Dipartimento di Informatica, Sistemistica e Comunicazione
Master Degree - Data Science

Data scientists in Italy

Candidates & roles:
Francesco Angiulli 897415
Data acquisition, Data cleaning, Data enrichment
Federico Ferretti 897902
Data acquisition, Data enrichment, Results from the research
Alice Parodi 898014
Data acquisition, Data cleaning, Data quality

ACADEMIC YEAR 2022/2023

Contents

Chapter 1

Introduction

During the university degree, the person still conceives himself as a student, and the feeling and need for autonomy do not emerge prominently. At the end of the studies, there is a clear perception of the exit from the role of students and the desire to establish themselves with a new role. Such a transition can lead to a feeling of bewilderment, disorientation and emptiness. The disorientation is determined by the change of equilibrium and habits. For a lot of years, he/she has a routine given by university lessons, exam periods and a full time autonomy: each of these aspects will change in the business world.

This fear is defeat or at least reduced if the outgoing student has already idea of the job market and can decide which is the best solution for his/her career. In order to do so, he/she has to understand which kind of field is the preferred one and search for positions.

The point is that usually we look only for the work position and the relative company without thinking about the geographical position of the new employment. Even if we usually don't think about it, it has a great influence on our general satisfaction. Moreover, there are a lot of studies about the fact that investing in employee happiness means investing in the company, because satisfied and happy workers are more productive and creative.

The WHO (World Health Organization) states that "Quality of Life is an individual's perception of their position in life in the context of the culture and value systems in which they live and in relation to their goals, expectations, standards and concerns" [1]. There are a lot of indicators to measure it, for example the Numbeo's Quality of Life Index, which measures eight indices: purchasing power (including rent), safety, health care, cost of living, property price to income ratio, traffic commute time, pollution, and climate [2].

Our research has the focus in Italy and to measure the quality of life we used data acquired from IlSole24Ore, from the 33rd investigation on the Quality of Life across the provinces in Italy [3]. The data are about the past year, 2022 and they reflect the setbacks for Covid19, the Ukrainian war, the inflation and the cost of electricity. In particular we will see in the chapter 6 that the crisis hit mostly the south of Italy, intensifying the distance with the other part of the country. The first symptoms of a population in shock from the rise of prices: households remain crushed under the weight of inflation never so high since the early 1980s, high energy prices are hitting businesses and local governments.

The goal of our research is to make a rank of jobs not only according to the company, but also according to the geographical allocation of the office. For answering, we will extract data from Glassdoor, a website where current and former employees anonymously review companies and where companies search for new workers. In particular we will focus on positions of data scientists across Italy. We will enrich this database with the indicators associated to the quality of the life related to all the Italian provinces. In order to combine the two databases, we use a third data source that matches cities, provinces and regions in Italy.

1.1 Questions of research

At the end of our research, we will be able to answer to the following question of interest:

Among the top firms across Italy, where a new employee should work according to the quality of life of that cities?

Moreover, studying the data we will also focus on secondary questions:

1. Are the job offers homogeneously distributed between the south and the north of Italy?

2. Rome and Milan are the springboards for a new data scientist, let's make a comparison between the two cities. Where is it better to move, taking into account the indices? Which are the best indices for Milan? And which for Rome? What about the top firms?

Chapter 2

Data acquisition

2.1 Scraping with Selenium Glassdoor

The acquisition of data began with the scraping of the Glassdoor site. The page of interest is constructed in the following way (see 2.1).

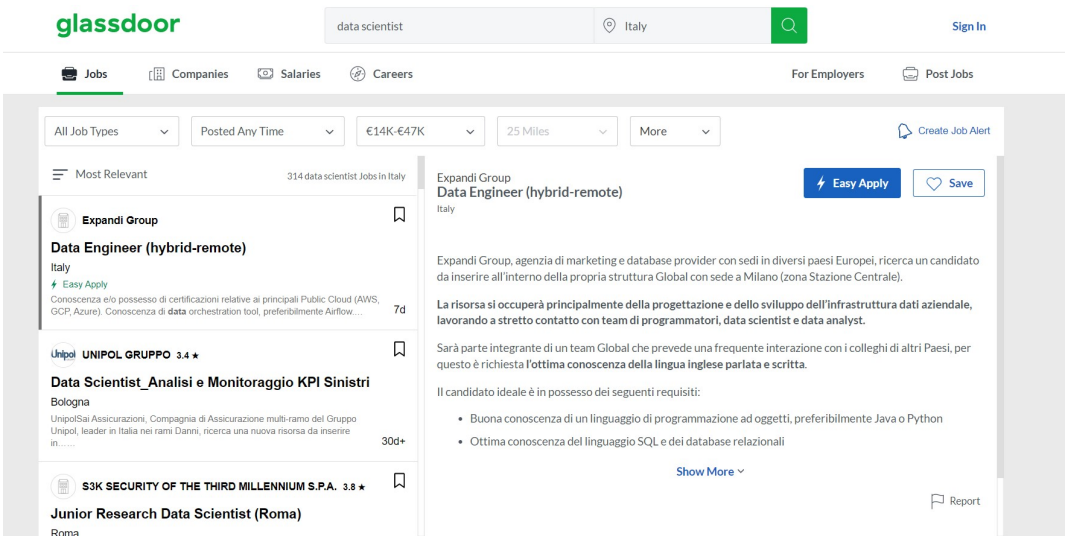


Figure 2.1: Glassdoor website

This part is a code description: each step of the code is described.

```
from selenium.webdriver import Chrome
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
chrome_driver=ChromeDriverManager().install()
from time import sleep

#to resolve HTTP 403 response
from urllib.request import urlopen, Request

from bs4 import BeautifulSoup
import requests
import re
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

Figure 2.2: Libraries

Before scraping the glassdoor site, we need to import several libraries and set driver (see figure 2.2).

1. Library imports:

   - "selenium.webdriver": Selenium main module for browser automation.
   - "webdriver_manager.chrome": Library to manage automatic installation of Chrome driver.
   - "selenium.webdriver.chrome.service": Module to manage Chrome Driver service.
   - "selenium.webdriver.common.by": Module to specify search methods for elements in web pages.
   - 'selenium.webdriver.chrome.options': Module to configure Chrome browser options.
   - 'urllib.request': Module to handle HTTP requests.
   - 'bs4': BeautifulSoup library for web scraping.
   - "requests": Library for making HTTP requests.
   - "re": Regular expression support module.
   - "pandas": Data manipulation library.

2. Chrome driver setup "chrome_driver=ChromeDriverManager().install()": Use 'ChromeDriverManager' to automatically install the Chrome driver needed for browser automation.

3. Imports and additional configurations

   "from time import sleep": Import the 'sleep' function from the 'time' module. It is used to insert delays in the code in order to simulate human's behaviour.

   "from urllib.request import urlopen, Request": Import the 'urlopen' and 'Request' classes from the 'urllib' and 'request' modules. They are used to handle HTTP requests.

4. Other Selenium and BeautifulSoup specific class and module imports:

   "from selenium.webdriver import Chrome" and "from selenium import webdriver": Base class imports for browser automation with Selenium.

   "from bs4 import BeautifulSoup": Import the 'BeautifulSoup' class to parse HTML markup.

