**List of frameworks and libraries to build recommender systems:**

1. Recsys <https://github.com/ocelma/python-recsys>
2. Mrec <http://mendeley.github.io/mrec/index.html>
3. Annoy <https://pypi.org/project/annoy/>
4. LightFM <https://github.com/lyst/lightfm>
5. crab <http://muricoca.github.io/crab/>
6. recommendation\_system <https://github.com/CodingVanGogh/Recommendation-System>
7. popularity\_based\_recommendation <https://github.com/vyashemang/popularity_based_recommendation/blob/master/Popularity%20based%20song%20recommendations.ipynb>
8. fastai <http://docs.fast.ai/>
9. surprise <http://surpriselib.com/>
10. implicit <https://github.com/benfred/implicit>
11. pyreclab <https://github.com/gasevi/pyreclab>
12. Turicreate <https://apple.github.io/turicreate/docs/userguide/> <https://github.com/apple/turicreate/tree/master/userguide/recommender>
13. Rexy <https://github.com/kasramvd/Rexy>
14. Lenskit <https://lkpy.lenskit.org/en/latest/>
15. SpotLight <https://github.com/maciejkula/spotlight>
16. Case Recommender <https://github.com/caserec/CaseRecommender>

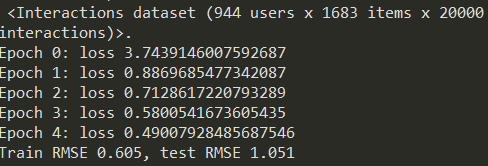
**Spotlight.**

Utiliza Pytorch, para matrix factorization. Para retroalimentación implícita, explícita, latent representations y modelos secuenciales. Spotlight la matrix de interaccinoes es convertida a un objeto de interacciones.

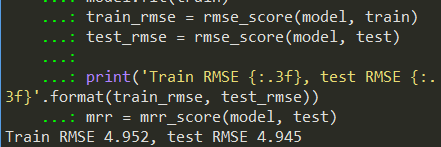
Las métricas como recall/precisión se dan en una tupla de arreglos numpy. La predicción es una valor asignado por todos los ítems existentes. Pero no te los acomoda dependiendo el valor.

Se evalúan los siguientes modelos para una base de datos de movieLens.

Factorización explícita.



Factorización implícita.

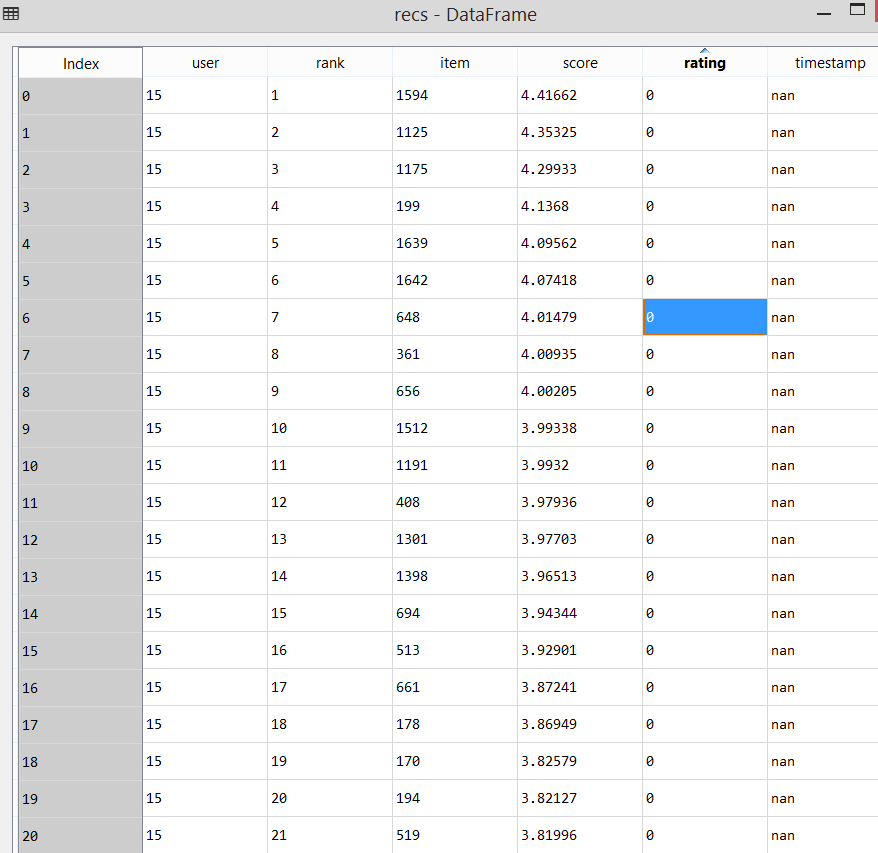
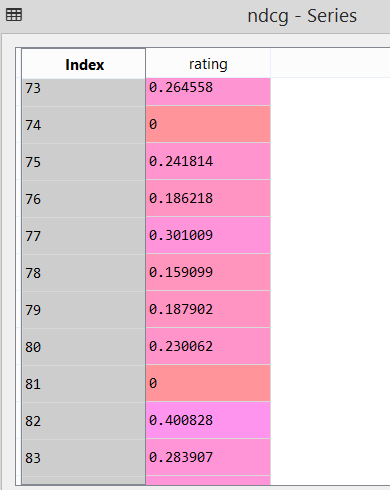


