

Document clustering in the context of Big Data

MSBDA-2021

Lab On: Data Clustering (Netflix Movies and TV Shows)

- Dataset Pre-processing.
- Feature Extraction.
- Cluster Analysis.
- Data visualization

1. Anaconda Installation and configuration

This part we will demonstrate how we can install Anaconda, a powerful package manager, on Microsoft Windows.

Anaconda is a package manager, an environment manager, and Python distribution that contains a collection of many open source packages. This is advantageous as when we are working on a data science project, we will find that we need many different packages (numpy, scikit-learn, scipy, pandas to name a few), which an installation of Anaconda comes preinstalled with. If we need additional packages after installing Anaconda, we can use Anaconda's package manager, conda, or pip to install those packages. This is highly advantageous as we don't have to manage dependencies between multiple packages yourself. Conda even makes it easy to switch between Python 2 and 3 (you can learn more here). In fact, an installation of Anaconda is also the recommended way to install Jupyter Notebooks .

Install Anaconda

- ✓ we Go to the Anaconda Website and we choose a Python 3.8 graphical installer.



les fenêtres			
Version Python	Nom	Taille	Hachage SHA256
Python 3.8	Miniconda3 Windows 64 bits	57,0 Mio	4fa22bba0497babb5b6608cb8843545372a99f5331c8120099ae1d803f627c61
	Miniconda3 Windows 32 bits	54,2 Mio	9c2ef76bae97246c85c206733ca30fd1feb8a4b3f90a2a511fea681ce7ebc661
Python 2.7	Miniconda2 Windows 64 bits	54,1 Mio	6973025404832944e074bf02bda8c4594980eed4707bb51baa8fdba4bf326c
	Miniconda2 Windows 32 bits	47,7 Mio	c8049d26f8b6b954b57bcd4e99ad72d1ffa13f4a6b218e64e641504437b2617b

Figure 6 :Installers windows

- ✓ When the screen below appears, we click on Next.



Figure 7 :Install anaconda

✓ We read the license agreement and we click on I Agree :

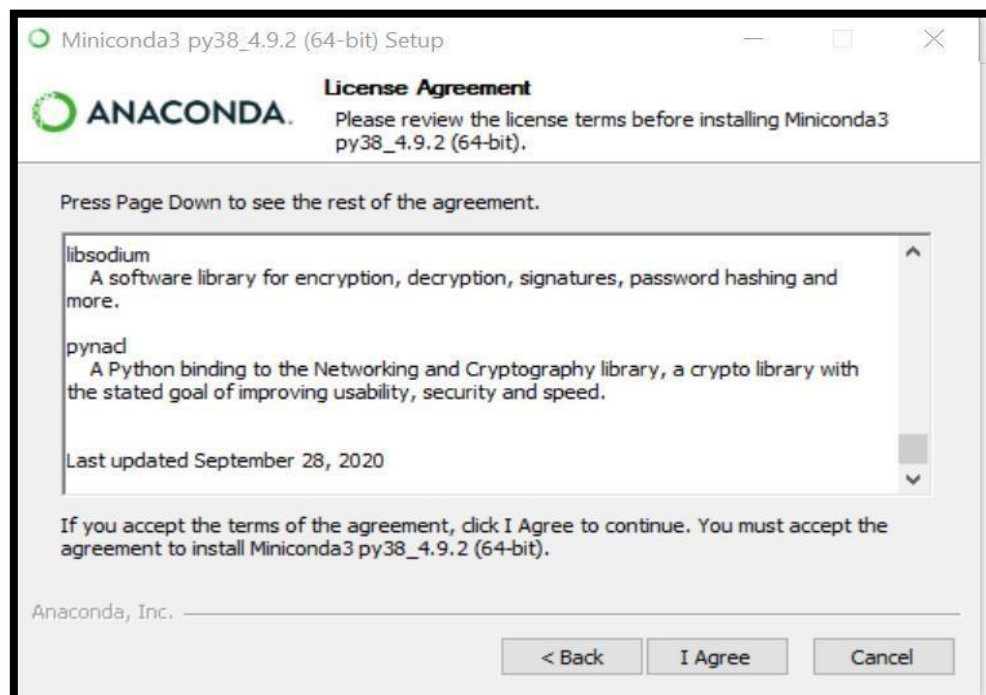


Figure 8: the license agreement

- ✓ We install just for us, we click on Next.

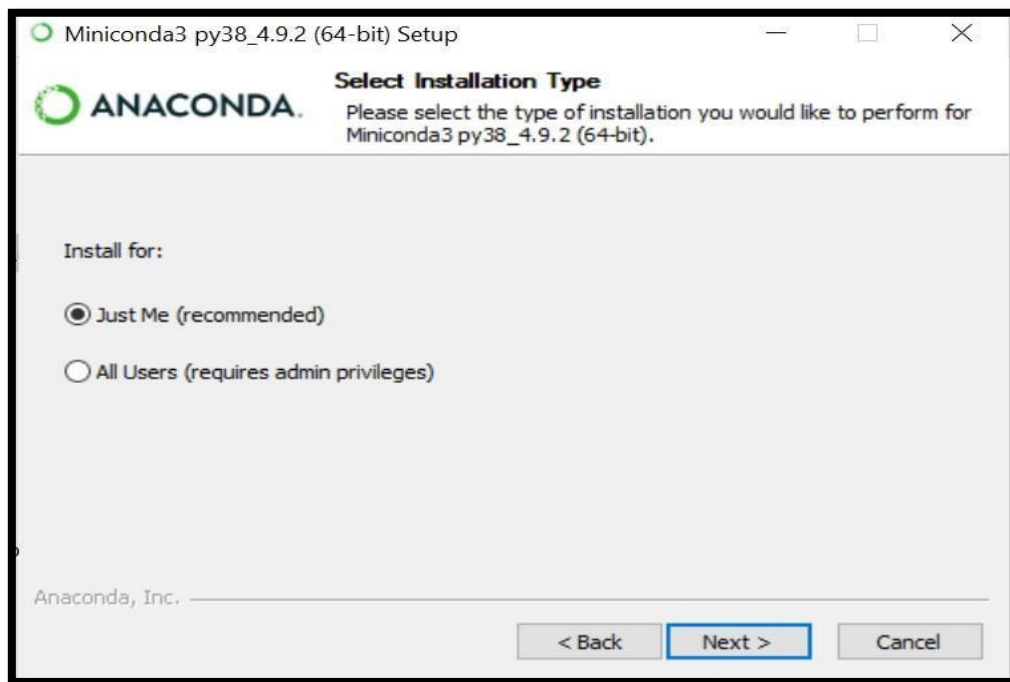


Figure 9 :Istallation anaconda

- ✓ We note our installation location and we click Next.

