

SISTEMA RECOMENDACION ANIME

Juan David Cruz
Kennet Santiago Sanchez
Alexander Sanchez
Juan Sebastián Pérez



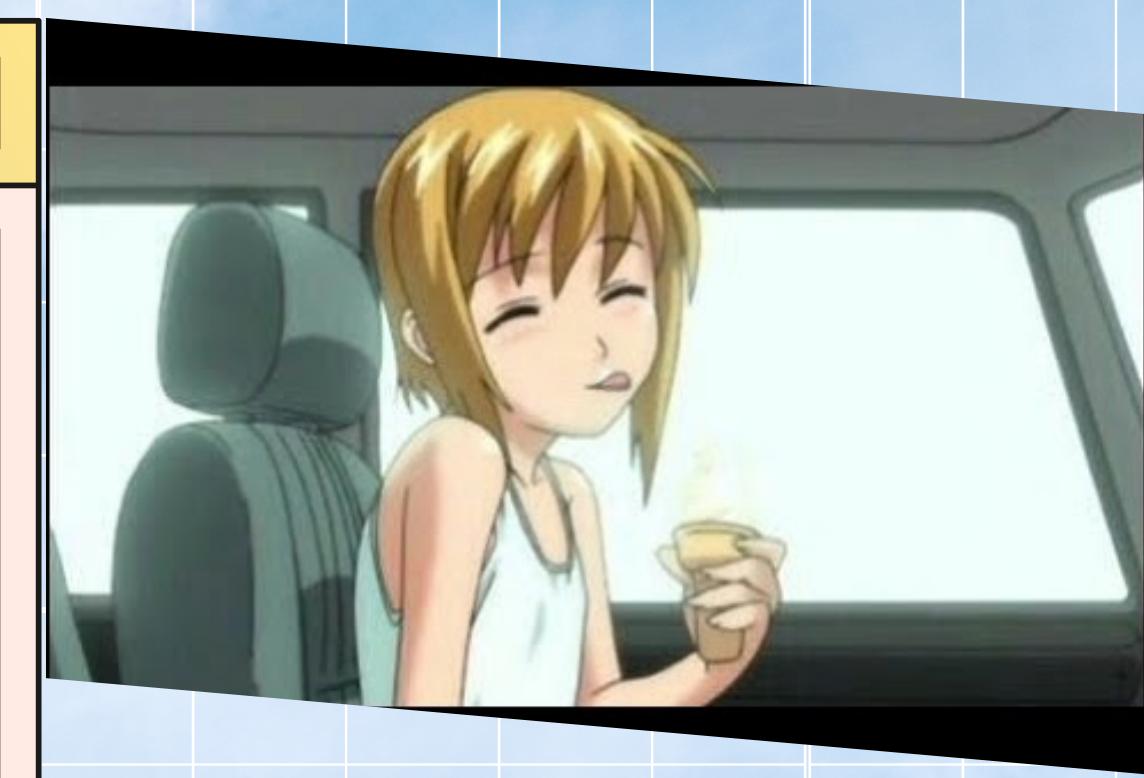
Entendiendo el objetivo del proyecto

ooo

+

Las principales propuestas se centran en:

1. ¿Cómo recomendar anime pertinente a un usuario basado en su comportamiento dentro de una plataforma de streaming?
2. ¿Cómo brindar un apartado caracterizado por la unicidad y personalización en función de los gustos del usuario?



¿Cómo logramos verificar si cumplimos?



1. Para brindar unicidad y sensación de personalización se ideó usar los géneros vistos por el usuario para las recomendaciones.
2. El sistema de recomendaciones se valida verificando si en el conjunto de recomendaciones posibles contiene un porcentaje alto de animes los cuales el usuario ya ha visto

Recomendaciones que ya haya visto / total Recomendaciones

Carga de datos

```
#Anime.csv  
path = Path(os.getcwd())  
path = str(path.parent.absolute())  
path = path+"/datos/anime.csv"  
dfAnime = pd.read_csv(path,na_values='?')  
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
```

```
#rating.csv  
path2 = Path(os.getcwd())  
path2 = str(path2.parent.absolute())  
path2 = path2+"/datos/rating.csv"  
dfRating = pd.read_csv(path2,na_values='?')  
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
```

Valores nulos y duplicados

- La cantidad de datos con valores nulos no es tan grande en comparacion al total así que podemos borrarlos.
- Eliminamos tambien los valores innecesarios. (En el caso de rating.csv, los rating con -1 son inutiles para nuestro problema puesto que simbolizan que el usuario no ha calificado el anime)

Ajuste de tipos de datos

- Convertimos "episodes" en una variable de numeros enteros para poder realizar adecuadamente el conteo
- Convertimos "type" en una variable de tipo categoria para poder realizar analisis sobre este

C

Decisión respecto a las películas

1

Separación de categorías

1

Selección de géneros

Teniendo en cuenta nuestro contexto y para evitarnos entrenar mas modelos de lo necesario, solo usaremos los animes que sean del tipo "series"

