**Movie Recommendation System**

Priyanshu Jain

Abstract:

Designed a movie recommendation system based on Machine Learning Algorithms.

Methodology and Preprocessing:

Movie Recommendation system is a system that predicts or filters movie preferences according to the user’s choices. The movies can also be predicted based on the genre of movie. Various Machine learning models can be applied for making such a system.

Throughout the course we studied various ML models like linear regression model, logistics regression, KNN , SVM, Xgboost , lightGBM , etc. Now we have to figure out what models out of these we can use in our project. We identified that for predicting or recommending good movies we need to find similarities between the movies.

This led us to use models like **KNN and Neural Networks**. We also made two models from scratch named **Content based and Genre based**.

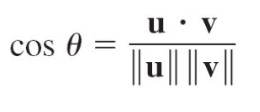
Coming to the preprocessing of the dataset, we could see that the movies dataset has genre column as a string separated by “ | “, this needed to be processed in such a way that it could be used to group the similar genre movies. For this we used an NLP component where we use a tokenizer which converts the strings to vectors with certain words and the word counts in them. This converts the string data to a vector for further analysis. The **Sklearn’s Count Vectorizer** was used for this task.

The other data in movies dataset and the ratings needn't be preprocessed and can be used in the way they are.

# Working and Experimentation:

**Content Based Recommendation:**

This is the first model that was implemented from scratch for the recommendation system. The concept behind the working of this model is basically based on the **similarity between two vectors**.

The angle between the two vectors gives an intuitive approach to how similar the two vectors are to each other in terms of the direction. Hence this would mean if the angle is extremely small then the two vectors are almost identical and if the angle is large then the vectors would vary greatly. As a result the **Cosine Similarity** function of sklearn was used for this as the **output of the cosine function varies inversely with the angle** and hence is more intuitive for this problem.

So, for all the samples of the dataset the cosine similarity is calculated and then based on this the top similar movies are used for recommendation of a given movie. Certain functions were made to diversify the number of recommendations of this model.

**The Neural Network Model:**

The Neural network embeddings is a method to represent discrete categorical variables as continuous vectors. As a learned **low-dimensional representation**, they are useful for finding similar categories as input into machine learning models.

Firstly we **encoded** movie id and user id using LabelEncoder imported from sklearn. Then we normalized the ratings and also splitted rating dataset into test and train. Then we imported all the necessary libraries and defined a function called **RecommendorV1**. Then fitted the train data into it and validated it into the train dataset. Thus model was trained. Then we took random user id . Further we projected the movies user has watched as well as the movies the user has not watched and predicted the labeled movie ids.

And finally unlabelled them into original movie ids.

**The KNN Model:**

The third model implemented for the movie recommendation is the KNN model, where the concept of KNN is used on the dataset. The dataset used here wasn’t directly taken; it had to be modified in order to use it. Here we had decided on using the **ratings as a criteria of grouping**, as in the movies with **similar ratings can be clustered together** to get recommendations of a certain movie in a certain cluster.

For the modified dataset we took the ratings dataset and created a **pivot matrix** out of it with userID on the columns and the movieID on the rows, and the values filled in a particular cell were the ratings given by a certain user to a certain movie. After making the matrix there were cells which had NaN values ( All users didn’t rate all the movies, hence NaN), These were first filled with values of 0 in them.