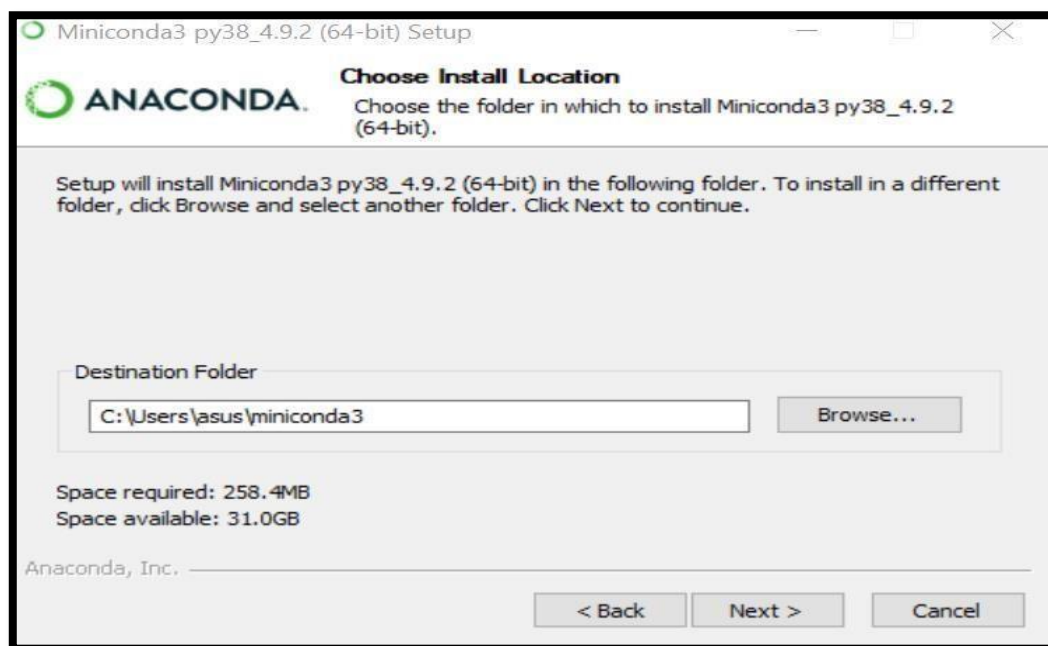


Figure 10 : Installation anaconda

- ✓ This is an important part of the installation process. We click on Install

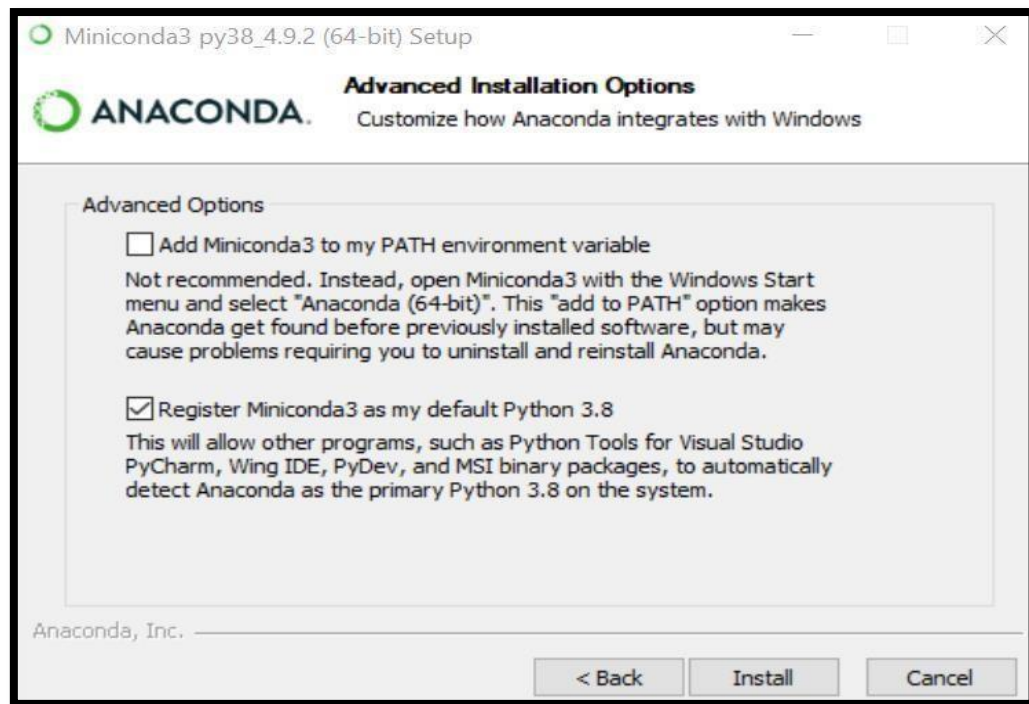


Figure 11 :advanced installation options

- ✓ Anaconda is being installed

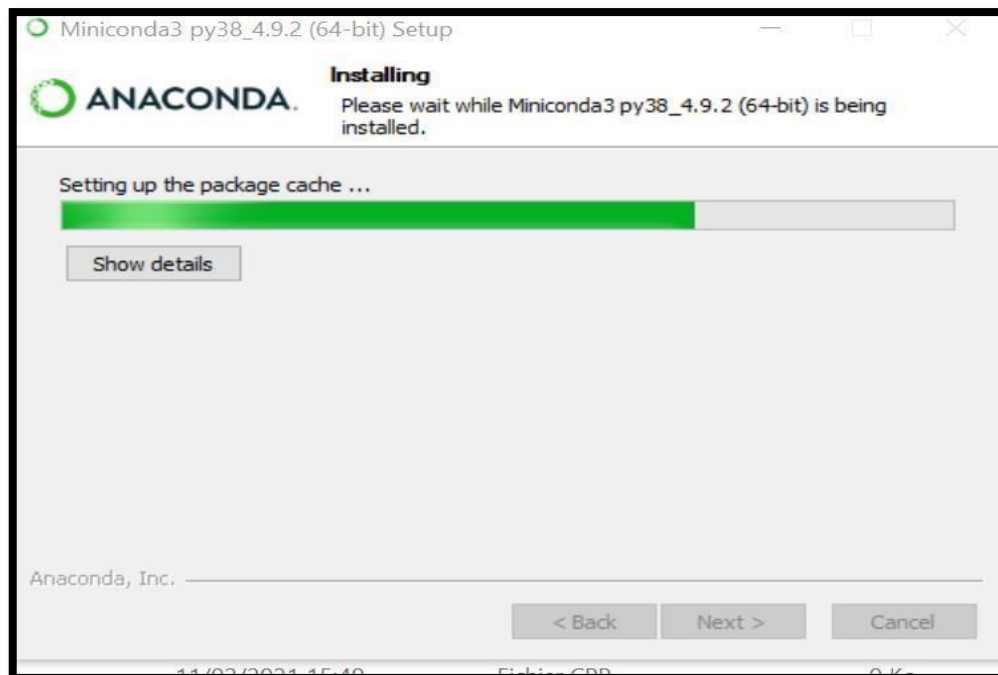


Figure 12:Anaconda is being installed

✓ We Click on Finish.

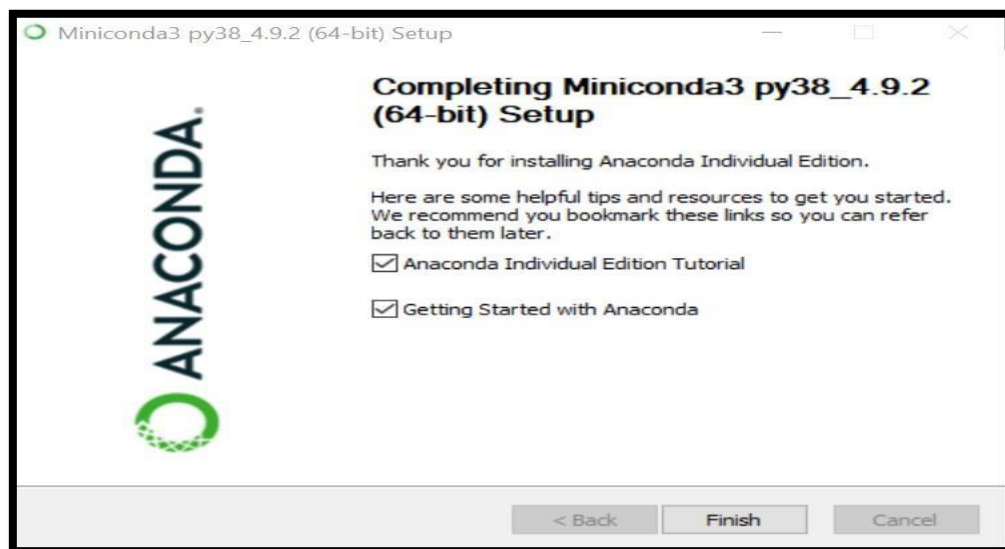


Figure 13 : Completing installation Miniconda

✓ Installation Anaconda success :

