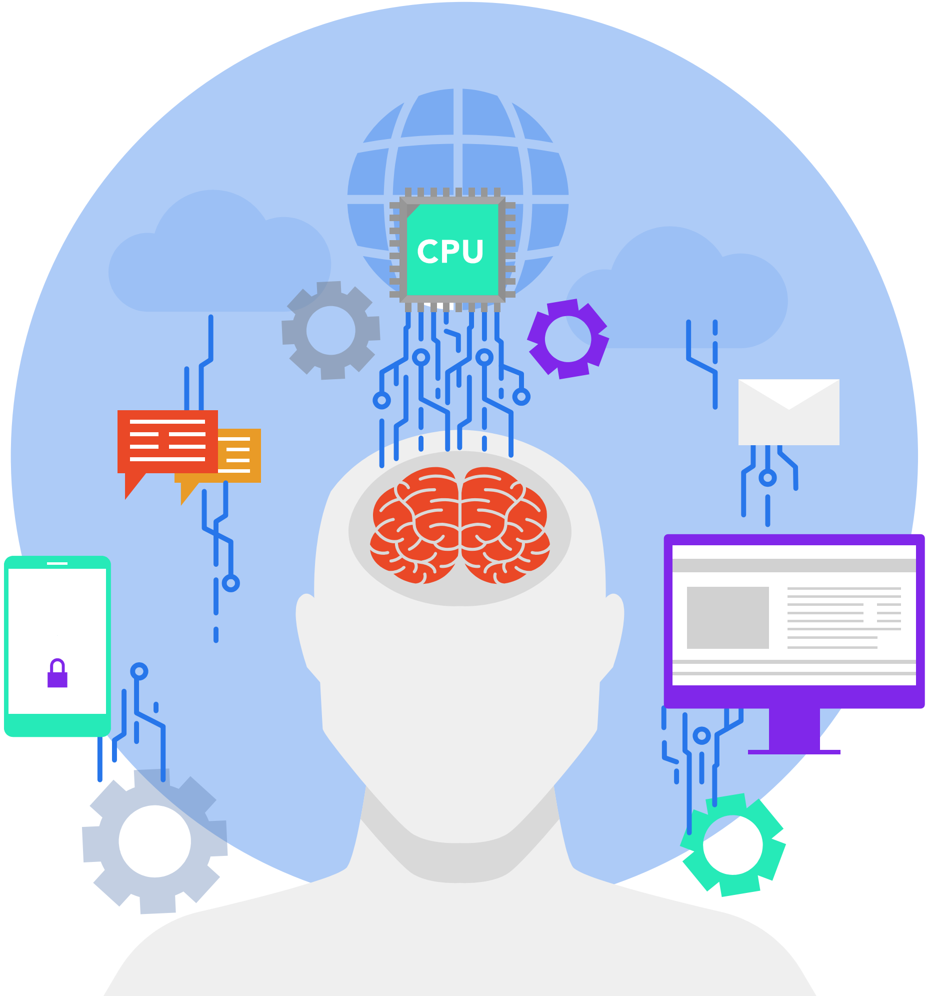
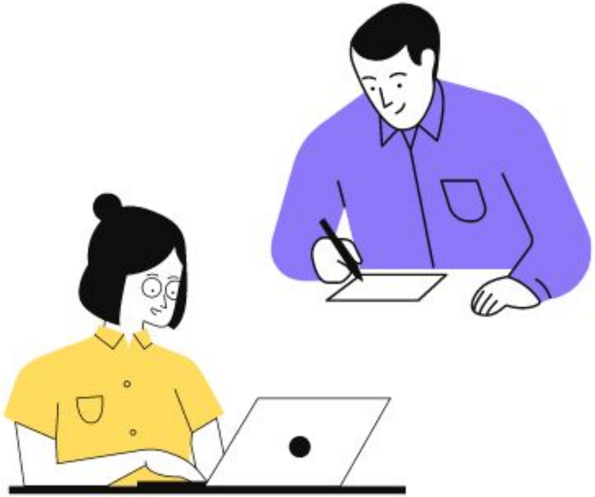


Machine Learning With Big Data Project



01 Introduction

02 Scraping Data

03 Clustering the Data using scikit-learn

04 Clustering the Data using PySpark

05 Python Packages used

06 Gained Knowledge

Goals of Our project

Collecting and analysing data from
job portals

Tasks :

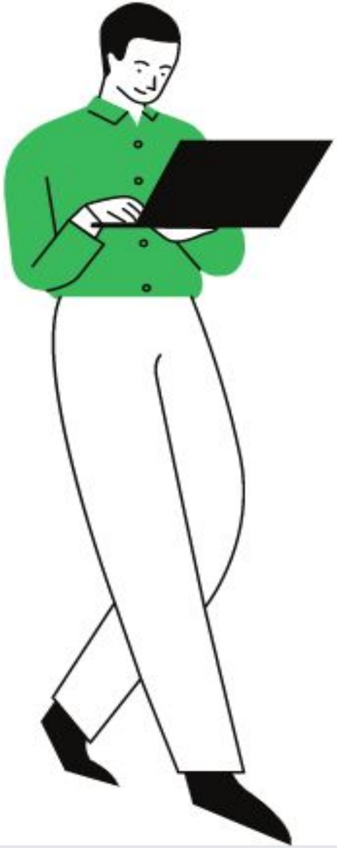
1-scrape data from the web

2-cluster the data using Scikit-learn

3-cluster the data using pyspark

1- Scraping

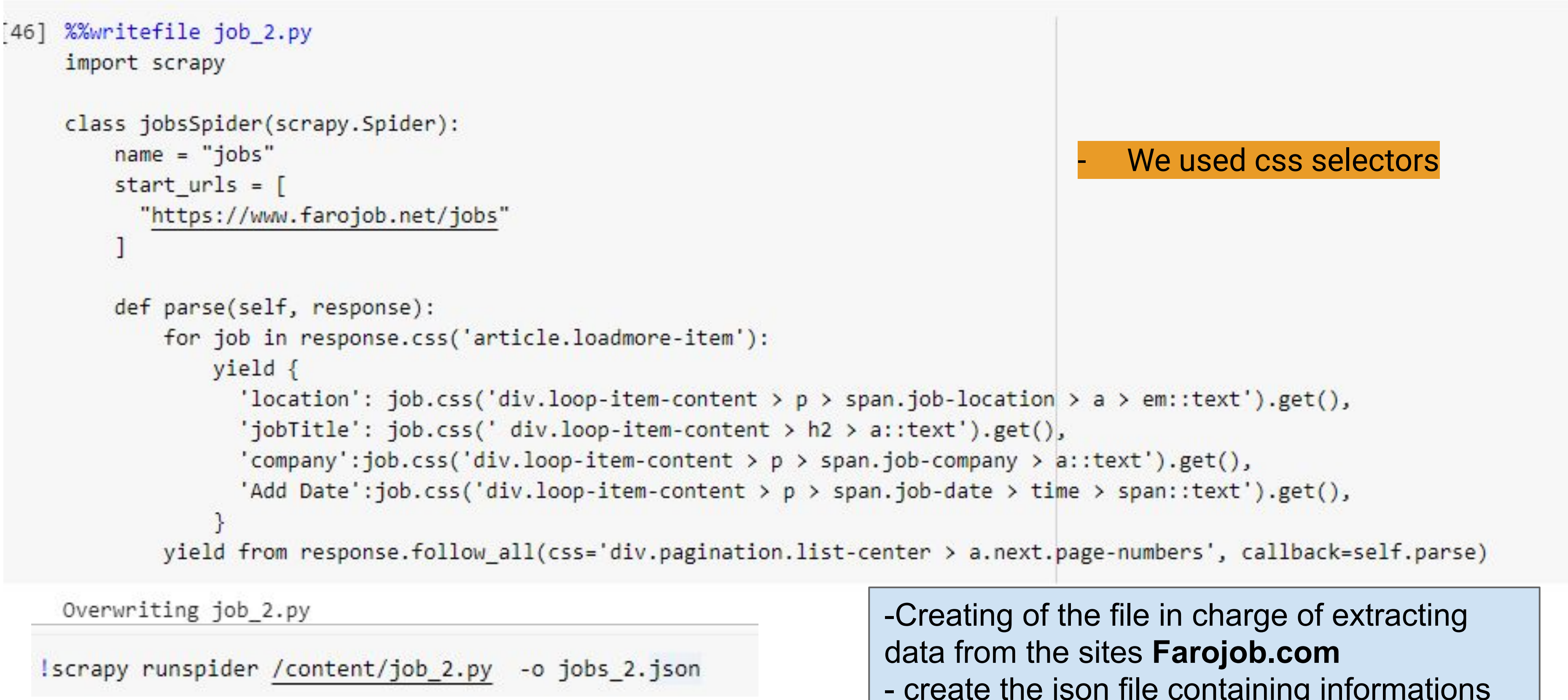
01 Def of scrapy



Explanation Of the code 02

Scrapy is a free and open-source web-crawling framework written in Python and developed in Cambuslang. Originally designed for web scraping, it can also be used to extract data using APIs or as a general-purpose web crawler. It is currently maintained by Zyte, a web-scraping development and services company.

