A close up of a sign

Description automatically generated ****

**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF INFORMATION AND COMMUNICATION TECHNONOLOGY**

**REPORT**

**GRADUATION RESEARCH 1 REPORT**

|  |  |  |
| --- | --- | --- |
| **Supervisor**  **Student**  **Student ID**  **Class** | **:**  **:**  **:**  **:** | **khung doiDo Cong Thuan**  **Pham Trung Hieu**  **20176755**  **ICT 02 – K62** |

**Ha noi, August of 2022**

**TABLE OF CONTENTS**

[**Problem Overview 3**](#_Toc111610594)

[**Review Dataset 4**](#_Toc111610595)

[**Movies\_metadata.csv 4**](#_Toc111610596)

[**Credits.csv 4**](#_Toc111610597)

[**Keywords.csv 5**](#_Toc111610598)

[**Ratings\_small.csv 6**](#_Toc111610599)

[**Cleaning and Feature Extraction 7**](#_Toc111610600)

[**Cleaning 7**](#_Toc111610601)

[**Drop all rows which have non-integer value(s) in the *metadata* dataset. 7**](#_Toc111610602)

[**Drop duplicate ID 7**](#_Toc111610603)

[**Merge datasets 8**](#_Toc111610604)

[**Feature extraction 9**](#_Toc111610605)

[**Machine Learning Algorithm 10**](#_Toc111610606)

[**Problem Formulation 10**](#_Toc111610607)

[**A Straight-Forward Regression Model 10**](#_Toc111610608)

[**A Classification Model 12**](#_Toc111610609)

[**Experimental Results 14**](#_Toc111610610)

[**A Straight-Forward Regression Model Result 14**](#_Toc111610611)

[**Learning Phase 14**](#_Toc111610612)

[**Recommendation Phase 16**](#_Toc111610613)

[**A Classification Model Result 17**](#_Toc111610614)

[**Learning Phase 17**](#_Toc111610615)

[**Recommendation Phase 20**](#_Toc111610616)

[**Issues and Overcome Solution 21**](#_Toc111610617)

[**Learning one user’s model at a time 21**](#_Toc111610618)

[**Problem of Real Value 22**](#_Toc111610619)

[**Future Improvement 23**](#_Toc111610620)

[**Matrix Factorization 23**](#_Toc111610621)

[**Multiclass Classification Model 23**](#_Toc111610622)

[**Try Big Data 23**](#_Toc111610623)

[**References 24**](#_Toc111610624)

# Problem Overview

Nowadays, movies have become an important type of entertainment. Users want to see movies but don’t know which one to see. To help that, I try to recommend related movies that are new to the users by building a recommendation system.

The Recommendation System is a tool that assists users in finding content and overcoming information overload. It predicts user interests and makes recommendations based on the user's interest model. The original content-based recommendation system is a continuation and evolution of collaborative filtering, which does not require item evaluation by the user. Instead, the similarity is calculated based on the information of products selected by consumers in the past, and the recommendation is made accordingly.

Take Netflix for example, Netflix estimates that only 20% of its subscriber video choices come from search, with the other 80% coming from recommendations. “80%” is an incredibly high amount. Netflix believes it could lose $1 billion or more every year from subscribers quitting its service if it weren't for its personalized recommendation engine. And that’s why Netflix thinks that its recommendation engine, however much it could be improved in the future, is already worth so much money to the company.

Inspired by the story of Netflix and other product recommendations, our team will conduct A Recommendation System, namely Movie Recommendation System.

This report introduces a content-based movie recommendation system with the aid of an available [dataset from Kaggle](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset).

# Review Dataset

In this project I use the [The Movies Dataset](https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset) on Kaggle. The dataset has 7 files: **credits.csv, keywords.csv, links.csv, links\_small.csv, movies\_metadata.csv, ratings.csv, ratings\_small.csv.**

However I only use **movies\_metadata.csv, credits.csv, keywords.csv, ratings\_small.csv**