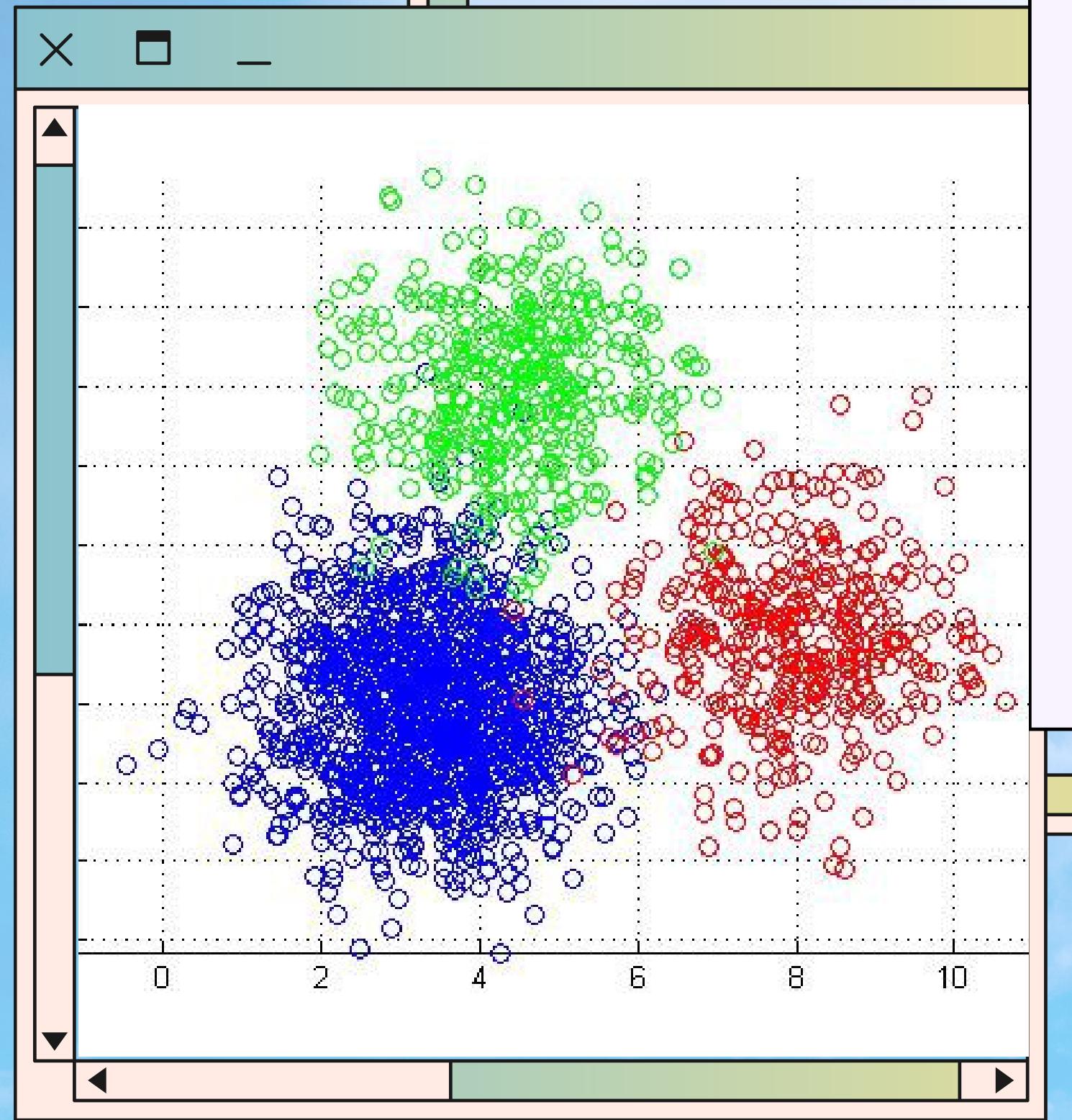
La estrategia aplicada es convertir la información contenida en esta columna en columnas correspondientes a los géneros.

De esta manera, una fila tendra 1 o 0 en una columna de genero dependiendo de si el anime pertenece o no a esta.

Los generos que contiene el dataframe son demasiados, es conveniente no usarlos todos. Por tanto la estrategia se baso en reducirlos y unirlos en función de los más conocidos.

