POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Design and Development of a Job Recommender System

Relatore
Prof. MAURIZIO MORISIO
Giuseppe Rizzo
Enrico Palumbo
Diego Monti

Candidato
Ayele Fassil Abebe

OCTOBER 2019

Table of Contents

ABSTRACT

The ACM RecSys Challenge 2017 was focused on the problem of job recommendations on XING in a cold-start scenario. Given fixed historic dataset and fixed targets the goal of the challenge is to identify those users that might be interested in getting notified about the job posting and are also appropriate candidates for the given job.
This dissertation analyzes and implements a general purpose supervised learning algorithm called factorization machines (FM) to make job recommendations to appropriate candidates. The algorithm can be used for both classification and regression tasks.
Since our task is to predict whether a user will positively interact with an item (job) or not, a Factorization machines classifier will be implemented. There are three major phases to implement this classifier, i.e. **Data preparation** (Pre-processing) which involves merging interactions data with user/item features data and one hot encoding, **Training phase**, which involves splitting the dataset to training and test, creating FM model then training the model, **Prediction phase**, that is the final step which tries to make predictions to the test set then generate a recommendation file.

ACKNOWLEDGMENT

My special gratitude goes to Giuseppe Rizzo, for the remarks and engagement through the making of my thesis, for making sure that I was provided with all the necessary guidance to accomplish the task I was assigned for. I also want to thank Enrico Palumbo and Diego Monti who closely provide me professional guidance and follow-ups while I was doing this thesis. And many special thanks to all of those who have been with me all along and helped me through different ways. Finally I would like to thank the almighty God for giving me all the courage I needed to accomplish this thesis with success.

List of Figures

Acronyms

FM	Factorization Machines
RSs	Recommender systems
CF	Collaborative Filtering
CBF	Content based Filtering
SVM	Support Vector Machines
ACM	Association of Computing Machinery
SVD	Single Value Decomposition

List of Tables

Chapter One

1. INTRODUCTION

Recommender systems (RSs) are a set of software tools and techniques that provide item suggestions that are most likely of interest to a particular user. It means they are primarily directed towards users who lack sufficient personal experience and competence to evaluate the overwhelmingly increasing number of alternative items that a website offers. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read.

“Item” is a term used to imply what the RS recommends to users. A RS normally focuses on a specific type of item (e.g. news, jobs, movies...) and, accordingly, its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item.

RSs are becoming the most important features of modern websites. Diverse applications in areas such as E-commerce, search engines, Internet music and video, gaming and Online dating apply similar techniques that leverage large volumes of data to better fulfill a user’s need in a personalized fashion. Especially commercial websites like *amazon* benefit from a boost in customer loyalty, click through rates and revenue when implementing recommender systems that provide each customer personalized product recommendations that the user might be interested in.
For many practitioners and researchers in the fields of recommendation systems, their main focus has been to accurately predict ratings i.e. predict a metric, real valued variable. This main focus of accurate rating prediction was mainly triggered by the Netflix prize challenge which took place from 2006 to 2009.

Most recommendation problems assume that we have a consumption/rating dataset formed by a collection <user, item, rating> tuples. This has been a starting point for most variations of collaborative filtering algorithms and they have proven to yield nice results, However in many applications, we have plenty of metadata(titles, categories, tags….) that can be used to make better prediction.

The main purpose of this thesis is to implement a general purpose Factorization machines model that in a natural way includes extra features in the model and make better predictions even under sparse settings.

Chapter 2

2. BACKGROUND OF RECOMMENDER SYSTEMS

2.1. What are Recommender Systems

Recommender systems emerged as an independent research area in the mid-1990s. In recent years, the interest in recommender systems has dramatically increased, as the following facts indicate:

1. Recommender systems play an important role in highly-rated Internet sites such as Amazon, YouTube, Netflix, Spotify, LinkedIn, Facebook and many others. Moreover many media companies are now developing and deploying RSs as part of the services they provide to their subscribers. For example, Netflix, the online provider of on-demand streaming media, awarded a million dollar prize to the team that first succeeded in substantially improving the performance of its recommender system.

2. There are conferences and workshops dedicated specifically to the field, namely the **Association of Computing Machinery's (ACM)** Conference Series on Recommender Systems (RecSys), established in 2007. This conference stands as the premier annual event in recommender technology research and applications. In addition, sessions dedicated to RSs are frequently included in more traditional conferences in the area of databases, information systems and adaptive systems. Additional noteworthy conferences within this scope include: ACM's Special Interest Group on Information Retrieval (SIGIR); User Modeling, Adaptation and Personalization (UMAP); Intelligent User Interfaces (IUI); World Wide Web (WWW); and ACM's Special Interest Group on Management Of Data (SIGMOD).

3. At institutions of higher education around the world, undergraduate and graduate courses are now dedicated entirely to RSs, tutorials on RSs are very popular at computer science conferences, and a book introducing RSs techniques has been published as well . Springer is publishing several books on specific topics in recommender systems in its series: Springer Briefs in Electrical and Computer Engineering. A large, new collection of articles dedicated to recommender systems applications to software engineering has also recently been published.

4. There have been several special issues in academic journals which cover research and developments in the RSs field. Among the journals that have dedicated issues to RSs are: AI Communications (2008); IEEE Intelligent Systems (2007); International Journal of Electronic Commerce (2006); International Journal of Computer Science and Applications (2006); ACM Transactions on Computer Human Interaction (2005); ACM Transactions on Information Systems (2004); User Modeling and User-Adapted Interaction (2014, 2012); ACM Transactions on Interactive Intelligent Systems (2013); and ACM Transactions on Intelligent Systems and Technology (2015).

2.2. Recommender Systems Evaluation

Recommender systems research is being conducted with a strong emphasis on practice and commercial applications. One very important issue related to the practical side of RS deployment is the necessity of evaluating the *quality* and *value* of the systems. Evaluation is required at different stages of the system's life cycle and for various purposes. At *design time*, evaluation is required to verify the selection of the appropriate recommender approach. In the design phase, evaluation should be implemented off-line and the recommendation algorithms, i.e., their computed recommendations, are compared with the stored user interactions. An off-line evaluation consists of running several algorithms on the same dataset of user interactions (e.g., ratings) and comparing their performances. This type of evaluation is usually conducted on existing public benchmark data if appropriate data is available, or, otherwise, on collected data. The design of the off-line experiments should follow known experiment design practices in order to ensure reliable results. Off-line experiments can measure the quality of the chosen algorithm in fulfilling its recommendation task. However, such evaluation cannot provide any insight about the user satisfaction, acceptance or experience with the system. The algorithms might be very accurate in solving the core recommendation.

2.3. Recommender Systems Applications

Recommender systems research, aside from its theoretical contribution, is generally aimed at practically improving industrial RSs and involves research about various practical aspects that apply to the implementation of the systems. Indeed, an RS is an example of large scale usage of machine learning and data mining algorithms in commercial practice. The common interest in the field, both from the research community and from the industry has leveraged the availability of data for research on one hand, and the evolving of enhanced algorithms on the other hand. Practical related research in RSs examines aspects that are relevant to different stages in the life cycle of an RS, namely, the design of the system, its implementation, evaluation, maintenance and enhancement during system operation. The Netflix Prize announced in 2006 was an important event for the recommender systems research community and industry, and their mutual interaction. It highlighted the importance of the recommendation of items to users and accelerated the development of many new data mining recommendation techniques. Even though the Netflix Prize initiated a lot of research activities, the prize was a simplification of the full recommendation problem. It consisted of predicting user's ratings while optimizing the Root Mean Square Error (RMSE) between the predicted and actual ratings.
Based on some specific application domains, we define more general categories of domains for the most common RS applications:
• **Entertainment –** RSs for *movies, music, games*...
• **Content -** newspapers, *documents*, web pages, e-learning applications, and email filters.
• **E-commerce -** products to buy such as books, computers and smartphones, PCs
etc. for different types of consumers.
• **Services -** *travel services*, experts for consultation, *houses to rent*, or *matchmaking services*.
• **Social -** people in *social networks*, content social media content such as tweets, Facebook news feeds, LinkedIn job updates, and others.

2.4. Recommendation Techniques

In order to implement its core function i.e. identifying useful items for the user, a RS must predict that an item is worth recommending. In order to do this, the system must be able to predict the utility of some items, or at least compare the utility of some items, and then decide which items to recommend based on this comparison. The prediction step may not be explicit in the recommendation algorithm but we can still apply this unifying model to describe the general role of an RS.

In General recommendations can be categorized into **personal** and **non personal** recommendations. Personal recommendations are offered as ranked list of personalized items. In the process of ranking, RSs predict what the most suitable items are based on the user's *preferences and constraints*, which are either explicitly expressed as ratings or inferred by interpreting actions of a user. Non personal recommendations are simpler to generate and are normally featured in magazines and newspapers (top n selections of books or CDs) and are not addressed by RS research.
The following diagram shows the different types of recommendation techniques.

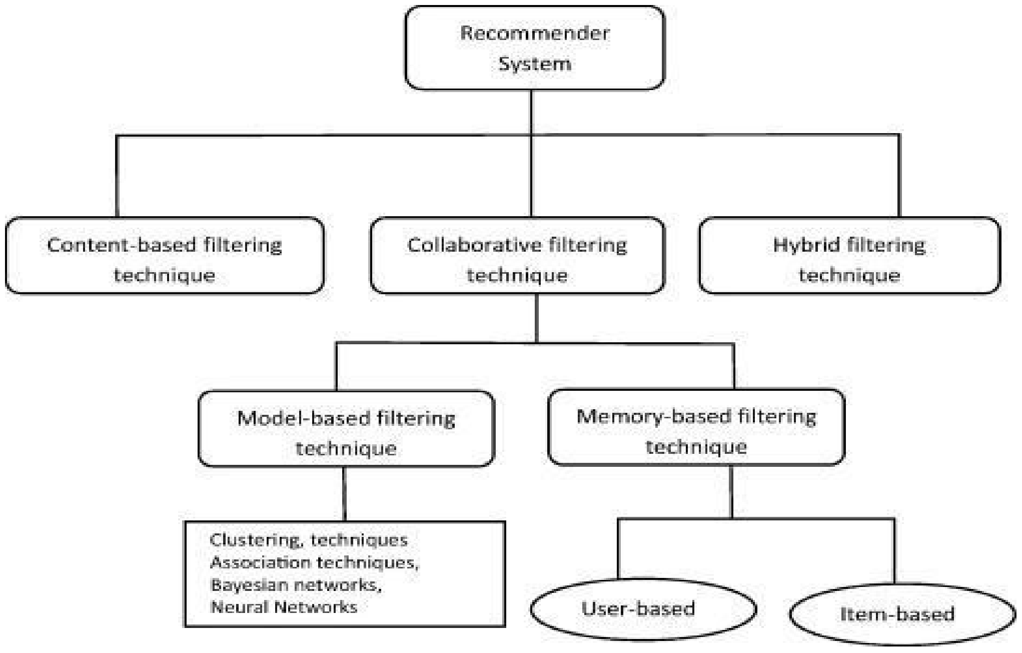

Recommender System
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based filtering technique
Memory-based filtering technique
Clustering, techniques Association techniques, Bayesian networks, Neural Networks
User-based
Item-based

Figure 1. Recommendation techniques (sciencedirect n.d.)

2.4.1. Collaborative Filtering (CF) Techniques

Collaborative filtering methods generate user specific recommendations of items based on previous ratings (interactions) without the need for extra information about items or users. In order to establish recommendations, CF techniques need to relate two fundamentally different entities: items (jobs in our case) and users. There are two main approaches to facilitate such a comparison, **The neighborhood approach** and **Latent factor models**.

1. The neighborhood approach focuses on relationships between
    - users (*user based recommendations*)
    - items (*item based recommendations*)
2. Latent factor models comprise an alternative approach by transforming both items and users to the same latent factor space. The latent space tries to explain ratings by characterizing both items and users on factors automatically extracted from user feedback. The widely known latent factor model is matrix factorization (aka, SVD).

The common problem faced when using CF techniques is that they are not able to deal with cold start situations, and they over-generalize i.e. they are not able to answer why certain list of items is being recommended to a user.

Most Popular CF Recommendation techniques

2.4.1.1. kNN (k Nearest Neighbors)

Recommender systems based on neighborhood automate the common principle that *similar users* prefer *similar items*, and similar items are preferred by similar users. kNN for Recommender Systems can generally be classified as **User-based kNN** and **Item-based kNN**.

User-based kNN

Let - **N(U)** - Neighbors of User U and

- $\mathbf{W_{uv}}$ - Similarity weight between users **U** and **V**

Problem: predict rating of user **U** for an item **i**, $\mathbf{r_{ui}}$,

Step 1. Calculate **similarity weight** among users, and build **user-similarity matrix.**

Step 2. Calculate **rating** of user **u** for item **i**,, $\mathbf{r_{ui}}$

Example: Given the rating matrix given below, to calculate predicted rating of *Eric* for the movie *Titanic*,

R[]	The matrix	***Titanic***	Die Hard	Forest Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	**?**	3	5	4
Diane	4	3	5	3	

Table 1 – User kNN example

Step 1. Calculate similarity weight among users, and build user-similarity matrix.
To calculate similarity weights we use the Pearson Correlation (PC) formula, i.e.

where:
Juv => Common set of Items rated by both user **u** and **v**

$\mathbf{r_{ui}}$ => rating of user **u** to item **I**

$\mathbf{r_{vi}}$=> rating of user **v** to item **I**

For instance to calculate similarity between John and Lucy,
$\mathbf{J_{JL}}$ = {The matrix, Titanic, Forest Gump, Wall-E} => common movies rated by John and lucy
$\mathbf{r_{J}}$ = (5+1+2+2) / 4 = **2.5 =>** mean value for john's ratings
$\mathbf{r_{L}}$ = (1+5+5+5) / 4 = **4.0 =>** mean value for lucy's ratings
Substituting these values in the above PC formula we get, **PC(J, L) = -0.938**. So we do the same for each user and build the similarity matrix shown below.

Step 2: Calculate Rating of Eric to Titanic, i.e. $\mathbf{r_{ET}}$.

$$\mathbf{r_{ET}} = \Big(\sum_{u \in N_i(E)} W_{EU}\, r_{UT}\Big) \Big/ \Big(\sum_{u \in N_i(E)} |W_{EU}|$$

Where
N_i(E) = k Nearest Neighbors of Eric, i.e. if k=2, N_1(E) = {**Lucy**} and N_2(E) = {**Diane**}
W_{EU} = Similarity of Eric with its neighborhood, i.e. from the user similarity
PC(Eric, Lucy) = 0.922 and PC(Eric, Diane) = -0.659.
$\mathbf{r_{UT}}$ **=** ratings of Eric's neighbors to Titanic, i.e. $\mathbf{r_{LT}}$ = 5 and $\mathbf{r_{DT}}$ = 3,
So, substituting this values in the above formula, We get

$\mathbf{r_{ET}}$ = (0.922 * 5) + (-0.659 * 3) / |0.922| + |-0.659|

$\mathbf{r_{ET}}$ = 4.61 + (-1.977) / 1.581

rET = <u>4.166</u>

Item-based kNN

Given the above example, to calculate predicted rating of *Eric* for the movie *Titanic:*

Step 1: Compute similarity among items and build item similarity matrix, using the Adjusted cosine formula, i.e.

$$AC(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_u)^2 \sum_{u \in \mathcal{U}_{ij}} (r_{uj} - \bar{r}_u)^2}}.$$

Where,

U_{ij} = Entire set of users who have rated for the movies i and j

r_{ui} = rating of user u (from set Uij) to movie i

r_{uj} = rating of user u (from set Uij) to movie j

$\bar{r}_u$ = **mean value** of ratings of user u

For instance to calculate the similarity between "The matrix" and "Titanic", we have

U_{MT} = {John, Lucy, Diane} => users who rated for both the matrix and titanic,

r	The **M**atrix	**T**itanic
John	5	1
Lucy	1	5
Diane	4	3

r_{mean} (John) = (5+1+2+2) / 4 = <u>2.5</u>

r_{mean} (Lucy) = (1+5+2+5+5) / 5 = <u>3.6</u>

r_{mean} (Diane) = (4+3+5+3) / 4 = <u>3.75</u>

So, Substituting, we get, AC(M, T) = **<u>-0.943</u>**

AC[]	**The matrix**	**Titanic**	**Die Hard**	**Forrest Gump**	**Wall-E**
The matrix	**1.000**	**-0.943**	0.882	-0.974	-0.977
Titanic	-0.943	**1.000**	-0.625	**0.931**	**0.994**
Die Hard	0.882	-0.625	**1.000**	-0.804	-1.000
Forrest Gump	-0.974	0.931	-0.804	**1.000**	0.930
Wall-E	-0.977	0.994	-1.000	0.930	**1.000**

Table 2 – Item kNN - Adjusted cosine

Step 2: Compute predicted ratings of Eric for Titanic,

$$\mathbf{r}_{ET} = (\sum_{I \in Ni(T)} \mathbf{W}_{TI}\, \mathbf{r}_{IE}) \,/\, (\sum_{I \in Ni(T)} |\mathbf{r}_{TI}|)$$

Where

N_i(T) = k Nearest Neighbors of *Titanic*, i.e. if k=2, N_1(T) = {**Forest Gump**} and N_2(T) = {**Wall-E**}

$\mathbf{W}_{TI}$ = Similarity of Titanic with its neighborhood, i.e. from the item similarity
AC(Titanic, Forest Gump) = 0.931 and **AC**(Titanic, Wall-E) = 0.994.

$\mathbf{r}_{IE}$ = Eric’s ratings to Titanic’s neighbors $\mathbf{r}_{FE}$ = 5 and $\mathbf{r}_{WE}$ = 3.

So, substituting this values in the above formula, We get

$\mathbf{r}_{ET}$ = (0.931 * 5) + (0.994 * 3) / (|0.931 + 0.994|) =

$\mathbf{r}_{ET}$ = 8.631 / 1.925

$\mathbf{r}_{ET}$ = <u>4.483.</u>

2.4.1.1. MATRIX FACTORIZATION (MF)

Basically to factorize a matrix means finding two or more matrices such that when multiplied we get back the original matrix. Matrix factorization is the general term used for models that are induced by factorization of the user-item rating matrix (aka SVD based models). Of all the several MF techniques in this section we'll see how the **SVD model** works along with a working example.

MF models map both users and items to a joint latent factor space of dimensionality **f** , such that user-item interactions are modeled as *inner products* in that space. The latent space tries to explain ratings by characterizing both products and users on factors automatically extracted from user feedback. For example, when the items are movies, factors might measure latent dimensions such as *comedy* vs. *drama*, *amount of action*, or *orientation to people with ages in range*; less well defined dimensions such as depth of character development or peculiar; or completely non interpretable dimensions. (Recommender systems handbook – second edition n.d.)
Given **K** as number of latent factors and **R[i*j]** as a sparse matrix of size **i** by **j**, the task is to fill the empty holes in the matrix (unrated values)

<u>SVD Example</u>

Rating matrix **R**[]	**I_1**	**I_2**	**I_3**	**I_4**
U_1	5	3	-	1
U_2	4	-	-	1
U_3	1	1	-	5
U_4	1	-	-	4
U_5	-	1	5	4

Table 3 – Matrix factorization example (quuxlabs n.d.)

Step 1. Randomly initialize **P**[i*K] and **Q**[j*K] where **K** = 2 such that **R** ~ $\mathbf{PQ}^{T}$ = **R' :**

P =

0.50	0.39
0.73	0.70
0.58	0.21
0.61	0.24
0.79	0.58

Q =

0.51	0.59
0.90	0.08
0.82	0.43
0.2	0.53

Where each row of P represents the strength of associations between a *user* and it's *features*, similarly each row of Q represents the strength of associations between an *item* and it's *features*.

Step 2: To get predictions of a user $\mathbf{u_i}$ for an item $\mathbf{i_j}$, we calculate the ***dot product*** of the two vectors corresponding to $\mathbf{u_i}$ and $\mathbf{i_j}$.

$$\mathbf{r'_{ij} = P_i^T q_j} = \sum\nolimits^{k}_{k=1} \mathbf{p_{ik}\ q_{kj}}$$

$\mathbf{p_{ik}}$ =

0.50	0.39
0.73	0.70
0.58	0.21
0.61	0.24
0.79	0.58

$\mathbf{q_{kj}}$ =

0.51	0.90	0.82	0.22
0.59	0.08	0.43	0.53

$\mathbf{r'_{ij}}$ =

0.48	0.48	0.57	0.31
0.78	0.71	0.89	0.53
0.41	0.53	0.56	0.23
0.45	0 19.08..56	0.60	0.26
0.74	0.75	0.89	0.48

Step 3: Minimize the difference between the original rating $\mathbf{r_{ij}}$ and estimated rating $\mathbf{r'_{ij}}$ using the ***Gradient descent algorithm***, which aims at finding the *local minimum* of the difference.

$e_{ij}^2 = (r_{ij} - r'_{ij})^2 = (r_{ij} - \sum^{k}_{k=1} p_{ik}\, q_{kj})^2 =$

4.52	2.52	0.57	0.69
3.21	0.71	0.89	0.53
0.41	0.53	0.56	0.23
0.45	0.56	0.60	0.26
0.74	0.75	0.89	0.48

At every iteration, to minimize the error we have to know in which direction we have to modify the values of p_{ik} and q_{kj}.

Step 4: Calculate the *gradient* at the current values, To do so we *differentiate* the above equation with respect to these two variables separately:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj}$$
$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{ik}$$

Step 5: Having obtained the gradient, we formulate *update rules* for $\mathbf{p_{ik}}$ and $\mathbf{q_{kj}}$

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij} q_{kj}$$
$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij} p_{ik}$$

Here, alpha is a constant whose value determines the rate of approaching the minimum. Usually we will choose a small value for alpha, say **0.0002**. This is because if we make too large a step towards the minimum we may run into the risk of *missing the minimum* and end up oscillating around the minimum. Now let’s calculate the new p_{ik} and q_{kj} with the above equation.

For instance, the first iteration of the gradient descent will be computed as follows:

iteration 1:

- i=0 and j=0, k=0
  - ✓ P00 = P00 + 0.0004 * (e00 * q00) = 0.50 + (0.0004*4.52*0.51) = **0.5009**
  - ✓ q00 = q00 + 0.0004 * (e00 * p00) = 0.51 + (0.0004*4.52***0.5009**) = **0.5109**
- i=0 and j=0, k=1
  - ✓ P01 = P01 + 0.0004 * (e00 * q10) = 0.39 + (0.0004*4.52*0.59) = **0.3910**
  - ✓ q10 = q00 + 0.0004 * (e00 * p01) = 0.59 + (0.0004*4.52***0.3910**) = **0.5907**
- i=0 and j=1, k=0
  - ✓ P00 = P00 + 0.0004 * (e01 * q01) = **0.5009** + (0.0004*2.52*0.59) = **0.5018**
  - ✓ q01 = q01 + 0.0004 * (e01 * p00) = 0.90 + (0.0004*2.52***0.5018**) = **0.9005**
- i=0 and j=1, k=1
  - ✓ P01 = P01 + 0.0004 * (e01 * q11) = **0.3910** + (0.0004*2.52*0.08) = **0.3911**
  - ✓ q11 = q11 + 0.0004 * (e01 * p01) = 0.08+ (0.0004*2.52***0.3911**) = **0.0803**

We will keep computing Pik and qkj for each i, j and k. Finally we will again take the dot product of the resulting P and Q and repeat every thing from step 3.

Pseudo-code (MF)

```
for step in NumberOfSteps
    for i in length( R )
        for j in length(R[i])
            e[i][j] = R[i][j] - R'[i][j]
            for k in K
                P[i][k] = P[i][k] + 2*alpha*e[i][j]*q[k][j]
                q[k][j] = q[k][j] + 2*alpha*e[i][j]*p[i][k]
    R' = dot product(P,q)
    for i in length( R )
        for j in length(R[i])
            e = e + (R[i][j] - R' )^2
    if e < threshold
        Exit()
```

Keep iterating until the specified number of steps to find the local minimize of the overall error **e**, or stop when the error reaches a certain threshold.

2.4.2. Content based Techniques

Content based methods learn to recommend items that are similar to the ones that user liked in the past. The similarity of items is calculated based on the features associated with the compared items, i.e. two items are similar if they have similar attributes. For example if a user has positively interacted with a job posting (item) that belongs to an Information technology industry, then the system can learn to recommend other job postings from this industry.
Classic Content based techniques aim at matching the attributes of the user profile against attributes of items. In most cases the items' attributes are simply keywords that are extracted from the items' descriptions.
The following example illustrates a running example of how Content based technique makes recommendations.

	Item/job ads	**HR**	**Consulting**	**Engineering**	**Development / programming**
I_1	*Full stack developer*			✓	✓
I_2	*Front end developer*			✓	✓
I_3	*Software Engineer*			✓	✓
I_4	*HR specialist*	✓			
I_5	*Senior java developer*		✓		✓

Table 4 – Content based recommendation example – Items and their categories

Suppose we have the Rating matrix, rated by the 5 users.

R	**U_1**	**U_2**	**U_3**	**U_4**	**U_5**
I_1	5.0			4.0	5.0
I_2	4.0			5.0	4.0
I_3			5.0		
I_4		5.0			
I_5	4.0		4.0	5.0	4.0

Table 5 – Content based recommendation – Rating matrix

From this observation data, we derive the users' degree of interest in categories (number of times users accessed items related to categories) as follows,

	U_1	**U_2**	**U_3**	**U_4**	**U_5**
HR		1			
Consulting	1		1	1	1
Engineering	2		1	2	2
Development	2		1	3	3

Table 6 – Content based recommendation – Users' degree of interest

For example if we want to make recommendations to U5, we will calculate the similarity of U5 with every item, as follows,

	HR	Consulting	Engineering	Development	**Sim(U_5,I_i)**
I_1			✓	✓	(2*3)/(3+2)=**4/5**
I_2			✓	✓	(2*3)/(3+2)=**4/5**
I_3			✓	✓	(2*3)/(3+2)=**4/5**
I_4	✓				**0**
I_5		✓		✓	(2*3)/(3+2)=**4/5**

Table 7– Content based recommendation – target user-item similarity

From this similarity matrix, we recommend to user 5 the items with highest similarity i.e. Recommendation (**U_5**) = > **Max(Similarity(U_5,I_i)) => {I_1, I_2, I_3, I_5}**.

2.4.3. Hybrid Techniques

Hybrid methods combine the characteristics of techniques A and B and use the advantages technique A to overcome the limitations of technique B.

They can be combined in various ways, for instance:

- *merging their individual predictions into a single* and more robust prediction or
- *adding content information into a collaborative filtering model*.

  CF methods suffer from new-item problems, or, that they cannot recommend items that have no ratings. This does not limit content-based approaches since the prediction for new items is based on their description (features) that are typically easily available.

2.4.3.1. How to implement hybrid RSs

In this section, How hybrid recommender systems are implemented and the different approaches of hybridization will be discussed in detail.

The most successful and commonly used approach is to combine CF and Content based techniques. These RSs are based on the combination of the above mentioned techniques. A hybrid system combining techniques A and B tries to use the advantages of A to fix the disadvantages of B. For instance, CF methods suffer from new-item problems, or, that they cannot recommend items that have no ratings. This does not limit content-based approaches since the prediction for new items is based on their description (features) that are typically easily available. Given two (or more) basic RSs techniques, several ways have been proposed for combining them to create a new hybrid system.
The context of the user when he or she is seeking a recommendation can be used to better personalize the output of the system. For example, in a temporal context, vacation recommendations in winter should be very different from those provided in summer. Or a restaurant recommendation for a Saturday evening with one’s friends should be different from that suggested for a workday lunch with co-workers.

2.4.3.2. Classes of Hybridization

2.4.3.2.1. Parallelized hybridization

In this scheme, the output of several existing recommendation techniques are combined and used in parallel. It uses some kind of *weighting* or *voting scheme* to evaluate each technique where the weights are learned dynamically on the go and then used to select the technique with better result.

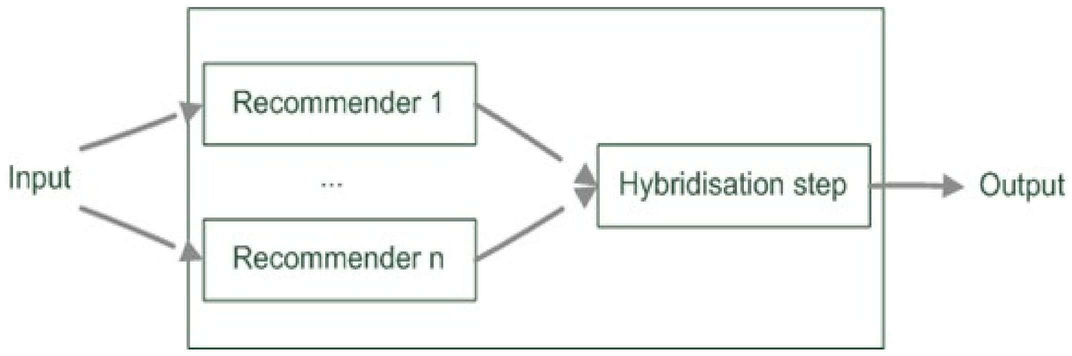
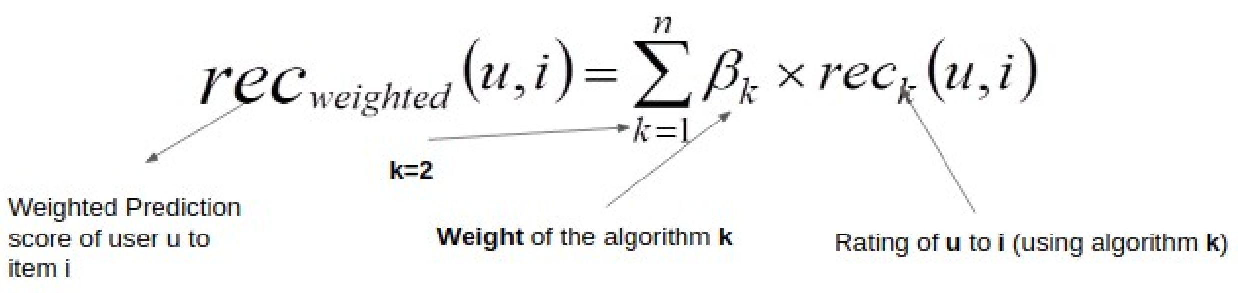


Figure 2. Parallelized hybridization design (Hybrid recommendation approaches n.d.)

There are three approaches for Parallelized hybridization: Weighted, Switching and Mixed. The next section explains each one in detail.

Weighted

Computes the scores (weights) of the items to recommend by *aggregating* the output *scores* of each recommendation technique using *weighted linear functions*. The weights of **CF** and **CBF** are set on a per-user basis enabling the system to determine the *optimal mix* for each user.



$$rec_{weighted}(u,i) = \sum_{k=1}^{n} \beta_k \times rec_k(u,i)$$

k=2
Weighted Prediction score of user u to item i
Weight of the algorithm **k**
Rating of **u** to **i** (using algorithm **k**)

<u>How to decide the **weights of the algorithms**?</u>
1. Try different random weights to the 2 algorithms (***beta1*** and ***beta2***) such that *beta1* +*beta2* = 1

2. Calculate Mean Absolute Error (**MAE**) of the linear function using beta1 and beta2 in the previous formula.

3. Choose the weights (beta1 and beta2) that minimizes the MAE.

Switching

In this technique, the system switches between different recommendation techniques according to some criteria. The criteria can be for example when there are few ratings in the system. Different versions of the same basic strategy (e.g. CBF1-CBF2 or CF1-CF2) can be integrated in a switching form. For instance a CF - CBF recommender can switch to CBF only when the CF strategy doesn't provide enough credible recommendations.

<u>**Example:**</u> The Daily-learner system uses a CF - CBF hybrid in which a CBF recommendation technique is employed first. If the CBF system cannot make a recommendation with sufficient confidence, then a CF recommendation is attempted.

Mixed

Results of different recommendation techniques are *presented* together. Recommendation result for user **u and** item **i** is the set of tuples **<score,k>** for each of its n constituting recommenders **rec$_{\mathbf{k}}$.**

$$rec_{mixed} = \overset{n}{\underset{k=1}{}} \left\langle rec_k(u,i), k \right\rangle$$

2.4.3.2.2. Monolithic Hybridization

This scheme technically has only a single recommendation component. The hybridization step is virtual in a sense that feature sources of different recommendation paradigms are combined.

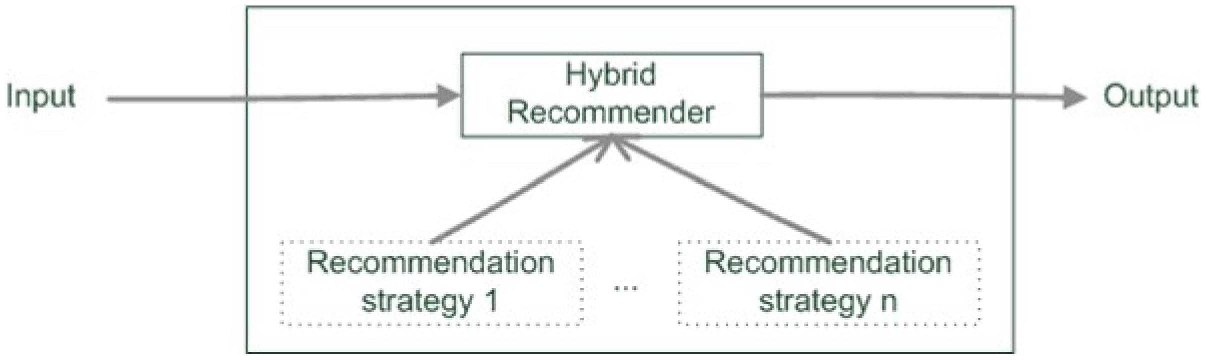


Figure 3. Monolithic hybridization design(Hybrid recommendation approaches n.d.)

There are two approaches for Monolithic hybridization: **Feature combination** and **Feature Augmentation**. The next section explains each one in detail.

Feature Combination

Ratings + **Explicit features** are used for similarity computation. Then one recommender's output is used as additional *feature data* to the other recommender. It is order sensitive.

The features can be of:

- Social features: Movies liked by a user.
- Content features: *Comedies* liked by user, *dramas* liked by a user.
- Hybrid features: user likes many movies that are comedies.

Feature Augmentation

One technique is employed to produce a rating or classification of an item and that information is then incorporated into the processing of the next recommendation technique. Note that this is different from feature combination in which raw data from different sources is combined.
For example, the Libra system (Mooney & Roy 1999) makes content-based recommendations of books based on data found in Amazon, using a naive Bayes text classifier. In the text data used by the system is included "related authors" and "related titles" information that Amazon generates using its internal collaborative systems. These features were found to make a significant contribution to the quality of recommendations. Augmentation is attractive because it offers a way to improve the performance of a core system, like the NetPerceptions' GroupLens Recommendation Engine or a naive Bayes text classifier, without modifying it. Additional functionality is added by intermediaries who can use other techniques to augment the data itself.

2.4.3.2.3. Pipe-lined Hybridization

One recommender system pre-processes some input for the subsequent one.

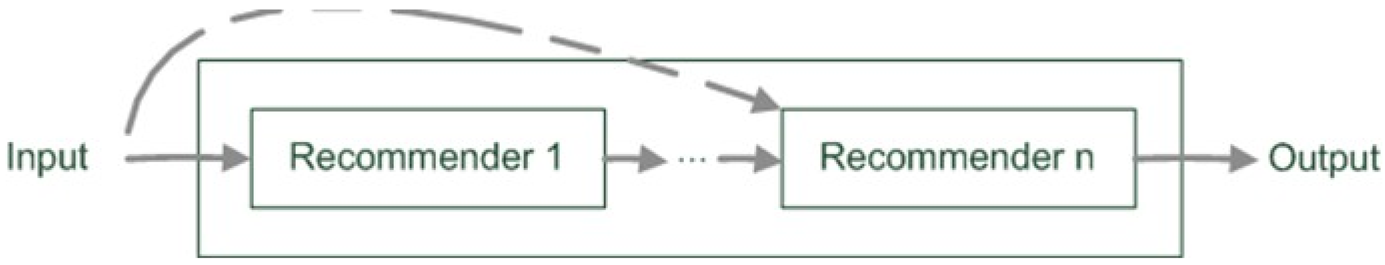


Figure 4. Pipe-lined hybridization design(Hybrid recommendation approaches n.d.)

There are two approaches for Pipelined hybridization: **Cascade** and **Meta-level**.

Cascade

Successor's recommendations are *restricted* by the predecessor.

$$rec_{cascade}(u,i) = rec_n(u,i)$$

where for all k > 1,

$$rec_k(u,i) = \begin{cases} rec_k(u,i) & : rec_{k-1}(u,i) \neq 0 \\ 0 & : otherwise \end{cases}$$

May not introduce additional items, but produce very precise results.

Meta-level

Successor exploits a model delta built by predecessor.

$$rec_{meta-level}(u,i) = rec_n(u,i,\Delta_{rec_{n-1}})$$

Example:

- Using **CBF** recommenders to build *item representation models*, and then employ this models in **CF** recommenders to match the items with user profiles.

2.4.4. **FACTORIZATION MACHINES (FM)**

Support Vector Machines (SVMs) are one of the most popular predictors in machine learning and data mining. Nevertheless in settings like collaborative filtering, SVMs play no important role and the best models are either direct applications of standard matrix/ tensor factorization models like PARAFAC. (Factorization Machines (n.d).)

FM models are at the cutting edge of Machine Learning techniques for personalization; they have proven to be an extremely powerful tool with enough expressive capacity to generalize methods such as Matrix/Tensor Factorization and Polynomial Kernel regression.

Why Factorization machines?

The following are the 3 major advantages of FMs:

1. FMs allow *parameter estimation under very sparse data* where SVMs fail.

2. FMs have *linear complexity*. So can scale to large datasets like Netflix with 100 millions of training instances.

3. FMs are general predictors that can work with any *real valued feature vectors*. (Factorization Machines (n.d).)

2.4.4.1. Model Equation

The model equation for FM of degree d = 2 is defined as follows:

$$\hat{y}(x) = w_0 + \sum_{i=i}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

(Factorization Machines (n.d.))

where the model parameters to be learned during the training phase are

W_0 – the global bias

W – a vector of size **n** containing the weights of each feature **X_i**

V – models the interaction between the ***i^{th}*** and **j^{th}** variables.

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$$

(Factorization Machines (n.d.))

A row in V represents the i^{th} variable with k factors.

The complexity of the straight forward computation of the above Factorization machines model equation is **O(kn^2)**, because all the pairwise interactions have to be computed.

But we can reformulate the model equation to be computed in linear time with complexity of **O(kn).**

Due to factorization of pairwise interactions, there's no model parameter that directly depends on the parameters i and j, so the pairwise interaction can be reformulated as.

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=i+1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j\rangle\, x_i\, x_j \\
=&\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j\rangle\, x_i\, x_j - \frac{1}{2}\sum_{i=1}^{n}\langle \mathbf{v}_i, \mathbf{v}_i\rangle\, x_i\, x_i \\
=&\frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f=1}^{k} v_{i,f}\, v_{j,f}\, x_i\, x_j - \sum_{i=1}^{n}\sum_{f=1}^{k} v_{i,f}\, v_{i,f}\, x_i\, x_i\right) \\
=&\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i\right)\left(\sum_{j=1}^{n} v_{j,f}\, x_j\right) - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2\right) \\
=&\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i\right)^2 - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2\right)
\end{aligned}
$$

(Factorization Machines (n.d.))

This reformulation makes the equation have linear complexity in both k and n, O(kn).

Why binary classification?

In our case, the target **Y** is “interaction type” provided in the RecSys dataset. “Interaction type” indicates the type of interaction a user performed on the item(job).

- 0 => **impression**: XING showed this item to a user.
- 1 => **click**: the user clicked on the item
- 2 => **bookmark**: the user bookmarked the item on XING
- 3 => the user clicked on the *reply button* or *application form button* that is shown on some job postings
- 4 => the user **deleted** a recommendation from his/her list of recommendation (clicking on "x") which has the effect that the recommendation will no longer been shown to the user and that a new recommendation item will be loaded and displayed to the user
- 5 => a recruiter from the items company showed interest into the user. (e.g. clicked on the profile)

We have implemented the Job recommendation as a **classification problem** where the Interactions of type **0**, **1, 2** and **5** are considered as *positive interactions* and interaction type **4** is considered as a *negative interaction*. As a result the target column will be replaced with a 0 or 1 accordingly.

The Factorization machines model can estimate parameters even if we have a very sparse interaction matrix, because it breaks the independence of the interaction parameters by factorizing them. In general It means that the data for one interaction helps also to estimate the parameters for related interatctions.

User				Jobs(items)				User Features				Item Features				Interaction date	Target (Interaction type)
1	0	0	…	1	0	0	…									DD/MM/YY	4 (0)
1	0	0	…	0	1	0	…									DD/MM/YY	3 (1)
0	1	0	…	0	1	0	…									DD/MM/YY	1 (1)
0	1	0		1	0	0										DD/MM/YY	1 (1)

Table 8 – Structure of Interaction data with features

2.4.4.2. Algorithm Explained with example

User (U)	**Job (I)**	**career_level_user (clu)**	**discipline_id_user (du)**	**career_level_item (cli)**	**discipline_id_item (di)**	**experience(ex)**
Alice(A)	J1	professional(3)	Engineering (6)	professional (3)	consulting(5)	5
Alice(A)	J2	professional(3)	Engineering (6)	professional (3)	consulting(5)	5
Alice(A)	J3	professional(3)	Engineering (6)	professional (3)	consulting(5)	5
Bob(B)	J3	manager(2)	HR(4)	professional (3)	HR(4)	7
Bob(B)	J4	manager(2)	HR(4)	professional (3)	consulting(5)	3
Charlie(C)	J1	Beginner(1)	Engineering (6)	professional (3)	consulting(5)	2
Charlie(C)	J3	Beginner(1)	Engineering (6)	professional (3)	consulting(5)	5

Table 9 – Factorization Machines example

Assume we have the above interaction data in our RecSys dataset, and we want to predict if Alice(A) will positively interact with Job 4(J4). We will first do a one-hot encoding on the interaction data which will result in the following data.

A	B	C	J1	J2	J3	J4	Clu	Du	Cli	Di	Ex	Target(Y)
1	0	0	**1**	0	0	0	3	6	3	5	5	1
1	0	0	0	**1**	0	0	3	6	3	5	5	1
1	0	0	0	0	**1**	0	3	6	3	5	5	1
0	**1**	0	0	0	**1**	0	2	4	3	4	7	1
0	**1**	0	0	0	0	**1**	2	4	3	5	3	1
0	0	**1**	**1**	0	0	0	1	6	3	5	2	1
0	0	**1**	0	0	**1**	0	1	6	3	5	5	1

Table 10 - Interaction data one hot encoding

There is no case x in the training data where both variables X_A and X_{J4} are non-zero and thus a direct estimate would lead to no interaction ($W_{A,J4} = 0$). But with factorized interaction parameters **$\langle V_A, V_{J4} \rangle$** we can estimate the interaction even in this case.

Bob and Charlie will have similar factor vectors V_B and V_C, because they both have interactions with Job 3 (J3) for predicting interactions i.e. **$\langle V_B, V_{J4} \rangle$** and **$\langle V_C, V_{J4} \rangle$** have to be similar.

Alice will have a different factor vector from Charlie because she has a different interactions with the factors of Job 1(J1) and Job 3(J3).

$$Y'(x) = w_0 + W_{clu} X_{clu} + W_{du} X_{du} + W_{cli} X_{cli} + W_{di} X_{di} + W_{ex} X_{ex} +$$

$$\langle V_A, V_{J1} \rangle X_A X_{J1} + \langle V_A, V_{J2} \rangle X_A X_{J2} + \langle V_A, V_{J3} \rangle X_A X_{J3} + \langle V_A, V_{J4} \rangle X_A X_{J4} +$$

$$\langle V_B, V_{J3} \rangle X_B X_{J3} + \langle V_B, V_{J4} \rangle X_B X_{J4} +$$

$$\langle V_C, V_{J1} \rangle X_C X_{J1} + \langle V_C, V_{J3} \rangle X_C X_{J3}$$

The 3 variables W0, Wi and V (marked with green)will be learned during the training phase with gradient descent method, as the one discussed in the previous chapter section 2.4.1.1.

The gradient of the FM model is:

$$\frac{\partial}{\partial\theta}\hat{y}(\mathbf{x}) = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2, & \text{if } \theta \text{ is } v_{i,f} \end{cases}$$

Since our prediction task is binary classification, after learning all the parameters during the training phase and get the final value of Y’(X), the sign of Y’(X) will be used i.e. if it’s negative, it implies a negative interaction and if it’s positive, it implies a positive interaction.

2.4.4.3. Features to be fed to the FM Model

<u>User Features</u>

Career level *	0 => unknown •1 => Student/Intern •2 => Entry Level (Beginner) •3 => Professional/Experienced •4 => Manager (Manager/Supervisor) •5 => Executive (VP, SVP, etc.) •6 => Senior Executive (CEO, CFO, President)
Discipline Id *	Anonymized IDs representing disciplines such as “consulting”, “HR” etc...
Industry Id *	Anonymized IDs representing industries such as “Internet”, “Automotive”, “Finance”
Premium	Implies whether the user is subscribed to XING’s premium membership. 0=>no subscription 1=>active subscription
Experience_years_experience*	It is the estimated number of years of work experience that the user has •0 = unknown •1 = less than 1 year •2 = 1 - 3 years •3 = 3 - 5 years •4 = 5 - 10 years •5 = 10 - 20 years •6 = more than 20 years
edu_degree *	estimated university degree of the user • 0 or NULL = unknown • 1 = bachelor • 2 = master • 3 = phd
edu_fieldsofstudy *	Fields of study of the user

Table 11 – User Features

Item Features

Career level *	• 0 => unknown •1 => Student/Intern •2 => Entry Level (Beginner) •3 => Professional/Experienced •4 => Manager (Manager/Supervisor) •5 => Executive (VP, SVP, etc.) •6 => Senior Executive (CEO, CFO, President)
Discipline Id *	Anonymized IDs representing disciplines such as “consulting”, “HR” etc...
Industry Id *	Anonymized IDs representing industries such as “Internet”, “Automotive”, “Finance”
is_paid	indicates that the posting is a paid for by a company
employment	The type of employment • 0 => unknown • 1 => full-time • 2 => part-time • 3 => freelancer • 4 => intern • 5 => voluntary

Table 12 – Job features

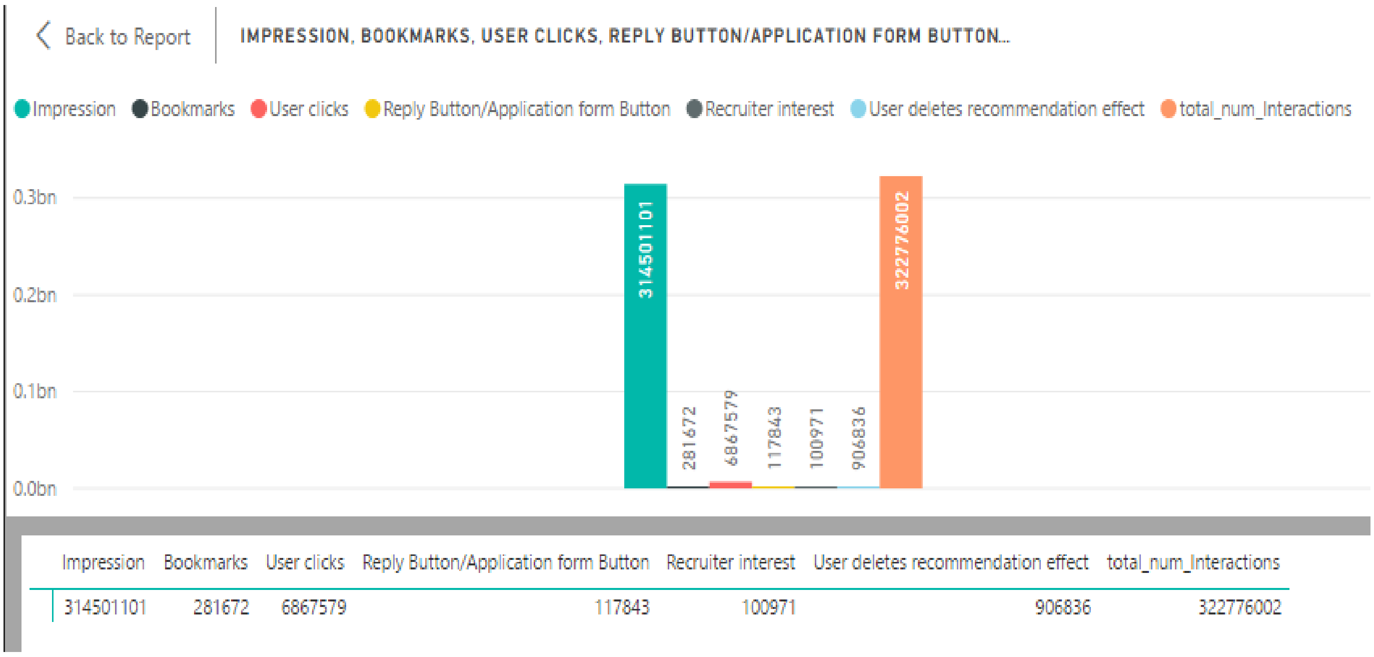
CHAPTER 4.

4. THE RECSYS CHALLENGE 2017 DATASET

In this section, different statistics information on the RecSys dataset will be elaborated with figures.

4.1. Interactions

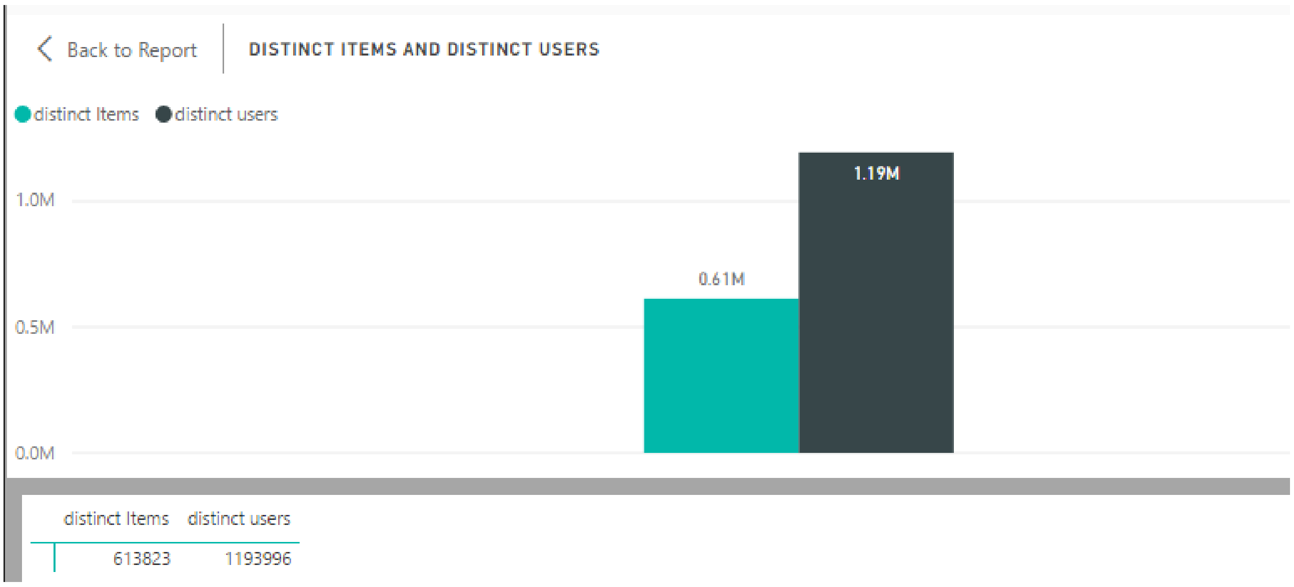
We have about 0.3 billion interactions, most of which are of type ‘impression’ as depicted below.



Impression	Bookmarks	User clicks	Reply Button/Application form Button	Recruiter interest	User deletes recommendation effect	total_num_Interactions
314501101	281672	6867579	117843	100971	906836	322776002

Figure 5 - Interactions type statistics

We have about 613 million unique items and 1 billion unique users in the given interaction data.

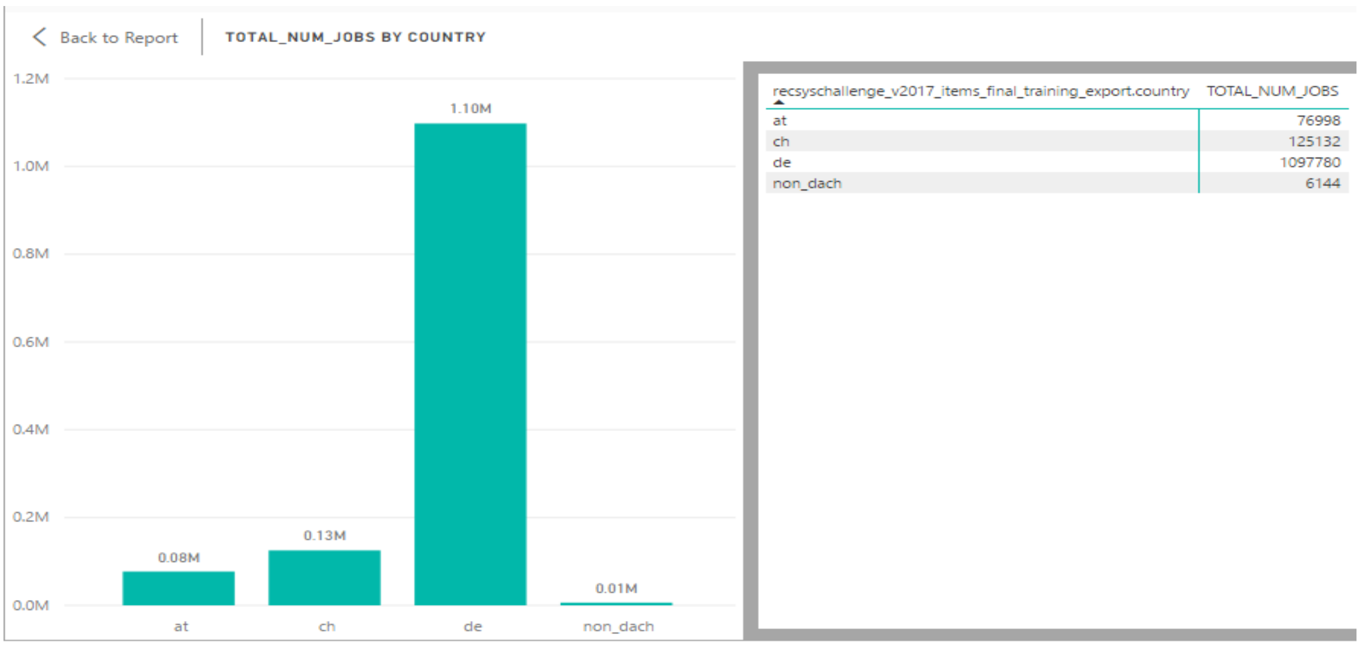


distinct Items	distinct users
613823	1193996

Figure 6 - Distinct Items and Users in Interactions data

4.2. Items (Jobs)

As seen below on the diagram, most of the job postings we have are from Germany, followed by Switzerland and Austria and other countries.



recsyschallenge_v2017_items_final_training_export.country	TOTAL_NUM_JOBS
at	76998
ch	125132
de	1097780
non_dach	6144

Figure 7 - Total number of Jobs by country

Almost 80% of the job postings are of career level 3 i.e. Professional/Experienced levels of postings, then follow Student/Intern and Management and Entry level postings.

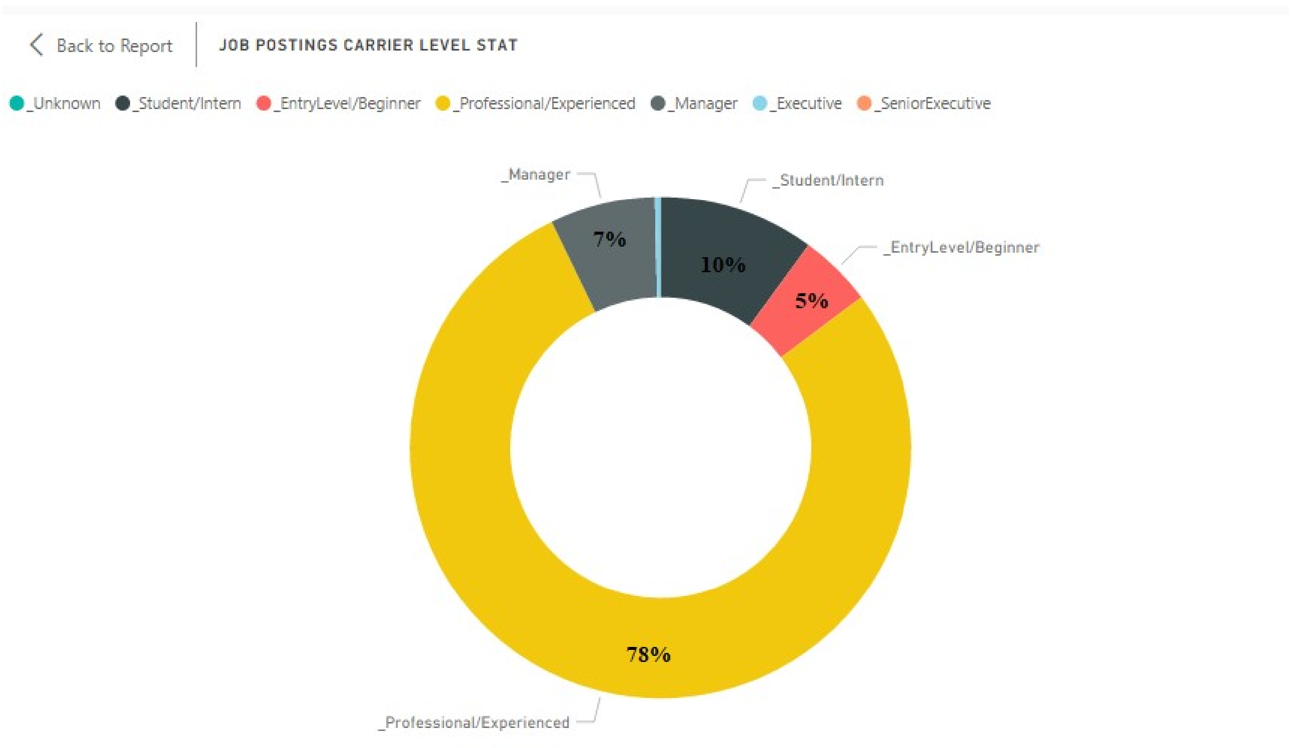


Figure 8 - Jobs career level

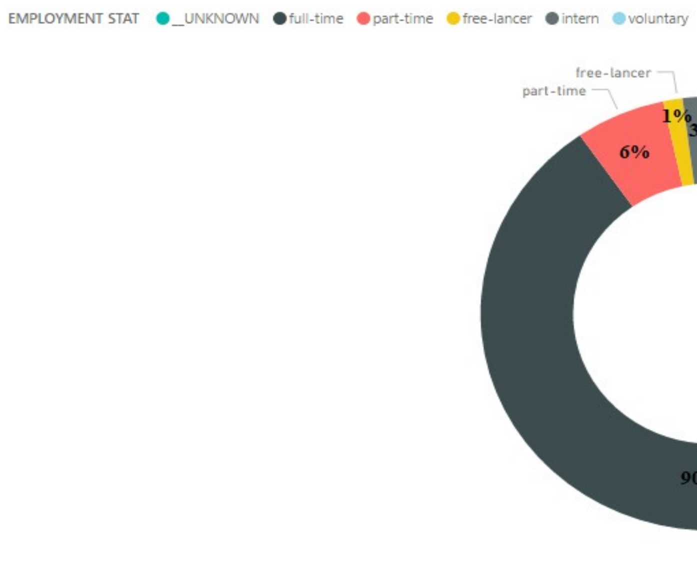
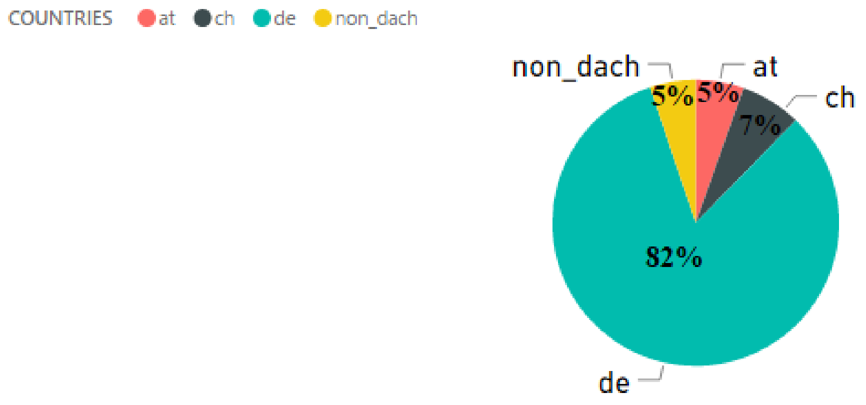
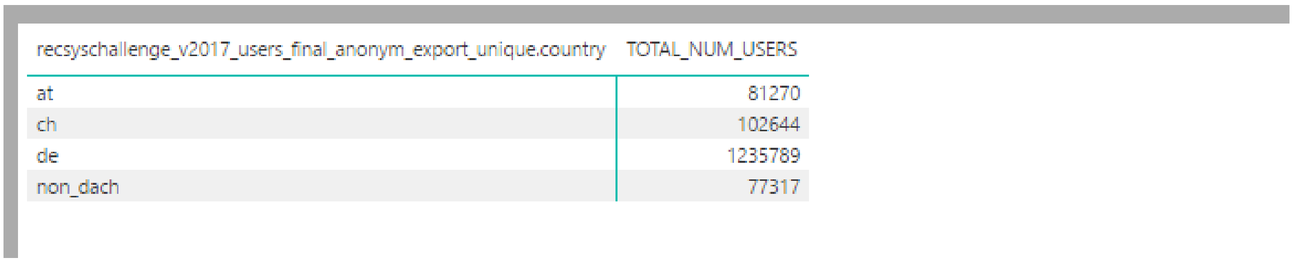


Figure 9 - Jobs' Employment type statistics

As seen above, Most of the Job postings are full type jobs.

4.3. Users

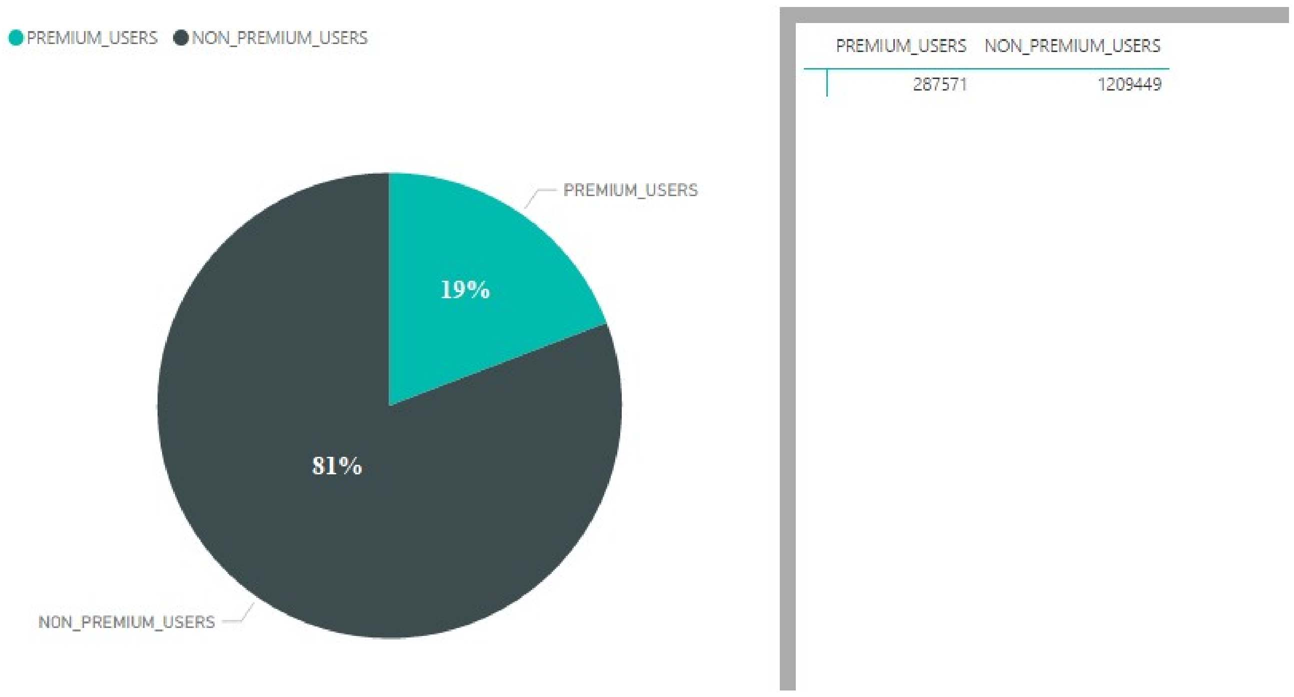




recsyschallenge_v2017_users_final_anonym_export_unique.country	TOTAL_NUM_USERS
at	81270
ch	102644
de	1235789
non_dach	77317

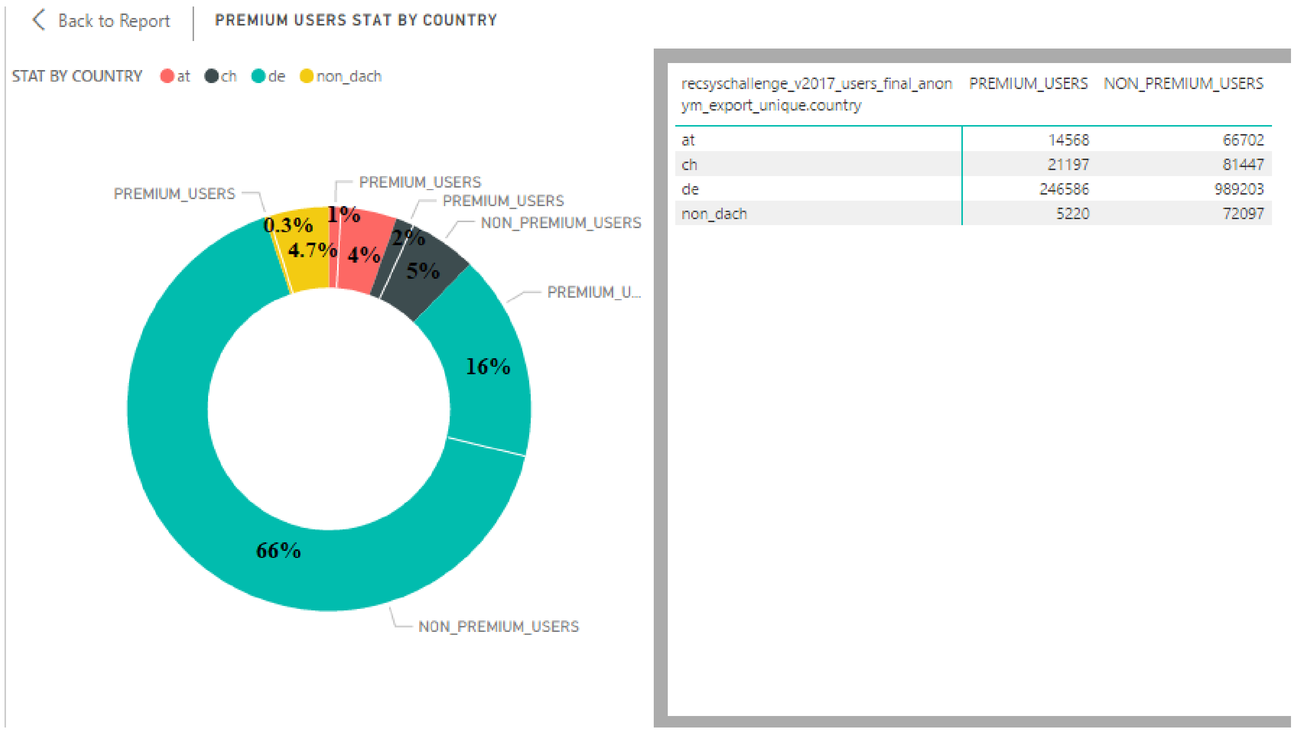
Figure 10 - Total number of users by country

Most of the users we have are from Germany. With more than 80% being non-premium users.



PREMIUM_USERS	NON_PREMIUM_USERS
287571	1209449

Figure 10 - Premium users



recsyschallenge_v2017_users_final_anonym_export_unique.country	PREMIUM_USERS	NON_PREMIUM_USERS
at	14568	66702
ch	21197	81447
de	246586	989203
non_dach	5220	72097

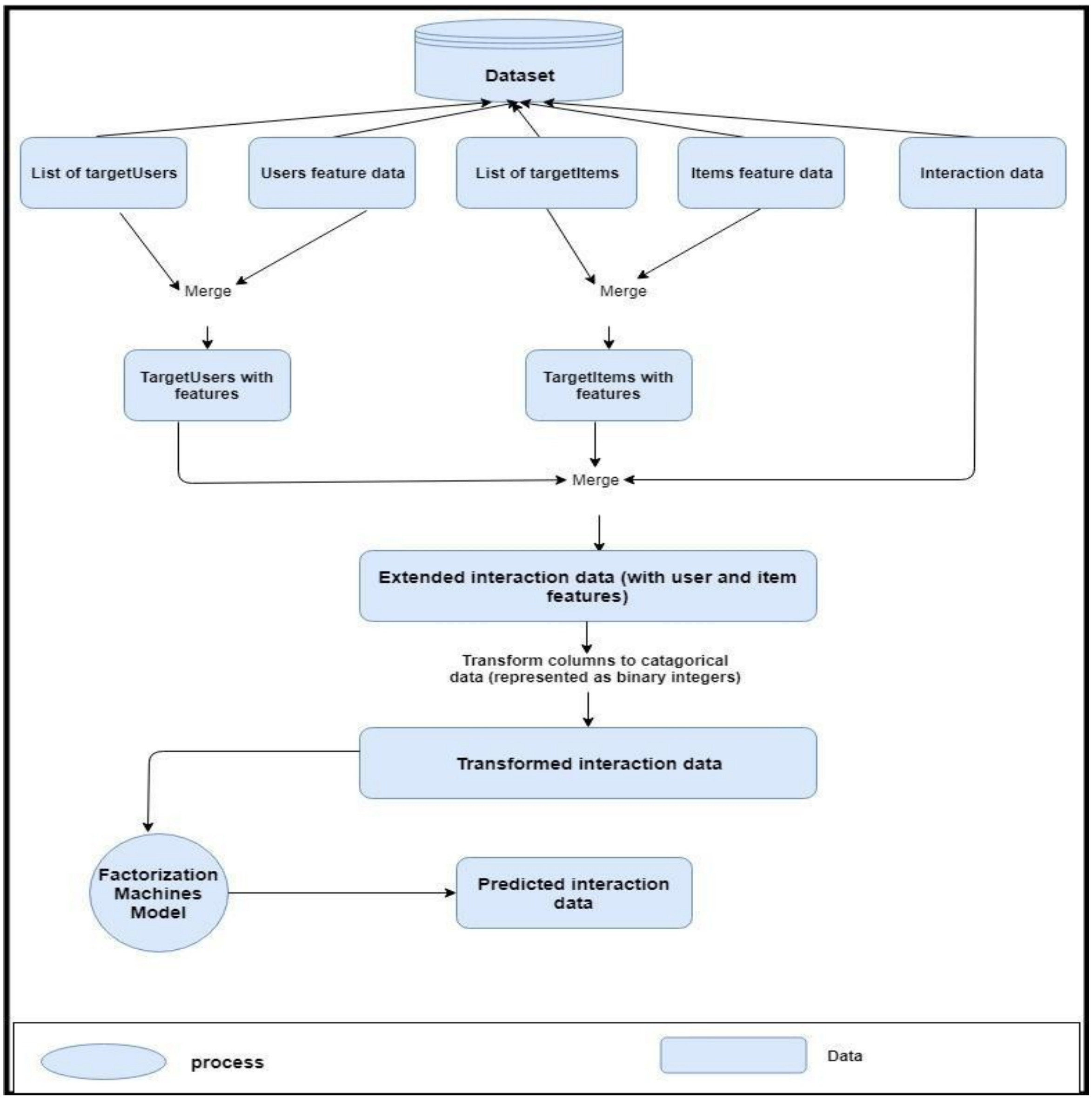
Figure 11 - Premium users by country

Chapter 5.

5. MODEL DESIGN

In this section, the architecture of the proposed recommender system and how the algorithm makes recommendations will be illustrated with diagrams.

5.1. SYSTEM ARCHITECTURE



Dataset
List of targetUsers
Users feature data
List of targetItems
Items feature data
Interaction data
Merge
Merge
TargetUsers with features
TargetItems with features
Merge
Extended interaction data (with user and item features)
Transform columns to catagorical data (represented as binary integers)
Transformed interaction data
Factorization Machines Model
Predicted interaction data
process
Data

Figure 12 – High level system architecture

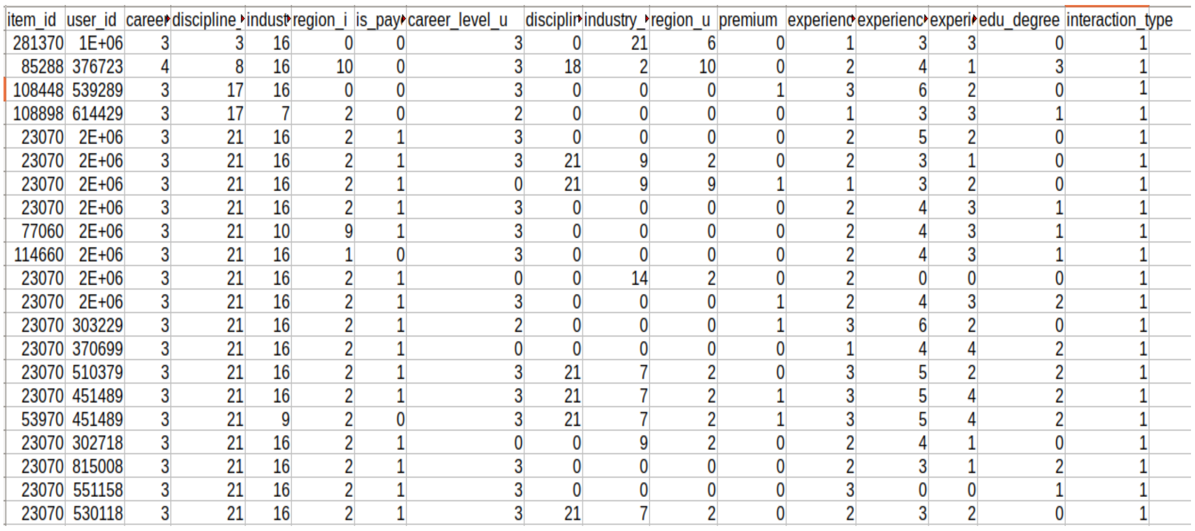
First, the Dataset contains users, Items(Jobs), Target users, Target items and Interactions. To finally obtain the extended Interaction data with user and item features, we follow the following steps.

1. Merge the target users list with users data columnwise using the user_id field as key.

2. Merge the target items list with items data columnwise using the item_id field as key.

3. Merge the Interaction data with the above two resulting data using user_id and item_id fields as keys.

After merging, the extended data will have the following structure;



item_id	user_id	caree	discipline_	indust	region_i	is_pay	career_level_u	disciplir	industry_	region_u	premium	experienc	experienc	experi	edu_degree	interaction_type
281370	1E+06	3	3	16	0	0	3	0	21	6	0	1	3	3	0	1
85288	376723	4	8	16	10	0	3	18	2	10	0	2	4	1	3	1
108448	539289	3	17	16	0	0	3	0	0	0	1	3	6	2	0	1
108898	614429	3	17	7	2	0	2	0	0	0	0	1	3	3	1	1
23070	2E+06	3	21	16	2	1	3	0	0	0	0	2	5	2	0	1
23070	2E+06	3	21	16	2	1	3	21	9	2	0	2	3	1	0	1
23070	2E+06	3	21	16	2	1	0	21	9	9	1	1	3	2	0	1
23070	2E+06	3	21	16	2	1	3	0	0	0	0	2	4	3	1	1
77060	2E+06	3	21	10	9	1	3	0	0	0	0	2	4	3	1	1
114660	2E+06	3	21	16	1	0	3	0	0	0	0	2	4	3	1	1
23070	2E+06	3	21	16	2	1	0	0	14	2	0	2	0	0	0	1
23070	2E+06	3	21	16	2	1	3	0	0	0	1	2	4	3	2	1
23070	303229	3	21	16	2	1	2	0	0	0	1	3	6	2	0	1
23070	370699	3	21	16	2	1	0	0	0	0	0	1	4	4	2	1
23070	510379	3	21	16	2	1	3	21	7	2	0	3	5	2	2	1
23070	451489	3	21	16	2	1	3	21	7	2	1	3	5	4	2	1
53970	451489	3	21	9	2	0	3	21	7	2	1	3	5	4	2	1
23070	302718	3	21	16	2	1	0	0	9	2	0	2	4	1	0	1
23070	815008	3	21	16	2	1	3	0	0	0	0	2	3	1	2	1
23070	551158	3	21	16	2	1	3	0	0	0	0	3	0	0	1	1
23070	530118	3	21	16	2	1	3	21	7	2	0	2	3	2	0	1

Figure 13 – Structure of data to be fed to the model

This data will be one-hot encoded then be fed to the Factorization Machine model.

