Massive Data Mining - Spring '18 - Final Project Report
Team Members: Tahiya Chowdhury, George Chantzialexiou, Christos Mitropoulos

TheNextHit: Predicting movie success and building recommendation system based on TMDB

Code and Dataset can be found here:
https://github.com/CMitropoulos/MassiveDataMining/tree/master/finalProject/Deliverable

INTRODUCTION

Movie industry is always looking for ways to maximize revenue from movies. To facilitate production companies we build a revenue prediction tool. We also build a classification tool to predict if a movie will make profit or loss. Along with predicted revenue and classification for movie success, we also built a recommendation engine to help consumers in movie industry to navigate through available movies without hassle. With the help of the recommendation engine, the consumers will have a better experience out from a large collection of movies.

This project has two parts. The first part is about predicting movie revenue and classifying movies into a success or a bust. The second part is about building regular and streaming movie recommendation systems.

MOTIVATION

Modern video-on-demand companies like Netflix, Amazon, Youtube RED have recently started producing original content.
These companies need to keep users satisfied by making good movie recommendations and maximize their profit by producing original content that are likely to attract the users.
There is a need for **recommendation systems**, movie **revenue predictions** and p**rediction of a movie's success/failure** to provide streamlined experience for both users and the companies

CONTRIBUTION

Our contribution is the following:

1. Movie-revenue prediction system to help companies make better investments.
2. Binary classifier that predicts if a movie will make a profit or a loss
3. Two types of movie-recommendation systems:
    a. Traditional recommendation system: This uses the information available on the database already to recommend movies based on the content and user ratings available. This type of recommendation

system specifically addresses the cold start problem faced by a user who is a new user of the service and do not have a lot of history.

b. Online - streaming recommendation system: This uses the information available from other users of the system. As the market for these type of services keeps expanding, new users and ratings are entering the system everyday. This substantially influence the recommendation results. Our online streaming recommendation addresses the problem of updating the recommendation engine with the ever-changing nature of the market.

DATASET

This dataset is an ensemble of data collected from TMDB and GroupLens. The Movie Details, Credits and Keywords have been collected from the TMDB Open API. These files contain metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages. This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

There are 5000 movies for which we have data on revenue and budget ratio. This is close to 10% of the entire dataset. Although this may seem small, this is enough to perform very useful analysis and discover interesting insights.

After some data preprocessing we come up with following features:

'belongs_to_collection' : Boolean variable that specifies if the movie belongs to a movie franchise, like ‘Lord of the Rings’ or ‘Star Wars’

'budget': The movie budget in dollars

‘popularity' : Popularity score assigned by the TMDB

'production_countries': Number of countries where the production took place

'runtime': the runtime of the movie in minutes

'spoken_languages': number of spoken languages in the film

'vote_average': The average rating of the movie

'vote_count': The number of votes by users, as counted by the TMDB

'year': the year the movie was released

'cast_size': the size of the cast

'crew_size': the size of the crew

'is_english': Boolean variable to describe if the movie is in english or not.

'is_non-popular_production': Boolean variable to describe if the production company has made a lot of movies. The exact number of movies required to make a production company popular is selected based on the validation set scores.

'is_non-popular_director': Boolean variable to describe if a director has made a lot of movies. The exact number of movies required to make a director popular is selected based on the validation set scores. 'is_Animation', 'is_Comedy', 'is_Family', 'is_Adventure', 'is_Fantasy', 'is_Drama', 'is_Romance', 'is_Action', 'is_Crime', 'is_Thriller', 'is_History', 'is_Science Fiction','is_Mystery', 'is_Horror', 'is_War', 'is_Music': All these are boolean variables to describe the genre that the film belongs 'is_Friday': Boolean variable to describe if a movie was released on a Friday. 'is_Holiday': Boolean variable to describe if a movie was released during holiday season.

