Recommending Your Favorite Manwhas, Mangas, Manhuas with a Machine Learning Approach

Sharath Kannan

Eric Cho

Introduction

Welcome to our cutting-edge recommendation system for manhwas, mangas, and manhuas! Are you tired of spending endless hours searching for your next captivating read in the world of graphic literature? Our advanced machine learning models are here to assist you in discovering hidden gems and popular masterpieces tailored to your unique preferences. In case you are unaware, manhwas are comics created in South Korea, mangas are comics created in Japan, and manhuas are comics created in China. These comics have gained immense popularity worldwide. By leveraging the power of data analysis and machine learning, we have created an intelligent platform that understands your preferences and suggests the most engaging and immersive reading experiences. So, how does our system work? It all starts with you. We invite you to embark on a journey of self-discovery by answering a series of questions that allow us to gain insight into your interests, preferred genres, and narrative themes. Your responses will be carefully analyzed and processed by our state-of-the-art machine learning models, which have been trained on an extensive dataset of manhwas, mangas, and manhuas.

Once we have captured your unique preferences, our models will work their magic, employing advanced pattern recognition techniques and collaborative filtering to match you with manhwas, mangas, and manhuas that align perfectly with your tastes. Whether you're a fan of thrilling action-packed adventures, heartwarming romance, mind-bending mysteries, or thought-provoking dramas, our system has got you covered. We also understand that preferences can evolve and change over time. As you explore the titles recommended to you, our system will continually learn from your interactions, adapting and fine-tuning its suggestions to ensure a personalized experience. The more you engage with our platform, the better it becomes at predicting your future preferences and introducing you to captivating stories that you might have missed otherwise. So, say goodbye to endless searching and let our recommendation system be your guide in the world of manhwas, mangas, and manhuas. Your next immersive and thrilling reading experience is just a few clicks away!

Recommender System

There are three types of Recommender Systems that are worth mentioning.

Content-based filtering

- Content-based filtering relies on analyzing the characteristics and attributes of the items being recommended. In the context of manhwas, mangas, and manhuas, content-based filtering would involve examining the content itself, such as the genres, themes, and other textual or visual features.
- The system match content resources to user characteristics. Content-based filtering techniques normally base their predictions on user's information(Isinkaye et. al, 2015), and then suggests items that have similar attributes to the ones they have enjoyed in the past.
- For example, if a user has shown a preference for romance and fantasy genres, the content-based filtering algorithm would recommend manhwas, mangas, or manhuas that contain similar elements.

Collaborative filtering

- Collaborative filtering recommends items by identifying other users with similar taste; it uses their opinion to recommend items to the active user(Isinkaye et. al, 2015). It works by finding similarities between users or items based on their interactions and preferences. There are two main types of collaborative filtering: user-based and item-based.
- User-based collaborative filtering: This approach finds users who have similar preferences to the target user and recommends items that those similar users have liked(Isinkaye et. al, 2015). For example, if User A and User B have similar reading habits and User B enjoyed a particular manhwa, then the system would recommend that manhwa to User A.
- Item-based collaborative filtering: In this approach, the system identifies similar items based on users' interactions. If User A enjoys a certain manhwa, the system will recommend other manhwas that users with similar tastes have also enjoyed.

