SISTEMA DE RECOMENDACIONES

Institución: Universidad Franz Tamayo

Título del proyecto: SISTEMA DE RECOMENDACIONES

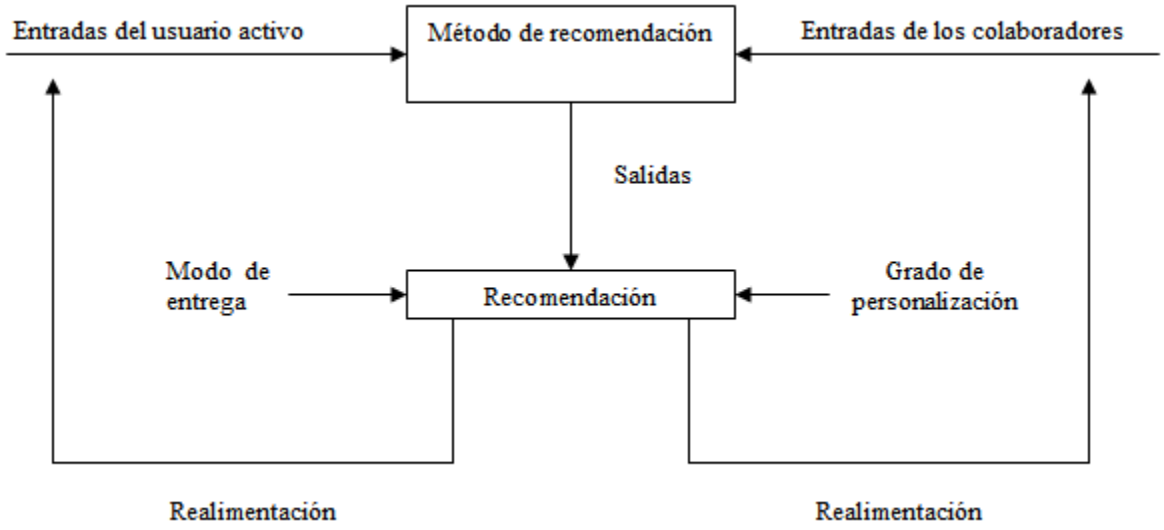
Nombre del estudiante: Mel Dusan Mejia Garcia

Nombres docentes: Miguel Paco

Contenido

ANTECEDENTES

Los sistemas de recomendaciones son herramientas que generan recomendaciones sobre un determinado objeto de estudio, a partir de las preferencias y opiniones dadas por los usuarios. El uso de estos sistemas se está poniendo cada vez más de moda en Internet debido a que son muy útiles para evaluar y filtrar la gran cantidad de información disponible en la Web con objeto de asistir a los usuarios en sus procesos de búsqueda y recuperación de información. En este trabajo realizaremos una revisión de las características y aspectos fundamentales relacionados con el diseño, implementación y estructura de los sistemas de recomendaciones.



OBJETIVO

Desarrollar una aplicación informática que ayude predecir los productos que querrá adquirir un usuario en particular.

OBJETIVOS ESPECIFICOS

1. Analizar la importancia de un sistema de recomendaciones.
2. Desarrollar la aplicación
3. Probar la aplicación (localmente)
4. Desplegar para su uso

METODOLOGIA

Método Deductivo En el método deductivo se utiliza la lógica y toda la información general para formular y dar una solución posible a un problema dado. Luego se comprueba la solución en varias situaciones normales. Por lo tanto, en el método deductivo, el razonamiento va de lo general a lo específico.

Por todo esto el método que utilizaremos será el método deductivo ya que es el que utiliza la información general para dar una solución a un problema dado.

Técnicas de investigación

Para el desarrollo de este sistema se utilizó el siguiente mecanismo para hacer la recolección de datos. Se recopilo información relacionada sobre los tipos de tecnologías para llevar un sistema de recomendaciones adecuado y también la recopilación de datos.

MARCO TEORICO

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Dash

Dash es un framework de Python que está pensado para construir aplicaciones web, pero se utiliza también mucho para crear visualizaciones, porque permite customizar mucho tu dashboard o cuadro de mando.

Dash está basado principalmente en **Flask, Plotly y ReactJS**, y estas tres herramientas en las que se basa.

