10/18 DATA ANALYTICS - IRONHACK BOOTCAMP

Sistema de Recomendación: Codewars

Alberto García Cobo

CONTENIDO

Red social para programadores

Honor y Score

+20 lenguajes de programación TOP

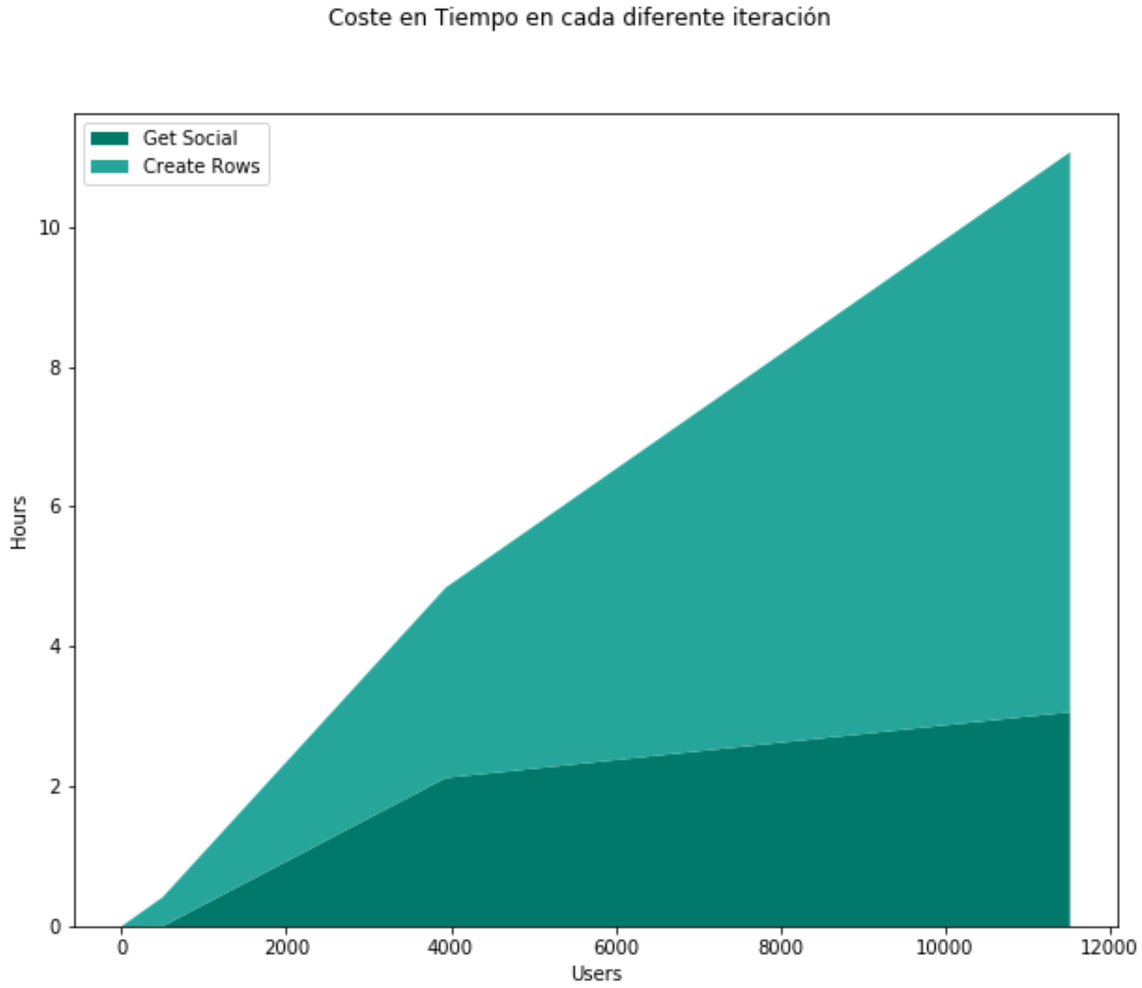
Descripción de Codewars

Usuarios de Codewars