Tiene mejor resultados la factorización explicita. En un futuro checar CNN con modelos de secuencia.

**Lenskit.**

Pobre y poca documentación y ejemplos.

Un ejemplo de la predicción del rating. Y provee la métrica de Normalised Discounted Cumulative Gain,

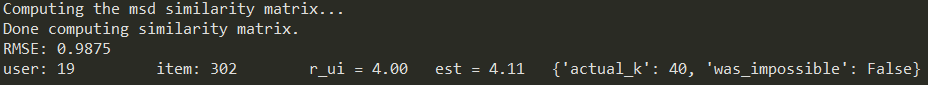
 

**Surprise.**

Poca documentación, es científica, como parte de la librería de SciKit Learn de Python para Machine Learning.

Se hacen dos pruebas con un SVD linear y un Knn simple.

* **SVD.** 
* **KNN**



Librería muy similar al seguimiento que se le da en SciKit Learn, tiene distintos métodos de similaridad, así como Singular Value Descomposition. Muy reciente 2017.

**Implicit.**

Librería sencilla de utilizar, para hacer predicciones la más amigable te muestra el número de elementos que quieres recomendar. Solo cuenta con 3 modelos: alterning least squares, Bayesian Personalised Ranking y Approximate Alternating Least Squares, todos ellos basados en la factorización de matrices. Se utilizó el modelo ALS, se obtuvieron los siguientes resultados:

* **Ranked ítems.**

Rankea los items para un usuario en una lista ordenada, los items son especificados.

Regresa, itemid, score.

* **Usuarios afines.**

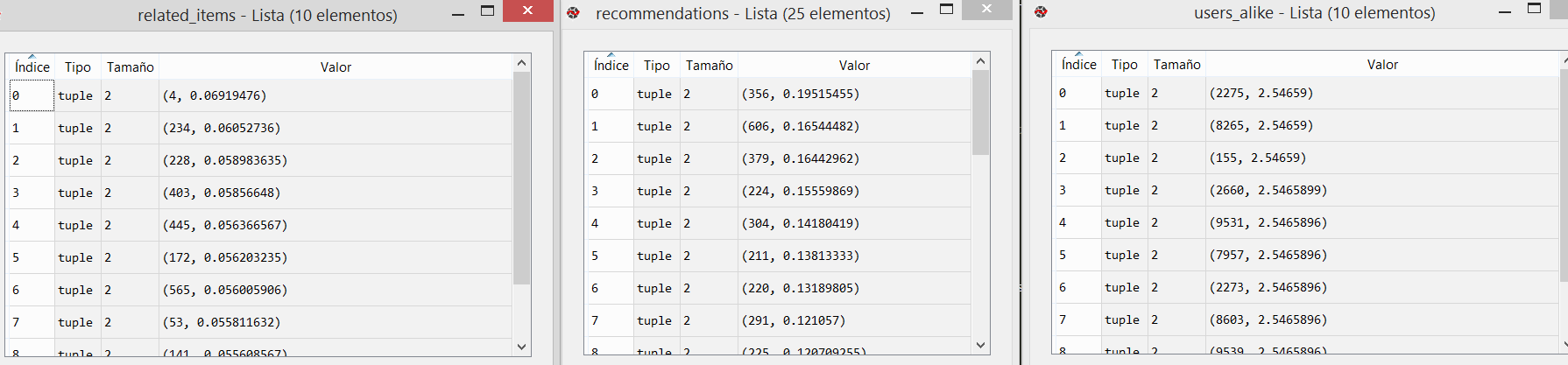
Los id´s de los usuarios más afines.