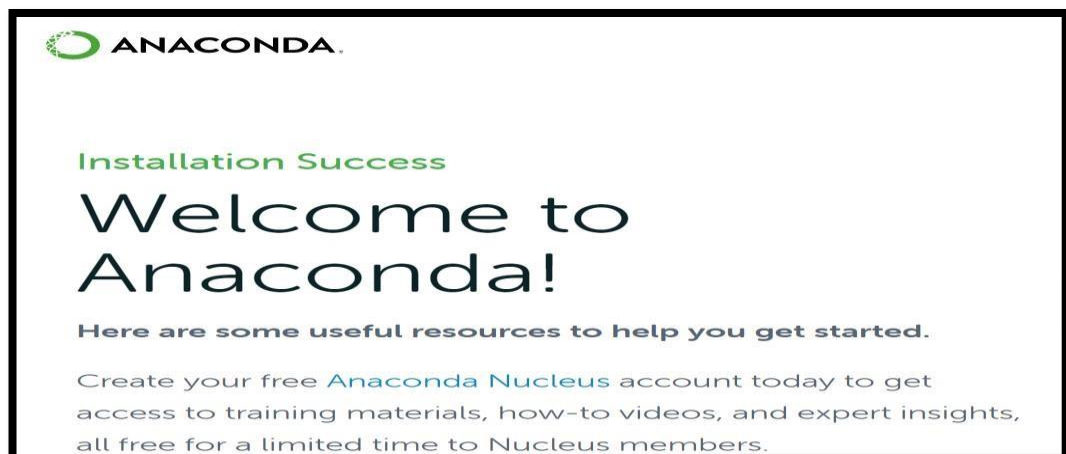


Figure 14 :Installation success

✓ We pass to configur Anaconda :activate

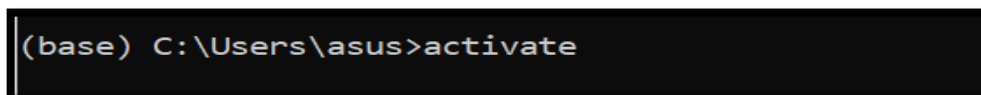


Figure15:Activate Anaconda

- ✓ We update anaconda

```
(base) C:\Users\asus>conda update --all
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\asus\miniconda3

The following packages will be downloaded:
```

package	build		
brotlipy-0.7.0	py38h294d835_1001	368 KB	conda-forge
ca-certificates-2020.12.5	h5b45459_0	173 KB	conda-forge
certifi-2020.12.5	py38haa244fe_1	144 KB	conda-forge
cffi-1.14.5	py38hd8c33c5_0	228 KB	conda-forge
chardet-4.0.0	py38haa244fe_1	224 KB	conda-forge
conda-4.9.2	py38haa244fe_0	3.1 MB	conda-forge
conda-package-handling-1.7.2	py38h8934438_0	732 KB	conda-forge
cryptography-3.4.4	py38hb7941b4_0	600 KB	conda-forge
idna-2.10	pyh9f0ad1d_0	52 KB	conda-forge
menuinst-1.4.16	py38h32f6830_1	95 KB	conda-forge
openssl-1.1.1j	h8ffe710_0	5.8 MB	conda-forge
pip-21.0.1	pyhd8ed1ab_0	1.1 MB	conda-forge
pycosat-0.6.3	py38h294d835_1006	102 KB	conda-forge
pycparser-2.20	pyh9f0ad1d_2	94 KB	conda-forge
pyopenssl-20.0.1	pyhd8ed1ab_0	48 KB	conda-forge
pysocks-1.7.1	py38haa244fe_3	28 KB	conda-forge
python-3.8.6	h7840368_5_cpython	19.1 MB	conda-forge
python_abi-3.8	1_cp38	4 KB	conda-forge
pywin32-300	py38h294d835_0	7.0 MB	conda-forge
requests-2.25.1	pvhd3deb0d_0	51 KB	conda-forge

Figure 16 :Update conda

- ✓ Getting started with JupyterLab

```
(base) C:\Users\asus>conda install jupyter
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\asus\miniconda3

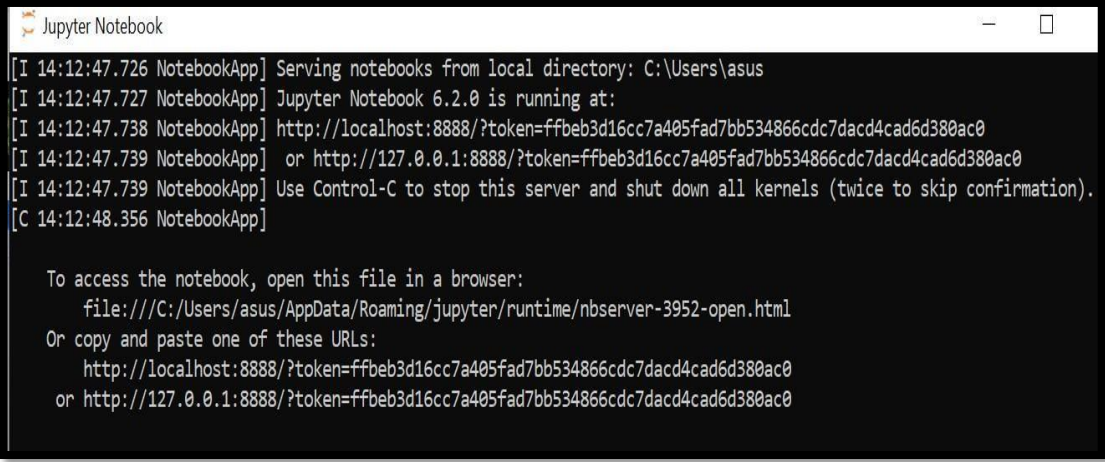
added / updated specs:
- jupyter

The following packages will be downloaded:
```

package	build		
argon2-cffi-20.1.0	py38h294d835_2	51 KB	conda-forge
async_generator-1.10	py_0	18 KB	conda-forge
attrs-20.3.0	pyhd3deb0d_0	41 KB	conda-forge
backcall-0.2.0	pyh9f0ad1d_0	13 KB	conda-forge
backports-1.0	py_2	4 KB	conda-forge
backports.functions_lru_cache-1.6.1	py_0	8 KB	conda-forge
bleach-3.3.0	pyh44b312d_0	111 KB	conda-forge
colorama-0.4.4	pyh9f0ad1d_0	18 KB	conda-forge
decorator-4.4.2	py_0	11 KB	conda-forge
defusedxml-0.6.0	py_0	22 KB	conda-forge
entrypoints-0.3	pyhd8ed1ab_1003	8 KB	conda-forge
icu-68.1	h0e60522_0	16.3 MB	conda-forge
importlib-metadata-3.4.0	py38haa244fe_0	21 KB	conda-forge
importlib_metadata-3.4.0	hd8ed1ab_0	3 KB	conda-forge

