Modelado datos de MovieLens

Se tienen 20,000,263 registros de los ratings dados por 138,493 usuarios a 26,744 películas desde 1995 hasta 2015.

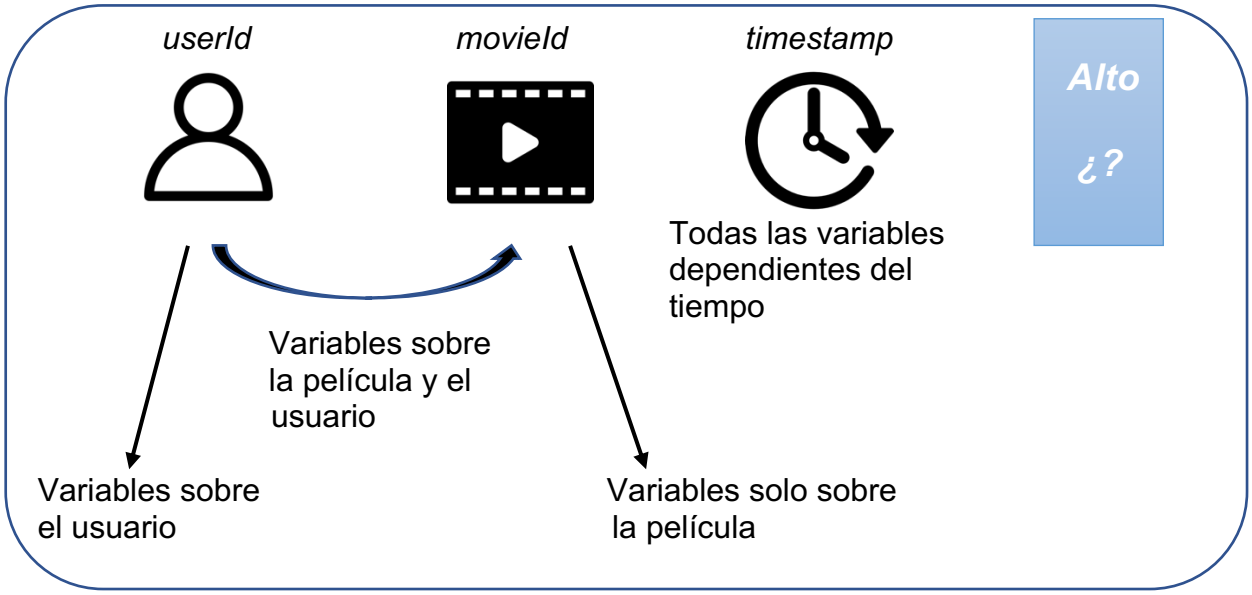
El objetivo es construir un modelo para predecir si un usuario va a calificar una película con “alto” o no, con:

- 1 o “alto” si rating >= 4
- 0 si rating < 4

Para cada película se cuenta con su género y año de estreno, y para 10,381 películas se conoce la relevancia de 1,128 tags.

Feature Engineering

Para entender la construcción de las variables observemos el siguiente esquema de planteamiento:



De modo que se construyeron variables desde 4 enfoques, que se detallan a continuacion.

Variables set 1: Del usuario hacia la película por género

En este conjunto se construyeron variables que indican el comportamiento del usuario respecto a las películas, como su predilección de género. Así, se tienen para cada género (19 géneros) y cada ventana de tiempo (7, 30, 90, 365 y 7300 días (toda la historia):

- *género_ventana*_tot: Número de películas que el usuario ha rankeado de ese género en cada ventana .
- *género_ventana*_rat_prom: Promedio del rating que ha dado el usuario a películas de ese género para cada ventana de tiempo.
- *género_ventana*_porc: Número de películas que el usuario ha rankeado de ese género entre el número total de películas que el usuario ha rankeado por cada ventana.

Además, para cada variable solo se consideró la información que se tiene hasta un día antes de la fecha de rankeo, esto debido a dos razones:

- Para evitar data leakage
- Pensando en un modelo en producción con ejecución diaria y carga de información diaria, lo máximo que se tendría de desfase de información sería un día.

Consideración 1

Variables set 2: Sobre el usuario (sin importar la película)

En este conjunto se construyeron variables que indican el grado de actividad del usuario, para conocer por ejemplo su grado de actividad:

- Antigüedad del usuario: tiempo desde que rankeó por primera vez.
- num_movies_*ventana*: Número de películas que ha rankeado el usuario para cada ventana de tiempo.
- porc_*ventana*: Número de películas que ha rankeado el usuario en cada ventana de tiempo entre el total de películas que ha rankeado.

Además se tomó la consideración 1.

Variables set 3: Sobre la película (sin importar el usuario)

En este conjunto se construyeron variables que expresaran la popularidad de la película, sin considerar el usuario en partícular.

- movie_tot_*ventana*: Número de usuarios que han visto la película para distintas ventanas de tiempo.
- movie_prom_rat_*ventana*: Promedio de calificación de la película respecto a todos los usuarios que la han visto en distintas ventanas de tiempo.
- Variables indicadoras para indicar el género de la película (19 géneros)
- Antigüedad de la película al momento en que el usuario la rankeó

Además se tomó la consideración 1.

Variables set 4: Respecto a los tags de las películas

Para las películas taggeadas se ajustó un modelo de cluster k-means con las películas estrenadas hasta el año 2011, resultando en 9 clusters, este número de clusters se seleccionó mediante el método del codo. Para el resto de películas se

asignaron a su clúster más cercano. Se realizó de esta forma pensando en que en un modelo en producción la clusterización podría actualizarse quizá cada mes o incluso hasta cada 6 meses, y las películas estrenadas en ese inter de tiempo solo se asignarían a su clúster más cercano.
Así se construyeron las variables:

- *cluster_ventana*_tot: Número de películas que el usuario ha rankeado de ese cluster en cada ventana de tiempo.
- *cluster_ventana*_rat_prom: Promedio del rating que ha dado el usuario a películas de ese cluster para cada ventana de tiempo.
- *cluster_ventana*_porc: Número de películas que el usuario ha rankeado de ese cluster entre el número total de películas que ha rankeado para distintas ventavas de tiempo.

Además se tomó la consideración 1.

Modelado

Para realizar el modelado se tomaron los conjuntos de entrenamiento y prueba de la siguiente forma:

- El conjunto de entrenamiento se tomó como una muestra de 4,000,000 de observaciones hasta el 2011 (fecha de rankeo), esto debido a las limitaciones de cómputo.
- El conjunto de prueba son todas las observaciones desde el 2012 (fecha de rankeo) hasta el 2015, con 2,177,490 millones de observaciones.

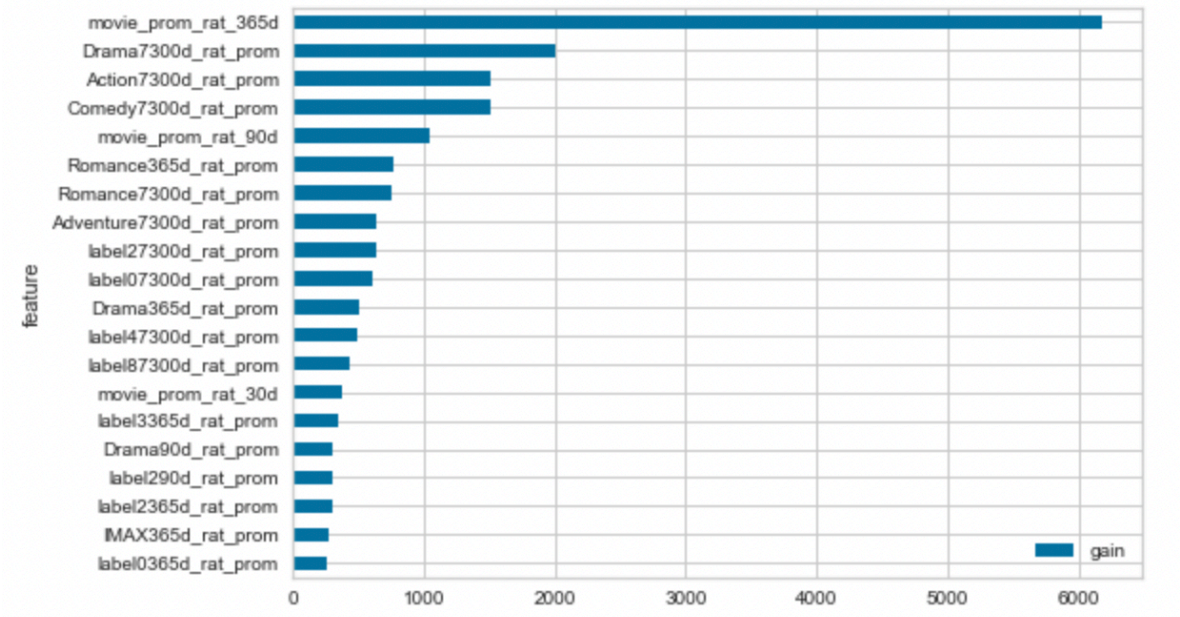
Se tomó de esta forma para analizar el performance del modelo en información futura.

Se trabajó con modelos XGBoost ya que en mi experiencia tienen un muy buen performance, además de que este modelo es bueno para tratar con datos sparse, y en este caso las features llegan a tener muchos missings.

El dataframe final contiene las variables userId, movieId, y (target), timestamp y 462 variables construidas, por lo que se buscó reducir el número de variables realizando lo siguiente:
Primero se obtuvo una muestra aleatoria para seleccionar solo 2 millones de registros de los 4 millones disponibles en el conjunto total de entrenamiento. Este conjunto resultante se dividió en entrenamiento y validación con el 70% y 30% de los datos respectivamente. Finalmente a este nuevo conjunto de entrenamiento se le ajustó un modelo XGBoost con *máxima profundidad* de 5 y *learning rate* de 0.3 (se consideró de esta forma ya que en mi experiencia estos son valores con buen desempeño). Así, se obtuvo en el conjunto de entrenamiento un AUC de 0.75669 y en el conjunto de validación un AUC de 0.75552.
Con el resultado de este modelo se seleccionaron las 150 variables con el mayor feature importance por *gain*. Se seleccionó esta métrica ya que así se puede saber

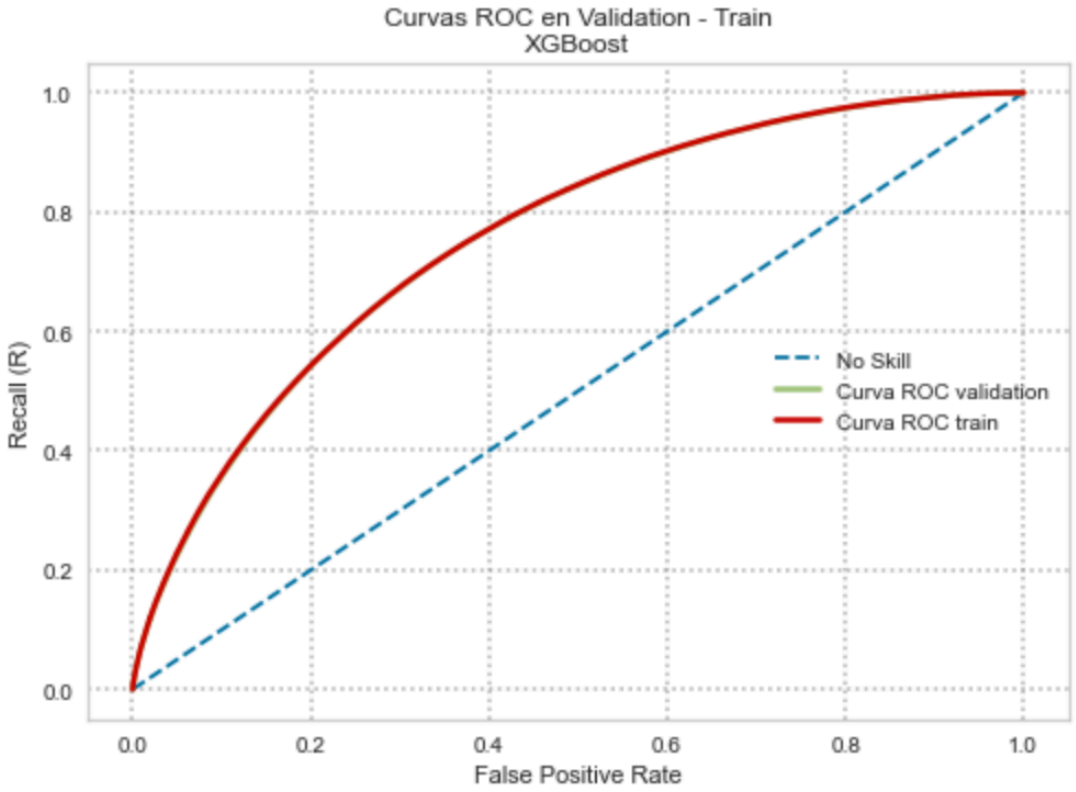
la información promedio que aporta cada variable en todos los splits en los que ésta es usada.

En la siguiente gráfica se muestra el gain de las 20 variables con mayor importancia. Aquí podemos observar que la variable que más aporta al modelo es el promedio general de la película en los últimos 365 días, podríamos interpretarlo como que el rating general de una película determina en gran medida la predilección del usuario. Las siguientes variables son respecto al usuario, a su predilección por cada género o grupo de películas (cluster). Luego también, se tiene variables que indican la antigüedad de la película o la del usuario.



Con las 150 mejores variables seleccionadas ya se prosiguió a trabajar con el tamaño total de muestra (4 millones).
Cabe mencionar lo siguiente: a los 4 millones de datos se les ajustó un modelo XGBoost haciendo una optimización de los hiperparámetros *profundidad máxima* y *learning rate* y el resto de parámetros por default. La *profundidad máxima* se consideró entre 3 y 7 y el *learning rate* entre 0.1 y 0.9 con granularidad de 0.1. La optimización se realizó mediante una Cross Validation Grid Search con 4 *folds*. Sin embargo, debido a la exhaustividad de la ejecución, por el tamaño de muestra, no se continuó.
Por esto, se probó con algunas configuraciones de hiperparámetros que por experiencia establecí: *profundidad máxima* de 5 y 6 y *learning rate* de 0.3 y 0.4. En general, realizar cross validation mejora marginalmente el AUC, así que lo que busqué con estos ejercicios fue no caer en overfitting, evaluar el desempeño y asegurar la estabilidad del modelo a través de tiempo.