Para resolver nuestro problema de recomendación de anime necesitaremos dos cosas, en primer lugar, unas categorías que clasifiquen adecuadamente los animes del dataframe, una vez hecho esto, usaremos un modelo de clasificación para adecuar a los datos las clasificaciones previamente creadas

Solucion

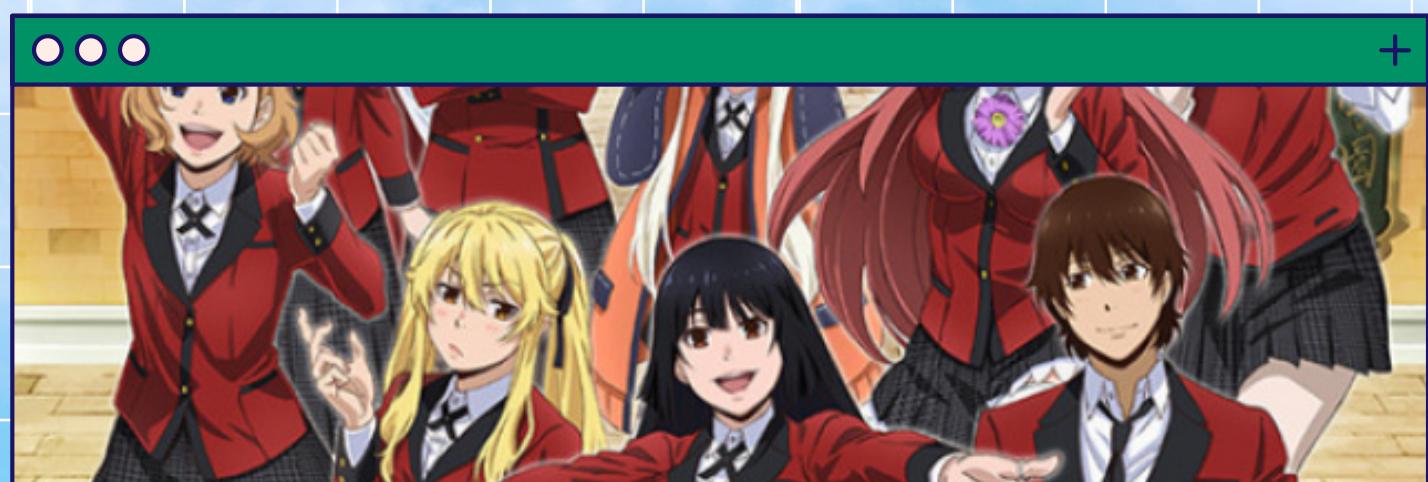
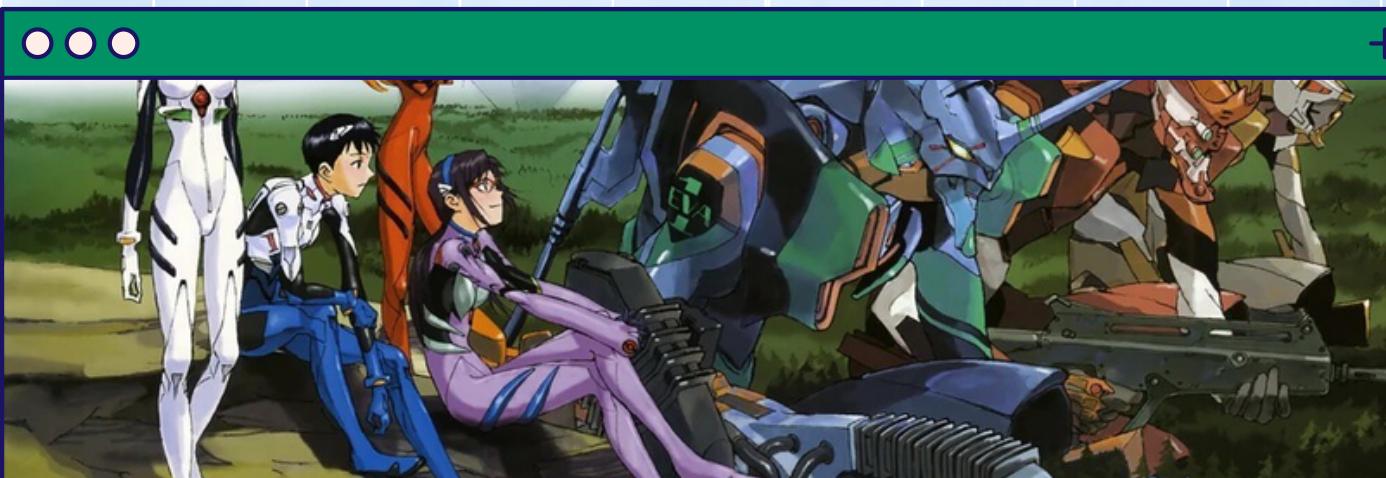
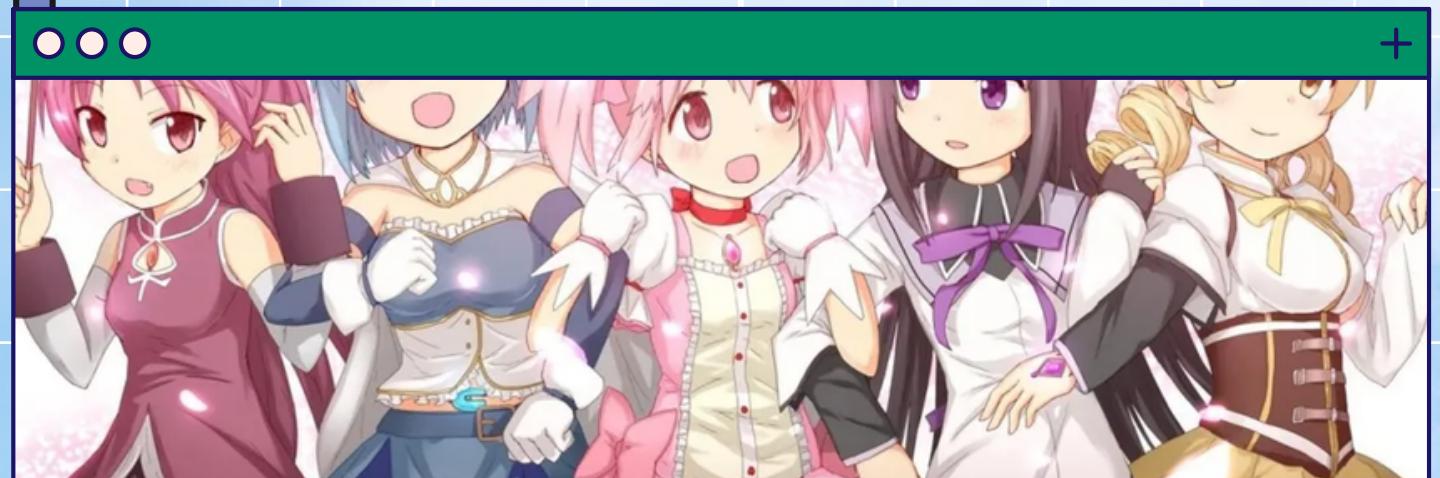


