


Recomendación en plataforma de streaming

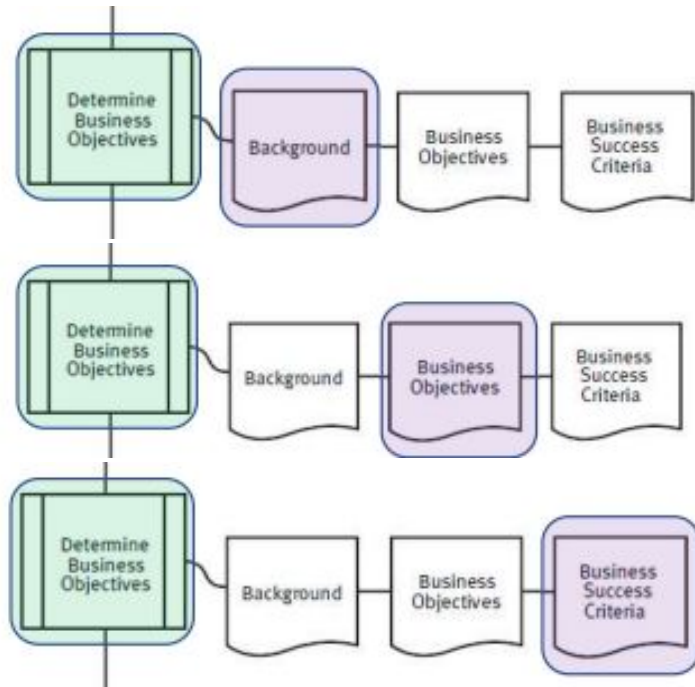


Juan David Cruz
Kennet Santiago Sánchez
Juan Sebastián Pérez
Alexander Sánchez



Business Understanding

Determinar los objetivos de negocio

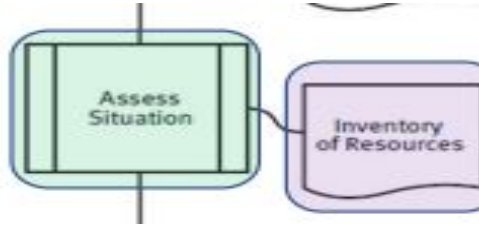


El auge del anime como tendencia mainstream, nos brinda un panorama amplio de aplicación a la hora de recomendar contenido audiovisual.

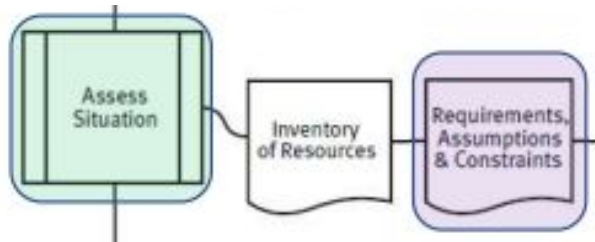
- **Clasificar usuarios.**
- **Recomendar contenido pertinente.**

