Motivation for project choice

We chose to work with movies as our scope for the project. This was mainly due to us wanting data that is easily accessible with meaningful features whilst still being a topic that we find interesting and affect our everyday lives. We also saw potential to apply what we've learned from the curriculum and previous projects to this space.

Argumentation of choices

Sentiment Analysis

In the domain of sentiment analysis, the utilization of Bertopic was initially attempted; however, its level of success did not meet the desired expectations. Consequently, a decision was made to shift towards the adoption of the naive Bayes approach, which demonstrated greater efficacy and ease of implementation, as determined by our assessment.

Decision Tree Classification - Movie Earnings

We wanted to see if different features on a movie could provide us with a prediction of how much the movie could earn. To this we decided to implement a classification model which uses multiple features to predict one label. The label we decided on was the page which the movie was on. There's 10 pages with 100 moves in each, meaning page 1 is top 100 and page 10 is top 1000. Each page has a range of which the movies have grossed as well as average earnings. We decided that this model was best suited for this purpose based on what we have learned from the curriculum.

Movie recommender

The "movie recommender" has been through many iterations, but in the end we have chosen to settle with these technologies and libraries, to achieve an efficient and functional movie recommender model.

- **Spacy**

We chose Spacy because it is a great NLP library, it is efficient and has many functionalities. It was primarily used to help tokenize the user input, and extract named entities like persons, which in our case was actors.

- **Embeddings**

Here we used Google’s Universal Sentence Encoder for generating embeddings on the movie summaries and actors. It is a pre-trained model that provides high-quality sentence embeddings. This is useful for tasks like semantic similarity comparison and clustering, which this movie recommender uses. We went with a pre-trained model because training one from scratch would take a considerably long time using our scraped movies dataset.

- **MultiLabelBinarizer**

The sklearn MultiLabelBinarizer was used to transform the genre data into a binary matrix. This is a necessary step inorder for the algorithm to be able to understand our genre data.

- **StandardScaler**

The sklearn StandardScaler was used to normalize the data features. This was an important step in ensuring that one feature does not dominate the others, due to its scale, which could potentially lead to bad performance in the recommendation algorithm.

- **NearestNeighbors**

So we used the NN-algorithm to find the closest movies in our feature space to the users input. We chose the NN-algorithm because it is a robust way of similarities on items based on their features.

- **t-SNE**

We used the sklearn TSNE library, to help us visualize the 3D feature space of the embeddings data that our NN-algorithm uses. It was chosen for its great use to perform dimensionality reduction on our movie data.

Outcomes

Sentiment Analysis

The model exhibits an average accuracy rate of approximately 88%; however, it has a tendency to skew towards positive predictions when incorrect. Within the frontend interface, users have the capability to evaluate the model's performance by inputting their own text for testing purposes. Notably, the model has been trained on extensive reviews, indicating that its predictive accuracy improves as the length of the review increases.

Decision Tree Classification - Movie Earnings

The outcome of the movie earnings classifier varies. The testing stage provided us with a low accuracy of 13,5% correct predictions. We think this mainly is due to some features with less importance being weighed higher than features that aren't. In our data of a 1000 movies there’s also not a strict correlation between the features and the label itself (page). Some movies on page 10 can be very highly rated and not earn as much while movies with similar features on page 1 have earned a lot more. Although, depending on the features you choose to change in the client, the model proves to be more accurate than in the test process.

Movie Recommender

The movie recommender model outputs the 10 nearest movies based on the users query, which we then display in the frontend. It also outputs the embeddings data which we use for visualization in our graph to display the 3D feature space of the embeddings, this way one can see the small clusters of movies and what movies were selected. The selected movies are then colored in the 3D graph.