Implementation of a Recommender System for Crowdfunding

Udacity Capstone Project - Machine Learning Nanodegree

Carlos Roso

1. Definition

Project Overview

This project aims to build a recommendation system for the most important Colombian crowdfunding site. The main goal is to analyze user behavior and build individual recommendations which consists of a set of campaigns they are most likely to fund.

The crowdfunding site is called Vaki and has been actively funding any kind of social cause across many domains (culture, politics, news, etc). The input data has been provided by the company which contains all the information about campaigns (~5k) and payments (~99k).

Problem Statement

The problem can be summarized as finding a set of campaigns that every individual user is most likely to fund next given their past funding behavior (both individual and collective). In order to solve this, two datasets have been provided which contain information about campaigns and payment transactions.

The strategy to solve this problem consists of building and testing 3 different recommendation models, comparing their performance and finally trying a hybrid model which combines all of them. The models that will be built are Popularity Based, Content Based, and Collaborative Filtering Recommendation Model. Each model will be thoroughly described in the Algorithms and Techniques section.

Based on existing literature[1], it's expected for Content Based Recommendation Systems to be less performant than Collaborative Filtering. This is because the former only considers the user's preference to recommend an item whereas the latter analyzes the relationships

[1] Google Developers. Recommendation Systems. Collaborative Filtering.
https://developers.google.com/machine-learning/recommendation/collaborative/basics?hl=en

between all the users and campaigns to make a more *intelligent* recommendation. Human behaviour plays a big role here too: even when humans have a well defined decision criteria, we will not always seek the same type of content (e.g. in food, movies, cars). There's always a big chance people will try something different if someone else with similar behaviour has done it before.

Finally, it's naturally expected that an ensemble system, which combines the most important models, will improve recommendations. Such a system is called a Hybrid Recommendation system and will also be implemented in this project.

2. Analysis

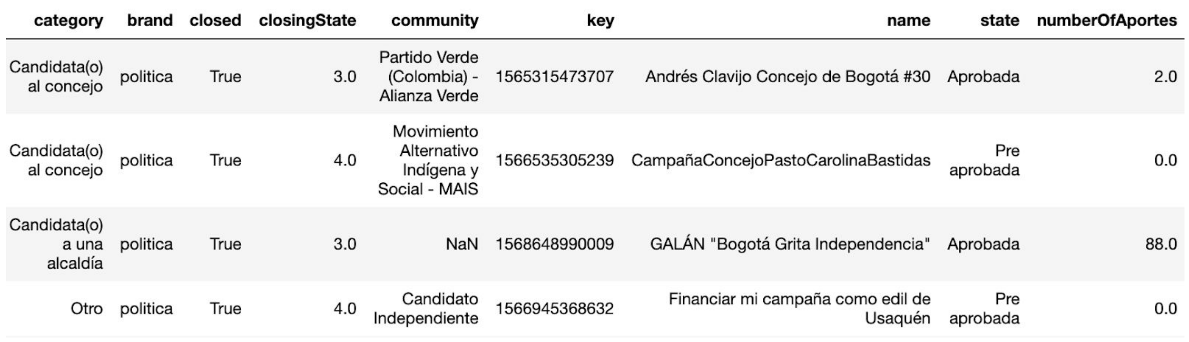
Data Exploration and Exploratory Visualization

All the data has been provided by the company in the form of a Firebase access with read-only permission to hundreds of thousands of documents. Therefore, the first step in data exploration involved some JavaScript scripting leveraging the Firebase SDK and custom pagination to download all the data. After this, the collections, alongside with its documents, were merged into the corresponding CSV files.

The data exploration will be divided into two parts: campaigns and payments. Each one corresponds to an individual dataset.

Campaigns

This dataset exposes information for both active and closed campaigns. Each campaign contains a description, tag, title, subheadline, categories, among others. It's important to note that the majority of the categories contain only some of these features. All texts are in spanish except for a handful of campaigns that do contain english descriptions. Descriptions are represented with HTML rich data.



category	brand	closed	closingState	community	key	name	state	numberOfAportes
Candidata(o) al concejo	politica	True	3.0	Partido Verde (Colombia) - Alianza Verde	1565315473707	Andrés Clavijo Concejo de Bogotá #30	Aprobada	2.0
Candidata(o) al concejo	politica	True	4.0	Movimiento Alternativo Indígena y Social - MAIS	1566535305239	CampañaConcejoPastoCarolinaBastidas	Pre aprobada	0.0
Candidata(o) a una alcaldía	politica	True	3.0	NaN	1568648990009	GALÁN "Bogotá Grita Independencia"	Aprobada	88.0
Otro	politica	True	4.0	Candidato Independiente	1566945368632	Financiar mi campaña como edil de Usaquén	Pre aprobada	0.0

Image 1. Head of the raw campaigns DataFrame

As mentioned earlier, each campaign contains relevant data in different features that adds up to its full description. As shown below, some of the campaigns only contain full length descriptions, others only categories, and some of them only their headlines. This is something important to note as it will be the basis of a very critical data cleaning step further below.