**((Recomendados Vistos y calificados) /
(promedio de calificación del usuario))
*100**

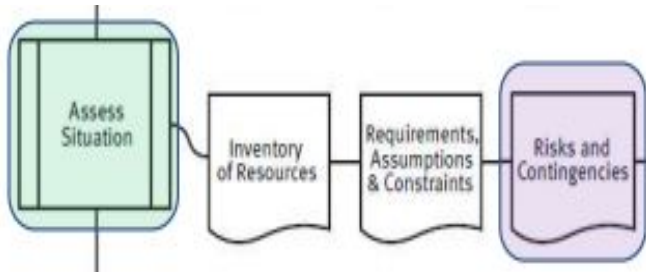
Evaluar la situación



**Personal, datos, recursos
computacionales y de software**

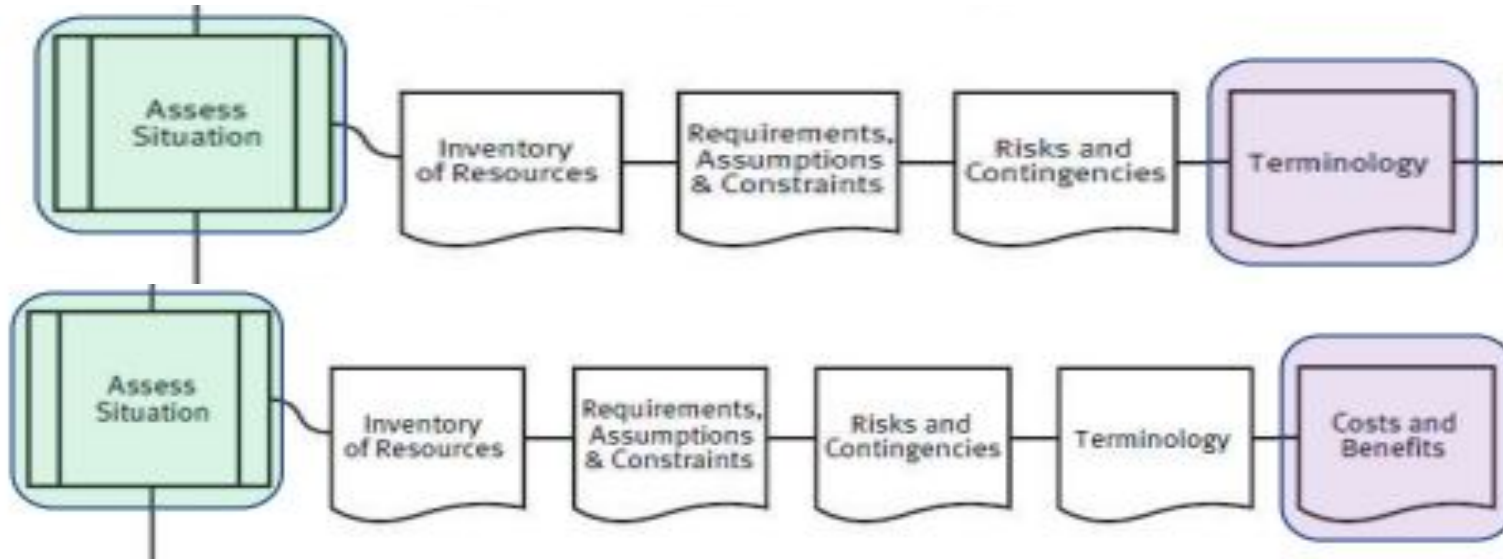


**Requerimientos (realización correcta de las
tareas propuestas).
Suposiciones
y restricciones (posibles limitantes).**

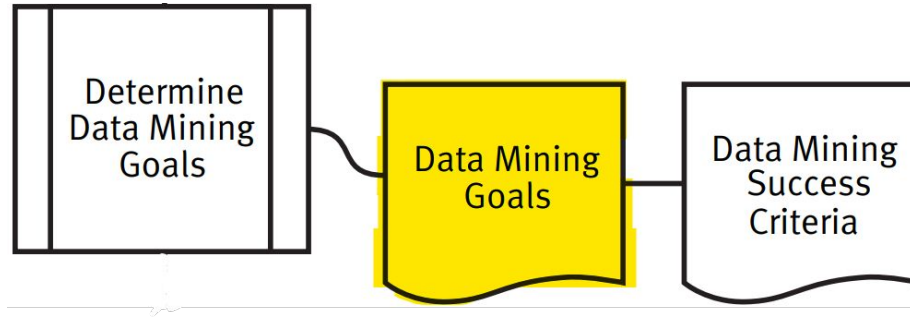


**Posibles riesgos en los que el
desarrollo del proyecto podría incurrir,
y sus contingencias.**

Assess situation



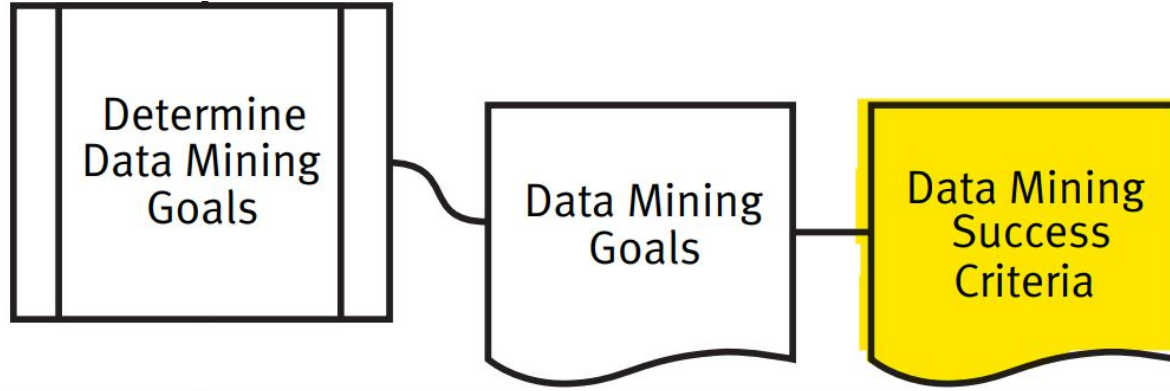
Determinar los objetivos de la minería de datos



- Segmentar
- Clasificar
- Asociar
- Poder

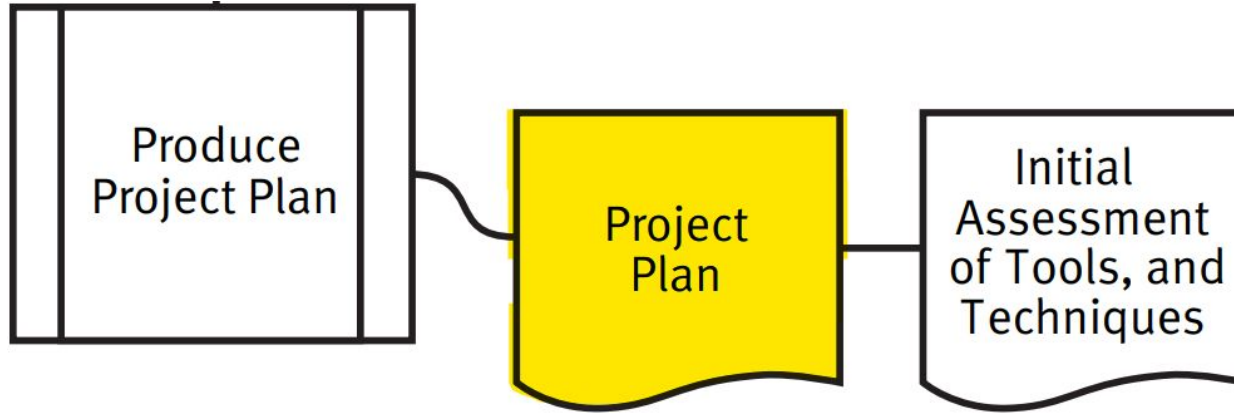
- Asignar
- Usar
- Predecir
- Arrojar

Determinar los criterios de éxito



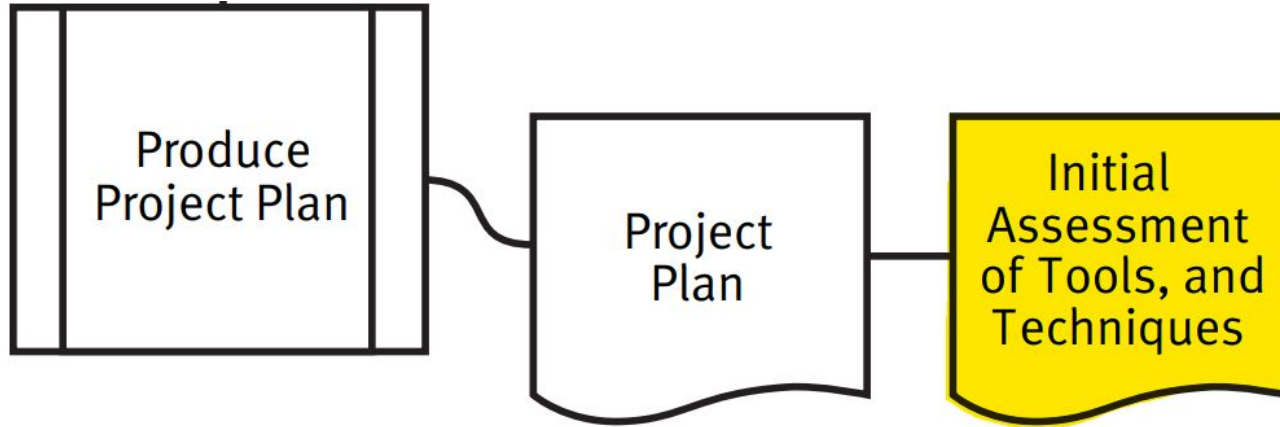
- 80% o más de buenas calificaciones por los usuarios en la plataforma
- Retención de usuarios por más de 3 meses