La construcción de un modelo de clustering nos ayuda a categorizar adecuadamente los animes. Debemos asegurarnos de usar parámetros adecuados y normalizar los datos y ajustar los hyperparametros adecuados.

Se usara KMeans++ puesto que es una version mejorada del kmeans standar que inicializa los centroides de manera inteligente.

Se eliminan 3 variables de genero

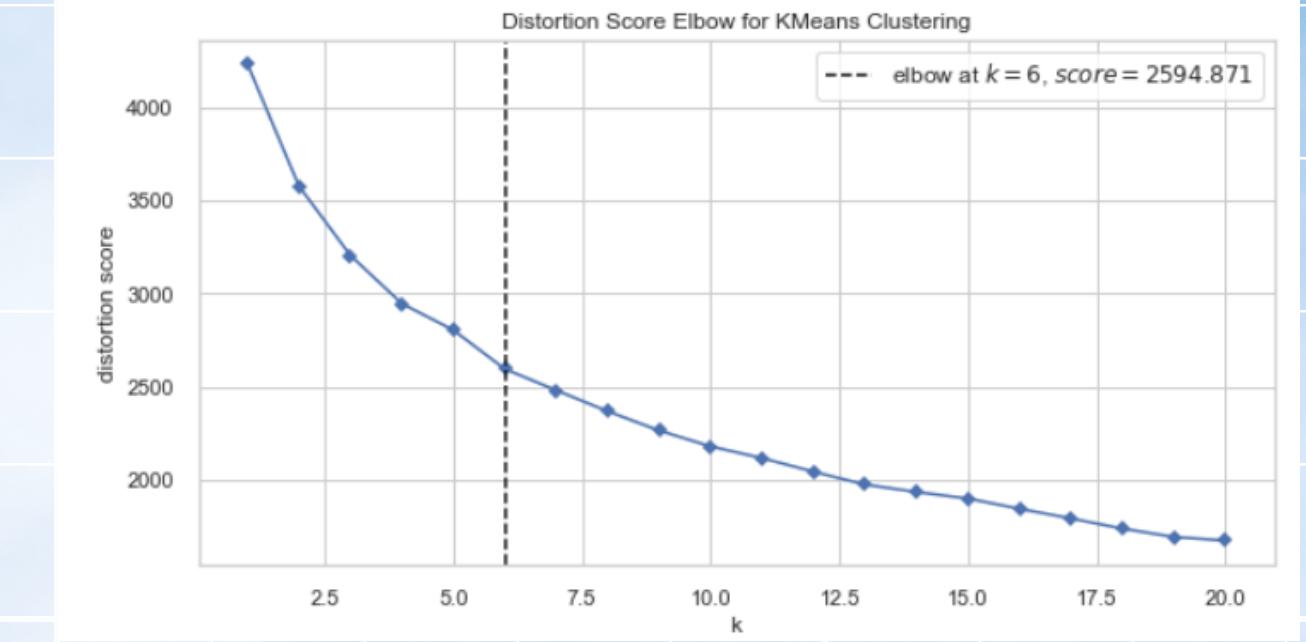
Shoujo, Mecha y School



...