Hybrid-based filtering

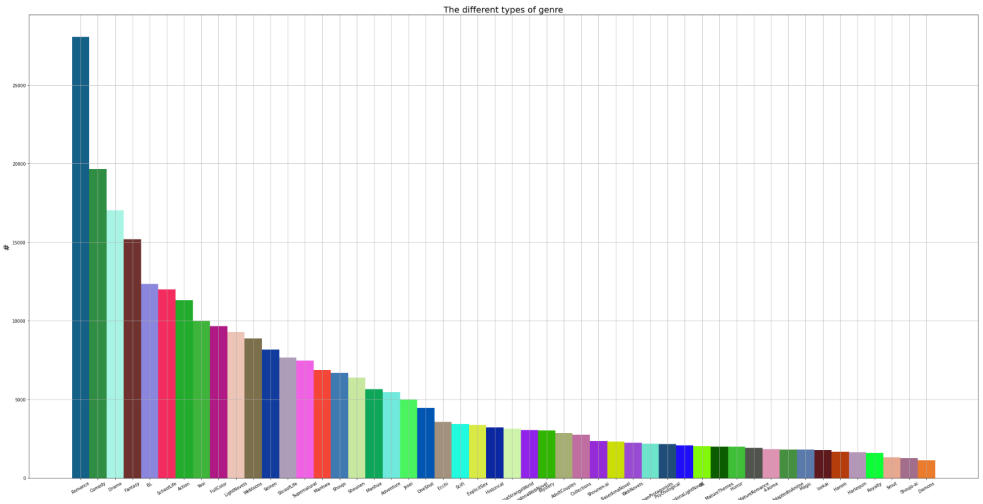
- Hybrid-based filtering combines two or more filtering techniques in different ways in order to increase the accuracy and performance of recommender systems has been proposed(Isinkaye et. al, 2015). By integrating both approaches, the system can leverage the advantages of each method and provide more diverse and accurate recommendations.
- In a hybrid-based filtering system, content-based and collaborative filtering algorithms may work independently or be combined in various ways. For example, the system might use collaborative filtering to identify a set of initial recommendations and then apply content-based filtering to refine those recommendations based on the specific preferences and attributes of the user.

- Hybrid-based filtering allows for flexibility and can be tailored to suit different recommendation scenarios. It addresses some of the weaknesses of individual approaches, offering a more robust and comprehensive recommendation solution

Dataset

The dataset that we used is a list of manga scrapped by Victor Soreiro in 2022 (link: https://www.kaggle.com/datasets/victorsoeiro/manga-manhwa-and-manhua-dataset). One of the first things we did was to find out how much data we were dealing with. We discovered that there were 70948 different anime,manhwa,manga and manhua in the entire dataset. There were 6 columns each with different properties such as title,description, rating, year, tags, and cover.

My partner decided that the best way to clean the data was to remove the ‘cover’ column since the information was irrelevant. Since there are 70,939 unique items available, we decided to get every single anime, manhwa, manga and manhua from the year 2000 and up since we thought that would be more appropriate for the general population. We also decided to drop every NaN value in either the year or ratings. Then sorting the entire dataset by the year. Then my partner and I thought that we should have a visualization such as a bar graph to see the most frequent appearing genres in the entire dataset. I decided that the best way to create a simple bar graph was to first create a dictionary on a key value pair, key being the name of genre and value being the number of genres. With that we chose the top 50 most frequent genres and outputted the bar graph.



As you can see, the most common tags seem to be romance, comedy, drama, and fantasy. There are a few suspicious tags in here, but we decided to keep them in because they are important factors in generalizing the list of manhwas for a good recommendation. Next, we decided to one-hot encode the tags column so the data can work with a machine learning model. The function below creates the one-hot-encoded dataset. From there, we appended that dataset to the main dataset and reset the indices. With that, the dataset is ready for recommendation!

Using K-NN as the backbone of our Recommendation System

K-Nearest Neighbors is a machine learning algorithm that uses the distances between feature values to calculate the nearest items in training data X to a sample Y. Its algorithm calculates the Euclidean Distance between Y and all elements in X, sorts them, and acquires the first K sorted distances. In classification, it labels Y with the most common label of the K sorted distances. In regression, this process is similar. For our case, we thought we could get K good recommendations for a specific manga, manhwa, or manhua by running KNN on the one-hot encoded tag features. In other words, we are using a Content-Based Filtering System to acquire recommendations. We decided to use sklearn's neighbors framework for their Nearest Neighbors model. Since we wanted 10 recommendations (not including Y itself), we decided to set K to 11.

A small drawback with KNN is that distance calculations can take time, especially if the size of X is massive. Luckily, sklearn's KNN implementation comes with a feature that allows us to skip a few calculations. This algorithm is called "ball tree", which "recursively divides the data into nodes defined by a centroid and radius, such that each point in the node lies within the hyper-sphere defined by [the radius and the centroid]" (sklearn, n.d). This in turn results in fewer distance calculations. With every parameter set, we are finally able to set up the model.

