MovieAdvisor

William & Ole-Mathias, 15.11.2024

***Notes**:*

- *The plan was to host this on a site like Render, but due to the large file size of the model (almost 1GB), the free version couldn't manage it.*
- *The model.pkl file together with the movies.csv file are uploaded to a private AWS S3 bucket, with the access key stored securely via environment variables.*
- *Together with the AWS variables, the OMDb API-key is also stored here.*

This being said, anyone can reproduce the results, but they will need to gain access to the (free) OMDb API, and store the model and movies.csv file on AWS, alternatively skipping AWS and storing it locally.

DESCRIBE THE PROBLEM

SCOPE

Project Goal & Business Impact: The goal of the MovieAdvisor project is to create a machine learning-based recommendation system that suggests movies to users based on genres, later on it is planned for this to also be based on history, preferences and potentially demographics. The business objective is to increase user engagement, retention and satisfaction by providing good and personalized recommendations, which will in return generate revenue through advertisements on the page.

The plan would be to launch this service on a website, and also on mobile apps.

Solution Use: Users can easily access quick and tailored movie recommendations. Many streaming platforms like Netflix and Viaplay provide these on their platform, but to really use them, you will need a subscription. MovieAdvisor is different from this, but it must be said that there are already apps/websites out there that are doing the same, but many of them require you to create a user, which makes you click out of the website immediately.

The **Business Metrics** will be measured by the number of active users, ad impressions/clicks, and how long the user spends on the platform.

The **Pipeline** for MovieAdvisor looks like this:

- **Input Data:** Movie metadata (title, genre, ratings, etc.) and user behavior (e.g., preferences, history) for future versions of MovieAdvisor
- **Recommendation Engine:** Process the input using machine learning to suggest relevant movies. For now, this recommends based on ratings from the dataset and matching it up with input genres.
- **Front-End Delivery:** Displays the recommendations seamlessly to the user through AJAX, in other words, it will display the data without refreshing the page or redirecting.
- **Ad System:** *(NOT IMPLEMENTED NOW)* Integrates with the UI to deliver ads without interrupting user experience. Possibly, there can be 3 max recommendations per user, before they can watch an advertisement to gain 3 more.

Project Stakeholders:

- **Users** seeking quick recommendations.
- **Advertisers** / Google AdMob realistically
- **Developers** maintain and update the platform.
- **MovieLens** providing the dataset
- **OMDb API** for providing movie information

Timeline:

- *Week 1-2:* Data collection and preprocessing.
- *Week 3:* Create a beta version of the service with basic front-end.

Not to be developed yet:

- *Week 4-5:* Work on creating the most optimal model for MovieAdvisor.
- *Week 6:* Complete the back end.
- *Week 7-10:* Complete the website and mobile app.

Resources required:

- *Hosting:* Web hosting service like AWS
- *Storage:* GitHub for code, and larger files (e.g., models) on AWS S3
- *Personnel:* 1-2 developers, with access to elevated machine learning expertise.
- *Financial:* Budget for hosting.

METRICS

Business Metrics:

- ***Active users daily/monthly*** that together with,
- ***Ad revenue*** (click-through rate, impressions) will be higher than the costs of keeping the service running.

Software Metrics:

- Good **precision**, **recall**, **RMSE** (with more) for recommendations (e.g., relevance of suggestions)
- **Latency** of predictions must be **low** (must be near-instant)
- **Uptime** and **response time**

DATA

- **Type:** Movie dataset with attributes; title, genre, IMDb or similar ID and user ratings.
- **Sources:** Public movie datasets (e.g., MovieLens 32M), this will have to be updated every once in a while, to keep up with new movies being released etc. (Harper and Konstan, 2015)

- **Volume:** Initial dataset could be smaller than MovieLens 32M, but it is now using 87585 movies rated by 200948 users. This will be, as said, and have to be updated as time goes on.
- **Privacy:** Minimal, close to none, as no sensitive user data will be collected.
- **Representation:** Features like genre, popularity, and user feedback will require cleaning, normalization, and encoding.

The labels to be used are already given by the dataset, so the ‘ground-truth’ labels are already given to us. There will potentially be a need for engineering the data, although you will need to be careful when doing this, so you don’t over/under sample too much, as this could lead to less popular films becoming more recommended because they only have a few ratings.

It also has to be said that the data used can be misleading as the genres from the dataset are not same as those from IMDb or other sources. The best way to fix this would be to go over every single movie in the dataset and add the common genres between multiple sources, or only keep genres from a reliable source. An example of this problem is “The Cremator”:

- Dataset genres: Sci-Fi
- IMDb genres: Dark Comedy, Satire, Comedy, Crime, Drama, Mystery, Thriller

As you can see, by training the model on the dataset genre, it will not be a realistic recommendation. The IMDb website also includes more genres averagely per movie than the movies in the dataset from MovieLens. This tells us that we would have to work on “feature-engineering” when it comes to the genres.

MODELING

The models to be used in the project are planned to be from scikit-surprise, which “… is a python scikit for building and analyzing recommender systems that deal with explicit rating data” (Hug, 2024). We will plan the baseline to be based on simple models, with minimal preprocessing, in this way the advanced version of the model can be compared to this baseline, and you can see if it’s either better or worse. The baseline will be evaluated on metrics, such as precision, accuracy, RMSE, and similar. The advanced model should **significantly outperform** the **baseline**.

We will have to **review potential mismatches** between the predicted and actual user preferences, for example if the user wanted the ‘Action’ genre only, and the top recommendation only included the ‘Action’ genre as one of 10. The model will be trained on a dataset that only includes the wanted genres and will not include genres that the user set to ‘Don’t include these genres.’

Feature importance will help determine which aspects of a movie are most influential in predicting whether a user will like or recommend a movie. But in this case, there are not many features, and the only ones we can really use are genres and ratings, we can also use tags, as with the MovieLens dataset we get a bunch of tags that the users have placed on each film. These tags could be interpreted for each movie, and we could by doing this classify a movie based on not only genres, but also what the movie is about, this could help recommending similar movies to what the user puts in as data in MovieAdvisor (not yet implemented). But as far as feature importance goes, we won’t get much out of analyzing this for MovieAdvisor.

DEPLOYMENT

The service will be **deployed** using, for example Flask as the web framework on AWS or Heroku. This will serve as an API endpoint for the front-end app.

For **monitoring** the system, we will track the accuracy, server uptime, ad performance, and user feedback from a dashboard or/also via emails.

The system will be **periodically updated** with fresh movies and user ratings to improve the model.

FUTURE STEPS

The future steps for MovieAdvisor would be to improve the genre problem that was talked about earlier in DATA and improve that model. Further models to be created would be for getting recommended movies based on movies and/without genres.

REFERENCES

Harper, F.M. and Konstan, J.A. (2015). The MovieLens Datasets. *ACM Transactions on Interactive Intelligent Systems*, 5(4), pp.1–19. doi:https://doi.org/10.1145/2827872.

Hug, N. (2024). *scikit-surprise: An easy-to-use library for recommender systems.* [online] PyPI. Available at: https://pypi.org/project/scikit-surprise/.