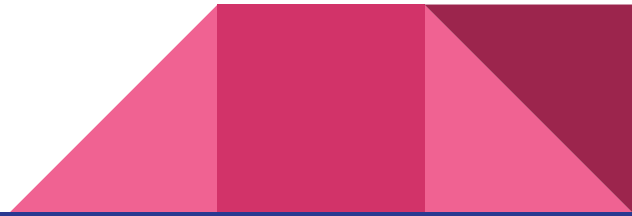
Web Scraping & Hierarchical Clustering Analysis

Table of Contents



Web Scraping

Introduction:

- Extract valuable information from website
- Convert poorly structured data into a usable, structured format
- Target specific information
- Use Python(Beautiful Soup and Requests) to automate web scraping process

Web Scrapping Methodology

HTML: Hyper Text Markup Language

- Standard format for webpages
- Defines content by various tags and classes

Beautiful Soup Library:

- Scrapes HTML and XML content
- Can pull data for various defined classes

General Procedure to do Scraping

Use Requests to get HTML from URL

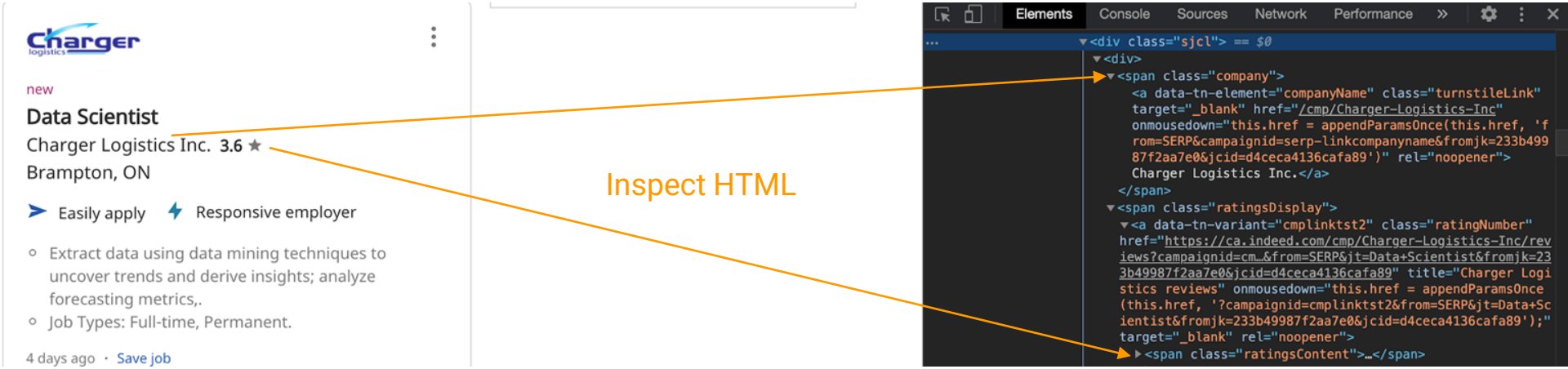
Use BeautifulSoup to parse the content in HTML with selected parser

Locate the information we need by finding corresponding tags and classes

Extract link from HTML and modify it to usable URL

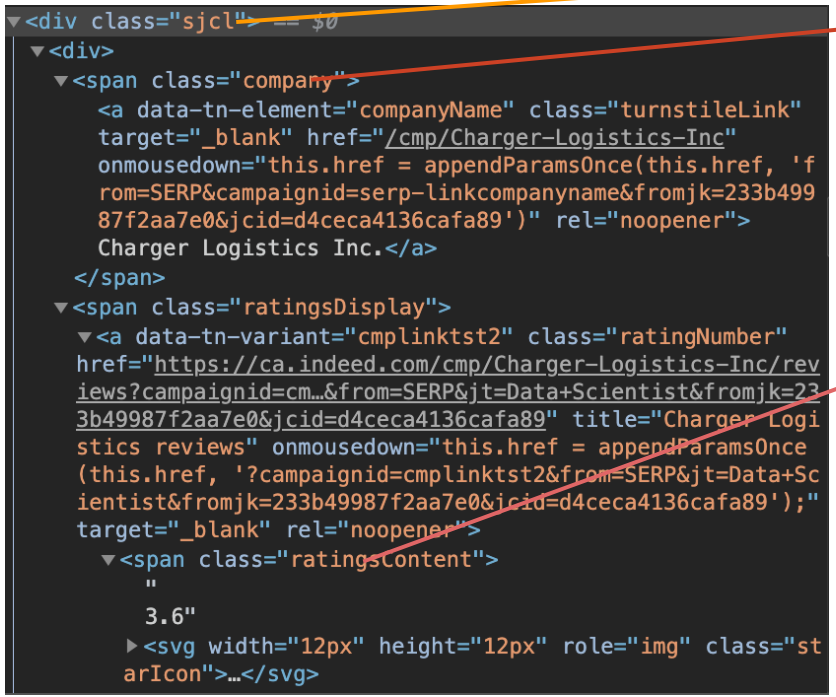
```
##output:Couple Lists of information such as title,locations,companies#
def get_page_info(url):
    #request url for getting the html content#
    response = requests.get(url)
    content = response.content
    #use beautiful soup to parse the content#
    html_soup = BeautifulSoup(content,'lxml')

    job_urls =[]
    #find h2 tag title class in soup#
    for divTag in html_soup.find_all('h2',{'class':'title'}):
        #find a tag in those h2 tag title class#
        for aTag in divTag.find_all('a'):
            #obtain url for each job#
            url ='https://ca.indeed.com'+ aTag.get('href')
            #obtain job titles#
            job_titles.append(aTag.get('title'))
            #save them in a list#
            job_urls.append(url)
```

