Recommendation Systems

INFO7374 - Team 6

Assignment 4

# **What are Recommendations?**

In today’s world with the plethora of information that is constantly tracked from us, it comes to no surprise that many online websites or social media websites come up with recommendations on what to buy or what to watch through their sites. The main components for these recommendation systems are our purchase/watch/search history.

For example: Let’s look into one of our Amazon Account

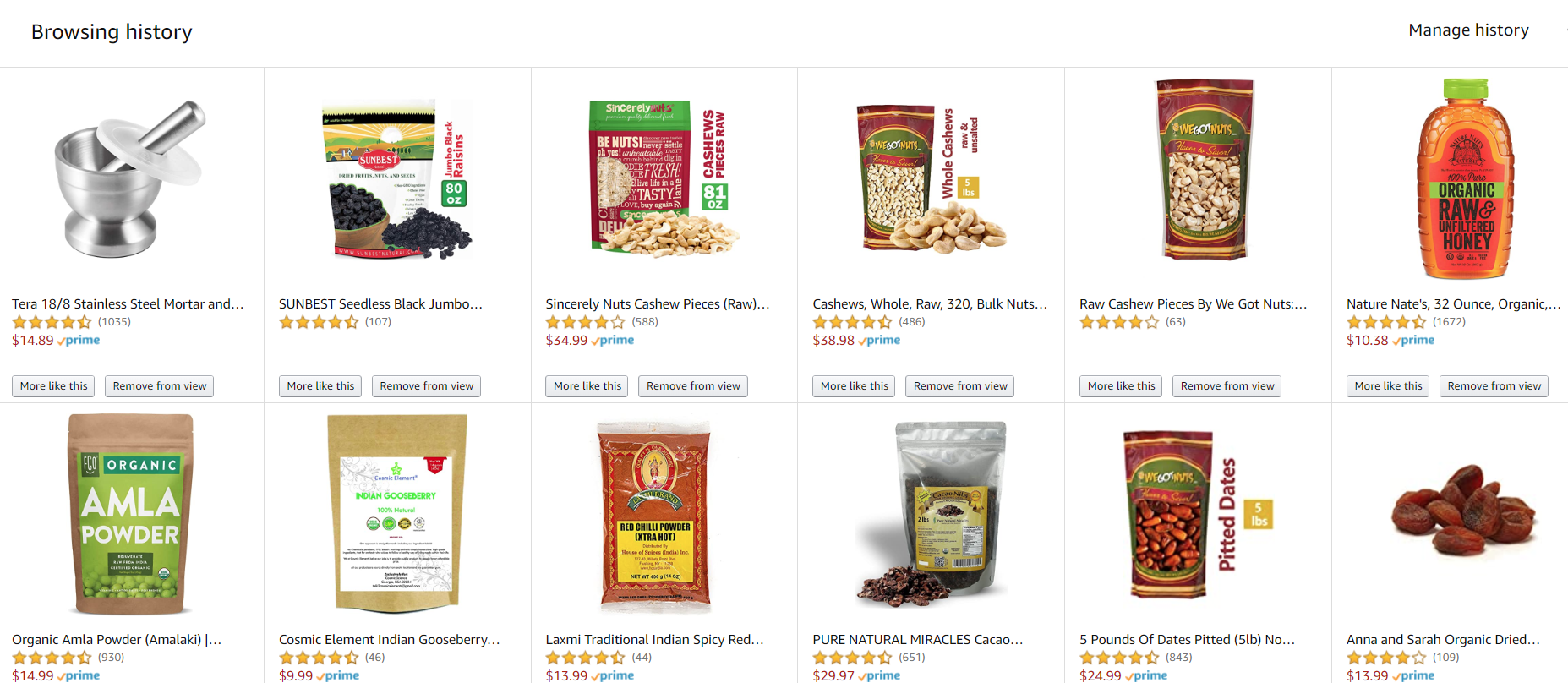


Figure 1- Browsing History in Amazon

So, now Amazon starts predicting our future buy and starts recommending similar products. Also, amazon knows that we are students in the United States and are currently facing an unexpected outbreak, so starts recommending products which might be useful in the immediate future.