* **Items relacionados.**

Id´s de los items más parecidos.

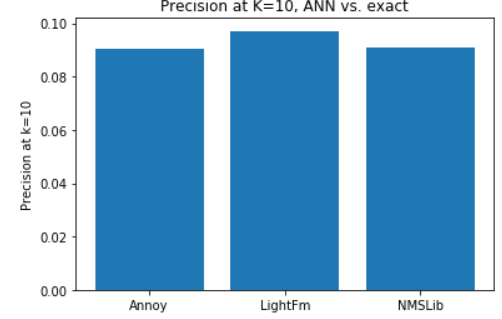
* **Recomendación.**

Recomendación de los n mejor items para un usuario.



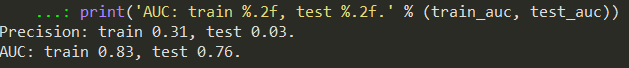
**Light FM.**

Es un modelo para sistemas de recomendación que se basa en un modelo hibrido de contenido-filtrado colaborativo, basado en la factorización de matriz que se desempeña de muhco mejor manera en las situaciones “cold start”, tomando en cuenta metadata y latent factors. Los latent factors son los users and items. La latent representations es dada por la suma de todos los features del sistema. Su paper muestra que tienen buena precisión.

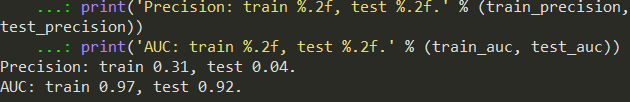


Se hacen dos pruebas con 2 modelos BPR “Bayesian Personalised Ranking” y Weighted approximate Rank pairwise.

Los resultados para **BPR** fueron:

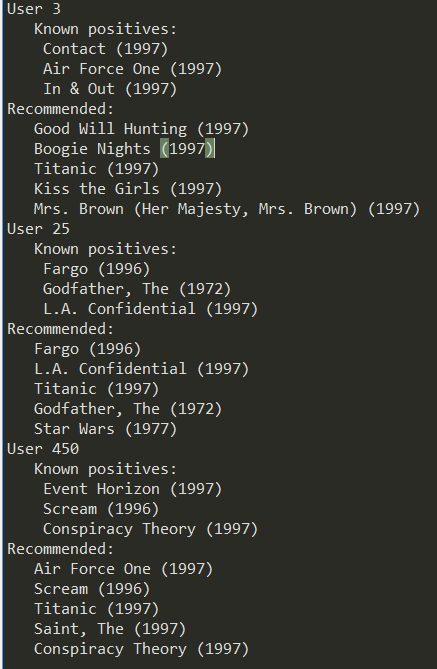


Por su parte los resultados para **Warp** fueron:



Se observa que los resultados fueron mejores para el modelo de Warp.