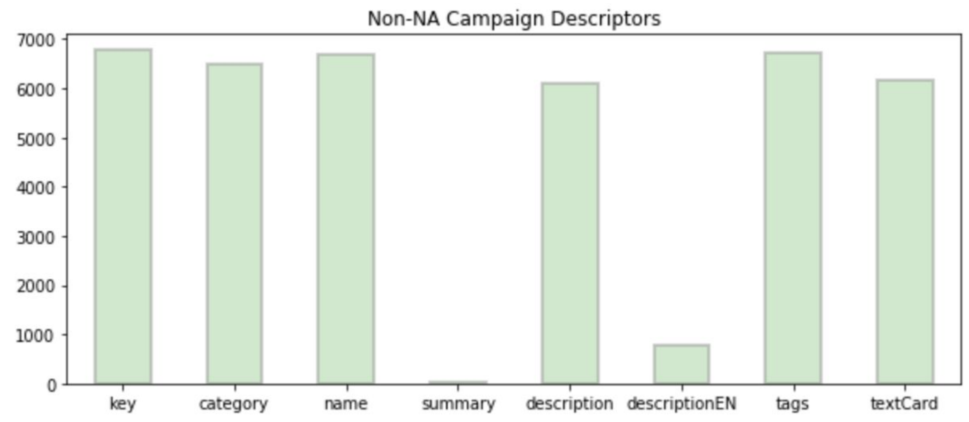


Image 2. Histogram Non-NA Campaign Descriptors for all campaigns.

It's also important to note that some users didn't fill in proper descriptions and left the placeholder text in that field. This fact will need to be taken into consideration for the data preprocessing step.

A key aspect of this dataset is that the majority of the campaigns are no longer active. This is naturally expected as the subset of currently active crowdfunding campaigns will always be much smaller than the whole set of historically available campaigns. For the purpose of this project, the campaign status will not be considered when deciding whether to recommend a campaign or not as it would otherwise imply training with much less data that's still relevant to the project. It would also be easy to achieve this filtering through scripting in the last step of the recommendation algorithm.

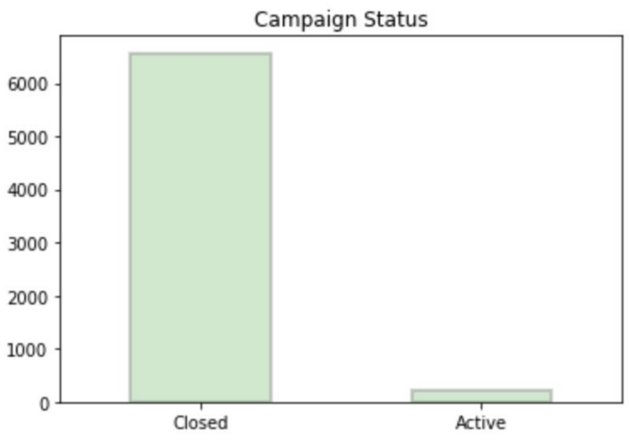
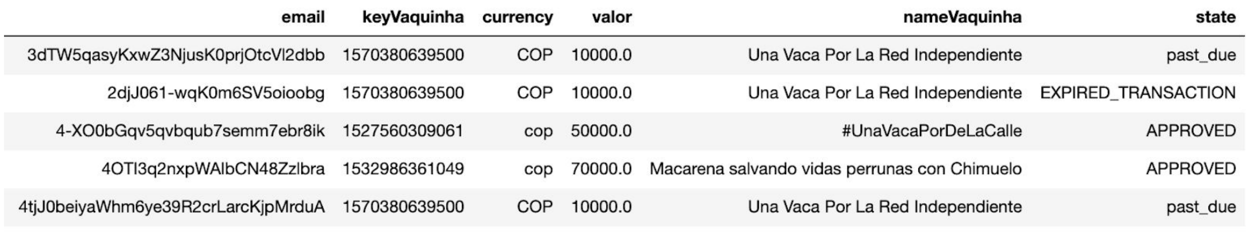


Image 3. Histogram Closed vs. Active campaigns.

Payments

This is the biggest dataset as it relates a user ID with a payment ID. User IDs have been hashed according to the company data disclosure policies. Each data instance represents a payment with the contributed amount, date, campaign name and payment status (success or failure).



email	keyVaquinha	currency	valor	nameVaquinha	state
3dTW5qasyKxwZ3NjusK0prjOtcVl2dbb	1570380639500	COP	10000.0	Una Vaca Por La Red Independiente	past_due
2djJ061-wqK0m6SV5oioobg	1570380639500	COP	10000.0	Una Vaca Por La Red Independiente	EXPIRED_TRANSACTION
4-XO0bGqv5qvbqub7semm7ebr8ik	1527560309061	cop	50000.0	#UnaVacaPorDeLaCalle	APPROVED
4OTl3q2nxpWAlbCN48Zzlbra	1532986361049	cop	70000.0	Macarena salvando vidas perrunas con Chimuelo	APPROVED
4tjJ0beiyaWhm6ye39R2crLarcKjpMrduA	1570380639500	COP	10000.0	Una Vaca Por La Red Independiente	past_due

Image 4. Head of the raw payments DataFrame

As shown below, it can be seen that roughly 40% of transactions correspond to unapproved payments. Despite this fact, it's been decided to keep all the transactions in place regardless of their payment status. This is because the recommendation engine will consider a payment intent as a strong enough signal to recommend a campaign in the future.