```
In [ ]: s=Service('/Downloads/chromedriver_mac64')
driver=Chrome(service=Service(chrome_driver))

#to resolve HTTP 403 response
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.3'}
get_url='https://www.glassdoor.com/Job/italy-data-scientist-jobs-SRCH_IL.0,5_IN120_KO6,20.htm?locKeyword=Italy&srs=RECENT_SEARCHE
req = Request(url=get_url, headers=headers)

html1 = BeautifulSoup(urlopen(req).read())

#Number of page to scraper

num_page=int(html1.find('div', class_='paginationFooter').get_text('div').split(' ')[-1])

Ad_list = []
lp=[]
list_url=[]
dataFinal=[]
list_detail=[]
loc_index=[] #indice per italy e altri region
name_loc=[] #name location
loc=0
firm=[]
for numero in range(num_page):
    req = Request(url=get_url, headers=headers)
    html = urlopen(req).read()
    html_tree = BeautifulSoup(html)
```

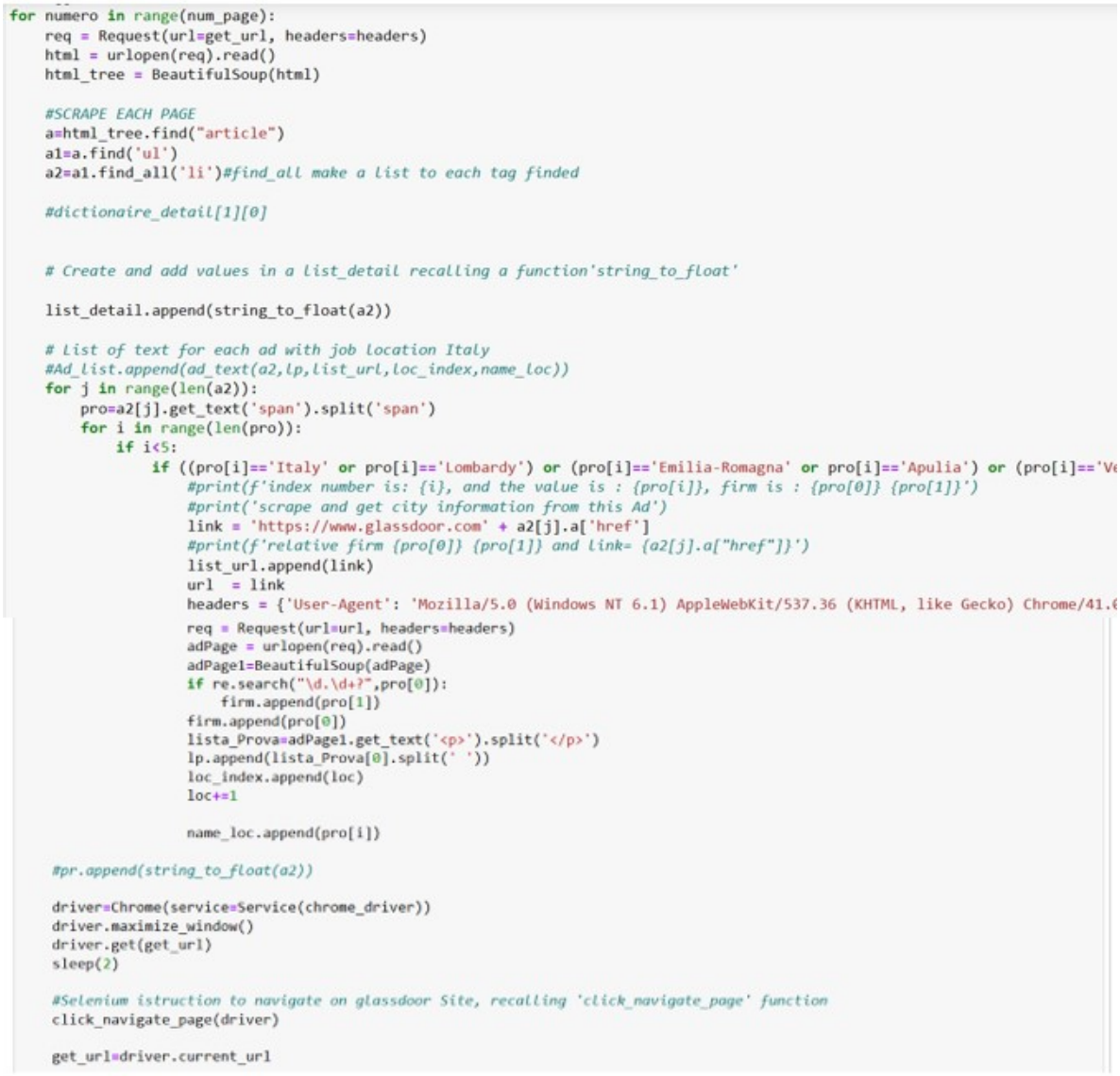
Figure 2.3: Chrome driver

The partial code above (figure 2.3), run the selenium driver initialization and configuration:

A 'Service' object is created using the Chrome driver path ("/Downloads/chromedriver_mac64"). Next, a 'Chrome' object is created using the created service. A header is set to resolve the HTTP 403 response. The URL of the site from which the data will be fetched is specified.

Moreover, to get the number of pages to analyze, a 'BeautifulSoup' object is created using the contents of the home page. The $< div >$ contains the information about layout and, indeed, the total number of pages from the div element is extracted. The value is converted to an integer using 'int()'.

Afterwards, a 'for' loop is created which iterates over the total number of pages. For each page, the following code flow is executed (see figure 2.4):



```
for numero in range(num_page):
    req = Request(url=get_url, headers=headers)
    html = urlopen(req).read()
    html_tree = BeautifulSoup(html)

    #SCRAPE EACH PAGE
    a=html_tree.find("article")
    a1=a.find('ul')
    a2=a1.find_all('li')#find_all make a list to each tag finded

    #dictionaire_detail[1][0]


    # Create and add values in a list_detail recalling a function'string_to_float'

    list_detail.append(string_to_float(a2))

    # list of text for each ad with job location Italy
    #Ad_list.append(ad_text(a2,lp,list_url,loc_index,name_loc))
    for j in range(len(a2)):
        pro=a2[j].get_text('span').split('span')
        for i in range(len(pro)):
            if i<5:
                if ((pro[i]=='Italy' or pro[i]=='Lombardy') or (pro[i]=='Emilia-Romagna' or pro[i]=='Apulia') or (pro[i]=='Ve
                    #print(f'index number is: {i}, and the value is : {pro[i]}, firm is : {pro[0]} {pro[1]}')
                    #print('scrape and get city information from this Ad')
                    link = 'https://www.glassdoor.com' + a2[j].a['href']
                    #print(f'relative firm {pro[0]} {pro[1]} and link= {a2[j].a["href"]}')
                    list_url.append(link)
                    url  = link
                    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.6
                    req = Request(url=url, headers=headers)
                    adPage = urlopen(req).read()
                    adPage1=BeautifulSoup(adPage)
                    if re.search("\d.\d+?",pro[0]):
                        firm.append(pro[1])
                    firm.append(pro[0])
                    lista_Prova=adPage1.get_text('<p>').split('</p>')
                    lp.append(lista_Prova[0].split(' '))
                    loc_index.append(loc)
                    loc+=1

                    name_loc.append(pro[i])

    #pr.append(string_to_float(a2))

    driver=Chrome(service=Service(chrome_driver))
    driver.maximize_window()
    driver.get(get_url)
    sleep(2)

    #Selenium istruction to navigate on glassdoor Site, recalling 'click_navigate_page' function
    click_navigate_page(driver)

    get_url=driver.current_url
```

Figure 2.4: Code for each page

An HTTP request is made using the ‘urllib.request‘ module to get the page content. The content is then parsed with BeautifulSoup to create a BeautifulSoup object. The $< article >$ tag is found which contains the job offers. As a consequence, also $< ul >$ element inside article and $< li >$ inside ul are found.

In order to extract information from advertisements a ‘for‘ loop is performed on all the ads. The ad text is extracted using ‘get_text($< span >$)‘ and split using span as a separator. Further operations are performed on the extracted data, such as extracting specific information (like Italy, Lombardy, Emilia-Romagna, etc.), creating full links for the ads and taking out more details from the ad pages. The extracted information is saved in several lists (‘lp‘, ‘list_url‘, ‘firm’, ‘lista_Prova‘, ‘loc_index‘, ‘name_loc‘). In the lp list, there is the list of the text of each advertisement found in the website. For some advertisement, the job location is not present in the preview. For this reason, we need to go through all the advertisements in order to search in the actual text, the information about the geographical location. To do so, we downloaded a new csv

("Elenco-comuni-IT.csv"), to recognize easily the cities in the text. The code is shown in figure 2.5:

```
lista_comuni=pd.read_csv("Elenco-comuni-IT.csv")
```

```
citta=[]
indice=[]
for z in range(len(lp)):
    for i in lista_com:
        for j in lp[z]:
            if(i==j):
                citta.append(i)
                indice.append(z)
                break
    if z not in indice:
        citta.append("smart working")
        indice.append(z)
```

Figure 2.5: Search for the city inside some advertisement

Lastly the completion of navigation through Selenium: A new Chrome driver is initialized using the service created earlier. The browser window is maximized and the URL of the current page is opened using 'get_url'. The 'click_navigate_page' function is called to perform some actions on the website which we will describe later. The current browser URL is obtained using 'driver.current_url'.

Then we implemented a function called click_navigate_page, in order to try to solve some issues occurred during the scraping process, see figure 2.6.