Se obtienen las recomendaciones para 3 usuarios, los resultados se muestran a continuación:

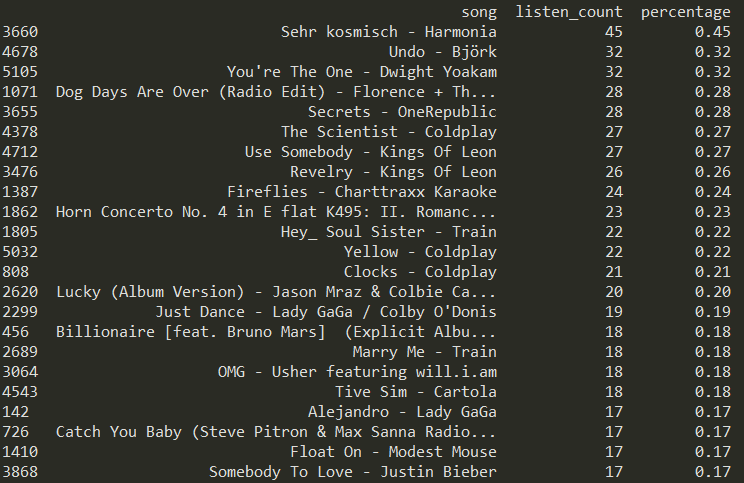


**Song recommender System.**

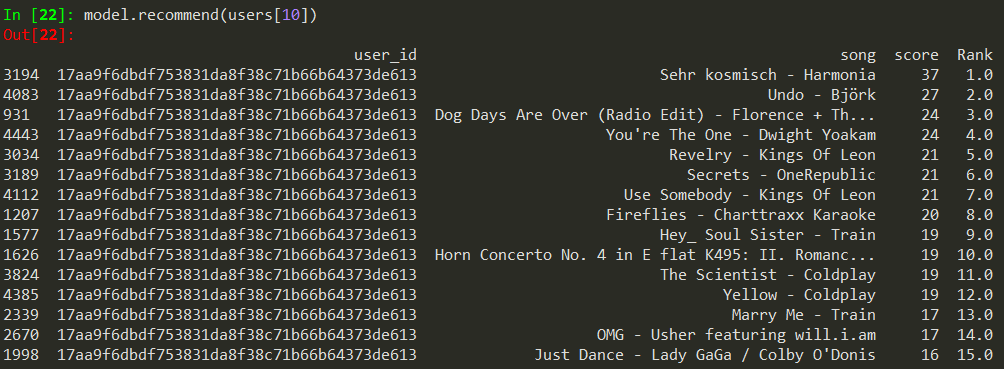
* **Recomendación basada en popularidad.**
* **Recomendación basada por canción.**
* **Recomendación basada por artista y situación geográfica.**

Como parte del approach, lo primero sería recomendar los más populares, esto sería tomando y agrupando los artistas y sumando todas sus reproducciones que tuvieron y obtener la frecuencia absoluta y relativa. A continuación se muestra el ejemplo.

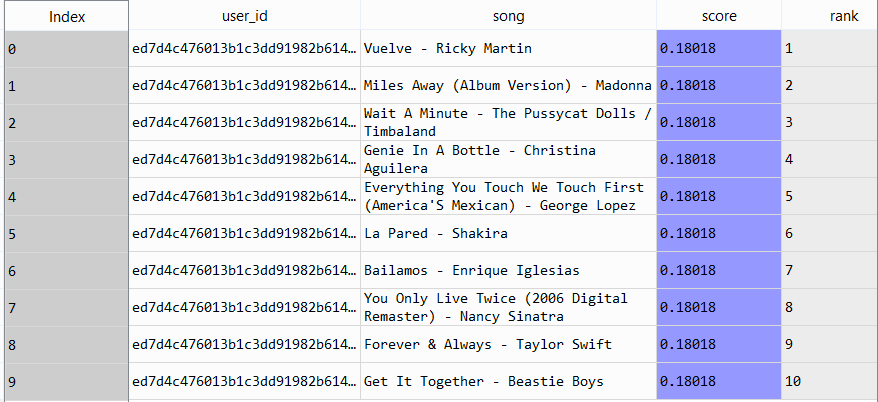
Para esta prueba se utilizó una dataset de 1millón de canciones.



