Informe de la cita, día 2017.11.16:

En la cita, presenté las respuestas a los cinco problemas que se plantearon en la cita anterior, ha sido una conversación larga, ya que había bastantes problemas que comentar, a medida que avancemos la conversación, se nos ocurrieron nuevas ideas. También se han observado cosas que la tutora no esperaba que hiciera, entre ellas se destacó el algoritmo de recomendación, eso fue, en vez de utilizar la ponderación de los valores Pearson, me inventé otro algoritmo en que se ponderaban los rankings para sacar la mediana, se comentará abajo.

En resumen, hemos hablado principalmente de los aspectos relacionados con el tratamiento de los **missing values** (abrev: **MV**, valores de tipo NaN), que son los siguientes:

1. El recommender implementado (RecUserBased) no puede tratar los MV, hay que mejorarlo de manera que sí lo pueda (evitar la computación cuando se encuantra un NaN), así sería más completo. Y hay que reducir el número de aparición de NaNs al máximo en el resultado de predicción.
2. Otra solución sería eliminar los MV antes de empezar el proceso de recomendación.
3. En esta implementación (la presentada en la cita), hemos asumido que los valores de tipo NaN tienen pésimos rankings, y se convierten en los valores de tipo Integer. Pero esta asunción puede ser muy arriesgada, porque en primer lugar, el NaN es muy delicado, y representa que una asignatura no está matriculada, las causas pueden ser: el alumno no quiere matricularse en ese año, el alumno la matriculó y lo ha anulado, el alumno la tiene convalidada. Por tanto un NaN no tiene por qué ser pésimo valor de ranking; en segundo lugar, cuando un alumno lleva varias asignaturas no matriculadas en un año (ya sea primer año o segundo, X e y), como todas tienen valor NaN, el ranking aplicado a ellas no tiene sentido. Sin embargo, no deja de ser considerada como una opción más, podemos sacar su resultado y comparar con los de demás opciones.
4. Por último, propuse reemplazar los MV mediante la recomendación, eso es, antes de empezar el proceso de recomendación (predicción). La tutora lo apreció porque es realmente bueno, que en el fondo, es una asunción de los MV razonable.

Arriba son los 4 tratamientos de MV que se pueden distinguir.

Problemas del **Random Forest (RF):**

Como un primer intento, antes del recommender, se utilizaba el Random Forest para la predicción. Concretamente, se utilizaban RandomForestRegressor (predicción mediante calificaciones) y RandomForestClassifier (predicción mediante

ranking) del paquete sklearn.ensemble. Como estas dos clases son de terceros, al usarlas, nos enfrentamos con algunas restricciones:

1. Requieren que el dataset (X, y) no contenga missing values (NaNs).
2. Es probable que predicen las labels (etiquetas de clase, o y's) de forma independiente.

El algoritmo de **ponderación de medianas del ranking (PMR):**

1. El funcionamiento: después de obtener los KNN basados en MAE (Mean Absolute Error), se calcula la **puntuación de precisión con margen (PPM)** entre el alumno en testeo y cada uno de los KNN, de la siguiente manera: se saca la diferencia de cada una de las diez calificaciones (de asignaturas de primer año), si la diferencia es menor que la margen establecida, sería un acierto, así el número de aciertos marcará la PPM entre los dos alumnos. Esta precisión sirve de ponderación para las medianas del ranking, el pseudo código sería:
   Crear un vector vacío ->***median_list***
   Para cada asignatura del segundo año ->***actual_asig:***
      Crear un vector vacío ->***weighted_ranks***
      Para cada alumno de los KNN ->***sim_stu_y***:
         Ponderar el ranking del actual_asig mediante el PPM del sim_stu_y, y se concatena el resultado con el weighted_ranks. Por ejemplo: siendo el ranking=5, PPM=3, el resultado sería [5]*3 = [5,5,5]

      Calcular la mediana del weighted_ranks y guardar la tupla de (actual_asig, median) en el median_list.

   Ordenar el median_list según el valor de la mediana, el orden resultante sería el ranking de la predicción para el alumno en testeo->***prediction***

   Nota: el sim_stu_y representa la fila de calificaciones del segundo año del alumno similar.

2. Problemas: en cuanto al algoritmo de predicción, no había usado lo de Sergi, porque me pareció demasiado complicada la computación de Pearson que él había implementado. La tutora advirtió computar la predicción del ranking mediante **ponderación de puntuaciones de similitud de Pearson (PPP)**, de versión original, que se debía hacer antes de uno que me he inventado, porque el Pearson es el standard.
   Como un primer intento, la evaluación del PMR con el tratamiento 3 de MV (asunción de los pésimos rankings) no ha sido muy buena; es más, el uso del PMR, en muchas ocasiones, no permite una clasificación estricta, ya que a

menudo podemos encontrar medianas con el mismo valor, de manera que, al ordenarlas, no tenemos un criterio para saber cuál va antes.

Tablas de evaluación como el resultado de experimentos:

Para la implementación de la funcionalidad de "predicción del ranking", hemos usado diferentes modelos de predicción (Random Forest y Recommender), que serán evaluados con diferentes medidas de validación (Accuracy, MAE, etc), y hemos tenido en cuenta dos aproximaciones:

1. Ranking mediante predicción de notas: notas (input) -> **predicción** -> notas (predichas) -> conversión -> ranking (resultado final)
2. Predicción de ranking: notas (input) -> conversión -> ranking (inicial) -> **predicción** -> ranking (resultado final)

Además, tenemos pensados dos algoritmos a la hora de computar la predicción: PMR y PPP, aunque muy probablemente vamos a dejar el PMR por los problemas comentados anteriormente.

Necesitamos englobar todas estas ideas en una tabla, de modo que la tabla muestre la conclusión de diferentes experimentos.

Tabla de evaluaciones (Predicción mediante el ranking)

Métodos\Medidas		Accuracy	MAE	medida3	...
Random Forest					
Recommender: RecUserBased (diferentes tratamientos de NaN)	*1. mantener*				
	2. eliminar				
	3. reemplazar				
	4. Asumir pésimo				
Método 3					
...					

Table 1: Tabla de evaluaciones, predicción mediante el ranking

Tabla de evaluaciones (Predicción mediante calificaciones)

Métodos\Medidas		Accuracy	MAE	medida3	...
Random Forest					
Recommender: RecUserBased (diferentes tratamientos de NaN)	*1. mantener*				
	2. eliminar				
	3. reemplazar				
Método 3					
...					

Table 2: Tabla de evaluaciones, predicción mediante calificaciones

Notas:

1. Las medidas de validación se calculan mediante el 10 Fold Cross Validation.
2. De momento dejamos el PMR en un lado, y nos enfocamos en el PPP.
3. Resumen de los 4 casos del missing value:
    a. Mantener los NaNs en el dataframe inicial
    b. Eliminar los NANs del dataframe inicial
    c. Reemplazo de NaNs vía recomendación en el dataframe inicial
    d. Asumir que los NaNs tienen pésimos rankings en el dataframe inicial
4. De momento probamos solamente el caso “b” del missing value para el Random Forest.

Tareas para la siguiente cita:

1. Implementar el algoritmo PPP
2. Implementar los tres primeros puntos del recommender
3. Estudiar el RandomForestRegressor y RandomForestClassifier
4. Rellenar las dos tablas