METHOD

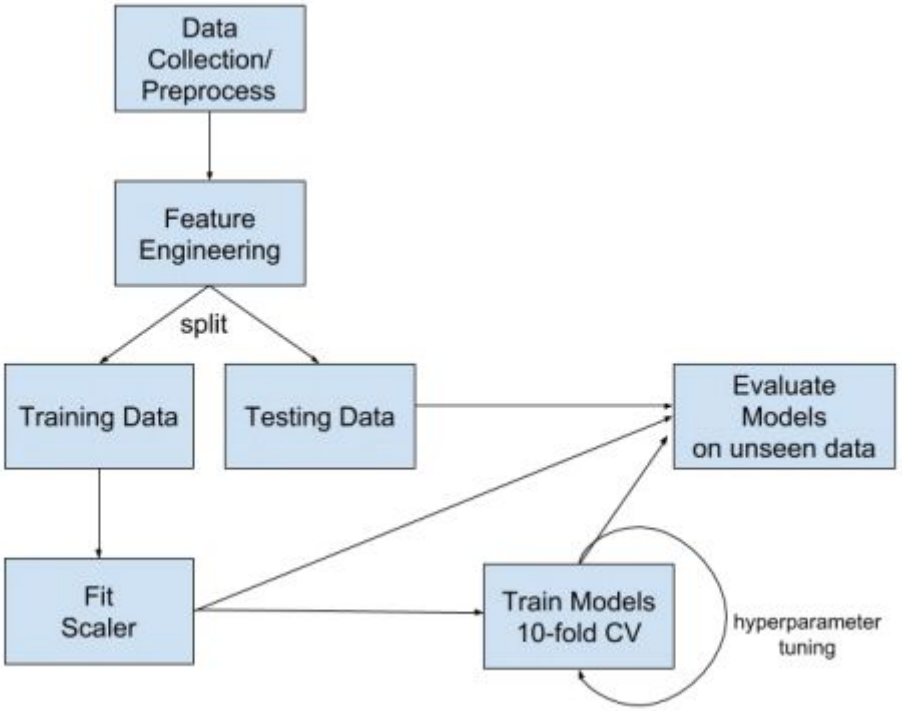
Revenue Prediction

We chose a variety of Regressors for our problem. The hyperparameters were chosen using grid search when that was possible. We will only report the numbers of the best performing hyperparameters. The regressors we used are the following: Extra Trees Regressor, K-Nearest Neighbors Regressor, Linear Regression, Random Forest Regressor, Adaboost Regressor, Gradient Boosting Regressor. The score we report in the result section refer to the coefficient of determination R^2 of the prediction.

Binary Classification

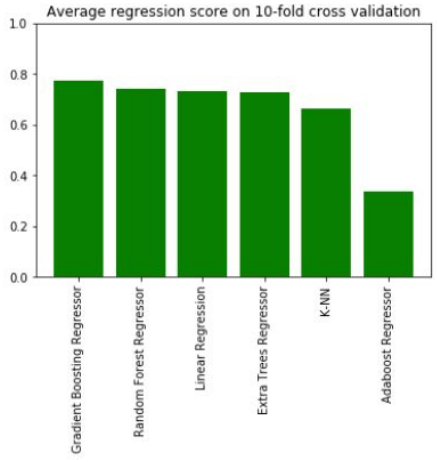
We chose a variety of Classifiers for the binary classification. The hyperparameters were tuned using a 10-fold cross validation. The classifiers we used are the following:
K-NN, Decision Tree, Random Forest, Neural Net, Naive Bayes, SGDClassifier.
We measure accuracy and recall of each classifier.

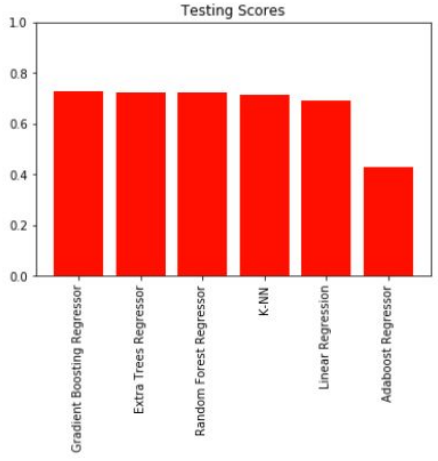
The system overview for the above methods is the following:



RESULTS

Revenue Prediction

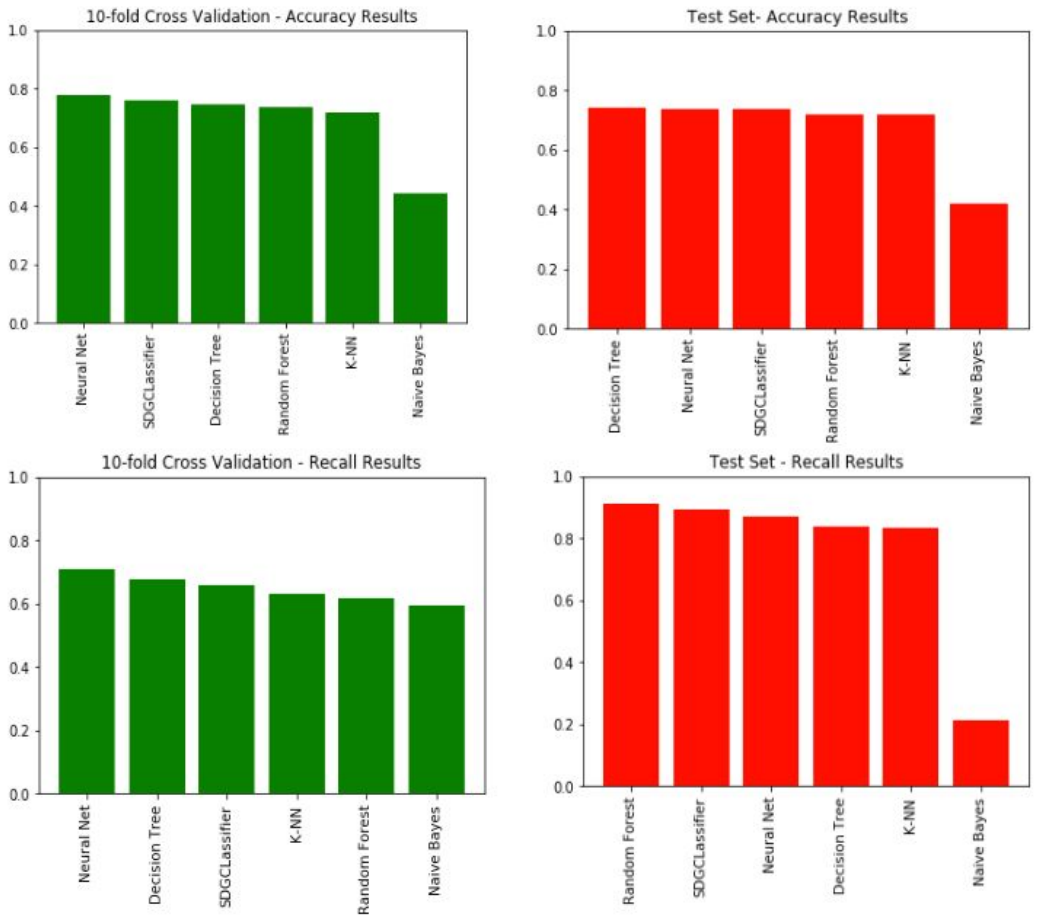




Best performing model is the Gradient Boosting Regressor (CV-score: 0.77, Test Score: 0.72).

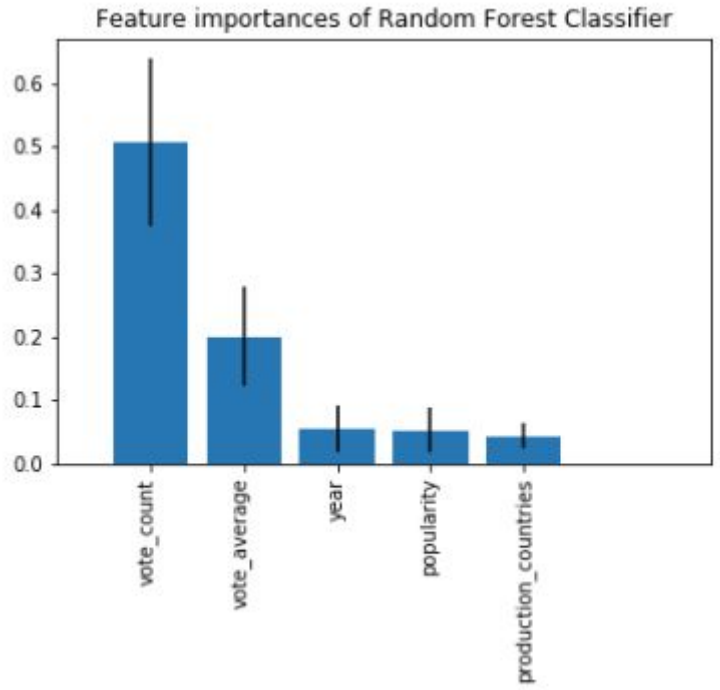
Most models, except AdaBoost Regressor, have similar performance.