Producir el plan del proyecto



1. Descarga y entendimiento de la base de datos
2. Clasificación de usuarios
3. Implementación del algoritmo de clasificación
4. Implementación del algoritmo que conecte las clasificaciones de usuarios con el género de las producciones

Valoración inicial de herramientas y técnicas



- **Python**
- **Jupyter Notebooks**
- **Librerías matemáticas y gráficas**

- **KNN**
- **Github**
- **Sphinx**
- **Microsoft Word**



Datos y análisis.

Content

Anime.csv

- anime_id - myanimelist.net's unique id identifying an anime.
- name - full name of anime.
- genre - comma separated list of genres for this anime.
- type - movie, TV, OVA, etc.
- episodes - how many episodes in this show. (1 if movie).
- rating - average rating out of 10 for this anime.
- members - number of community members that are in this anime's "group".

Rating.csv

- user_id - non identifiable randomly generated user id.
- anime_id - the anime that this user has rated.
- rating - rating out of 10 this user has assigned (-1 if the user watched it but didn't assign a rating).

Carga de datos

```
In [2]: 1 #Anime.csv
        2 path = Path(os.getcwd())
        3 path = str(path.parent.absolute())
        4 path = path+"/datos/anime.csv"
        5 dfAnime = pd.read_csv(path,na_values='?')
        6 dfAnime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12294 entries, 0 to 12293
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   anime_id    12294 non-null  int64
1   name        12294 non-null  object
2   genre       12232 non-null  object
3   type        12269 non-null  object
4   episodes    12294 non-null  object
5   rating      12064 non-null  float64
6   members     12294 non-null  int64
dtypes: float64(1), int64(2), object(4)
memory usage: 672.5+ KB
```

```
1 #rating.csv
2 path2 = Path(os.getcwd())
3 path2 = str(path2.parent.absolute())
4 path2 = path2+"/datos/rating.csv"
5 dfRating = pd.read_csv(path2,na_values='?')
6 dfRating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7813737 entries, 0 to 7813736
Data columns (total 3 columns):
#   Column      Dtype
---  -
0   user_id     int64
1   anime_id    int64
2   rating      int64
dtypes: int64(3)
memory usage: 178.8 MB
```

Tipos de dato adecuados

```
1 #anime.csv
2 dfAnime['name'] = dfAnime['name'].astype("string")
3 dfAnime['genre'] = dfAnime['genre'].astype("string")
4 dfAnime['type'] = dfAnime['type'].astype("string")
5 dfAnime['episodes'] = pd.to_numeric(dfAnime.episodes, errors='coerce').dropna().astype(int)
6 dfAnime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 12294 entries, 0 to 12293
```

```
Data columns (total 7 columns):
```

#	Column	Non-Null Count	Dtype
0	anime_id	12294 non-null	int64
1	name	12294 non-null	string
2	genre	12232 non-null	string
3	type	12269 non-null	string
4	episodes	11954 non-null	float64
5	rating	12064 non-null	float64
6	members	12294 non-null	int64

```
dtypes: float64(2), int64(2), string(3)
```

```
memory usage: 672.5 KB
```

rating.csv ya tenia los tipos de datos adecuados

Busqueda y eliminacion de valores nulos o duplicados

anime.csv

```
1 nullValues = []
2
3 for i in range(len(dfAnime)):
4     if(dfAnime.iloc[i].isnull().sum() != 0):
5         nullValues.append(i)
6 print("Se encontraron ",len(nullValues)," valores nulos")
7 #La cantidad de datos con valores nulos no es tan grande en comparacion asi que podemos borrarlos
8 #Unknown es contado como nulo porque no podemos asumir la cantidad de episodios de un anime en base a otros
```

Se encontraron 464 valores nulos

```
1 dfAnime=dfAnime.drop(nullValues)
2 dfAnime['episodes'] = dfAnime['episodes'].astype(int)
3 dfAnime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11830 entries, 0 to 12293
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   anime_id    11830 non-null  int64
1   name        11830 non-null  string
2   genre       11830 non-null  string
3   type        11830 non-null  string
4   episodes    11830 non-null  int32
5   rating      11830 non-null  float64
6   members     11830 non-null  int64
dtypes: float64(1), int32(1), int64(2), string(3)
memory usage: 693.2 KB
```



```

1 duplicados = dfAnime[dfAnime.duplicated()].shape[0]
2 print("Numero de datos duplicados ",duplicados)

```

Numero de datos duplicados 0

```

1 #Convirtamos type en una variable de numeros enteros para poder usar histogramas a futuro
2 dfAnime=dfAnime.replace({'Movie': '0', 'TV': '1', 'OVA': '2', 'ONA': '2', 'Special': '3', 'Music': '4'})
3 dfAnime['type'] = dfAnime['type'].astype(int)

```

```

1 dfAnime.head(5)

```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	0	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	1	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	1	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	1	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	1	51	9.16	151266

rating.csv

```
1 print("La cantidad de datos nulos es:")
2 dfRating.isna().sum().to_frame().T.style.set_properties(**{"background-color": "#2a9d8f","color":"white","border": "1.5px solid #2a9d8f"})
```

