RUTGERS

The Next Hit

Predicting Movie Success and Building A Recommendation Engine

Capstone Project for Massive Data Mining - Spring ‘18

Team 14

Tahiya Chowdhury, George Chantzialexiou, Christos Mitropoulos

Motivation of the project

- 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.
- There is a need for **recommendation systems**, **movie revenue predictions** and **prediction of a movie’s success/failure** to provide streamlined experience for both users and the companies

Contribution

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**
   b. **Online - streaming** recommendation system

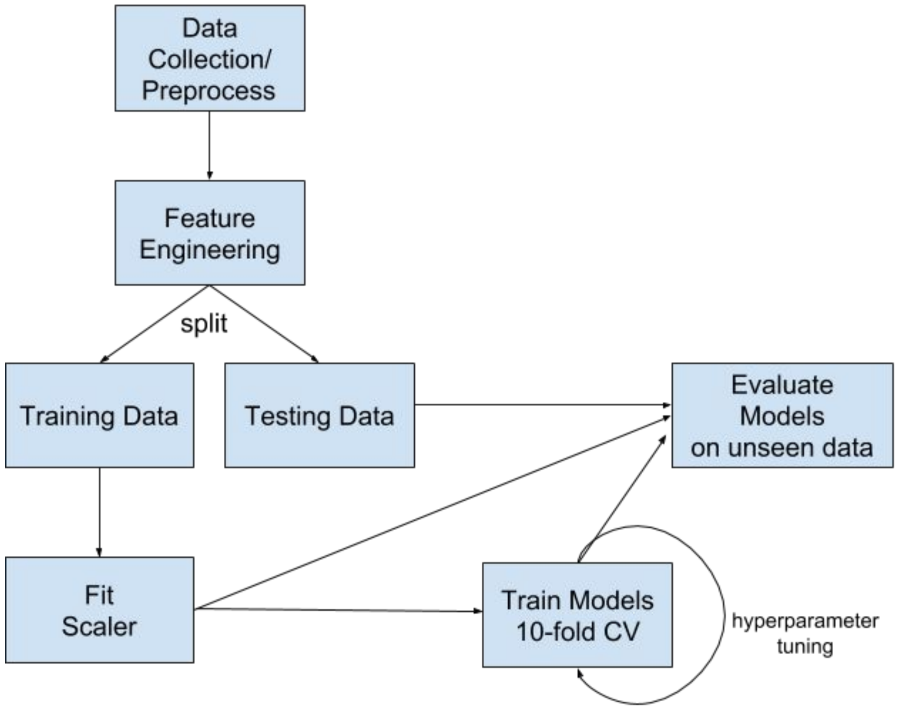
Dataset Description

- An ensemble of data collected from **TMDB** and **GroupLens**. The Movie Details, Credits and Keywords have been collected from the **TMDB Open API** (on or before July 2017.)
- These files contain:
    - Metadata for all 45,000 movies listed in the Full MovieLens Dataset.
      Data points include : **cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.**
    - 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.

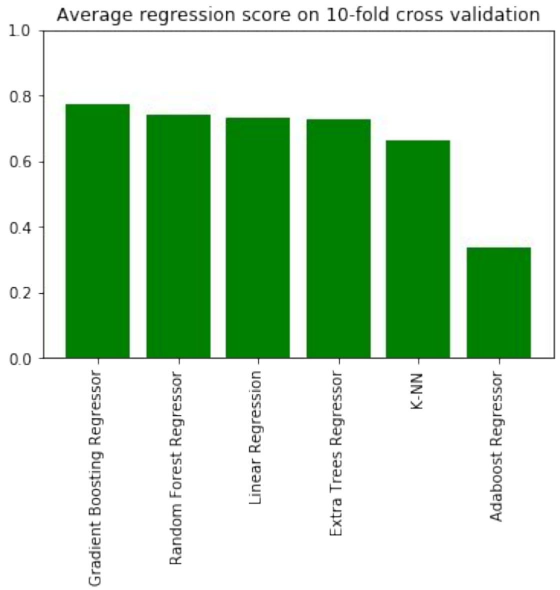
Movie revenue prediction - Binary Classification