```
def click_navigate_page(x):
    x.find_element(by=By.ID, value="onetrust-accept-btn-handler").click()
    sleep(2)

    try:
        x.find_element(by=By.XPATH, value="//span[@class='SVGInline modal_closeIcon']").click()
        sleep(2)
    except:
        x.find_element(by=By.XPATH, value="//button[@class='nextButton job-search-1iiwzeb e13qs2072']").click()
        sleep(2)
    #nextButton job-search-1iiwzeb e13qs2072 new

   # nextButton css-1hq9k8 e13qs2071
    try:
        x.find_element(by=By.XPATH, value="//button[@class='nextButton job-search-1iiwzeb e13qs2072']").click()
        sleep(2)
        prova=1
    except:
        #driver.find_element(by=By.XPATH, value="//button[@class='nextButton css-1hq9k8 e13qs2071']").click()
        prova=2
```

Figure 2.6: Function for navigation in the page

1. Acceptance of cookies: The function searches for an item on the page with the ID "onetrust-accept-btn-handler" and clicks on it. This represents a button or element that requires consent for the use of cookies. After the click, the code pauses for 2 seconds using the 'sleep(2)' function.

2. Management of a possible signup invitation pop-up dialog on the site: Using a try-except block, the code looks for an element in the page using the XPath expression "//span[@class='SVGInline modal_closeIcon']". If the item is found, the code clicks on it. This represents a dialog that is closed by clicking on a close icon. After the click, the code pauses for 2 seconds using the 'sleep(2)' function. If the element is not found, the code jumps to the 'except' block.

3. Click on the button to move to the next page: In the ‘except‘ block, the code searches for an item on the page using the XPath expression ”//button[@class=’nextButton job-search-1iiwzeb e13qs2072’]”. This element is a button to go to next page. If the item is found, the code clicks on it. This could be a button to advance to the next page or perform some other related action. After the click, the code pauses for 2 seconds using the ‘sleep(2)‘ function.

```
##CREATE A DICTIONAIRE WHICH WILL BE THE SKELETON FOR THE DB
dictionary_detail={}

g=0
for k in range(len(list_detail)):
    for j in range(len(list_detail[k])):
        dictionary_detail[g]={'firmScore':list_detail[k][j][0], 'firm':list_detail[k][j][1],'ad_Firm_Role':list_detail[k][j][2],'
        g+=1
```

Figure 2.7: Creation of the dictionary

In this section of the code (figure 2.7), a dictionary called ‘dictionary_detail‘ is created which will contain detailed information about the previously extracted job postings.

The process occurs through a double ‘for‘ loop. The first ‘for‘ loop iterates over the ‘list_detail‘ list, which contains the details pulled from the job postings. The second ‘for‘ loop iterates over the ‘list_detail[k]‘ list, which represents the details of a single job posting.

Within the ‘for‘ loop, the precise and true job posting values are assigned to the ‘dictionary_detail‘ dictionary. The values are extracted from the ‘list_detail[k][j]‘ list, where ‘k‘ represents the index of the item in the main list and ‘j‘ represents the index of the item in the sub-list.

The extracted details are assigned as dictionary values ‘dictionary_detail[g]‘, where ‘g‘ is an index that is incremented on each iteration to create a unique key for each item in the dictionary. Specific details are assigned to dictionary keys as follows:

- ’firm_Score’: the score of the company.
- ’firm’: the company name.
- ’ad_Firm_Role’: the specific role in the job posting.
- ’Job_Location’: the location of the job.
- ’n’: a numerical value associated with the job posting.

At the end of the cycle, the ‘dictionary_detail‘ dictionary will contain all the detailed information extracted from the job postings, with a unique key associated with each posting.

```
column_name=['firm_Score','Firm','n','ad_Firm_Role','Job_Location']
#delete_info(dictionary_detail)
data=pd.DataFrame(dictionary_detail)
data1=data.transpose()
data1.columns=column_name
Tabella1 = data1
Tabella1
```

Figure 2.8: Construction of the final data set

In this last section of code (see 2.8), some variables are defined and a data frame is created using the pandas library.

The variable ‘column_name‘ is a list of strings representing the column names. The values in the list correspond to the field names extracted from the job postings: ‘’firm_Score’‘, ‘’Firm’‘, ‘’n’‘, ‘’ad_Firm_Role’‘ and ‘’Job_Location’‘.

Next, a DataFrame called ‘data‘ is created using the ‘dictionary_detail‘ dictionary. This DataFrame is transposed using the ‘transpose()‘ function and assigned to the variable ‘data1‘.

Finally, the ‘column_name‘ list is assigned as column names of the ‘data1‘ DataFrame using the ‘columns‘ attribute. The DataFrame ‘data1‘ is then assigned to the variable ‘Table1‘.

In this way, the variable ‘Table1‘ will represent an organized data frame with the detailed information extracted from the job advertisements, with the columns corresponding to the names defined in ‘column_name‘.

2.2 Scraping Sole24Ore

Starting from the first data base coming from Glassdoor, our focus is now to find ranks about the quality of life in the provinces in Italy. To perform this analysis, we extract data through a process called scraping. This because the specific system, for us Sole24Ore, doesn’t have API.

We extract data from ranks about six different indicators: Cultura e tempo libero, Ambiente e servizi, Demografia e società, Giustizia e sicurezza, Affari e lavoro, Ricchezza e consumi. At the end, we combine all of them in a general, unique value as an arithmetic mean of them. The picture 2.9 shows an example of the first rank.

Figure 2.9: Sole24Ore's rank about culture and free time

Web scraper can be done through KNIME or Python. In this case, as before, Python is used and, in particular, the library Beautifulsoap. It is essential to know how a website is constructed in order to be able to identify which section of it is important for our goal and extract it.

The first step is "translating" the webpage into an html form. After this, we search for the score and the relative cities through the use of the regular expressions, a sequence of characters that forms a search pattern. They can be used to check if a string contains a specified search pattern, as shown in the figure 2.10.

Figure 2.10: Use of regex to extract the score and the name of the associated city

Thanks to the library Pandas, a final data frame is created with the rank of each indicator, as shown below in figure 2.11.

```
In [7]: import pandas as pd
        Cultura_e_tempolibero= pd.DataFrame({'Nome':lista2_nome, 'Cultura_e_tempolibero':lista2_punteggio})
        Cultura_e_tempolibero
```

Out[7]:

	Nome	Cultura_e_tempolibero
0	Firenze	516.5
1	Trieste	510.1
2	Gorizia	501.7
3	Siena	501.6
4	Milano	491.9
...	...	...
102	Foggia	210.5
103	Caltanissetta	200.3
104	Reggio Calabria	193.9
105	Vibo Valentia	193.7
106	Crotone	140.7

107 rows × 2 columns

Figure 2.11: The dataframe related to "Cultura e tempo libero"

The same steps are followed for the other five indicators: we have six different dataframe that are now merged together computing the mean for each city and ordering them in a descendent order. Bologna seems to have the highest quality of life according to that indicators, followed by Bolzano and Firenze (figure 2.12).

```
In [38]: together4= together4.sort_values("Average_punt", ascending=False)
         together4
```

Out[38]:

	Nome	Cultura_e_tempolibero	Affari_e_lavoro	Ricchezza_e_consumi	Giustizia_e_sicurezza	Demografia_e_società	Ambiente_e_servizi	Average_pun
9	Bologna	470.1	545.6	688.8	629.9	688.1	518.977	590.24616
20	Bolzano	425.8	493.2	677.0	762.1	603.8	552.341	585.70683
0	Firenze	516.5	537.3	621.9	648.8	622.7	543.792	581.83200
3	Siena	501.6	445.6	609.1	733.4	616.8	564.523	578.50383
13	Trento	448.5	519.3	662.3	701.1	591.8	536.632	576.60533
...	...	...	...	...	...	...	...	...
105	Vibo Valentia	193.7	447.8	373.2	620.8	397.0	402.737	405.87283
102	Foggia	210.5	374.8	409.1	591.5	446.3	372.888	400.84800
103	Caltanissetta	200.3	371.6	398.6	699.2	356.9	347.051	395.60850
98	Isernia	222.0	358.8	501.7	524.5	431.1	308.834	391.15566
106	Crotone	140.7	433.3	312.0	692.8	378.6	353.322	385.12033

Figure 2.12: Final dataframe about the quality of life

2.3 API Open-Cage

An API, Application Programming Interface, is a secure interface that allows everyone to request data from a data provider. For each city in the list of the cities extracted from Glassdoor, we use the function GET to require data from the web site Open-Cage. Then we created two lists: the first with the cities from Glassdoor and the other with the values in "county" referred to that specific city that is exactly the respective province, as shown in figure 2.13.