+

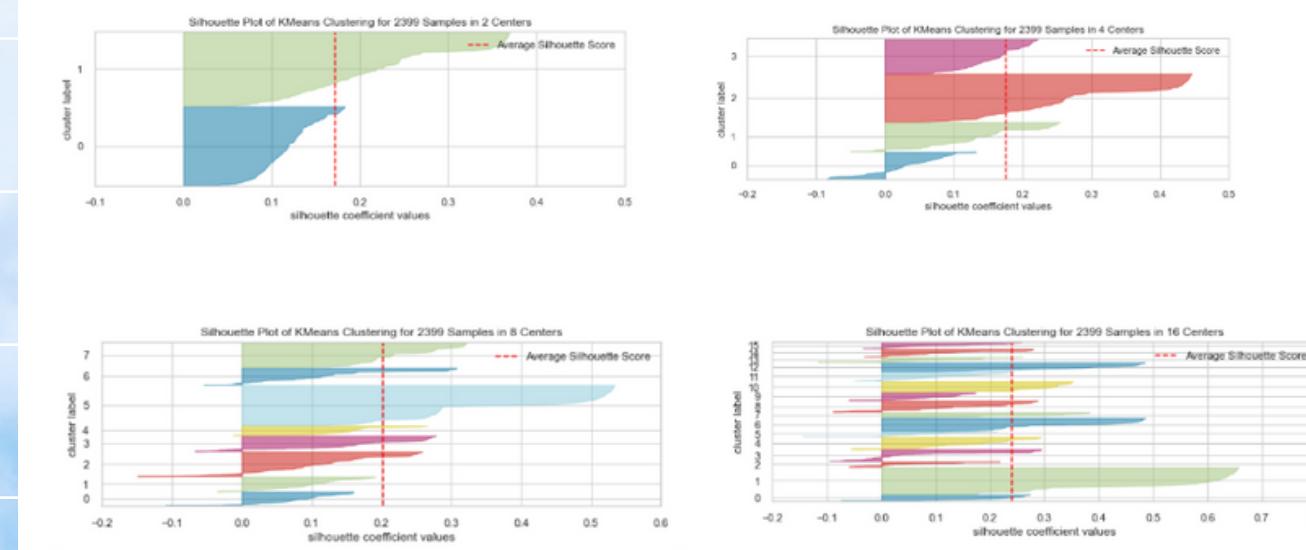
Metodo del codo



...

+

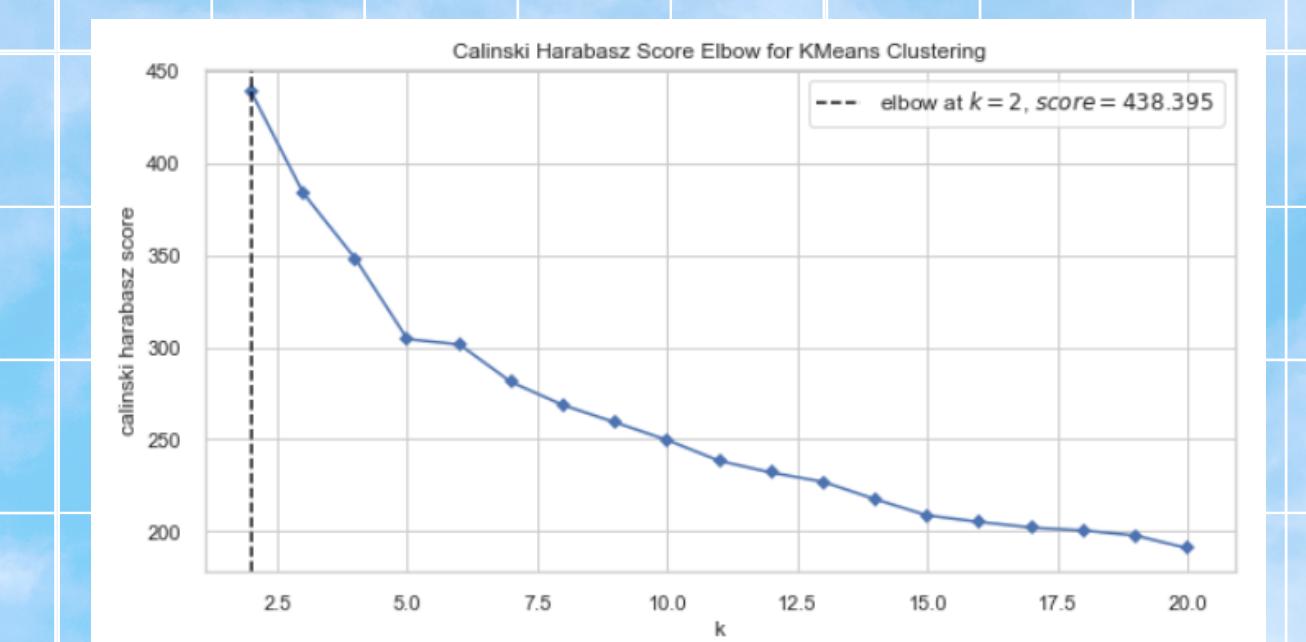
Metodo de las siluetas

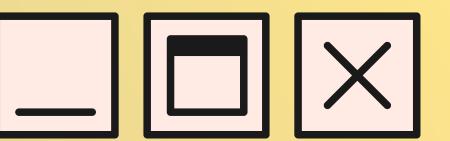


...

