
data_words = list(sent_to_words(data))
# remove stop words
data_words = remove_stopwords(data_words)

# Create Dictionary
id2word = corpora.Dictionary(data_words)
# Create Corpus
texts = data_words
# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]
```

Then we are going to create 10 topics for which we are going to run the LDA.

```
# number of topics
num_topics = 10
# Build LDA model
lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                       id2word=id2word,
                                       num_topics=num_topics)

# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

At last we are visualizing our topics.

```
# Visualize the topics
pyLDAvis.enable_notebook()
LDAvis_data_filepath =
os.path.join('./plots/basic_visualizations/ldavis_prepared_'+str(num_
topics))

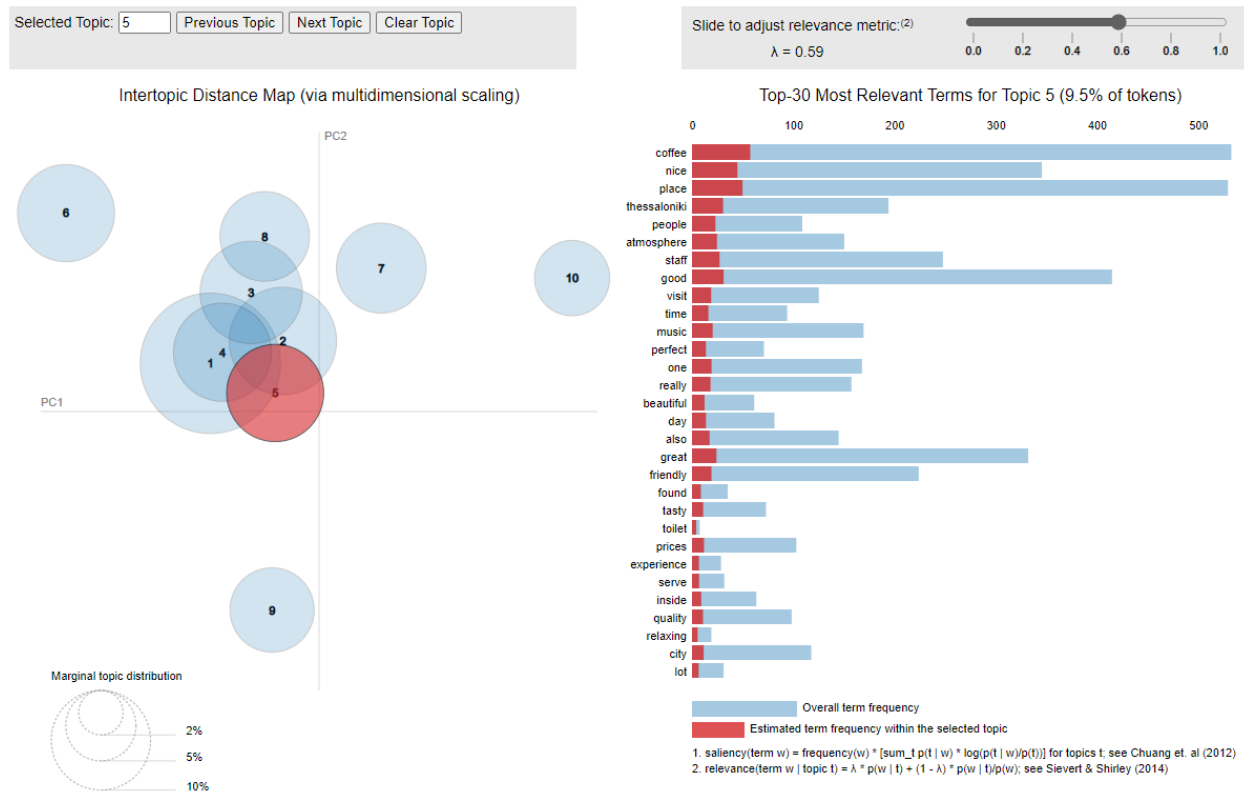
LDAvis_prepared = pyLDAvis.gensim_models.prepare(lda_model, corpus,
id2word)
with open(LDAvis_data_filepath, 'wb') as f:
    pickle.dump(LDAvis_prepared, f)

# Load the pre-prepared pyLDAvis data from disk
with open(LDAvis_data_filepath, 'rb') as f:
    LDAvis_prepared = pickle.load(f)
pyLDAvis.save_html(LDAvis_prepared,
'./plots/basic_visualizations/ldavis_prepared_'+ str(num_topics)
+'.html')
LDAvis_prepared
```

So for example for topic 5 which is:

```
(5,
 '0.023*"place" + 0.019*"great" + 0.016*"coffee" + 0.012*"best" + '
 '0.010*"good" + 0.008*"staff" + 0.008*"really" + 0.008*"service" + '
 '0.007*"atmosphere" + 0.007*"thessaloniki"')
```

And $\lambda = 0.59$ we have:



Question 5.

Visualize in a map the locations of the reviewers based on the collected data with color coding based on the location frequency. Prefer countries over cities to identify more than one occurrence.

First we need to create a list with the countries and how many reviewers from this country we had.