Web Scraping + API requests

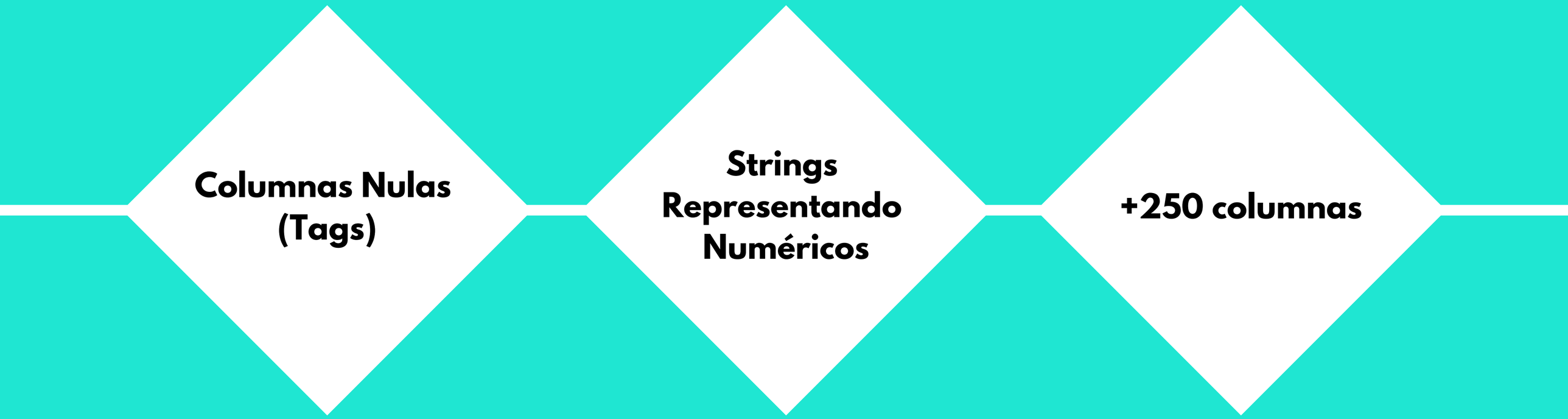
11500 Registros 3 Iteraciones

Descripción del Dataset



Descripción del Dataset

0.0+0.4 horas	500
2.1+2.7 horas	3939
3.0+8.0 horas	11500



Data cleaning and wrangling

"JAVASCRIPT (5 KYU)"
"3 (1 REPLIES)"
"SEP 2018"
"3,101"