## Movies\_metadata.csv

Here is the fields I will extract from **Movies\_metadata.csv**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Explanation** | **Example** |
| budget | Budget to Product The Movies(in **$** unit) | 30000000 |
| genres | Genres of the movie | {'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 10751, 'name': 'Family'} |
| id | Id of the movie | 862, 987 |
| original\_language | Language use in the movie | en, fr, … |
| production\_companies | Studios participate in production of movie | [{'name': 'TriStar Pictures', 'id': 559}, {'name': 'Teitler Film', 'id': 2550}, {'name': 'Interscope Communications', 'id': 10201} |
| production\_countries | Countries participate in production of movie | [{'iso\_3166\_1': 'DE', 'name': 'Germany'}, {'iso\_3166\_1': 'US', 'name': 'United States of America'}] |
| revenue | Income of Movie | 373554033.0 |
| runtime | Movie’s Runtime in minutes | 81 |
| vote\_average | Average vote of movie on IMDB | 7.7, 4.0, 0.0, … |
| vote\_count | Number of Votes for movie on IMDB | 5415.0, 92.0, 173.0 |

## Credits.csv

Here is the fields I will extract from **credits.csv**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Explanation** | **Example** |
| cast | Actors of movies | {'cast\_id': 2, 'character': 'Max Goldman', 'credit\_id': '52fe466a9251416c75077a8d', 'gender': 2, 'id': 6837, 'name': 'Walter Matthau', 'order': 0, 'profile\_path': '/xJVkvprOnzP5Zdh5y63y8HHniDZ.jpg'} |
| crew | The movie’s managers | {'credit\_id': '52fe466a9251416c75077a89', 'department': 'Directing', 'gender': 2, 'id': 26502, 'job': 'Director', 'name': 'Howard Deutch', 'profile\_path': '/68Vae1HkU1NxQZ6KEmuxIpno7c9.jpg'} |
| id | Id of Movie | 862, 987 |

## Keywords.csv

Here is the fields I will extract from **keywords.csv**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Explanation** | **Example** |
| keywords | keywords about movie | [{'id': 949, 'name': 'terrorist'}, {'id': 1562, 'name': 'hostage'}, {'id': 1653, 'name': 'explosive'}, {'id': 193533, 'name': 'vice president'}] |
| id | ID of movie | 862, 987 |

## 

## Ratings\_small.csv

Here is the fields I will extract from **ratings\_small.csv**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Explanation** | **Example** |
| userId | user’s ID | 1, 2, …, 671 |
| movieId | ID of movie | 862, 987 |
| rating | Rating user rated on movie | 0.5, 1.0, …, 4.5, 5.0 |

# 

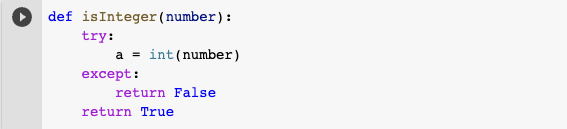
# Cleaning and Feature Extraction

## Cleaning

In the cleaning phase, I use the pandas package to load, manipulate and store the datasets.

### Drop all rows which have non-integer value(s) in the *metadata* dataset.

I begin by writing a simple function to check an integer number:



Then use this function to find all the rows in *metadata* that have non-integer value:





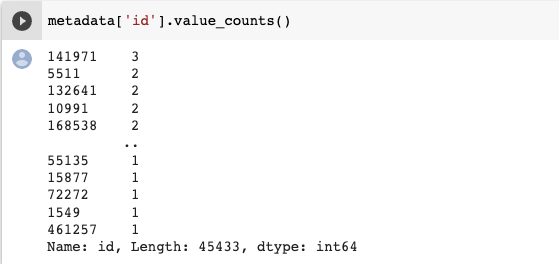
After that, I drop those rows by ID and return None:



### Drop duplicate ID

Since all records should have their own ID, I have to drop those having duplicate ID.

First, we take a look at the number of times an ID appears in *metadata*:



(ID 141971 appears 3 times whereas 5511 appears twice)

Now we know that there are duplicate rows in *metadata*, I proceed to drop those duplicates:







Finally, return the *metadata*, *credits* and *keywords* with all duplicate IDs removed.