A Scrapy spider typically generates many dictionaries containing the data extracted from the page. To do that, we use the `yield` Python keyword in the callback, as you can see below:

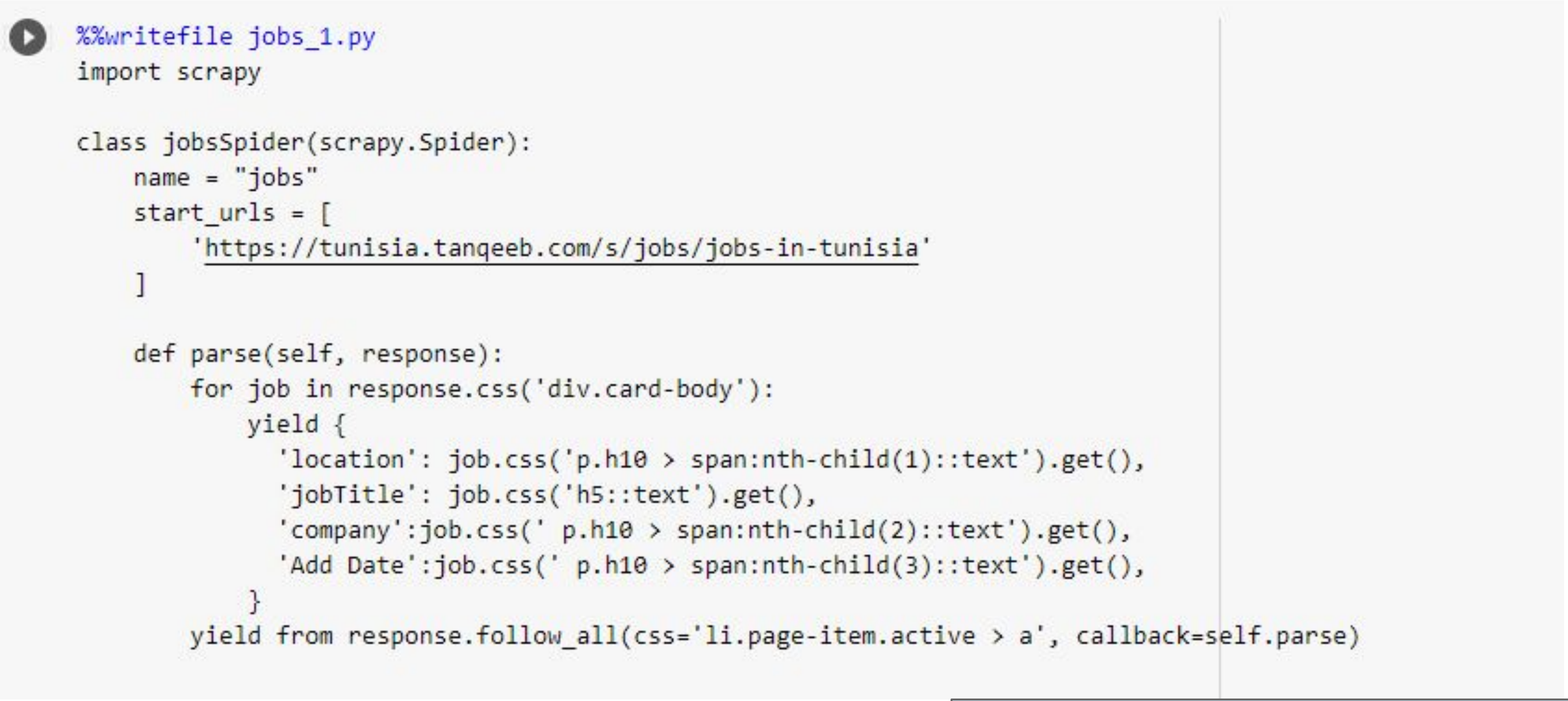


```
[46] %%writefile job_2.py
     import scrapy

     class jobsSpider(scrapy.Spider):
         name = "jobs"
         start_urls = [
           "https://www.farojob.net/jobs"
         ]

         def parse(self, response):
             for job in response.css('article.loadmore-item'):
                 yield {
                   'location': job.css('div.loop-item-content > p > span.job-location > a > em::text').get(),
                   'jobTitle': job.css(' div.loop-item-content > h2 > a::text').get(),
                   'company':job.css('div.loop-item-content > p > span.job-company > a::text').get(),
                   'Add Date':job.css('div.loop-item-content > p > span.job-date > time > span::text').get(),
                 }
             yield from response.follow_all(css='div.pagination.list-center > a.next.page-numbers', callback=self.parse)

     Overwriting job_2.py
```

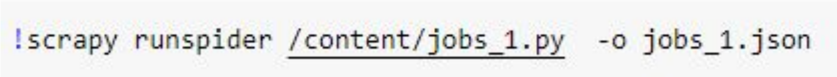
```
!scrapy runspider /content/job_2.py  -o jobs_2.json
```

- We used css selectors

-Creating of the file in charge of extracting data from the sites **Farojob.com**
- create the json file containing informations such as job titles , location, date and company



```
%%writefile jobs_1.py
import scrapy

class jobsSpider(scrapy.Spider):
    name = "jobs"
    start_urls = [
        'https://tunisia.tanqeeb.com/s/jobs/jobs-in-tunisia'
    ]

    def parse(self, response):
        for job in response.css('div.card-body'):
            yield {
              'location': job.css('p.h10 > span:nth-child(1)::text').get(),
              'jobTitle': job.css('h5::text').get(),
              'company':job.css(' p.h10 > span:nth-child(2)::text').get(),
              'Add Date':job.css(' p.h10 > span:nth-child(3)::text').get(),
            }
        yield from response.follow_all(css='li.page-item.active > a', callback=self.parse)
```



```
!scrapy runspider /content/jobs_1.py  -o jobs_1.json
```

-Creating of the file in charge of extracting data from the sites **tunisia.tanq**eeb.com
- create the json file containing informations such as job titles , location, date and company

What just happened under the hood?

Instead of implementing a `start_requests()` method that generates `scrapy.Request` objects from URLs, you can just define a `start_urls` class attribute with a list of URLs. This list will then be used by the default implementation of `start_requests()` to create the initial requests for your spide

The `parse()` method will be called to handle each of the requests for those URLs, even though we haven’t explicitly told Scrapy to do so. This happens because `parse()` is Scrapy’s default callback method, which is called for requests without an explicitly assigned callback.