+

Metodo de calinski-harabasz

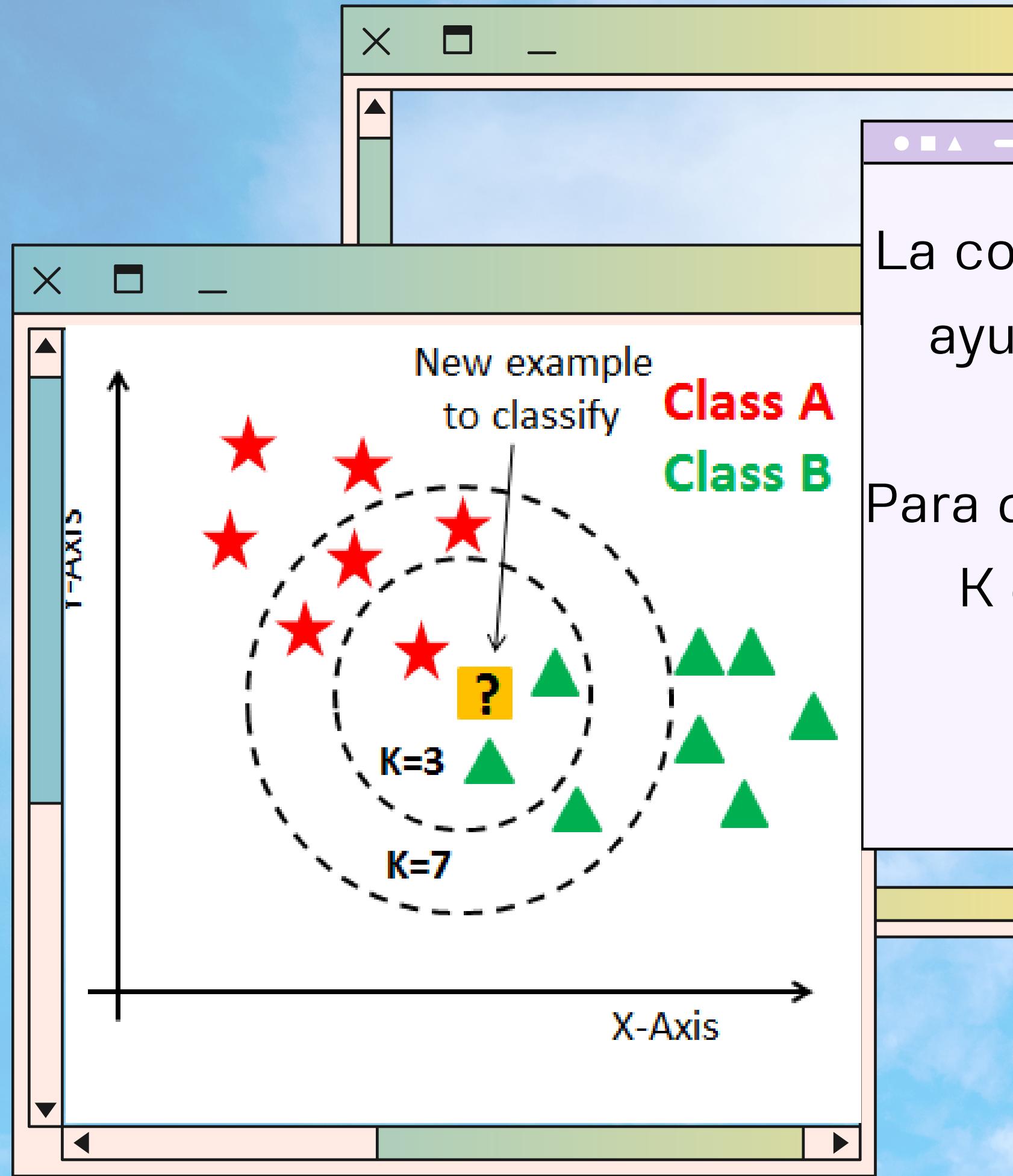




Como los métodos no proveen una solución clara, se usa como valor de K a 12

```
1 #K-means
2 -----
3 k = 12 #Número de grupos que se escogió después del análisis previo.
4
5 #Ahora se instancia el objeto para utilizar el agrupamiento con k-means.
6 #Para ver todas las opciones del constructor, consulte: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
7 #Nota: el algoritmo de k-means disponible en scikit-learn funciona únicamente con métricas euclídeas.
8 #Si requiere aplicar k-means con otras métricas de distancia, puede consultar la documentación de scikit-learn.
9 kmeans = KMeans(n_clusters = k,                      #Se define el número de grupos.
10                  init      = init,                 #Se define el método de inicialización.
11                  n_init    = n_init,                #Número de inicializaciones aleatorias.
12                  max_iter  = max_iter,              #Número MÁXIMO de iteraciones posibles.
13                  random_state = random_seed)
14
15 #Hagamos el ajuste (i.e.: encontraremos los centroides).
16 kmeans.fit(df_x_train)
17 predict = kmeans.predict(df_series_norm)
18
19 df_series['Classification'] = pd.Series(predict, index=df_series_norm.index)
```

Se usa K porque es el número de géneros resultantes después de la previa eliminación



La construcción de un modelo de clasificación nos ayuda a adecuar a los datos las clasificaciones previamente creadas. Para que esto tenga sentido, se debe usar la misma K que se utilizó para el modelo de clustering.