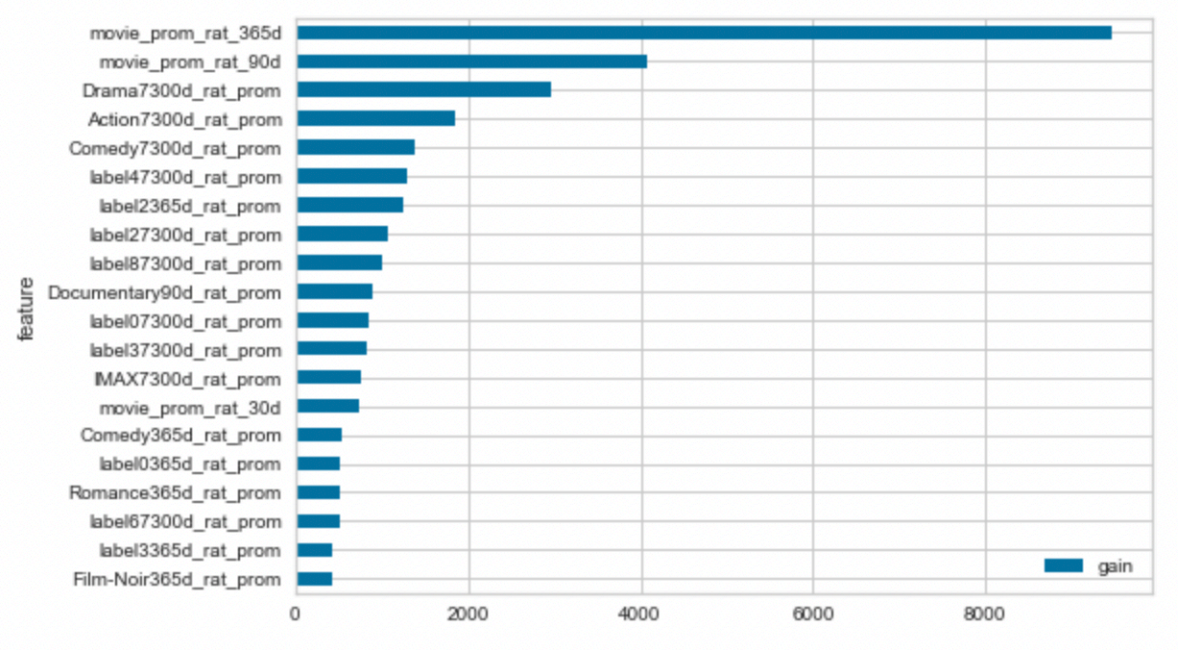
Para cada prueba de hiperparámetros el AUC resultó similar, por lo que se seleccionaron los que no producían un overfitting en el entranamiento respecto a la validación. A continuación se muestra la curva ROC y el AUC resultante.



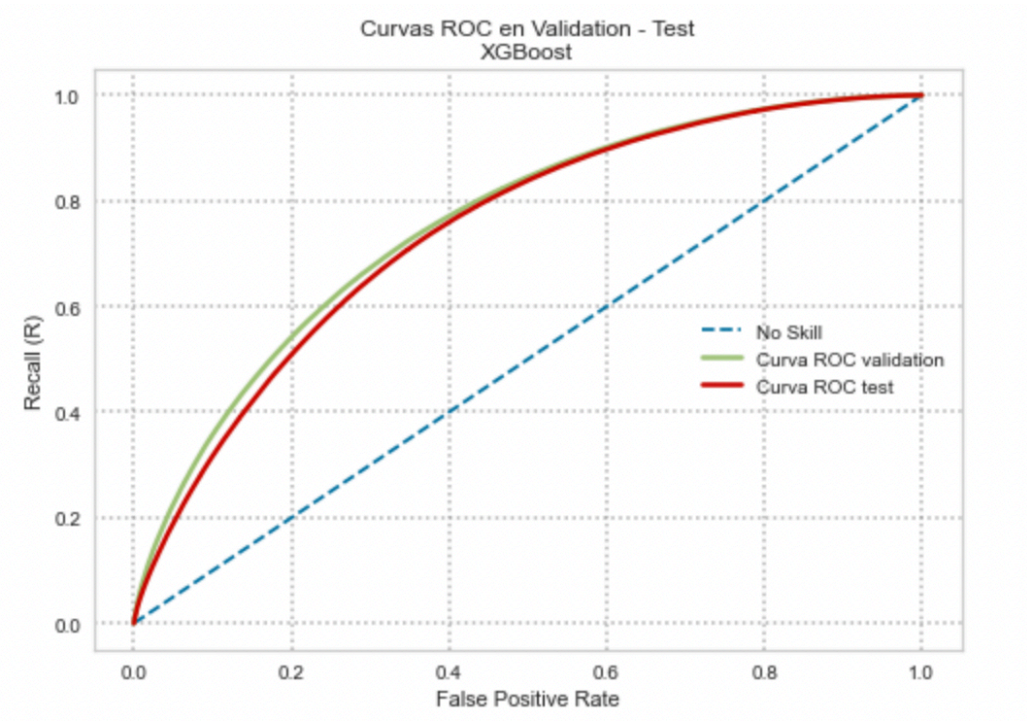

```
El área bajo la curva ROC de validation es 0.7555367562567538
El área bajo la curva ROC de train es 0.7566964948077614
```

Como se observa, el desempeño del modelo en el conjunto de entrenamiento es muy similar al desempeño en validación.