```
In [8]: Citta=[]
        Provincia=[]
        for i in l2:
            base_url = f"https://api.opencagedata.com/geocode/v1/json?q={i}&key=a5b0d680bd0642a1b15239795b9a778f"
            response = requests.get(base_url)
            e=response.json()
            b=e["results"]
            if b!=[]:
                Provincia.append(b[0]['components']['county'])
                Citta.append(i)
            print(Citta)
```

```
['Milano']
['Milano', 'Turin']
['Milano', 'Turin', 'Rho']
['Milano', 'Turin', 'Rho', 'Castellanza']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra', 'Vercelli']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra', 'Vercelli', 'Bari']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra', 'Vercelli', 'Bari', 'Torino']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra', 'Vercelli', 'Bari', 'Torino', 'Rozzano']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra', 'Vercelli', 'Bari', 'Torino', 'Rozzano', 'Rimini']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra', 'Vercelli', 'Bari', 'Torino', 'Rozzano', 'Rimini', 'Monsel
ice']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra', 'Vercelli', 'Bari', 'Torino', 'Rozzano', 'Rimini', 'Monsel
ice', 'Castel Maggiore']
['Milano', 'Turin', 'Rho', 'Castellanza', 'Rome', 'Roma', 'Ispra', 'Vercelli', 'Bari', 'Torino', 'Rozzano', 'Rimini', 'Monsel
ice', 'Castel Maggiore', 'Oristano']
```

Figure 2.13: Code for the creation of the two lists

At the end, we construct the data frame with the two columns above descripted (see figure 2.14).

```
In [10]: link= pd.DataFrame({'Città':Citta, 'Provincia':Provincia})
         link
```

Out[10]:

	Città	Provincia
0	Milano	Milano
1	Turin	Torino
2	Rho	Milano
3	Castellanza	Varese
4	Rome	Roma
5	Roma	Roma
6	Ispra	Varese
7	Vercelli	Vercelli
8	Bari	Bari
9	Torino	Torino
10	Rozzano	Milano
11	Rimini	Rimini
12	Monselice	Padova
13	Castel Maggiore	Bologna
14	Oristano	Aristanis/Oristano
15	Parma	Parma
16	Verona	Verona

Figure 2.14: Final data frame with the relation city-province

Chapter 3

Data cleaning

After acquiring the data, using the techniques extensively discussed above, we notice that the data extracted by scraping from the Glassdoor site are not at all neat and clean. After having extracted the parameters respectively in their belonging fields ("Firm_score", "Firm", "ad_Firm_role", "Job_Location"), the displayed records do not respect their belonging fields, since for some extracted records we have either more information or less information than we actually need.

The database of Glassdoor at first looked as follows in figure 3.1 (we visualize only the first and last rows):

	firm_Score	Firm	n	ad_Firm_Role	Job_Location
0	Expandi Group	KEY NOT FOUND: ctas.save	Data Engineer (hybrid-remote)	Italy	Easy Apply
1	INSIGHTUP SRL	KEY NOT FOUND: ctas.save	Data Engineer	Porto San Giorgio	€32K - €38K
2	3.3	Teamsystem	KEY NOT FOUND: ctas.save	Data Scientist - Milano/Torino/Pesaro/Campobasso	Italy
3	1.4	VJ Technology Srl	KEY NOT FOUND: ctas.save	DATA SCIENTIST	Telelavoro
4	EXAK Srl	KEY NOT FOUND: ctas.save	PHP web developer	Italy	€25K - €40K
...	...	...	...	...	...
295	4.1	GSK	KEY NOT FOUND: ctas.save	Clinical Science Lead (CSL) - Respiratory Vacc...	Italy
296	3.7	Alti Profili	KEY NOT FOUND: ctas.save	DATA SCIENTIST	Roma
297	3.9	ManpowerGroup	KEY NOT FOUND: ctas.save	Internship Industry 4.0 and Software Engineer	Zola Predosa
298	3.8	Istituto Italiano di Tecnologia	KEY NOT FOUND: ctas.save	Postdoc on Robot learning and control	Genoa
299	3.8	Istituto Italiano di Tecnologia	KEY NOT FOUND: ctas.save	Postdoc on Perception and Situational Awarenes...	Genoa

Figure 3.1: Glassdoor dataset before data cleaning

As we can see in the image 3.1, we have values that are not related to the "firm_Score" column that are actually in it, and the same happens for the other fields. For example, INSIGHTUP SRL is the name of a company that should be under Firm instead it is allocated in "firm_Score".

One peculiarity that emerges from this raw database is the middle column, called "n". We used this field as a passing field to then fix and clean up our data base. In fact "KEY NOT FOUND" is repeated for each row, but in different columns: it represents a data capture error for the HTML class used for scraping.

To go and clean our data and drop them all into the correct field, we made a data translation method for each column:

```
indexSTR=data1[data1['firm_Score'].str.isnumeric()==False].index
indexSTR=list(indexSTR) #index str values
dataSistemare=data1[data1['firm_Score'].str.isnumeric()==False]
```

```
dataSistemare['Job_Location']=dataSistemare['ad_Firm_Role']
dataSistemare['ad_Firm_Role']=dataSistemare['n']
dataSistemare['n']=dataSistemare['Firm']
dataSistemare['Firm']=dataSistemare['firm_Score']
```

```
dataSistemare['firm_Score']='Null'
dataSistemare.head(6)
```

Figure 3.2: Code for cleaning

- We identify the indices of each row that does not have the numeric value in "firm_Score".
- We extract these rows and create a sub-database with them.
- Starting from the last column, we shift the values of one column: all the values of "ad_Firm_Role" are now in "Job_Location" and so on for the remaining columns.
- The "firm_score" column has now the same values of the column "Firm", so we entered the value 'Null' because we don't have information about the score.
- The middle column 'n', now contains only the value "KEY NOT FOUND" for each of the row.

	firm_Score	Firm	n	ad_Firm_Role	Job_Location
2	Null	JP Mawel srl	KEY NOT FOUND: ctas.save	Junior Machine Learning Engineer	Nola
6	Null	Transcrime - Università Cattolica del Sacro Cuore	KEY NOT FOUND: ctas.save	Data Scientist	Italy
12	Null	Diskover	KEY NOT FOUND: ctas.save	Junior Data Scientist	Pescara
28	Null	HESPLORA	KEY NOT FOUND: ctas.save	Junior Data Scientist	Italy
32	Null	Alan Advantage	KEY NOT FOUND: ctas.save	Junior Data Scientist	Roma
33	Null	Terishield	KEY NOT FOUND: ctas.save	Data Scientist	Telelavoro

Figure 3.3: Final sub-dataset with no information about firm_Score

It is an example of what the data set looks like, partially cleaned.
Moreover, column 'n' is a field that :

- Is not useful for our study.
- Does not give us useful information.

