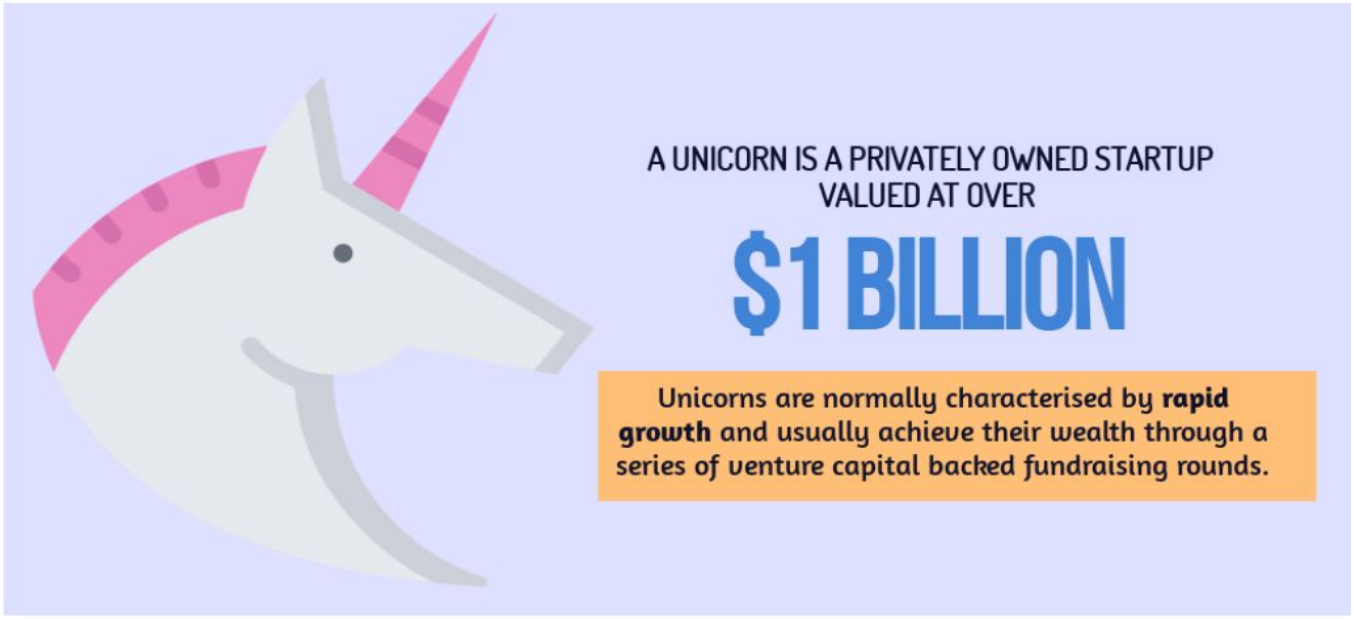


Unicorn Prediction & Startup Investor Recommendation System

Nithin Gollanapally | Id: 013820599
Prathamesh Karve |Id: 013850928
Pranav Dixit |Id: 013838058
Sagar Bonde |Id: 013761566

Innovation highlights

1. Two stage prediction server

Offline recommender system – Generates similarity matrix for user-user similarity and startup-startup similarity

Online recommender system – Uses web cookie data to provide dynamic recommendation based on user's 'Add to Portfolio' action.

2. Model

Unicorn prediction uses a custom designed function for ranking the startups. Startup Recommendation System uses a novel hybrid approach based on latent factor models and collaborative filtering for recommendation.

3. Implicit feedback based recommendations

The website is responsive to user actions

4. Explanations on recommendation

Two recommendation lists shown to user – one on the basis of past investments and second on the basis of user's location. Explanation helps build user's trust in the system

5. AWS-hosted website with backend in NodeJS MongoDB

Built a scalable backend system deployed on cloud to serve the recommendations

6. Evaluation

Confusion matrix for the Unicorn Prediction ML algorithm

RMSE for the offline hybrid recommendation system

Leave one out cross validation for the online recommendation system

Introduction

Predictive analytics is emerging as one of the most successful applications of Artificial Intelligence today. There is a plethora of fields which are using this, from music (Spotify Discover playlist) to medicine (Google has developed a tool which can predict cardiovascular risk factors). Generally, Algorithms are categorically superior than humans in identifying patterns and applying them to future scenarios based on thousands, if not millions of data points. So this way, it seems logical that an algorithm should be able to do the work of unicorn prediction - after all, they are just making a highly informed prediction.