2- Scikit Learn

01 Create a Pandas Dataframe

02 Create a corpus

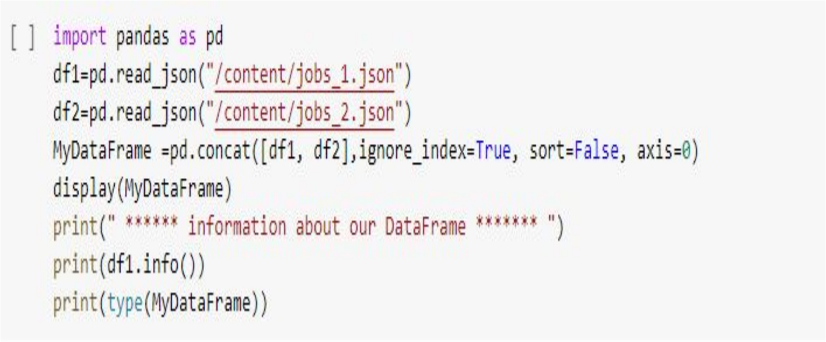
03 Remove noise function

04 Create a Tf-idf matrix

05 Clustering with Kmean algo

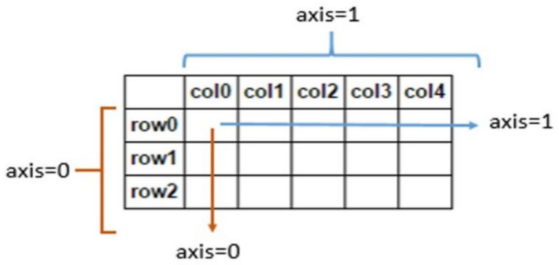
06 Dimension reduction

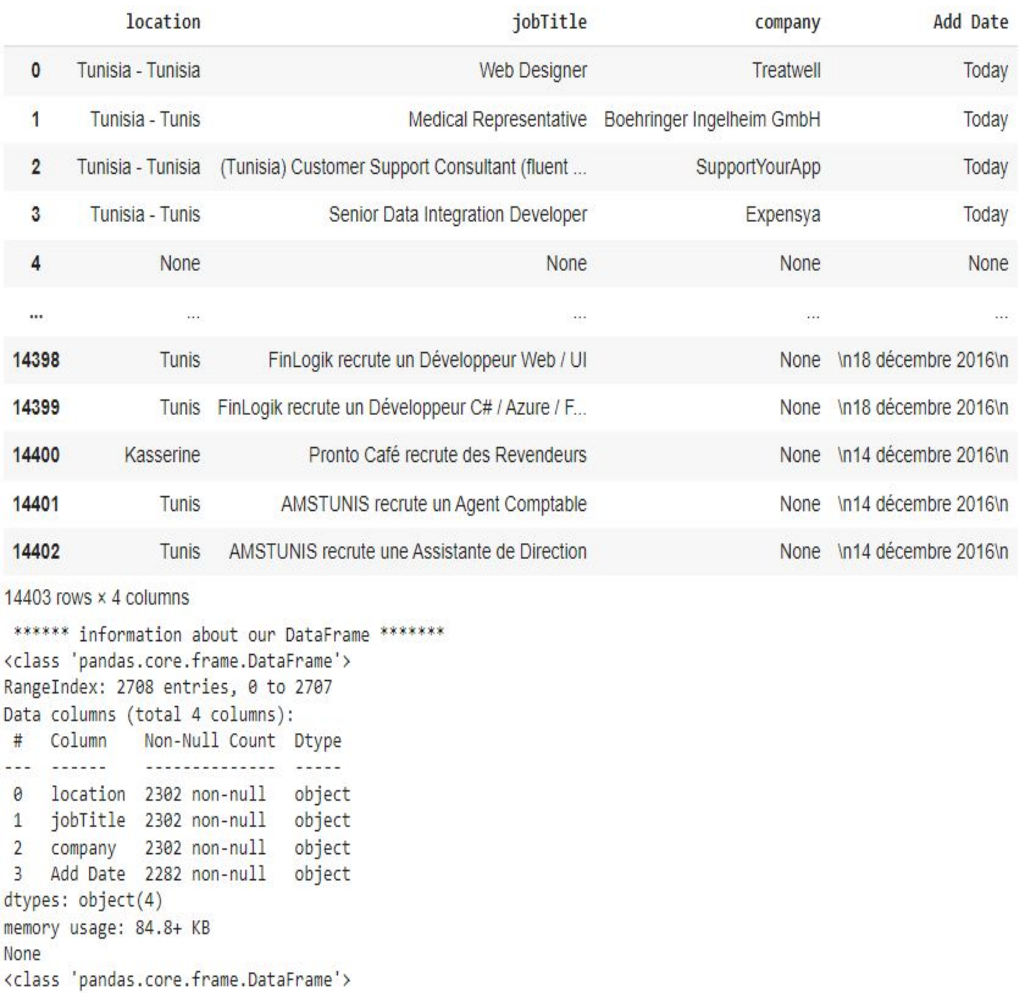
01 Create a Pandas Dataframe



```
[ ] import pandas as pd
    df1=pd.read_json("/content/jobs_1.json")
    df2=pd.read_json("/content/jobs_2.json")
    MyDataFrame =pd.concat([df1, df2],ignore_index=True, sort=False, axis=0)
    display(MyDataFrame)
    print(" ****** information about our DataFrame ******* ")
    print(df1.info())
    print(type(MyDataFrame))
```

- Pandas DataFrame is a 2-dimensional labeled data structure like any table with rows and columns.
- pandas.concat used to concatenate pandas objects along a particular axis.
- axis = 0 is the axis to concatenate along



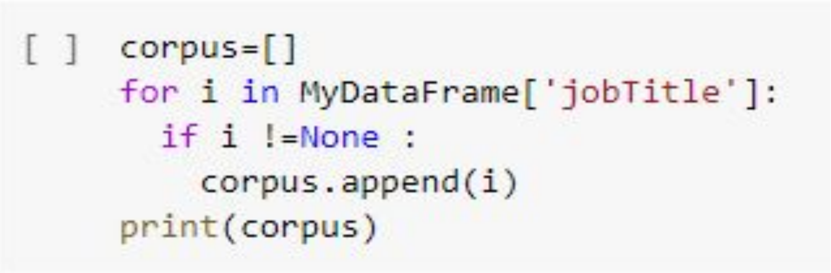


	location	jobTitle	company	Add Date
0	Tunisia - Tunisia	Web Designer	Treatwell	Today
1	Tunisia - Tunis	Medical Representative	Boehringer Ingelheim GmbH	Today
2	Tunisia - Tunisia	(Tunisia) Customer Support Consultant (fluent ...	SupportYourApp	Today
3	Tunisia - Tunis	Senior Data Integration Developer	Expensya	Today
4	None	None	None	None
...	...	...	...	...
14398	Tunis	FinLogik recrute un Développeur Web / UI	None	\n18 décembre 2016\n
14399	Tunis	FinLogik recrute un Développeur C# / Azure / F...	None	\n18 décembre 2016\n
14400	Kasserine	Pronto Café recrute des Revendeurs	None	\n14 décembre 2016\n
14401	Tunis	AMSTUNIS recrute un Agent Comptable	None	\n14 décembre 2016\n
14402	Tunis	AMSTUNIS recrute une Assistante de Direction	None	\n14 décembre 2016\n

14403 rows × 4 columns

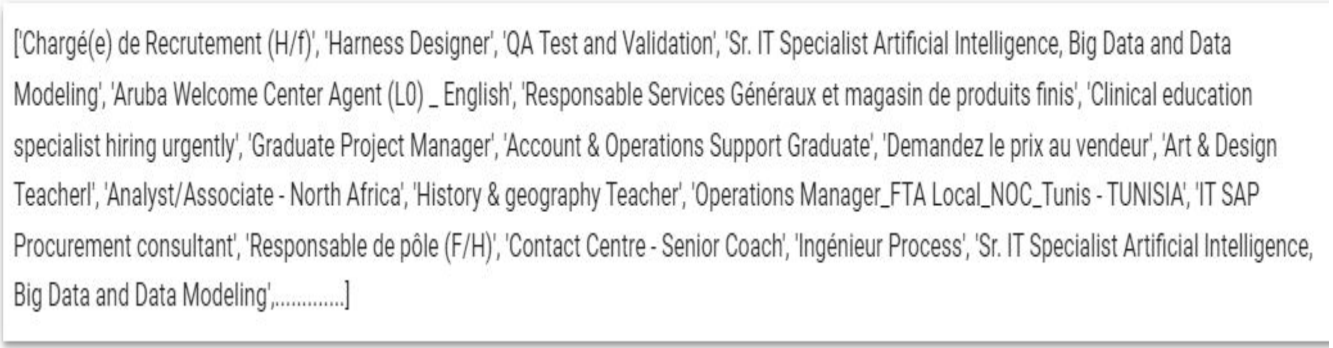
```
 ****** information about our DataFrame *******
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2708 entries, 0 to 2707
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   location  2302 non-null   object
 1   jobTitle  2302 non-null   object
 2   company   2302 non-null   object
 3   Add Date  2282 non-null   object
dtypes: object(4)
memory usage: 84.8+ KB
None
<class 'pandas.core.frame.DataFrame'>
```

02 Create a corpus



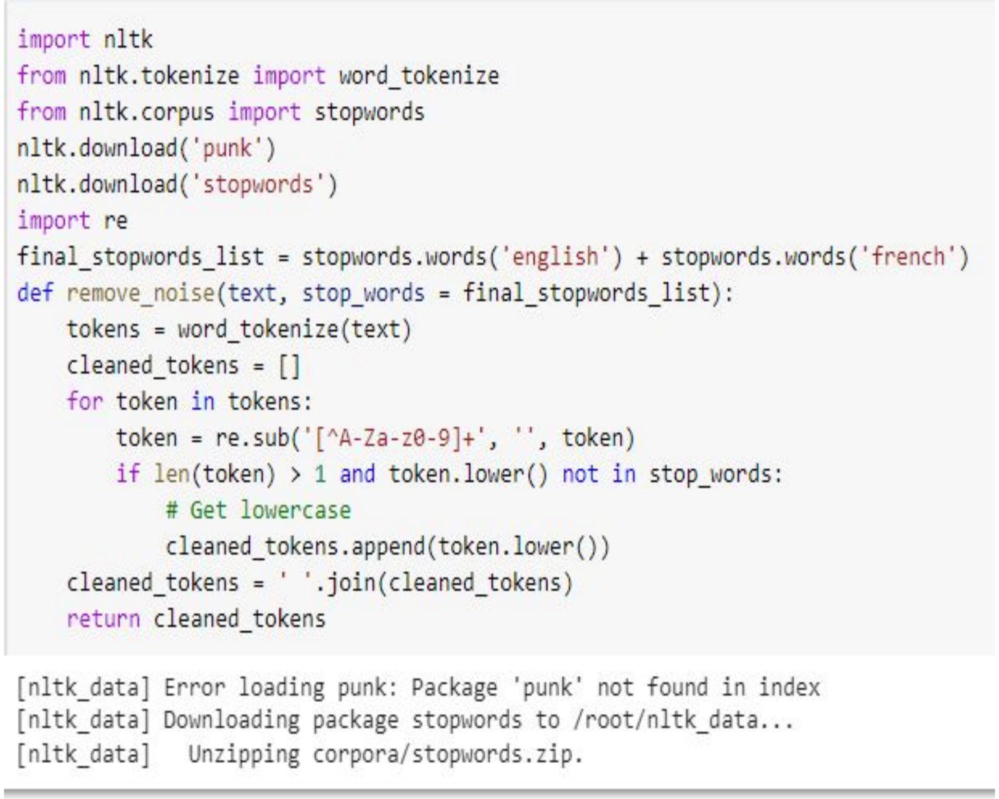
```
[ ]  corpus=[]
     for i in MyDataFrame['jobTitle']:
       if i !=None :
         corpus.append(i)
     print(corpus)
```

