Recomendador Anime

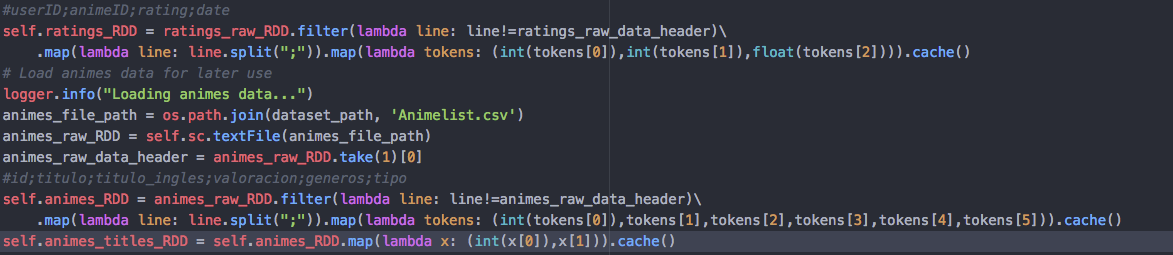
El objetivo de la tarea es hacer funcionar ese código (<https://github.com/jadianes/spark-movie-lens>) y luego modificarlo para que funcione con las valoraciones del dataste con valoraciones de series y películas de anime.

El proyecto consta de 3 ficheros, y 2 dataset ( se encuentran dentro de la carpeta anime).

Los dataste sobre los que trabaja el modelo para entrenar son Anime\_ratings.csv contiene un listado (124774 entradas) de valoraciones de los animes (userID;animeID;rating;date), el segundo dataset contiene un listado (15073 entradas) de animes (id;titulo;titulo\_ingles;valoracion;generos;tipo)

Fichero 1 engine.py contiene las clase RecommendationEngine dicha clase tiene definidos una serie de métodos los cuales trabajaran para realizar las diferentes tareas del recomendado.

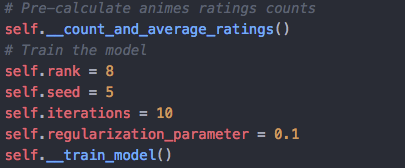
En el constructor de la clase encontramos la extracción de los dos dataset para transformarlos en un objeto RDD de Spark. El dataset Animelist.csv será extraído en la variable global self.anime\_RDD (id;titulo;titulo\_ingles;valoracion;generos;tipo)

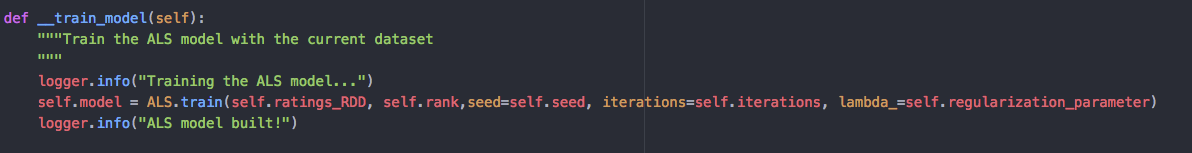


El dataset Anime\_ratings.csv será extraído en la variable global self.rating\_RDD (userID;animeID;rating;date)