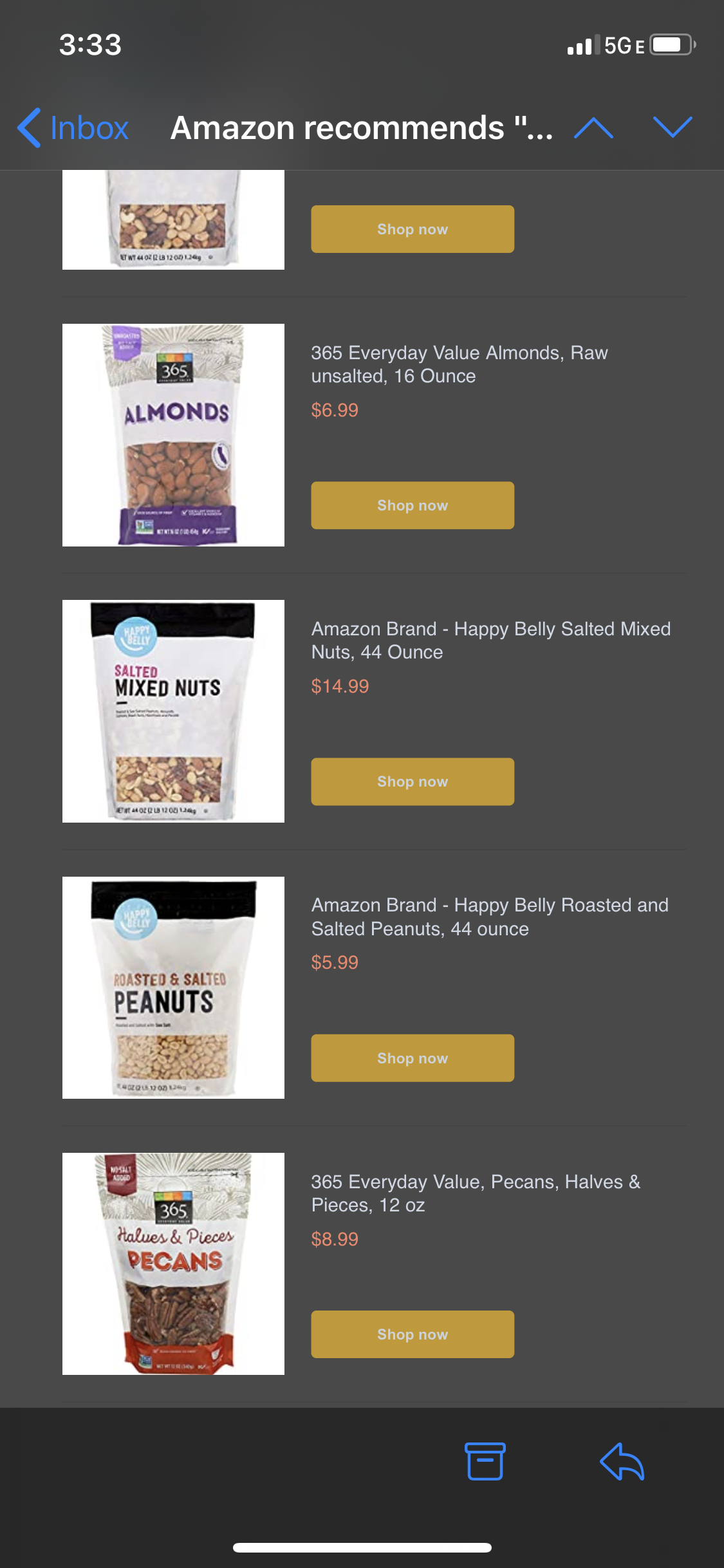