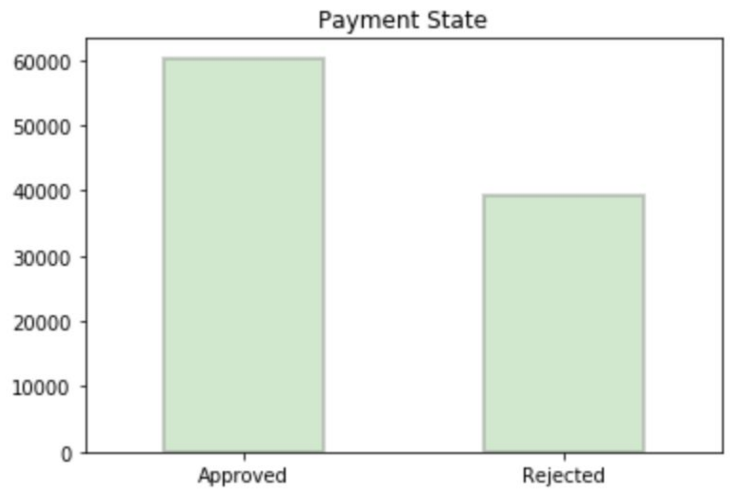


Image 5. Histogram Approved vs. Rejected campaigns.

Algorithms and Techniques

This project explores 3 recommendation models and a hybrid ensemble model which combines 2 of these into one. Below are the explanation and in-depth reviews of the algorithms and techniques used to build each one of these models:

Popularity Filtering

This is the most straightforward model in the proposed solution. It will generate candidates purely based on the number of active contributions for the currently active campaigns. The algorithm is straightforward as it will require to count and sort campaigns based on the number of associated transactions. Then, for each user, the model will grab the top N and remove the campaigns the user has already contributed to.

Content-based Filtering[2]

For this approach the campaign data will first need to be cleaned and prepared. Specifically, all relevant features will be merged into one general description; then, this main description will be cleaned and tokenized using TF-IDF. With campaign feature vectors and the relationship between users and campaigns, the user feature's vector can then be inferred (ie. weighted sum of all the features of his/her funded campaigns, or train a linear regressor to find the relation between his/her behavior and the funded campaigns). The user's feature vector will then be used to find campaigns according to their preferences using either cosine similarity, dot product or euclidean distance.

Collaborative Filtering

This approach will require building a different Dataframe with users as rows and campaigns as columns. Let's call this matrix, M. $M(i,j)$ will be 1 if user i funded campaign j, 0 otherwise. This data structure will then be used to learn campaigns and users features based on the overall population behaviour. Matrix Factorization will be used to learn these parameters. The method used in this project is Singular Value Decomposition (SVD) which minimizes an objective function that represents the difference between the original interaction matrix. SciPy provides a method to do SVD on sparse matrix which will be used in this project to calculate the matrix factorization.

Evaluation Metrics

This project will be approached as a ranking task which will ultimately consider only the top-k candidates as the recommendation for the user. According to this, the model will be evaluated by how accurate

[2] Google Developers. Recommendation Systems. Content-based Filtering. https://developers.google.com/machine-learning/recommendation/content-based/basics?hl=en

these recommendations are based on the user's previous interaction. The chosen metric is Recall@N[3] which works as follows:

1. For each user u_i, pick M campaigns the user hasn't funded in the past. The "hard" assumption here is that the user is not interested in these campaigns.
2. Built a set of items composed of the M campaigns found in the previous step plus one campaign funded by the user. For explanation purposes, this campaign will be referenced as c_i. The $M+1$ set will be called the Metrics Set M_i.
3. Generate the recommended candidates for u_i from the set of items M_i.
4. Evaluate if c_i is contained within the top-N recommended items.
5. Aggregate the top-N accuracy metric over all the users.

This project will specifically evaluate the model based on Recall@5 and Recall@10 for benchmarking purposes.

The dataset has a very particular feature which forces a more creative way to measure the model performance. It has to do with the fact that the vast majority of users (~98%) have only funded one campaign; models like popularity and content-based will then be very biased by this fact. Therefore, the recall metrics will be presented and compared based on a new convention Recall@N@M; this means how accurate is the model to recommend good items that are ranked in the top-N, for users that have funded M or more campaigns.

Benchmark

This project mainly attacks two different models: Content Based Model and Collaborative Filtering Model. Both will be compared against the Recall@N metrics as explained in the Evaluation Metrics section. One

[3] Kaggle. Recommender Systems in Python 101. Evaluation. https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101#Evaluation

last model hybrid model will be explored which is a combination of the previously mentioned models.

The model implemented in this project will be Matrix Factorization solved through SVD. It'll be used to learn the user and item features for candidate generation using collaborative-filtering. In content-based filtering there will be no learning involved as similarity is only based on the existing features. Different similarity metrics will be compared such as cosine similarity, pearson correlation and dot product.

A good benchmark model which the results will be compared to is the recommendation system for the Articles Sharing and Reading from CI&T Deskdrop Kaggle Dataset. This recommender was built by Gabriel Moreira as an educational Python Notebook. See footnote for reference[4].