```
data2=data1.drop(labels=indexSTR, axis=0)
data1New=pd.concat([data2, dataSistemare], ignore_index=True)
Glassdoor=data1New[['firm_Score','Firm','ad_Firm_Role','Job_Location']]
```

Figure 3.4: Concatenate the two datasets

Then, through the portion of code shown above (3.4), the drop() function removes the rows that we have cleaned from the starting dataset, without going to lose or alter data for our study. To wrap things up, we re-establish our database concatenating the sub-dataset without the rows to be cleaned and the dataset cleaned with the values all shifted to the right.

The figure 3.5 is the final result:

	firm_Score	Firm	ad_Firm_Role	Job_Location
0	3.4	Attitude	JUNIOR DATA ENGINEER	Roma
1	4.4	Chiesi Farmaceutici	Scientist	Parma
2	3.6	Generali Italia	Data scientist	Italy
3	4.1	Siemens	Machine Learning Engineer / Data Scientist (m/...	Italy
4	3.8	Euronext	Borsa Italiana - Associate, Junior Data Scientist	Italy
...	...	...	...	...
285	Null	RE COLLECTION SRL	Senior System Architect	Mangone
286	Null	RE COLLECTION SRL	Solution Architect	San Donato Milanese
287	Null	BIP - Business Integration Partners	Business Analyst	Italy
288	Null	Avanade	Data Scientist | Categorie Protette - Centre o...	Bologna
289	Null	Bespoke Recruitment	Medical Director - Physician - Neuroscience Cl...	Italy

Figure 3.5: Final dataset

Chapter 4

Data enrichment

Data preparation for the data integration phase is one of the most important tasks. In order to carry out the integrations in the best possible way, the strings must be 100% consistent with each other. Merging two strings is very complex as the words on the right (right data set) must be exactly the same of those on the left (left data set). It is difficult when there are special characters, or when one of the two strings is expressed in a different idiom (e.g. english) which still has the same meaning but written in a different way. In MySql terms, it is basically like applying a LEFT JOIN between the two data sets.

For our first data integration, we have to merge two data sets (Glassdoor and Comuni_Province_Regioni) to obtain, as a result, a data set that has, not only the information on each job advertisement, but also information on the provinces of the cities associated with each job advertisement.

The merging is mainly done using the following fields (figure 4.1):

- 'Job_Location': belong to Glassdoor data set;
- 'Città': belong to Comuni_Province data set;

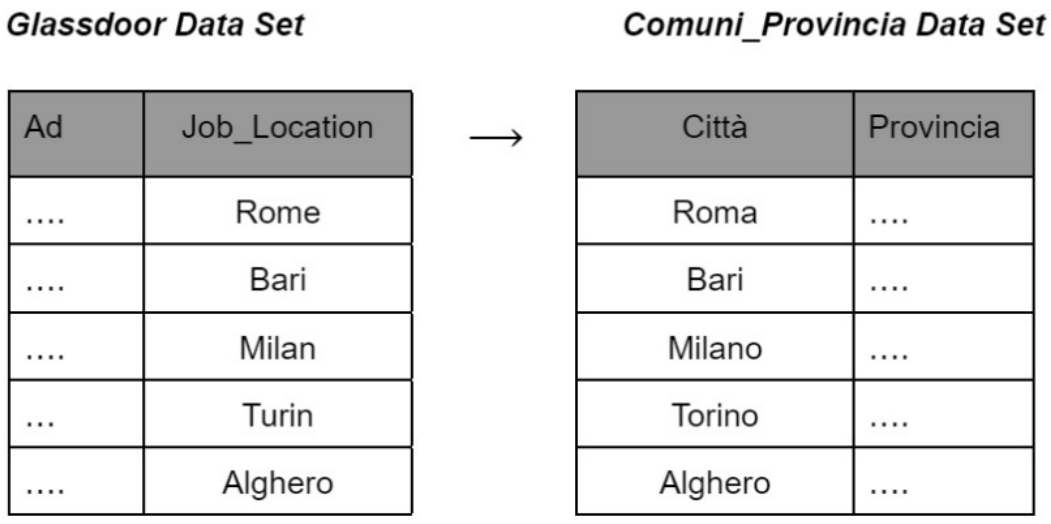


Figure 4.1: Keys for joining the two data sets

The problem encountered in this data merge concerns the language in which the names of the cities were acquired. As shown in the example above, when acquiring the data, added in the 'Job_Location' field, many of these cities were acquired in English. In contrast, all cities acquired via the API, are strings in Italian. So this caused a character conflict when merging two strings. To merge these strings, and overcome this conflict, we implemented an algorithm, which allowed us to detect the cities written in English and merge them with the respective cities written in Italian.

The final result of this integration, after implementation of the algorithm, was successful, and the result obtained is shown below in figure 4.2.

	firm_Score	Firm	ad_Firm_Role	Job_Location	Provincia	Regione
1	1.4	VJ Technology Srl	DATA SCIENTIST	Roma	Roma	Lazio
2	4.2	Newel Health	Junior Data Scientist	Remote	NaN	NaN
3	4.9	Agile Lab	Data Scientist II (Remote)	Turin	Torino	Piedmont
4	3.8	Wind Tre S.p.A. con Socio Unico	Data Scientist Junior	Rho	Milano	Lombardy
5	3.5	TENOVA	Data Scientist	Castellanza	Varese	Lombardy
6	3.8	Merkle	Junior Data Scientist	Remote	NaN	NaN
7	4.5	i-EM	Data Scientist	Rome	Roma	Lazio
8	3.5	Sferanet	Data Scientist Junior	Roma	Roma	Lazio
9	5	AMA European Consulting	Data Scientist (EU Joint Research Program)	Ispra	Varese	Lombardy

Figure 4.2: Table after the first join

From the image, it can be seen that in our data set, there are also entries with the value 'Remote' which is not associated with any province. This is because from the API acquisition of Italian municipalities, there is no municipality called 'Remote', because remote indicates work from home or from anywhere in the world. This case will be better explained in the next chapter 5 on Data quality.

As for our first integration, here we consider the data set obtained from the integration described above and we merge it with the data concerning lifestyle indices for each Italian province.

So we have the same scenario, two data sets to be merged, and this time the fields allowing the integration are (figure 4.3):

- 'Provincia': belong to previously integrated data set;
- 'Provincia': belong to Sole_24 Data set;

Glassdoor Data Set con Province

Ad	Job_Location	Provincia
....	Monselice	Padova
....	Rozzano	Milano
....	Castellanza	Milano
...	Moncalieri	Torino
....	Castel Maggiore	Bologna

→

Sole 24 Data Set

Provincia	Index
Padova	....
Milano	....
Milano	....
Torino	....
Bologna	....

Figure 4.3: Keys for joining the two data sets

Unlike the first integration, in this case we had no conflict and no inconsistency of records. No algorithm needed to be implemented. The final result of this integration was successful, and the result obtained is shown below, figure 4.4.

	firm_Score	Firm	ad_Firm_Role	Job_Location	Provincia	Regione	Cultura_e_tempolibero	Affari_e_lavoro	Ricchezza_e_consumi	Giustizia_e_sicurezz
1	1.4	VJ Technology Srl	DATA SCIENTIST	Roma	Roma	Lazio	421.1	566.4	597.5	532.
2	4.2	Newel Health	Junior Data Scientist	Remote	NaN	NaN	NaN	NaN	NaN	Na
3	4.9	Agile Lab	Data Scientist II (Remote)	Turin	Torino	Piedmont	393.6	492.1	648.1	639.
4	3.8	Wind Tre S.p.A. con Socio Unico	Data Scientist Junior	Rho	Milano	Lombardy	491.9	632.1	674.3	549.
5	3.5	TENOVA	Data Scientist	Castellanza	Varese	Lombardy	343.8	499.9	637.9	688.
6	3.8	Merkle	Junior Data Scientist	Remote	NaN	NaN	NaN	NaN	NaN	Na
7	4.5	i-EM	Data Scientist	Rome	Roma	Lazio	421.1	566.4	597.5	532.
8	3.5	Sferanet	Data Scientist Junior	Roma	Roma	Lazio	421.1	566.4	597.5	532.
9	5	AMA European Consulting	Data Scientist (EU Joint Research Program)	Ispra	Varese	Lombardy	343.8	499.9	637.9	688.

Figure 4.4: Final data set

It can be seen that the integration done previously was a transitional integration, and that the Comuni_Province_Regioni dataset is a bridging data set, to carry out the latter integration.

4.1 Relational database

To answer to our question of research, we clearly need a relation between firm and city. This is a many-to-many relation, because a firm can be in more than one city in Italy and a city can have multiple job offers. Based on this information we decided to use a relational database. The relational paradigm is the best option because relational

databases are the best at managing a finite number of many-to-many relationships thanks to an efficient indexing. Analyzing all possibilities, we confirmed by exclusion that our choice is correct:

- Key-Value, can handle the scaling of large amounts of data and an high lecture speed but here these are not required since the dimension of the dataset is quite small. Morever, the observations are all of the same type.
- Column Family: same reasons as above since the main difference is that column-family databases store data in key order, rather than by computing a hash but in general they are very similar among the NoSQL databases.
- Graph database: it's optimal for many data join, while here we speak about simple join between firms and geographical location so the advantages given by the model would be useless.