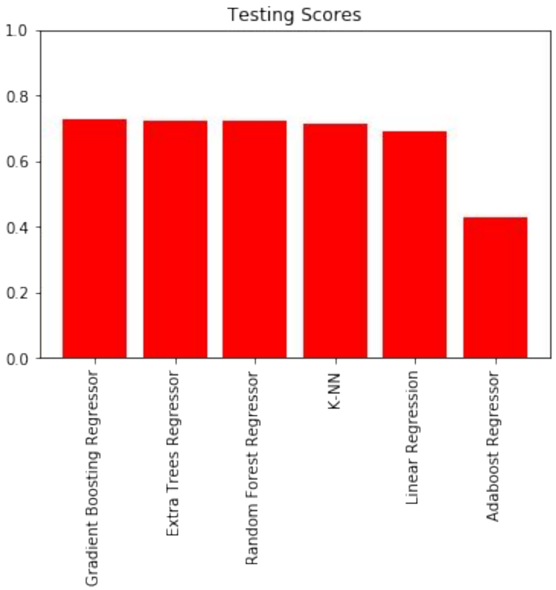
Goal: Facilitate production companies in decision making by building

1. Revenue prediction tool.
2. Classification tool to predict if a movie will make profit or loss.



Results of Movie Revenue Prediction

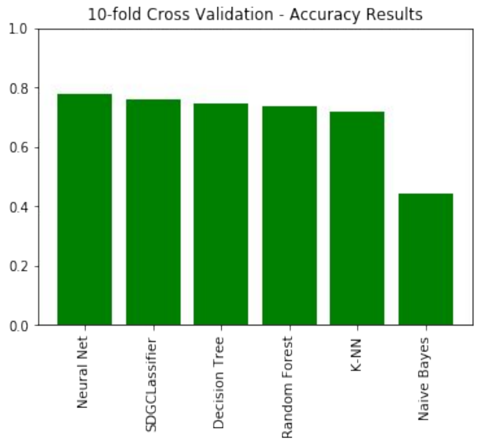


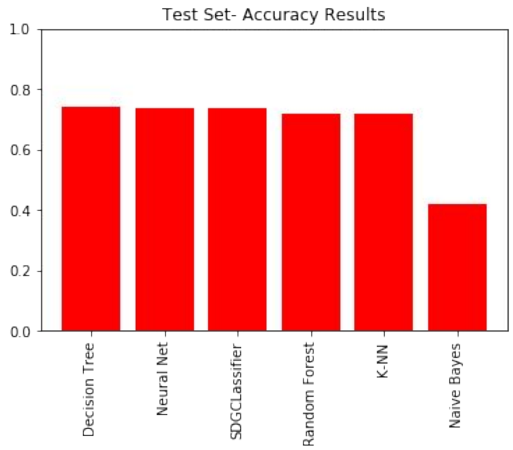


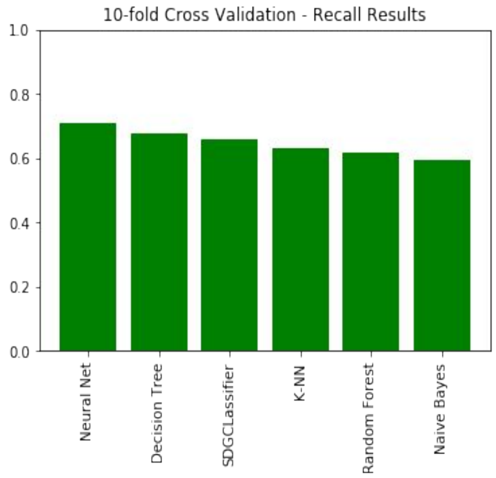
Best performing model is the Gradient Boosting Regressor (CV-score: 0.77, Test Score: 0.72).
Most models, except AdaBoost Regressor, have similar performance.

Note: Score reported is *the coefficient of determination R^2 of the prediction.*

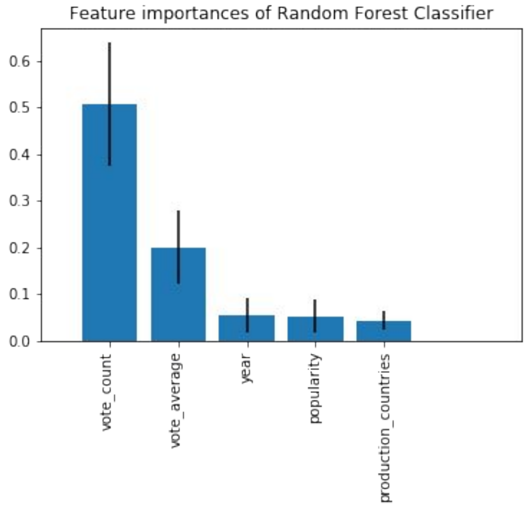
Results of Binary Classification into profit or loss