3. Methodology

Data Preprocessing

Both campaigns and payments datasets needed some data cleaning and preprocessing. Find below the process for each one of these datasets:

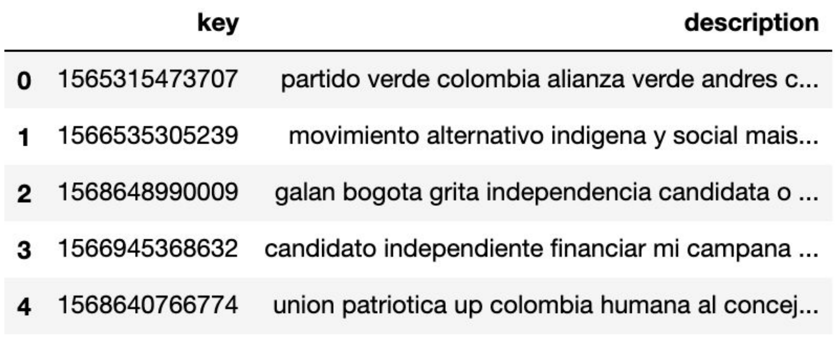
Campaigns

The first step will be to drop any duplicate campaign from the dataset. As argued before, campaign status will not be considered in the recommendation model. The number of contributions column will also be dropped as it seems to only serve the purpose of quick lookup (i.e. it could be inferred by doing a JOIN on the payments dataset).

Content based and Collaborative filtering will both depend on features extracted from the campaign descriptions. Therefore, all the text

[4] Kaggle. Recommender Systems in Python 101. https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101

columns need to be collapsed into one big description and then passed through an HTML-to-plain text algorithm. As mentioned earlier, it's also important to remove the descriptions that only contain the placeholder text as this will be misleading for the algorithm in a later stage.



	key	description
0	1565315473707	partido verde colombia alianza verde andres c...
1	1566535305239	movimiento alternativo indigena y social mais...
2	1568648990009	galan bogota grita independencia candidata o ...
3	1566945368632	candidato independiente financiar mi campana ...
4	1568640766774	union patriotica up colombia humana al concej...

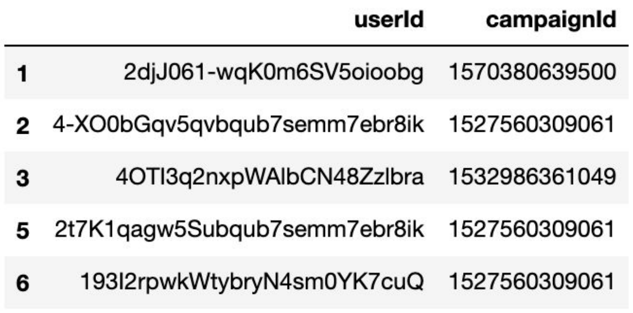
Image 6. Head of the final campaigns DataFrame

Payments

This is a very important part of the project as the assumptions made here are the basis of the underlying behavior of the recommender system. First, it won't matter how much money a user contributed to a campaign; that is, money doesn't imply a user is more strongly connected to a campaign, even more when the majority of the campaigns are social causes. Second, the transaction status will be ignored; it's assumed that the payment could have gone wrong for external reasons so the relevant information bit is the "funding intent". Finally, and as a corollary of the first assumption, the number of times a user funds the same campaign is irrelevant to the problem; that is, the bond strength between a user and campaign is not correlated to the number of fundings said user made to the campaign.

As a consequence of the thoughts described in the preceding paragraph, the raised amount and the transaction status columns will be dropped from the dataset. Also, as per the third assumption, duplicates of the tuple (user_id, campaign_id) will be dropped too.

Finally, it was found that some data instances had funding intents to campaigns no longer existent in the campaigns dataset. In other words, some contributions had been made to invalid campaigns. These instances will be dropped from the dataset



	userId	campaignId
1	2djJ061-wqK0m6SV5oioobg	1570380639500
2	4-XO0bGqv5qvbqub7semm7ebr8ik	1527560309061
3	4OTl3q2nxpWAlbCN48Zzlbra	1532986361049
5	2t7K1qagw5Subqub7semm7ebr8ik	1527560309061
6	193l2rpwkWtybryN4sm0YK7cuQ	1527560309061

Image 7. Head of the final payments DataFrame

Implementation

The implementation architecture is mostly inspired by the Recommender System notebook for the CI&T Kaggle Dataset. This structure proposes a class-based system in which each recommender is written as a class that implicitly implements a common interface: it must implement the "recommend_items" method.

The following paragraphs explains the implementation process for each model on a high level. The evaluation section will be based on the previously proposed metrics: recall@5@1,2,3 and recall@10@1,2,3. For implementation details, please refer to the Jupyter notebook *“Capstone Project - Recommender System - II. Implementation”* which is attached to this report.

Popularity Based Filtering

Model overview

The main idea is to compute the most popular campaigns and recommend the top N that the user hasn't contributed to yet.

Training algorithm

1. Group the payments dataset by campaign
2. Count the number of contributions for each campaign
3. Sort the dataset in descending order. Let's name the resulting dataset C_f.

Recommendation algorithm

Recommend N campaigns for user U_i as follows:

1. Compute the campaigns that user U_i has funded. Let's call this list C_{ui}.
2. Filter out from C_f the campaigns found in C_{ui}. This will be C_{f-ui}.
3. Take the top N campaigns from C_{f-ui}.

Evaluation

recall@5@1	**recall@10@1**	**recall@5@2**	**recall@10@2**	**recall@5@3**	**recall@10@3**
0.862	0.933	0.828	0.928	0.705	0.833

This model performed impressively well despite being the simplest recommendation algorithm. It may be a consequence of unbalanced data (very few popular campaigns that happen to be funded by most users), or it may be showing that, indeed, the users on this platform are very biased to fund the most successful (ie popular) campaigns. In general, this is an expected behavior in most recommendation systems[5]. Popularity models

[5] Kaggle. Recommender Systems in Python 101. Popularity Model. https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101#Popularity-model

perform very well as humans tend to follow viral trends regardless of individual preference.

Content Based Filtering

Model overview

This filtering method works under the assumption that users will potentially like items that are similar to the ones they previously interacted with.

Training algorithm

1. For each campaign c_i, clean up and merge all text features that can be considered relevant for descriptions. This process involves removing spanish stop words and stemming.
2. Train a TfidfVectorizer with 5000 words on the test and then transform each campaign to a text embedding of shape (1, 5000). Let's call the resulting dataset C_v, where $C_v.iloc[i]$ represents the feature vector for campaign i.
3. Calculate user profiles based on the campaigns they have liked before:
    a. For each user U_i, use the 'payments' dataset to compute the campaigns this user has funded. Let's call this list C_{ui}.
    b. Filter from C_v the campaigns found in C_{ui}. This will be called C_{v-ui}.
    c. Create vector u_{i-v} by averaging the values from the TFIDF embeddings in C_{v-ui}. This will represent a user as it contains, on average, what features are more important for this user when choosing a campaign to fund. This vector will have shape (1, 5000).
    d. Aggregate all the user profiles into one dataset called U_p.

Recommendation algorithm

Recommend N campaigns for user U_i as follows:

1. Extract the user profile from U_p. This is $U_p^{(i)}$.
2. Calculate the cosine similarity of the user profile with the feature vector of every campaign in C_v. Let's call this dataset C_{cs} which represents how similar a campaign C_i is to the campaigns user U_i has previously contributed to.
3. Sort the results in descending order.
4. Compute the campaigns that user U_i has funded. Let's call this list C_{ui}.
5. Filter out from C_{cs} the campaigns found in C_{ui}. This will be C_{cs-ui}.
6. Take the top N campaigns from C_{cs-ui}.

Evaluation

recall@5@1	**recall@10@1**	**recall@5@2**	**recall@10@2**	**recall@5@3**	**recall@10@3**
0.998	0.998	0.948	0.961	0.745	0.813

As per the metrics obtained, it can be seen that the model works very well for users that have only funded one campaign but quickly decreases in performance for more active users (more than 3 funded campaigns). This was expected as the cosine similarity will of course score ‘1’ for the exact similar campaign that the user funded, and the dataset is very influenced by users with contributions to only 1 campaign.

This is also a byproduct of the main characteristic of this dataset: the vast majority of the users have only funded 1 campaign, and most of these users have funded the top 5 popular campaigns. This is a problem known as cold start which is essentially what this model is aimed for:

given very limited user information, do your best to, at least, find items similar to the only ones this user has liked in the past.

Collaborative Filtering

Model overview

This is the most powerful method out of the three investigated in this project. It recommends items based on the behavior of the entire population. If two users U_a and U_b have funded similar campaigns, then it makes sense to recommend U_a some of the campaigns that U_b has contributed to before.

Training algorithm

The most important concept in this type of filtering is the funding matrix F. This matrix represents the relations between the users and the campaigns they have funded. F_{ij} equals 1 if user i funded campaign j, or 0 otherwise.

In the algorithm below, P_F refers to the payments dataset:

1. Add a new "funded" column with value '1' in P_F.
2. Build matrix F by pivoting columns vs values in P_F. Use the "funded" column as the relationship value.
3. Factorize F with Singular Value Decomposition (SVD) using an arbitrary number of factors equals to 15. This iteration is based on the same number of factors used in the benchmark in order to compare performance on the same basis later on. This factorization will return three matrices: U, Σ and V.
4. Calculate the user prediction matrix by multiplying matrices U, Σ and V. This will, of course, be similar to the original F matrix but with real values instead of 0s and 1s. Let's call this matrix U_p.

5. Normalize U_p so that the max value becomes 1 and the lowest becomes 0.

At this point, $U_p^{(ij)}$ represents the prediction for user i to fund campaign j.

Recommendation algorithm

Recommend N campaigns for user U_i as follows:

1. Find the predictions vector for user U_i by accessing U_p with index U_{i-id}. Let's call this vector r_{ui}.
2. Sort vector r_{ui} in ascending order. This will put on top the campaigns that are most likely to be funded by the user.
3. Compute the campaigns that user U_i has funded. Let's call this list C_{ui}.
4. Filter out from r_{ui} the campaigns found in C_{ui}. This will be $r_{f-\{ui\}}$.
5. Take the top N campaigns from $r_{f-\{ui\}}$.

Evaluation

recall@5@1	**recall@10@1**	**recall@5@2**	**recall@10@2**	**recall@5@3**	**recall@10@3**
0.916	0.961	0.939	0.990	0.970	0.990

This result validates the initial hypothesis: Collaborative Filtering is much more accurate than Content-based filtering when there are more users with several funded campaigns. It can be then inferred that this population tends to fund campaigns that their similar peers have funded. Item similarity plays an important role but it's not the sole underlying motivation for funding.

Refinement

It's well known that ensemble methods are very popular ways to improve the performance of several already optimized individual models. This was evidenced when Netflix granted 1 million dollars to the winner of its recommendation engine competition: the winner created a solution that ensembled dozens of recommendation models.

As shown in the previous sections, content based models work well for users with very few funded campaigns (almost 1), and collaborative filtering shows great performance for users who have contributed to 2 or more campaigns. An ensemble will then allow both models to contribute to recommendations that are well suited for both types of users.

Hybrid Model

Model overview

This section will explore the performance of a Hybrid model which will recommend campaigns based on a weighted combination of the most relevant methods previously explored: Content-based and Collaborative Filtering.

