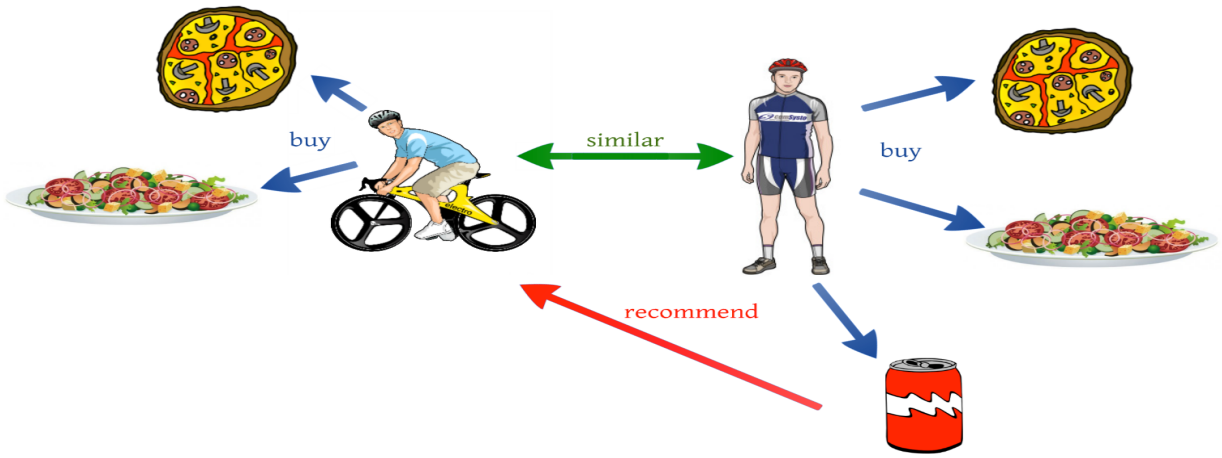
Recommendation Engine Case Study

Objective: Targeted Upselling and cross-selling of services, tailored specifically for customer purchase behavior**.**

Client: Medium-sized digital wallet company with multiple utilities payment and recharge services.

Approach: Use of collaborative filtering model to capture the complex relationship between users and their purchase behavior to produce a custom list of recommended services for each user.

In collaborative filtering, algorithms are used to make automatic predictions about a user's interests by compiling preferences from several users.



Understanding Collaborative filtering from layman's perspective

In our case, we had transaction data as interactions between users and services. Transactional data is an implicit form of interaction where we cannot infer that if a user has bought a service, he liked it or not. This is because the user has not rated the service after buying it. And thus, there is no feedback from user available.

But this approach has a small caveat. Since, there is no feedback from user regarding the service, we can assume that if the user has taken a service multiple times, he has liked the service. With this small practical assumption we have made our collaborative filtering model.

Method:

1. Active customers are recognized on the platform (around 600k).
2. Transaction log for all active customers are acquired for all the successful transactions.
3. Transation log is converted into count data where the count of number of times a user has bought a service is captured.
4. This data is fed to a collaborative filtering model. We have used matrix factorization for this purpose.
5. Historical transaction behavior data is used to improvise the collaborative filtering model.

Model Details:

Matrix Factorization trains a model capable of predicting a score for each possible combination of users and items. The internal coefficients of the model are learned from known scores of users and items. Recommendations are then based on these scores.
The user-item rating matrix is factorized as the product of two lower dimensional matrices, the first one has a row for each user, while the second has a column for each item. The row or column associated to a specific user or item is referred to as *latent factors*.

The predicted ratings can be computed as $\tilde{R} = HW$, where $\tilde{R} \in \mathbb{R}^{users \times items}$ is the user-item rating matrix, $H \in \mathbb{R}^{users \times latent factors}$ contains the user's latent factors and $W \in \mathbb{R}^{latent factors \times items}$ the item's latent factors.

Specifically, the predicted rating user *u* will give to item *i* is computed as:

$$\tilde{r}_{ui} = \sum_{f=0}^{nfactors} H_{u,f} W_{f,i}$$

The expressive power of the model is tuned by changing the number of latent factors. Matrix factorization with one latent factor is equivalent to a *most popular* or *top popular* recommender (e.g. recommends the items with the most interactions without any personalization). Increasing the number of latent factor improves personalization, therefore recommendation quality, until the number of factors becomes too high, at which point the model starts to overfit and the recommendation quality will decrease.

Result:

Cross-selling of services from recommender has shown **18% increase in Revenue** for active users on the platform.