- **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.
- Feature importance can provide interesting conclusions



Recommender System

Content-based Recommender

- Useful when a new user enters the system and has no ratings (cold start problem)
- Recommends movies based on particular movie or genre

Our approach

- Weighted vote, vote count, vote average, genres from the movie metadata
- Top 3 cast members and director from cast data
- Frequently appeared keywords from keywords data
- Computed Cosine similarity score of the resultant matrix (feature extraction)
- Recommended movies based on user suggested item

Recommendation result for a particular movie

```
recommendations('The Godfather')
```

	title	year	vote_count	vote_average	genres	wr
1199	The Godfather: Part II	1974	3418	8	[Drama, Crime]	7.689586
1186	Apocalypse Now	1979	2112	8	[Drama, War]	7.530356
1934	The Godfather: Part III	1990	1589	7	[Crime, Drama, Thriller]	6.623473
1312	Dracula	1992	1087	7	[Romance, Horror]	6.499201
3635	The Conversation	1974	377	7	[Crime, Drama, Mystery]	6.060771
24284	The Drop	2014	859	6	[Drama, Crime]	5.746547
2025	The Outsiders	1983	293	6	[Crime, Drama]	5.549223
1614	The Rainmaker	1997	239	6	[Drama, Crime, Thriller]	5.513054
8911	Rumble Fish	1983	141	6	[Action, Adventure, Crime, Drama, Romance]	5.430061
754	Jack	1996	340	5	[Comedy, Drama, Science Fiction]	5.137319

Collaborative Filtering based recommender

Useful for recommending movies by predicting ratings on unseen movies by a user

Our Approach

- **Used Surprise library for Python**

	SVD	SlopeOne	KNNBaseline
Mean RMSE	0.8968	0.9286	0.8972
Mean MAE	0.6906	0.7109	0.8972

- Built a test set of movies that are unseen by the users
- Created Recommendation list for each user’s unseen movies based on the predicted ratings

Recommendation result for a particular user

	uid	iid	r_ui	est
2034527	4	73290	3.543608	5.000000
2037288	4	55732	3.543608	5.000000
2034088	4	44197	3.543608	5.000000
2029805	4	1252	3.543608	5.000000
2030579	4	1228	3.543608	5.000000
2030055	4	1221	3.543608	5.000000
2030066	4	912	3.543608	5.000000
2029786	4	608	3.543608	5.000000
2030285	4	318	3.543608	5.000000
2029877	4	50	3.543608	5.000000

Problem

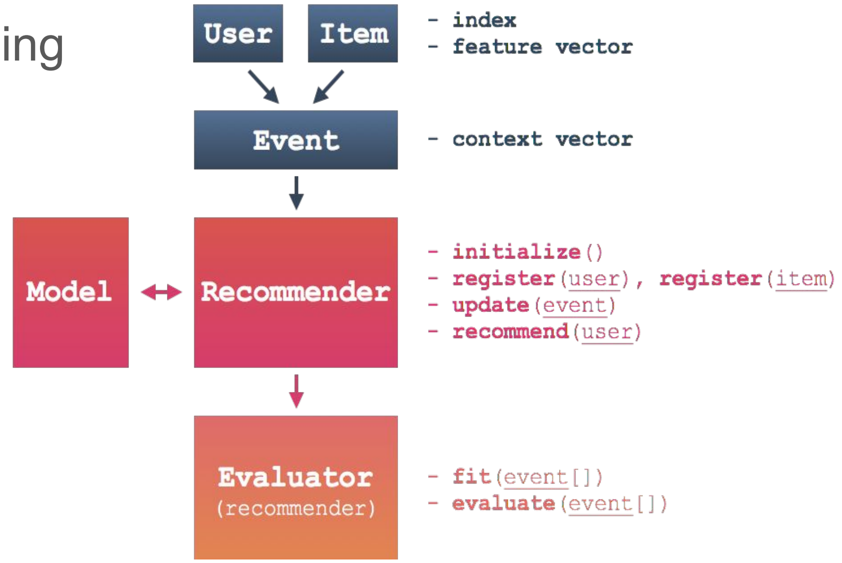
New ratings are coming all the time!