The corpus is a list of job titles



```
['Chargé(e) de Recrutement (H/f)', 'Harness Designer', 'QA Test and Validation', 'Sr. IT Specialist Artificial Intelligence, Big Data and Data
Modeling', 'Aruba Welcome Center Agent (L0) _ English', 'Responsable Services Généraux et magasin de produits finis', 'Clinical education
specialist hiring urgently', 'Graduate Project Manager', 'Account & Operations Support Graduate', 'Demandez le prix au vendeur', 'Art & Design
Teacherl', 'Analyst/Associate - North Africa', 'History & geography Teacher', 'Operations Manager_FTA Local_NOC_Tunis - TUNISIA', 'IT SAP
Procurement consultant', 'Responsable de pôle (F/H)', 'Contact Centre - Senior Coach', 'Ingénieur Process', 'Sr. IT Specialist Artificial Intelligence,
Big Data and Data Modeling',............]
```

03 Remove noise function



```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
nltk.download('punk')
nltk.download('stopwords')
import re
final_stopwords_list = stopwords.words('english') + stopwords.words('french')
def remove_noise(text, stop_words = final_stopwords_list):
    tokens = word_tokenize(text)
    cleaned_tokens = []
    for token in tokens:
        token = re.sub('[^A-Za-z0-9]+', '', token)
        if len(token) > 1 and token.lower() not in stop_words:
            # Get lowercase
            cleaned_tokens.append(token.lower())
    cleaned_tokens = ' '.join(cleaned_tokens)
    return cleaned_tokens
```

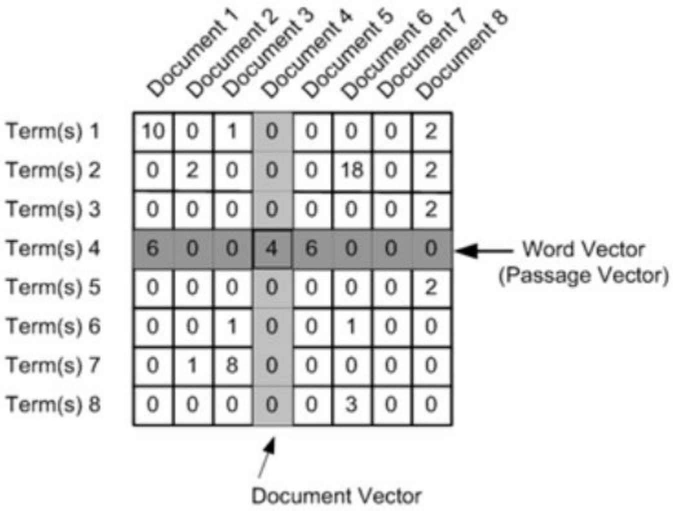
```
[nltk_data] Error loading punk: Package 'punk' not found in index
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

- The text in itself cannot analysed before creating into smaller parts called tokens by using NLTK work-tokenize method

- Remove all special characters from tokens
- Check if it contains to any stop words
- Return the cleaned tokens

04 Creating Tfi-idf matrix

- Document term matrix formed
- Most elements in matrix are zeros

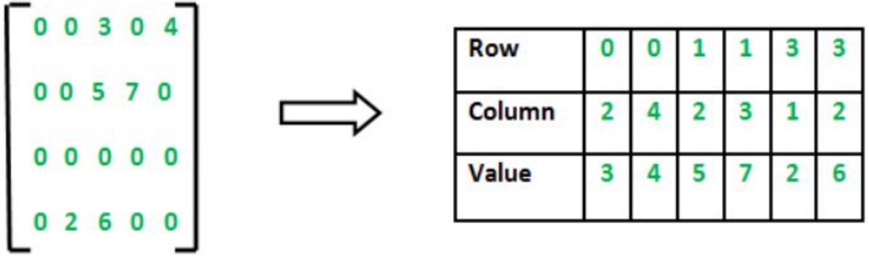


	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Word Vector
(Passage Vector)
Document Vector

Source

- Sparse matrix is created

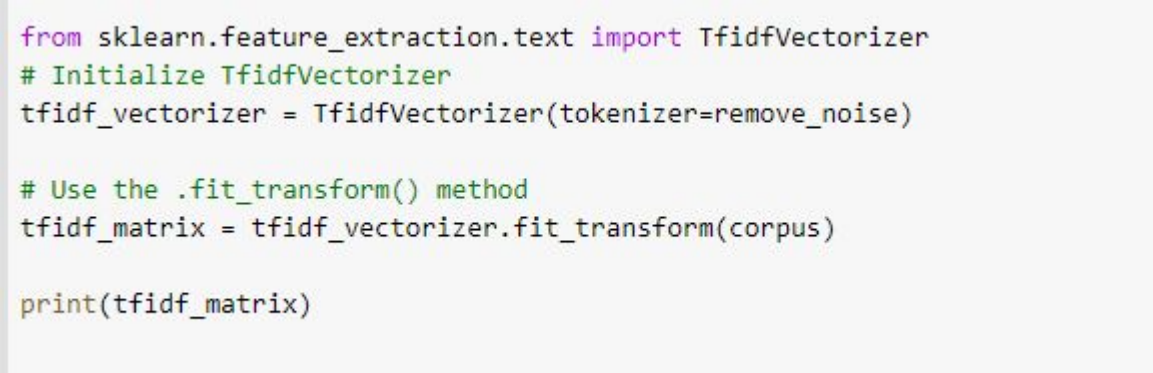


$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix} \Rightarrow$$

Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

Source

-matrix formed with the terms and documents as dimensions
-an element of the matrix signifies how many times a term has occurred in each document

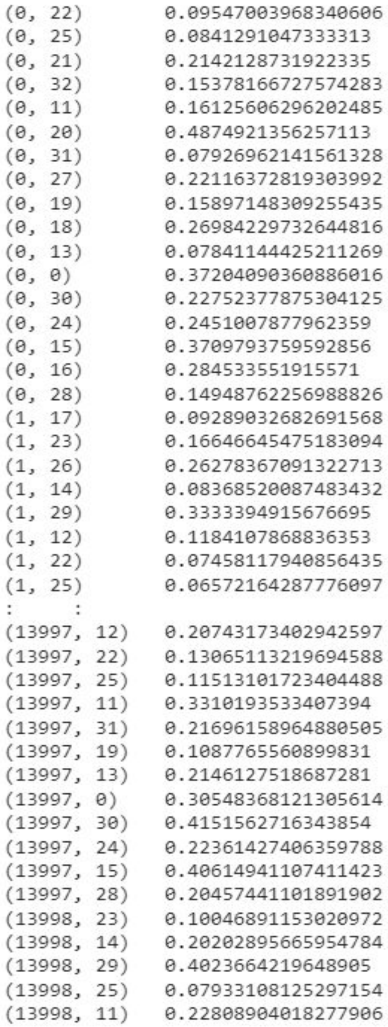


```python
from sklearn.feature_extraction.text import TfidfVectorizer
# Initialize TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(tokenizer=remove_noise)

# Use the .fit_transform() method
tfidf_matrix = tfidf_vectorizer.fit_transform(corpus)

print(tfidf_matrix)
```

-To find the tf-idf of terms in a group of documents , we use the tf-idf vectorizer class of Sklearn

-The fit_transform method creates the tf-idf matrix for the data which is sparse matrix

-a sparse matrix only contains terms which have none zero elements