- Document database: it can be a good option. A RDBMS is focused on creating relationships between files to store and read data, while document databases are focused on the data itself and relationships are represented with nested data. Although SQL databases have great stability and vertical power, they struggle with super-sized databases. However, as stated above, we are working on a restrained number of observations, so we decided for the stability of the RDMS compared to the schema-less of the document based. However, in the last chapter 7, some further developments are explained also in this context.

Chapter 5

Data quality

Data quality is a measure of the condition of the data based on factors such as accuracy, completeness, currency and consistency. It is essential in particular for data-driven companies that base their decisions on the data analysis to meet their business goals. If the data are not clean, for example presence of outliers or missing data, the data can lead to wrong decisions that cause negative business outcomes. According to some studies, poor data quality highly influences the loss in costs organizations.

As an old saying goes, "gargabe in, garbage out", and this is true for example for machine learning procedures that, if they learn from bad data, we will have inaccurate results. In order to understand if quality is enough, trade-offs are needed to diagnose data issues.

5.1 Accuracy

Accuracy is defined as the closeness between the real world and the representation in the dataset. Extracting data from Glassdoor, there are companies that uses italian city names such as Milano and some others that keep the english name, Milan for example. The syntactic accuracy is defined when the vocabulary or reference domain of values is known and is the degree to which values correctly represent the domain values of underlying vocabularies or reference domains. In this case, the cities should be in Italian languange in order to be recognized as key by the data frame of cities-provinces.

We both know the domain in italian and english, so it can be easily modified, through the use of an algorithm, from english to italian. For each row, if we find "Milan" change with "Milano" or if "Florence" is written, change it into "Firenze" for example. We are 100% sure that the translation doesn't present bias since we have both the domains and the translations.

5.2 Completeness

Completeness is the coverage with which the observed phenomenon is represented in the data set. There are four kinds of completeness:

- Tuple completeness, there are 36 observations that in "Job_Location" have "Remote". This is due to the fact that there is no information about the location of

that jobs. As a consequence, after the enrichment with the other two data frames, 8 columns from "Provincia" to the end have "Null" value. In fact, API doesn't find the province for the city "Remote" so it leaves the column with the null value and, consequently, also the columns related to the quality of life for the provinces are empty. Mathematically, for these tuples we have a tuple completeness of $\frac{4}{12} = 0.33$.

Moreover, there are 40 tuples that don't have the score of the firm. In this case we have a tuple completeness for these ones of $\frac{11}{12} = 0.916$.

- Attribute completeness, for the same reason stated above, the following attributes have not the entire set of values: firm_score, Job_Location, Provincia, Cultura e tempo libero, Ambiente e servizi, Demografia e società, Giustizia e sicurezza, Affari e lavoro, Ricchezza e consumi, average_punteggio. Mathematically, for firm_score we have a completeness of $\frac{274}{314} = 0.87$. While for the others, $\frac{278}{314} = 0.88$.

- Table completeness, count for the missing values in the entire dataset and compare this number with the total number of observations in the dataset. In this case, for the reasons stated above, 420 null values are present out of 4396 total number of values. The table completeness is $1 - \frac{420}{4396} = 0.90$.

- Object completeness, the objects that can be represented are more than the observations in the table. This because, taking into account data scientists job and Italy as geographical position, probably there will be other offers in different websites so the objects are not all in this dataset. However, we have no idea about the amount of further requests in internet and, for this reason, we can't give a numerical estimate of the object completeness.

5.3 Currency

A first measure of the currency is the time delay between the time t1 of the event in the real world that caused the variation of the data, and the instant t2 of its registration in the information system. In this case, data about the quality of life are referred to 2022, while the job offers in glassdoor were related to a day at the beginning of march. As soon as the indexes of the quality of life from 2023 arrives, they has to be updated.

5.4 Consistency

Just looking at the dataset, it seems the data are consistent. Otherwise, we didn't check if actually in that municipalities there is the company headquarters! For example, a job offer can show as location Milan, but it can be in Rho, just few chilometers away from it. If this would happen, there will be inconsistency in the dataset.

Chapter 6

Results from the research

Bologna is at the top of the rank: it is the Italian province where people live better. This gold medal is not the first one, in fact among the thirty-three editions Bologna reached the top four times before this one. The second place is occupied by Bolzano, that is actually very rarely below the tenth position and it is followed by Florence and Pisa, two Tuscan provinces.

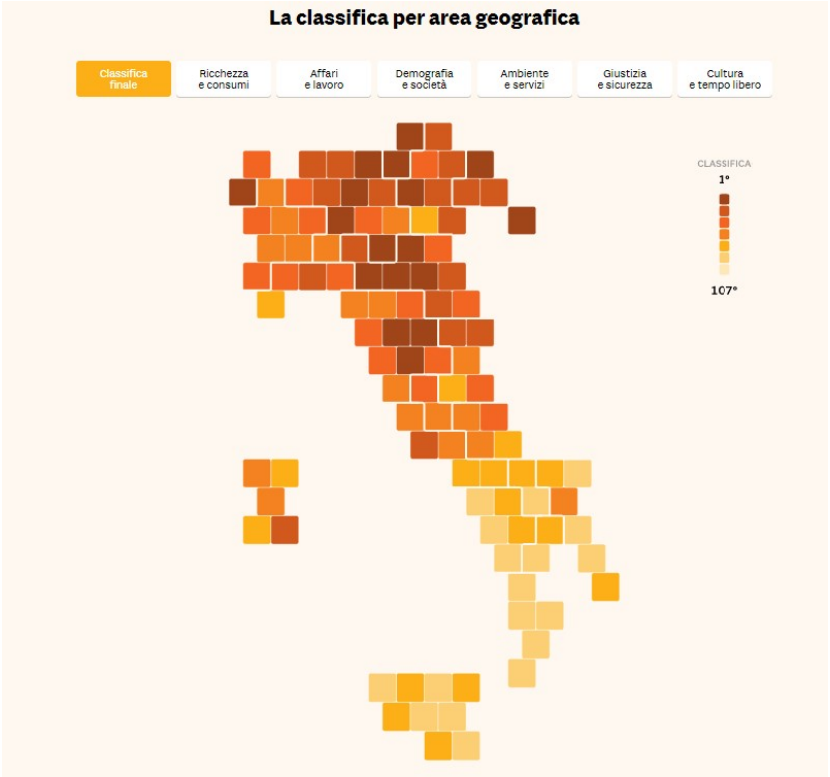


Figure 6.1: Geographical representation of quality of life [3]

Trentino Alto Adige, with the two provices Trento and Bolzano, has both of them in the first five positions: this region is the italian flagship with one of the highest quality of air, high levels of sustainable mobility and the top for waste management. On the other side, Milan, Rome and Turin decrease significantly performances for three different reasons: the Lombard city for the low levels of "Ricchezza and consumi" and

the mean population that can't afford its costs anymore, the capital instead collapses for the index related to wellness of youngest. Lastly, Turin is going down for the bad quality of air and high number of crimes.

The last ten cities are all in the south of the country between Sicily, Calabria, Puglia and Molise. This is a result of the great division between the two parts as a consequence of bad political choices and emigration of young people, leaving a deep productive hole. The biggest problem is related to the indicator NEET, that states the percentage of people under 29 who don't study neither work; it is 36% in 2020 (see [4]).

Related to this topic, the data set has six indicators as stated in the section 2.2. In figure 6.2, seven boxplots are shown: one for each indicator and the last that is the average of the six before.

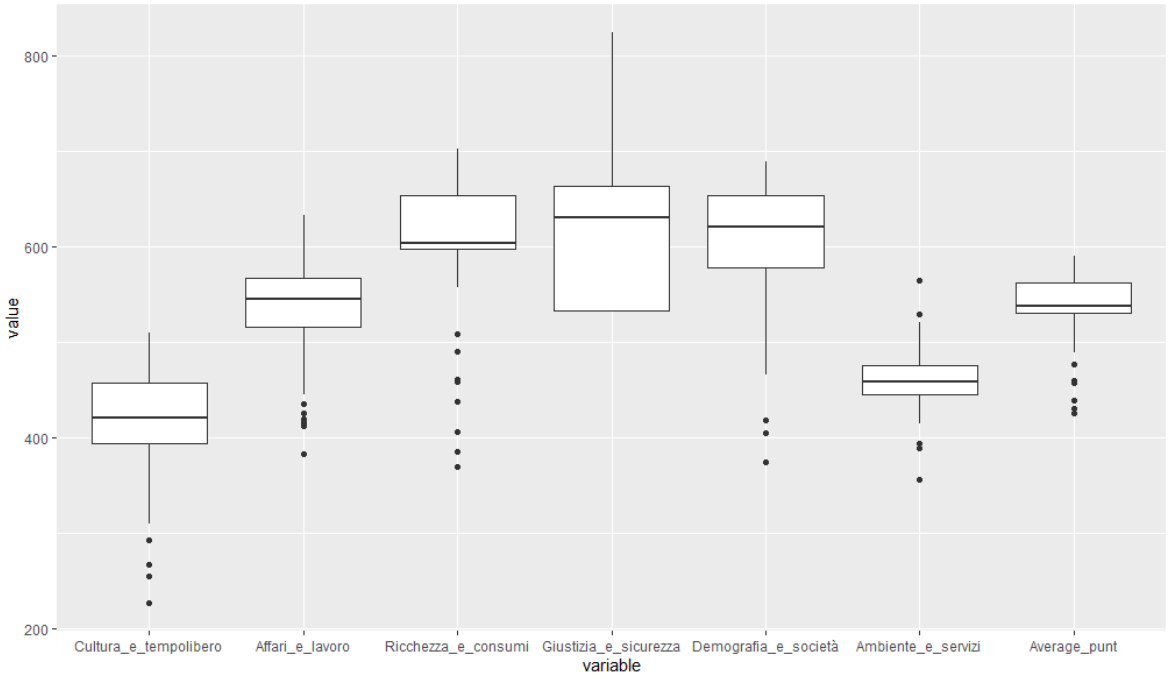


Figure 6.2: Boxplot variables scores

In the following picture 6.3, the top firms are gathered according to the regions. This can be useful if a data scientist has already the idea of in which region to live. For example, in Lazio the top firms are Open Search Network, Sorgenia, Translated and i-EM, while in Lombardia TapMyLife, AMA European Consulting, Prima Assicurazioni and Techyon. The following region with top firms are Piemonte and Campania.

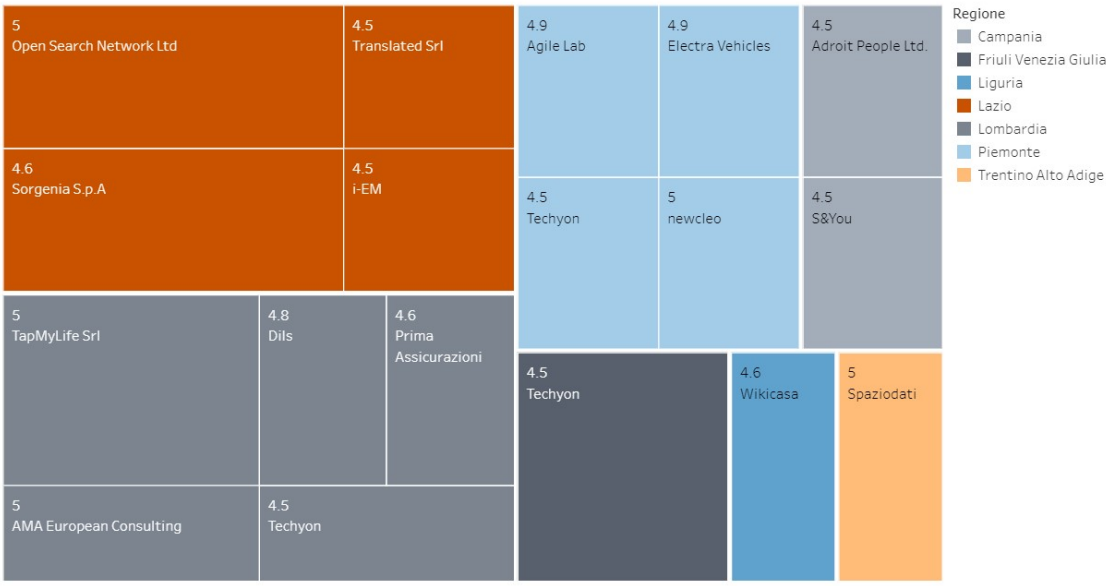


Figure 6.3: Top firms per region (see [5])

6.1 Among the top firms across Italy, where a new employee should work according to the quality of life of that cities?

We performed a query on the final data set (see figure 6.5) to determine which are the top job offers available, for the position of data scientist, according to the firm score. Having obtained this information, we wanted to analyze which of these 10 job positions are located in the city with the best quality of life.