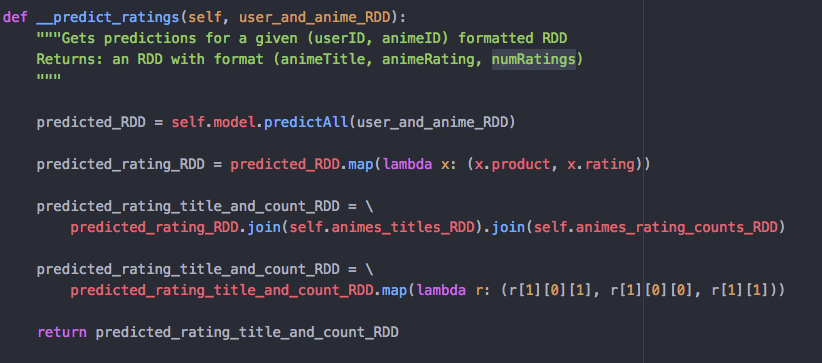


En el constructor también entrenamos el modelo con el self.rating\_RDD



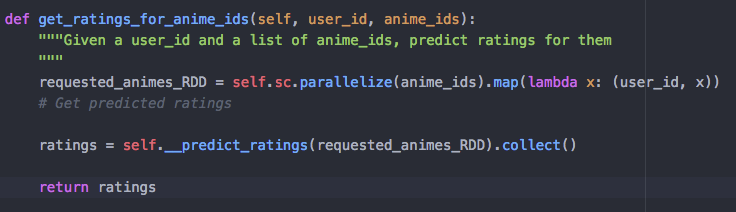


Una vez entrenado el modelo creamos la función \_\_predict\_ratings(dataset\_usrID\_animeID) la cual recibe pasarle un dataset del tipo (userID, animeID) para que nos prediga la valoración del usuario para dicho anime. El resultado de la función el una tupla de (userId,rating,nValoraciones)

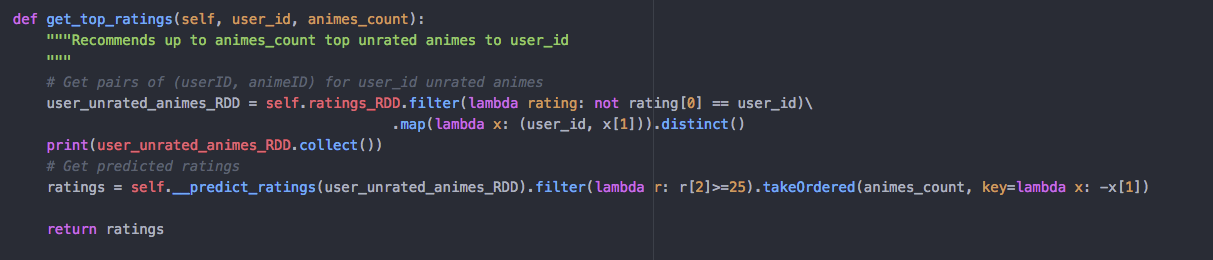


Una vez que disponemos del modelo y una funciones que nos prediga podemos empezar a jugar con los dataset de entrada para poder predecir.

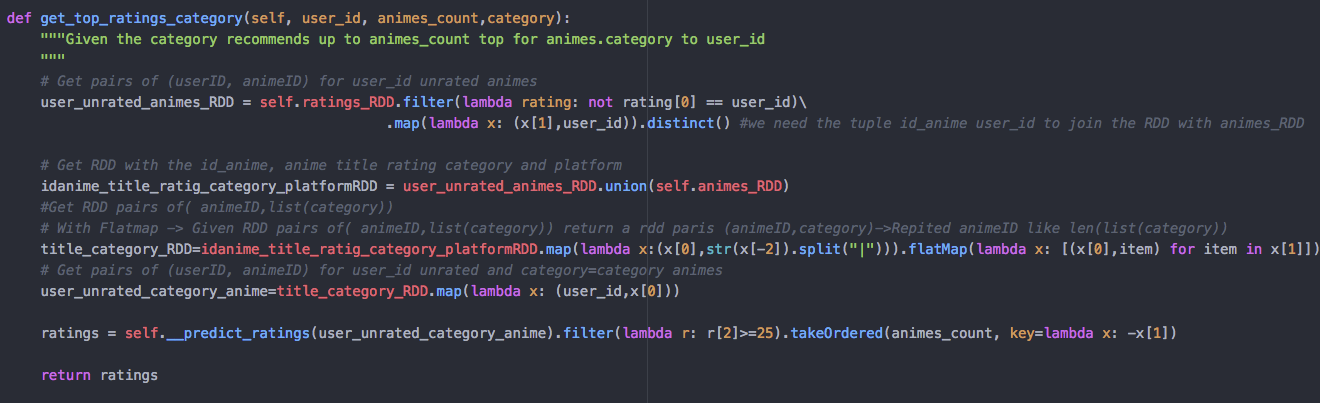
La primera función get\_ratings\_for\_anime\_ids(usr\_id,anime\_ids) devolverá la predicción del modelo creado para el usuario introducido y el anime introducido



La segunda función get\_top\_ratings(usr\_id,anime\_count) devolver los animes que el usuario introducido no haya valorado (supondremos que no la abra visto) que nuestro modelo prediga que serían los mas valorados para un usuario introducido



La tercer función get\_top\_ratings\_category(user\_id, anime\_count,category) devolver los animes que el usuario introducido no haya valorado (supondremos que no la abra visto) y que pertenezcan a la categoría seleccionada, que nuestro modelo prediga que serían los mas valorados para un usuario y categoría introducido



Al recibir las categorías separada por | debemos realizar un map para poder extraer todas las categorías del anime. Para poder extraer las categorías utilizamos la función flatmap la cual a partir del un RDD(idanime,(list->ncategorias) de n filas obtenemos un RDD de mayor numero de filas. Con eso conseguimos un RDD del tipo (idanime, categoría), realizamos un filtrado por categoría seleccionada disponiendo de un RDD con el id del anime que el usuario no a valorado y pertenezca a la categoría seleccionada.