Figure 17 :Install Jupyter

- ✓ now we can access to home page for jupyter via l'url



```
Jupyter Notebook

[I 14:12:47.726 NotebookApp] Serving notebooks from local directory: C:\Users\asus
[I 14:12:47.727 NotebookApp] Jupyter Notebook 6.2.0 is running at:
[I 14:12:47.738 NotebookApp] http://localhost:8888/?token=ffbeb3d16cc7a405fad7bb534866cdc7dacd4cad6d380ac0
[I 14:12:47.739 NotebookApp] or http://127.0.0.1:8888/?token=ffbeb3d16cc7a405fad7bb534866cdc7dacd4cad6d380ac0
[I 14:12:47.739 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 14:12:48.356 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/asus/AppData/Roaming/jupyter/runtime/nbserver-3952-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=ffbeb3d16cc7a405fad7bb534866cdc7dacd4cad6d380ac0
    or http://127.0.0.1:8888/?token=ffbeb3d16cc7a405fad7bb534866cdc7dacd4cad6d380ac0
```

Figure 18 : jupyter Notebook

- ✓ This is home page fot Jupyter where we can write our programs

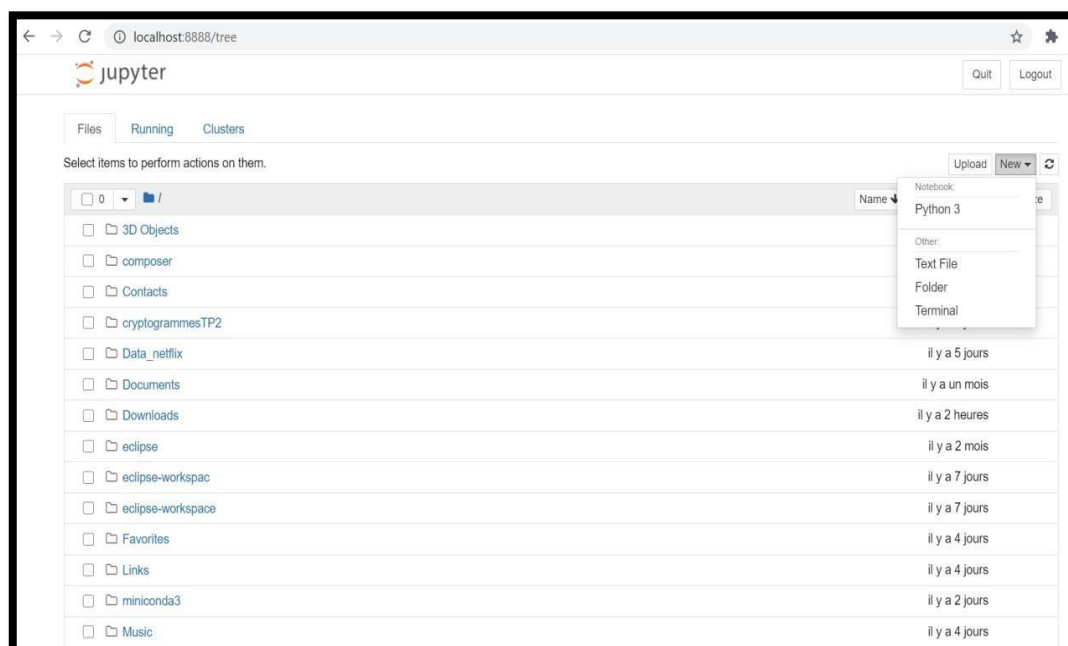


Figure 19 : Page home Jupyter

2. Program

2.1 Introduction

The idea behind this article is to understand how to represent text data (Data Netflix Movies and TV Shows) so that it can be used as input for the K-Means algorithm.

The whole process can be divided into the following stages:

- A. Pre-processing of the Dataset.
- B. Feature Extraction with TF-IDF.
- C. Running K-Means and Cluster Analysis.
- D. Data visualization

2.2 Implementation program

A. Pre-processing of the Dataset

Importing libraries

One of the key strengths of programming in Python, the wealth of libraries available can also be a source of concern for a beginner because it must be admitted, finding your way around at startup can be an impossible mission.

- ✓ The Python 3 environment comes with many helpful analytics libraries installed. for us we need some libraries to run our project for example pandas , numpy, nltk, sklearn, WordCloud...

```
Entrée [1]: import numpy as np
import pandas as pd

import re
import nltk.corpus
import nltk.tokenize
from nltk.tokenize import word_tokenize
from nltk.tokenize import word_tokenize
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import normalize

import plotly.graph_objs as go
from sklearn import cluster

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
from sklearn.metrics import silhouette_samples, silhouette_score
from wordcloud import WordCloud

from plotly.offline import download_plotlyjs, plot, init_notebook_mode, iplot
init_notebook_mode(connected=True)# initiate notebook for offline plot
```

Figure20: The libraries installed.

➤ **Pandas**

This library, widely used in data science, allows among other things to:

- Simplify data manipulation with data frames (missing values, columns, etc.)
- Allow to aggregate and merge data very easily with thanks to the group by
- Benefit from sophisticated and easy-to-use indexing

➤ **Numpy**

Numpy is a fundamental library for performing numerical calculations with Python. It greatly facilitates the management of data arrays with a large number of functions allowing to generate objects of type ndarray. These objects have the advantage of being easily manipulated and contain elements of the same type.

➤ **Scikit Learn**

Fundamental library, it offers a whole bunch of tools widely used in data science. Firstly, it will allow you to efficiently prepare your data to optimize the operation of machine learning algorithms. To do this, it provides numerous classes and functions. Thanks to them, you will be able in particular to reduce the size of your data set, center reduce your data. Secondly, Scikit Learn will give you the possibility of easily implementing Machine Learning algorithms related to the resolution of various problems(clustering, classification...)

➤ **Matplotlib**

Matplotlib is a powerful tool for plotting and visualizing data . It is intended to draw graphs of all kinds (circular diagram, histogram, cloud of points, etc.). Matplotlib contains

a pyplot sub-library which creates an interface similar to commercial Matlab software which contains functions very similar to this one.

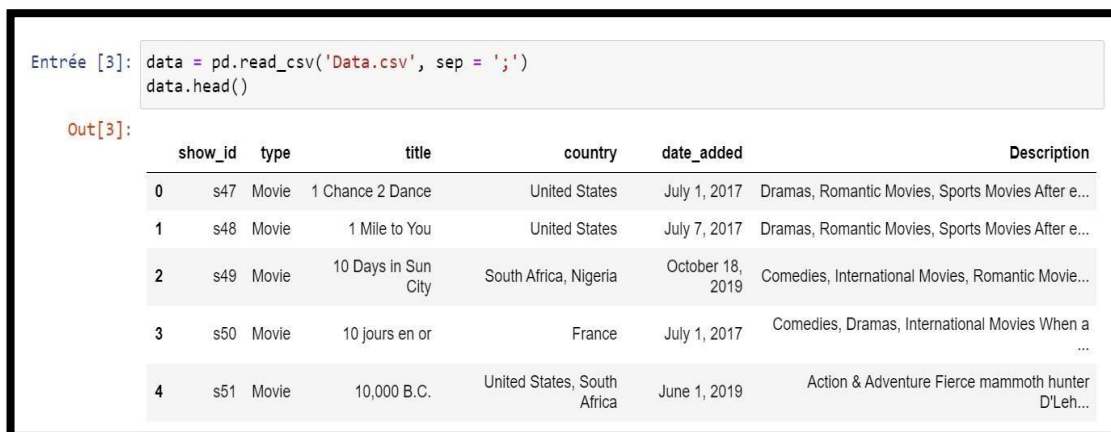
➤ Seaborn

Seaborn is a Python library for data visualization, specializing in statistical analysis. Based on the Matplotlib library, it is fully adapted to Pandas data frames. Thus, beyond a visually improved interface, Seaborn makes it possible to quickly and intuitively produce quality statistical graphics.