```
  (0, 22)	0.09547003968340606
  (0, 25)	0.0841291047333313
  (0, 21)	0.214212873192235
  (0, 32)	0.15378166727574283
  (0, 11)	0.16125606296202485
  (0, 20)	0.487492135625711 3
  (0, 31)	0.07926962141561328
  (0, 27)	0.22116372819303992
  (0, 19)	0.1589714830925543 5
  (0, 18)	0.26984229732644816
  (0, 13)	0.07841144425211269
  (0, 0)	0.37204090360886016
  (0, 30)	0.22752377875304125
  (0, 24)	0.245100787796235 9
  (0, 15)	0.370979375959285 6
  (0, 16)	0.28453355191557 1
  (0, 28)	0.14948762256988826
  (1, 17)	0.09289032682691568
  (1, 23)	0.16646645475183094
  (1, 26)	0.26278367091322713
  (1, 14)	0.08368520087483432
  (1, 29)	0.333339491567669 5
  (1, 12)	0.118410786883635 3
  (1, 22)	0.07458117940856435
  (1, 25)	0.06572164287776097
  :	:
  (13997, 12)	0.20743173402942597
  (13997, 22)	0.13065113219694588
  (13997, 25)	0.11513101723404488
  (13997, 11)	0.3310193533407394
  (13997, 31)	0.21696158964880505
  (13997, 19)	0.1087765560899831
  (13997, 13)	0.2146127518687281
  (13997, 0)	0.30548368121305614
  (13997, 30)	0.4151562716343854
  (13997, 24)	0.22361427406359788
  (13997, 15)	0.40614941107411423
  (13997, 28)	0.20457441101891902
  (13998, 23)	0.10046891153020972
  (13998, 14)	0.20202895665954784
  (13998, 29)	0.4023664219648905
  (13998, 25)	0.07933108125297154
  (13998, 11)	0.22808904018277906
```

05 Clustering with Kmean algo

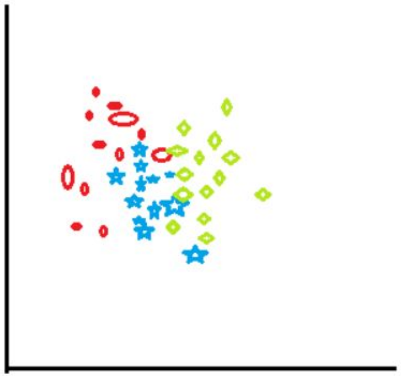


fig 1: before applying k-means clustering

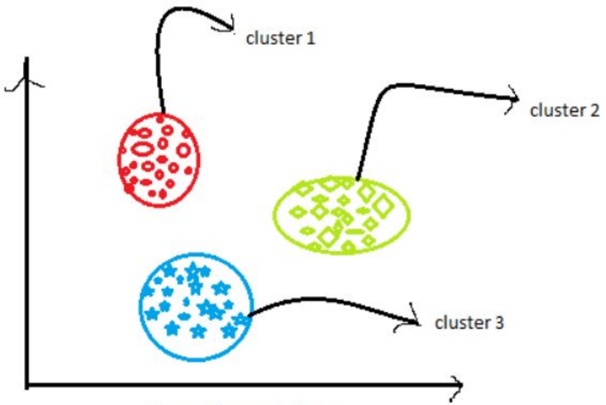
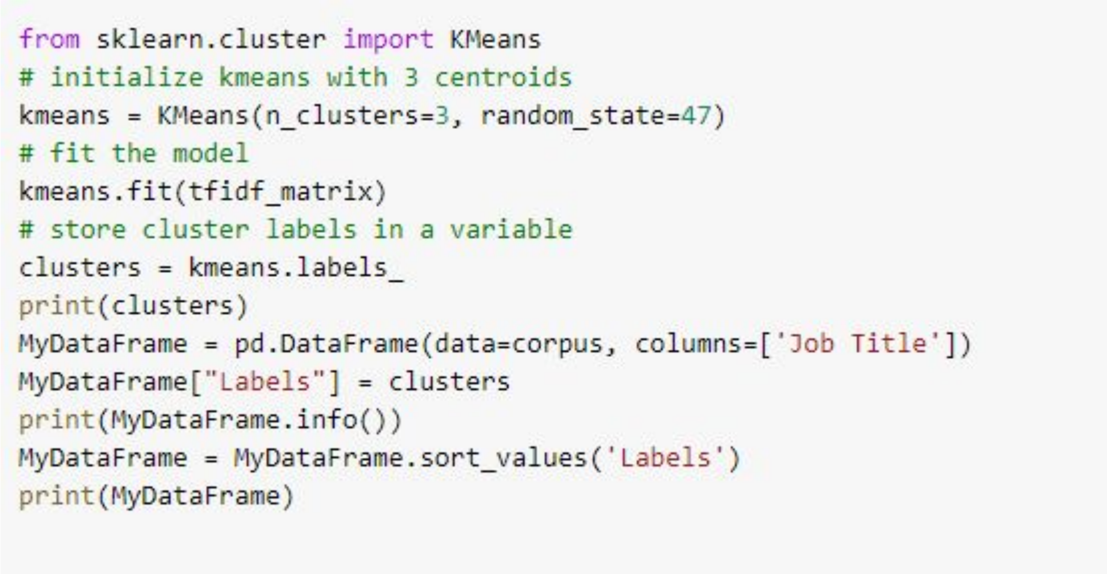
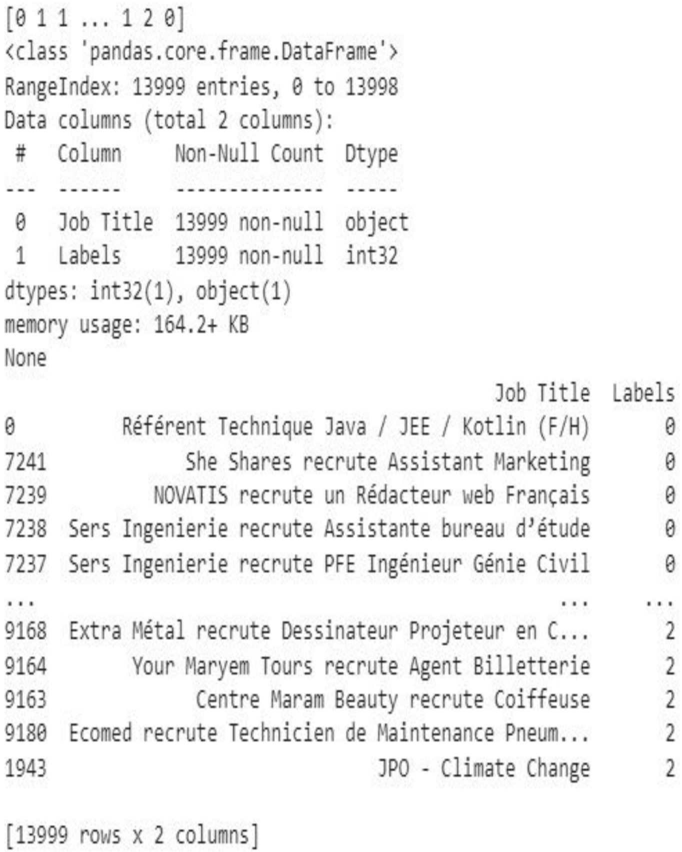


fig 2: After applying K-means clustering

- Finds clusters of samples
- Number of clusters must be specified
- Implemented in sklearn or scikit-learn



```python
from sklearn.cluster import KMeans
# initialize kmeans with 3 centroids
kmeans = KMeans(n_clusters=3, random_state=47)
# fit the model
kmeans.fit(tfidf_matrix)
# store cluster labels in a variable
clusters = kmeans.labels_
print(clusters)
MyDataFrame = pd.DataFrame(data=corpus, columns=['Job Title'])
MyDataFrame["Labels"] = clusters
print(MyDataFrame.info())
MyDataFrame = MyDataFrame.sort_values('Labels')
print(MyDataFrame)
```

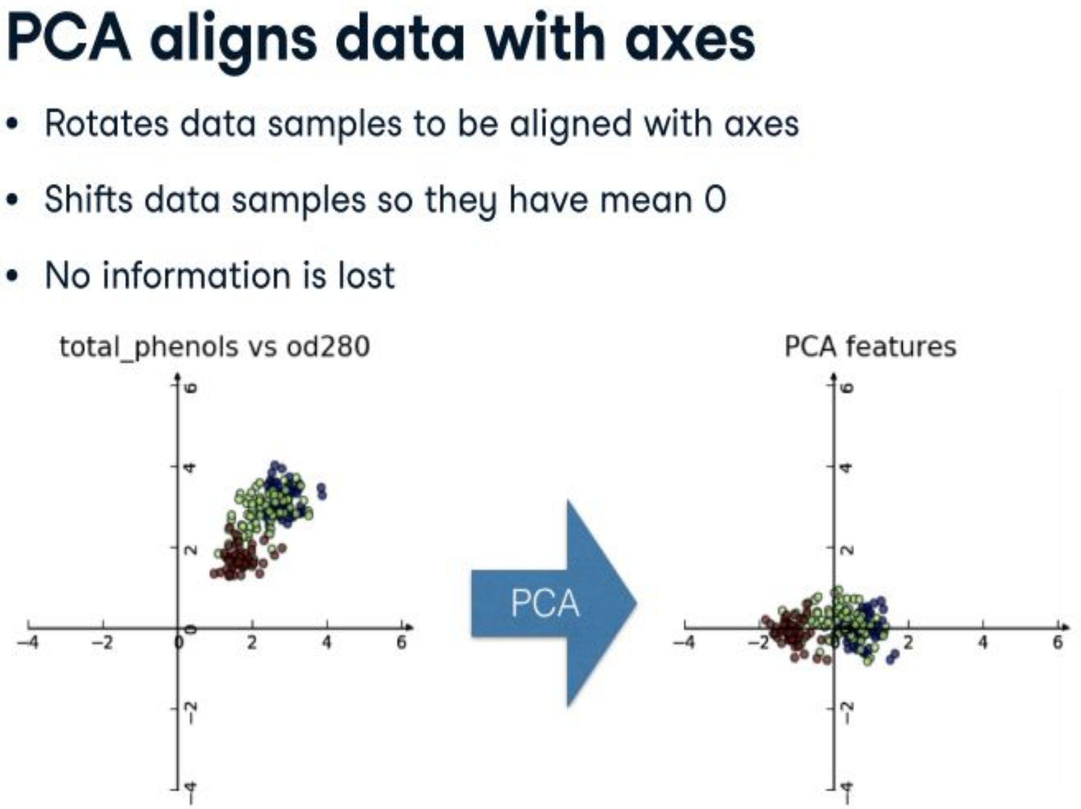