La predicción se realiza igual que en las funciones anteriores.

Función cuatro get\_top\_ratings\_tips(user\_id,anime\_count,tipe) devolver los animes que el usuario introducido no haya valorado (supondremos que no la abra visto) y que pertenezcan a la tipo introducido ('TV', 'Movie', 'Special', 'ONA','Unknown','Music' )

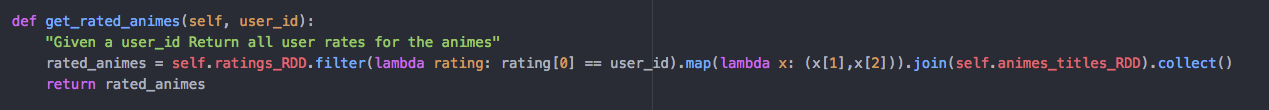


RDD\_tipe\_predecir=idanime\_tipe=idanime\_title\_ratig\_category\_platformRDD.map(lambda x: (x[0],x[-1])).filter(lambda x: x[1]==tipe) obtenemos un RDD con la tupla idanime, tipo, filtramos por el tipo introducido.

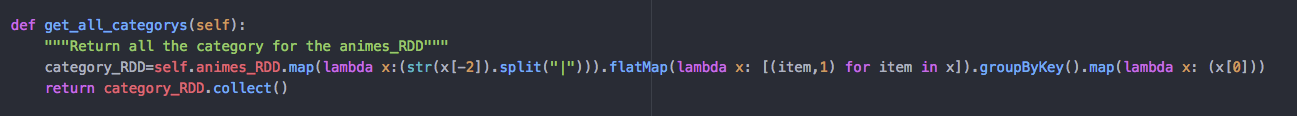
Realizamos la predicción del RDD filtrado siguiendo los mismos pasos de los ejemplos anteriores.

Ademas de las 4 funciones anteriores que predicen, realice 2 funciones para poder visualizar los datos de los ficheros:

Funcion get\_rated\_anime(usrid) devuelve todas las valoraciones del usuario introducido



Funcion get\_all\_categorys() devuelve un listado de todas las categorías que encontramos en el dataset



El fichero app.py es el encargado de interactuar con el usuario a partir de la dirección URL introducida por el usuario en su explorador de internet podrás visualizar el contenido en formato JSON. (APPI)

Las URL que puede consultar son:

* /<int:user\_id>/category/<int:anime\_count>/<string:category>
  + Llamara a la función get\_top\_ratings\_category(user\_id, anime\_count,category)
* /infocategory
  + Devuelve un listado de todas las categorías que encontramos en el dataset
* /info/<int:anime\_id>
  + Devuelve la puntuación media para dicho anime
* /<int:user\_id>/ratings/top/<int:count>
  + Devolver los animes que el usuario introducido no haya valorado (supondremos que no la abra visto) que nuestro modelo prediga que serían los mas valorados para un usuario introducido
* /<int:user\_id>/rated
  + Devuelve historial de valoraciones del usuario introducido
* /<int:user\_id>/tipe/<int:anime\_count>/<string:tipe>
  + Devolver los animes que el usuario introducido no haya valorado (supondremos que no la abra visto) y que pertenezcan a la categoría seleccionada, que nuestro modelo prediga que serían los mas valorados para un usuario y categoría introducido
* /infotipe
  + Devolvera un listado de los tipos que encontramos en el dataset

Por ultimo el fichero server.py es el encargado de levantar el servidor de localhost para poder realizar las consultas de la appi anteriormente descritas.

En dicho fichero podemos configurar el puerto y la dirección donde queremos que se levante el servicio en mi caso seria la dirección 127.0.0.1 (LocalHost) y el puerto utilizado el 8100