El feature importance se muestra en la siguiente gráfica, donde observamos, al igual que en el primer modelo para la selección de variables, las variables más importantes son sobre la popularidad de la película.

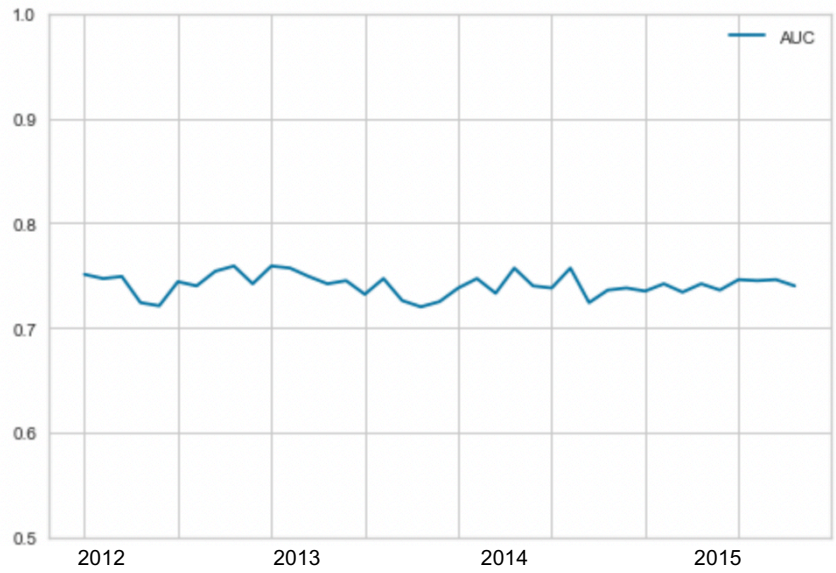


A continuación se muestra la curva ROC para el conjunto de prueba comparada con el conjunto de validación, en donde se observa que el desempeño del modelo en el test es muy similar al de la validación, lo que también se puede notar en el AUC que pasó de 0.755 a 0.742




```
El área bajo la curva ROC de validation es 0.7555367562567538
El área bajo la curva ROC de test es 0.7422840843879459
```

Además, en la siguiente gráfica se observa la evolución del AUC para cada mes de 2012 a 2015.



Como podemos observar el AUC es estable a través del tiempo.

Punto de corte

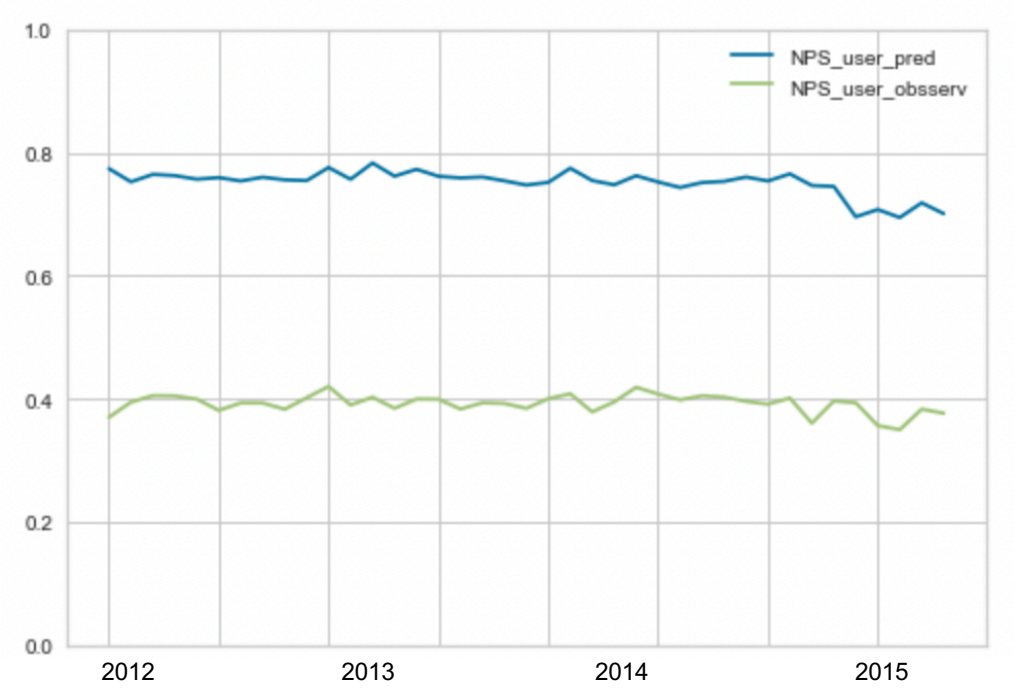
Para establecer un punto de corte de la probabilidad estimada de que *y* sea 1 se estableció una medida de satisfacción del servicio desde una perspectiva Customer Centric. Suponiendo que a través de este modelo se realizan recomendaciones de películas a los usuarios se puede saber, dado que se recomienda una película a un usuario, cuál es su grado de satisfacción basados en el rating que asigna a ésta. Así, se consideró un análogo de la medida NPS (Net Promotor Score) por usuario de la siguiente forma:

Si el rating asignado es de 0.5 a 2.5 se considera como un rankeo detractor (DET), muy probablemente el usuario tuvo una mala experiencia. Si el rating es de 3 a 3.5 se considera como un rankeo pasivo, pues si bien no fue una buena recomendación, su grado de insatisfacción no es alto. Por último, si el rating es de 4 a 5 se considera como promotor (PROM), lo cual se traduce de una muy buena experiencia.

Así, el objetivo de maximizar la experiencia de uso general de los cliente, dado que se le recomendó una película, se traduce en encontrar el punto de corte tal que se maximice el promedio del NPS por usuario:
NPS_{user} = %PROM - %DET, con la condición de recomendar al menos una película al 90% de los usuarios.

El promedio del NPS_{user} observado en todo el conjunto de entrenamiento es de **0.399**. Y realizando la optimización planteada para el conjunto de entrenamiento se obtuvo un **punto de corte 0.677** y un **NPS_{user} promedio de 0.7211**, es decir, con el modelo desarrollado se aumentó este KPI en 33 puntos porcentuales.

Para el conjunto de test el promedio del **NPS_{user} observado es de 0.438**, mientras que el que se obtiene con el punto de corte óptimo es de **0.683**. Es decir, en los datos de prueba la mejora en este KPI es de 25 puntos porcentuales. Más aún, en la siguiente gráfica se observa el promedio del NPS_{user} que se obtiene con el modelo y el observado para cada mes:



Con esta gráfica podemos observar que el promedio del NPS_{user} que se obtiene con el modelo es robusto a través de tiempo, y así mismo la ganancia en puntos porcentuales comparado con el KPI que se obtiene antes de la aplicación del modelo.

Conclusiones

Se puede concluir que el performance del modelo ajustado proporciona una mejora significativa en el promedio del NPS_{user}, es decir en el grado de satisfacción general de los usuarios. Además de que el modelo resulta ser robusto a través del tiempo.

Este modelo es un buen punto de partida, sin embargo, puede ser mejorado con la construcción de otro tipo de variables. Por ejemplo, con la tabla de *tags* que hace cada usuario se podría construir el Promedio del rating que ha dado el usuario a películas con el mismo tag. Incluso con la tabla de *links* se podría explorar la

información que arrojan las APIs, información como cast de la película, director, duración, presupuesto, entre otras puede aportar mucho al modelado.

También restaría analizar el impacto que tiene el tiempo de desfase de información, en este caso se consideró un día de desfase, pero sería interesante revisar qué pasa si se considera más tiempo. Por otro lado, podrían ajustarse otro tipo de modelos como Random Forest o Redes Neuronales y comparar su desempeño.

Por último, para establecerse un punto de corte podría haber otra métrica en la que el negocio esté interesado, por lo que teniendo este conocimiento se podría realizar otro enfoque para realizar esta optimización.

Se anexan las notebooks de trabajo:

1. Kueski_Feature_Engineering
2. Kueski_Modelado