Binary Classification



Decision Trees have higher accuracy in the Test set, however the Random Forest Classifier performs better with regards to recall.
A balanced dataset could provide better results.

We also report the feature importances of one of our best performing classifiers; Random Forest Classifier. The important features can provide interesting conclusions about what makes a movie successful.



Content-based and Collaborative Filtering-based Recommendation System

When users first enter a service like Netflix or IMDB, a common problem faced is the 'cold start problem'. This particularly arises when the new user or movie have little to no history of ratings which can be used to predict the ratings required to recommend movie. One solution this problem is content based recommendation.

To build a content based movie recommendation system for a new user, we hypothesized that such users, at the initial service period, will search for movies using either -

1) A particular genre or
2) A particular movie that they have liked

To address this problem, we used the movies metadata available for all 45,000 movies listed in the Full MovieLens Dataset. The dataset contains 24 different datapoints. Judging by the goal, we decided to use from which we decided to use: title, genres, vote_count, vote_average, year, and popularity.

Content-based Recommendation

Approach

Using the vote_average and vote_count available in the sub dataset, we calculated a weightage for each movie as a metric for a movie to appear on the list. To calculate the weightage we have used the standard weightage formula for IMDB movies.

weighted rating , wr = (v/(v+m))*R + (m/(v+m))*M [2]

R and v represents the average vote_count and vote_average for a particular movie, whereas m and M are the minimum votes required to be on the list and the average ratings of all ratings.

Using the weighted rating along with the datapoints, we created a simple recommendation system that recommends movie based on a specific genre query.

Results

For example, a search for ‘Crime’ returns the following list of 10 movies.

```
genre_list('Crime').head(10)
```

	title	year	genre	vote_count	vote_average	popularity	wr
12481	The Dark Knight	2008	Crime	12269	8	123.167	7.926567
292	Pulp Fiction	1994	Crime	8670	8	140.95	7.897324
314	The Shawshank Redemption	1994	Crime	8358	8	51.6454	7.893652
834	The Godfather	1972	Crime	6024	8	41.1093	7.854816
46	Se7en	1995	Crime	5915	8	18.4574	7.852298
586	The Silence of the Lambs	1991	Crime	4549	8	4.30722	7.811259
289	Leon: The Professional	1994	Crime	4293	8	20.4773	7.800891
3030	The Green Mile	1999	Crime	4166	8	19.9668	7.795313
1057	Reservoir Dogs	1992	Crime	3821	8	12.2203	7.778453
1178	The Godfather: Part II	1974	Crime	3418	8	36.6293	7.754867

For ‘Animation’:

```
genre_list('Animation').head(10)
```

	title	year	genre	vote_count	vote_average	popularity	wr
359	The Lion King	1994	Animation	5520	8	21.6058	7.837175
5481	Spirited Away	2001	Animation	3968	8	41.0489	7.780031
9698	Howl's Moving Castle	2004	Animation	2049	8	16.136	7.611398
2884	Princess Mononoke	1997	Animation	2041	8	17.1667	7.610152
5833	My Neighbor Totoro	1988	Animation	1730	8	13.5073	7.554643
40251	Your Name.	2016	Animation	1030	8	34.461252	7.344597
5553	Grave of the Fireflies	1988	Animation	974	8	0.010902	7.318899
19901	Paperman	2012	Animation	734	8	7.19863	7.181326
39386	Piper	2016	Animation	487	8	11.243161	6.966480
20779	Wolf Children	2012	Animation	483	8	10.2495	6.962069

Based on observation, the system gives a fairly good recommendation for a genre query.