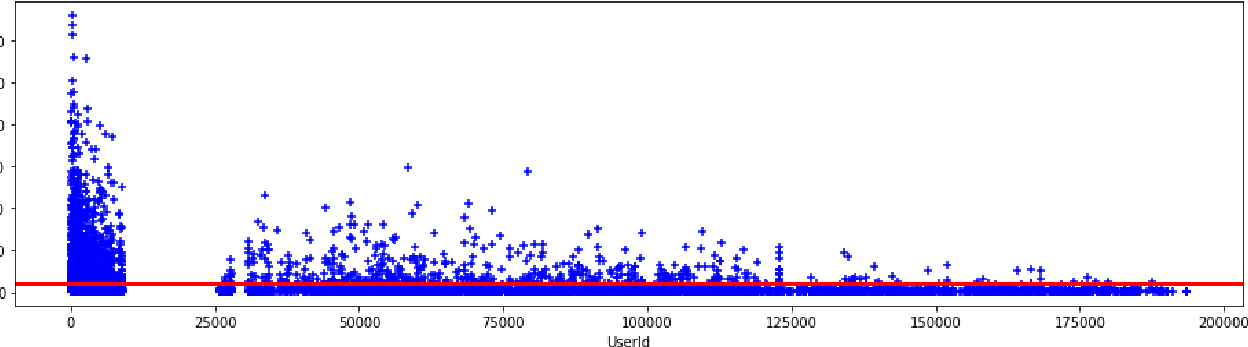
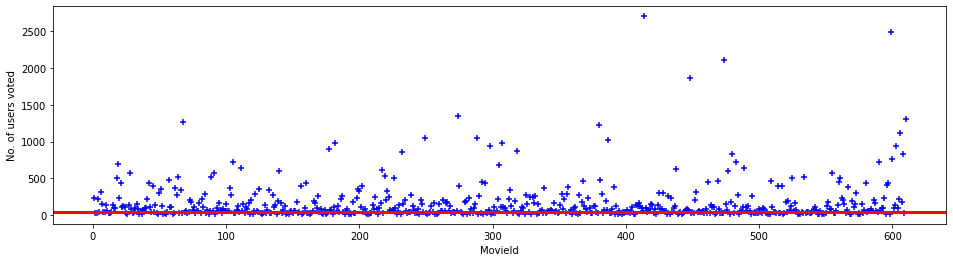
Now the matrix required further processing, as all the movies and all the user data cannot be used and we needed to **set a certain threshold** to filter out the data which had **significance** out of the whole data. The arrays with information on the votes casted by every user and the votes received by each movie were calculated and plotted.

# The votes by each user The votes to each movie

As we can see from the plotted graph the threshold of **10** in the **votes by each user** and **50** on **votes to each movie**, does separate out the majority of the significant data for further analysis.

As there is a huge number of 0 entries in the matrix it needed to be converted into a **CSR Matrix.** This matrix was then fitted in the KNN model and then the main function for recommendation was made and the recommendations were reported.

**Genre Based Recommendation:**



This is the last model for the recommendation system and this was made from scratch. The main idea behind the working of this model comes from **how the K-Means algorithm works**. The K-Means algorithm uses **minimum l2-Norm** as a metric to decide upon which point belongs to which cluster.

This idea was used in this genre based recommendation to find the nearest neighbors. The **norm** was calculated on the **genre array for each movie** and the **cluster center** was the **movie whose recommendations are to be predicted**. Then the algorithm iterated over the desired number of movies (**specified by the user**) to be recommended.

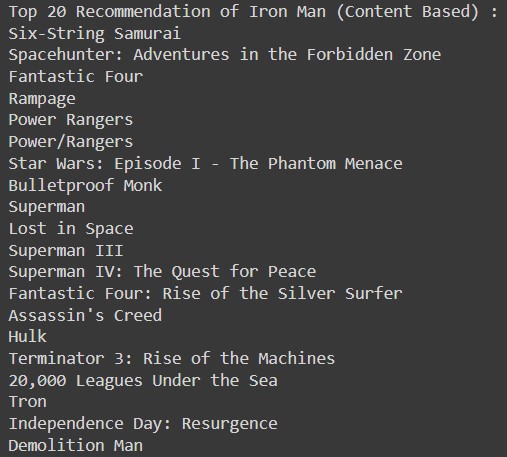
Hence, the recommended movies were then reported.