Example: Indeed.ca



By Inspecting HTML, we can match each element in web page with a tag and a class in HTML.
Therefore, we can extract the information by locating the tag and the class in HTML.

Example Cont.




```
#find div tag sjcl class in soup#
for divTag in html_soup.find_all('div',{'class':'sjcl'}):
    for spanTag in divTag.find_all('span',{'class':'company'}):#find span tag in those div tag#
      if spanTag != None:
        #obtain company name and save them in a list#
        companies.append(spanTag.get_text().replace("\n",""))
      else:
        companies.append(None)

    #find span tag location class in those div tag#
    #obtain locations#
    #save it in a list#
    try:
      ratings = divTag.find('span',{'class':'ratingsContent'}).get_text()
      company_ratings.append(ratings.replace("\n",""))
    except AttributeError:
      ratings = float('nan')
      company_ratings.append(ratings)
```

Big loop through class:sjcl -> includes company & rating

1st small loop through class:company

-> company name

2nd small loop through class:ratingsContent

-> rating

Build a Dictionary to Find Skills I

- Build a dictionary with key representing the skill category, and values that are keywords related to that skill like:

```
'SAS': ['SAS'],
'SQL/databases': ['SQL', 'databases'],
```

- Search relevant keywords in the job page text for each category
  - If the skill keyword will be found : category_found 1

```
# search for the skills
soup_job_text = soup_job.text
for skill_category, skills in skills_keywords_dict.items():
  category_found = 0   # variable used to store results of the intermediate check (loop below)
  for skill in skills:        # loop over all skills in the sublist of 'skills_keywords_dict'
    if soup_job_text.find(skill) != -1:     # if skill from the sublist is found, set 'category_found' to 1
        category_found = 1
  results_dict[job_url][skill_category] = category_found   # skill set to 1 if found, 0 if not, in 'results_dict'
```

Build a Dictionary to Find Skills II

- Build a new dictionary (results_dict)
  - key : job_url
  - values : a dictionary showing skill_category as key and its existence with 0 and 1 as a value
- Covert results_dict to a dataframe

```
fvj=1&vjs=3': {'Excel': 1, 'Python': 1, 'R': 1, 'Java': 0, 'Scala': 0, 'C/C++': 0, 'MATLAB': 0,
fvj=1&vjs=3': {'Excel': 1, 'Python': 1, 'R': 1, 'Java': 0, 'Scala': 0, 'C/C++': 0, 'MATLAB': 0,
fvj=1&vjs=3': {'Excel': 1, 'Python': 1, 'R': 1, 'Java': 0, 'Scala': 0, 'C/C++': 0, 'MATLAB': 0,
fvj=1&vjs=3': {'Excel': 1, 'Python': 1, 'R': 1, 'Java': 0, 'Scala': 0, 'C/C++': 0, 'MATLAB': 0,
fvj=1&vjs=3': {'Excel': 1, 'Python': 1, 'R': 1, 'Java': 0, 'Scala': 0, 'C/C++': 0, 'MATLAB': 0,
fvj=1&vjs=3': {'Excel': 1, 'Python': 1, 'R': 1, 'Java': 0, 'Scala': 0, 'C/C++': 0, 'MATLAB': 0,
fvj=1&vjs=3': {'Excel': 1, 'Python': 1, 'R': 1, 'Java': 0, 'Scala': 0, 'C/C++': 0, 'MATLAB': 0,
```