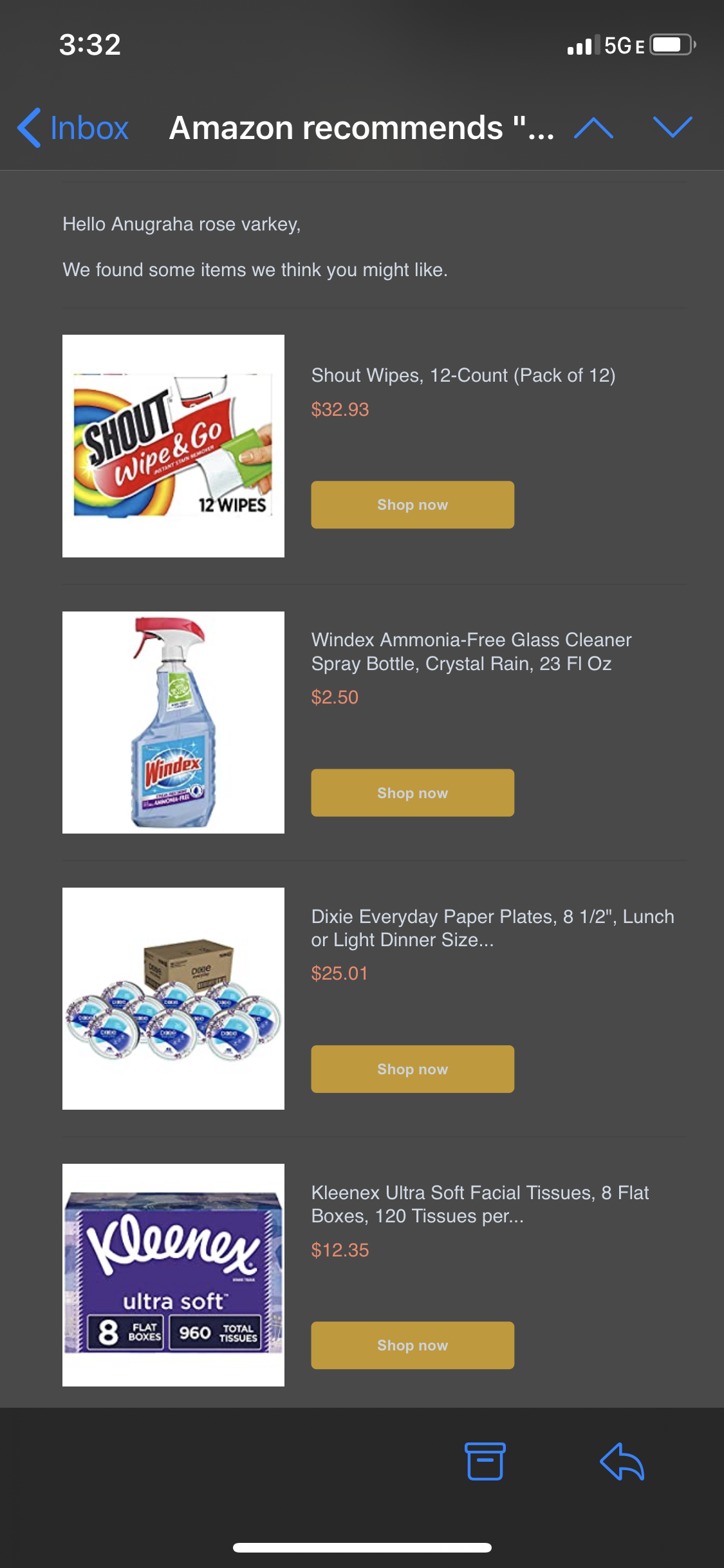
 

Figure 2- Amazon Recommendations

There are two type of approaches which is generally used in recommendation system:

1- Content Based Filtering

2- Collaborative based filtering

## Content Based Filtering

Linear attribution gives a more balanced look at the whole marketing strategy. This is a great model to analyze whether certain events are overvalued or undervalued.Since the linear model highlights all events, contribution of mid-funnel channels can be understood and sometimes patterns can be spotted in buyer journeys.

## Collaborative Based Filtering

It is based on the idea that people who share the same interest in certain kinds of items will also share the same interest in some other kind of items unlike content based which basically rely on metadata while it deals with real life activity. This type of filtering is flexible to most of the times but because of cold start problem, data sparsity (which was handled by matrix factorization) these type of algorithm faces some setback in some scenario

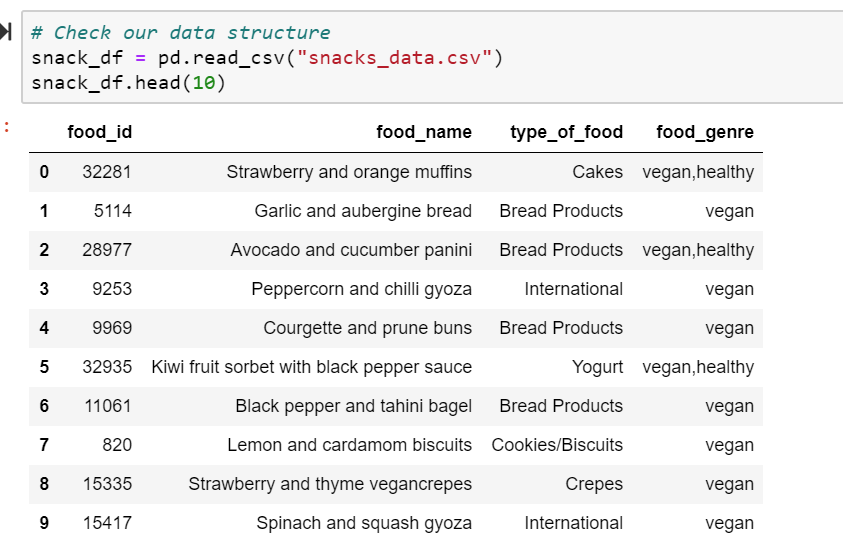
## Matrix Factorization

is the most used variation of Collaborative filtering. It uses a fixed inner product of the user-item matrix to learn user-item interactions, but Matrix Factorization used in this type of problem is generally sparse because there is a chance that one user might rate only some movies. This is when Neural Collaborative Filtering (NCF) comes into play

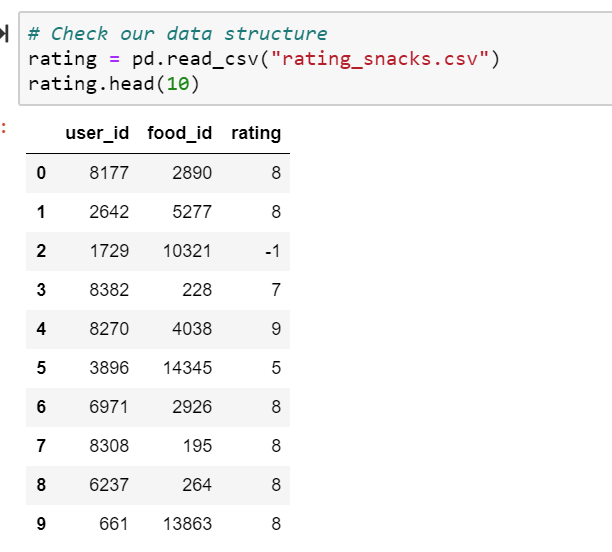
# **Recommendation System**

In our recommendation system, we have built, evaluated and deployed a recommendation system based on the above mentioned method of NCF. We have used a sample dataset which has the user’s purchase history and their corresponding rating history. we created this dataset using the Faker package in Python

[Snack Data](https://drive.google.com/file/d/12UvhC3dp9HVEmDqUS7AaI5dy5rrAi9E6/view?usp=sharing)



[Rating Data](https://drive.google.com/file/d/1KbQ489YLuqUvZNh42Q0molZH8htRPFkL/view?usp=sharing)