# Results and Performance Analysis:

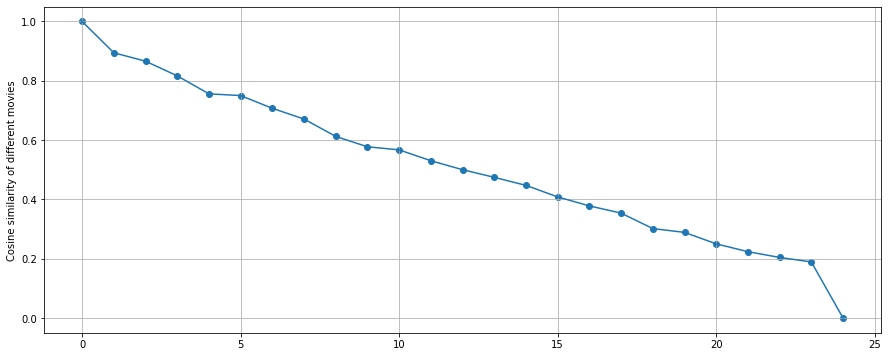
For the uniformity between the results, the recommendation for all the models was done on the movie “Iron Man” and for each model 20 recommendations were made and reported, Except the neural network as it recommends for a particular user.

1. **Content based model:**

The recommendations:

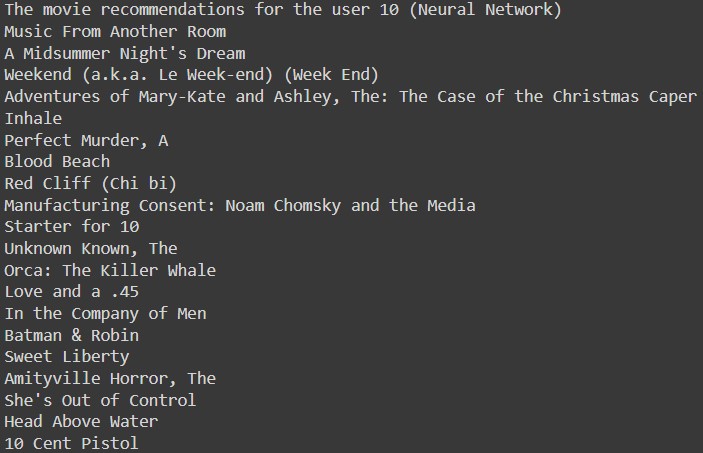


The following was the trend for the cosine similarity of all the movies in the dataset with respect to the movie “Iron Man”:



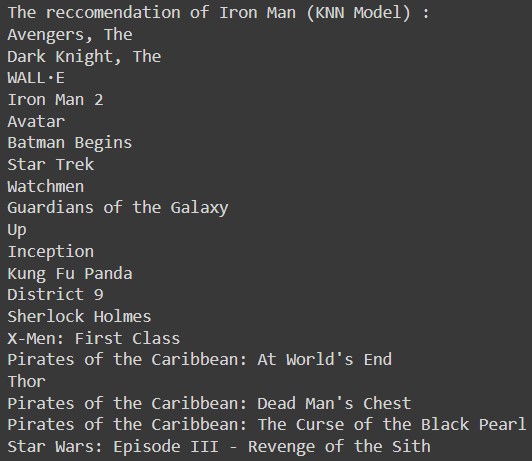
The movies with the **highest values of the cosine similarity** are reported for the recommendations.

1. **Neural network:**

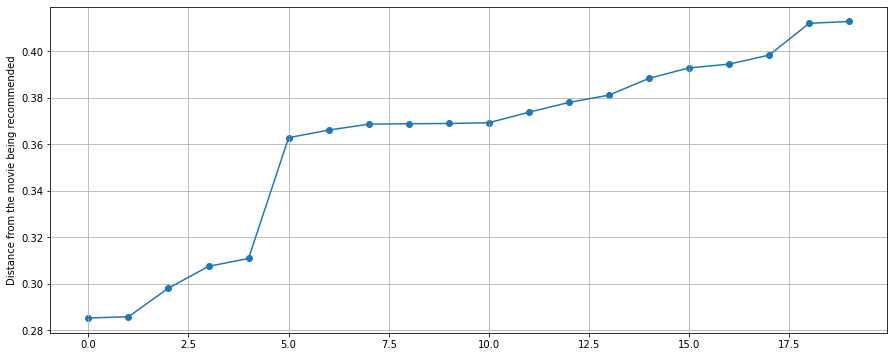


1. **K-NN Model:**

The recommendations are:



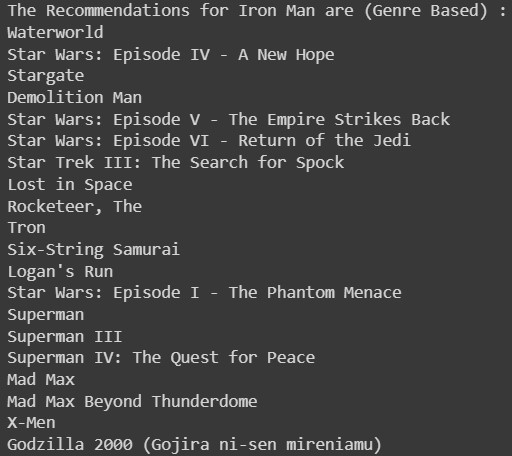
The plot of the distance of each movie in the dataset from the movie that is being recommended:



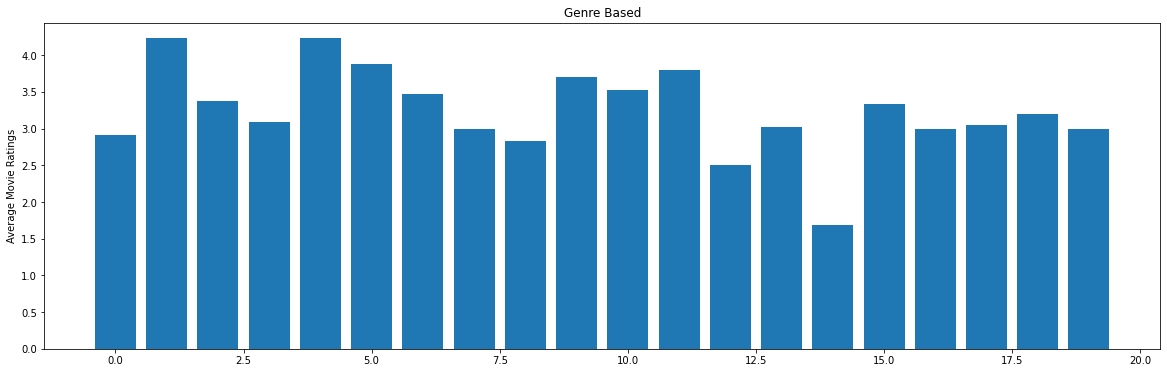
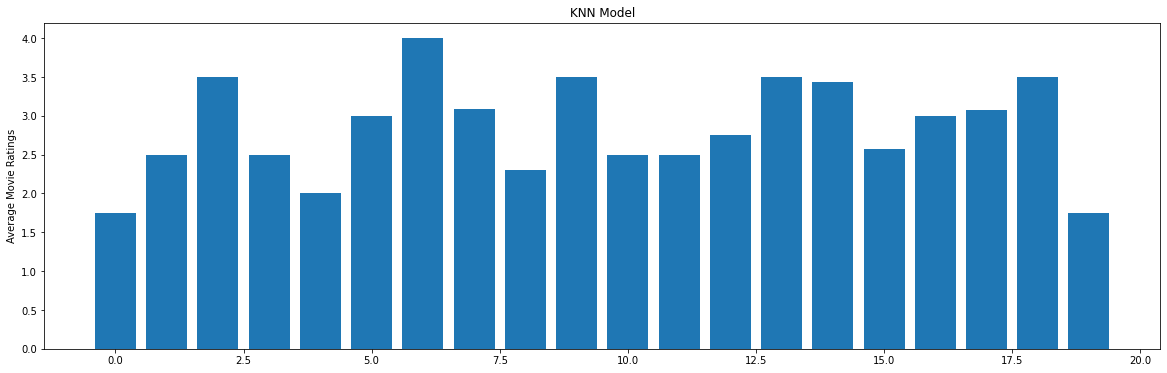
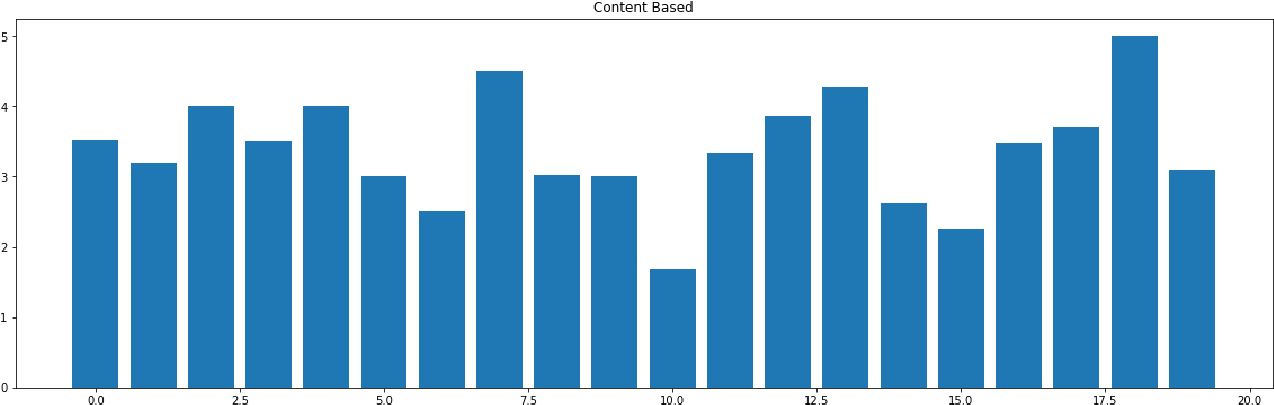
Here the **movies with the least distance** from the **movie that is being recommended** are taken in ascending order and then the top movies are reported as the recommended movies.