➤ NLTK (Text_mining)

NLTK is a fundamental library for building Python programs for working with human language data. It offers easy-to-use interfaces to corpora or lexical resources such as WordNet, as well as tools for word processing, classification, tokenization, stemming, markup, analysis and semantic reasoning.

We'll use pandas to read the csv file containing information concerning TV Shows and Movies added to the Netflix catalog. The dataset is collected from Flixable which is a third-party Netflix search engine.



```
Entrée [3]: data = pd.read_csv('Data.csv', sep = ';')
            data.head()
```

Out[3]:

	show_id	type	title	country	date_added	Description
0	s47	Movie	1 Chance 2 Dance	United States	July 1, 2017	Dramas, Romantic Movies, Sports Movies After e...
1	s48	Movie	1 Mile to You	United States	July 7, 2017	Dramas, Romantic Movies, Sports Movies After e...
2	s49	Movie	10 Days in Sun City	South Africa, Nigeria	October 18, 2019	Comedies, International Movies, Romantic Movie...
3	s50	Movie	10 jours en or	France	July 1, 2017	Comedies, Dramas, International Movies When a ...
4	s51	Movie	10,000 B.C.	United States, South Africa	June 1, 2019	Action & Adventure Fierce mammoth hunter D'Leh...

Figure 21: Show data

- ✓ Then we'll extract the **description** column into a list of texts for our corpus.

```
Entrée [4]: corpus = data['Description'].tolist()
           corpus[0:1000][0:447]

Out[4]: ['Dramas, Romantic Movies, Sports Movies After escaping the bus accident that killed his girlfriend, a high school student channels his grief into running, with the help of a new coach.',
        'Dramas, Romantic Movies, Sports Movies After escaping the bus accident that killed his girlfriend, a high school student channels his grief into running, with the help of a new coach.',
        'Comedies, International Movies, Romantic Movies After his girlfriend wins the Miss Nigeria pageant, a young man faces unexpected competition of his own when he joins her on a campaign in South Africa.',
        'Comedies, Dramas, International Movies When a carefree bachelor is unexpectedly left in charge of a young boy, the two embark on a road trip that will change both of their lives.',
        'Action & Adventure Fierce mammoth hunter D'Leh sets out on an impossible journey to rescue the woman he loves from a vicious warlord and save the people of his village.',
        'International TV Shows, Romantic TV Shows, TV Dramas Upon losing his memory, a crown prince encounters a commoner's life and experiences unforgettable love as the husband to Joseon's oldest ballerina.',
        'Documentaries, International Movies Spanish photographer JosÃ© Xadaz spends 100 days living alone on a remote mountain, connecting to nature and documenting the beauty of his surroundings.',
        'Docuseries, Science & Nature TV One hundred hardy souls from diverse backgrounds participate in playful experiments exploring age, sex, happiness and other aspects of being human.',
        'Dramas, International Movies, Sports Movies A man who is diagnosed with multiple sclerosis responds by training for an Ironman triathlon. with his cranky father-in-law's help.']
```

Figure 22: Extraction the description

🔗 Corpus Processing

We will do a data engineering routine with our description dataset so later we can make a good statistical model. In order to do so, we'll remove all words that don't contribute to the semantic meaning of the text (words that are not within the english alphabet) and keep all of the remaining words in the simplest format possible, so we can apply a function that gives weights to each word without generating any bias or outliers. To do that there are many techniques to clean up our corpus, among them we will remove the most common words (stop words) and apply stemming, a technique that reduces a word to it's root.

- ✓ The methods that apply stemming and stop words removal are listed bellow. We will also define a method that removes any words with less than 2 letters or more than 21 letters to clean our corpus even more.

```
Entrée [5]: # removes a list of words (ie. stopwords) from a tokenized list.
def removeWords(listOfTokens, listOfWorks):
    return [token for token in listOfTokens if token not in listOfWorks]

# applies stemming to a list of tokenized words
def applyStemming(listOfTokens, stemmer):
    return [stemmer.stem(token) for token in listOfTokens]

# removes any words composed of less than 2 or more than 21 letters
def twoLetterWord(listOfTokens):
    twoLetterWord = []
    for token in listOfTokens:
        if len(token) <= 2 or len(token) >= 21:
            twoLetterWord.append(token)
    return twoLetterWord
```

Figure 23: Stemming and stop words

- ✓ Now we'll define the main processing function, that uses regular expressions for noise removal and calls the functions defined above.

```
Entrée [6]:
def processCorpus(corpus, language):
    stopwords = nltk.corpus.stopwords.words(language)
    param_stemmer = SnowballStemmer(language)
    other_words = [line.rstrip('\n') for line in open('word.txt')] # Load .txt file Line by Line

    for document in corpus:
        index = corpus.index(document)
        corpus[index] = corpus[index].replace(u'\ufffd', '8') # Replaces the ASCII '☐' symbol with '8'
        corpus[index] = corpus[index].replace(',', '') # Removes commas
        corpus[index] = corpus[index].rstrip('\n') # Removes line breaks
        corpus[index] = corpus[index].casefold() # Makes all letters lowercase

        corpus[index] = re.sub('\W+', ' ', corpus[index]) # removes specials characters and leaves only words
        corpus[index] = re.sub("\S*\d\S*", "", corpus[index]) # removes numbers and words concatenated with numbers IE h4ck3r.
        corpus[index] = re.sub("\S*@*\S*", "", corpus[index]) # removes emails and mentions (words with @)
        corpus[index] = re.sub(r'http\S+', '', corpus[index]) # removes URLs with http
        corpus[index] = re.sub(r'www\S+', '', corpus[index]) # removes URLs with www

        listofTokens = word_tokenize(corpus[index])
        twoLetterWord = twoLetters(listofTokens)

        listofTokens = removeWords(listofTokens, stopwords)
        listofTokens = removeWords(listofTokens, twoLetterWord)

        listofTokens = removeWords(listofTokens, other_words)

        listofTokens = applyStemming(listofTokens, param_stemmer)
        listofTokens = removeWords(listofTokens, other_words)

        corpus[index] = " ".join(listofTokens)
        corpus[index] = unicode(corpus[index])

    return corpus
```

Figure 24: Function process Corpus

- ✓ And after processing our corpus, this is what our data looks like!

```
Entrée [7]: language = 'english'
            corpus = processCorpus(corpus, language)
            corpus[0:1000][0:460]

Out[7]: ['drama romant movi sport movi escap bus accid kill girlfriend high school student channel grief run coach',
'drama romant movi sport movi escap bus accid kill girlfriend high school student channel grief run coach',
'comedi intern movi romant movi girlfriend win miss nigeria pageant young man face unexpected competit join campaign south afr
ica',
'comedi drama intern movi carefre bachelor unexpected left chang young boy embark road trip chang live',
'action adventur fierc mammoth hunter d'leh set imposs journey rescu woman love vicious warlord save peopl villag',
'intern show romant show drama lose memori crown princ encout commoneraEUR(tm) life experi unforgett love husband joseonaEU
R(tm) oldest bacheloret',
'documentari intern movi spanish photograph josa(c) daaz spend day live remot mountain connect natur document beauti surroun
d',
'docuseri scienc natur hundr hardi soul divers background particip play experi explor age sex happi aspect human',
'drama intern movi sport movi man diagnos multipl sclerosi respond train ironman triathlon cranksi father-in-law',
'movi led seventh-grad c.j student warn danger high school decid make middle-school year',
'documentari documentari chronicl elouis cobel long fight u.s. govern gross mismanag mineral-rich nativ american land',
'drama intern movi high school young woman marri man father choic face possibl religion consid union invalid',
'british show intern show realiti stylist hair design makeup artist team give britain biggest fashion disast much-need makeu
nd',
'drama intern movi random receiv handsom polit bribe sweet poor elder woman decid treat shop spree n't smooth',
'drama independ movi sport movi deal person demon death son prizefight attempt return ring challeng rival rematch',
'intern show korean show romant show pregnant teen forc famili leav boyfriend assum ident america year coupl reunite korea'.
```