La cantidad de datos nulos es:

	user_id	anime_id	rating
--	---------	----------	--------

0	0	0	0
---	---	---	---

```
1 #Eliminamos los -1 porque indican cuando no se califico un anime y siendo asi entonces no nos sirve
2 dfRating = dfRating[dfRating.rating != -1]
```

```
1 duplicados = dfRating[dfRating.duplicated()].shape[0]
2 print("Numero de datos duplicados ",duplicados)
```

Numero de datos duplicados 1


```
In [12]: 1 dfRating.drop_duplicates(keep='first',inplace=True)
         2 duplicados = dfRating[dfRating.duplicated()].shape[0]
         3 print("Numero de datos duplicados ",duplicados)
```

Numero de datos duplicados 0

```
In [13]: 1 dfRating.head(5)
```

Out[13]:

	user_id	anime_id	rating
47	1	8074	10
81	1	11617	10
83	1	11757	10
101	1	15451	10
153	2	11771	10

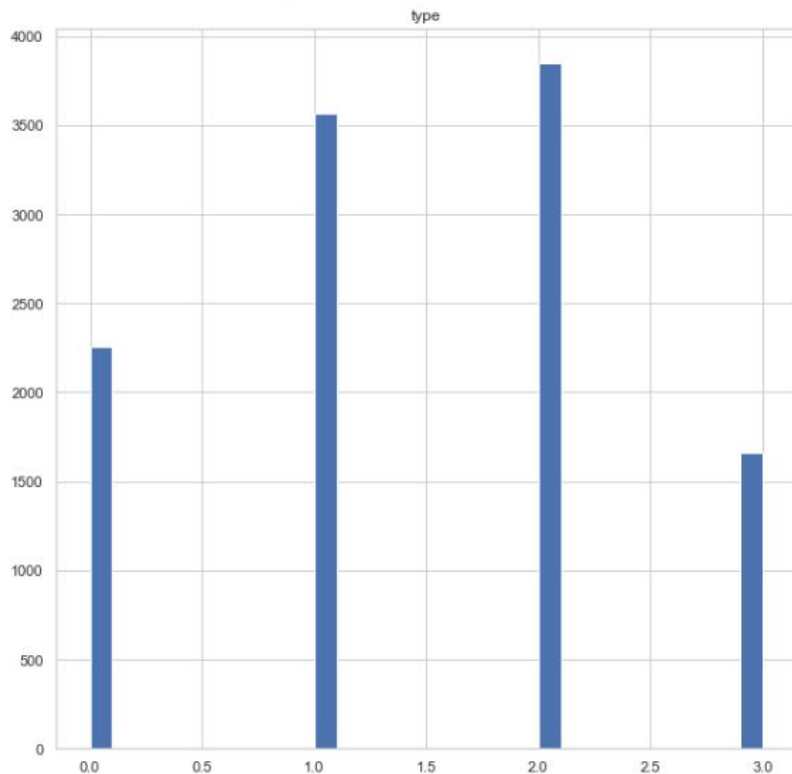
```
In [33]: 1 dfRating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6337240 entries, 47 to 7813736
Data columns (total 3 columns):
#   Column      Dtype
---  -
0   user_id     int64
1   anime_id    int64
2   rating      int64
dtypes: int64(3)
memory usage: 193.4 MB
```

Histogramas

```
In [29]: 1 #Generemos algunas gráficas directamente desde pandas.  
2 atr = 'type'  
3 num_bins = 30 #Número de columnas del histograma  
4 dfAnime.hist(column=atr,range=[0, 3], bins=num_bins, figsize=(10,10)) #Histograma.
```

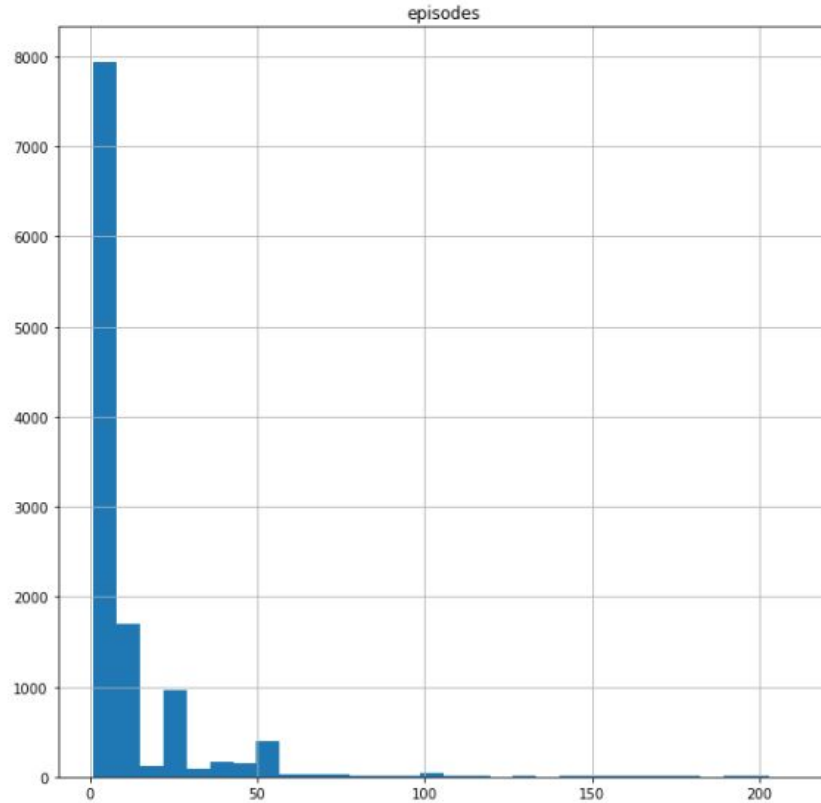
```
Out[29]: array([[<AxesSubplot:title={'center':'type'}>]], dtype=object)
```