After developing a function to output an easy-to-read string representation of the results, we decided to test KNN with a manga called "The Eminence of Shadow ". As seen above, the distances of each recommendation appears to increase as we go down. This is expected of the KNN algorithm, so we have successfully acquired the 10 nearest recommendations for "The Eminence of Shadow". However, there is a problem. **How do we know that these results are good recommendations?**

Evaluating Recommendations

In the example above, we noticed that the tags of all recommendations are very similar to the tags of "The Eminence of Shadow". As we go further down the recommendation list, the similarity in tags start to vanish. Also, when reading the descriptions of the recommended items, we see parallels between the general plot of "The Eminence of Shadow" and the recommendations. The best way to verify the recommendation is by actually reading the recommendations, but we do not have the time for that. All three methods are decent for evaluation, but none of them are rooted in the ground truth.

After discussing this issue, we decided to modify the way we make recommendations. The best way to validate a model is by comparing a feature's value generated by the machine and value from the ground truth. For classification, we look for the loss curve with a loss function. For regression, we would use statistics that compare the generated value to the actual value, such as Root Mean Squared Error.

In a real world setting, a user profile will have the mangas, manwhas, and manhwas that they've read, and a score on whether they've liked it or not. A recommender system would use this information (as well as information from other users if going for a Collaborative-Filtering approach) to provide recommendations for the user. So, we decided to simulate our own "user

profile" with 25 mangas, manwhas, and manhwas we've read, labeled with 1. We then grab 35 random items in prep_df not already in our "user profile", and label them with a 0.

Essentially, we've turned this problem into a classification problem. By using a model that can generalize the trends of tags in our "user profile", we can figure out if a manga, manhwa, or manhua is worth recommending to us. This then raises the opportunity to see which classification model performs the best for recommendation. The three models we will be testing are KNN, Decision Trees, and a Neural Network. The reason for choosing these three is that they seem to be the most popular models in recommendation systems.

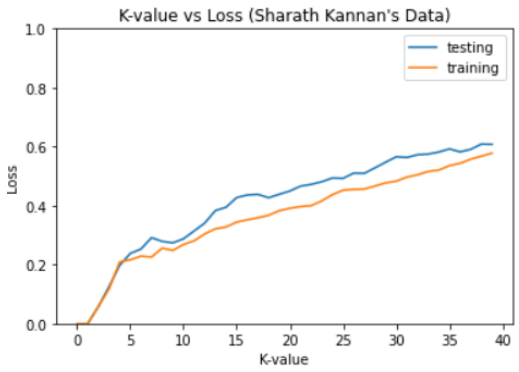
Sharath's taste in manhwa, manga, and manhua appears to be the trope of reincarnating in a fantasy world, and prefers manga. Eric's taste appears to be action-heavy and prefers manhwa. The goal now is to see if these three models can generalize these trends. With that, we can check if a new manga, manhwa, or manhua is suitable for us.**By using the model with the least loss, we can run the model with potential items and find the next best manga, manhwa, or manhua to enjoy!**