Data cleaning and wrangling

Combinar Datos de la API y Web Scraping

Treemap: javascript + json ad hoc

Sistema de Recomendación Vectorial

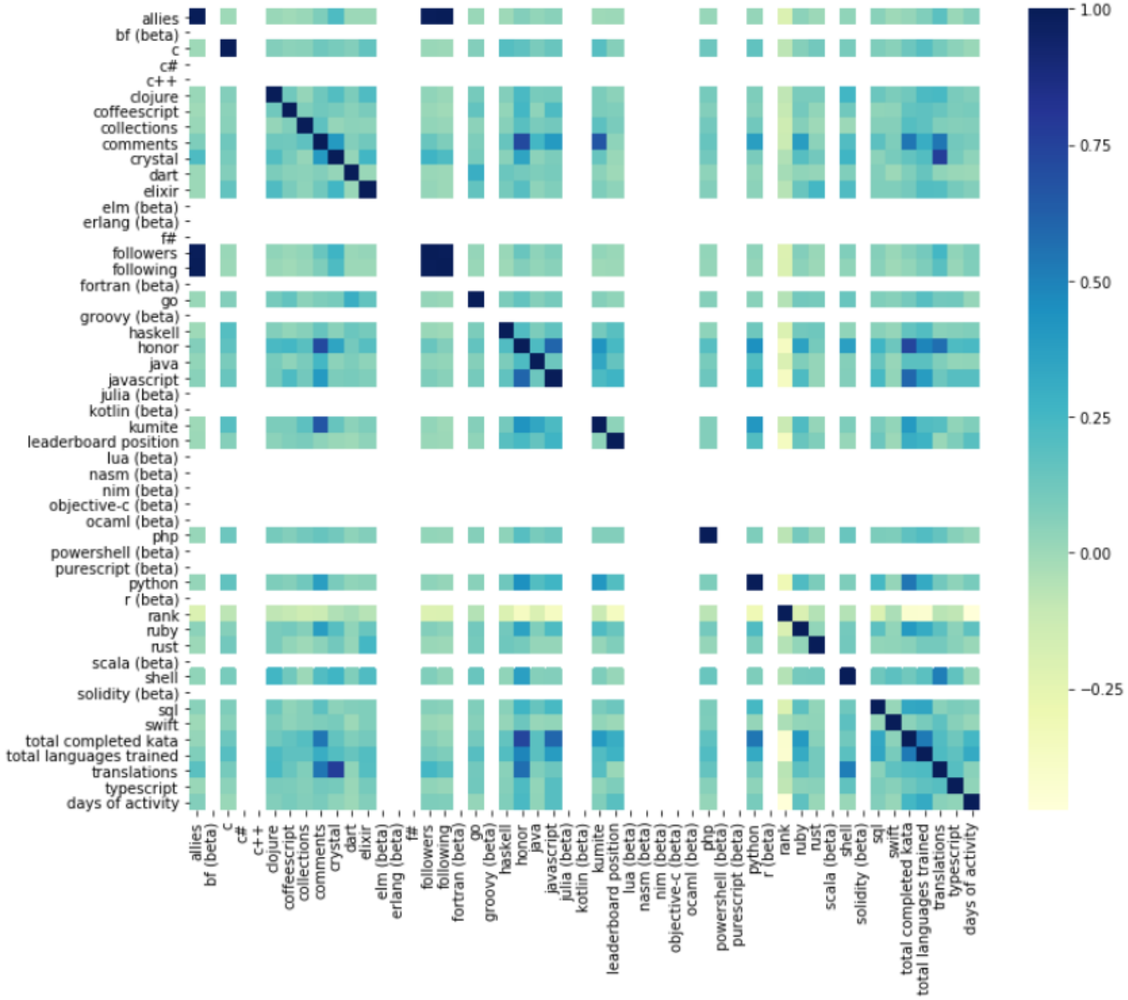
Mayores retos

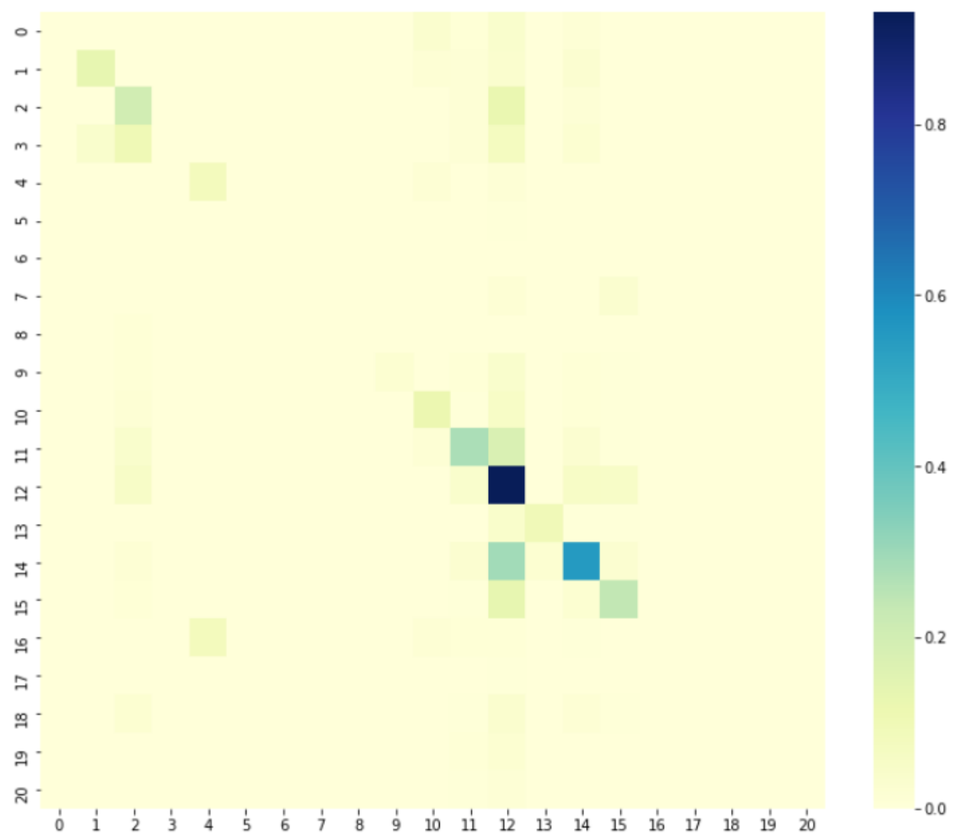
Correlación heatmap

Supervised Learning: score = 0.51

Unsupervised Learning

Resultados ML





Correlation y confusion heatmap