```
import pandas as pd

dataset = pd.read_csv('FINITO.csv')

nome_colonna = "firm_Score"

righe_valore_5 = dataset.loc[dataset[nome_colonna] == "5"]


nome_colonna2="Average_punt"
dataset_sorted_by_region_score = righe_valore_5.sort_values(by=nome_colonna2, ascending=False)

print("Migliori lavori in base al punteggio dell'azienda e alla qualità della vita della città:")
dataset_sorted_by_region_score
```

Figure 6.4: Query for top 10 job positions offered in Italy

As can be seen from the images below , we can determine that the best job positions, by company score and city score, are offered by the Fraunhofer Italia company located in Bolzano (1st and 2nd position) and by the Spaziodati company located in Trento (3rd position).

firm_Score	Firm	ad_Firm_Role	Job_Location	Provincia	Regione	Cultura_e_tempolibero	Affari_e_lavoro	Ricchezza_e_consumi	Giustizia_e_sicurezza	Demografia_e_società	Ambiente_e_servizi	Average_punt
5	Fraunhofer Italia	COMPUTER SCIENTIST FOR APPLIED RESEARCH IN AUT...	Bolzano	Bolzano	Trentino-Alto Adige	425.8	493.2	677.0	762.1	603.8	552.341	585.707
5	Fraunhofer Italia	ENGINEER OR SCIENTIST WITH EXPERIENCE IN SUSTA...	Bolzano	Bolzano	Trentino-Alto Adige	425.8	493.2	677.0	762.1	603.8	552.341	585.707
5	Spaziodati	Machine Learning Engineer / Data Scientist	Trento	Trento	Trentino-Alto Adige	448.5	519.3	662.3	701.1	591.8	536.632	576.605
5	TapMyLife Srl	Data scientist	Bergamo	Bergamo	Lombardy	413.6	490.3	651.4	768.3	555.7	488.336	561.273
5	TapMyLife Srl	Data scientist	Bergamo	Bergamo	Lombardy	413.6	490.3	651.4	768.3	555.7	488.336	561.273
5	Open Search Network Ltd	Data Scientist	Rome	Roma	Lazio	421.1	566.4	597.5	532.3	653.7	458.858	538.310
5	Open Search Network Ltd	Senior It Architect	Roma	Roma	Lazio	421.1	566.4	597.5	532.3	653.7	458.858	538.310
5	newcleo	Computational Scientist	Turin	Torino	Piedmont	393.6	492.1	648.1	639.8	578.1	433.093	530.799
5	AMA European Consulting	Data Scientist (EU Joint Research Program)	Ispra	Varese	Lombardy	343.8	499.9	637.9	688.4	560.0	444.486	529.081
5	Modelway	Data Scientist	Remote	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 6.5: Top 10 positions in Italy according to the score of the firm and the average score of the quality of life

It is therefore determined that Trentino can be considered the region that offers the best job offers in terms of quality of life and company quality.

6.1.1 Are the job offers homogeneously distributed between the south and the north of Italy?