```
[0 1 1 ... 1 2 0]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13999 entries, 0 to 13998
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Job Title  13999 non-null  object
 1   Labels     13999 non-null  int32 
dtypes: int32(1), object(1)
memory usage: 164.2+ KB
None
                                             Job Title  Labels
0           Référent Technique Java / JEE / Kotlin (F/H)       0
7241             She Shares recrute Assistant Marketing       0
7239          NOVATIS recrute un Rédacteur web Français       0
7238  Sers Ingenierie recrute Assistante bureau d’étude       0
7237  Sers Ingenierie recrute PFE Ingénieur Génie Civil       0
...                                                ...     ...
9168  Extra Métal recrute Dessinateur Projeteur en C...       2
9164         Your Maryem Tours recrute Agent Billetterie       2
9163              Centre Maram Beauty recrute Coiffeuse       2
9180  Ecomed recrute Technicien de Maintenance Pneum...       2
1943                                JPO - Climate Change       2

[13999 rows x 2 columns]
```

06 Dimension reduction

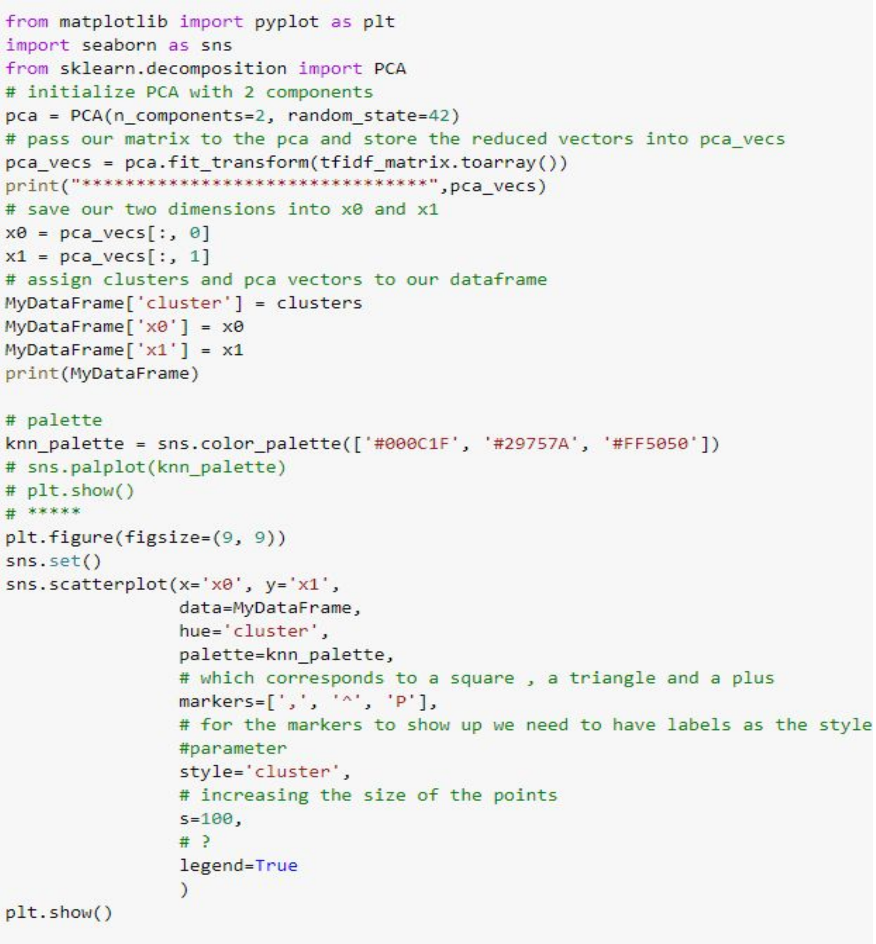
-Dimension reduction finds patterns in data and uses these patterns to re-express it in a compressed form

-Principal Component A analysis (PCA) is a dimensionality reduction technique used to transform high-dimensional datasets into a dataset with fewer variables

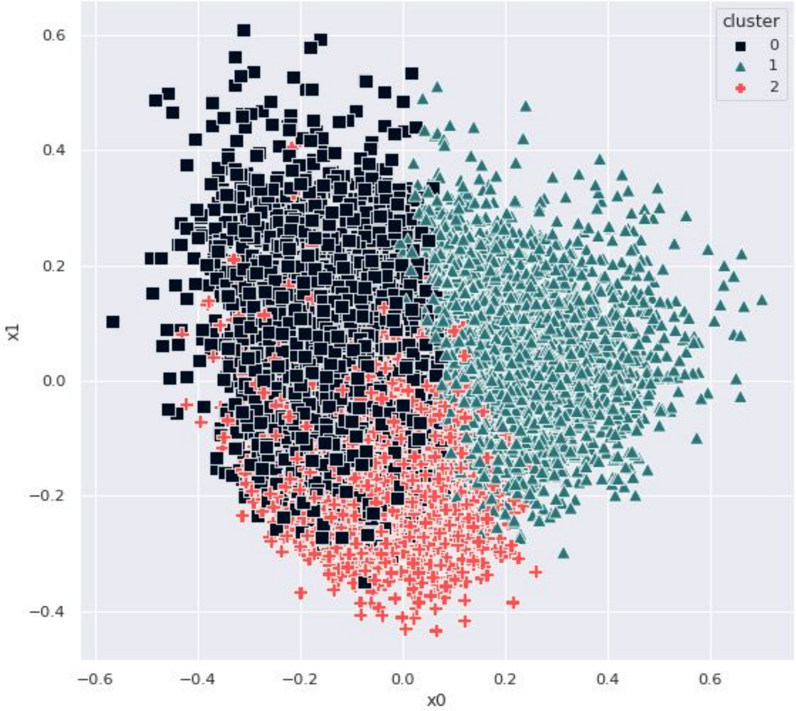


PCA aligns data with axes

- Rotates data samples to be aligned with axes
- Shifts data samples so they have mean 0
- No information is lost



```
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
# initialize PCA with 2 components
pca = PCA(n_components=2, random_state=42)
# pass our matrix to the pca and store the reduced vectors into pca_vecs
pca_vecs = pca.fit_transform(tfidf_matrix.toarray())
print("********************************",pca_vecs)
# save our two dimensions into x0 and x1
x0 = pca_vecs[:, 0]
x1 = pca_vecs[:, 1]
# assign clusters and pca vectors to our dataframe
MyDataFrame['cluster'] = clusters
MyDataFrame['x0'] = x0
MyDataFrame['x1'] = x1
print(MyDataFrame)

# palette
knn_palette = sns.color_palette(['#000C1F', '#29757A', '#FF5050'])
# sns.palplot(knn_palette)
# plt.show()
# *****
plt.figure(figsize=(9, 9))
sns.set()
sns.scatterplot(x='x0', y='x1',
                data=MyDataFrame,
                hue='cluster',
                palette=knn_palette,
                # which corresponds to a square , a triangle and a plus
                markers=[',', '^', 'P'],
                # for the markers to show up we need to have labels as the style
                #parameter
                style='cluster',
                # increasing the size of the points
                s=100,
                # ?
                legend=True
                )
plt.show()
```



PySpark

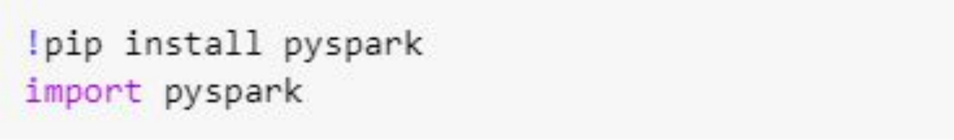
01 Creating a spark context and reading the Data

02 Creating a pipeline and clustering with kmean algo

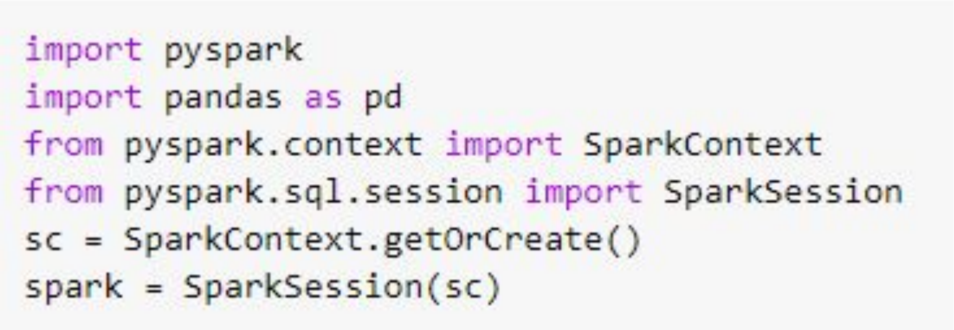
PySpark

PySpark is the Python API for Apache Spark, an open source, distributed computing framework and set of libraries for real-time, large-scale data processing. If you're already familiar with Python and libraries such as Pandas, then PySpark is a good language to learn to create more scalable analyses and pipelines.

01 Creating a spark context and reading the Data

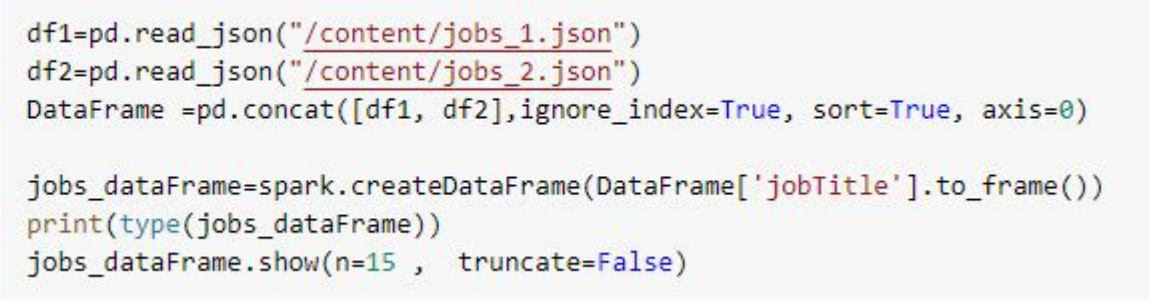