- Observamos que aquellos tipos de anime que mas aparecen son "TV" y "OVA"

```
In [14]: 1 #Generemos algunas gráficas directamente desde pandas.  
2 atr = 'episodes'  
3 num_bins = 30 #Número de columnas del histograma  
4 dfAnime.hist(column=atr,range=[1, 210], bins=num_bins, figsize=(10,10)) #Histograma.
```

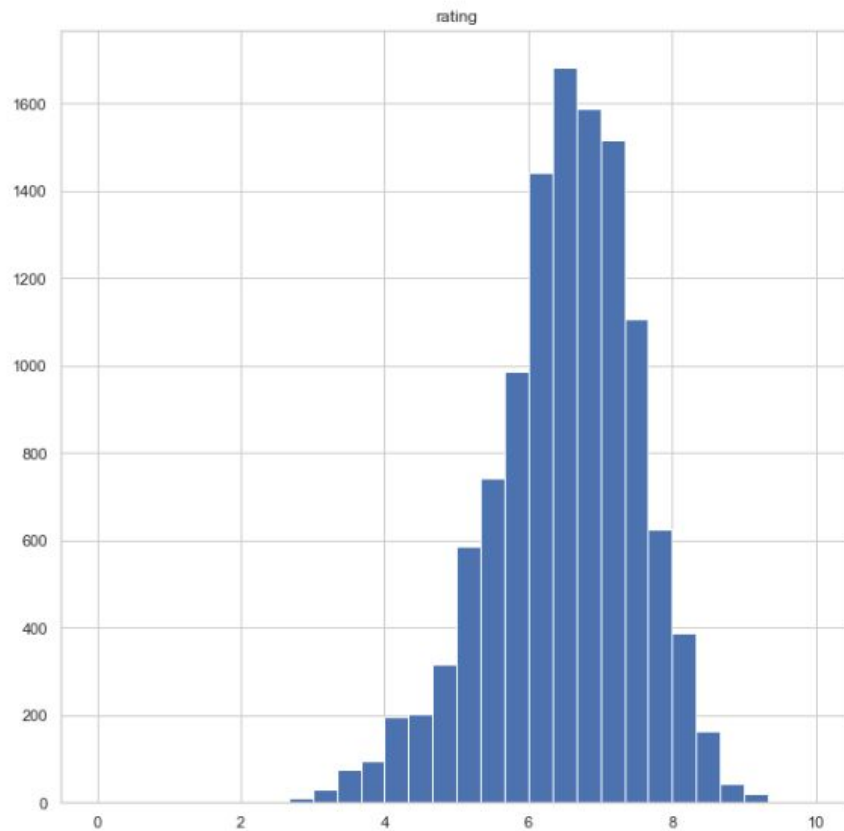
```
Out[14]: array([[<AxesSubplot:title={'center':'episodes'}>]], dtype=object)
```



- Como se puede observar hay una gran cantidad de valores en 1, esto es debido a que muchos de los anime son películas u ovas por lo que solo cuentan con un episodio técnicamente

```
1 #Generemos algunas gráficas directamente desde pandas.  
2 atr = 'rating'  
3 num_bins = 30 #Número de columnas del histograma  
4 dfAnime.hist(column=atr,range=[0, 10], bins=num_bins, figsize=(10,10)) #Histograma.
```

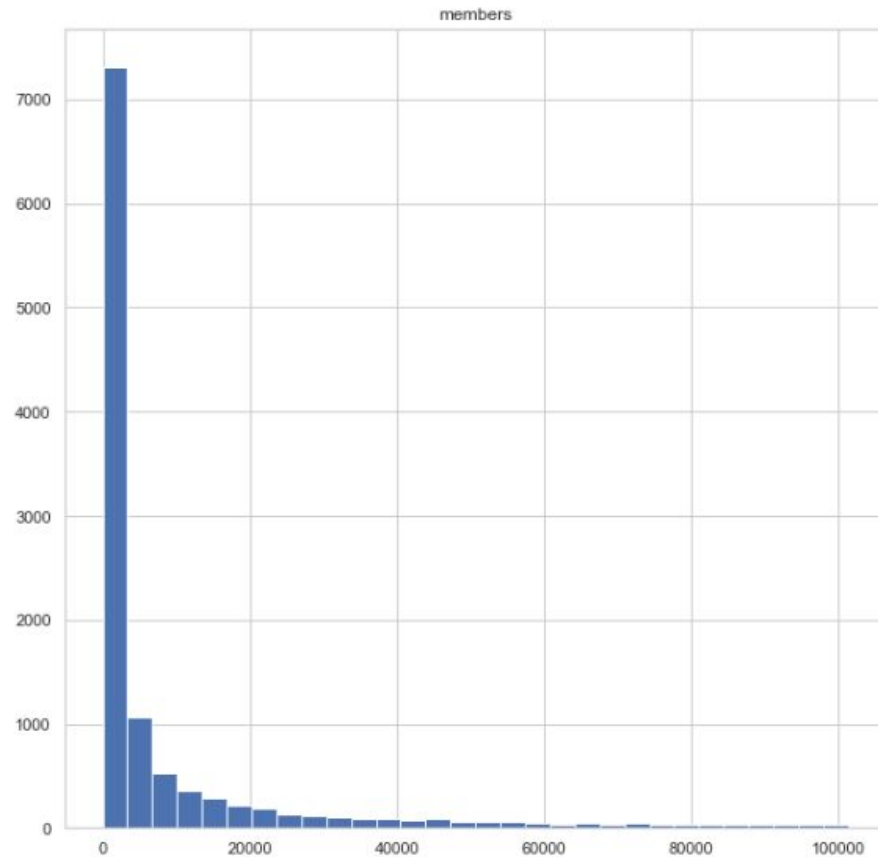
```
array([[<AxesSubplot:title={'center':'rating'}>]], dtype=object)
```