While 85 percent of unicorn prediction values comes from quantifiable factors including funding, trends, timing and addressable customer potential. Therefore, predictive analytics could quantify 85 percent of any investment decision - and it also gives more than just a normal "Invest / do not invest" decision: It could also predict 10-15 years valuation, rate of return of ventures and can guess about the potential market share of a company.

After having the information on the trends of the startups which are going to be successful, we can use this information in recommending investors to companies and similarly companies to invest for investors. So along with Unicorn prediction, we have scaled up our project for Investor - Startup recommendation system and vice versa.

By this, we are not only improving sales and customer experience, but also at the same time it is very valuable to both investors and companies. More importantly, recommending investors is valuable to companies because it gives them an idea to which investors to approach when they are looking to raise a round. And similarly, recommending startups to investors is also valuable as it helps in identifying which startups will have successful valuations and be a good investment options. For this, we have tried to use the information embedded in the investment relationships. For instance, consider a scenario where two investors A and B both invested in a company X. If investor A also invested in company Y, then investor B is more likely to invest in the company Y as well. Therefore, we thought it would be interesting to approach this recommendation system by understanding the relationships between companies and investors.

DataSet Description :

No direct data was available which we can use directly, so we have done web scraping for data collection. BeautifulSoup python package has been used for parsing HTML pages.

Firstly, the names and basic information of the startups was scraped from seeddb.com and angel.co websites and was stored in a csv file which includes fields like Name, State, Crunchbase link, Angel link, website, start date, funding. And then for each company in this list we have collected more information like Categories, Region, IPO Status, Last Funding Type, Number of Funding Rounds,

Number of Articles, Number of Employees, Number of investors using the corresponding crunchbase.com links.

As crunchbase.com website does not allow web scraping for free, the requests cannot be sent continuously using automated scripts. So, a random delay is added after every request. Even after putting the delay, requests got blocked after sometime. After this we have added a code snippet that checks the point data which has been collected and starts hitting requests for startups after that point.

From all the raw data collected, we have used below four parameters for prediction and evaluation-

1) Startup location

2) Funding amount

3) Domain of the startup

4) Number of articles citing the startup.

Data Cleaning :

The data collected from web scraping had many inconsistencies and missing values, so data cleaning was necessary. If a row has many missing values , we have dropped the values. We have also tried to remove unnecessary punctuation and characters from the attributes.

Data Preprocessing :

Unicorn Prediction:

1. Categories –

The categories in which each startup falls is used to calculate the similarity index of each startup. The intuitive idea behind this is that the startups which are part of common domains like E-commerce or software will have a large similarity index while the startups which are in domains like Bitcoin or autonomous vehicles have a low similarity index as not many startups are working in this domain. Following is an example of a list of startups with its categories-

```
Airware  ->  ['Aerospace', 'Analytics', 'Drones', 'Enterprise_Software', 'Robotics']
ALOHA  ->  ['Fitness', 'Food_and_Beverage', 'Health_Care', 'Wellness']
App Annie  ->  ['Analytics', 'Big_Data', 'Mobile', 'SaaS']
AptDeco  ->  ['E_Commerce', 'Furniture']
Arcus  ->  ['E_Commerce', 'Mobile', 'Mobile_Payments']
Arthena  ->  ['Machine_Learning', 'Marketing']
Binded  ->  ['Art', 'Artificial_Intelligence', 'Big_Data', 'Digital_Media', 'Intellectual_Property', 'Machine_Learning']
Blockstack  ->  ['Developer_Platform', 'Internet', 'Open_Source']
Blockstream  ->  ['Bitcoin', 'Computer', 'Cryptocurrency', 'Data_Storage', 'FinTech']
Blokable  ->  ['Building_Material', 'Construction', 'Manufacturing', 'Property_Management', 'Real_Estate', 'Smart_Home']
Brandfolder  ->  ['Brand_Marketing', 'Digital_Media', 'SaaS', 'Software']
Brandwatch  ->  ['Marketing', 'Social_Media']
Breather  ->  ['Interior_Design', 'Internet', 'Mobile_Apps', 'Real_Estate', 'Web_Apps']
Bright  ->  ['Career_Planning', 'Recruiting', 'Staffing_Agency']
Capillary Technologies  ->  ['Analytics', 'Enterprise_Software', 'Retail', 'SaaS']
CARD.com  ->  ['Banking', 'Credit', 'FinTech', 'Mobile', 'Payments']
```