Recommendation algorithm

Recommend N campaigns for user U_i as follows:

1. Compute the recommendations for user U_i using the content based model. Let's call this matrix R_{cb}.
2. Compute the recommendations for user U_i using the collaborative filtering model. Let's call this matrix R_{cf}.
3. Merge both matrices by column Campaign ID. Let's call this matrix R_w. Every row will then be a tuple of the form:

$$(campaignID,\ contentBasedStrength,\ collaborativeStrength)$$

4. Compute the new recommendation score by weighting both scores in a new column $hybridStrength$. As collaborative filtering showed to be more consistent for $Recall@N@M,\ M > 1$, this model will then have more weight in the final score than the content based model.
5. Sort R_w by $hybridStrength$ in descendent order. Let's call it R_{wf}.
6. Take the top N campaigns from R_{wf}.

Evaluation

recall@5@1	**recall@10@1**	**recall@5@2**	**recall@10@2**	**recall@5@3**	**recall@10@3**
0.999	0.999	0.998	1.0	1.0	1.0

The hybrid model indeed outperforms both models (content and collaborative). It shows great performance for users with low number of campaigns (*cold start*) and almost perfect scores for users with 2 or more funded campaigns.

4. Results

Model Evaluation and Validation

The previous section showed individual analysis for the results obtained with every model (popularity, content, collaboration and hybrid). This section will show the most important result which proves how the ensembled outperformed its peer models.

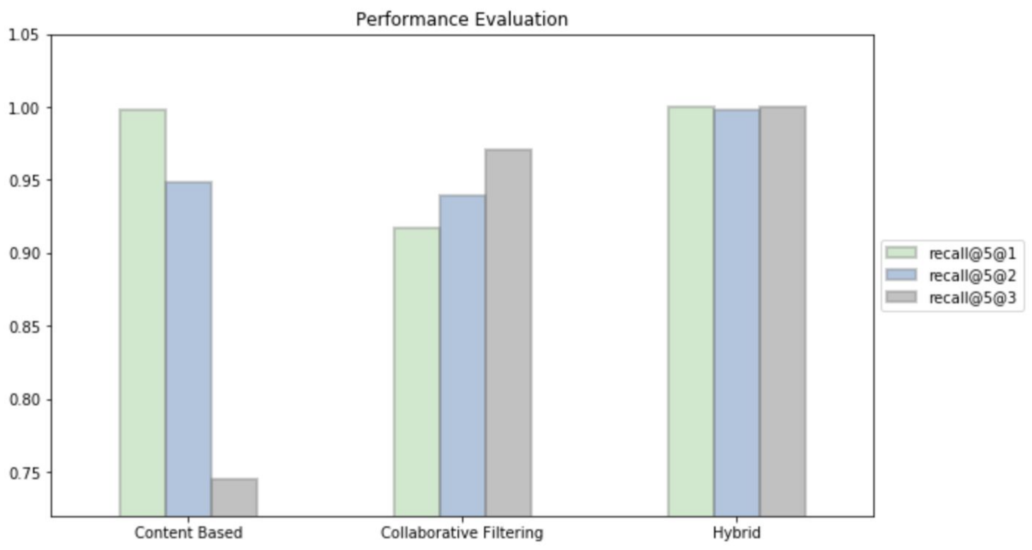


Image 8. Model comparison based on Recall@5@M

The figure above compares the Recall@5@M metric for content based, collaborative and hybrid models. These bars show some very interesting points discussed below:

1. It's clearly evidenced that the content based is a very good fit to solve the *cold start* problem as it works almost flawlessly for users who have contributed to one campaign only. This behaviour is very representative of Crowdfunding as the vast majority of users will only contribute to 1 or 2 campaigns. It can also be seen how the model recall decreases for more active users as they don't only contribute to similar campaigns but instead are influenced by some other factors.

2. Collaborative filtering is very well suited for richer datasets; e.g. datasets in which users interact much more with the items. This can be clearly seen as the model performance increases when evaluated over a dataset with users that have contributed to 3 or more campaigns.

3. It was possible to build the winner model using the simplest ensemble strategy: weighted sum. The bars clearly show how it's possible to get the best of both models: content and collaborative

filtering. It performs well on both ranges: passive and *very* active users.

Justification

This section will be divided into two parts: benchmark model comparison and examples of real recommendations given by the model.

Benchmark Comparison

The benchmark set for comparison at the beginning of this report is a Recommender System notebook for the CI&T Kaggle Dataset. As the nature of the datasets are intrinsically different (i.e. that dataset is much richer as it has several users interacting with several items at once), so are the absolute values obtained in the Recall metrics. Nevertheless, the relative deltas between the different models can still show similarity in the results.

Let's compare the performance of the main models: Content Based, Collaborative Filtering and Hybrid. The benchmark model was trained on users that had at least 2 interactions with the items; therefore, the metric used to compare against the benchmark will be Recall@5@3 and Recall@10@3.

Content Based Model

Despite the fact that the recall metrics are very different in its absolute values, it's clear that the results are somewhat similar in some sense:

1. None of the models achieved an spectacular performance (0.16 in benchmark, 0.74 in capstone) which suggests, again, that the content based model is not as performant for rich datasets.