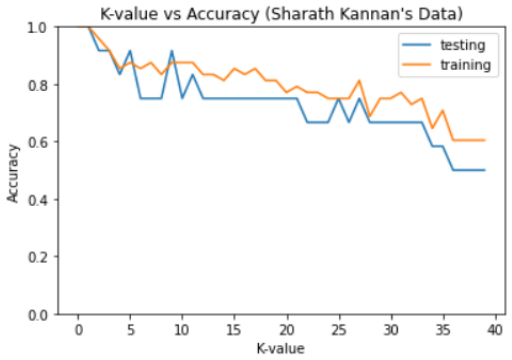
Finding the Best Model

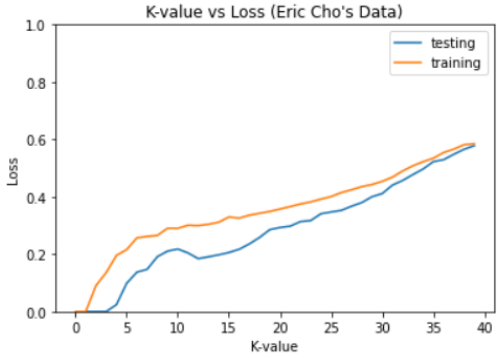
K-Nearest Neighbors

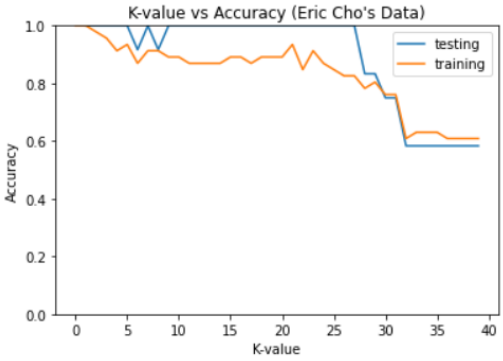
Let us start with K-Nearest Neighbors. Once again, we use the sklearn implementation of the model. This time, we wouldn't have to use the "ball tree" algorithm since we are training with 60 values. Brute forcing the distances would be more efficient. We've also used a technique called "cross validation" to split our "user profile" data into a training and test set (80% and 20% respectively). With this, we can calculate training and testing loss. This can then tell us if the model is overfitting or underfitting. Having K be 10 might not be as optimal here, so we need to choose the best K value that best generalizes the data.

An easy way to do this is to compare model loss for different K values. For our loss function, we decided to go with logistic loss. Below are our results from this experiment.









As expected, as the K value increases, both loss and accuracy decrease. This is because the model is underfitting the data at high K-values, resulting in wrong classifications. We noticed minimal loss for both datasets between the K-values of 0 and 5. The accuracy of the training and testing data for both datasets seems to be following a similar pattern—it is maximized between the K-values of 0 and 5. This tells us that the most optimal K value is within this interval. We see 100% accuracy and 0 loss when K is 1 or 2, but we risk overfitting the model in this case. This means that items that we test with later on have a higher chance of being wrongly classified.

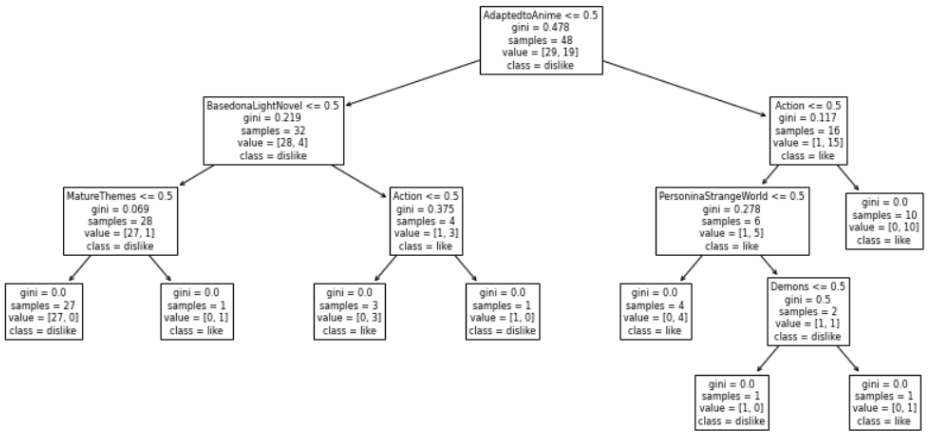
After careful consideration, we decided to settle with K=3. Below are the accuracies of a 3-NN model trained with our datasets. As a result, a 91% testing score for Sharath's dataset and

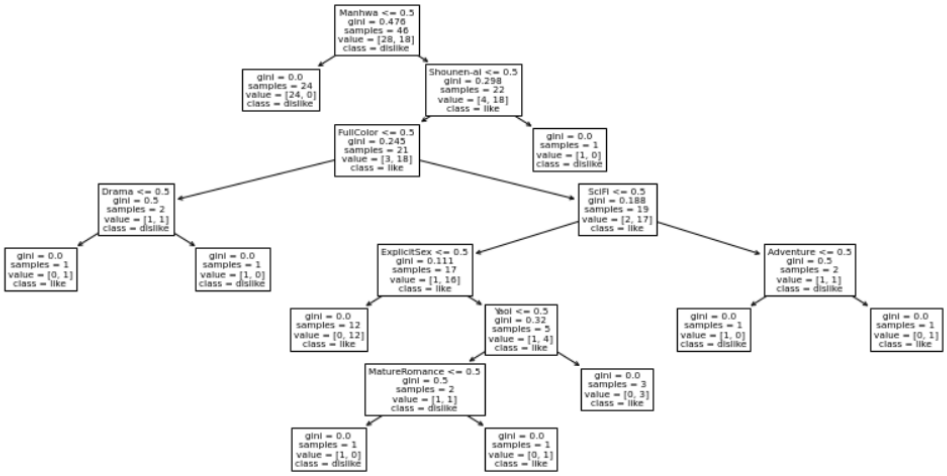
a 100% testing score for Eric's dataset are good values for recommendation. Thus, using KNN to classify a like or a dislike seems like a good idea.