DESARROLLO

Cargamos las librerías que utilizaremos

```python
import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import sklearn
```

Cargamos y previsualizamoás los 3 archivos de datos csv que utilizaremos:

```python
df_users = pd.read_csv("users.csv")
df_repos = pd.read_csv("repos.csv")
df_ratings = pd.read_csv("ratings.csv")
print(df_users.head())
print(df_repos.head())
print(df_ratings.head())
```

	userId	username	name
0	1	iris9112	Isabel Ruiz Buriticá
1	2	dianaclarke	Diana
2	3	nateprewitt	Nate Prewitt
3	4	oldani	Ordanis Sanchez
4	5	waflessnet	waflessnet

	repoId	title	categories	stars
0	1	airbnb / javascript	completar	NaN
1	2	kamranahmedse / developer-roadmap	Roadmap to becoming a web developer in 2019	85800.0
2	3	microsoft / vscode	Visual Studio Code	80855.0
3	4	torvalds / linux	Linux kernel source tree	78761.0
4	5	ytdl-org / youtube-dl	Command-line program to download videos from Y...	53909.0

	userId	repoId	rating
0	1	1	2
1	1	2	3
2	1	3	4
3	1	4	5
4	1	5	3

Vemos que tenemos un archivo con la información de los usuarios y sus identificadores, un archivo con la información de los repositorios y finalmente el archivo «ratings» que contiene la valoración por usuario de los repositorios. Como no tenemos REALMENTE una valoración del 1 al 5 -como podríamos tener por ejemplo al valorar películas-, la columna rating **es el número de usuarios que tienen ese mismo repositorio** dentro de nuestra base de datos. Sigamos explorando para comprende un poco mejor:

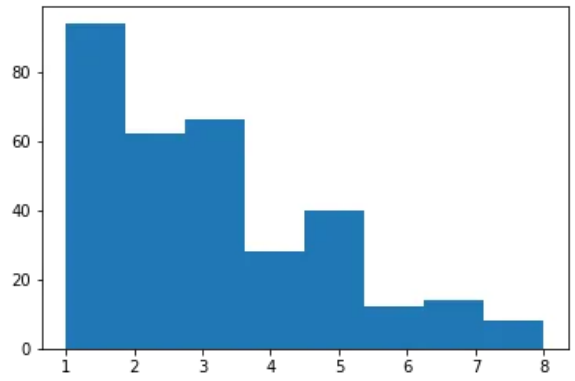
```
n_users = df_ratings.userId.unique().shape[0]
n_items = df_ratings.repoId.unique().shape[0]
print (str(n_users) + ' users')
print (str(n_items) + ' items')
```

```
30 users
167 items
```

Vemos que es un dataset reducido, pequeño. Tenemos 30 usuarios y 167 repositorios valorados.

```
plt.hist(df_ratings.rating,bins=8)
```

```
1 plt.hist(df_ratings.rating,bins=8)
```

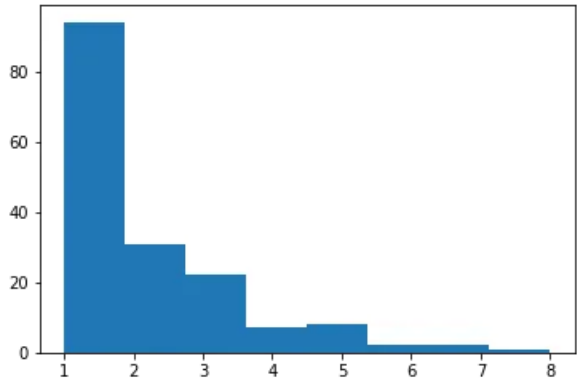


Tenemos más de 80 valoraciones con una puntuación de 1 y unas 40 con puntuación en 5. Veamos las cantidades exactas:

```
1 df_ratings.groupby(["rating"])["userId"].count()
```

```
rating
1 94
2 62
3 66
4 28
5 40
6 12
7 14
8 8
Name: userId, dtype: int64
```

```
1 plt.hist(df_ratings.groupby(["repoId"])["repoId"].count(),bins=8)
```



Aquí vemos la cantidad de repositorios y cuantos usuarios «los tienen». La mayoría de repos los tiene 1 sólo usuario, y no los demás. Hay unos 30 que los tienen 2 usuarios y unos 20 que coinciden 3 usuarios. La suma total debe dar 167.

Creamos la matriz usuarios/ratings

Ahora crearemos la matriz en la que cruzamos todos los usuarios con todos los repositorios.

```python
df_matrix = pd.pivot_table(df_ratings, values='rating', index='userId', columns='repoId').fillna(0)
df_matrix
```

repoId	1	2	3	4	5	6	7	8	9	10	...
userId											
1	2.0	3.0	4.0	5.0	3.0	1.0	5.0	1.0	0.0	4.0	...
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
3	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	...
4	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	...
5	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
10	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...