Tiempos de ETL

Web Scraping de una Web Dinámica

Creación de Grafos

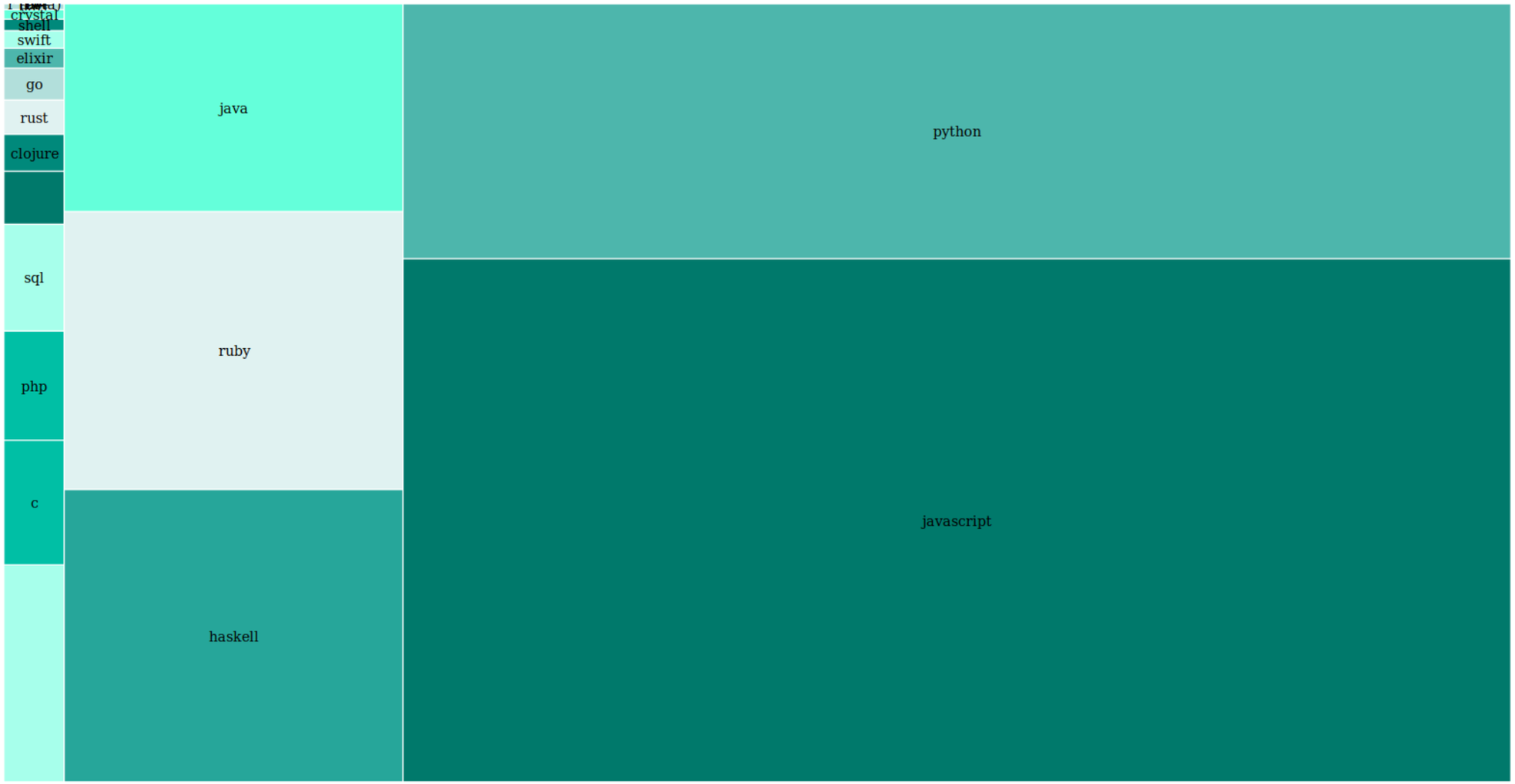
Problemas encontrados

Javascritp, la gran ganadora

Sistema de recomendación por distancia euclidea

Grafos de relaciones

Conclusiones


crystal
shell
swift
elixir
go
rust
clojure
sql
php
c
java
ruby
haskell
python
javascript