Streaming Recommendation - Goal

- Traditional training method is not scalable
- Companies (Netflix) keep receiving new ratings from Users
  - New ratings should be accounted to the recommender system.
- Users tastes are changing overtime
  - System should be able to evaluate based on the latest choices of the user

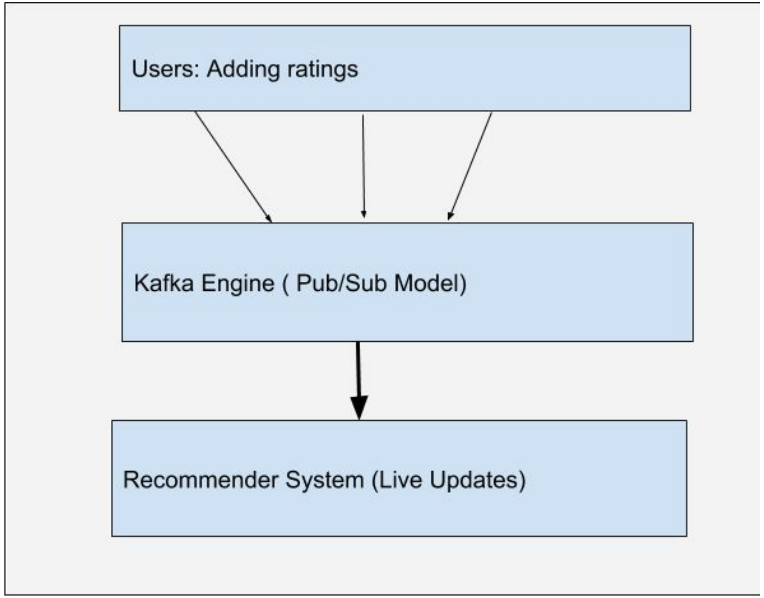
Streaming Recommendation - How

- Online Updating via Structured streaming
- Model:
  - Collaborative: K-Nearest Neighbors



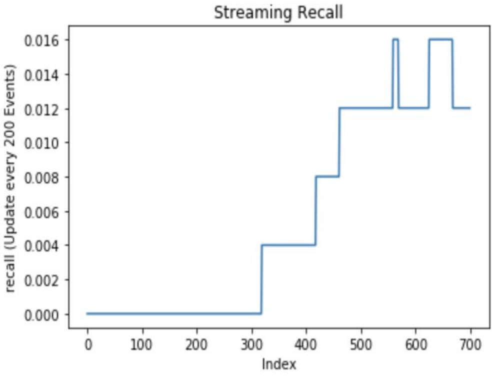
Streaming Recommendation - How

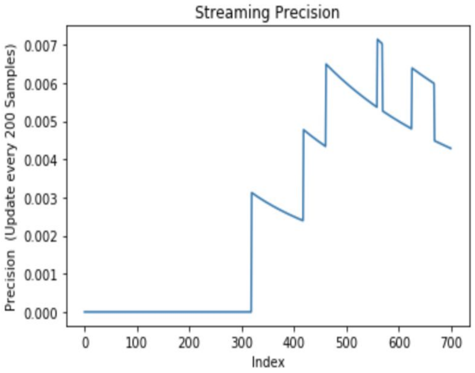
- Streaming Architecture
  - Apache Kafka : Distributed Messaging System
  - Structure Recommender System
- Apache Kafka
  - Push (Input from user)
  - Pull (Recommender pulls new data and update Recommender Engine

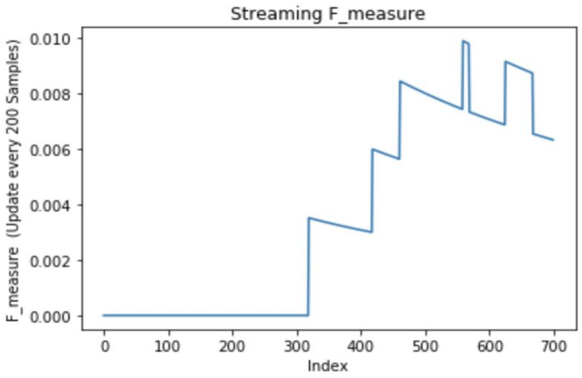


Streaming Recommendation - Results

- Evaluation Method: Test-and-Learn
  - Initial batch training
  - Later: Recommender sequentially launch top-10 recommendation and check if observed item is correctly included in the recommendation list
  - Continuous Monitoring of live updates







Questions?

Thank you!

References

1. Incremental Factorization Machines for Persistently Cold-starting Online Item Recommendation (https://arxiv.org/pdf/1607.02858.pdf)
2. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. (http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html)
3. Surprise: A python scikit for recommender system (http://surpriselib.com)