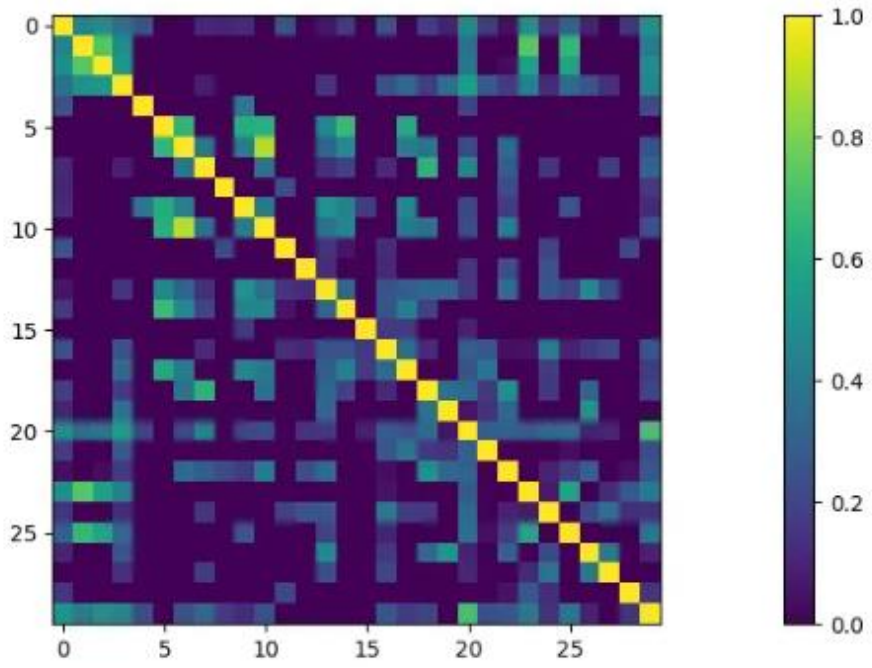
Matriz de Similitud: Distancias por Coseno

Ahora calculamos en una nueva matriz la similitud entre usuarios.

```python
sim_matrix = 1 - sklearn.metrics.pairwise.cosine_distances(ratings)
print(sim_matrix.shape)
```

```
(30, 30)
```

```python
plt.imshow(sim_matrix);
plt.colorbar()
plt.show()
```

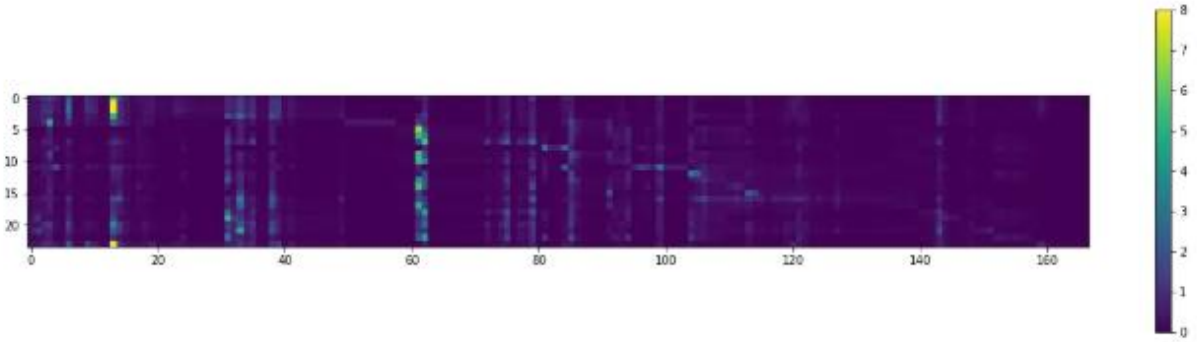


Cuanto más cercano a 1, mayor similitud entre esos usuarios.

Predicciones -ó llamémosle «Sugeridos para ti»-

```
#separar las filas y columnas de train y test
sim_matrix_train = sim_matrix[0:24,0:24]
sim_matrix_test = sim_matrix[24:30,24:30]

users_predictions = sim_matrix_train.dot(ratings_train) / np.array([np.abs(sim_matrix_train).sum(axis=
```

```
plt.rcParams['figure.figsize'] = (20.0, 5.0)
plt.imshow(users_predictions);
plt.colorbar()
plt.show()
```



Vemos pocas recomendaciones que logren puntuar alto. La mayoría estará entre 1 y 2 puntos. Esto tiene que ver con nuestro dataset pequeño.

Hito3

Método 3: Collaborative Filtering Basado en Items

In [41]:
```
#basado en items
n_repos = ratings_train.shape[1]
n_repos
```

Out[41]:
```
167
```

In [42]:
```
neighbors = NearestNeighbors(n_repos, 'cosine')
neighbors.fit(ratings_train.T)
```

Out[42]:
```
NearestNeighbors(algorithm='auto', leaf_size=30, metric='minkowski',
                 metric_params=None, n_jobs=None, n_neighbors=167, p=2,
                 radius='cosine')
```

In [43]:
```
top_k_distances, top_k_items = neighbors.kneighbors(ratings_train.T, return_distance=True)
top_k_distances.shape
```

Out[43]:
```
(167, 167)
```

In [44]:
```
top_k_items
```

Out[44]:
```
array([[  0,  15,  18, ...,  62,  13,  61],
       [  1, 144, 151, ...,  62,  13,  61],
       [  2,   0,  18, ...,  62,  13,  61],
       ...,
       [164, 163, 162, ...,  62,  61,  13],
       [164, 163, 162, ...,  62,  61,  13],
       [164, 163, 162, ...,  62,  61,  13]])
```