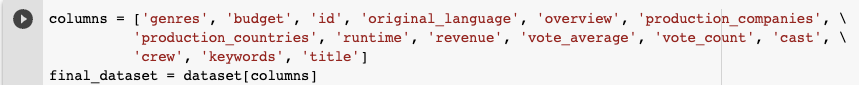
### Merge datasets

In the following step, *metadata*, *credits* and *keywords* are merged into *dataset* with a database-style join on the ID column:

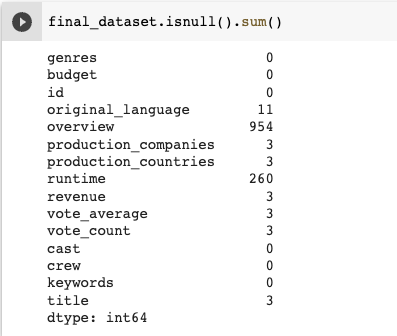


Since our project only works with *genres, crew, cast, keywords, popularity, vote\_average, language, runtime, vote\_count, production\_companies, production\_countries, overview*,

the *final\_dataset* uses only those features as its columns:



However, the *final\_dataset* still contains *null value* fields:



Therefore, I have to fill in those null value indexes before ending the cleaning phase.

As *overview* and *runtime* take up most of the null field, I only handle these 2 columns:

* Fill each *runtime* null value with the mean value of *runtime* in *final\_dataset*



* Fill each *overview* null value with an empty string in place of ‘null’



## Feature extraction

Feature Extraction codes are in file **profiles.ipynb** or **profiles\_colab.ipynb**

There is a module FeatureExtractor.py

This module use **genres, production\_companies, production\_countries, cast, crew, keywords** columns to generate features including **CastsRank, NumLeadActors, HasTop50Actors, NumCrews, crewsTeamRank, NumTopCrew, HasTopCrew, NumStudios, StudioRank, NumTopStudios, HasTopStudio, CountryRank, keywordRank, Action, Adventure, Animation, Comedy, Crime, Documentary, Drama, Family, Fantasy, Foreign, History, Horror, Music, Mystery, Romance, Science Fiction, TV Movie, Thriller, War, Western**

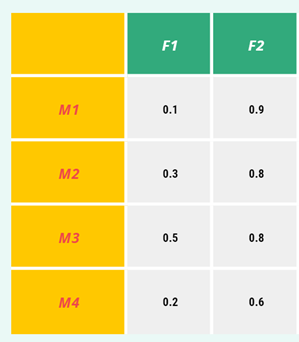
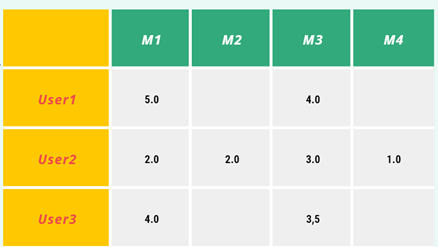
The generation is based on the revenue of movies to determine how famous casts, crews, production\_companies are. For **genres**, it is one hot encoding.

# Machine Learning Algorithm

In this project, I build a **Content Based Recommendation System**.

## Problem Formulation

Given profile of movies **M**, **R** true ratings by users to some movies.



**M** is the profile of movies with shape of (2830, 39) which stands for 2830 movies, 38 features and 1 intercept factor(all ones). This matrix is standardized from the original feature extracted dataset to the mean of 1 and standard deviation of 0 before adding the intercept factor 1.

**R** is true ratings by users to movies with shape of (671, 2830) which stands for 671 users and 2830 movies.

Will find the user model **U** and **b** which I will call a matrix with shape of (671, 38) which stands for 671 users and 38 features. Each row is the model of each user, which can be called “user’s interest”. And **b** is a matrix with shape (671, 1) the biased (intercept).

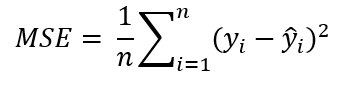
## A Straight-Forward Regression Model

First predict the ratings by:



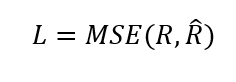
is the predicted ratings with same shape as of 671 x 2830, operator \* is matrix multiplication and [U,b] means that they are concatenated axis = 1. (Stack horizontal and in this case the new shape is **(671, 39)**)

After getting the predicted ratings, I use Mean Squared Error (MSE) to define a loss for our model.

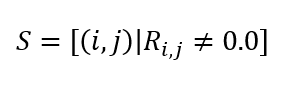


Use MSE because it penalizes the output that is far from the true output.