Recommendation for a particular movie

Approach

To facilitate recommendation for a particular movie query, we decided to use additional data such as main cast, director, and keywords. The idea was to use the information that are likely to influence user’s choice of movies and use them to find the user similar movies (e.g. a user may be particularly interested in movies like ‘The Godfather’).

To accomplish this goal, from small sub dataset of credits and keywords of all movies, we picked the director, top 3 cast member, frequently appearing keywords and genre. We then merged all 4 categories of information in one single dataframe. We then converted it to a matrix and calculated the cosine similarity for each pair.

Using the cosine similarity for each movie to all other movies, we create a recommendation list for all the movies similar to the query movie.

Results

For example, when searched for movies similar to ‘The Godfather’, the system recommends the following list of 10 movies:

```
movie_recommendation('The Godfather').head(10)
```

	title	year	genres	vote_count	vote_average	popularity	wr
1199	The Godfather: Part II	1974	[Drama, Crime]	3418	8	36.6293	7.876798
1186	Apocalypse Now	1979	[Drama, War]	2112	8	13.5963	7.805979
1934	The Godfather: Part III	1990	[Crime, Drama, Thriller]	1589	7	17.1853	6.839442
1312	Dracula	1992	[Romance, Horror]	1087	7	16.7777	6.774806
3635	The Conversation	1974	[Crime, Drama, Mystery]	377	7	13.2456	6.477064
24284	The Drop	2014	[Drama, Crime]	859	6	11.695	5.881436
2025	The Outsiders	1983	[Crime, Drama]	293	6	6.43593	5.733297
1614	The Rainmaker	1997	[Drama, Crime, Thriller]	239	6	6.6834	5.697202
8911	Rumble Fish	1983	[Action, Adventure, Crime, Drama, Romance]	141	6	8.20519	5.598616
754	Jack	1996	[Comedy, Drama, Science Fiction]	340	5	6.28724	5.078367

For the movie ‘Memento’, the system identified all notable movies by Director Christopher Nolan along with movies of similar genre and keywords:

```
movie_recommendation('Memento').head(10)
```

	title	year	genres	vote_count	vote_average	popularity	wr
15651	Inception	2010	[Action, Thriller, Science Fiction, Mystery, A...	14075	8	29.1081	7.969033
12589	The Dark Knight	2008	[Drama, Action, Crime, Thriller]	12269	8	123.167	7.964533
23076	Interstellar	2014	[Adventure, Drama, Science Fiction]	11187	8	32.2135	7.961151
11463	The Prestige	2006	[Drama, Mystery, Thriller]	4510	8	16.9456	7.905607
18442	The Dark Knight Rises	2012	[Action, Crime, Drama, Thriller]	9263	7	20.5826	6.970199
10210	Batman Begins	2005	[Action, Crime, Drama]	7511	7	28.5053	6.963392
45843	Dunkirk	2017	[Action, Drama, History, Thriller, War]	2712	7	30.938854	6.902223
5302	Insomnia	2002	[Crime, Mystery, Thriller]	1181	6	11.425	5.909906
11855	Disturbia	2007	[Thriller, Drama, Mystery]	1038	6	18.0069	5.899151
29787	Regression	2015	[Horror, Mystery, Thriller]	600	5	10.0363	5.051557

These two recommendation approaches solved the cold start problem for a new user. To utilize the ratings available from the users for the movies in database in the recommendation process, we built a collaborative filtering based recommendation. Since a user is likely to not having watched all the movies in the dataset, the recommendation system is expected to recommend unseen movies to the users based on predicted ratings.

Collaborative Filtering based Recommendation

Approach

To accomplish this goal, we used the ratings from a small dataset of 9000 movies (measuring the similarity of the large dataset was computationally very expensive and slow). In this dataset there were 671 users with ratings for 9066 movies.

We used Surprise library in Python [1] to predict the ratings using different prediction algorithm and evaluate their performance based on RMSE and MAE. We used different algorithms available in the library package to come with the best fit. Below we present the 5-fold cross validation performance of the algorithms that performed well.