In [45]:
```
data = df_repos[df_repos['title'] == 'jbagnato / machine-learning']
repo_ver = data.iloc[0]['repoId'] -1
#print(repo_ver)
```

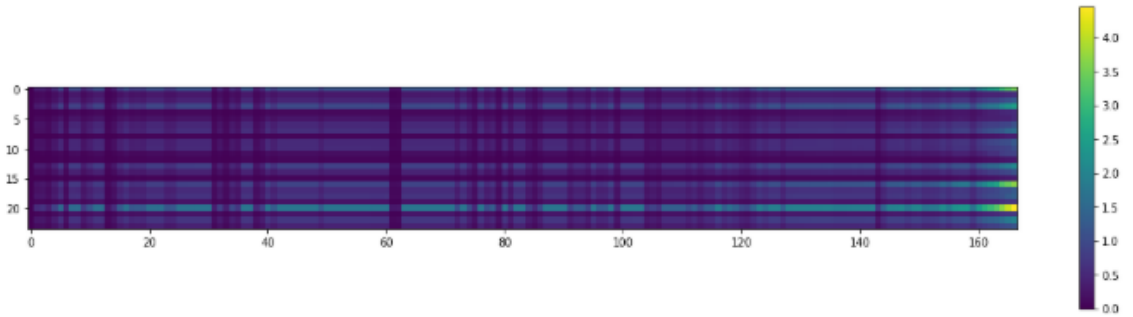
In [46]:
```
df_repos[df_repos['repoId'].isin([repo_ver+1])]
```

Out[46]:

	repoId	title	categories	stars
61	62	jbagnato / machine-learning	Código Python, Jupyter Notebooks, archivos csv,...	48.0

In [47]:
```
item_preds = ratings_train.dot(top_k_distances) / np.array([np.abs(top_k_distances).sum(axis=1)])
```

In [48]:
```
plt.imshow(item_preds);
plt.colorbar()
plt.show()
```



Método 4: recomendacion por Correlacion

In [52]:
```
average_rating = pd.DataFrame(df_ratings.groupby('repoId')['rating'].mean())
average_rating['ratingCount'] = pd.DataFrame(df_ratings.groupby('repoId')['rating'].count())
average_rating.sort_values('ratingCount', ascending=False).head()

mi_repo_ratings = df_matrix[62]
similar_to_mine = df_matrix.corrwith(mi_repo_ratings)
corr_mine = pd.DataFrame(similar_to_mine, columns=['pearsonR'])
corr_mine.dropna(inplace=True)
corr_summary = corr_mine.join(average_rating['rating'])
corr_summary[corr_summary['rating']>=1].sort_values('pearsonR', ascending=False).head(10)
```

Out[52]:

repoId	pearsonR	rating
62	1.000000	7
87	0.341515	3
92	0.341515	3
95	0.341515	3
139	0.336601	1
138	0.336601	1
137	0.336601	1
140	0.336601	1
88	0.336601	1
67	0.336601	1

In [53]:
```
# Veamos uno de los recomendados
df_repos[df_repos['repoId'] == 92]
```

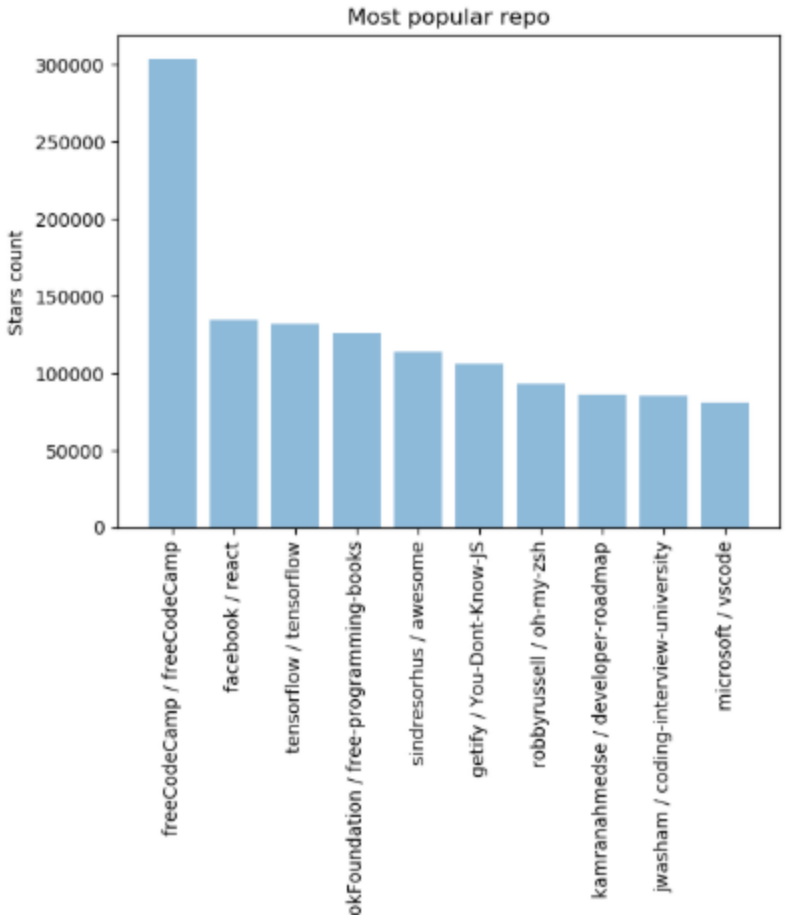
Out[53]:

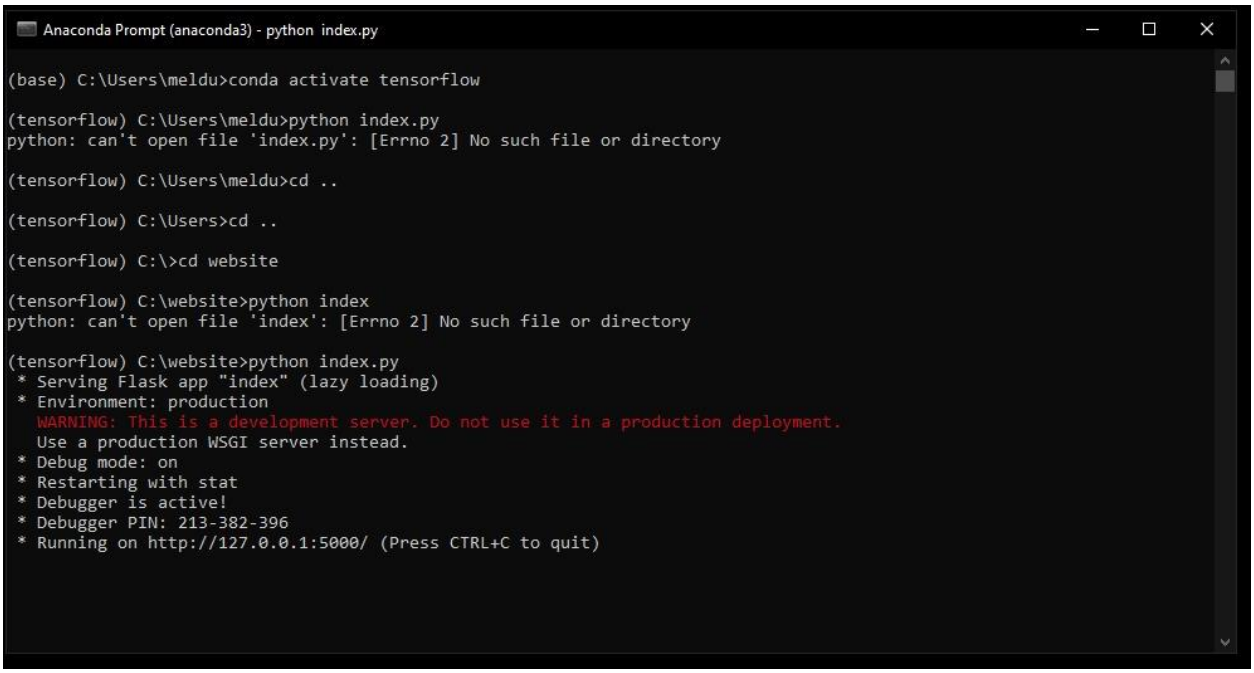
	repoId	title	categories	stars
91	92	joanby / python-ml-course	Curso de Introducción a Machine Learning con P...	156.0