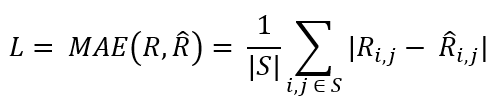
For our case:



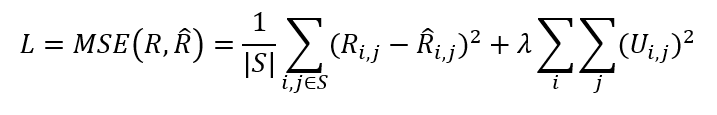
Let **S** is the list of pairs of indexes of movies that user rates for, this will not include the index of the unrated movie.



The loss MAE becomes:



Add more regularization term L2,



Here I apply L2 regularization on U only, not on the intercept b.

After I have the loss, compute: 

With the partial derivatives of U I can update U after each loop to train U.

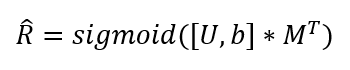
In the implementation in pytorch, I use the Autograd module of Pytorch to automatically calculate the derivatives and use the built-in Adam optimizer to train for params(**U**). The main MSE loss will use **MSELoss()** and regularization terms will be replaced as **weight\_decay** param.

The metrics is also MSE Loss (Mean Squared Error).

## A Classification Model

In this model **R\_hat** will not be the predicted rating any more. It will be the **probability** that the user will rate more than 4.0 for the movie. In the file Content\_Based\_Classification.ipynb I will transform the ratings dataset to 1 and 0 for the rating field.

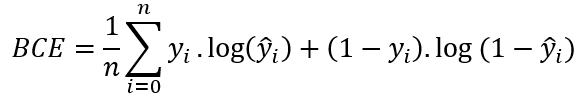
First predict the probs by:



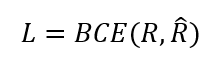
is the probability matrix with same shape as of 671 x 2830, operator \* is matrix multiplication and [U,b] means that they are concatenated axis = 1. (Stack horizontal and in this case the new shape is **(671, 39)**).

Sigmoid is applied to all elements of a matrix after the multiplication of 2 matrices.

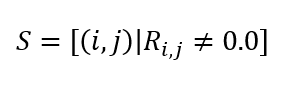
After getting the probability matrix, I use Binary Cross Entropy to define a loss for our model.



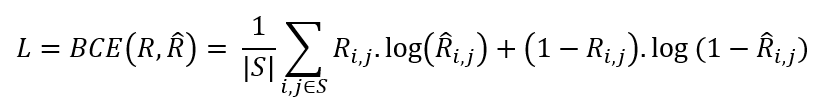
For our case:



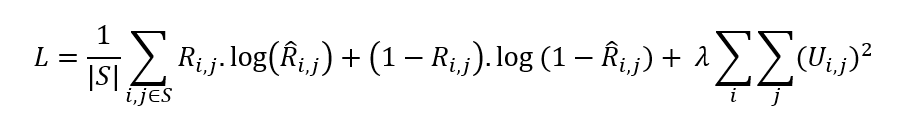
Let **S** is the list of pairs of indexes of movies that user rates for, this will not include the index of the unrated movie.



The loss MAE becomes:



Add more regularization term L2,



Here I apply L2 regularization on U only, not on the intercept b.

After we have the loss, compute: 

With the partial derivatives of U we can update U after each loop to train U.

In the implementation in pytorch, I use the Autograd module of Pytorch to automatically calculate the derivatives and use the built-in Adam optimizer to train for params(**U**). The main Binary Cross Entropy Loss will use **BCELoss()** and regularization terms will be replaced as **weight\_decay** param.

The metrics I use use **Accuracy** which is the number of output guessed right.

# Experimental Results

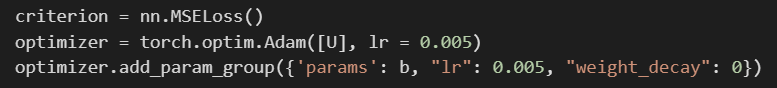
## A Straight-Forward Regression Model Result

### Learning Phase

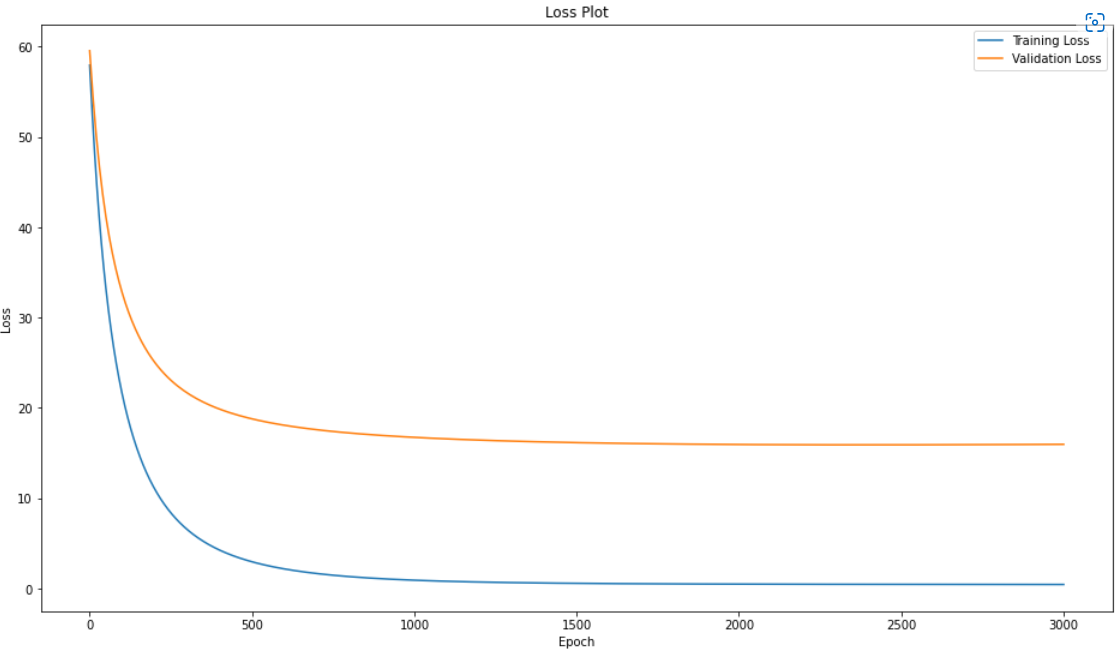
After splitting the rating dataset into two parts **ratings\_train.csv** and **ratings\_test.csv,** we use them as train dataset and validation dataset.

Using MSELoss and Adam Optimizer.

* Without regularization



Take EPOCH of 3001, here is the plot of **MSE** loss.

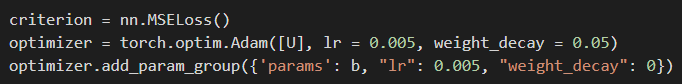


MSE Loss Min on Train Set: **0.467**

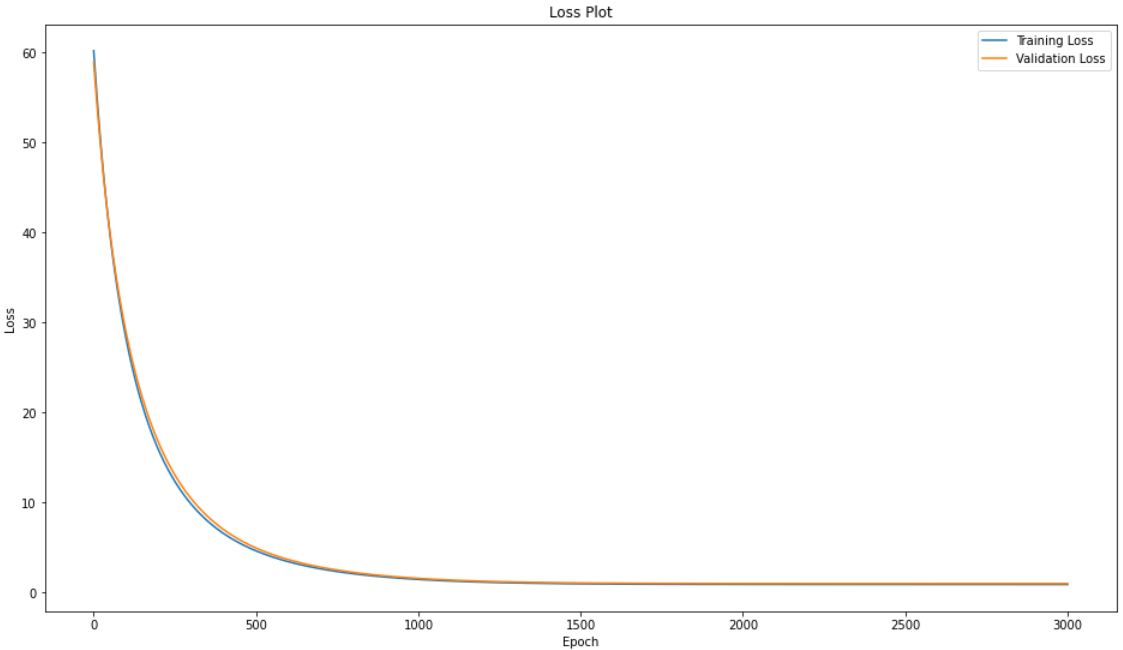
MSE Loss Min on Validation Set: **15.95**