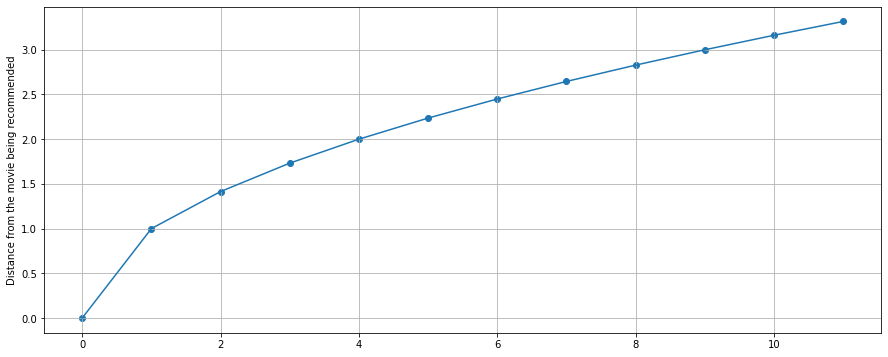
1. **Genre based Model:**

The recommended movies:



The plot of the distance of the each movie from the movie being recommended:





Here the **movies with the least distance** from the **movie that is being recommended** are taken in ascending order and then the top movies are reported as the recommended movies.

Coming to the performance of each model, we can’t directly compare them as there **isn’t any logical metric for comparing them on the same grounds**. Hence, we made a function that takes in the **recommendations of each model** and then **calculates the average ratings** of that movie from the **ratings dataset** and then stores this data for all the movies. This data is then plotted for visualization and **based on the best average ratings of each model the best model is decided**.

And from the average ratings the best model is:

