**Predicting Movie Recommendations by Leveraging Deep Learning and MovieLens Data**

**Project Overview:**

Recommendation systems use consumer data to develop personalized preferences to customers. Tech companies have honed in on this strategy as it has proven to be successful in enhancing user-experience, popular examples include restaurant suggestions on Grubhub, playlist suggestions on Spotify, product suggestions on Amazon, or movie suggestions on Netflix. Recommendation systems typically use clustering, nearest neighbor, or matrix factorization techniques. Deep learning models have recently increased in popularity though to overcome limitations of these methods and increase prediction accuracy.

We are accessing the MovieLens dataset which consists of 100k ratings on 3,900 movies from 6,040 MovieLens users and leveraging deep learning. Our goals include finding new applications and to build better movie recommendation systems that more accurately provide personalized content for the modern consumers.

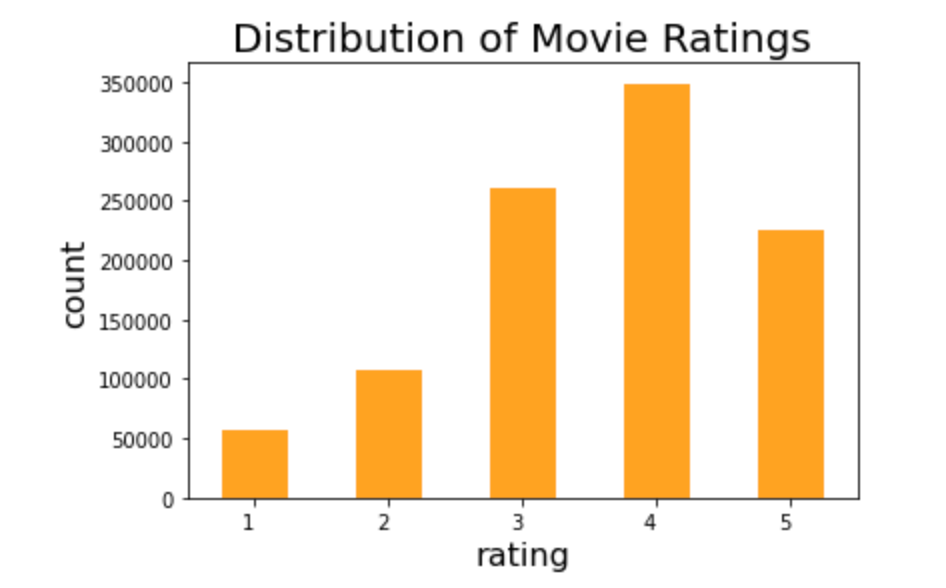
While our main objective is to predict movie recommendations using MovieLens data, our aim is to replicate a previously created model and improve upon it and also find new applications. This first blog post will shed light on the data collection processes, exploratory data analysis, model methodology, and next steps for this project. Please do take note that although changes were made, information was used from the original research project site to assist with our analysis.

Our study is based off of James Le’s research, “[The 4 Recommendation Engines That Can Predict Your Movie Tastes](https://le-james94.medium.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223)”. [James Le](https://jameskle.com/) is a graduate researcher who is focused on deep meta-learning techniques that qualify recommendation systems to attain high performance accuracy, applicable when data is limited, and macro-scale appropriately. We were interested in Jame’s Le research over other research due to many reasons. His approach is clear, transparent, applicable, and achieves quality results. Additionally, his knowledge with recommendation systems is well accredited so we believe his analysis is “state of the art” in terms of what is publicly available.

Our interest in exploring recommendation systems using MovieLens data stems from our desire to learn more about recommendation systems. Recommendation systems are so ubiquitous, we many times do not even notice them. We wanted to know more about the effectiveness of these algorithms, as they not only enhance content and services for a customer but also attempts to draw the most value out of a customer as well. Incorporating deep learning models into recommendation systems has become quite popular as of late, as it can defeat limitations of other algorithms. Although we have never directly learned about recommendation systems yet(as is the last topic to be covered in our class), we wanted to challenge ourselves and utilize the deep learning models we have been taught, and apply those to improve our results.

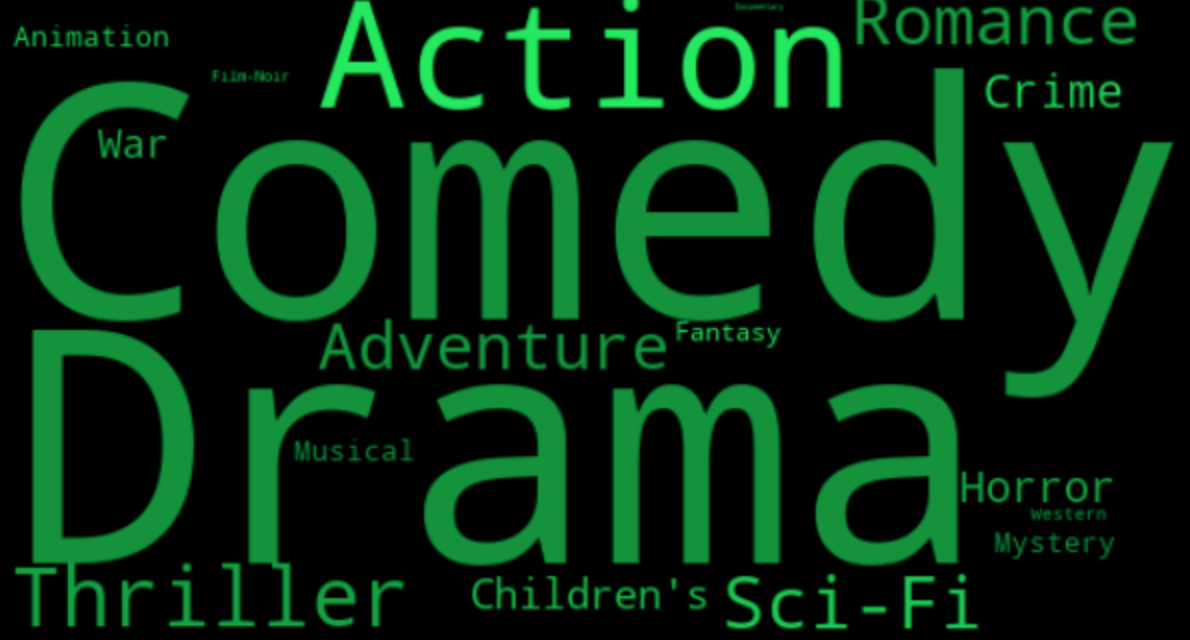
**Exploratory Data Analysis:**

For us to better understand the features in the MovieLens dataset, exploratory data analysis (EDA) was performed. EDA helped to familiarize us with the three datasets used (Movies, Ratings, and Users). It also helped us with the three datasets when they were merged, by capturing common data patterns and to help us with data visualization. Additionally, our EDA process will help us discover if the deep learning approaches being applied are applicable.

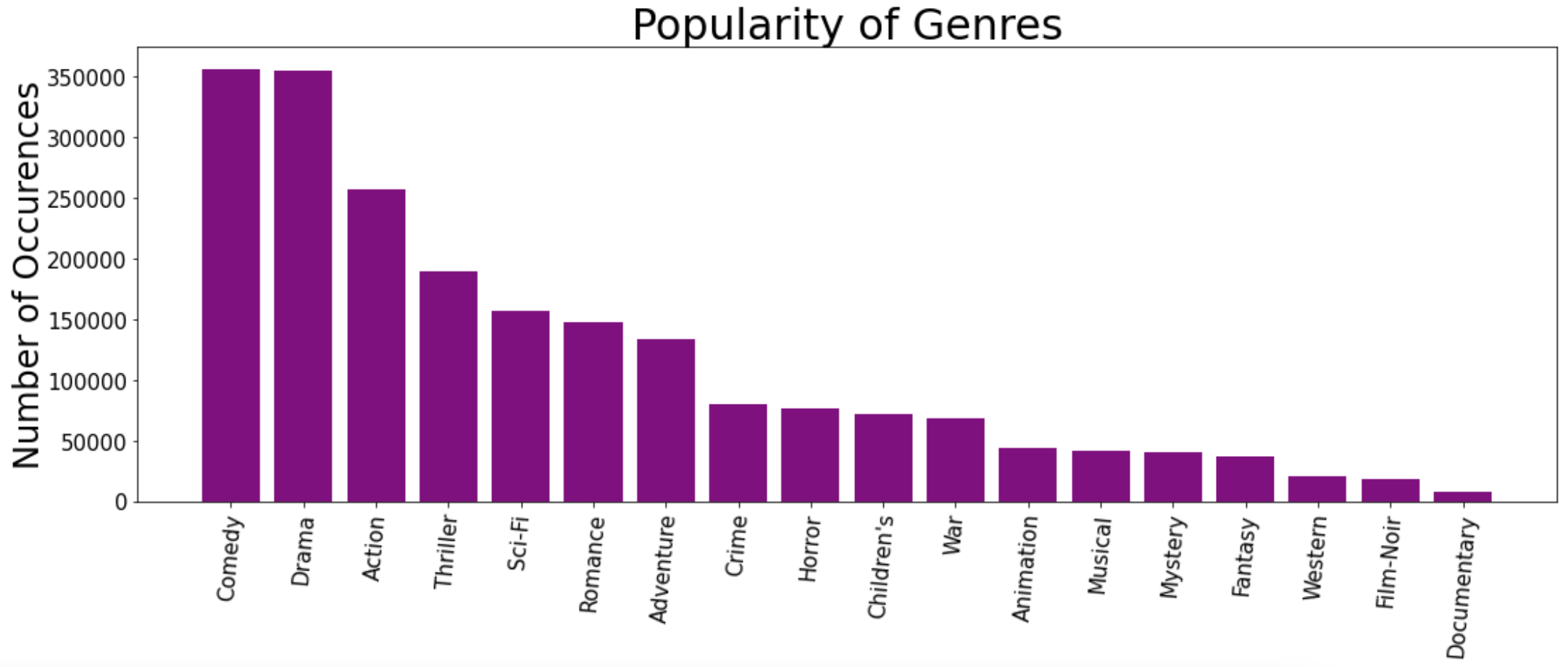
We started by checking to see if any of the three datasets had missing or confusing information, as we would desire to remove missing values or fix any faulty or corrupted aspects of the data. Luckily, no unusable items were found, but we did notice that the genres feature had 301 different types which we will discuss in more detail later on in this blog post. Next, we wanted to get a sense of the distribution of ratings in the dataset. This bar chart reveals the dataset is skewed to the left, slightly imbalanced, and the rating 4 was the most dominant. This is important to know so in the future we can potentially stratify (e.g. using stratified K fold, may try this later for improving the performance) to ensure all of the ratings are adequately represented across training, validation, and testing datasets.

*Bar chart of the count for each rating*

After checking the ratings, we decided to build a Wordcloud visualization and bar chart to see the number of occurrences for each genre. Genre output was edited because there were 301 different categories. Since, some movies had multiple genres (ex. romantic comedy would count as romance and comedy) they were counted as each genre separately. We can see that comedy and drama are the most present genres.This gives us a better idea of what potential bias we might have in the training set. Therefore, we can try to eliminate the bias during our model constructing stage.



*Word Cloud visualization of the most popular genre*



*Bar chart of the number of occurrences for each genre*

**Baseline Model Methodology & Results:**

Our baseline model is based off of James Le’s deep learning model. When reviewing his preprocessing methods we see that his overall approach was limited as he uses individualized approaches in his 4 implemented models. Because we will be focusing on his deep learning approach, we see that he creates a training and validation set by shuffling randomly the values from the original ratings dataset. This is a pretty simple method and it might behoove us to look into more complicated preprocessing methods that can assist with imbalances in the data or maximize on the dataset we have. In terms of EDA, James EDA is quite robust but he could have looked at a few more distributions to visualize more features’ data patterns

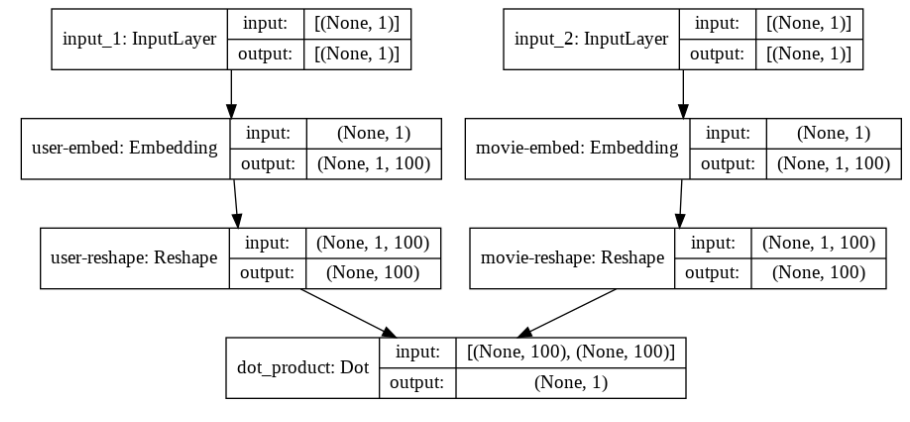
James analysis consisted of 3 different approaches besides deep learning:

* Content-based Filtering depends on alike content or items that are being recommended, so if you are interested in something then you will also be interested in something similar. The more a user takes actions on the recommendations, the more accurate it will be.
* Collaborative Filtering relies mostly on previous actions from a user and is not focused as much on the context, it looks at how alike one user is to other users and then makes recommendations.
* Matrix Factorization recommends content or items to users depending on their search history or previous actions on its past search/activity.