5.2. HIGH LEVEL SYSTEM DESIGN

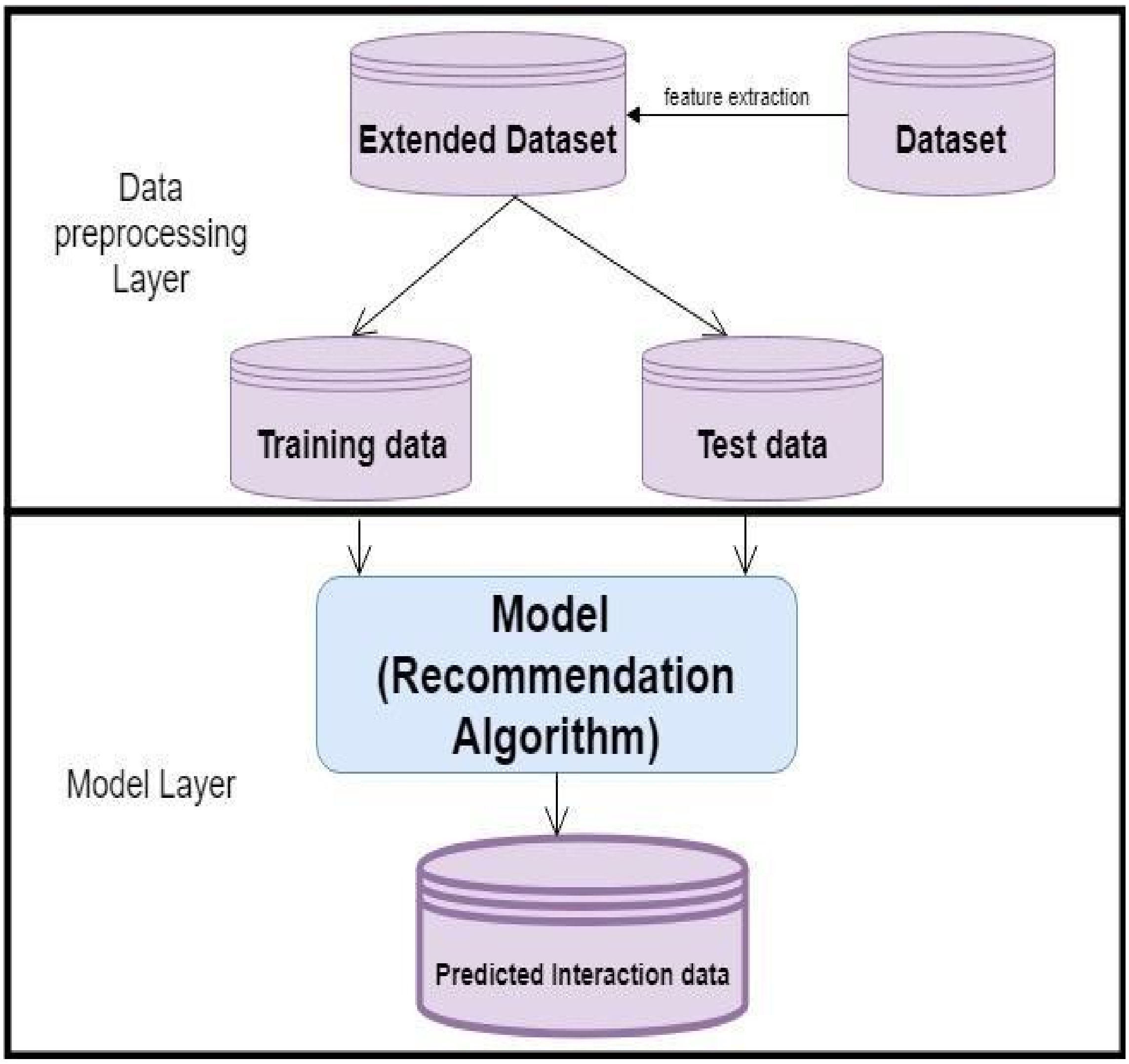


Figure 14 – High level design

From a higher point of view, we have two layers of the overall task, the data Pre-processing layer and Model layer.

In the Pre-processing layer, we build up the extended dataset using the techniques discussed in the above section, Then split the data to Training and Test set with 80/20 slices.

In the the Model layer, we do most of the training the model with training set and predicting for test data.

5.3. WORKFLOW OF PREDICTION

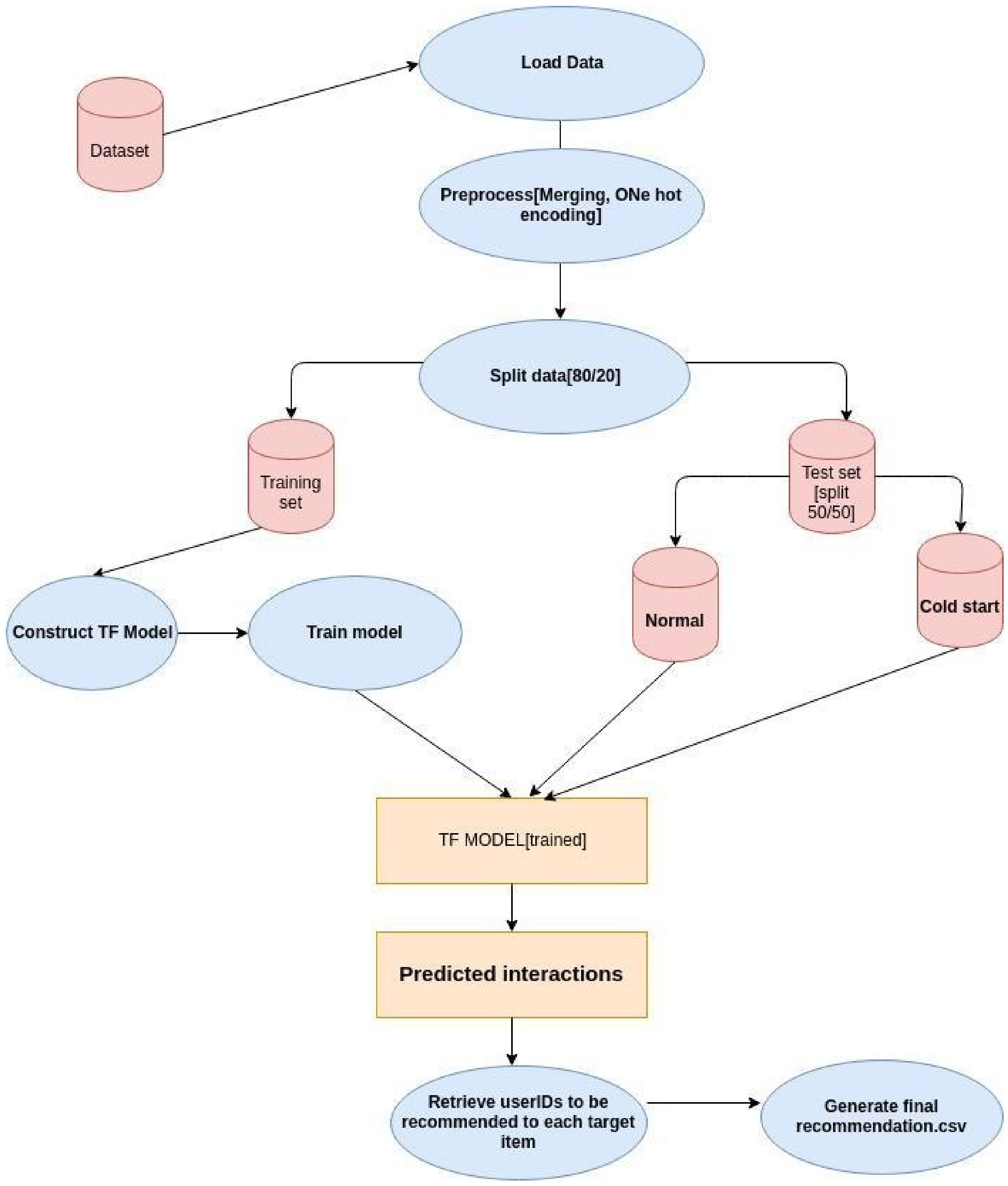


Figure 15– Workflow of Prediction

CHAPTER 6.

6. EVALUATION SETUP

6.1. Evaluation Metrics

Important terms

- **True Positives (tp)**: The cases in which we predicted YES and the actual output was also YES.
- **True Negatives (tn):** The cases in which we predicted NO and the actual output was also NO.
- **False Positives (fp):** The cases in which we predicted YES and the actual output was a NO.
- **False Negative (fn):** The cases in which we predicted NO and the actual output was a YES.

6.1.1. Precision and Recall

Precision indicates the ability of a classifier not to label as positive a sample that is negative. It is defined by the ratio: **tp/(tp+fp)**

where **tp**= Number of true positives and **fp**= Number of false positives.

It takes as input an array of the *true values* of the target and an array of the *predicted values* of the target then outputs a floating point number indicating the *precision of the positive class.*

Recall measures how many true relevant results are returned where **tp**= Number of true positives and **fn**= number of false negatives. It is defined by the ration: tp / (tp+fn)

6.1.2. F-measure (F1 score)

Balanced F-score or F-measure is a weighted average of the *precision* and *recall* and calculated as **F1 = (2 * (precision * recall)) / (precition + recall).**

6.1.3. Accuracy

It is the ratio of the number of correct predictions to the total number of input samples.

- $$\textbf{Accuracy} = \frac{\textbf{Number of correct predictions}}{\text{Total number of input samples}}$$

In other words,

Accuracy = (tp + tn) / (tp + tn + fp + fn)

6.1.4. Confusion Matrix

It is a matrix used to describe the *performance* of a classification model (classifier) on a test data for which the true values are known.

Example:

n = 5888	**Predicted** NO	**Predicted** YES
Actual NO	TN = 4058	FP = 20
Actual YES	FN = 70	TP = 1740

Table 13 – Confusion matrix

6.1.5. Receiver Operating Curve (ROC)

It's a metrics used to evaluate the output quality of a classifier. It typically features *true positive rate* on the x-axis and *false positive rate* on the y-axis.

6.2. Model Setup

order: defines the order of the factorization machines model i.e. the number of parameters we want to estimate interactions between. In our case the order is 2 because we want to model interactions between a specific user and item.

learning_rate: It’s a *floating point hyper-parameter* used for the stochastic gradient descent optimization algorithm. If the learning rate is too high, the model parameters will not converge while if it’s too low the algorithm is no longer time efficient.

n_epoch: It’s an integer number representing the number of iterations for training the model.

batch_size: It’s an integer number representing the number of samples in mini-batches. Set to -1 for full gradient i.e. the whole training set in each batch.

```
model = TFFMClassifier(
  order=2,
  rank=6,
  optimizer=tf.train.AdamOptimizer(learning_rate=0.01),
  n_epochs=50,
  batch_size=1024,
  init_std=0.01,
  input_type='dense',
  seed=42
)
```

6.3. Experimental Results

Total number of samples: 14,719

Training samples: 11,775

Test samples: 2944

order [fixed]	learning_rate	No. of epochs	Batch size	Precision_score	Accuracy_score	Recall	F1-Measure	Confusion matrix
2	0.01	50	1024	**0.978634141276 [weighted]**	**0.9786005435**	**0.978600543478[weighted]**	**0.978485298106[weighted]**	**[[4120 32] [94 1642]]**
2	0.01	50	1024	**0.980338951735 [macro]**	**0.9816576087**	**0.975471353041 [macro]**	**0.977862118664 [macro]**	**[[4109 39] [69 1671]]**
2	0.01	50	1024	**0.984714673913[micro]**	**0.9847146739**	**0.984714673913[micro]**	**0.984714673913[micro]**	**[[4058 20] [70 1740]]**
2	0.01	100	1024	**0.9840353261**	**0.9840353261**	**0.9840353261**	**0.9840353261**	**[[4069 32] [62 1725]]**
2	0.02	100	1024	**0.97265625**	**0.97265625**	**0.97265625**	**0.97265625**	**[[4090 32] [129 1637]]**
2	0.03	100	1024	**0.9711277174**	**0.9711277174**	**0.9711277174**	**0.9711277174**	**[[4119 26] [144 1599]]**
2	0.04	100	1024	**0.9588994565**	**0.9588994565**	**0.9588994565**	**0.9588994565**	**[[4083 44] [198 1563]]**
2	0.05	100	1024	**0.9436141304**	**0.9436141304**	**0.9436141304**	**0.9436141304**	**[[4034 97] [235 1522]]**
2	0.001	100	1024	**0.9755434783**	**0.9755434783**	**0.9755434783**	**0.9755434783**	**[[4048 42] [102 1696]]**

Table 14 – Experimental Results

As seen from this table of results, Best precision and accuracy is achieved with a learning rate 0.01, the model being trained for 50 iterations. We can also see that as we increase the value of the learning rate, the precision and accuracy of the classifier tend to lower.