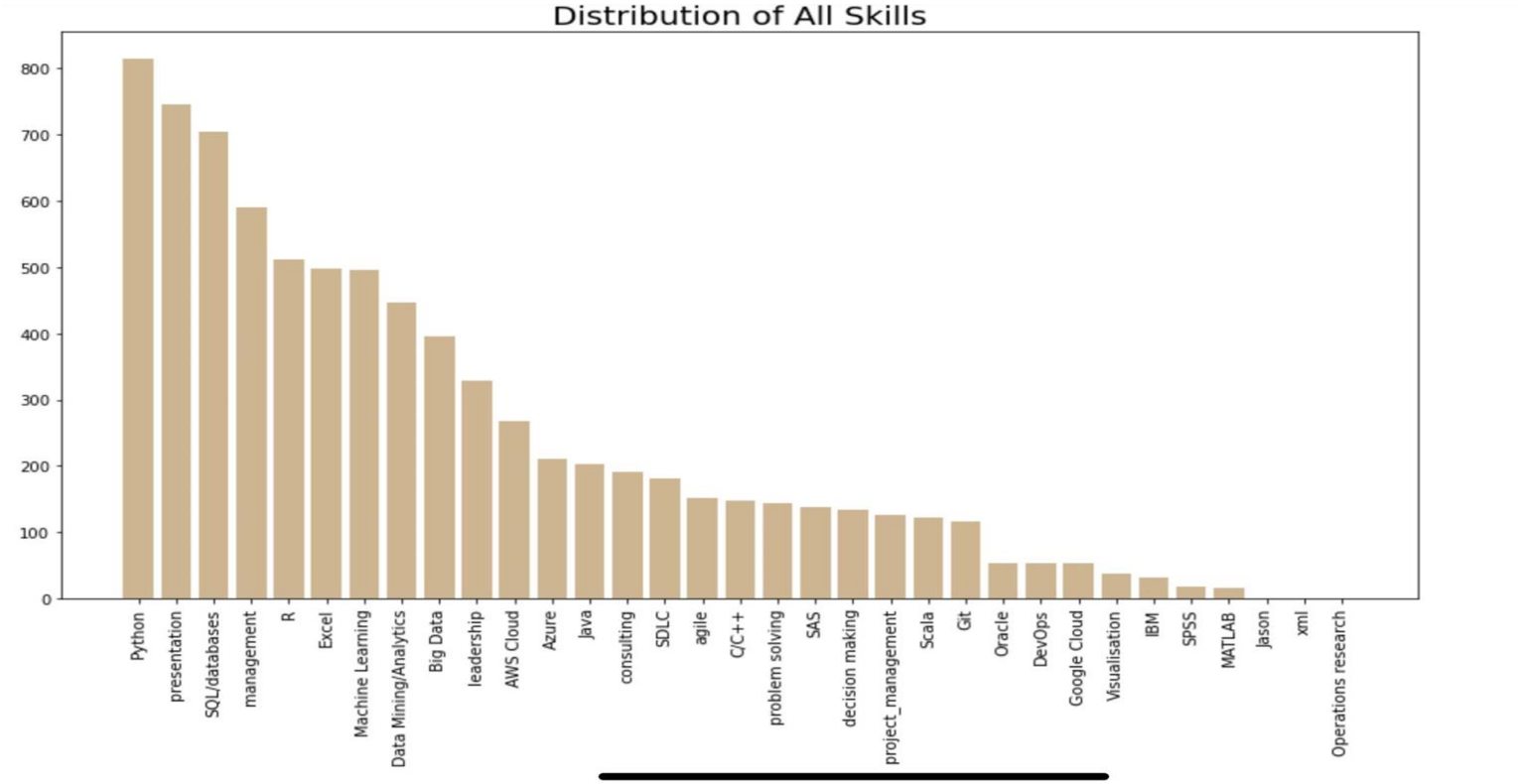
Scraping Result

In the red box, all detailed information of a job is shown

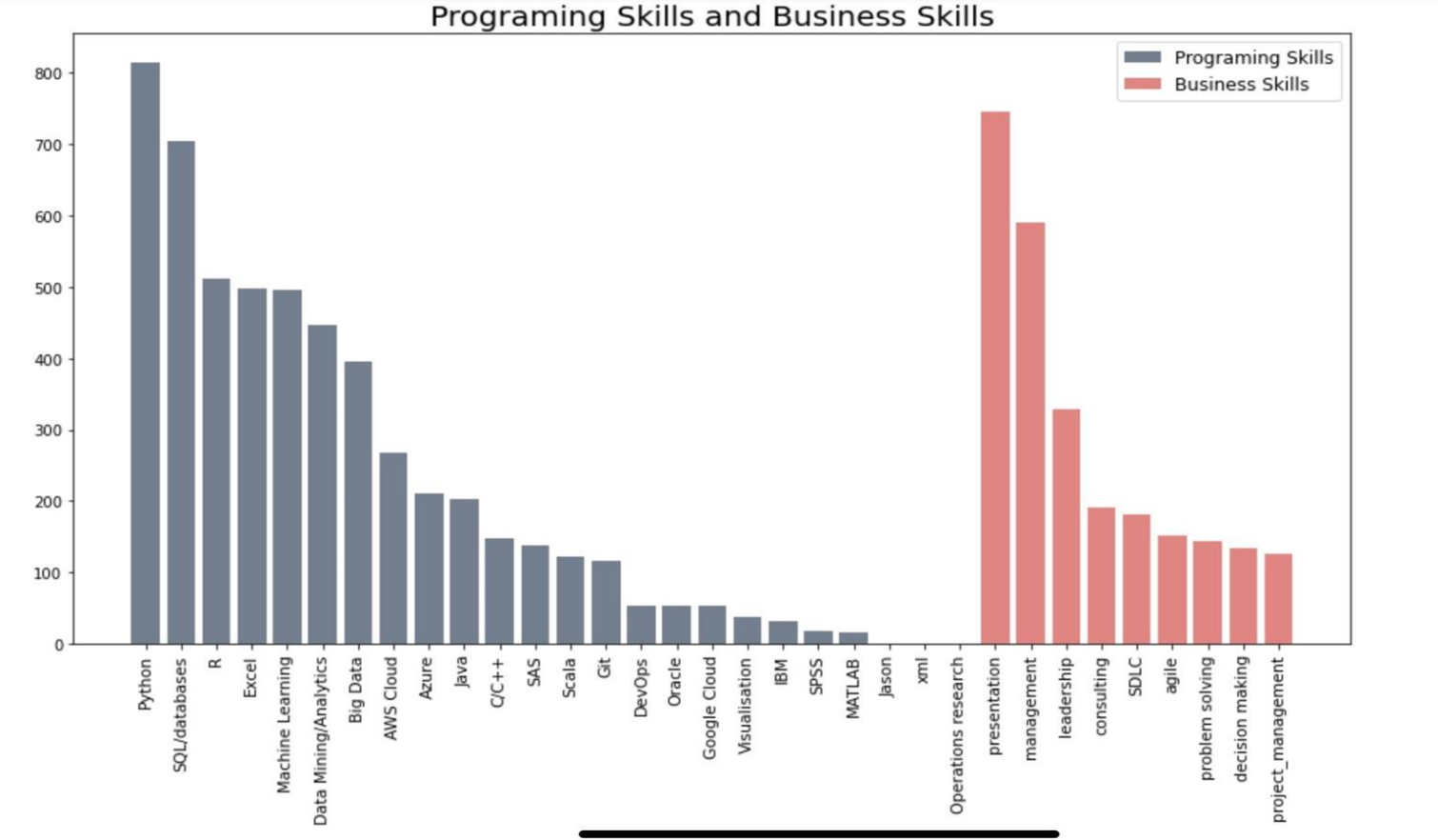
In the blue box, software skills
and business skills are marked
As 1 if it is mentioned in job description
And 0 if it is not

	job_titles	companies	company_ratings	locations	post_dates	Salary	Excel	Python	R	Java	Scala	C/C++	MATLAB	SAS	SQL/dat
0	Senior Data Scientist	MaxSold	3.6	NaN	8 days ago	None	1	1	0	0	0	0	0	0	
1	Data Scientist, Solar & Storage (Remote)	Power Factors	NaN	NaN	30+ days ago	None	0	1	1	0	0	0	0	0	
2	Data Scientist	Charger Logistics Inc.	3.6	NaN	3 days ago	None	1	1	1	0	0	0	0	0	
3	Junior Data Scientist with Python experience	Samiti Technology	NaN	Toronto, ON	Today	$250 - $350 a day	0	1	0	0	0	0	0	0	
4	Data Scientist	CakeAI	NaN	Toronto, ON	Today	None	0	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
115	Senior Data Scientist, GANs TORONTO, ONSOFTWARE	Tonal	NaN	Toronto, ON	23 days ago	None	0	0	0	0	0	0	0	0	
116	Lead Data Scientist	Peak Power	NaN	Toronto, ON	8 days ago	None	1	1	0	0	0	0	0	0	
117	Data Analyst	Microsoft	4.2	Vancouver, BC	30+ days ago	None	0	1	0	0	0	0	0	0	
118	Senior Data Scientist, KPMG	KPMG	4.0	Montréal, QC	22 days ago	None	1	0	1	0	0	0	0	0	

Distribution of Various Skills



Distribution of Various Skills



Hierarchical Clustering

Hierarchical Clustering

- Another plot that we used to interpret results from the job postings is a dendrogram visualizing the hierarchical clustering of the skills

- This plot indicates how the skills demanded by employers in the job postings relate to one another

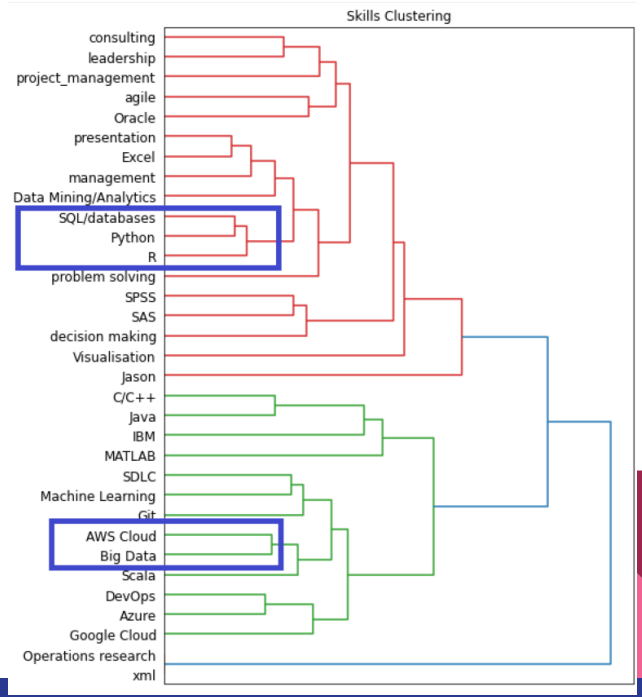
Hierarchical Clustering

Excel	Python	R	Java	Scala	C/C++	MATLAB	SAS	SQL/dat
1	1	0	0	0	0	0	0	
0	1	1	0	0	0	0	0	
1	1	1	0	0	0	0	0	
0	1	0	0	0	0	0	0	

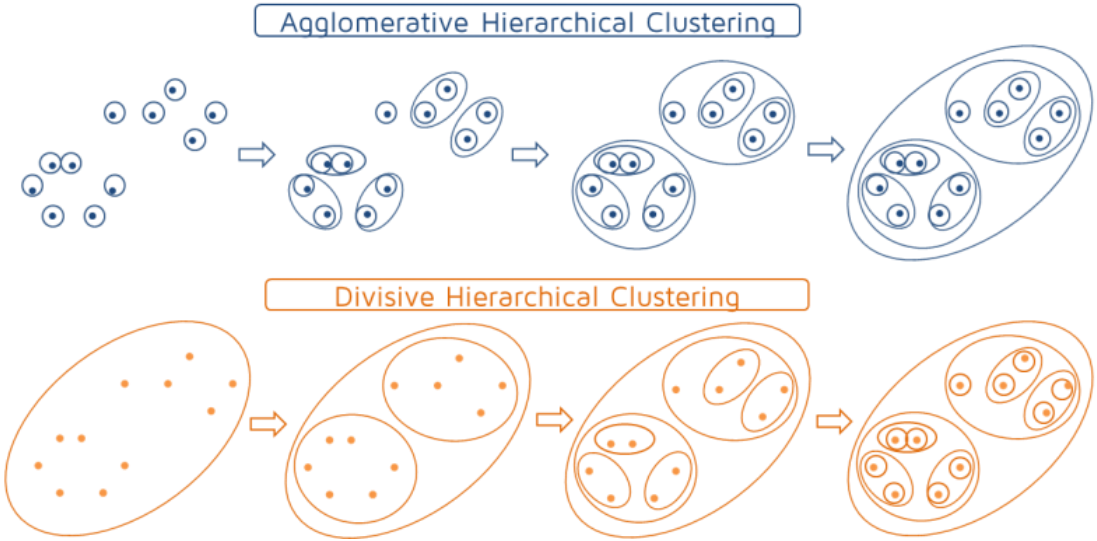
- Re-encoded the skills to capture co-occurrences
  - used as the distance metric for the Hierarchical Clustering
- A matrix is created, where for each pair of skills the total number of job posts where both skills are mentioned is recorded
- The results are normalized and used to construct dendrograms using Hierarchical Clustering

- Skills that co-occur in multiple job postings end up in the same cluster
- As the frequency of co-occurrences of a pair of skills increases,

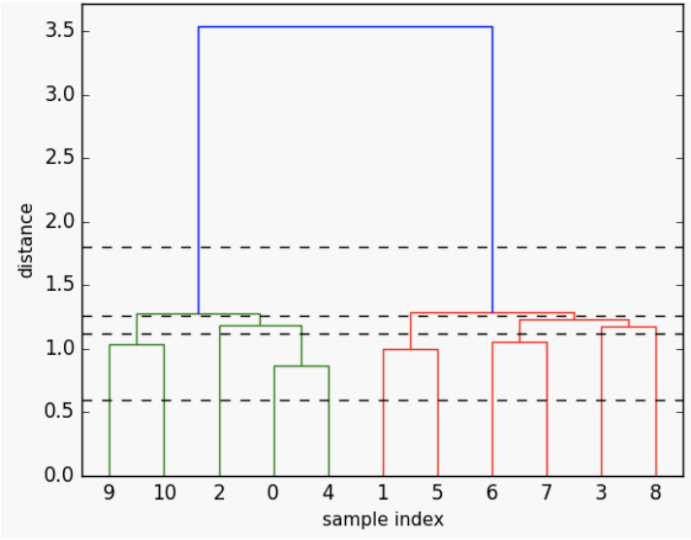
  the clusters in which skills are assigned become closer



Hierarchical Clustering



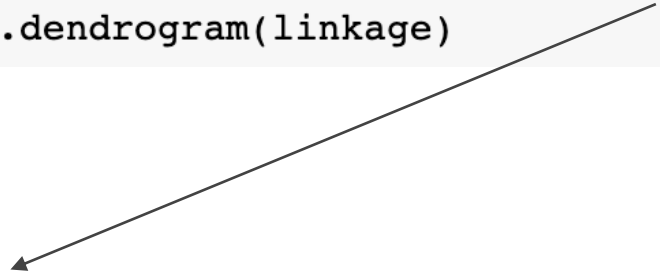
Dendrogram



credit: https://quantdare.com/hierarchical-clustering/

Hierarchical Clustering with Scipy

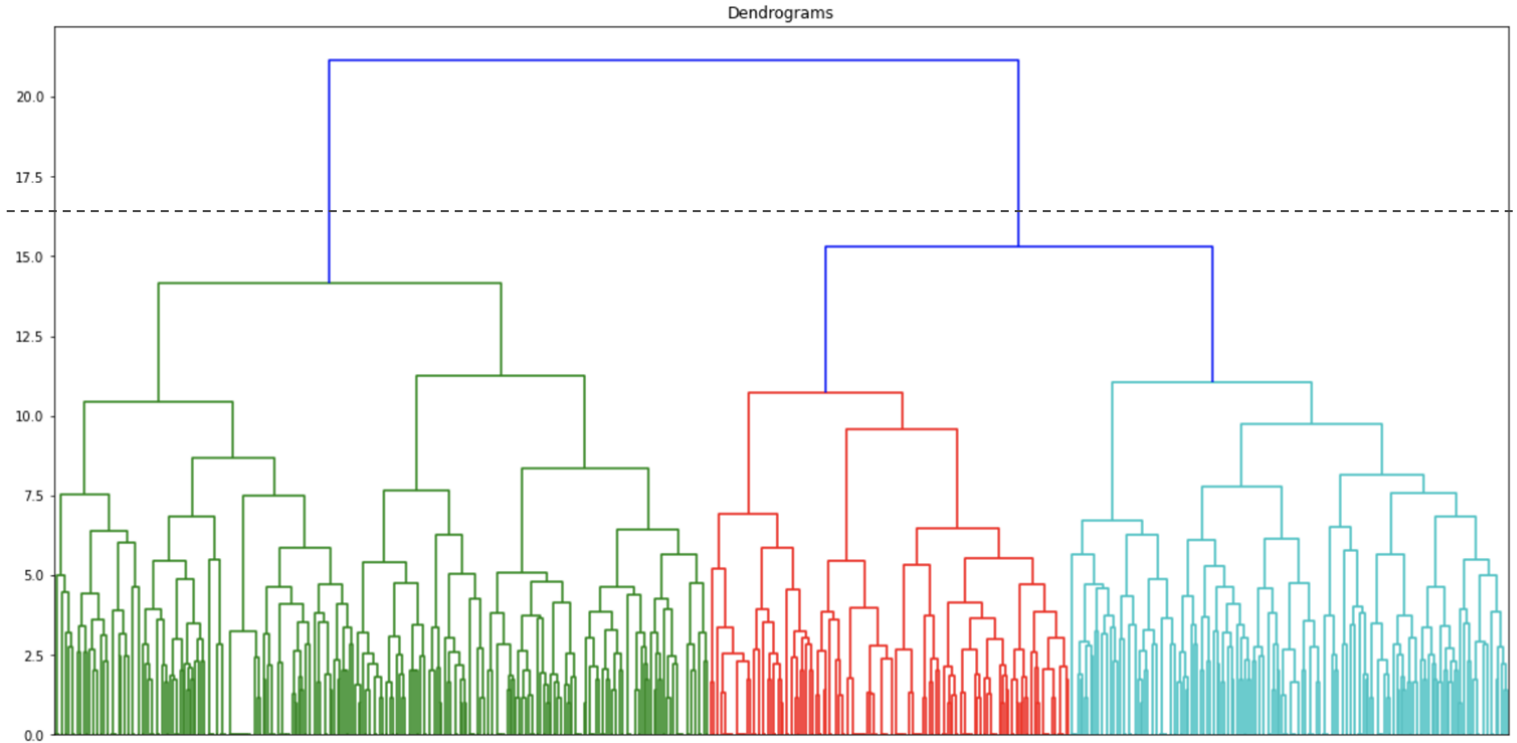
```
from scipy.cluster import hierarchy
#hierarchy.linkage(distance matrix, distance calculation method)
linkage = hierarchy.linkage(df.iloc[:,2:], 'ward')
hierarchy.dendrogram(linkage)
```



Distance Matrix:

Required Skills from 700+ Indeed Jobs

Dendrogram of Job Clusters by Programming & Business Skill

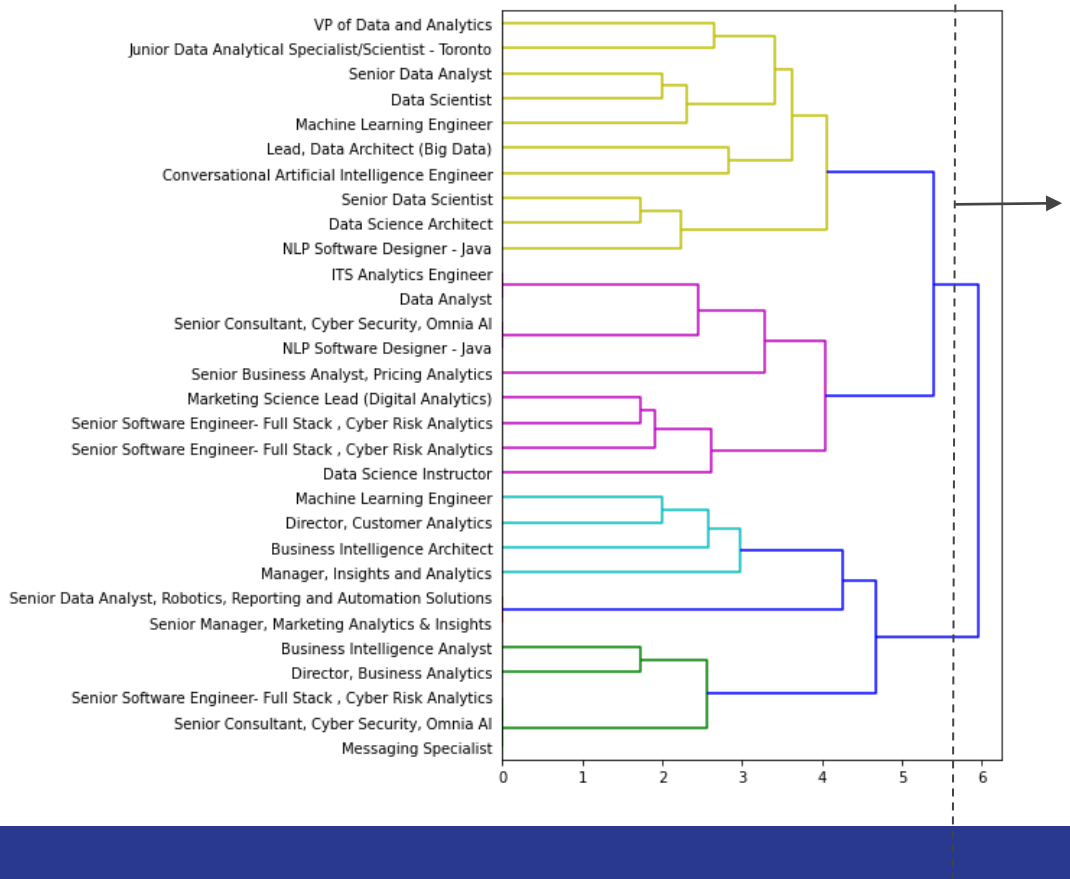


Clusters by Similar Skills Required

Manually Set Threshold Suggests 2 Classes:

- Programming
- Business

Random Sample Investigation

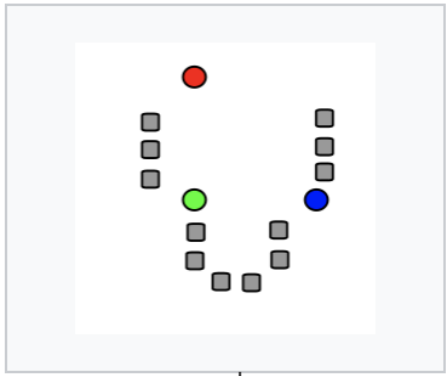


30 Random Samples

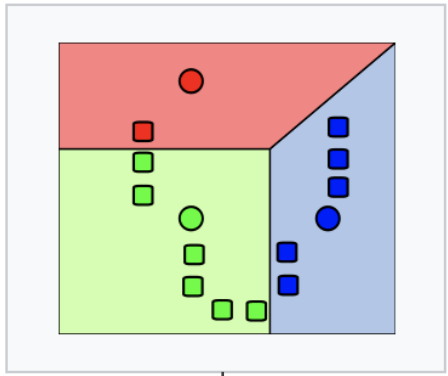
for Better Visualization:

- Most of top rows leaning towards analytical skills
- Most of bottom rows leaning towards business skills

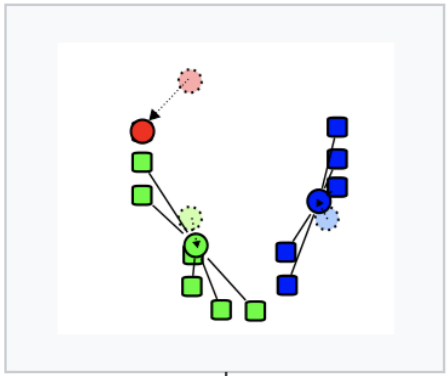
K-Means Clustering



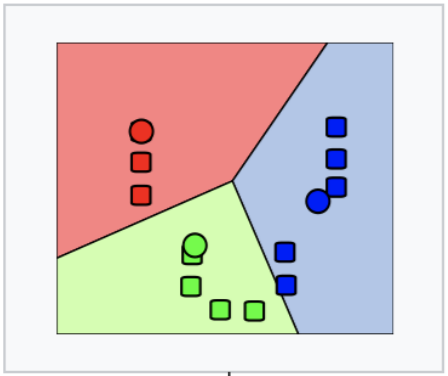
Manually Determine N Centers



Cluster by Nearest Means



Centroids of Clusters Become New Centers



Iterate until Convergence

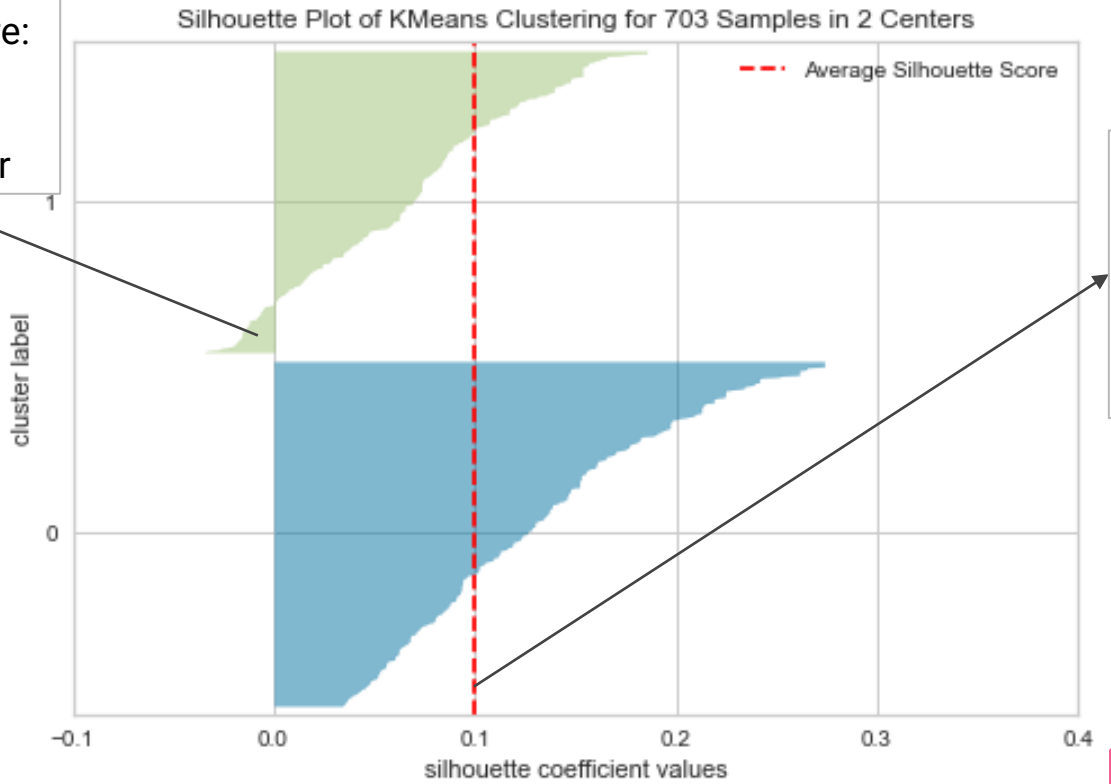
credit: https://en.wikipedia.org/wiki/K-means_clustering

Silhouette Plot of K-Means Clustering

Negative score:

Overlap or

Potential Error



Silhouette coefficient ranges from -1 to 1:

0.1 suggests very weak separation between classes

Thank you for watching!

ANY QUESTIONS?