Tf-Idf is used to generate a vector for each startups based on the category keywords. Term Frequency-Inverse Document Frequency is method which generates a number for each keyword and that number reflects the relative importance of the keyword in representing the document. In the case of startup categories each category appears only once in the description of each startup so term frequency is always 1. So technically only idf is being calculated for each keyword. Typical functions of idf is defined as –

$$idf\,(t, D) = log(N/n_i)$$

Once vector for each startup is defined, cosine similarity is used to find the similarity of each startup with all other startups. Cosine similarity is a measure of the similarity between two vectors by calculating the cosine of angle between the two vectors.

$$cos\ \theta\ = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}.\sqrt{\sum_{i=1}^{n} b_i^2}}$$

After calculating the cosine similarity, a n by n matrix is generated with similarity of each startup with the other.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		Accredible	Airware	ALOHA	App Annie	AptDeco	Arcus	Arthena	Binded	Blockstack	Blockstrea	Blokable	Brandfolde	Brandwatc	Breather	Bright	Capillary T	CARD.com
2	Accredible	0	0	0	0	0	0	0	0	0	0	0	0	0	0.308729	0	0	0
3	Airware	0	0	0.169969	0	0	0	0	0	0	0	0	0	0	0	0.298219	0	0
4	ALOHA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	App Annie	0	0.169969	0	0	0.197176	0	0.174447	0	0	0	0.192218	0	0	0	0.436965	0.133696	0
6	AptDeco	0	0	0	0	0.196711	0	0	0	0	0	0	0	0	0	0	0	0
7	Arcus	0	0	0	0.197176	0.196711	0	0	0	0	0	0	0	0	0	0	0.134869	0
8	Arthena	0	0	0	0	0	0	0.244792	0	0	0	0	0.524746	0	0	0	0	0
9	Binded	0	0	0	0.174447	0	0	0.244792	0	0	0	0.229286	0	0	0	0	0	0
10	Blockstack	0	0	0	0	0	0	0	0	0	0	0	0	0.12055	0	0	0	0.196865
11	Blockstrea	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.103147	0
12	Blokable	0	0	0	0	0	0	0	0	0	0	0	0	0.169906	0	0	0	0
13	Brandfolde	0	0	0	0.192218	0	0	0	0.229286	0	0	0	0	0	0	0.168627	0	0
14	Brandwatc	0	0	0	0	0	0	0.524746	0	0	0	0	0	0	0	0	0	0
15	Breather	0	0	0	0	0	0	0	0	0.12055	0	0.169906	0	0	0	0	0	0.154225
16	Bright	0.308729	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	Capillary T	0	0.298219	0	0.436965	0	0	0	0	0	0	0	0.168627	0	0	0	0	0
18	CARD.com	0	0	0	0.133696	0	0.134869	0	0	0	0.103147	0	0	0	0	0	0	0

The mean of each row is generated. This value represents the mean similarity of a startup with all the other startups. This is the similarity index of each startup.

$$Similarity\ Index\,(x) = \frac{1}{N} * \sum_{i=1}^{N} Cosine\,(\underline{x}, i)$$

2. Funding amount –

Funding amount is an important factor in deciding the success of the startup. Higher funding amount usually tends to higher chance of a startup becoming successful. Crunchbase database provides the funding amount in dollars. But the funding amount cannot be used directly in our model. The time required to gather the capital also plays an important role. For example, a startup securing $10 Million in