Figure 25: Processing our corpus

B. Feature Extraction with TF-IDF.

🔗 Statistical Weighting of Words

The pre-processing phase was done so that this stage can yield the best possible results. Here we want to represent how important a word is to a set of documents so that the encoded data is ready to be used by an algorithm. This mapping process of text data into real vectors is known as feature extraction.

TF-IDF, short for Term Frequency-Inverse Document Frequency is a numerical statistic that

intends to reflect the importance of a word in a corpus, which a term with a high weight is considered relevant.

- ✓ Now we will apply the TF-IDF function, short for term frequency inverse document frequency, which is a numerical statistic that's intended to reflect how important a word is to a document in a corpus by giving each word in a document a score that ranges from 0 to 1.

```
Entrée [8]: vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
tf_idf = pd.DataFrame(data = X.toarray(), columns=vectorizer.get_feature_names())

final_df = tf_idf

print("{} rows".format(final_df.shape[0]))
final_df.T.nlargest(5, 0)

1024 rows

Out[8]:
```

	0	1	2	3	4	5	6	7	8	9	...	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023
channel	0.332186	0.332186	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
grief	0.332186	0.332186	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
coach	0.311611	0.311611	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
bus	0.303788	0.303788	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
accid	0.276437	0.276437	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows x 1024 columns

Figure 26: Function TF-IDF

C. Running K-Means and Cluster Analysis

K-Means

- ✓ Function that runs the K-Means algorithm *max_k* times and returns a dictionary of each k result

```
Entrée [13]: def run_KMeans(max_k, data):
max_k += 1
kmeans_results = dict()
for k in range(2, max_k):
    kmeans = cluster.KMeans(n_clusters = k
                           , init = 'k-means++'
                           , n_init = 10
                           , tol = 0.0001
                           , n_jobs = -1
                           , random_state = 1
                           , algorithm = 'full')

    kmeans_results.update( {k : kmeans.fit(data)} )

return kmeans_results
```

Figure 27: function runs the K-Means

Silhouette Score

- ✓ The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

```

Entrée [14]: def printAvg(avg_dict):
    for avg in sorted(avg_dict.keys(), reverse=True):
        print("Avg: {} \tK: {}".format(avg.round(4), avg_dict[avg]))

def plotSilhouette(df, n_clusters, kmeans_labels, silhouette_avg):
    fig, ax1 = plt.subplots(1)
    fig.set_size_inches(8, 6)
    ax1.set_xlim([-0.2, 1])
    ax1.set_ylim([0, len(df) + (n_clusters + 1) * 10])

    ax1.axvline(x=silhouette_avg, color="red", linestyle="--") # The vertical line for average silhouette score of all the values
    ax1.set_yticks([]) # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.2, 0, 0.2, 0.4, 0.6, 0.8, 1])
    plt.title(("Silhouette analysis for K = %d" % n_clusters), fontsize=10, fontweight='bold')

    y_lower = 10
    sample_silhouette_values = silhouette_samples(df, kmeans_labels) # Compute the silhouette scores for each sample
    for i in range(n_clusters):
        ith_cluster_silhouette_values = sample_silhouette_values[kmeans_labels == i]
        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_between(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values, facecolor=color, edgecolor=color, alpha=0.5)

        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i)) # Label the silhouette plots with their cluster numbers at the middle
        y_lower = y_upper + 10 # Compute the new y_lower for next plot. 10 for the 0 samples
    plt.show()

def silhouette(kmeans_dict, df, plot=False):
    df = df.to_numpy()
    avg_dict = dict()
    for n_clusters, kmeans in kmeans_dict.items():
        kmeans_labels = kmeans.predict(df)
        silhouette_avg = silhouette_score(df, kmeans_labels) # Average Score for all Samples
        avg_dict.update( {silhouette_avg : n_clusters} )

    if(plot): plotSilhouette(df, n_clusters, kmeans_labels, silhouette_avg)

```

Figure 28: Silhouette Score

```

In [11]: # Running Kmeans
k = 8
kmeans_results = run_KMeans(k, final_df)

# Plotting Silhouette Analysis
#silhouette(kmeans_results, final_df, plot=True)

```

Figure 29: Running kmeans

Cluster Analysis

- ✓ Now we can choose the best number of K and take a deeper look at each cluster. Looking at the plots above, we have some clues that when K = 5 is when the clusters are best defined. So first we will use a simple histogram to look at the most dominant words in each cluster:


```

Entrée [22]: def get_top_features_cluster(tf_idf_array, prediction, n_feats):
              labels = np.unique(prediction)
              dfs = []
              for label in labels:
                  id_temp = np.where(prediction==label) # indices for each cluster
                  x_means = np.mean(tf_idf_array[id_temp], axis = 0) # returns average score across cluster
                  sorted_means = np.argsort(x_means)[::-1][:n_feats] # indices with top 20 scores
                  features = vectorizer.get_feature_names()
                  best_features = [(features[i], x_means[i]) for i in sorted_means]
                  df = pd.DataFrame(best_features, columns = ['features', 'score'])
                  dfs.append(df)
              return dfs

def plotWords(dfs, n_feats):
    plt.figure(figsize=(8, 4))
    for i in range(0, len(dfs)):
        plt.title(("Most Common Words in Cluster {}".format(i)), fontsize=10, fontweight='bold')
        sns.barplot(x = 'score', y = 'features', orient = 'h', data = dfs[i][:n_feats])
        plt.show()

```

Figure 30: the most dominant word

```

Entrée [23]: best_result = 5
              kmeans = kmeans_results.get(best_result)

              final_df_array = final_df.to_numpy()
              prediction = kmeans.predict(final_df)
              n_feats = 20
              dfs = get_top_features_cluster(final_df_array, prediction, n_feats)
              plotWords(dfs, 13)

```

Figure 31: the best result for Number of clustering

- ✓ Here is the result of the k_means algorithm ,to select the best number of clusters I went through all the different clusters in each grouping by looking at graphs with the most dominant words in each group. I came to the conclusion that the groups made much more sense when they were divided into 5 different clusters.