Note: The reviewers from the USA have their state instead of country so we have to change it to 'United States of America'. Also, the country 'Republic of North Macedonia' in the json we will use later for the map creation is named 'Macedonia' so we are going to change that too.

```
with open('united_states.csv', newline='') as f:
    reader = csv.reader(f)
    united_states = list(reader)
```

```
united_states = [row[0] for row in united_states]
```

```
individual_handles = []
reviewers = []
individual_countries = []
```

```

countries_n = []
for item in collection.find():
    reviewer = item["reviewer"]
    if 'handle' in reviewer:
        if reviewer["handle"] not in individual_handles:
            individual_handles.append(reviewer["handle"])
        if "location" in reviewer:
            location = reviewer["location"].split(', ', 1)
            if len(location) == 2:
                country = location[1]
                if country in united_states:
                    if [reviewer["handle"], 'United States of America']
not in reviewers:
                        reviewers.append([reviewer["handle"], 'United
States of America'])
                elif country == 'GREECE':
                    reviewers.append([reviewer["handle"], 'Greece'])
                elif country == 'Republic of North Macedonia':
                    reviewers.append([reviewer["handle"], 'Macedonia'])
                elif country == 'Nevada, United States':
                    reviewers.append([reviewer["handle"], 'United
States'])
            else:
                reviewers.append([reviewer["handle"], country])
            if(country not in individual_countries):
                if country in united_states:
                    if 'United States of America' not in
individual_countries:
                        individual_countries.append('United States
of America')

                elif country == 'GREECE':
                    individual_countries.append('Greece')
                elif country == 'Republic of North Macedonia':
                    individual_countries.append('Macedonia')
                elif country == 'Nevada, United States':
                    individual_countries.append('United States')
                else:
                    individual_countries.append(country)

for state in united_states:
    if state[0] in reviewers[1]:
        reviewers[1] = list(map(lambda x: x.replace(state[0], 'United
States of America'), reviewers[1]))

```



```

for country in individual_countries:
    count = 0
    for reviewer in reviewers:
        if reviewer[1] == country:
            count = count + 1
    countries_n.append([country, count])

# open the file in the write mode
with open('countries.csv', 'w') as f:
    # create the csv writer
    writer = csv.writer(f)

    # write a row to the csv file
    for row in countries_n:
        writer.writerow(row)

```

When we finally have all the countries and the reviewer count, we are going to create the map. First install folium.

```

pip install folium

```

Then use it to create our map.

```

import folium

# initialize the map and store it in an object
m = folium.Map(location = [40, -95],
                zoom_start = 4)

countries_geo =
"https://raw.githubusercontent.com/python-visualization/folium/main/examples/data/world-countries.json"

columns=['country', 'reviewers']
countries_csv = 'countries.csv'
df = pd.read_csv(countries_csv, names=columns)

folium.Choropleth(

    # geographical locations
    geo_data = countries_geo,
    name = "choropleth",

```

```

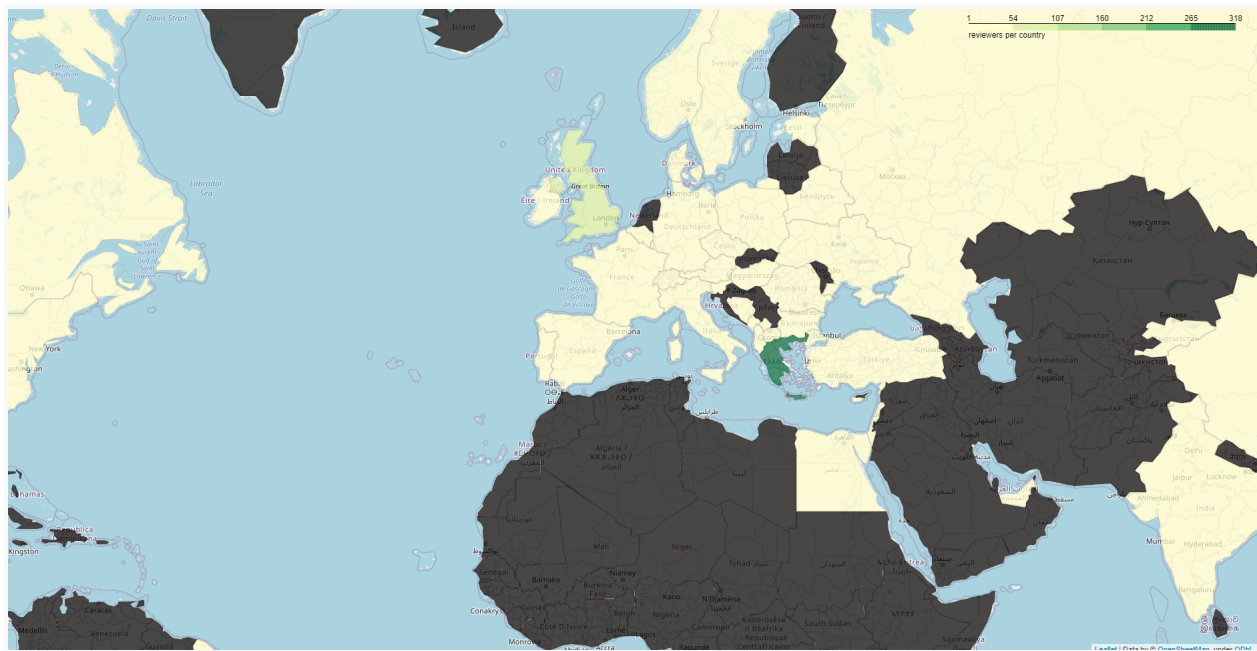
# the data set we are using
data = df,
columns = ["country", "reviewers"],

# YlGn refers to yellow and green
fill_color = "YlGn",
fill_opacity = 0.7,
line_opacity = .1,
key_on = "feature.properties.name",
legend_name = "reviewers per country",
).add_to(m)

m.save('final_map.html')

```

The file 'final_map.html' that is produced can be opened with a browser where we can also use the zoom feature.



As we expect most of the reviewers are from Greece with around 300 reviewers, the second country with the most reviewers is the United Kingdom which has 88 reviewers and the rest of the countries are yellowish white because they have less than 54 reviewers. The black countries are the ones for which there are no reviewers.

Sentiment Analysis

For sentiment analysis, we are going to use the TextBlob python library to process the data. So before we do anything else we need to make sure we have installed the TextBlob library by using the command below:

```
pip install -U textblob
```

Textblob is dependent on the NLTK library which is a natural language processing toolkit. NLTK will be also installed during textblob installation but we will also need to download some NLTK plugins that we need for the analysis we will do.

To install the needed NLTK plugins, we execute the python commands below.

```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('brown')
nltk.download('averaged_perceptron_tagger')
```

Before we start with the preprocessing and the analysis, we need to decide how we will handle and process data and how/if we will save them. For the purpose of this exercise, we decided to handle all the data in memory, as there are not so many, using some classes to handle them ([check section 4](#)).

Now let's do some preprocessing on the reviews' text to make them easier to work with.

For the reviews' text, we perform the following preprocessing steps:

- Convert all the texts to lowercase
- Remove links and emails
 - To remove links, we use the regular expression below

```
'http\S+'
```

- To remove the emails, we use the regular expression below

```
'\S*@\S*\s?'
```

- Using textblob, we iterate on each sentence of each review and perform the following action on each sentence
 - We ignore single letter words
 - We remove all the special characters using the regular expression below
- We correct the spelling of each word of the sentence using the textblob library

```
'[^A-Za-z0-9 ]+'
```

- We lemmatize each verb and noun of the sentence using the textblob library. We do not lemmatize adjectives because each different adjective can have a different impact on the sentiment. For example “best” is more positive than “better” which is more positive than “good”.
- We choose not to stem the words due to stemming issues resulting in unrecognizable words but we do lemmatize verbs and nouns.
- Lastly we form a new text with the changed sentences and save it in order to perform the sentiment analysis on them
- [Check section 5 of the code](#)

At this point we choose not to remove the stop words because some of them (like no, non and not) can give some context when trying to calculate the sentiment polarity of a review (how positive or negative it is). As an alternative solution, we could decide what stop words we need to keep and remove the rest. For this example we decided to keep them all and exclude just the single letter words.

Using textblob functionality, we calculate the polarity score of each sentence of each review. Each sentence has a polarity between 1 and -1 and the total polarity score of the review is the sum of all the polarities of each sentence. We get the sum of polarities and not the mean polarity because we want to give more weight to bigger and more detailed reviews and don't take into consideration the polarity of neutral sentences (which will have polarity 0). [Check section 6](#) for the code implementation.

Below are our findings based on the mentioned process.

Top 5 positive reviews

Original review	Processed review	Polarity Score
<i>The best coffee-cocktail bar in the city! It is hard to choose the best-favorite cocktail because all of them are delicious! Also, there you can find the best quality for coffee! Beautiful and unique decoration! Many congrats!!! Best of luck!</i>	the best coffeecocktail bar in the city. it be hard to choose the bestfavorite cocktail because all of them be delicious. also there you can find the best quality for coffee. beautiful and unique decoration. many congress. best of luck.	4.47
<i>This place is a gem! It is so unique and exceptional! The carrot cake is the best I have ever tried, whilst the new your cheese cake is simply perfect. The flavors are balanced so well, the servings are perfect and the freshness of the deserts is amazing. Try the macarons, they are full of flavor and as chewy as the best you can ever eat. The people that run the place are true connoisseurs of the art, using the best ingredients available. Support their effort, they truly deserve it. And...they make superior Hellenic coffee....</i>	this place be gem. it be so unique and exceptional. the carrot cake be the best have ever try whilst the new your cheese cake be simply perfect. the favor be balance so well the serve be perfect and the freshness of the desert be amaze. try the macarons they be full of flavor and a chew a the best you can ever eat. the people that run the place be true connoisseur of the art use the best ingredient available. support their effort they truly deserve it. another make superior hellenic coffee.	4.19

<p><i>Walking across Mitropoleos street this cafe will immediately fall into your consideration. Excellent design,nice layout and beautiful decoration. Walking in the store, the staff is welcoming you warm hearted and the guy at the door will propose you a table/position to sit at, which best fits you and your company (as opposed to number of people etc.), leaving you the option to choose whether you like it or if you want to sit somewhere else. The staff which is very polite and hospitable,will recommend beverages including a long list of wines and signature cocktails, as well as very delicious and fine chasers like salads,bruschettes and pizza to combine with you drink (try the ARC pizza, brown ocean bruschettes and salmon bruschettes they are amazing). A place for any occasion either business meeting, going out with friends or with your loved one. Nice music, nice people and reasonable prices. Just perfect!!!</i></p>	<p>walk across mitropoleos street this cafe will immediately fall into your consideration. excellent design,nice layout and beautiful decoration. walk in the store the staff be welcome you warm hearted and the guy at the door will propose you tableposition to sit at which best fit you and your company a oppose to number of people etc. leave you the option to choose whether you like it or if you want to sit somewhere else. the staff which be very polite and hospitablewill recommend beverage include long list of win and signature cocktail a well a very delicious and fine chaser like saladsbruschettes and penza to combine with you drink try the arc penza brown ocean bruschettes and salmon bruschettes they be amaze. place for any occasion either business meet go out with friend or with your love one. nice music nice people and reasonable price. just perfect. .</p>	4.06
<p><i>If you like speciality coffee and you did not visit this place while you were in Thessaloniki, it is like going to Paris without visiting the Eiffel Tower. You'll find a great number of freshly roasted speciality coffee's for pour-over, V60 and even for espresso! They also have a very nice shop where you can find all the right gear. The coffee's are of really exceptional quality. Deep and rich of flavours. Great service too! We had the honor to talk to the owner. He is a very professional coffee roaster and true coffee geek! We had a lovely experience!</i></p>	<p>if you like specially coffee and you do not visit this place while you be in thessaloniki it be like go to paris without visit the life tower. you find great number of freshly roast specially coffee for pourover ve and even for express. they also have very nice shop where you can find all the right gear. the coffee be of really exceptional quality. deep and rich of favour. great service too. we have the honor to talk to the owner. he be very professional coffee roast and true coffee week. we have lovely experience.</p>	3.77
<p>I love this place everyone and everything was excellent. All the pastry and sandwiches were so fresh and healthy. All the staff are so kind and generous. I recommend everyone to visit this excellent bakery in Thessaloniki. One of the best places I've been in the last years. Keep it up Albeta and hope to see you soon. Can't wait to eat my favorite Cheesecake and drink my favourite coffee. Thank you</p>	<p>love this place everyone and everything be excellent. all the pantry and sandwich be so fresh and healthy. all the staff be so kind and generous. recommend everyone to visit this excellent baker in thessaloniki. one of the best place give be in the last year. keep it up albert and hope to see you soon. can wait to eat my favorite cheesecake and drink my favourite coffee. thank you.</p>	3.75

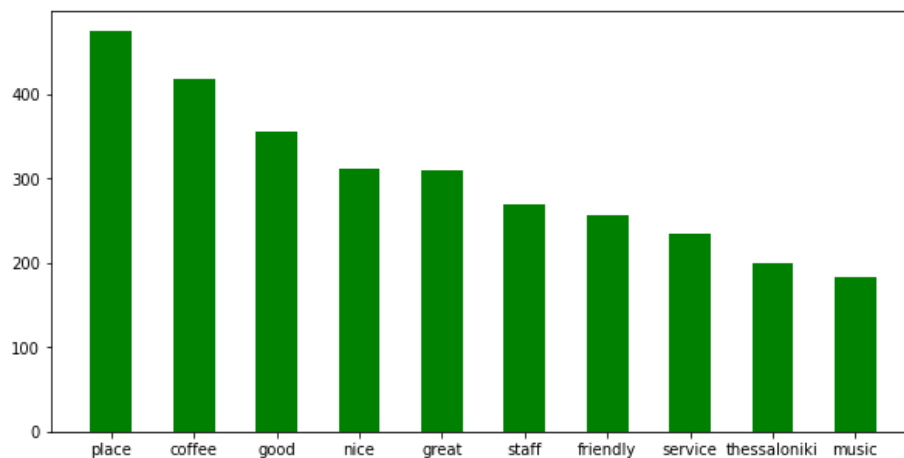
Top 5 negative reviews

Original review	Processed review	Polarity Score
<i>I was going to the same Cafe Bar the last 20 years on weekends. The last years they start having bad attitude with customers. Today was the worst services i ever had. I tried to have a drink and chill out for a while. They did 10-15 minutes to give order. When they brought my drink they asked me to paid. After 30 minutes waitress came to me and told me that she have to takes my drink and anything that exist on the table. I didn't understand what see mean and i let here to clean the table. After that see asked me to give a new order or else that i have to leave and let other to sit down and order. So she didn't allow me to drink my half finished glass and my last experience was truly awful. I will not going back there again for any reason. Now They are one more Cafe Bar with Bad Services and Bad Quality Drinks</i>	be go to the same cafe bar the last 20 year on weekend. the last year they start have bad attitude with customer. today be the worst service ever have. try to have drink and chill out for while. they do 1015 minute to give order. when they bring my drink they ask me to pay. after 30 minute witness come to me and tell me that she have to take my drink and anything that exist on the table. didn understand what see mean and let here to clean the table. after that see ask me to give new order or else that have to leave and let other to sit down and order. so she didn allow me to drink my half finish glass and my last experience be truly awful. will not go back there again for any reason. now they be one more cafe bar with bad service and bad quality drink.	-2.06
<i>We had been to Allegro before for a cup of coffee, and decided to go there for dinner. Unfortunately, we were greatly disappointed due to the following reasons: 1. While in the middle of the COVID-19 outbreak, the staff was not following any health security measures. The staff's masks did not cover their mouth and nose at any point while serving our food, making us very uncomfortable. Masks should not be just for show or in case an inspection takes place. 2. Not sufficient social distancing. Tables were too close together, and chairs were literally back to back. 3. Food was mediocre. The bruschetti were terrible. The bread was not even the average bread you can get at any local bakery shop, it was ready-made sliced bread, the one you get at the supermarket! The beef tagliata was not great either, we were surprised not being asked how we would like it cooked, in fact it was served overdone. Overall, it is a nice place to enjoy coffee or drinks with a view, but better spend your money elsewhere for dinner.</i>	we have be to allegro before for cup of coffee and decide to go there for dinner. unfortunately we be greatly disappoint due to the follow reason while in the middle of the covid19 outbreak the staff be not follow any health security measure. the staff mass do not cover their mouth and nose at any point while serve our food make u very uncomfortable. mass should not be just for show or in case an inspection take place. not sufficient social distend. table be too close together and chair be literally back to back. food be mediocre. the bruschetti be terrible. the bread be not even the average bread you can get at any local baker shop it be readymade slice bread the one you get at the supermarket. the beef tagliata be not great either we be surprise not be ask how we would like it cook in fact it be serve everyone. overall it be nice place to enjoy coffee or drink with view but better spend your money elsewhere for dinner.	-1.99
<i>When we arrived, the waiter turned and looked at us in a very unpleasant way, like we were doing something wrong. I said we were here for a drink and was that OK? He said "Excuse me?" as if I'd said something completely ridiculous. I didn't know how to respond, so we continued and he followed us in and talked to the bartender in Greek as he made our (very expensive!) drinks. I don't know what he was saying, but he was gesturing towards us and</i>	when we arrive the waiter turn and look at u in very unpleasant way like we be do something wrong. say we be here for drink and be that ok. he say excuse me. a if id say something completely ridiculous. didn know how to respond so we continue and he follow u in and talk to the bartender in greek a he make our very expensive. drink. dont know what he be say but he be restore towards u and the bartender be laugh awkwardly. once we have	-1.78

<p><i>the bartender was laughing awkwardly. Once we had our drinks, we went outside and tried to find a seat. We saw someone sitting alone with 3 empty seats by him, and went over and asked if we could take the spare seats. He said there was someone sitting opposite, so we asked to take the two seats next to them, and he said "Sorry, it's for 2 people." So we said, "Well, if we don't use the table, can we please take the chairs?" he said that was OK, so we pulled the seats a short distance from the table and sat down, already feeling very awkward. His friend came out and started gesturing at us and shouting. The waiter came out and told us to move to a different table. We had done absolutely nothing wrong and not attempted to bother these people in anyway, just wanted to sit in nearby seats. The waiter was talking to them in an apologetic tone. We finished our drinks and left as quickly as possible, feeling like we'd been thoroughly rejected.</i></p>	<p>our drink we go outside and try to find seat. we saw someone sit alone with empty seat by him and go over and ask if we could take the spare seat. he say there be someone sit opposite so we ask to take the two seat next to them and he say sorry it for people. so we say well if we dont use the table can we please take the chair. he say that be ok so we pull the seat short distance from the table and sit down already feel very awkward. his friend come out and start restore at u and shout. the waiter come out and tell u to move to different table. we have do absolutely nothing wrong and not attempt to bother these people in anyway just want to sit in nearby seat. the waiter be talk to them in an apologetic tone. we finish our drink and leave a quickly a possible feel like wed be thoroughly reject.</p>	
<p><i>Terrible service, expensive drinks, rude owner. One of the worst cafes i have ever visit in Thessaloniki.</i></p>	<p>terrible service expensive drink rude owner. one of the worst case have ever visit in thessaloniki.</p>	-1.6
<p><i>A place with a really bad vibe, awful decoration, reminds you of a hospital. Both food and drinks are expensive, and the service is cold with you. Not enough space on the inside. I wouldn't reccomend this place.</i></p>	<p>place with really bad vice awful decoration remind you of hospital. both food and drink be expensive and the service be cold with you. not enough space on the inside. wouldn recommend this place.</p>	-1.4

[Check section 7](#) for the code related to the most positive and negative reviews.

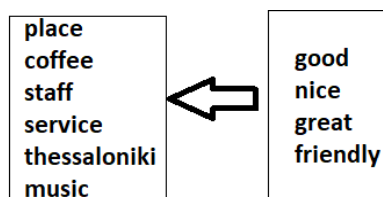
Most common words among positive reviews



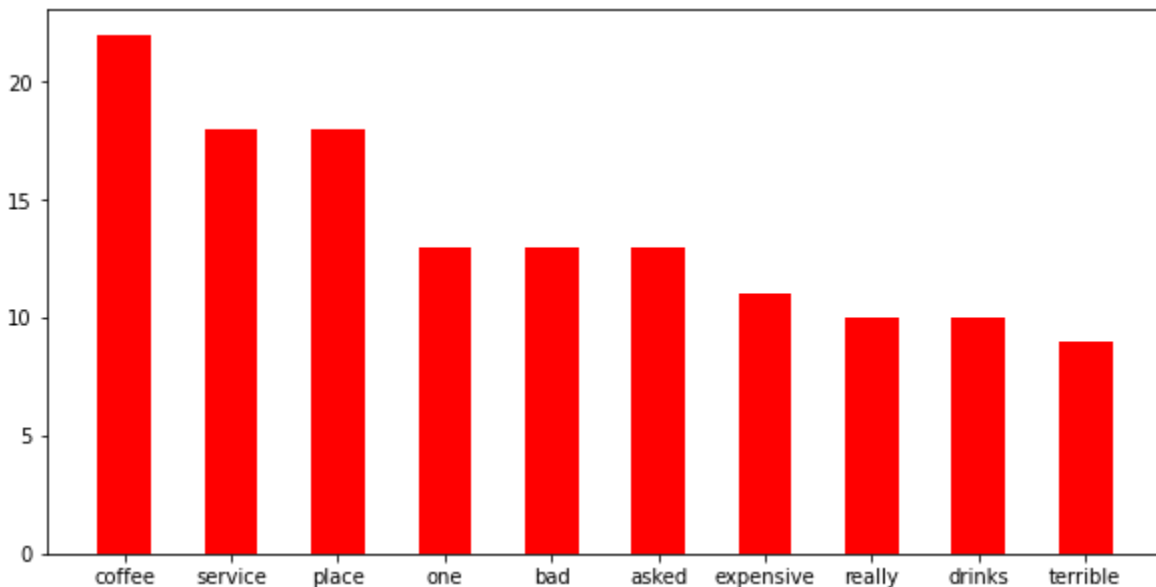
On the X axis we have the most popular words and on the Y axis the times those words occurred.

We see that the most common words for positive reviews are “place” and “coffee” which is something expected. That’s because we expect the reviews to characterize the place, the product and the service in some way (we also see the words “staff”, “service” and “thessaloniki” related to the service and place). In our case the product is coffee and drinks because we scraped data for bars and coffees.

We also see words with positive meaning like good, nice, great and friendly, as we would expect from positive reviews. These words characterize the place, the product and the service in a positive way.



Most common words among negative reviews

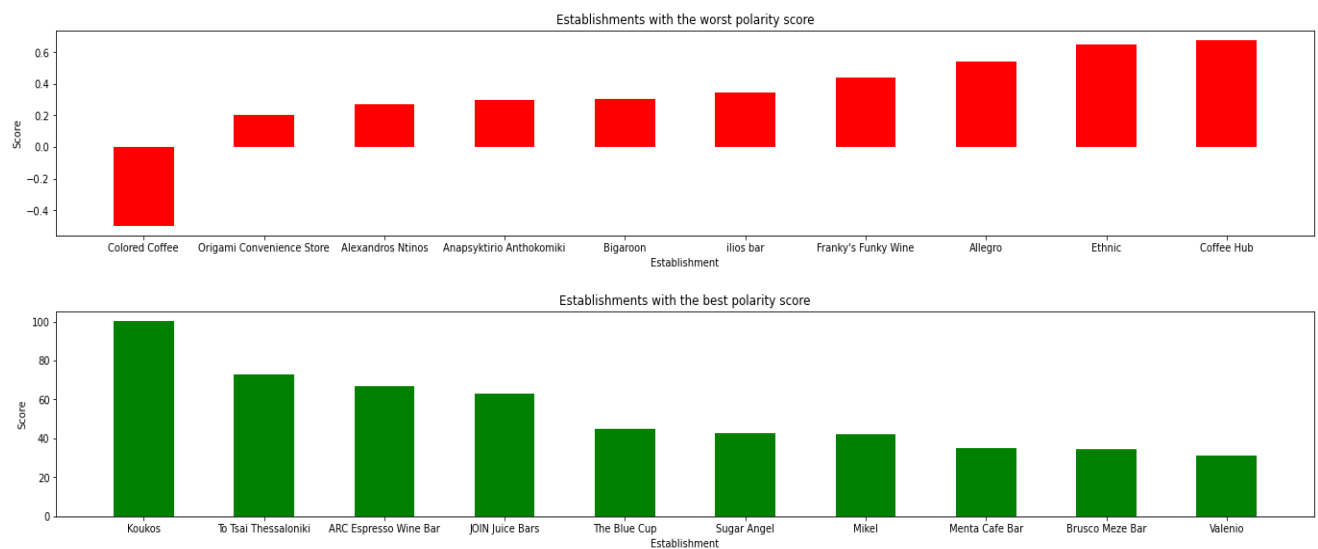


On the X axis we have the most popular words and on the Y axis the times those words occurred.

As with the positive reviews, we see that the most common words for negative reviews are coffee, service and place. We also see negative words like bad, expensive and terrible as we would expect from negative reviews.

[Check section 8](#) for the code related to the most common positive and negative words.

Polarity score of most popular and unpopular establishments



On the X axis we see the name of the most popular/unpopular establishments and on the Y axis the total sentiment score which is the sum of all the sentiment scores of all the establishment's reviews. Note that we do not calculate the mean sentiment because neutral reviews (with 0

polarity score) would negatively impact the sentiment of an establishment with positive overall sentiment and positively the sentiment of an establishment with negative overall sentiment. We want neutral reviews to not impact the sentiment at all, which is the case when we sum all the polarity scores of all an establishment's reviews.

On the actual TripAdvisor page, we have the following most popular establishment (from highest to lowest rating - we mark with green color those that are matching with our top 10 findings):

1. **To Tsai Thessaloniki**
2. **Koukos**
3. **Judah Club**
4. **Menta Cafe Bar**
5. **Valenio**
6. **The Blue Cup**
7. **JOIN Juice Bars**
8. **Sugar Angel**
9. **ARC Espresso Wine Bar**
10. **The Pub**

We managed to find 8 out of the top 10 TripAdvisor establishments. The two establishments that were not found are still on the first page of TripAdvisor but they are listed lower.

The findings are based on the text content of the review and not the actual rating of the reviewer. This proves that the sentiment of the text and rating of the review are correlated as we would expect.

[Check section 9](#) for the code implementation.

User profiling

For the user profiling, we are going to use the reviewer's gender as the attribute of our choice. That means that as a label and what we are going to predict will be the reviewer's gender. Therefore, we will use a subset of the original dataset, and more specifically, the subset with information about the reviewer's demographics.

Before any preprocessing, the dataset that contains all required information are 419 records.

As a next step, we are going to deal with the missing values. We chose to separate the features into 2 categories.

- The first category will include the features : age and distribution. When we find missing values on these features we will drop them because we can not make an estimation regarding the age of the data or the distribution of their votes.
- The second category will include the features : contributions, cities_visited, helpful_votes, photo. When we find missing values on these features, we will fill them with the mean and round the result, because for example a user can not have visited 22,5 cities. After testing our models including these features and not including them, the results would show us that the model performance is better when we include them.

```
# Dealing with missing values
```

```
updated_df = dataf.dropna(subset=['age'])
updated_df = updated_df.reset_index(drop= True)
```

```
updated_df['contributions']=updated_df['contributions'].fillna(updated_df['
contributions'].mean())
updated_df['contributions'] = updated_df['contributions'].apply(lambda x:
round(x))
```

```
updated_df['cities_visited']=updated_df['cities_visited'].fillna(updated_df
['cities_visited'].mean())
updated_df['cities_visited'] = updated_df['cities_visited'].apply(lambda x:
round(x))
```

```
updated_df['helpful_votes']=updated_df['helpful_votes'].fillna(updated_df['
helpful_votes'].mean())
updated_df['helpful_votes'] = updated_df['helpful_votes'].apply(lambda x:
round(x))
```

```
updated_df['photo']=updated_df['photo'].fillna(updated_df['photo'].mean())
updated_df['photo'] = updated_df['photo'].apply(lambda x: round(x))
```

```
updated_df.dropna(subset=['distribution'], inplace=True)
```

```
updated_df = updated_df.reset_index(drop= True)
```

After the preprocessing of missing values, the final length of the dataset is 309 records.

Next, we noticed that the feature distribution is included in the dataframe as a dictionary and we could not directly use it for our models. So, we included in our preprocessing steps the management of this feature by vectorizing it.

```
# Dealing with Distributions
```

```
from sklearn.feature_extraction import DictVectorizer  
vec = DictVectorizer()
```

```
updated_df['distribution'].head()  
vec_df = vec.fit_transform(updated_df['distribution']).toarray()
```

```
# creating a list of column names  
column_values = vec.get_feature_names()
```

```
vec_df = pd.DataFrame(data = vec_df, columns= column_values)
```

```
updated_df = updated_df.join(vec_df).drop(columns=['distribution'])
```

Afterwards, for the feature engineering part, we created some features based on the linguistic structure of the given texts. More specifically, we created features like the punctuation count, capitals count, number of exclamation marks, questions marks, number of punctuation and number of emojis.

```
# Define a list of commonly found punctuations
```

```
punctuation = ('!', ", " , "\' " , "; " , "\" " , ". " , "- " , "? " )
```

```
# Define a list of commonly found emojis
```

```
emojis = ('🤔', '👍', '😭', '🙏', '😓', '😄', '😍', '😘', '😊', '😁')
```

```
updated_df['punc_count'] = updated_df['text'].apply(lambda x : len([a for a  
in x if a in punctuation]))
```

```
updated_df['capitals'] = updated_df['text'].apply(lambda comment: sum(1 for  
c in comment if c.isupper()))
```

```
updated_df['num_exclamation_marks'] = updated_df['text'].apply(lambda x:  
x.count('!'))
```

```
updated_df['num_question_marks'] = updated_df['text'].apply(lambda x:
```

```
x.count('?'))
updated_df['num_punctuation'] = updated_df['text'].apply(lambda x:
sum(x.count(w) for w in punctuation))
updated_df['num_emojis'] = updated_df['text'].apply(lambda x:
sum(x.count(w) for w in emojis))
```

Making use of these newly created features, we noticed relationships between them and the label we want to predict, increasing significantly the performance of the models.

Also, we calculated the polarity score of the processed text of the review using the TextBlob library and imported it into our dataframe for each record.

```
updated_df['polarity'] = updated_df['Review Text'].map(lambda text:
TextBlob(text).sentiment.polarity)
```

In addition, we created a new feature called sentiment, based on the user's ratings. We separated the values into 3 classes.

- Positive when the rating was 5-star
- Neutral when the rating was 4-star
- Negative when the rating was 3-star or less.

```
neutral_range = {"low": 40, "high": 50}
updated_df["Sentiment"] = "neutral"
updated_df["Sentiment"].loc[df["review_rating"].astype(int) <
neutral_range["low"]] = "negative"
updated_df["Sentiment"].loc[df["review_rating"].astype(int) >=
neutral_range["high"]] = "positive"
updated_df['Sentiment'].value_counts()
```

Also as the final step of our preprocessing, we used the one-hot encoding to convert the categorical data to numerical data and more specifically on the features age and sentiment.

```
# We will use one-hot-encoding to convert categorical data to numerical
data

categorical_data_cols = ['age', 'Sentiment']

onehot_encoder = OneHotEncoder()

onehot_encoder_df =
pd.DataFrame(onehot_encoder.fit_transform(updated_df[categorical_data_cols]
).toarray())
```

```
updated_df =
updated_df.join(onehot_encoder_df).drop(columns=categorical_data_cols)
```

We used the standard 80%-20% split between the train and test data and also used cross validation for the training.

We tried different classifiers such as Logistic Regression, SVM, Neural Net, Decision Tree and Random Forest.

For the evaluation of the performance of our models we used the p_score.

```
# Performance
p_score = lambda model, score: print('Performance of the %s model is
%0.2f%%' % (model, score * 100))
```

In the following figure, we see the results and more specifically the minimum score, max score and mean score for each classifier.

	Name	Min Score	Max Score	Mean Score
2	Linear SVM	0.516129	0.548387	0.532258
0	Logistic Regression	0.548387	0.629032	0.583871
4	Neural Net	0.661290	0.758065	0.729032
1	Logistic Regression with Polynomial Hypotheses	0.838710	0.935484	0.896774
3	RBF SVM	0.822581	0.951613	0.906452
6	Random Forest	0.870968	0.967742	0.938710
5	Decision Tree	0.903226	0.967742	0.941935

For our classification problem, the Random Forest and Decision Trees are performing the best. So, in order to answer the following questions we will use the model using the Decision Tree.

Question 1 - Calculate the percentage of each class (e.g., male vs. female or age groups) in your training data users and in the whole dataset (after you use your model for prediction).

In the following figure, we see the percentage of each class (men and women) in our training dataset and in the dataset after the prediction. As expected if we take into consideration the high performance of our selected model, the results of the train and predicted are very close together. That means that we have a good prediction on our classification problem, having results close to the training dataset.

Train

```
# Percentage of each class in the training data users  
y_train.value_counts(normalize=True)
```

✓ 0.1s

```
man      0.554656  
woman    0.445344  
Name: sex, dtype: float64
```

Predicted

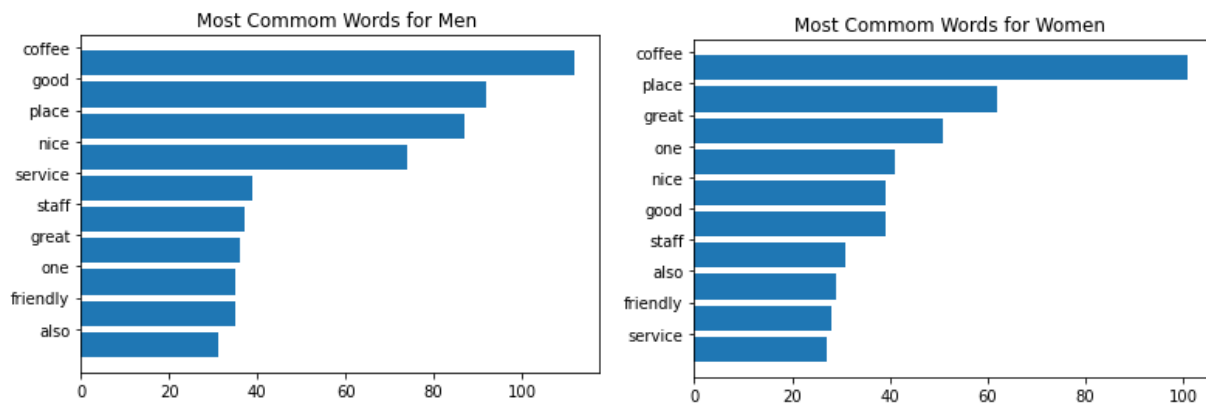
```
#y_pred  
y_pred[0].value_counts(normalize=True)
```

✓ 0.5s

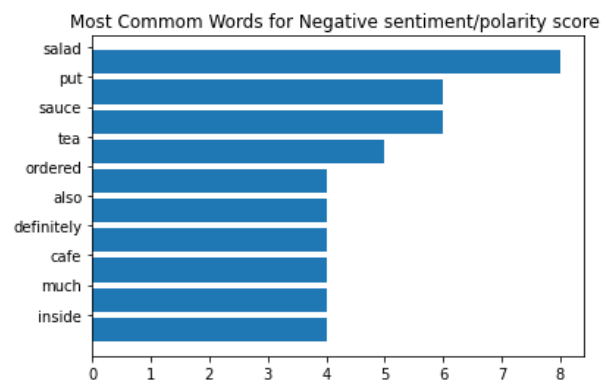
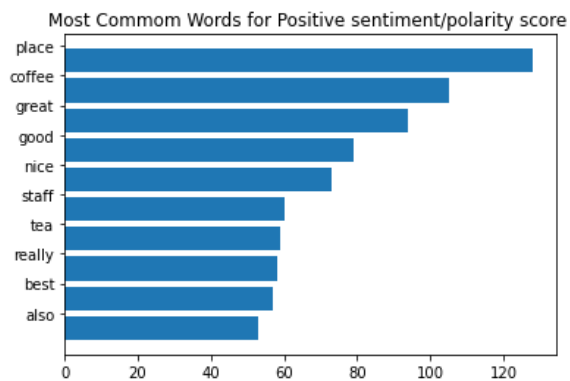
```
man      0.564516  
woman    0.435484  
Name: 0, dtype: float64
```

Question 2 - Visualize the most common words per class (e.g., male vs. female or age groups) and sentiment/polarity (negative vs. positive) in a bar chart or word cloud.

In the following figure, we see the most common word per class (men and women). As expected from the topic that we have analyzed, the most common words are coffee, place, service etc. Although, there are no major differences in the most common words between the two user classes.

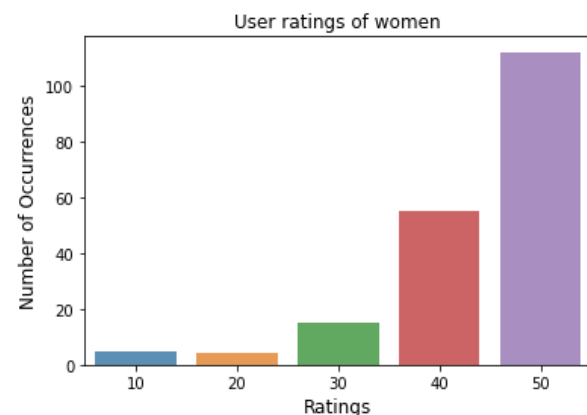
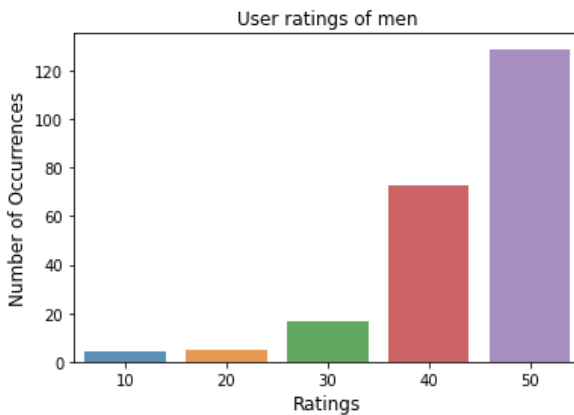


In the following figure, we see the most common word for positive vs negative polarity score/sentiment. As expected the most common words for the positive polarity are words like great, good, nice etc. In comparison though to the negative sentiment, due to the lack of efficient data, the words are not easily associated with a specific sentiment.

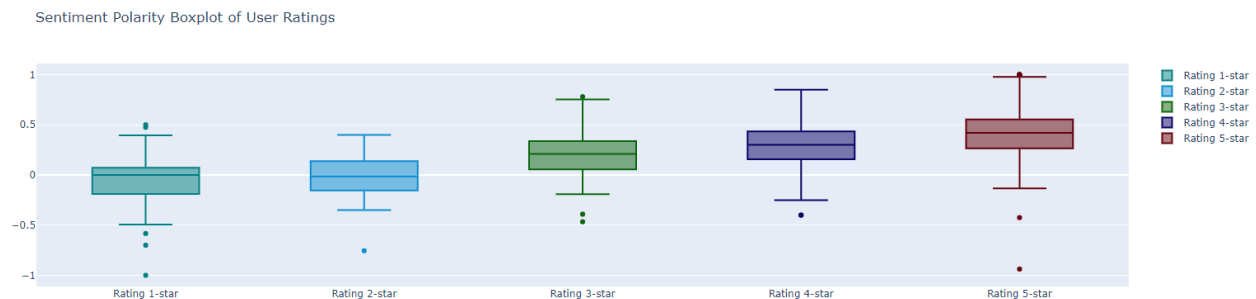


Question 3 - Create a box plot of user ratings visualizing the differences in ratings between user classes. Also, create a similar box plot for sentiment/polarity scores.

In the following figure, we see the user ratings word per class (men and women). There are no major differences between the two user classes and as a result both men and women have approximately the same preferences based on the user ratings.



Finally, in the following boxplot we can see the differences in rating given the polarity score. As expected, the higher the rating the higher the polarity score. That can be interpreted as the users that in their review text have positive words, it is reflected also to the user rating.



Conclusions

I hope you enjoyed this tutorial!

You should now have a good understanding of how to install and set up Selenium and how to scrape the desired information by combining Selenium with Python. Also, an interesting conclusion from our research is that Selenium is an excellent tool to automate almost anything on the web like performing repetitive tasks on a site.

Furthermore, by analyzing the scraped textual data we unveiled further knowledge and produced results on our topic. Most specifically, after preprocessing the textual data, we produced some very interesting visualizations which gave us some deep understanding of them. Also, using TextBlob we made sentiment analysis on the review texts and produced results very close to the ones that appear on the TripAdvisor website. In addition, we trained several machine learning models trying to predict the user's gender and finally used the one with the higher performance in order to produce the results on the questions.

The above-scraped dataset can be found here:

https://github.com/MnAppsNet/Web-Scraping-Tutorial/blob/main/scraped_data.csv

The jupyter notebooks used for scraping and data analysis can be found here:

<https://github.com/MnAppsNet/Web-Scraping-Tutorial>

I hope you found this information useful and thanks for reading!

Authors

1. Chatziara Dimitra, 98*
2. Kalyvas Emmanouil, 93*
3. Grigoroudis Efstratios, 74*

** All authors contributed equally to this tutorial*