2 years is more impressive that a startup generating $10 Million in 10 years. To factor this a decay function is used to get a normalized funding amount value. The decay function is given as –

$$Normalized\ amount\,(x) = x.e^{-0.1t}$$

$$where\ \ x = funding\ amount$$

$$t = time$$

3. Number of articles -

Crunchbase data provides the number of articles published regarding a startup. This number indicates the publicity generated by the startup. But this number alone cannot be used as it is in the unicorn prediction model. Similar to funding amount, a startup generating more articles related to it in small time will generate more hype. Therefore, the rate in increase in the number of articles also has to be taken into consideration. To take into consideration both the factors, a weighted mean is taken of the growth in increase as well the number of articles published to generate a popularity index.

$$Popularity\ Index\,(x) = 0.6 * growth_{rate} + 0.4 * x$$

After preprocessing the data we get a tuple for each startup with similarity index, normalized funding amount and popularity index.

```
Airware    ->    [0.0267191211237504, 53020817.76583214, 97.375]
ALOHA    ->    [0.039998067114698936, 3333681.99306773, 11.4]
App Annie    ->    [0.08138156018327156, 63424866.91953346, 913.7333333333333]
AptDeco    ->    [0.02834221186835523, 1241463.2594785236, 4.857142857142857]
Arcus    ->    [0.05393480393677566, 10372539.922177099, 11.0]
Arthena    ->    [0.027258452650310572, 727836.7916551601, 6.76]
Binded    ->    [0.025463456326412094, 909795.9895689501, 18.2]
Blockstack    ->    [0.019093147367330542, 38855863.83545707, 16.5]
Blockstream    ->    [0.01807732369695364, 61259596.63097598, 143.0]
Blokable    ->    [0.0053257901733862216, 8445327.715771584, 2.4]
Brandfolder    ->    [0.05048285103995965, 6036927.99703429, 8.5]
Brandwatch    ->    [0.03151071477523105, 15954823.567021936, 177.5857142857143]
Breather    ->    [0.02325879543842245, 67229425.42151824, 14.500000000000002]
Bright    ->    [0.005799493668923776, 1865959.5627196897, 4.5]
Capillary Technologies    ->    [0.05680254291949309, 33986137.64557392, 75.0]
CARD.com    ->    [0.04035926158429542, 9236486.650520217, 6.314285714285715]
```

Startup-Investor Recommendation System:

MultiLabelBinarizer:

A startup company is described by the categories in which it works, location and the funding round it is in. Similarly investors have their own preference of startup in a particular category, location and investment round. To describe this data of startup and investor a MultiLabelBinarizer is used. This creates a vector for each startup and investor with 1s and 0s. The category, location and funding rounds which match with the investor and startups profile are set 1 while all the other values are set 0.

Feature scaling:

Feature scaling is an important data preprocessing stage. If the different features are having different scales, then it can lead to different problems like generating useless Euclidean distances. After initial preprocessing the startup data has three features – Similarity index, Normalized funding amount and Popularity index. But each of these is having different scales. Similarity index ranges from 0 to 1. Funding amount can be 0 and extent to any value. Popularity index can also take any amount between 0 to infinity. Scaling is required before they can be used in the model. All the three features are scaled between 1 to 10. MinMaxScaler provided sklearn library is used for this purpose. Data after scaling looks as follows-

```
Airware    ->    [3.95487196 1.06694799 1.19608787]
ALOHA    ->    [5.42339276 1.00752046 1.01230814]
App Annie    ->    [10.          1.63123019  1.23456972]
AptDeco    ->    [4.13436983 1.00299791 1.00456957]
Arcus    ->    [6.96465875 1.00724397 1.03834303]
Arthena    ->    [4.01451672 1.00431321 1.0026698 ]
Binded    ->    [3.81600778 1.01222075 1.00334282]
Blockstack    ->    [3.11151428 1.01104568 1.1436954 ]
Blockstream    ->    [2.99917417 1.09848485 1.22656096]
Blokable    ->    [1.58898    1.00129949 1.03121478]
Brandfolder    ->    [6.58290672 1.00551593 1.02230674]
Brandwatch    ->    [4.48477508 1.12239114 1.05899044]
Breather    ->    [3.57219398 1.00966324 1.24864179]
Bright    ->    [1.64136695 1.00275105 1.00687942]
Capillary Technologies    ->    [7.28180248 1.05148197 1.12568356]
CARD.com    ->    [5.46333731 1.00400512 1.03414107]
```

Modeling:

Unicorn Prediction:

The chance of a startup becoming a unicorn depends mostly on the novelty of idea. A startup doing work in a domain which has never been done before attracts attention and customers and has a higher chance of becoming a unicorn. The inverse of similarity index can give the uniqueness index which can be used as an indicator of how unique the startup is. Funding and publicity also play a crucial role is a startup's success.