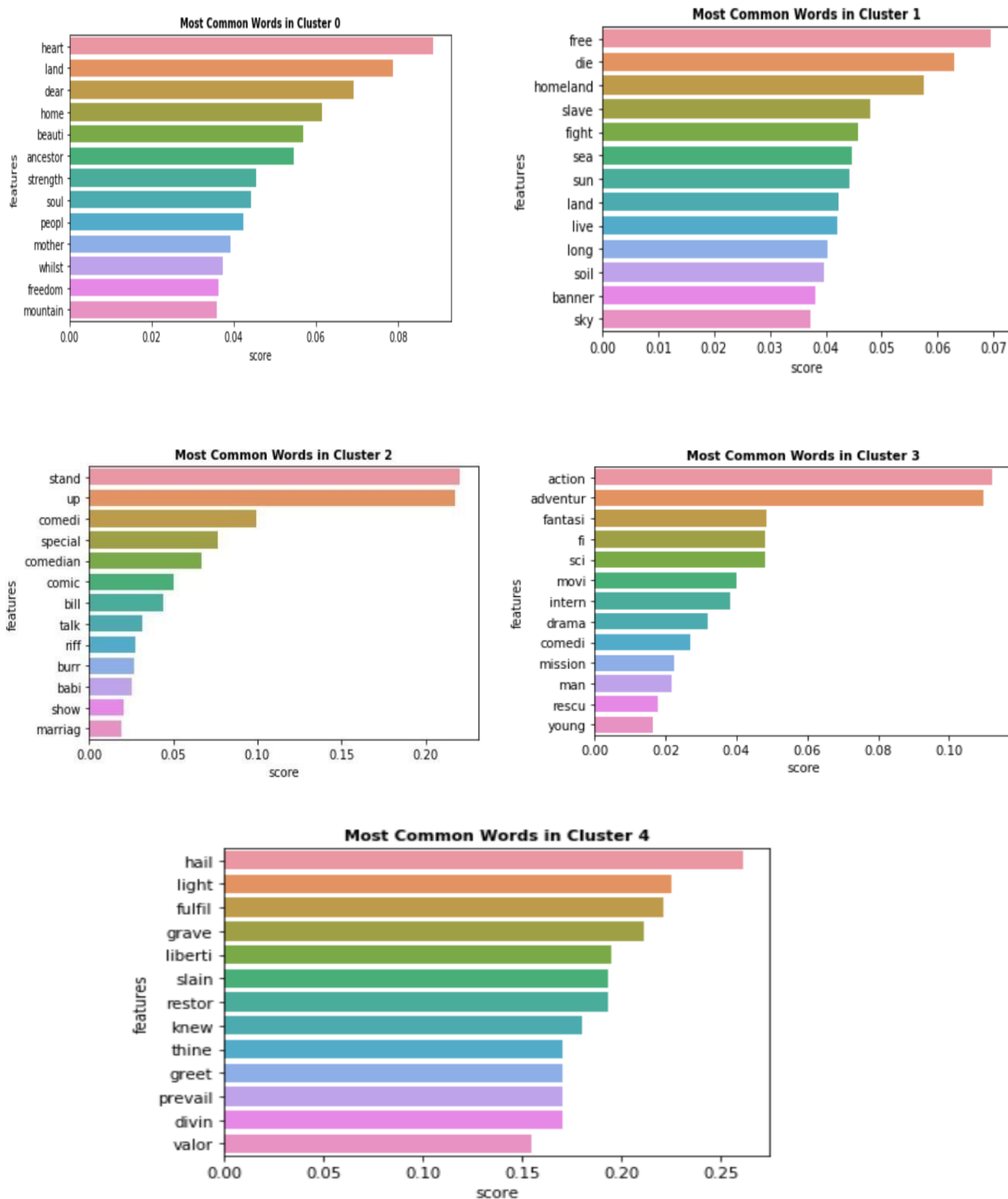


Figure 32:Most dominant words by cluster

- ✓ Looking at the clusters it's clear that the words in each one of them have a theme. we will specify each cluster and its characteristics :

- **Cluster 0**: this cluster contains the words related to documentary movie which process different subjects like: Life, sport ,world, explor ,intern, activits.
- **Cluster 1**: this cluster contains the words related to show TV which deals different topics like crime, intern, seri.
- **Cluster 2**: this cluster contains the words related to comedy like :Stand up, comedi, comedian, special, talk comic
- **Cluster 3**: this cluster contains the words related to action movie like :adventur, fantasi, action.
- **Cluster 4**: this cluster contains the words related to drama movie like: romant, drama, love, music.

D. Data visualization

Data visualization is the graphical representation of information and data. By using **visual elements like charts, graphs, and maps**, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

- ✓ For this reason we are only focused on visualizing to help the **Netflix** company to analyze and make decisions based on its data.
- ✓ We realized a comparison between Movies and TV Shows :



Figure 34: Movies and TV Shows Content Comparison

- ✓ From this circle (figure 30), we observe that **movies** are more demanded than **TV show**

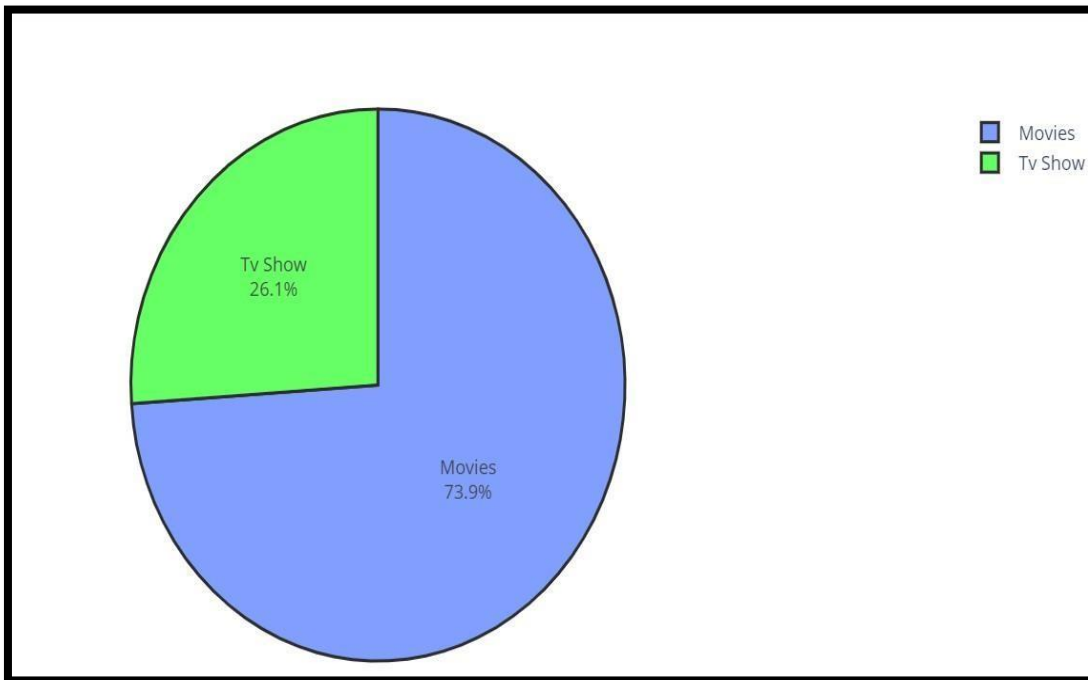


Figure 35: Cercle represent comparison between Tv show and movies

- ✓ We content the countries which produces the Tv show and movies.

```
Entrée [29]: netflix_df_counts = data.country.value_counts()
netflix_df_counts

Out[29]: United States      331
         India             135
         United Kingdom     43
         Japan              35
         Canada             29
         ...
         Ghana              1
         France, Canada, Belgium 1
         Spain, Italy       1
         United Kingdom, Spain, United States 1
         Netherlands, Denmark, South Africa 1
         Name: country, Length: 161, dtype: int64
```

Figure 36: Content the countries

- ✓ We realized the chart to represent the country more appear in data

```
Entrée [30]: trace = go.Bar(x=netflix_df_counts.index[:15], y=netflix_df_counts, marker=dict(
                opacity=0.8,
                color=np.arange(15)
            ))
fig = go.Figure(data=[trace])

fig.update_layout(title="Top 15 Countries by Content")
fig.update_xaxes(title="Country")
fig.update_yaxes(title="Count")

iplot(fig)
```

Figure 37: Chart countries

- ✓ This chart represent top 15 countries by content, we observe we observe that United states more appear in data so it produces more the Tv show and movies.