```
import pandas as pd

df = pd.read_csv('FINITO.csv')

df['zona_italia'] = df['Regione'].apply(lambda x: 'Nord' if x in ['Lombardy', 'Veneto', 'Piedmont','Friuli-Venezia Giulia','Tren
                                                  else 'Centro' if x in ['Lazio', 'Toscana', 'Emilia-Romagna','Tuscany','Marche'
                                                  else 'Sud' if x in ['Apulia', 'Sardinia', 'Sicily','Campania','Abruzzo','Calab
                                                  else 'Smartworking')

jobs_count_by_zone = df['zona_italia'].value_counts()

print("Numero di lavori al Nord Italia:", jobs_count_by_zone['Nord'])
print("Numero di lavori al Centro Italia:", jobs_count_by_zone['Centro'])
print("Numero di lavori al Sud Italia:", jobs_count_by_zone['Sud'])
print("Numero di lavori in Smartworking:", jobs_count_by_zone['Smartworking'])
```

```
Numero di lavori al Nord Italia: 139
Numero di lavori al Centro Italia: 105
Numero di lavori al Sud Italia: 34
Numero di lavori in Smartworking: 36
```

Figure 6.6: Query for job positions divided for clusters in Italy

In order to evaluate whether the job offers are homogeneously distributed between southern and northern Italy, a query (see figure 6.6) was performed on our dataset which contains data regarding the job positions offered in the country.

The query returned the following results: there are 139 job offers in northern Italy and 34 job offers in southern Italy.

On the basis of these results, we can state that job offers are not evenly distributed between southern and northern Italy. Northern Italy presents a significantly higher number of job offers than the south, with 139 proposals in the north and only 34 in the south. This suggests a strong disparity in the availability of job opportunities between the two areas. Northern Italy offers a more dynamic labor market full of opportunities, while the south has a much lower demand for labor.

It should be noted that further analysis and surveys may be needed to fully understand labor market dynamics and regional differences in terms of job opportunities.

Using Tableau [5], a visual analytics platform, some graphical representations such us the one below, figure 6.7, can be visualized. Thanks to colour and the division in regions, it can be easily see the disparity that was explained in the lines above.

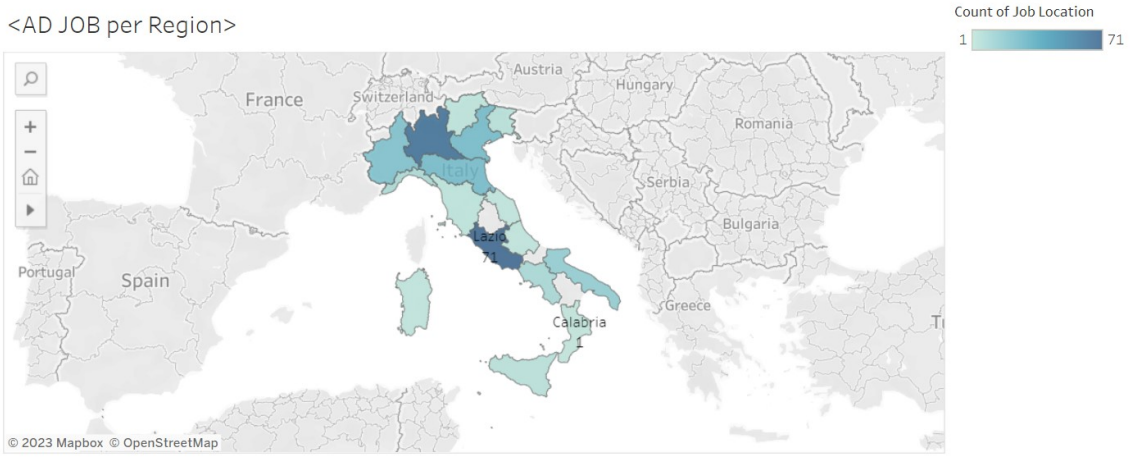


Figure 6.7: Jobs opportunities between the north and the south of Italy

6.1.2 Rome and Milan are the springboards for a new data scientist, let's make a comparison between the two cities. Where is it better to move to taking into account the indices? Which are the best indices for Milan? And which for Rome? What about the top firms?

firm_Score	Firm	ad_Firm_Role	Job_Location	Provincia	Regione	Cultura_e_tempolibero	Affari_e_lavoro	Ricchezza_e_consumi
5.0	Open Search Network Ltd	Data Scientist	Rome	Roma	Lazio	421.1	566.4	597.5
5.0	Open Search Network Ltd	Senior It Architect	Roma	Roma	Lazio	421.1	566.4	597.5
4.8	Dils	Data Engineer Senior	Milano	Milano	Lombardy	491.9	632.1	674.3
4.6	Prima Assicurazioni	Senior Data Scientist	Milan	Milano	Lombardy	491.9	632.1	674.3
4.6	Sorgenia S.p.A	Data Scientist – Digital Market	Rome	Roma	Lazio	421.1	566.4	597.5

Giustizia_e_sicurezza	Demografia_e_società	Ambiente_e_servizi	Average_punt
532.3	653.7	458.858	538.310
532.3	653.7	458.858	538.310
549.2	620.4	475.642	573.924
549.2	620.4	475.642	573.924
532.3	653.7	458.858	538.310

Figure 6.8: Subset of jobs in Rome or Milan ranked by the score of the firm

We select all the job proposals placed in Milan and Rome. Among them, the one with the best quality of life is Milan with an average score of 573,924, while Rome has a lower score of 538,310.

Let's analyze in which of the two cities there are the companies with the highest score and consequently the best job proposals from a business perspective. The results of this analysis is that the best company, Open Search Network Ltd, is located in the city of Rome with a score of 5.0, while two of the companies in Milan, Dils and Prima Assicurazioni, show a score of 4.8 and 4.6, respectively. They are followed by Sorgenia S.P.A in Rome with a score of 4.6.

A data scientist can decide the index of interest and choose the best work according to it. For example, Rome has an average score that is lower than the average score of Milan. However, according to "Demografia_e_sicurezza", Rome occupies an higher position compared to Milan, respectively 653.7 and 620.4. On the other hand, for indexes such as "Cultura_e_tempolibero", "Affari_e_lavoro" e "Ricchezza_e_consumi" the difference between Milan and Rome is equal or greater than 70!

Chapter 7

Conclusions and further development

Since "Il Sole 24 Ore is open to the solicitations of readers who are openly invited to make contributions for even more in-depth journalistic work" [6], this work will be presented to the editorial team of Il Sole24Ore for a possible service for their readers. This can be a great opportunity for young future data scientists like us to grow up professionally and for the newspaper to offer a new insight based on the data of the quality of life that they have already analyzed.

For anyone wishing to resume and continue our analysis, we report some possible changes that could be added in the future. The new European legislation, approved by the Strasbourg Parliament on March 30th of 2023, with 427 over 582 votes in favor, mandates pay transparency in job advertisements, with the aim of combating wage discrimination and promoting gender equality. Adding the salary in this analysis can have an interesting impact in ranking also for the salary.

Moreover, this analysis was circumscribed to a specific job (data scientist), but it could be implemented to every kind of possible employment. In the same way, the borders can overcome Italy! In this case it should be take into account the quality of life per country or per region of each single country in Europe or in the world.

Similarly, we perform scraping from Glassdoor, but the number of job offers is probably bigger considering also other websites. The unique restriction on it is the fact that data from different sources must have the same structure of Glassdoor's one otherwise other part of code has to be carried out.

Another important point is that the data should be real-time data. In such a way, the job offers can be updated each hour and the data scientists can search for real works. However, a more complex structure must be used to store data and to manage them. The possibilities taken into account are the document database with the BASE properties and the advantages of the schema-less, open format and no foreign keys or the structure given by the data warehouse, that provide a stable, centralized repository for large amounts of data. Probably if the research will be extended to more jobs or a bigger geographical base, the structure of the data warehouse would be preferred. On the contrary, if the number of job offers will increase because of the observation of other websites, we would suggest the use of the document NoSQL model.

Bibliography

[1] WHOQOL, World Heath Organization, Quality of life, *Measuring Quality of Life*, https://www.who.int/tools/whoqol.

[2] Quality of Life by Country, The quality of life index, https://worldpopulationreview.com/country-rankings/standard-of-living-by-country

[3] IlSole24Ore, Qualità della vita, https://lab24.ilsole24ore.com/qualita-della-vita/

[4] Sole24ore, Il divario tra Nord e Sud in Italia: perché va sempre peggiohttps://www.econopoly.ilsole24ore.com/2023/05/08/distanza-nord-sud-italia/

[5] Graphical representations in Tableau, Francesco Angiulli, Tableau https://public.tableau.com/app/profile/francesco7156/viz/RicchezzaRegioniItaliane_16854635203880/Storia1?publish=yes

[6] IlSole24Ore, Linee guida editoriali https://st.ilsole24ore.com/linee-guida-editoriali/