	SVD	**SlopeOne**	**KNNBaseline**
Mean RMSE	0.8968	0.9291	0.8973
Mean MAE	0.6904	0.7114	0.6875

Based on the performance, we decided to use Singular Value Decomposition or SVD for our recommendation system. We trained our model on the entire dataset and created a testset which is comprised of the movies that have not been rated. After predicting the ratings with the prediction model, we create a list of top 10 recommended movies for each user, which looks like following. Note that these movies have not been rated by the user, so we are considering them as unseen by the respective user.

Results

For the first 20 users, the recommendation list looks as following:

```
(1, [4973, 318, 899, 2542, 3462, 1197, 2858, 4011, 48516, 1228])
(2, [6016, 1252, 908, 1203, 3462, 2064, 926, 2318, 1197, 923])
(3, [1221, 913, 2064, 3462, 858, 969, 904, 1252, 923, 1276])
(4, [527, 923, 4993, 593, 1304, 608, 1172, 2571, 48516, 2973])
(5, [3462, 2064, 926, 1945, 904, 1252, 1212, 1224, 232, 2692])
(6, [58559, 3462, 913, 969, 318, 2064, 1203, 3504, 1221, 1223])
(7, [858, 3462, 926, 913, 7502, 969, 1252, 1221, 922, 1304])
(8, [1221, 2064, 1204, 3462, 1193, 1217, 2019, 2318, 2973, 913])
(9, [858, 1252, 1203, 2858, 50, 1228, 296, 1945, 926, 969])
(10, [969, 750, 1221, 3462, 858, 1204, 913, 912, 1172, 922])
(11, [858, 232, 745, 58559, 2064, 1221, 1172, 1208, 904, 1719])
(12, [4993, 3037, 1172, 5952, 58559, 2064, 7502, 260, 1254, 3462])
(13, [858, 2064, 5995, 1172, 1276, 922, 926, 954, 745, 1945])
(14, [3683, 2064, 1617, 913, 318, 50, 750, 1252, 1060, 3359])
(15, [1927, 1104, 3037, 1080, 7502, 89774, 1537, 475, 1278, 31658])
(16, [1212, 926, 858, 2019, 608, 2186, 1304, 953, 899, 1172])
(17, [1208, 994, 1222, 44555, 1939, 46578, 2132, 2318, 5971, 2064])
(18, [858, 2318, 2064, 318, 1193, 1945, 1228, 905, 969, 6016])
(19, [926, 1960, 7502, 2064, 1953, 905, 2973, 2959, 1281, 2300])
(20, [1252, 3462, 3362, 911, 7502, 515, 3468, 4235, 307, 2692])
```

Streaming - online Recommendation System

Problem:

Traditional Training of Recommendation System is not Scalable! Companies, like Netflix are receiving ratings every day from the users, which should be accounted to the recommender system. Nowadays, more than 40% of the night internet traffic is done by video streaming providers like Netflix.
Offline training could be a bottleneck. Offline training is not taking account immediately the new votes, which means it would take until the next training for the Users to be recommended the new Hits. Furthermore, it would be computationally intensive to retrain the whole dataset which is growing exponentially.

Another problem is that the evaluation methods that are used by Recommender Systems are not taking account the recent tastes of the Users. As an example, if a user is a 10-year-old boy, prefers to watch movies for kid. 1-2 years later, the child is growing up and starts watching movies for adults. The problem is that the system is trained to suggest movies for kids and it will take a lot of time until the system understands the new preferences of the User and start making again correct recommendations. How this problem could be solved?