- Entre 6 y 7 están la mayoría de ratings

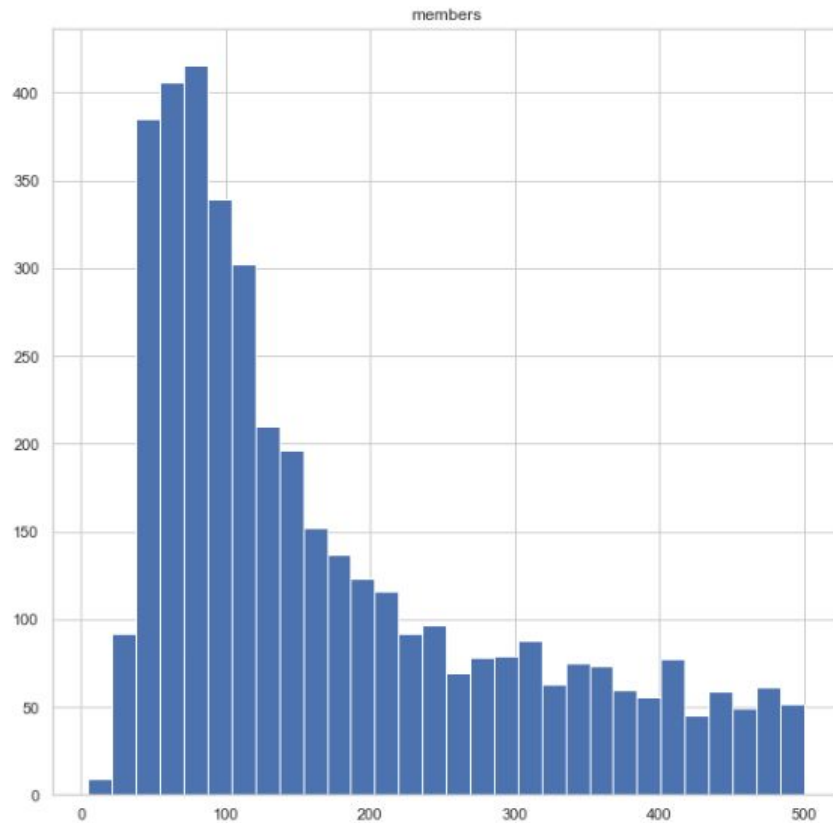
```
1 #Generemos algunas gráficas directamente desde pandas.  
2 atr = 'members'  
3 num_bins = 30 #Número de columnas del histograma  
4 dfAnime.hist(column=atr,range=[5, 101397], bins=num_bins, figsize=(10,10))  
5 #El rango es demasiado amplio, debemos disminuir el rango
```

```
array([[<AxesSubplot:title={'center':'members'}>]], dtype=object)
```



```
1 atr = 'members'
2 num_bins = 30 #Número de columnas del histograma
3 dfAnime.hist(column=atr,range=[5, 500], bins=num_bins, figsize=(10,10))
```

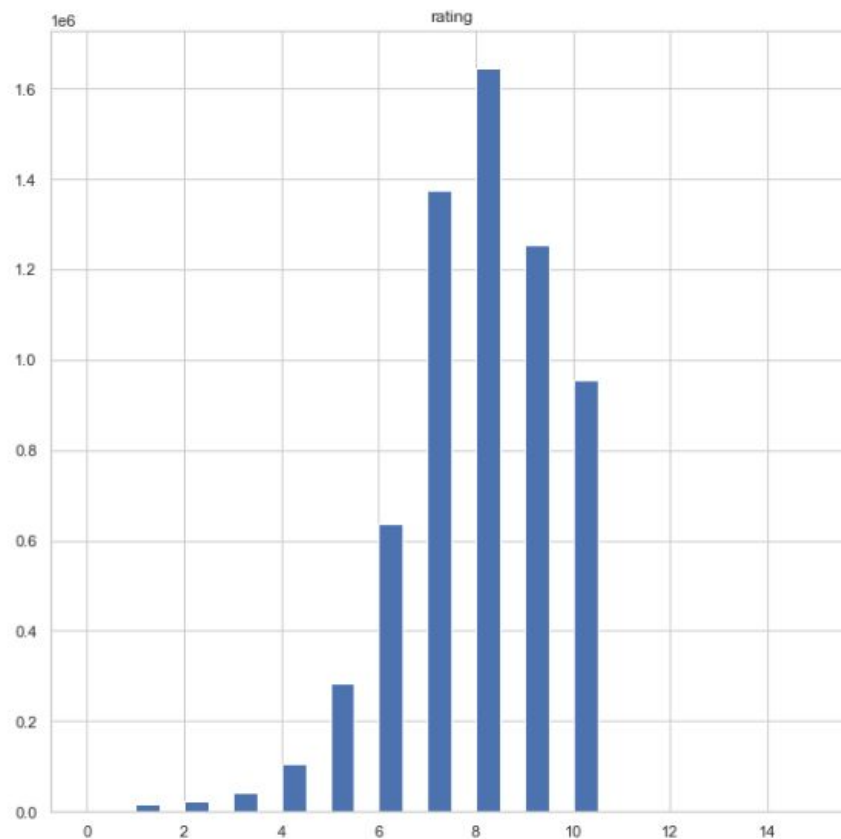
```
array([[<AxesSubplot:title={ 'center': 'members' }>]], dtype=object)
```



- Tener entre 20 y 80 miembros es lo mas comun para la mayoria de animes

```
1 atr = 'rating'
2 num_bins = 30 #Número de columnas del histograma
3 dfRating.hist(column=atr,range=[0, 15], bins=num_bins, figsize=(10,10))

array([[<AxesSubplot:title={'center':'rating'}>]], dtype=object)
```



- La mayoría de usuarios puntua 8 a los animes

Manejo de outliers

Para manejar los outliers decidimos separar el dataset en dos. Uno para las series (tipo 1), y el otro para los demás tipos de producciones.

```
df_series = pd.DataFrame()
df_other_content = pd.DataFrame()

for i in range (len(dfAnime)):
    if(dfAnime.iloc[i].type == 1):
        df_series = df_series.append(dfAnime.iloc[i])
    else:
        df_other_content = df_other_content.append(dfAnime.iloc[i])
```