**Se usará el algoritmo K-Nearest-
Neighbours (KNN).**

ooo

+

Entradas y salida esperada

ooo

+

Entrenamiento del modelo

División del conjunto de datos

```
1 input_attr = features
2 target_attr = 'classification'
3
4 input_attr.remove('rating')
5
6 print(input_attr)
7
8 df_x_knn = deepcopy(df_final[input_attr])
9 df_y_knn = deepcopy(df_final[target_attr])
10
```

Modelo

```
1 k = 12
2
3 knn = neighbors.KNeighborsClassifier(n_neighbors=k)
4 knn.fit(df_x_train_knn, df_y_train_knn)
```

¿Cómo evaluar el modelo?

Puesto que nuestro modelo de clasificación surgió del modelo de clustering, es evidente que los resultados obtenidos por metodos de evaluacion de KNN seran excesivamente altos.

Por tanto, no tiene sentido evaluar los modelos de manera individual sino de manera conjunta.

Se evaluara tanto cualitativa como cuantitativamente.



Fusion de animes

Currently Watching Completed On Hold Plan to Watch

Watching

Rank	Title	Type	Score	Episodes	Status
# 1	Bakuten Shoot Beyblade G Revolution	TV	7	26/52 +	Completed
# 2	Kagaku Ninja-tai Gatchaman	TV	8	20/105 +	Completed
# 3	Urusei Yatsura	TV	8	19/195 +	Completed
# 4	Yu-Gi-Oh! Duel Monsters	TV	224+	Score: 8	Watching
# 5	Mobile Suit Zeta Gundam	TV	5/224 +	Score: 7	Watching
# 6	Dr. Slump: Arale-chan	TV	2/243 +	Score: 9	Watching
# 7	Getter Robo	TV	2/51 +	Score: 8	Watching
# 8	Peeping Life TV: Season 17!	TV	2/12 +	Score: 8	Watching
# 209	Melty Blood: Type Lumina	Movie	7	Score: 8	Completed
# 210	Big Order	OVA	8	Score: 8	Completed
# 211	Aquarion Logos	TV	11	Score: 8	Completed
# 212	Rolling Girl: Chibi Rolling Girls Korokoro Gekijou	TV	13	Score: 8	Completed
# 213	Hacka Doll The Animation	ONA	13	Score: 8	Completed
# 214	New Initial D Movie: Legend 3 - Mugen	Movie	13	Score: 8	Completed
# 215	Overlord: Ple Ple Pleiades	ONA	60 +	Score: 8	Completed
# 216	Monster Musume no Iru Nichijou: Hobo Mainichi OO! Namappoi Douga	TV	13	Score: 8	Completed
# 217	JK-kun!	TV	13	Score: 8	Completed
# 218	Owari no Seraph: The Beginning of the End	Special	13	Score: 8	Completed
# 219	One Punch Man Specials	TV	13	Score: 8	Completed
# 220	Comet Lucifer: Garden Indigo no Shisou Kara	ONA	13	Score: 8	Completed
# 221	Koukaku no Pandora: Ghost Urn	Movie	13	Score: 8	Completed
# 222	Kono Subarashii Sekai ni Shukufuku wo!	TV	13	Score: 8	Completed
# 223	Luck & Logic	ONA	13	Score: 8	Completed

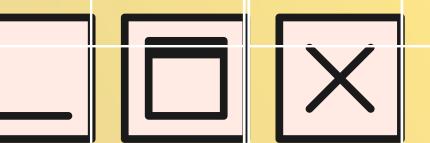
TOTALS:

TV: 134, MOVIES: 44, OVA: 33, SPCL: 11, EPS: 80, DAYS: 1.30, MEAN SCORE: 7.8, SCORE DEV: 0.48

MyAnimeList.net is a property of MyAnimeList, LLC. ©2015 All Rights Reserved.

En una plataforma de anime es evidente que un usuario estará en contacto con múltiples animes y que por tanto, le podrán gustar o no más de un anime.

Para esto, simularemos este comportamiento a través de crear nuevos datos usando la información de dos o más animes.