Decision Trees

Next, we decided to test decision trees. When it comes to training a decision tree, it essentially looks for the feature with the highest information gain using GINI and splits items in that dataset based on that feature. This recursive process ends when all items in each child have the same class.

The trends in our datasets can be determined on whether or not a certain tag appears, so we assume that a decision tree would work extremely well to generalize this data and predict a like or dislike. For this, we used sklearn's implementation of a decision tree. Once again, we split our dataset into training and a test set, and trained the model with our training set. We could end the training process early by limiting the height of the tree, but since there are only 48 items to train, we decided against it. Afterwards, we acquired the accuracy of running the model on the training and testing data. Using a visualization of the data, we can see which features the model prioritized. Below are the decision trees made from Sharath's and Erics data.





Something important to note is that the training accuracy for both datasets is 100%. This means that the training data is generalized perfectly. Usually, this means that the model is overfitting the dataset. There is a chance that this occurs with Sharath's dataset as we get a testing accuracy of 83%. Though at the same time, we see a testing accuracy of 100% on Erics dataset. What is the reason for this?

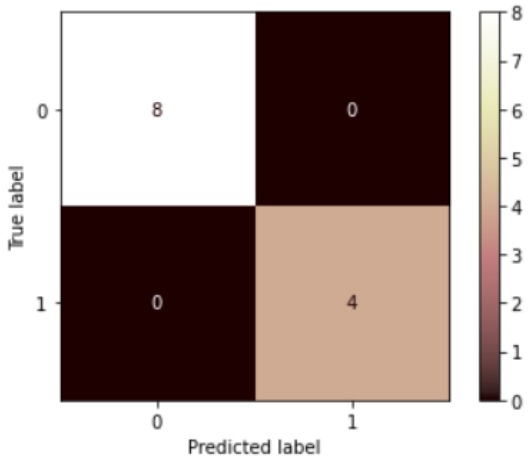
Note that Eric has labeled all the manwha in his list with 1, since he is a manhwa fan. As seen in his decision tree above, the first split is based on the "manhwa" tag. If it is not a manhwa, it is immediately labeled with a 0 class. Meanwhile, Sharath's dataset has more variety in the tags, which is why the trees appear more balanced. That said, we came to the conclusion that a decision tree works better for "user profile lists" that have a defined trend in the tags. It doesn't perform as well for lists with more variance.

Neural Network

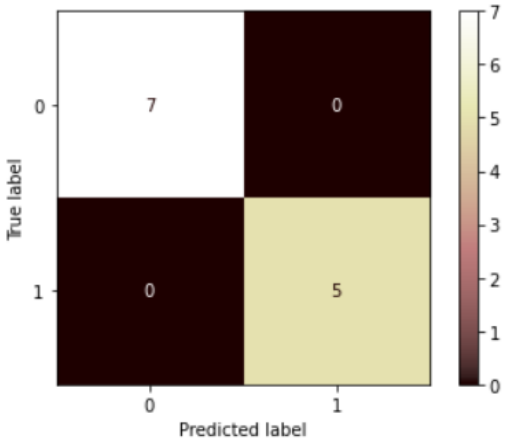
As mentioned before, neural networks are becoming popular for recommendation systems with the advent of big data. The benefit of neural networks is that it is known to classify non-linear distributions with accuracy. Suppose we use a neural network to generalize our "user profile" dataset.