```
!pip install pyspark
import pyspark
```

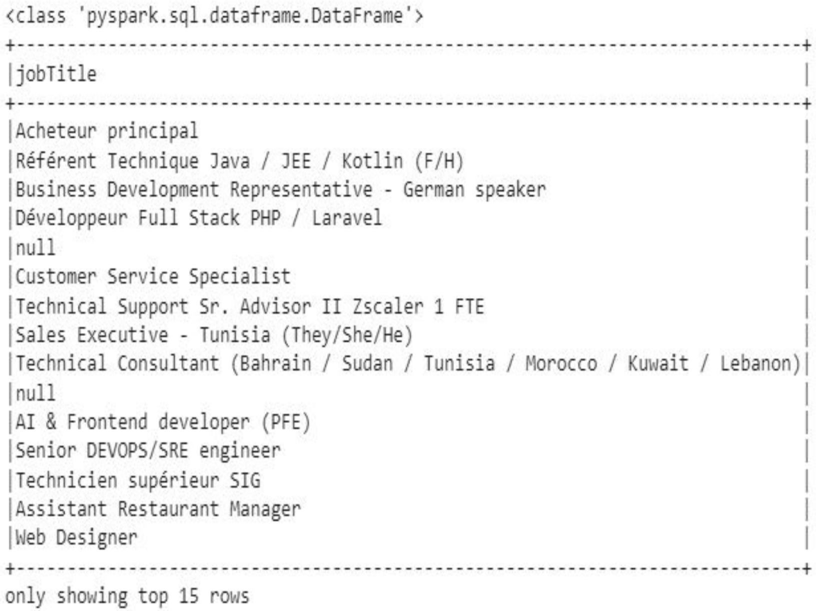


```
import pyspark
import pandas as pd
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
sc = SparkContext.getOrCreate()
spark = SparkSession(sc)
```

- Installing the PySpark
- creating a new Spark context and a new spark session



```
df1=pd.read_json("/content/jobs_1.json")
df2=pd.read_json("/content/jobs_2.json")
DataFrame =pd.concat([df1, df2],ignore_index=True, sort=True, axis=0)

jobs_dataFrame=spark.createDataFrame(DataFrame['jobTitle'].to_frame())
print(type(jobs_dataFrame))
jobs_dataFrame.show(n=15 ,  truncate=False)
```

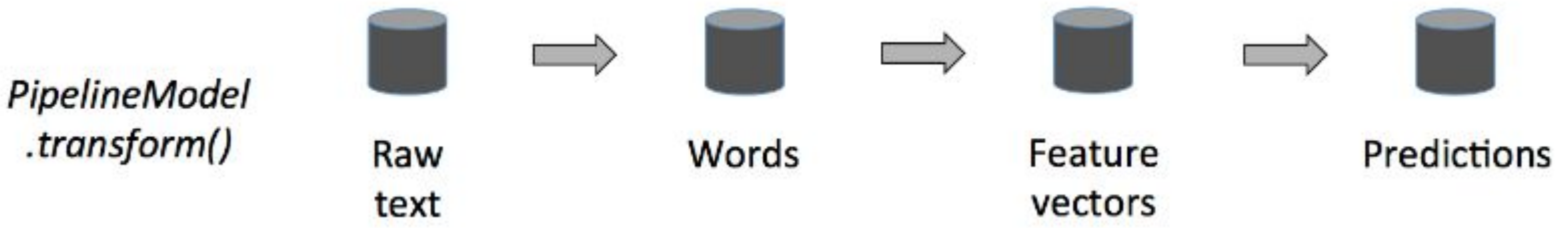