Límites inferiores y superiores

Para determinar qué datos serán considerados atípicos decidimos utilizar cuartiles.

```
series_iqr = df_series.episodes.quantile(0.75) - df_series.episodes.quantile(0.25)
series_lower_bound = df_series.episodes.quantile(0.25) - (1.5 * series_iqr)
series_upper_bound = df_series.episodes.quantile(0.75) + (1.5 * series_iqr)

other_content_iqr = df_other_content.episodes.quantile(0.75) - df_other_content.episodes.quantile(0.25)
other_content_lower_bound = df_other_content.episodes.quantile(0.25) - (1.5 * other_content_iqr)
other_content_upper_bound = df_other_content.episodes.quantile(0.75) + (1.5 * other_content_iqr)
```

Separación de datos atípicos

Los datos que estén por debajo del límite inferior, o por encima del límite inferior serán considerados atípicos, y serán removidos del dataset.

```
series_atypicals = []
other_content_atypicals = []

for index in range(len(df_series)):
    data = df_series.iloc[index]

    if(data.episodes < series_lower_bound or data.episodes > series_upper_bound):
        series_atypicals.append(index)

for index in range(len(df_other_content)):
    data = df_other_content.iloc[index]

    if(data.episodes < other_content_lower_bound or data.episodes > other_content_upper_bound):
        other_content_atypicals.append(index)
```

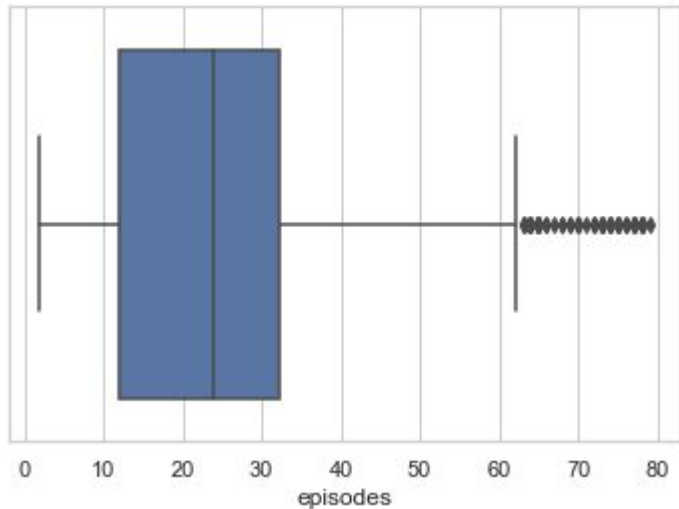
```
depurated_series = pd.DataFrame()
depurated_other_contents = pd.DataFrame()

for i in range(len(df_series)):
    if(not series_atypicals.__contains__(i)):
        depurated_series = depurated_series.append(df_series.iloc[i])

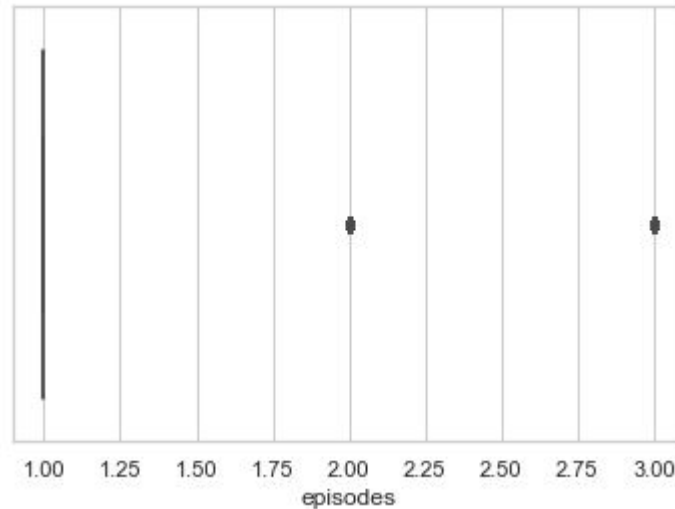
for i in range(len(df_other_content)):
    if(not other_content_atypicals.__contains__(i)):
        depurated_other_contents = depurated_other_contents.append(df_other_content.iloc[i])
```