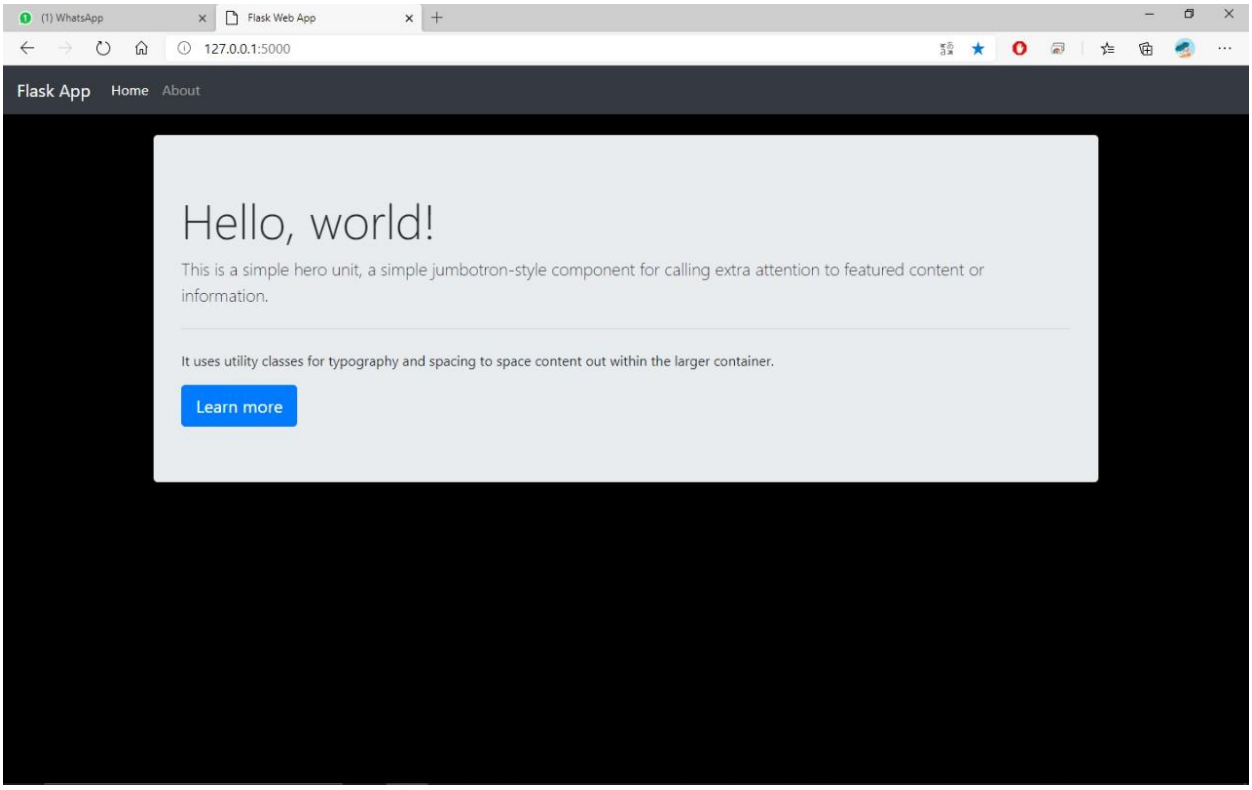
Método 5: Repo mas popular -sin collaborative filtering-

Esta es la manera básica y sin uso de machine learning de ofrecer recomendaciones.
estas no serán personalizadas, serán iguales para cualquier usuario

In [54]:
```
popular_repo = df_repos[['title','stars']].groupby('stars').sum().reset_index()
popular_repo_top_20 = popular_repo.sort_values('stars', ascending=False).head(n=10)

plt.rcdefaults()

objects = (list(popular_repo_top_20['title']))
y_pos = np.arange(len(objects))
performance = list(popular_repo_top_20['stars'])

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects, rotation='vertical')
plt.ylabel('Stars count')
plt.title('Most popular repo')

plt.show()
```







HITO 4

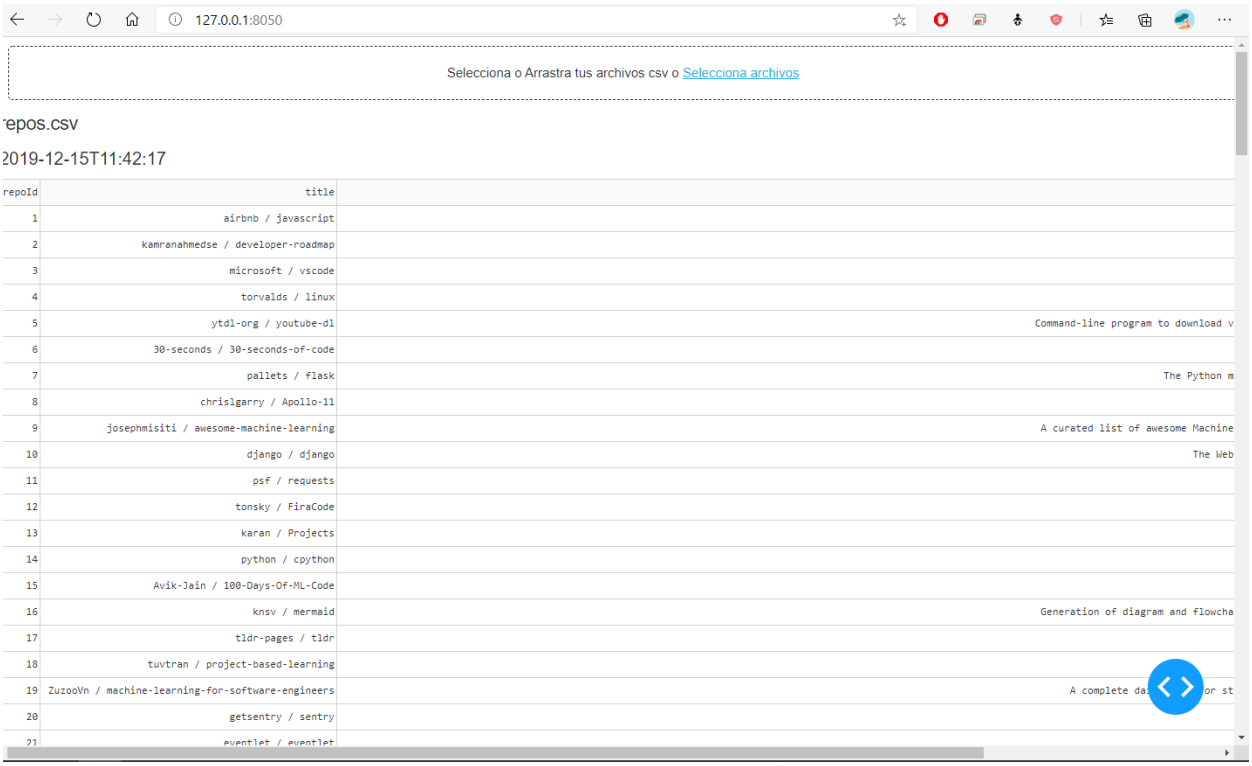
```
import base64

import datetime

import io
```

```
import dash
from dash.dependencies import Input, Output, State
import dash_core_components as dcc
import dash_html_components as html
import dash_table

import pandas as pd



external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)

app.layout = html.Div([
    dcc.Upload(
        id='upload-data',
        children=html.Div([
            'Selecciona o Arrastra tus archivos csv o ',
            html.A('Selecciona archivos')
        ]),
        style={
            'width': '100%',
            'height': '60px',
            'lineHeight': '60px',
            'borderWidth': '1px',
            'borderStyle': 'dashed',
            'borderRadius': '5px',
            'textAlign': 'center',
```

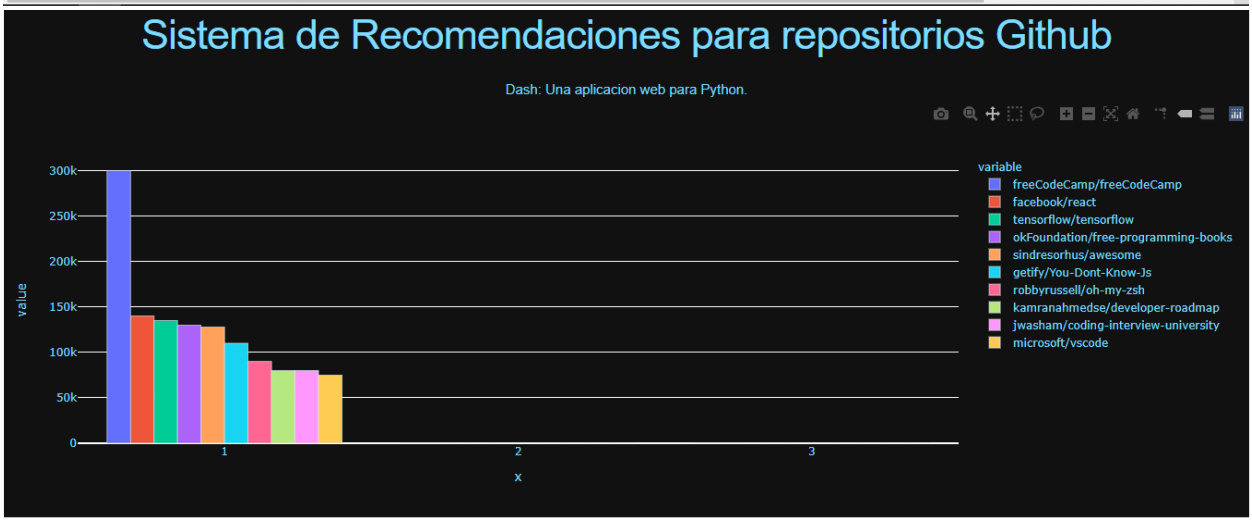
```
            'margin': '10px'
        },
        # Allow multiple files to be uploaded
        multiple=True
    ),
    html.Div(id='output-data-upload'),
])


def parse_contents(contents, filename, date):
    content_type, content_string = contents.split(',')

    decoded = base64.b64decode(content_string)
    try:
        if 'csv' in filename:
            # Assume that the user uploaded a CSV file
            df = pd.read_csv(
                io.StringIO(decoded.decode('utf-8')))
        elif 'xls' in filename:
            # Assume that the user uploaded an excel file
            df = pd.read_excel(io.BytesIO(decoded))
    except Exception as e:
        print(e)
        return html.Div([
            'There was an error processing this file.'
        ])

    return html.Div([
        html.H5(filename),
```

```
        html.H6(datetime.datetime.fromtimestamp(date)),

        dash_table.DataTable(
            data=df.to_dict('records'),
            columns=[{'name': i, 'id': i} for i in df.columns]
        ),

        html.Hr(),  # horizontal line

        # For debugging, display the raw contents provided by the web browser
        html.Div('Raw Content'),
        html.Pre(contents[0:200] + '...', style={
            'whiteSpace': 'pre-wrap',
            'wordBreak': 'break-all'
        })
    ])



@app.callback(Output('output-data-upload', 'children'),
              [Input('upload-data', 'contents')],
              [State('upload-data', 'filename'),
               State('upload-data', 'last_modified')])
def update_output(list_of_contents, list_of_names, list_of_dates):
    if list_of_contents is not None:
        children = [
            parse_contents(c, n, d) for c, n, d in
            zip(list_of_contents, list_of_names, list_of_dates)]
        return children
```

```
if __name__ == '__main__':
    app.run_server(debug=True)
```