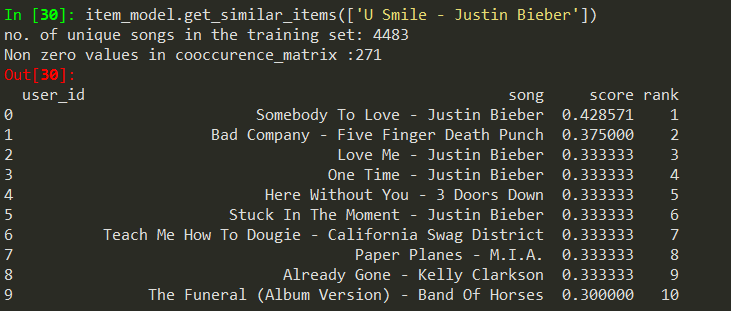
Observamos las recomendaciones para un usuario en específico. Estas recomendaciones serán las mismas, ya que es basado en la popularidad de los artistas.



El segundo approach utilizado es por filtro colaborativo basado en la similaridad de la escucha de canciones. Se basa en la construcción de una matriz de Co-ocurrencia. Para un usuario en específico. Se muestra a continuación:

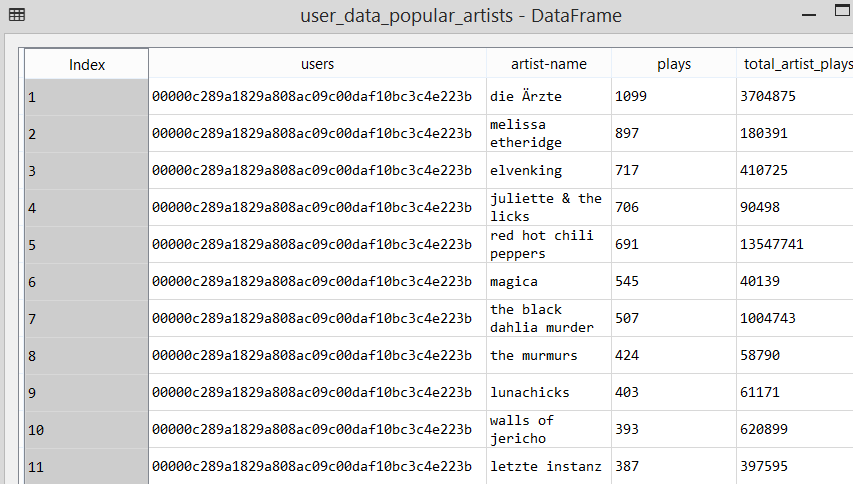


Ahora bien encontrar recomendaciones según una canción:



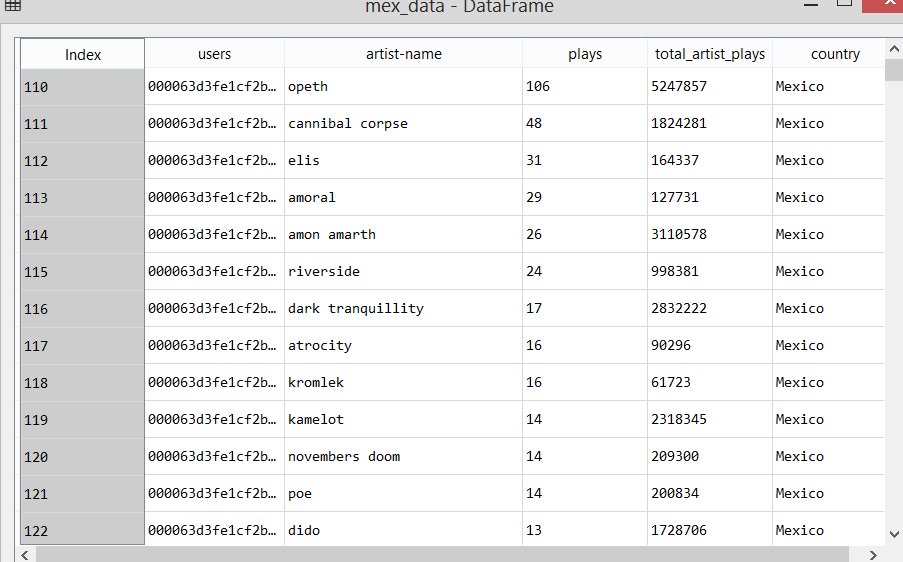
* El siguiente approach es **enfocado a los artistas** que escuchan los usuarios dependiendo su **situación geográfica**.