2. There's a marginal growth between top-5 and top-10 of about 10% in both models. This was expected since taking more items to evaluate will increase the probability of finding the recommended item in the top.

	recall@5@3	**recall@10@3**
Benchmark	0.16	0.26
Capstone	0.74	0.81

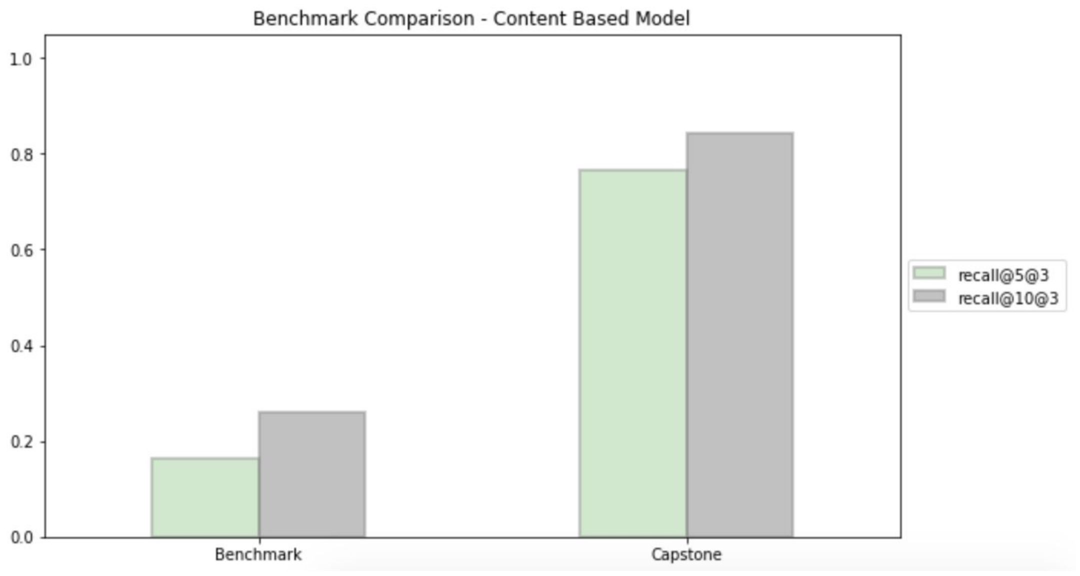


Image 9. Benchmark evaluation for the Content-based model

Collaborative Filtering Model

As shown below, both models presented a very important increase in its performance from Content-based to Collaborative filtering (almost 16 points in recall score). This is indeed reinforcing the hypothesis that populations tend to interact in a very collaborative way instead of relying purely on their own preferences.

	recall@5@3	**recall@10@3**
Benchmark	0.33	0.46
Capstone	0.97	0.99

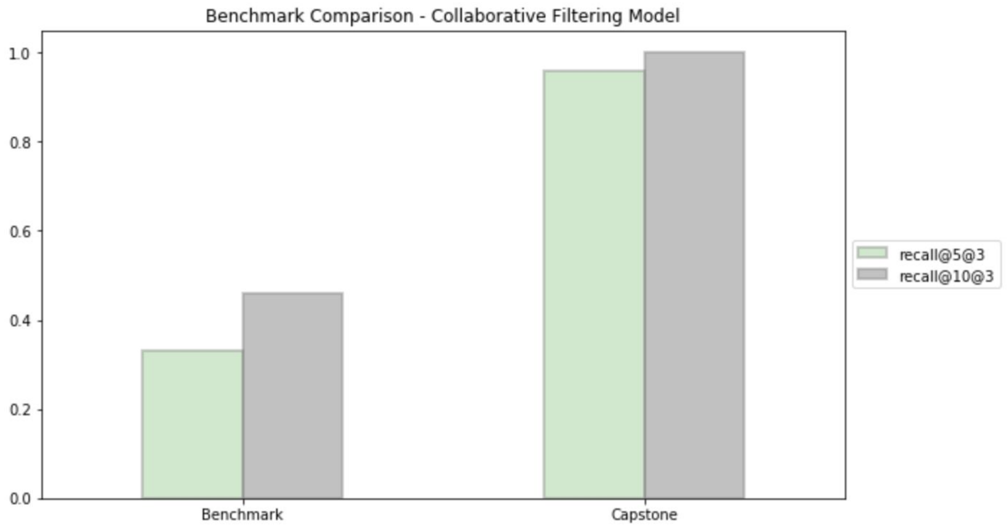


Image 10. Benchmark evaluation for the Collaborative Filtering model

Hybrid Model

The benchmark showed that ensemble methods are likely to outperform its individual composing models, at least when applied to recommendation systems. The metrics reported above for the Capstone showed the same behavior effectively aligning to the initial hypothesis: a hybrid system would outperform both the content and collaborative filtering models. The bar chart presented below shows a summary of the performance for the models studied both in benchmark and capstone projects. Comparisons are based on recall@5@3. In both cases, Collaborative Filtering models are better than Content Based models, but the most performant one is always the Hybrid model.