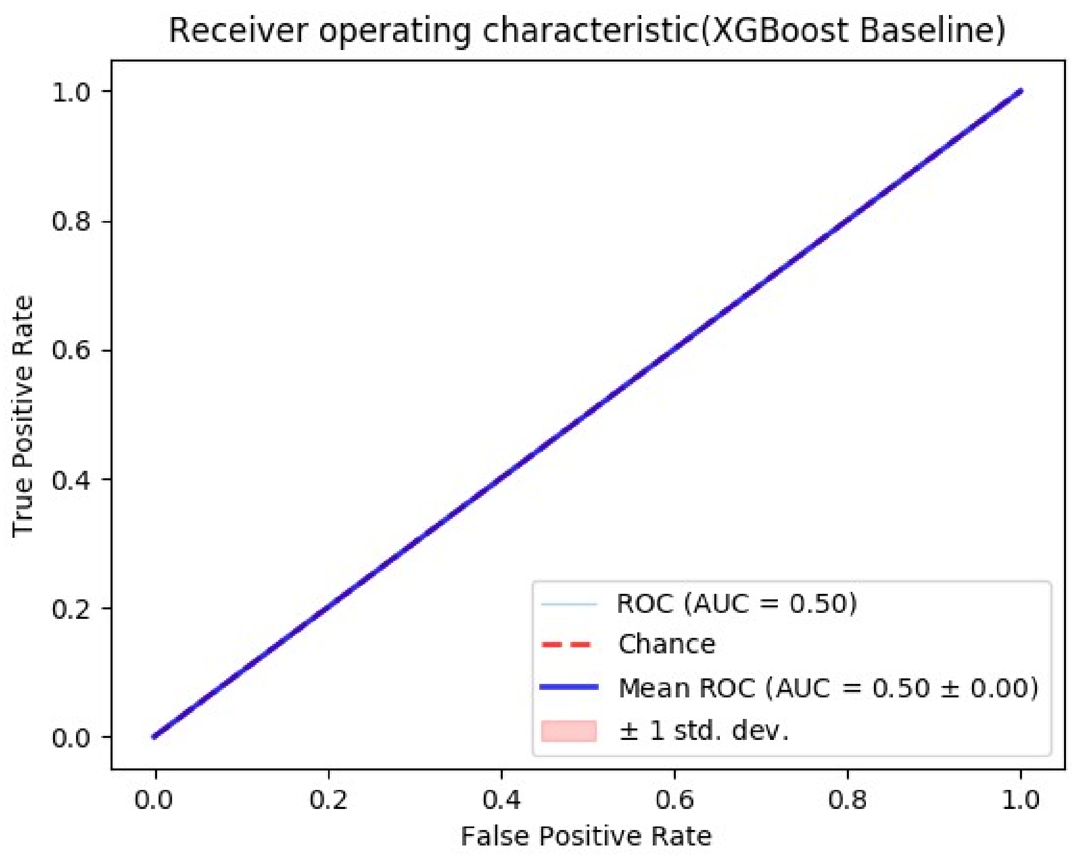


Figure 16 - ROC curve (XGboost baseline)

As seen above, the ROC curve is a performance measurement for classification problems at various threshold settings. It tells us how much the model is capable of distinguishing between positive and negative classes. Figure 16 shows the resulting ROC curve tested on the same dataset used by the Factorization machines. The dotted red line covered with the blue line indicates the random behavior of the classifier. The AUC value represents the degree or measure of separability.

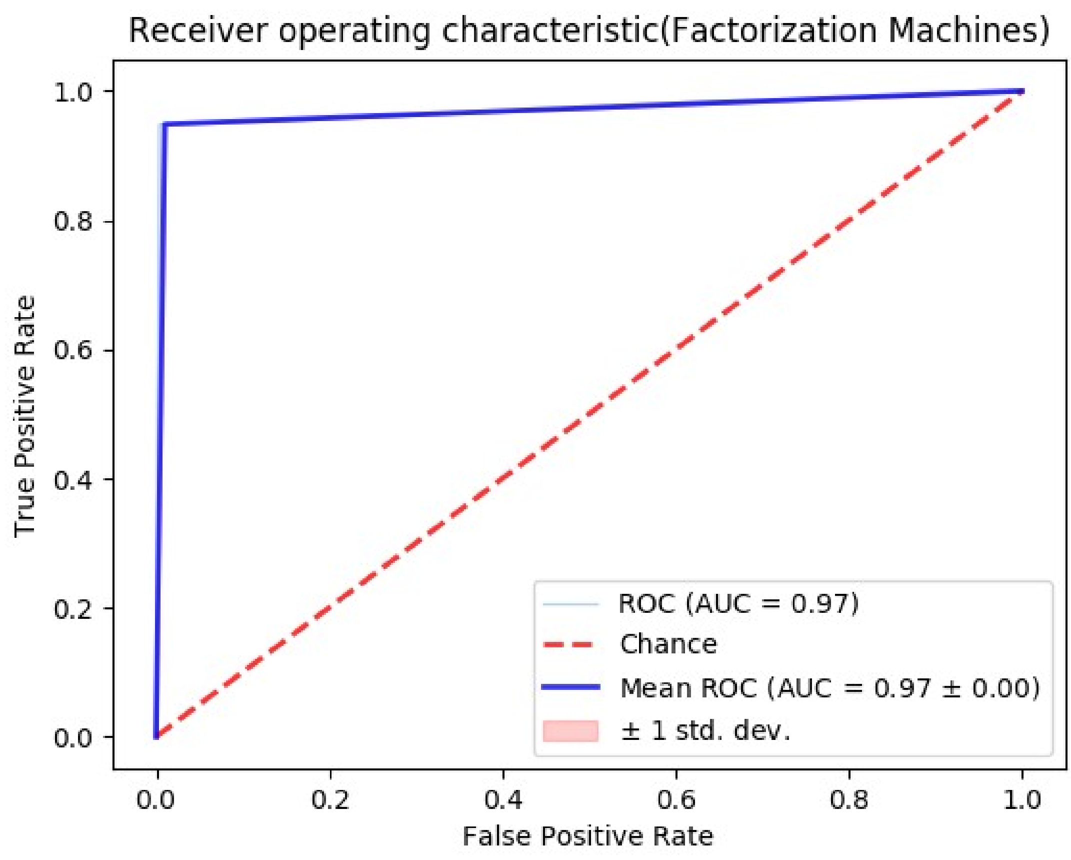


Figure 17 – ROC curve - Factorization machines

An excellent model has an AUC near 1 which indicates that it has a good measure of separability. A poor model has an AUC near 0, which indicates that it has the worst measure of separability, In fact it means it’s reciprocating the results, It’s predicting 0s as 1s and 1s as 0s. Figure 17 shows us that the Factorization machines has it’s curve passing through the top left, and has and AUC = 0.97 meaning that our model is performing well on such dataset.

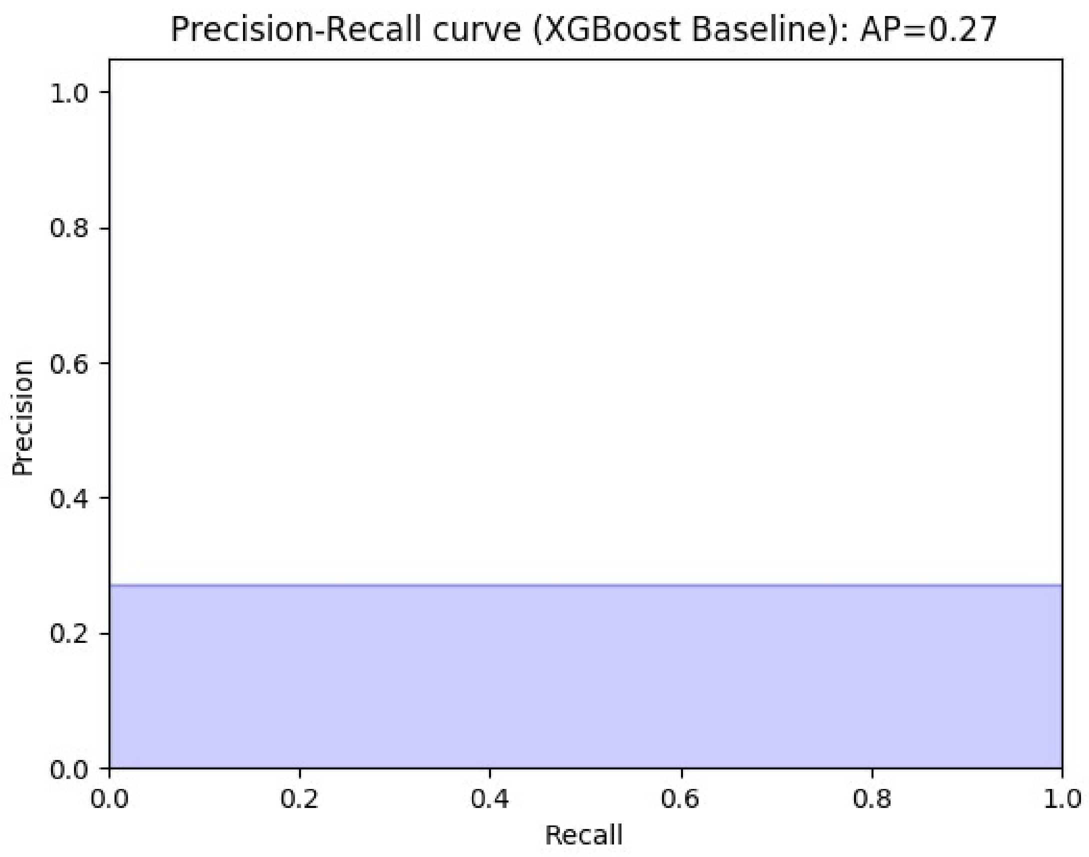


Figure 18 – Precision-Recall curve (XGboost baseline)

Evaluating both precision and recall is useful in cases where there is a label imbalance in the observation data. Specifically there are many examples of class 0 and only few examples of class 1. The value AP represents the average precision score of the classier.

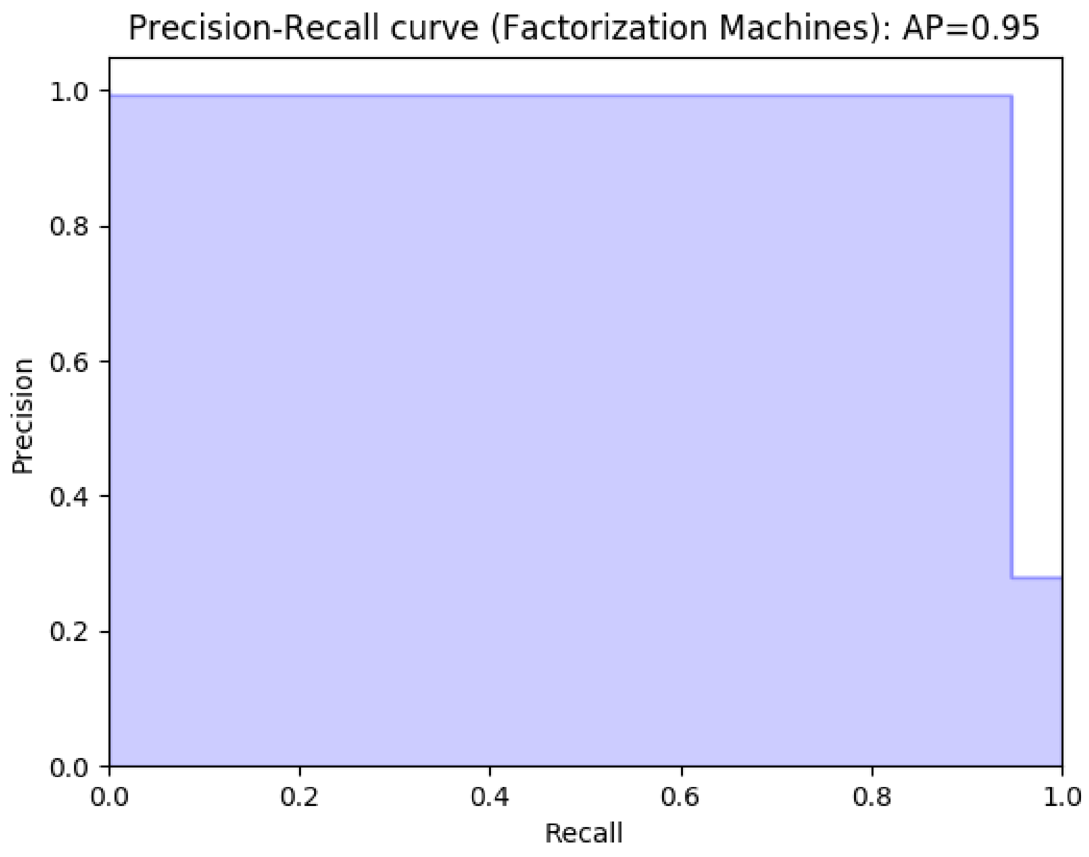


Figure 19 – Precision-Recall curve (Factorization Machines)

The Average precision score for the Factorization machines model is 0.97, which shows us that the model is performing well on such dataset.

Chapter 7.

7. CONCLUSION AND FUTURE WORKS

This paper presents the implementation of the Factorization Machines algorithm for Job recommendations of the ACM RecSys challenge. It is experimentally proven that implementing Factorization Machines for this purpose solves the issue of cold start situations i.e. having to predict for users who never interacted, by integrating content information (features) of both users and items (jobs) in the training data. It is also tested that the algorithm improves precision and accuracy without having to increase its computational complexity. According to the results of this thesis, the Factorization machines can be applied to similar sparse datasets obtaining maximum efficiency and lower computational resources.

Bibliography

(n.d.).

1. Factorization Machines (n.d). Rendle, Steffen. (2010). Factorization Machines. 995-1000. 10.1109/ICDM.2010.127.

2. Hybrid recommendation approaches (n.d). https://www.math.uci.edu/icamp/courses/math77b/lecture_12w/pdfs/Chapter%2005%20-%20Hybrid%20recommendation%20approaches.pdf

3. quuxlabs. (n.d). http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/

4. Recommender systems handbook. (n.d). Recommender systems handbook – second edition by Francesco Ricci . Lior Rokach Bracha shapira

5. sciencedirect. (n.d). https://www.sciencedirect.com/science/article/pii/S1110866515000341