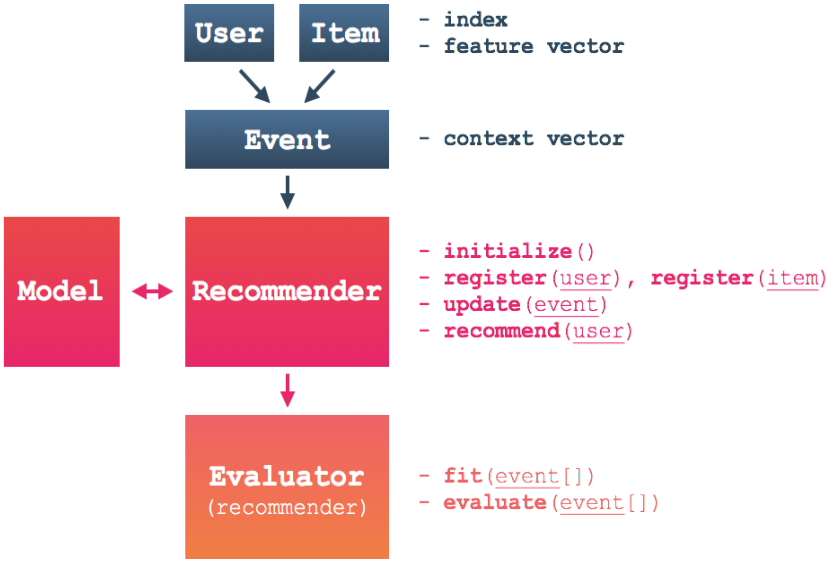
Contribution/Solution:

We designed a new streaming recommendation system, which will be updated continuously, based on the User Votes.

Challenges:

What are the algorithms, frameworks, systems should be used to support Data-Intensive Streaming?

Online updates are done via Structured Streaming.



Each User is an account (example Netflix account)
Each Item is a vote (example Netflix rating vote)

The combination of the User- Vote is the Event which is going to be streamed to the Recommender Engine.

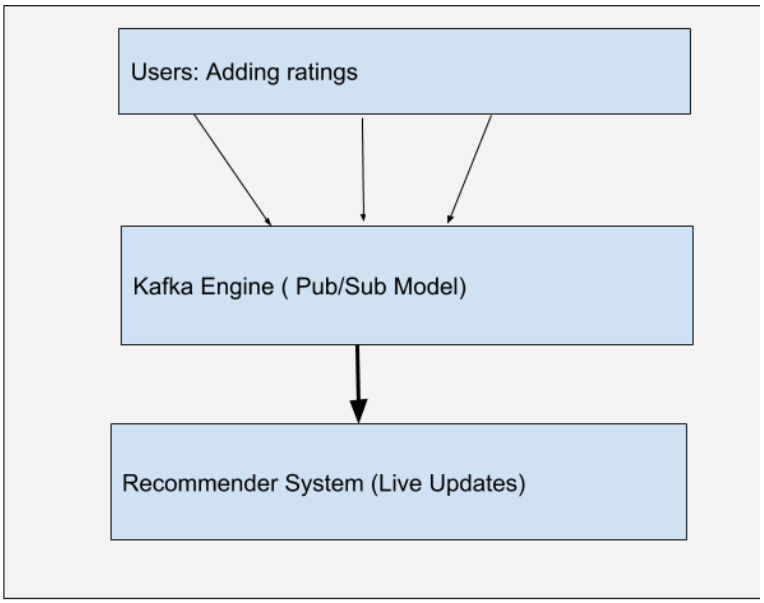
The recommender Engine is updating the Recommendation system based on the model that the Company is using. We decided to use the Collaborative Model and therefore our Recommender engine is using a K-Nearest neighbor algorithm.

System Design:

Based on recode [4] Netflix has now 118 million subscribers!! The system challenge that we had to solve is to make sure that our software will be able to **support up to 118 million concurrent votes and Recommender updates**! How should a system be implemented to support that?

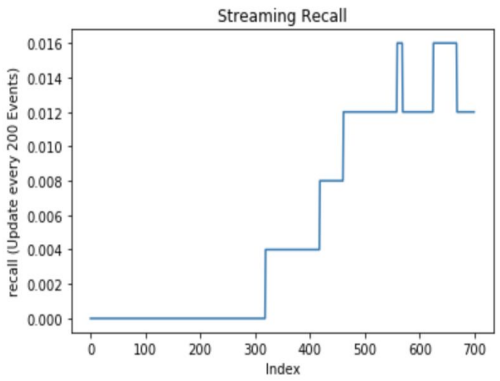
We decided to Use Apache Kafka [5] to read the votes of the Users. Kafka is a scalable low-latency distributed Messaging System which follows a Publish/ Subscribe Design Pattern. Kafka is initially built by LinkedIn to transfer Data across Hadoop Clusters and has been enjoying an incredible success after that, since more and more companies are using this system at their Pipeline. Kafka is spawning Data Brokers across compute nodes to support the incoming traffic. If The traffic is increasing, it is very easy to add more brokers to the system to manage the incoming data rates. The Recommender System, which I described above, will be a