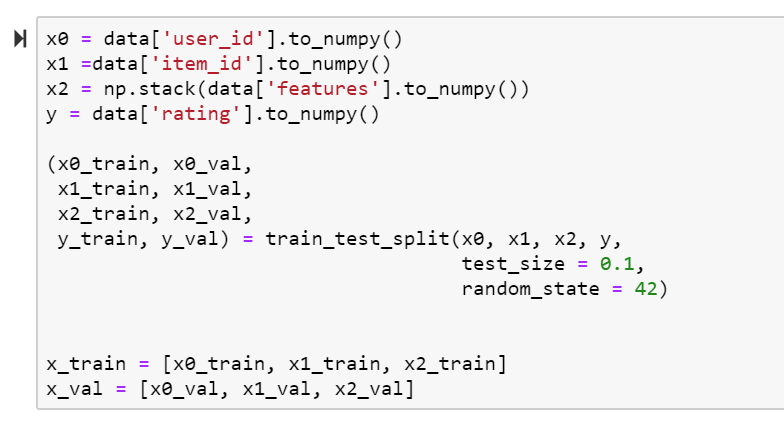


## Preparing the data (Stage 1)

### Construct training and testing sets

Our current dataset is incomplete, since we need to generate rows including snacks that users' haven’t bought (*negative instances*). We need to emphasize that we don't want every user to have a row for every anime, to not fill up our entire RAM memory. Hence we set that every rating will trigger 4 negative entries (we picked 4 just as a fiducial value from the original repo). To generate these records, we simply sample 4 unbought snacks for each user rating, later we randomly splitting our dataset into test and train, as

we do not have any data which gives the information about the timing of the purchase made, we cannot do a chronological split.



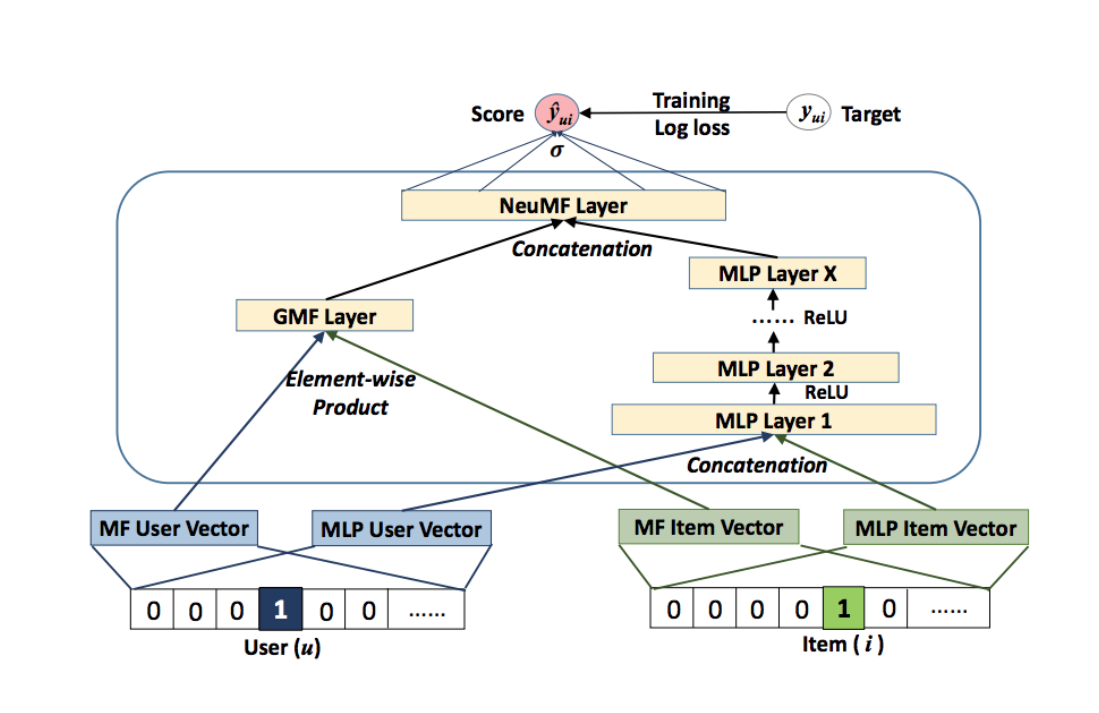
# **Model Implementation (Stage 2)**

## Neural Collaborative Filtering (NCF)

replaces the user-item inner product with a neural architecture. NCF has 2 components GMF and MLP with the following benefits

* **GMF** that applies the linear kernel to model user-item interactions like vanilla MF
* **MLP** that uses multiple neural layers to layer nonlinear interactions