Unfortunately, although each approach has been thoroughly outlined and researched by James, due to time constraints and our focus on deep learning, we will only be improving upon his deep learning approach. By focusing on his deep learning approach, we hope that we are able to better understand this approach and make significant improvements on it.

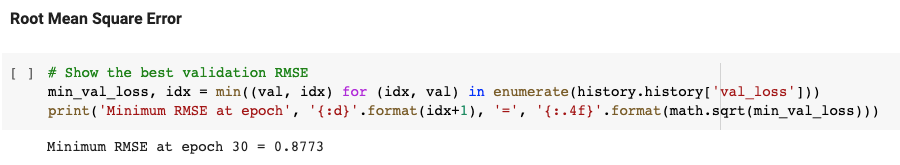
James’ deep learning model is our baseline model. The deep learning model is similar to the idea of matrix factorization, except that to make recommendations, the values from embedding the matrix are captured and learned. By embedding the matrix, a categorical factor is translated to continuous-valued. Efficiency is increased in terms of expressing larger dimensional vectors to a smaller dimensional space.

The image below depicts the baseline structure of the model which uses a sparse matrix factoring algorithm. The model consists of a left side which takes a user id into the input layer and from there the embedding layer develops users through a latent factors matrix, it will output the latent factor vectors for the user. The right side is very similar except it uses a movie id for the input layer and the embedding layer develops movies which will result in latent factor vectors for the movie. Finally, the merge layer outputs a rating based on the dot product of the latent factor vectors users and movies.



*Baseline deep learning model structure*

James chose RMSE as an evaluation metric as it is one of the more favored metrics used to assess accuracy for predicting ratings. We mimic that approach when implementing the baseline model. The model is compiled using MSE as the loss function(take the root at end for RMSE) and the callbacks monitor validation loss so every time improvement occurs, model weights are remembered. We achieved a minimum RMSE at epoch 30 of 0.8773, so the leading validation loss was 0.7627. It is of note that in terms of run time, he used CPU and it took 3 hours, our model used GPU and took about 45 minutes for 30 epochs. We should keep this in mind in the future, as our process seems to be running faster than his.



The minimum RMSE is *0.8773* at epoch 30

In addition to the RMSE calculation, we also include the recommendation from the baseline model that we based our model off of. Please see below for a list of 20 recommendations of unrated movies sorted by prediction value when the user\_id =2000

*Movie Ratings Prediction of User with ID = 2000*

*Recommended Movies for User with ID = 2000 based on Predicted Ratings*

**Next Steps:**

Our next step is to further explore two methods for this project: the “Algorithm” approach and the “Application” approach. Although, we may decide to narrow down to one approach later if we find it more promising.

With the “Algorithm” approach, we aim to improve upon the results from the baseline model. This will be accomplished by further looking into developing more complicated deep learning models to attempt to estimate movie recommendations more accurately. By adjusting the hyperparameters and building and refining our model structure we can improve upon our performance and achieve a higher accuracy score. We will also explore different models for transfer learning, dense layers using transfer learning, adding regularization, changing the learning rate schedules, changing the optimizer, embeddings for matrix factorization and feature creation, and experiment with collaborative filtering.

With the application approach, we will pick a deep learning application and explore how best to solve using different models or focusing on improving one model. Although the author currently implements an embedding model for collaborative filtering, our strategy will be to carry out a different type of deep learning model such as transformers, RNN, CNN. From our research, some applications that have had confounding results include [variational autoencoder](https://github.com/noveens/svae_cf) and a model which incorporates a [transformer layer](https://keras.io/examples/structured_data/movielens_recommendations_transformers/) in the structure to better see the sequential signals behind a customer’s action sequences for recommendations.

In order to continue exploring these technical approaches, we plan to continue researching, reach out to the author, and look at other noteworthy implementations and processes.

**Citations:**

<https://www.onceupondata.com/2019/02/10/nn-collaborative-filtering/>

<https://medium.com/@jdwittenauer/deep-learning-with-keras-recommender-systems-e7b99cb29929>

<https://keras.io/examples/structured_data/movielens_recommendations_transformers/>

<https://github.com/noveens/svae_cf>

<https://le-james94.medium.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223>

<https://fenris.org/2016/03/07/index-html>