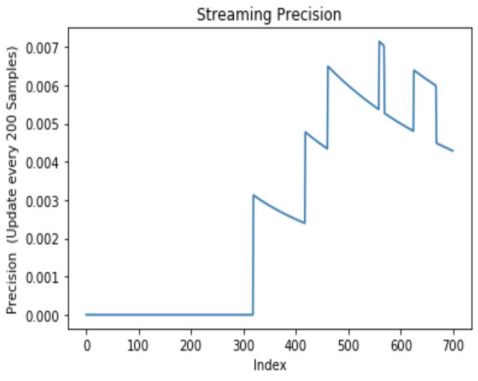
Subscriber at the Kafka cluster. The system will be pulling new votes serially at each own pace in order to update the Recommendation Engine. Since Kafka supports fault tolerance, does not have an end-point of failure, which makes it production ready.

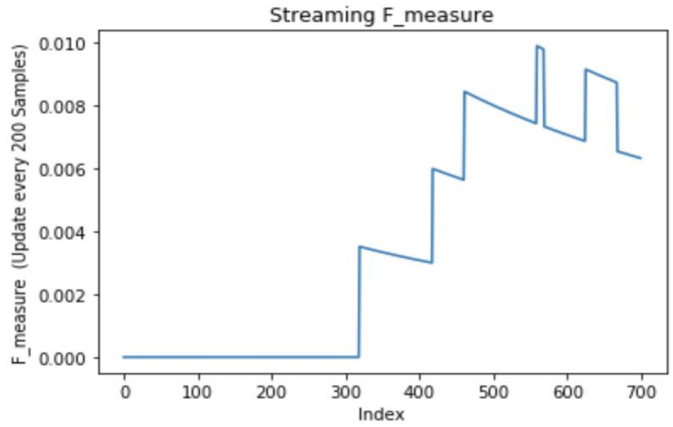


Evaluation Method & Results

For this system, we decided to implement a new Evaluation Method which is called Test-and-Learn . What makes this evaluation powerful is that it is **able to understand the trend and the new preference of the user.** This evaluation method is based on a recent research method [6] and takes into account the last x votes of the User. Therefore, it is able to understand the new trend and also it can continuously monitor the live updates of the Recommender System.







Above we can see the continuous updates of the metrics of our System, which is using up to 700 votes. As we can see, the system with only 300 votes is not able to make right recommendations, but after that the system is increasing its performance, and it is able to identify the latest Movie Trend.

Future Work:

The above system is using the KNN algorithm for the Recommender Engine, which scales only a single Node. A good extension of this project would be to change the KNN algorithm to a parallel KNN implementation in order to improve the performance of the updates.

The parallel KNN implementation could be also-GPU based and can follow the algorithm which is presented on the paper : **Design and Evaluation of a Parallel K-Nearest Neighbor Algorithm on CUDA-enabled GPU** [7]

The current system software is designed in a way to support different Recommender Engines in a plug-and-play manner. Therefore, another idea is to test different algorithms (apart from KNN) and evaluate their performance.

MEMBER CONTRIBUTION

Christos: Revenue prediction, binary classification

Tahiya: Content based and Item based Collaborative filtering recommendation system

George: Streaming recommendation system

References:

1. Surprise, a Python library https://github.com/NicolasHug/Surprise
2. IMDB weighted rating function https://math.stackexchange.com/questions/169032/understanding-the-imdb-weighted-rating-function-for-usage-on-my-own-website
3. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
4. https://www.recode.net/2018/1/22/16920150/netflix-q4-2017-earnings-subscribers
5. https://kafka.apache.org/
6. Incremental Factorization Machines for Persistently Cold-starting Online Item Recommendation (https://arxiv.org/pdf/1607.02858.pdf)
7. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5607480