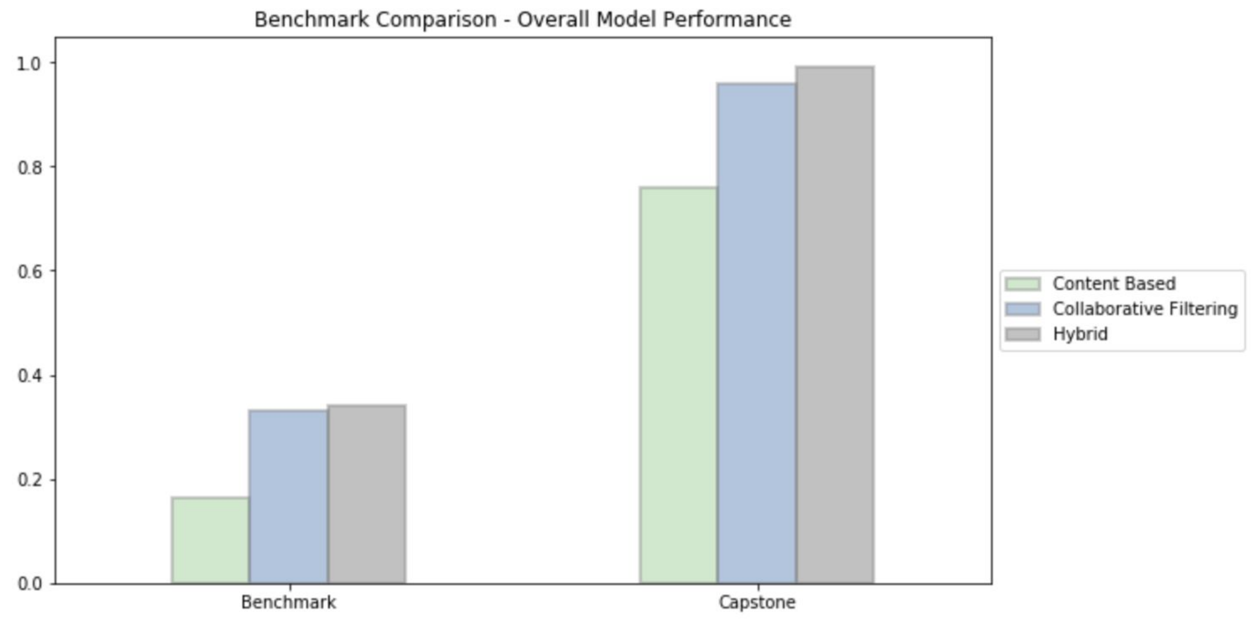


Image 11. Benchmark evaluation for the Hybrid model

Recommendation Engine: Inference testing

This section presents the final outcome of the recommendation engine applied to real users and campaigns. Even when the model was trained with descriptions, the results will only show their original names to keep the examples simple. Brief descriptions will be presented in front of some campaigns to give more context. Campaign names are written in Spanish. Test users are chosen arbitrarily to demonstrate the model effectiveness. User IDs are obfuscated from their original values.

Example 1

UserID: 2dSV4rSxyWZ5ZoOT57uunnnQu8Y

User Campaigns:

- #SueñoRobótico: *Help school children participate in a robotics competition in the US.*

Recommendations:

- Mundial ROBOTICA HOUSTON
- Pasajes para Niños amantes de la tecnología al mundial de Robótica en Rumania
- Grupo de robótica representará a Colombia en RoboRAVE internacional
- Cubrir el costo de mis estudiantes a Corea del Sur
- Estudiantes de Robótica por un Gran Sueño!

This is a great example of how the model handles a cold start. The user only contributed to one campaign to help students participate in a robotics competition abroad. The top-5 campaigns recommended by the model are all related to this topic: help children travel abroad to participate in robotics competitions.

Example 2

UserID: 5tDc3K6xx5Syoqya39SFmbjOtcVl2dbb

User Campaigns:

- Realizar mi sueño, Mi Primer Disco: *Help a local indie artist to record an album.*

Recommendations:

- Unidos de la mano en una sola voz
- "Manglares" Nuevo CD Urpi Barco
- Lanzamiento álbum Martino Park
- Primer sencillo. ¡El paso inicial de mi carrera!
- Lanzar el sencillo de la Juan Direction Orquesta

Similar to the previous example, here the model demonstrates it can effectively cope with a cold start. The user had funded a campaign to fund a local artist record her first album and all the recommendations were related to exactly that cause: help indie artists record their music.

Example 3

UserID: 29Pc4KqwxqyAlbCN48Zzlbra

User Campaigns:

- Del Urabá a la Nieve 2018
- #HablemosDeLasNiñas: Una conversación de Mutante
- Desarrollar empresa cervecera para la inclusión social
- Operación de Reemplazo de cadera

Recommendations:

- NOSOTRAS DOCUMENTAL
- Proyecto El Río
- LA NEGOCIACION DOCUMENTAL
- Del Urabá a la Nieve 2019
- Líderes Sin Olvido

In this example, the user has contributed to 4 campaigns which makes the problem more suitable to a collaborative influence in the recommendation. In fact, the funded campaigns are somewhere in the line of social change, war and peace, and documentary funding. The recommendations are also aligned with these principles.

5. Conclusions

Similar to many other types of machine learning problems, recommendation systems are heavily dependent on the quality of the dataset. A *high quality* dataset for these models would be one that contains many users interacting with many items.

The particular shape of this dataset, which is normal in crowdfunding systems, made it necessary to explore new ways of measuring and comparing model performance: it’s not very representative to compare with Recall@5 only as the data is biased towards users funding 1 campaign. Recall@N@M was introduced in an attempt to solve this problem.

Although popularity based recommendations were not included in the final hybrid solution, it was shown that these models can definitely work well as a very simple recommendation system. Even more, its recommended items can be randomly included in the final N candidates.

Ensemble models proved, once again, to be very powerful. This could be evidenced as the hybrid model outperformed Content-based and Collaborative Filtering effectively dealing with *cold starts* and users with more data associated.