```
<class 'pyspark.sql.dataframe.DataFrame'>
+-----------------------------------------------------------------------------+
|jobTitle                                                                     |
+-----------------------------------------------------------------------------+
|Acheteur principal                                                           |
|Référent Technique Java / JEE / Kotlin (F/H)                                 |
|Business Development Representative - German speaker                         |
|Développeur Full Stack PHP / Laravel                                         |
|null                                                                         |
|Customer Service Specialist                                                  |
|Technical Support Sr. Advisor II Zscaler 1 FTE                               |
|Sales Executive - Tunisia (They/She/He)                                      |
|Technical Consultant (Bahrain / Sudan / Tunisia / Morocco / Kuwait / Lebanon)|
|null                                                                         |
|AI & Frontend developer (PFE)                                                |
|Senior DEVOPS/SRE engineer                                                   |
|Technicien supérieur SIG                                                     |
|Assistant Restaurant Manager                                                 |
|Web Designer                                                                 |
+-----------------------------------------------------------------------------+
only showing top 15 rows
```

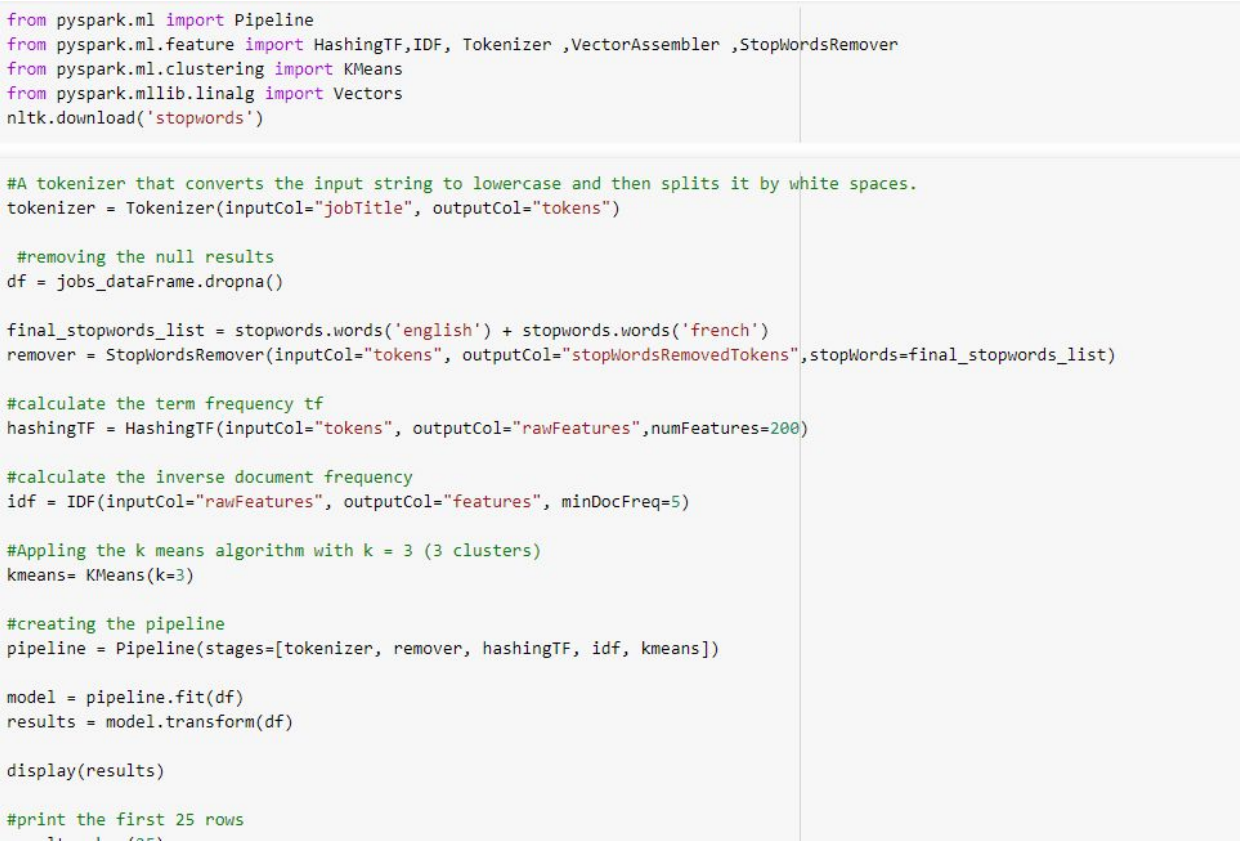
- Read the data from the data file using pandas
- Scrape job offers from farojob and tunisia.tanqeeb so we are creating the data frame

02

Creating a pipeline and clustering with kmean algo



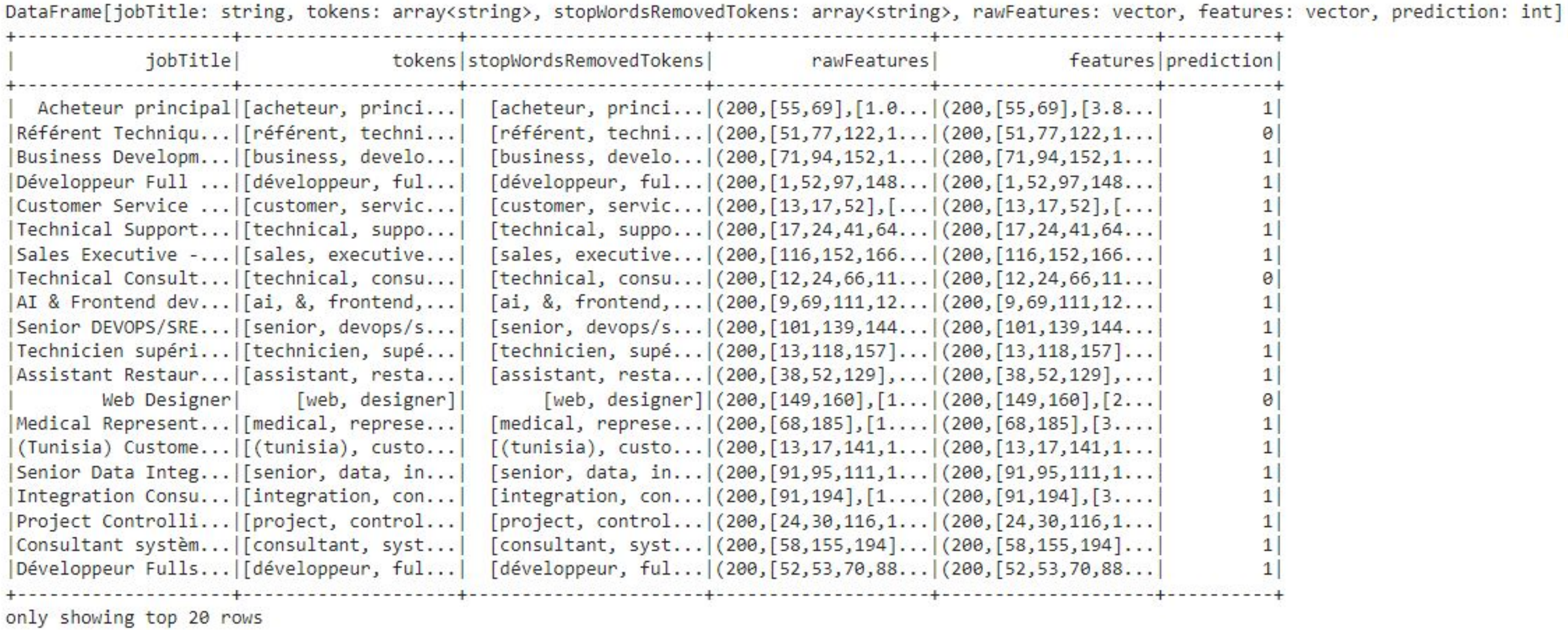
- A Pipeline is a specified as a sequence of stages , and each stage is either a transformer or an estimator. These stages are run in order , and the input Data Frame is transformed as it passes through each stage



```python
from pyspark.ml import Pipeline
from pyspark.ml.feature import HashingTF,IDF, Tokenizer ,VectorAssembler ,StopWordsRemover
from pyspark.ml.clustering import KMeans
from pyspark.mllib.linalg import Vectors
nltk.download('stopwords')
```

```python
#A tokenizer that converts the input string to lowercase and then splits it by white spaces.
tokenizer = Tokenizer(inputCol="jobTitle", outputCol="tokens")

 #removing the null results
df = jobs_dataFrame.dropna()

final_stopwords_list = stopwords.words('english') + stopwords.words('french')
remover = StopWordsRemover(inputCol="tokens", outputCol="stopWordsRemovedTokens",stopWords=final_stopwords_list)

#calculate the term frequency tf
hashingTF = HashingTF(inputCol="tokens", outputCol="rawFeatures",numFeatures=200)

#calculate the inverse document frequency
idf = IDF(inputCol="rawFeatures", outputCol="features", minDocFreq=5)

#Appling the k means algorithm with k = 3 (3 clusters)
kmeans= KMeans(k=3)

#creating the pipeline
pipeline = Pipeline(stages=[tokenizer, remover, hashingTF, idf, kmeans])

model = pipeline.fit(df)
results = model.transform(df)

display(results)

#print the first 25 rows
```

Creating the tokenizer ⇒ Applying the tf and idf ⇒ Creating the pipeline ⇒ applying the Kmean algorithm



```
DataFrame[jobTitle: string, tokens: array<string>, stopWordsRemovedTokens: array<string>, rawFeatures: vector, features: vector, prediction: int]
+--------------------+--------------------+----------------------+--------------------+--------------------+----------+
|            jobTitle|              tokens|stopWordsRemovedTokens|         rawFeatures|            features|prediction|
+--------------------+--------------------+----------------------+--------------------+--------------------+----------+
|  Acheteur principal|[acheteur, princi...|  [acheteur, princi...|(200,[55,69],[1.0...|(200,[55,69],[3.8...|         1|
|Référent Techniqu...|[référent, techni...|  [référent, techni...|(200,[51,77,122,1...|(200,[51,77,122,1...|         0|
|Business Developm...|[business, develo...|  [business, develo...|(200,[71,94,152,1...|(200,[71,94,152,1...|         1|
|Développeur Full ...|[développeur, ful...|  [développeur, ful...|(200,[1,52,97,148...|(200,[1,52,97,148...|         1|
|Customer Service ...|[customer, servic...|  [customer, servic...|(200,[13,17,52],[...|(200,[13,17,52],[...|         1|
|Technical Support...|[technical, suppo...|  [technical, suppo...|(200,[17,24,41,64...|(200,[17,24,41,64...|         1|
|Sales Executive -...|[sales, executive...|  [sales, executive...|(200,[116,152,166...|(200,[116,152,166...|         1|
|Technical Consult...|[technical, consu...|  [technical, consu...|(200,[12,24,66,11...|(200,[12,24,66,11...|         0|
|AI & Frontend dev...|[ai, &, frontend,...|  [ai, &, frontend,...|(200,[9,69,111,12...|(200,[9,69,111,12...|         1|
|Senior DEVOPS/SRE...|[senior, devops/s...|  [senior, devops/s...|(200,[101,139,144...|(200,[101,139,144...|         1|
|Technicien supéri...|[technicien, supé...|  [technicien, supé...|(200,[13,118,157]...|(200,[13,118,157]...|         1|
|Assistant Restaur...|[assistant, resta...|  [assistant, resta...|(200,[38,52,129],...|(200,[38,52,129],...|         1|
|        Web Designer|     [web, designer]|       [web, designer]|(200,[149,160],[1...|(200,[149,160],[2...|         0|
|Medical Represent...|[medical, represe...|  [medical, represe...|(200,[68,185],[1....|(200,[68,185],[3....|         1|
|(Tunisia) Custome...|[(tunisia), custo...|  [(tunisia), custo...|(200,[13,17,141,1...|(200,[13,17,141,1...|         1|
|Senior Data Integ...|[senior, data, in...|  [senior, data, in...|(200,[91,95,111,1...|(200,[91,95,111,1...|         1|
|Integration Consu...|[integration, con...|  [integration, con...|(200,[91,194],[1....|(200,[91,194],[3....|         1|
|Project Controlli...|[project, control...|  [project, control...|(200,[24,30,116,1...|(200,[24,30,116,1...|         1|
|Consultant systèm...|[consultant, syst...|  [consultant, syst...|(200,[58,155,194]...|(200,[58,155,194]...|         1|
|Développeur Fulls...|[développeur, ful...|  [développeur, ful...|(200,[52,53,70,88...|(200,[52,53,70,88...|         1|
+--------------------+--------------------+----------------------+--------------------+--------------------+----------+
only showing top 20 rows
```

pandas

matplotlib

seaborn

Scrapy

TOKENIZATION
IN
PYTHON

NLTK

scikit
learn

Thanks for your Attention

Amani Salah

Nidhal Naffati

Mohamed Ali Bouajila