The unicorn prediction uses a hybrid recommendation system approach along with a custom ranking function to rank a startup's unicorn probability. The system is partially based on content-based recommendation system as tf-idf and cosine similarity is used to find the similarity index of each startup. A unicorn index is calculated using the 3 value data tuple. The unicorn index is directly proportional to normalized funding amount and popularity index and inversely proportional to similarity index.

$$Unicorn\ Index \propto \frac{(NF * PI)}{SI}$$

$$NF = Normalized\ Funding\ Amount$$

$$PI = Popularity\ Index$$

$$SI = Similarity\ Index$$

Using this function unicorn index value is generated for each startup and sorted according to highest to lowest value. From our limited data set following is the list of startups with the highest unicorn index and having a high probability of becoming a unicorn as per our model-

Name	Unicorn Index
STILT	1.0086
MaestroQA	0.8038
Medium	0.7735
SolveBio	0.7043
Chai	0.7006

Startup - Investor Recommendation System:

The Startup recommendation system is about recommending the Potential investor for a particular startup based on the category it belongs to, the region its located in, & the existing investors onboard as well as recommending best investment deal for a venture capitalist based on his past investments.

We are using pipelined hybrid recommendation systems technique for implementing this part.

Phase I: Collaborative Filtering approach for calculating similarity between two investors, as well as similarity between Startups.

In our implementation the Collaborative Filtering approach is giving us an estimated similarity scores between two startups / two investors. We are treating Startups as Items, and are applying item - item based similarity on startups. Similarly the investors are treated as Users and User-User based similarity is being applied on investors.
This approach briefly mimics the classic customers who have bought this have also bought this type of recommendation algorithms

For both User-User & Item-Item similarity we are using a KNNMeans algorithm for finding the similarity. These algorithms are derived from the nearest neighbor algorithm. To compute the Pearson coefficient between the pairs of startups or the investors the pearson baseline function is using the centering technique.
The Pearson baseline correlation coefficient can be defined as-

$$pearson\_baseline\_sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - b_{ui}).(r_{vi} - b_{vi})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - b_{ui})^2}.\sqrt{\sum_{i \in I_{uv}} (r_{vi} - b_{vi})^2}}$$

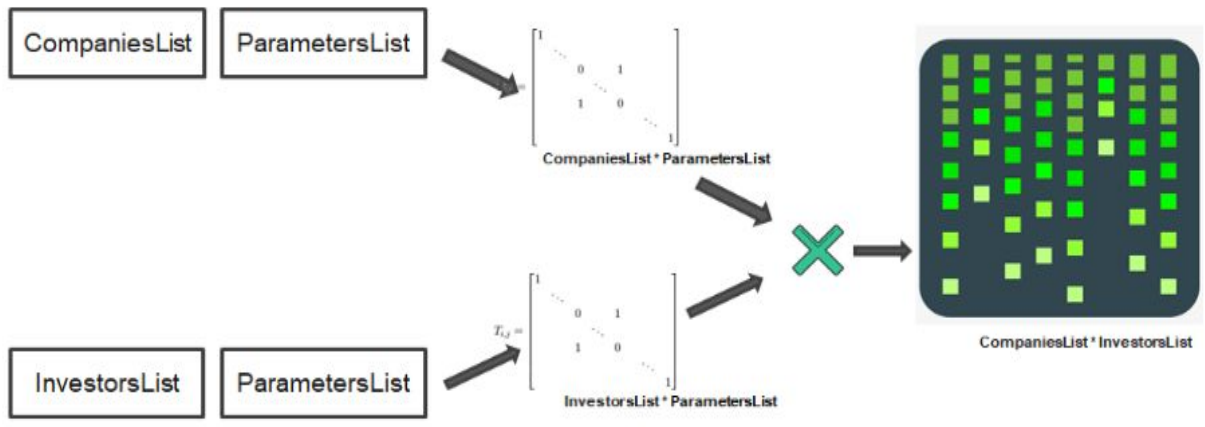
Further the KNNmean estimates the actual number of neighbors that are aggregated based on various values of k.
The best value of k found is 10.

Phase II: Go - No Go Approach by clustering based ranking using Matrix Multiplication (Inverse of Matrix Factorization)

Chronologically this part actually runs before the Phase I, and thus filters the actual number of Investors that can be recommended to a Startups and vice-a-versa.

Based on the decision parameters mentioned in the earlier part of the report such as Field / domain of the company operations, the funding parameters, as well as location which act as latent factors for the two matrices 1 with startups score with the latent factors and other with the investor scores with the latent factors, which when multiplied with each others gives the similarity score for every startup with every investor.

The following diagram represent the approach discussed above.

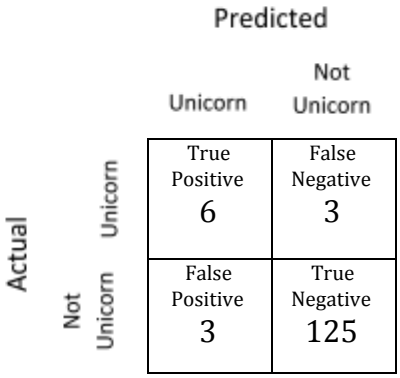


The Final Similarity score is computed by multiplying the scores from Phase I and Phase II

Evaluation –

Unicorn Prediction:

For evaluating the unicorn prediction model a test data set was created with known existing unicorns in it. The dataset contained 9 unicorns and 128 non-unicorn startups. The data was provided to the model for generating unicorn index rankings. Confusion matrix and Accuracy were used for evaluating the model. A confusion matrix shows the number of items classified into a category versus the actual number.



$$Accuracy = \frac{TP+TN}{total} = 0.95$$

$$True\ Positive\ Rate\ (Sensitivity) = \frac{TP}{Positives} = 0.66$$

Evaluation of the Startup - Investor Recommendation:

The Collaborative Filtering algorithm can be evaluated on the basis of the RMS Error value.
The same can be given as :

$$RMSE = \frac{1}{N} * \sum_{i=1}^{N} (r_{ui} - \hat{r_{ui}})^2$$

For the KNNmeans algorithm the k values define the highest no. of neighbors to be taken into account for aggregation.
Following is a table of RMSE values for different k values.

K Value	Root Mean Square Error Value
K= 2	1.4489
K= 5	1.2547
K= 10	1.2161

Deployment:

Website:

Link - http://ec2-54-149-213-124.us-west-2.compute.amazonaws.com:3000/

Since one module of our recommendation system depends on current cookie data, clear the cookies for this website after each use.

Distributed backend:

Deployed on a distributed and scalable backend infrastructure that is hosted on AWS. The frontend website is backed by a MongoDB database. The database can be replicated across multiple instances and can be used in a Map Reduce framework.

MongoDB – Database that stores similarity matrices for both users and startups

NodeJS – Scalable server that can support multiple users simultaneously

Bootstrap – Intuitive and responsive UI

Recommendation Tuning:

Online recommendation model uses incoming data as implicit ratings from the user.

When user clicks 'Add to Portfolio' on the recommendation page, we add it to a web cookie.

Online recommender model runs immediately to generate new top-3 recommendations based on this data. So, as the user keeps using the system, he keeps seeing new and interesting recommendations based that is similar to the user's choices.

Online recommendation validation:

Leave-one-out cross validation used to test online recommendation model.

For a user's portfolio, one of the startups in the portfolio is left out and the model is run.

```
def HitRate(topNPredicted, leftOutPredictions):
    hits = 0
    total = 0

    # For each left-out rating
    for leftOut in leftOutPredictions:
        userID = leftOut[0]
        leftOutStartupID = leftOut[1]
        # Is it in the predicted top 10 for this user?
        hit = False
        for startupID, predictedRating in topNPredicted[int(userID)]:
            if (int(leftOutStartupID) == int(startupID)):
                hit = True
                break
        if (hit) :
            hits += 1

        total += 1
```

If this startup appears in the top-N recommendations for that user, it is considered a hit.

The hit rate is captured for all the users for each startup in the portfolio.