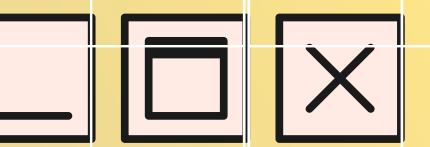


Fusionando "Gintama" y "Mushishi Zoku Shou"

	name	rating	members	Comedy	Action	Adventure	Sci-Fi	Fantasy	Shounen	Drama	Romance	Kids	Supernatural	Magic
Gintama°		9.25	114262	1	1	0	1	0	1	0	0	0	0	0

	name	rating	members	Comedy	Action	Adventure	Sci-Fi	Fantasy	Shounen	Drama	Romance	Kids	Supernatural	Magic
Mushishi Zoku Shou		8.8	101351	0	0	1	0	1	0	0	0	0	1	0

	name	rating	members	Comedy	Action	Adventure	Sci-Fi	Fantasy	Shounen	Drama	Romance	Kids	Supernatural	Magic
Gintama° Mushishi Zoku Shou		9.025	107806.5	1	1	1	1	1	1	0	0	0	1	0



Añadiendo a esta fusion "Jojos Bizarre Adventure"

	name	rating	members	Comedy	Action	Adventure	Sci-Fi	Fantasy	Shounen	Drama	Romance	Kids	Supernatural	Magic
JoJo no Kimyou na Bouken: Stardust Crusaders 2...	JoJo no Kimyou na Bouken: Stardust Crusaders 2...	8.6	93657	0	1	1	0	0	1	1	0	0	1	0
	name	rating	members	Comedy	Action	Adventure	Sci-Fi	Fantasy	Shounen	Drama	Romance	Kids	Supernatural	Magic
Gintama° Mushishi Zoku Shou JoJo no Kimyou na...	Gintama° Mushishi Zoku Shou JoJo no Kimyou na...	8.8125	100731.75	1	1	1	1	1	1	1	0	0	1	0

ooo

+

ooo



Recomendando animes populares

```
def recommendation(df):
    predicted_value = int(knn.predict(df.drop('name',axis=1)))
    category=df_series[df_series.Classification == predicted_value]
    return category.sort_values(by=['members','rating'], ascending=False)
```



```
1 ex=recommendation(fusionByNameList("Pokemon"))
2 ex.head(10)
```

Sekai ni Shukufuku wo!	10	8.03	244877	1	0	1	0	1	0	0	0	0	0
Pokemon	276	7.43	229157	1	1	1	0	1	0	0	0	0	1
Ookami to Koushinryou II	12	8.46	210491	0	0	1	0	1	0	0	0	1	0
Digimon Adventure	54	7.89	182208	1	1	1	0	1	0	0	0	0	1





```
ex=recommendation(fusionByNameList("Fate/Zero"))
ex.head(10)
```

name	episodes	rating	members	Comedy	Action	Adventure	Sci-Fi	Fantasy	Shounen
Shingeki no Kyojin	25	8.54	896229	0	1	0	0	1	1
Sword Art Online	25	7.83	893100	0	1	1	0	1	0
Ao no Exorcist	25	7.92	583823	0	1	0	0	1	1
Sword Art Online II	24	7.35	537892	0	1	1	0	1	0
Akame ga Kill!	24	7.84	492133	0	1	1	0	1	0
Fate/Zero	13	8.51	453630	0	1	0	0	1	0
Kuroshitsuji	24	8.06	424919	1	1	0	0	1	1
Fate/stay night	24	7.58	374880	0	1	0	0	1	0
Zero no Tsukaima	13	7.62	346828	1	1	1	0	1	0
Fate/Zero 2nd Season	12	8.73	340973	0	1	0	0	1	0



```
ex=recommendation(fusionByNameList("Neon Genesis Evangelion"))
ex.head(10)
```

	name	episodes	rating	members	Comedy	Action	Adventure	Sci-Fi	Fantasy
	Code Geass: Hangyaku no Lelouch	25	8.83	715151	0	1	0	1	0
	Steins;Gate	24	9.17	673572	0	0	0	1	0
	Code Geass: Hangyaku no Lelouch R2	25	8.98	572888	0	1	0	1	0
	Tengen Toppa Gurren Lagann	27	8.78	562962	1	1	1	1	0
	One Punch Man	12	8.82	552458	1	1	0	1	0
	Psycho-Pass	22	8.50	509109	0	1	0	1	0
	Cowboy Bebop	26	8.82	486824	1	1	1	1	0
	Neon Genesis Evangelion	26	8.32	461946	0	1	0	1	0
	Guilty Crown	22	7.81	460959	0	1	0	1	0
	Deadman Wonderland	12	7.48	453454	0	1	0	1	0

Evaluacion cuantitativa

```
1 import random
2 predictions = 0
3 users = df_final['user_id'].unique()
4 for user in users:
5     userRatings = df_final.loc[df_final['user_id'] == user]
6     randomNumber = random.randint(0, len(userRatings)-1)
7     firstRating = userRatings.iloc[randomNumber]
8     percent = seenVsRecommendation(userRatings,firstRating)
9     if(percent >= 0.1):
10         predictions += 1
11
12 averagePercent = predictions/len(users)
```

De esta manera se obtiene el siguiente resultado:

```
1 print(averagePercent)
```

0.25787404430646027

Lo que indica que en el 25% de los casos, 10% de los animes recomendados por el modelo le han gustado previamente a el usuario