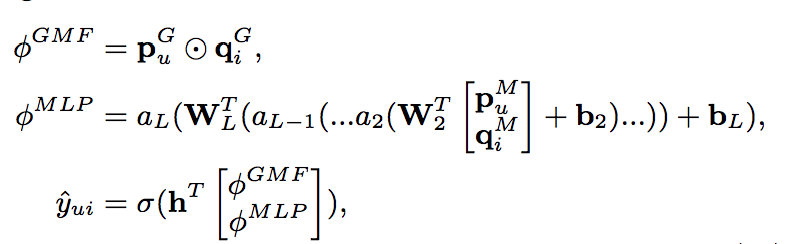
NCF combines these models together to superimpose their desirable characteristics. NCF concatenates the output of GMF and MLP before feeding them into the NeuMF layer.



### Steps of Execution:

1. GMF/MLP have separate user and item embeddings. This is to make sure that both of them learn optimal embeddings independently.
2. GMF replicates the vanilla MF by element-wise product of the user-item vector.
3. MLP takes the concatenation of user-item latent vectors as input.
4. The outputs of GMF and MLP are concatenated in the final NeuMF(Neural Matrix Factorisation) layer.

The score function of equation 1 is modeled as



***G:*** GMF

***M:*** MLP

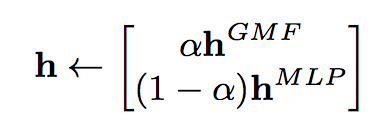
***p:*** User embedding

***q:*** Item embedding

This model combines the linearity of MF and non-linearity of DNNs for modeling user-item latent structures through the ***NeuMF (Neural Matrix Factorisation)*** layer.

Due to the non-convex objective function of NeuMF,gradient-based optimization methods can only find locally-optimal solutions. This could be solved by good weight initializations. To solve this ***NCF*** initializes GMF and MLPwith pre-trained models. There are 2 ways to do this

1. Random Initialisation  
   1. Train GMF+MLP with random initializations until convergence.  
   2. Use model parameters of 1 to initialize NCF.  
   3. The weights of the two models are concatenated for the output layer as
2. GMF + MLP from scratch



where

***h(GMF)***: h vector of the pre-trained GMF

***h(MLP)***: h vector of the pre-trained MLP

***alpha:*** Hyper-parameter determining the trade-off between the 2 pre-trained models

1. GMF + MLP from scratch  
   1. ***Adaptive Moment Estimation (Adam)*** adapts the learning rate for each parameter by performing smaller updates for frequent and larger updates for infrequent parameters. The Adam method yields faster convergence for  
   both models than the vanilla SGD and relieves the pain of tuning the learning rate.  
   2. After feeding pre-trained parameters into NeuMF, we optimize it with the vanilla SGD, rather than Adam. Adam needs to save momentum information for updating parameters. As the initialization with pre-trained networks does not store momentum information.

This completes the theory of NCF

### In short:

NCF learns a probabilistic model that emphasizes the binary property of implicit data. We discussed how MF can be expressed and generalized under NCF (Using General Matrix Factorisation {GMF}). NCF explores the use of DNNs for collaborative filtering, by using a multi-layer perceptron (MLP) to learn the user-item interaction function. Lastly, we discussed a new neural matrix factorization model called NeuMF, which ensembles MF and MLP under the NCF framework; it unifies the strengths of linearity of MF and non-linearity of MLP for modeling the user-item latent structures

<https://medium.com/@paritosh_30025/recommendation-using-matrix-factorization-5223a8ee1f4>

# **Evaluation (Stage 3)**

Logistic loss (sometimes it is called simply log loss, or cross-entropy loss) is a useful metric to evaluate the hit accuracy. It is defined as the negative log-likelihood of the true labels given the predictions of a classifier.

An Epoch value of 3 is taken so that a more balanced dataset is created



It can be seen that the loss is more close to 0, and, by definition, it means that the predictions are generating better results.

# **Streamlit Prototype**

We have put together the entire model and created a dashboard for our recommendation system (NCF) using Streamlit. The end user of this dashboard can enter a user\_id and the dashboard displays “Activity of the user” and the “Recommendations of the user” given by our model