Diagramas de caja, episodios

Series

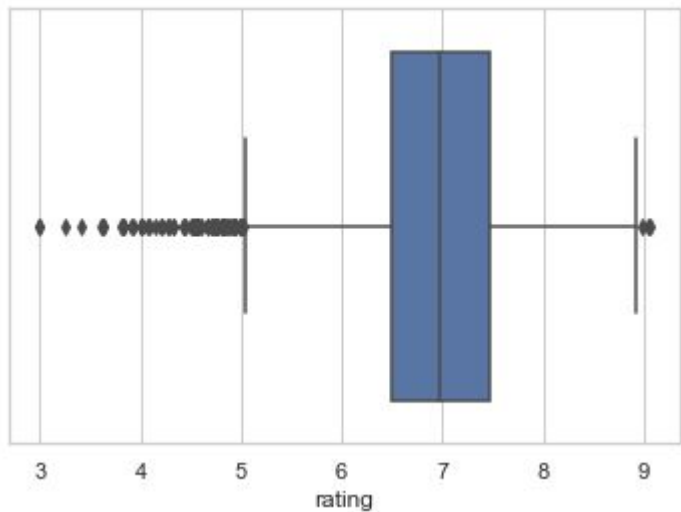


Otras producciones

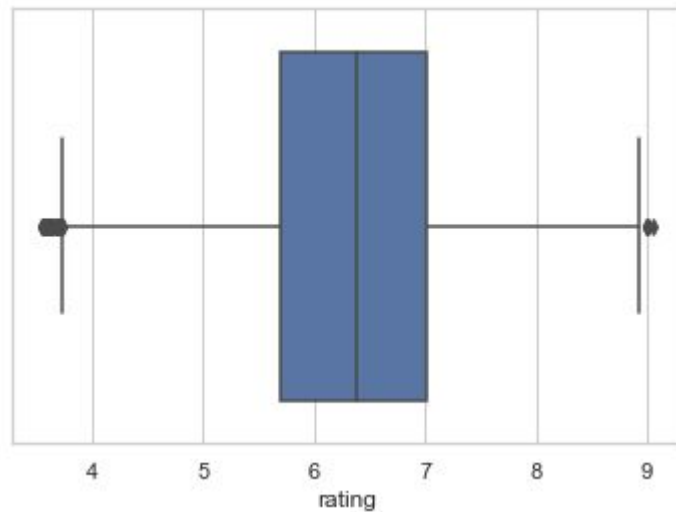


Diagramas de caja, calificación

Series

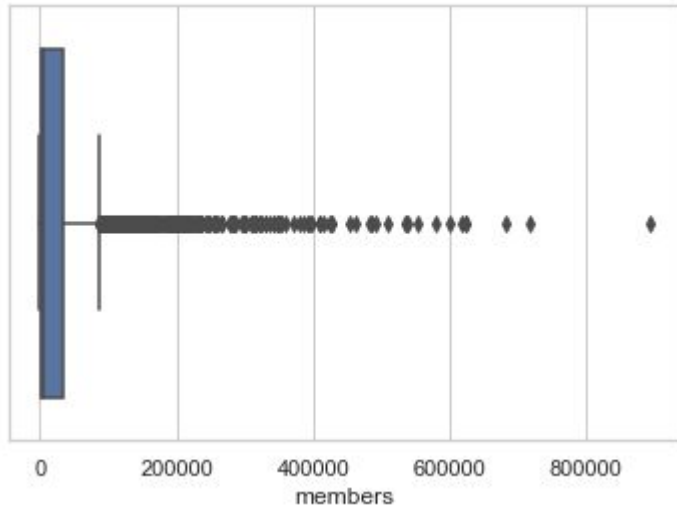


Otras producciones

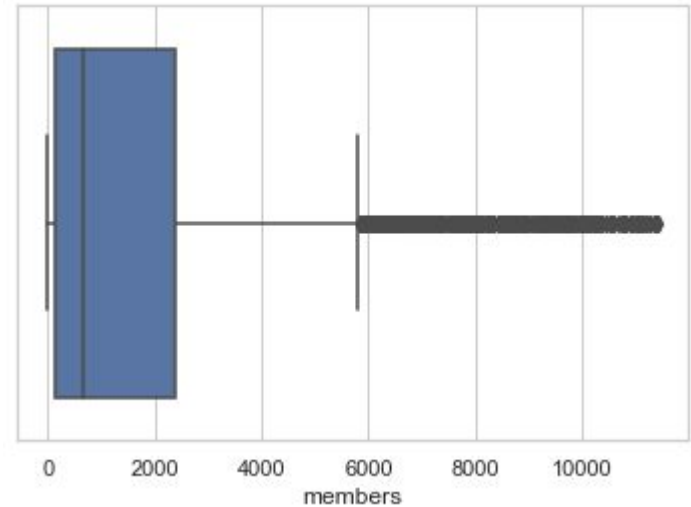


Diagramas de caja, espectadores

Series

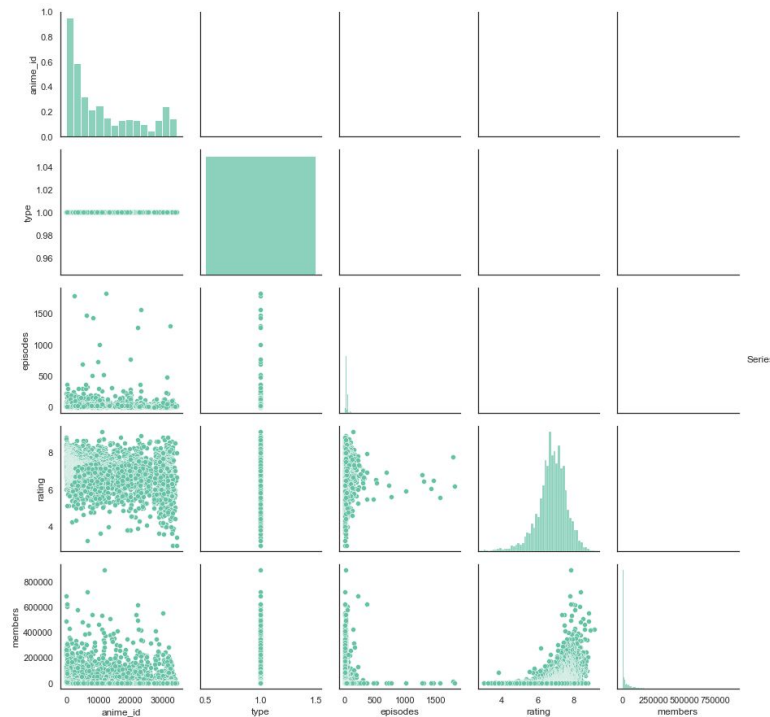


Otras producciones



Correlación de variables en las series

- Hay poca correlación entre la calificación y la cantidad de episodios de una serie, sin embargo esta parece ser positiva (a mayor número de episodios mejor calificación)
- Hay una correlación positiva entre la cantidad de usuarios que han visto una serie y su calificación (a mayor número de usuarios mayor calificación)
- Hay una correlación negativa entre la cantidad de episodios y los usuarios que ven una serie (a mayor número de episodios menos usuarios la ven)



Correlación de variables en las producciones especiales

Hay poca correlación entre la calificación y la cantidad de episodios de una producción especial, sin embargo esta parece ser positiva (a mayor número de episodios mejor calificación)

Hay una correlación positiva entre la cantidad de usuarios que han visto una producción especial y su calificación (a mayor número de usuarios mayor calificación)

Hay una correlación negativa entre la cantidad de episodios y los usuarios que ven una producción especial (a mayor número de episodios menos usuarios la ven)

