Speaking Notes

Find a film similar to one you like:

* We wanted to complete a project with a practical, everyday application of machine learning. Suggestion algorithms are a common use of machine learning, and a successful movie suggestion algorithm is accessible to the public at large.

Proposed Project Design

* Rather than the current suggestion algorithms currently based on users for specifics apps, we want to build a web-based model that includes a wider array of films. The project does not attempt to incorporate individual user ratings.
* The full project contains a html UI that accepts a movie title as user input, which is sent into the flask application, which will call modules within our K-means machine learning model. The model will return 5 unique suggestions to the application, which will display them dynamically on the webpage.

Data collection

* We have multiple datafiles all sourced from IMDb. Initial data collection incorporated other data from kaggle that inevitably did not work for our purposes and the project had to revert back a few stages to find a more suitable dataset.

Data Cleanup

* Initial data cleanup was conducted to solve several issues, key of which was reducing the size of the dataset for simplicity:
  + Removing translated titles
  + Explicitly specifying US films only
  + Handle duplicate titles by appending the year to the title to ensure uniqueness
  + Removal of TV shows from the dataset
  + Some films have no genre identified – the model relies on this so these titles are removed.
  + Unused information removed from the tables
  + Finally, trim the list of films to only those from 1920 forward.

Database ERD

* All of the datasources are to be linked in our database by one common variable across all datasets.

Database Integration

* The database is constructed from the datasets in Postgre SQL and hosted on AWS. The model explicitly calls this data at the outset.

Building the model (K-means):

* We chose not to use any ground-truthing/training data, so our chosen models are unsupervised.
* For the K-means model, we chose 3 PCA components and assigned the data to 4 clusters. Films in each cluster are assigned a class, and the shortest distnaces within each class are found to generate recommendations

Building the models (hierarchical clustering):

* A second unsupervised model. This one also uses 3 PCA components. 5 clusters are assigned. Same as K-means, the films are assigned a class and the nearest neighbors within each class are determined. These are our suggestions.

Model Metrics (Why K-means):

* For unsupervised learning models, the Davies-Bouldin index provides a reliable metric for evaluating the accuracy of the model. Based on these analyses, the hierarchical clustering model actually produces more accurate results.
* So why are we using K-means? Efficiency. In the time it takes the HC model to process 100 films, the K-means model can go through our entire cleaned dataset of over 100,000 films. When attempting to run the HC model with the full dataset, a warning will appear for the typical user alerting them to the 32gb of RAM required to run the model and that it is not available. The resources required to run this model are not reasonable for the average home user.

Building the webpage:

* The webpage will accept a user input as text, will provide data validation. At the push of a button the app will call the model which will return the suggestions to the webpage. We attempted to incorporate a title suggestion algorithm into the app and html files, but this problem proved difficult to implement a solution for.

K-Means suggestions examples:

* Here are two sample results generated after the model was complete. A user entry of Toy Story provided 5 suggestions that are all Disney and/or Pixar films. Two of them also Toy Story sequels. Entering Goldfinger turned out other James Bond films and 2 other suggestions that appear to be fairly well correlated.

Future Analyses/Project:

* In order to have a more robust model with better suggestions, given more time we would like to have a model that incorporates individual user ratings in order to suggest films based on viewers with similar tastes and viewing habits.
* We would have liked to have implemented some kind of output filtering to restrict suggestions based on genre, studio, years, etc.
* A more visually enticing UI. Specifically implementing movie posters would be ideal. We looked into this option however, and while IMDb has an API, it is locked behind a paywall and not immediately accessible despite that. This was not ideal for a school project.
* It also would have been nice to have the output link to or possibly embed the trailers for the suggested films

Looking Back and Lessons Learned:

* At the project outset we immediately identified the data we wanted to use and moved worward with building the model based on this data. However, it proved to be incompatible with our project goals and we had to take a few steps back to find new data to work with.
* Certain guidelines of the project were not clear and the assumptions we moved forward with proved to be incorrect. Adjustments had to be made to meet the criteria of the project.
* Throughout the course, as a team we wish there had been more opportunities to work as a team to have experience with it.