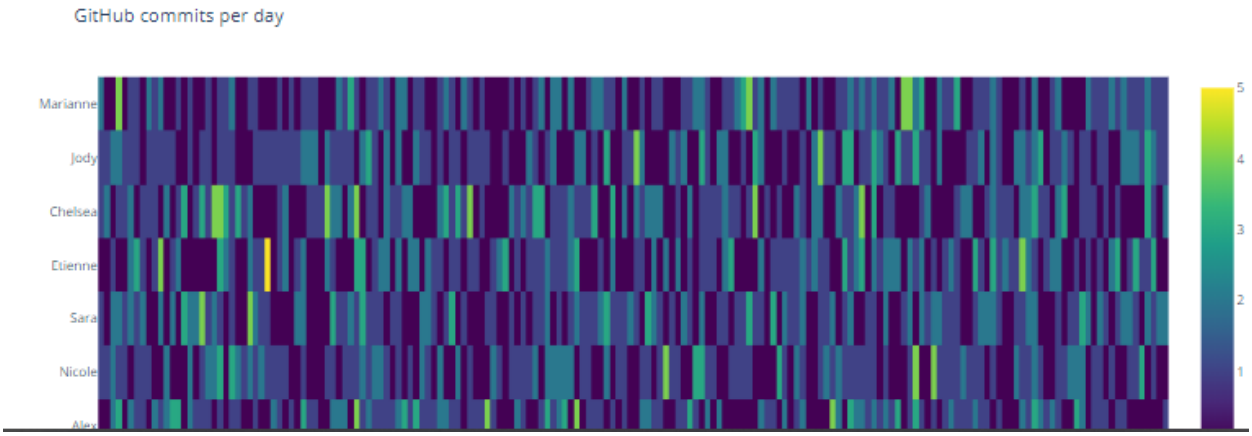


repoId	title	
1	airbnb / javascript	
2	kamranahmedse / developer-roadmap	
3	microsoft / vscode	
4	torvalds / linux	
5	ytdl-org / youtube-dl	Command-line program to download v
6	30-seconds / 30-seconds-of-code	
7	pallets / flask	The Python m
8	chrislgarry / Apollo-11	
9	josephmisiti / awesome-machine-learning	A curated list of awesome Machine
10	django / django	The Web
11	psf / requests	
12	tonsky / FiraCode	
13	karan / Projects	
14	python / cpython	
15	Avik-Jain / 100-Days-Of-ML-Code	
16	knsv / mermaid	Generation of diagram and flowcha
17	tldr-pages / tldr	
18	tuvtran / project-based-learning	
19	ZuzooVn / machine-learning-for-software-engineers	A complete da or st
20	getsentry / sentry	
21	eventlet / eventlet	



Heatmap with Datetime Axis

```
import plotly.graph_objects as go
import datetime
import numpy as np
np.random.seed(1)

programmers = ['Alex','Nicole','Sara','Etienne','Chelsea','Jody','Marianne']

base = datetime.datetime.today()
dates = base - np.arange(180) * datetime.timedelta(days=1)
z = np.random.poisson(size=(len(programmers), len(dates)))

fig = go.Figure(data=go.Heatmap(
        z=z,
        x=dates,
        y=programmers,
        colorscale='Viridis'))

fig.update_layout(
    title='GitHub commits per day',
    xaxis_nticks=36)

fig.show()
```



CONCLUSIONES

Vimos que es relativamente sencillo crear un sistema de recomendación en Python y con Machine Learning. Se debe tener volúmenes altos de información. También es central el valor que utilizaremos como rating -siendo una valoración real de cada usuario o un valor artificial que creemos adecuado-. Luego será cuestión de evaluar entre las opciones de motores user-based, ítem-based y seleccionar la que menor error tenga.

BIBLIOGRAFIA

Donaldson, T. (2014). *Python*. San Francisco, CA: Peachpit Press.

Knowlton, Jim (2009). *Python*. tr: Fernández Vélez, María Jesús (1 edición). Anaya Multimedia-Anaya Interactiva. ISBN 978-84-415-2513-9.

Martelli, Alex (2007). *Python. Guía de referencia*. tr: Gorjón Salvador, Bruno (1 edición). Anaya Multimedia-Anaya Interactiva. ISBN 978-84-415-2317-3.