Cuando tenemos un dataset más grande, para la popularidad vemos la distribución de las reproducciones a un determinado número de artistas y elegir un threshold. En este caso tomé alrededor del 3.5% de los artistas totales, que juntos suman casi 900,000 reproducciones.

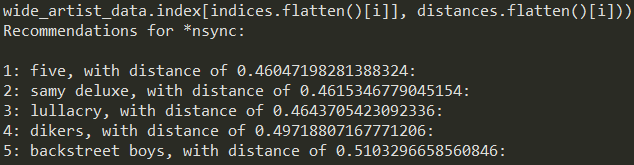


Se realiza un filtro colaborativo, utilizando la aproximación por medio de nearest neighbours bajo la métrica de similaridad del coseno.

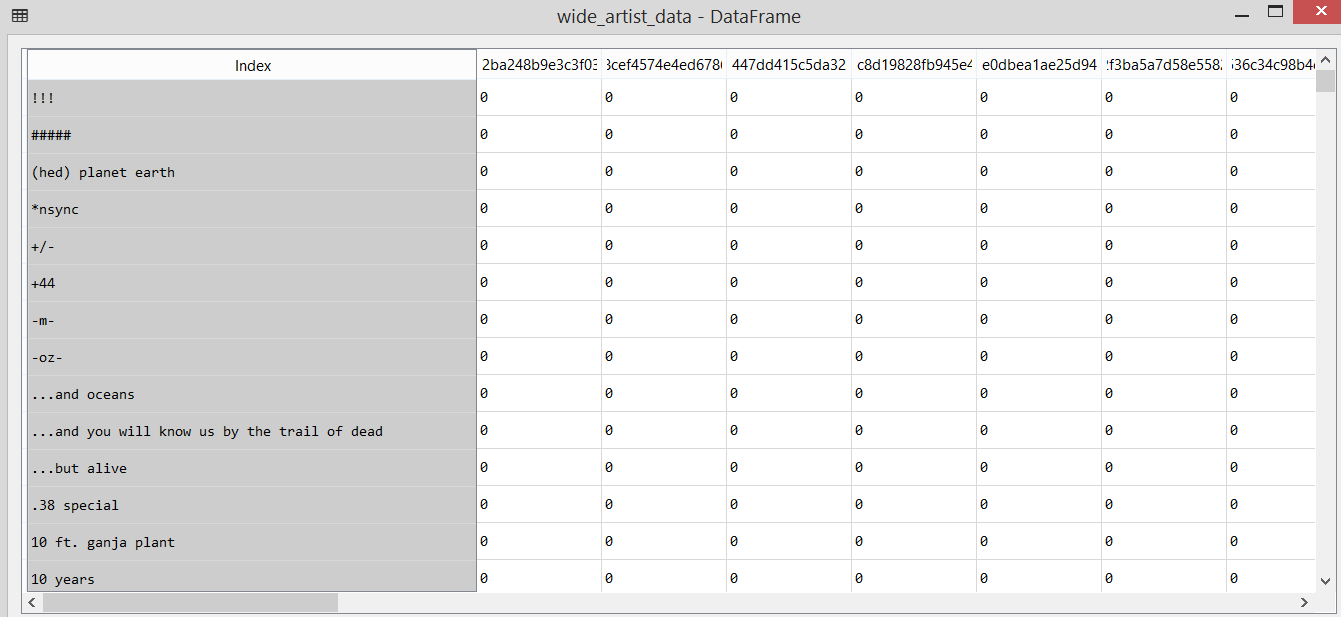
Seleccioné por área geográfica para facilitar el costo computacional. En este caso se eligió México.



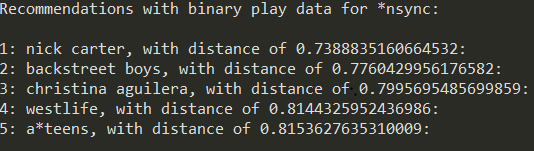
Hacemos la recomendación para el artista nsync:



Comparamos con otro approach en vez de KNN, transformando la matriz a artistas vs usuarios. Esta matriz contiene las veces que el usuario reprodujo canciones del artista.



Hacemos la recomendación:



Obervamos que éste approach tiene mejor similaridad que KNN.