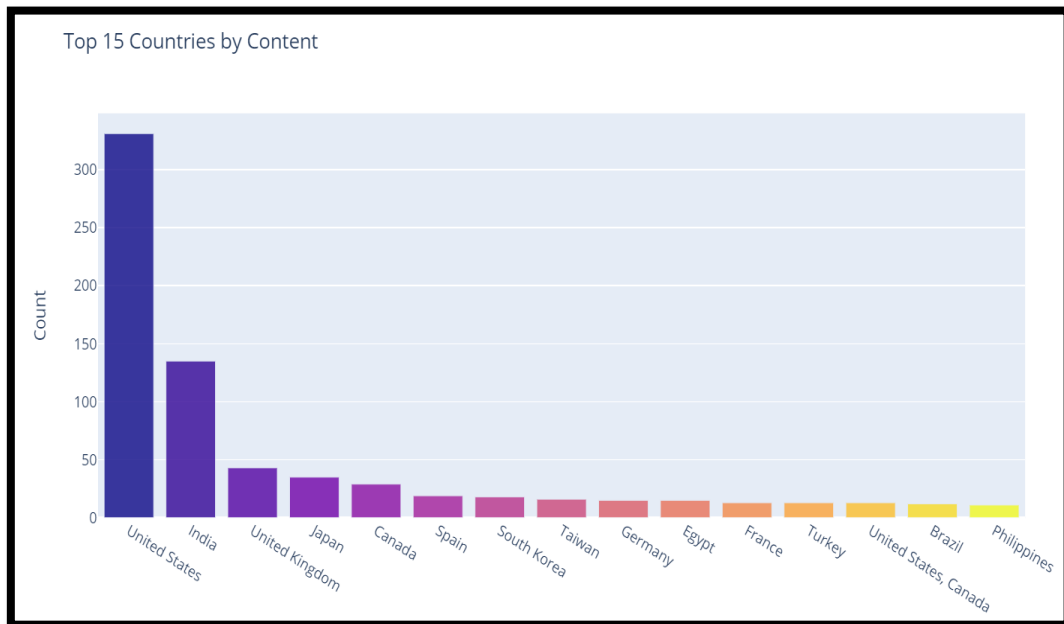


Figure 38: chart represent the top 15 countries

- ✓ We content date added on Netflix

```
Entrée [32]: year_wise_content = data.date_added.value_counts().index[:20]
year_wise_content

Out[32]: Index(['December 31, 2019', 'October 1, 2017', 'April 1, 2018',
               'November 1, 2019', 'March 1, 2018', 'November 1, 2018',
               'January 1, 2020', 'July 1, 2020', 'July 1, 2017', 'July 12, 2019',
               'November 20, 2019', 'December 1, 2018', 'August 1, 2017',
               'September 1, 2020', 'December 1, 2020', 'August 7, 2017',
               'October 1, 2019', 'December 1, 2019', 'February 15, 2019',
               'January 1, 2019'],
              dtype='object')
```

Figure 39:year_added content

- ✓ We realized the chart to represent year wise added on netflix

```
Entrée [33]: plt.figure(figsize=(12,10))
sns.set_style("dark")
sns.countplot(data=data, y="date_added", order=year_wise_content, palette="viridis")
```

Figure 40: Plot date_added on netflix

✓ We realized the chart represent content date added on Netflix

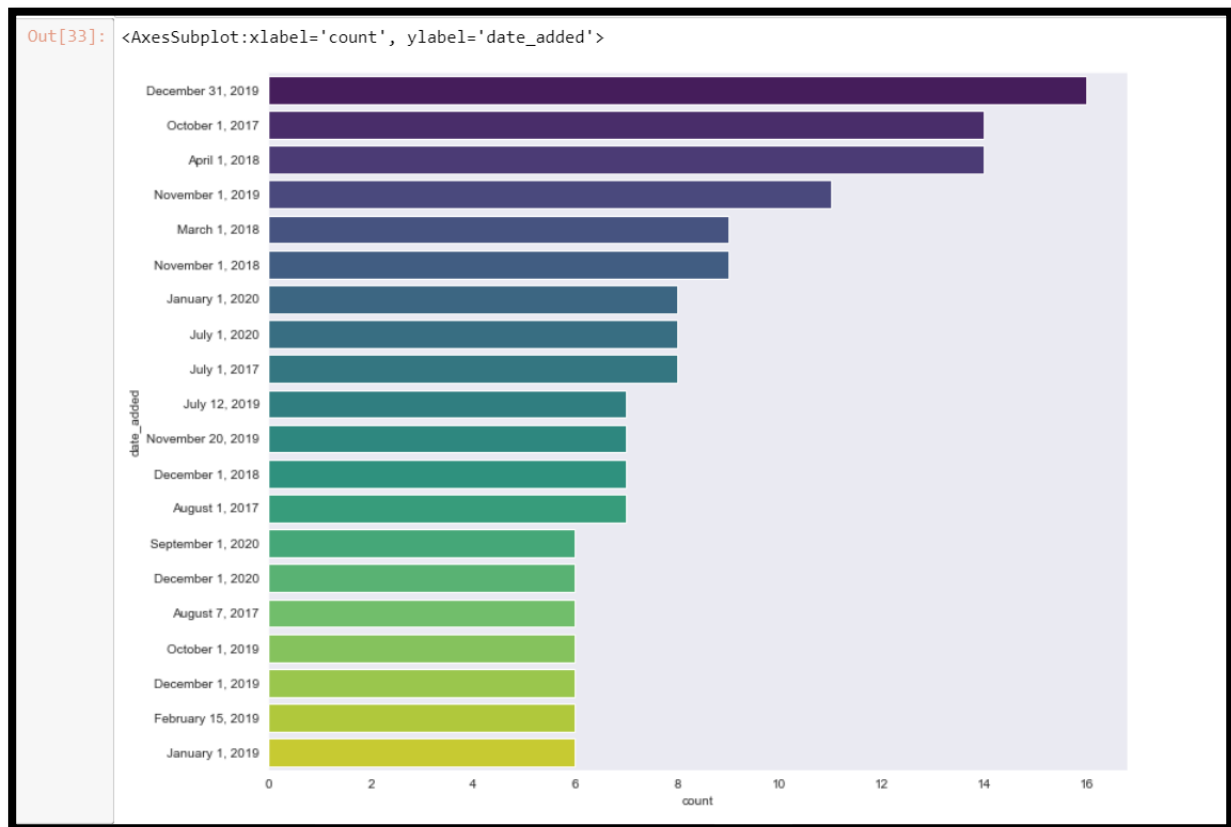


Figure 41: chart represent content date added on Netflix

Exercise: Try to practice the same approach on Spark-MILib.

Lab-Conclusion

Text clustering is a process that involves *Natural Language Processing* (NLP) and the use of a clustering algorithm. This method of finding groups in unstructured texts can be applied in many different segments, such as feedback analysis, research segmentation, etc.

In K-Means there are many variables that influence the algorithm's result, such as different distance metrics (euclidean, manhattan, cosine similarity, etc), the centroids initial positions, and grouping analysis can be very subjective.

It's also important to note that K-Means is not the only way of clustering data, there are other different methods, such as hierarchical clustering algorithms.