There are many good rules of thumb to start building a neural network. For us, we decided to choose ReLU as our activation function. The algorithm we chose for backpropagation is stochastic gradient descent. A good initial hidden layer size is "2/3 the size of the input layer, plus the size of the output layer" (Krishnan, 2021). In our case, that would be (2/3)*48 + 1 = 33. We experimented with different organizations of these 33 neurons, and we settled with 3 layers: 15 on the 1st, 15 on the 2nd, and 3 on the 3rd. With all that in mind, we set up an MLPClassifier (Multi-Layer-Perceptron) from sklearn and fit our datasets.

For 35 iterations, we see the loss of the dataset continuously go down, which is expected for a well made neural network. For Sharath's dataset, we see a testing accuracy of 91% and a training accuracy of 100%. This might signify overfitting, but with such a high training accuracy, we believe it should be fine. When it comes to neural networks, the more neurons and layers there are in the hidden layer, the more complex the model gets, and the accuracy should be better. So, we decided to try to double the neurons at each layer and see what we get.



As assumed, the testing score accuracy jumped up to 100%. It is also important to note that the runtime of the model does not take that long due to the size of the training dataset. The confusion matrix above also shows us that everything was classified correctly with no false positives or false negatives. So, we decided to use the same neural network build for Eric's dataset.



Again, we see a 100% in training and testing accuracy. The confusion matrix also shows us that there are no false positives or false negatives. The neural network does a better job in generalizing datasets when there is a defined trend as well as datasets where there is a little more variance. We thought that a neural network would need more data points to train and completely generalize the dataset, but that doesn't seem to be the case. Due to the high accuracy and low runtime of the neural network on our "reading lists", we think that the best model to recommend a user to their next manga, manhwa, or manhua is a **neural network.**

Conclusion

In conclusion, our machine learning project has successfully developed a recommendation system that assists users in discovering the best manhwas, mangas, and manhuas best suited to their taste. Throughout the project, we explored various techniques and considerations to enhance the accuracy and personalization of our recommendations. One important future modification that we will be considering is incorporating the ratings and the years correlated with each of the comics. By considering user ratings, we would be able to factor the collective opinions of the readership, giving prominence to highly rated works and potentially influencing the recommendations. This will help us create a better system that can help users discover popular titles they have never heard of.

Furthermore, although we implemented content-based techniques in our recommendation system, we acknowledge the potential benefits of exploring alternative models such as Support Vector Machine (SVM). SVM is known for its ability to handle high-dimensional feature spaces and is effective at classification tasks. Integrating SVM into our system could provide an additional perspective for generating accurate and diverse recommendations.

Although we did not include every single machine learning model in our project to test the variety of results, **my partner and I decided that a Neural Network model might be the**

best choice. This is because the model achieved 100% accuracy for both lists that my partner and I created. This would mean that the Neural Network model is best for generalization and is fantastic for the average reader.

To further enhance our recommendation system, we can acquire multiple user lists by conducting surveys. By gathering data from a diverse range of users with varying preferences, we can enrich our training dataset and enable broader coverage of genres, art styles, and narrative themes. Incorporating these user lists into our system would allow us to fine-tune our recommendations and account for the unique preferences of different user segments.

Lastly, employing a hybrid filtering method allows for the combination of strengths from both content-based and collaborative filtering. By integrating both approaches, we can leverage the advantages of each method and create a versatile recommendation system. This hybrid approach could further improve recommendation accuracy by considering the attributes of the items being recommended.

In summary, our machine learning project has developed a sophisticated recommendation system that takes into account user preferences, and the potential for alternative models like SVM. Additionally, by acquiring multiple user lists and exploring hybrid filtering methods, we can enhance the personalization and accuracy of our recommendations. Through our efforts, we aim to guide users towards their next captivating read in the vast world of manhwas, mangas, and manhuas, ensuring an immersive and tailored experience for every reader.

Works Cited

Isinkaye, F. O., Folajimi, Y. O., Ojokoh, B. A., (2015). Recommendation systems: Principles, Methods and Evaluation. Egyptian Informatics Journal. 16(3), 261-273, DOI: https://doi.org/10.1016/j.eij.2015.06.005

Krisnan, S., (2021). How do determine the number of layers and neurons in the hidden layer? Geek Culture. https://medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3

Scikit Learn. (2011). Nearest Neighbors. https://scikit-learn.org/stable/modules/neighbors.html#unsupervised-neighbors