d3 Treemap

Buscamos expertos en: python, java y javascript

Blind4Basics

https://github.com/Blind4Basics

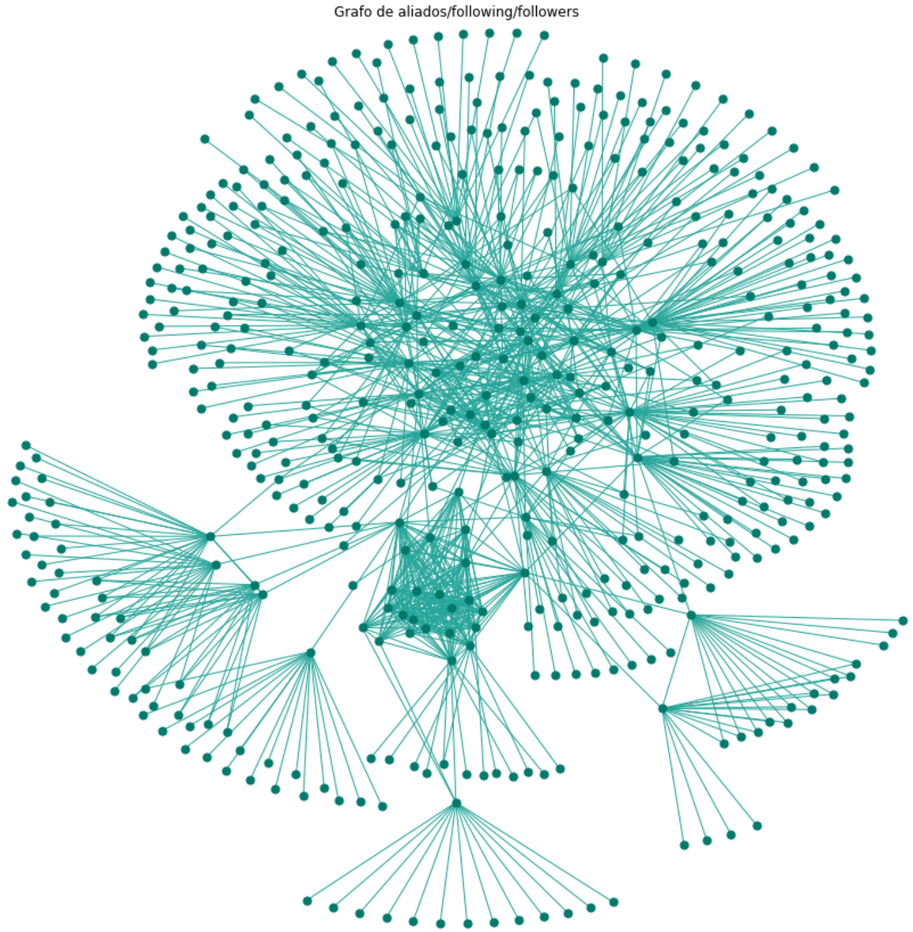
Bubbler

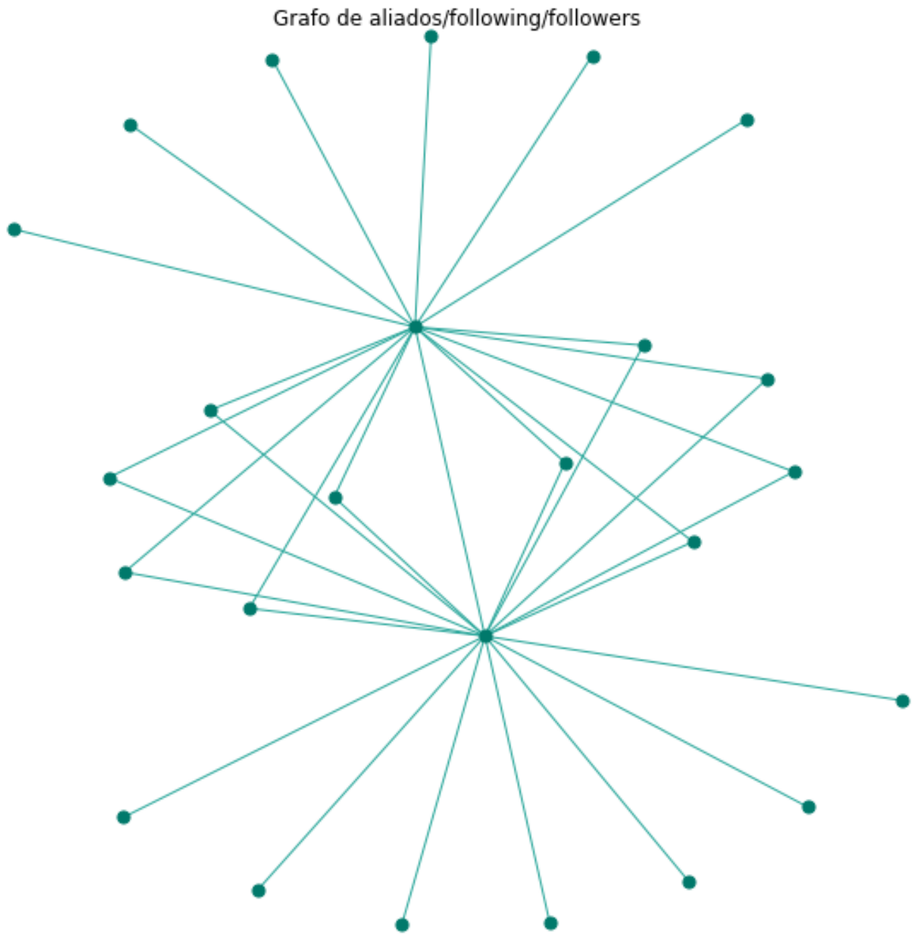
https://github.com/Bubbler-4

Lyapunov

https://github.com/Lyapunov

Sistema de recomendación





Grafos de networkx

Diferentes aproximaciones

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import networkx as nx
import requests
import json
from bs4 import BeautifulSoup
```

Código

API Codewars

```
def get_all_stats(user):
    url = 'https://www.codewars.com/users/{}'.format(user)
    html = requests.get(url).content
    soup = BeautifulSoup(html, "lxml")

    x = {}
    x['username'] = user
    for s in soup.select('.stat-box div'):
        if s.text.split(':')[0] != 'Profiles':
            x[s.text.split(':')[0].lower()] = s.text.split(':')[1]
        else:
            try:
                for e in s.find_all('a', href=True):
                    if 'github' in e['href']:
                        x['github'] = e['href']
                    if 'linkedin' in e['href']:
                        x['linkedin'] = e['href']
            except:
                x['github'] = ''
                x['linkedin'] = ''
    return x
```

Web Scraping

```
def get_score_language(user_json, language):
    try:
        score = user_json['ranks']['languages'][language]['score']
    except:
        score = 0
    return score


def get_scores(user_json):
    res = {}
    for lang in get_languages():
        res[lang] = get_score_language(user_json, lang)
    return res
```

```
def get_row(user):
    '''
    Crea un diccionario con los datos recopilados que se puede
    añadir como fila en nuestro dataframe de pandas
    '''
    user_json = get_user_api(user)
    res = get_all_stats(user)
    res.update(get_scores(user_json))
    return res
```

Networkx Graph

```
plt.figure(figsize=(8,8))

G = nx.from_pandas_edgelist(df=df_social2, source='from', target='to', create_using=nx.Graph())
nx.draw_kamada_kawai(G, node_size=50, node_color='#00796B', edge_color='#26A69A', with_labels=Fals
e)

plt.suptitle('Grafo de aliados/following/followers')
plt.savefig('../output/Grafo-set-users.png')
plt.show()
```

Treemap d3

```
def create_d3(data):
    df = data
    idiomas = []
    languages = [lang for lang in get_languages() if lang in df.columns]
    for col in languages:
        idiomas.append({'name': col, "size": int(df[col].sum())})
    res = {"name": "Lenguajes","children": [{"name": "Lenguajes","children": [{"name": "Lenguajes"
,"children": idiomas}]}]}

    return json.dumps(res, ensure_ascii=False)

def save_d3_languages_json(lang_json, filename='../representacion_d3/lenguajes.json'):
    with open(filename, 'w') as f:
        f.write(lang_json)
```

Sistema de Recomendación

```python
def get_ask_vector(skills, languages):
    res = sorted([(l, 1) if l in skills else (l, 0) for l in languages])
    return [v for k, v in res]
# get_ask_vector(['python', 'c'], get_languages())

def distacia_euclidea(v1, v2):
    return sum([(x1-x2)**2 for x1, x2 in zip(v1, v2)])**0.5
# distacia_euclidea([3, 4], [0, 0])

def get_languages_vector(row, languages):
    # return [(k, v) for k, v in row.items() if k in languages]
    res = sorted([(k, v) for k, v in row.items() if k in languages])
    return [v for k, v in res]
# get_languages_vector(df_norm.loc[4, :], get_languages())
```