The model overfit the train data.

* With regularization lamba of **0.05**



Take EPOCH of 3001, here is the plot of MSE Loss while training:

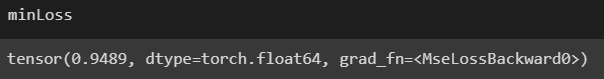


MSE Loss Min on Train Set: **0.85**

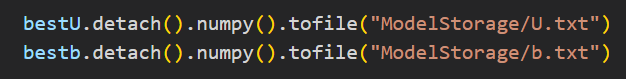
MSE Loss Min on Validation Set: **0.95**

Regularization has reduced overfitting.

I save the model with the lowest **MSE loss** on the Validation Set.



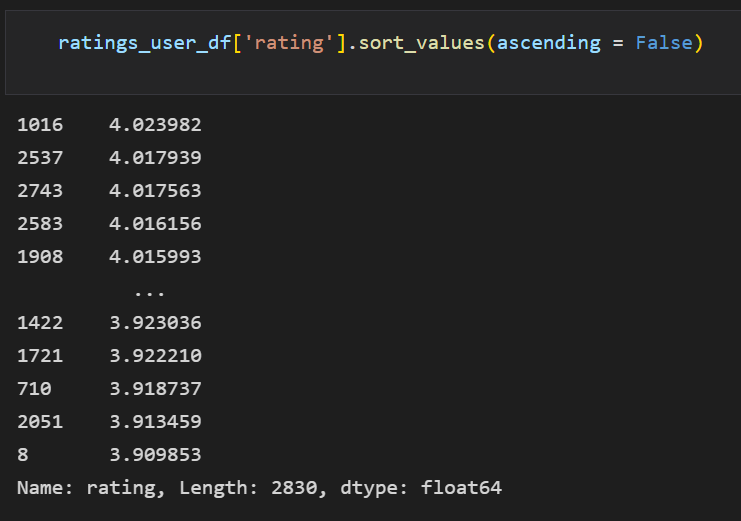
The parameters to save are U and b



### Recommendation Phase

Specify **user = 10**

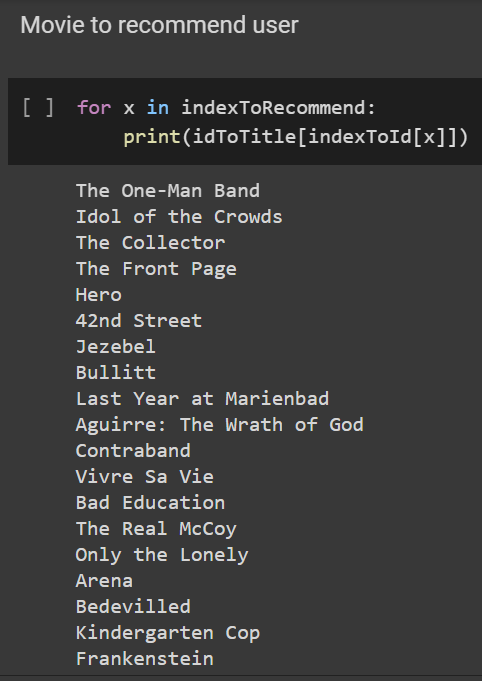
Predict the ratings of the user for each movie and sort decreasing order.



After that remove from that list the movie that the user has seen.



Take the first 20 movies and recommend them to the user.



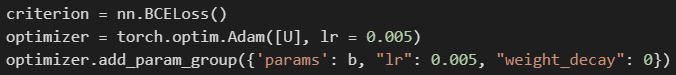
## A Classification Model Result

### Learning Phase

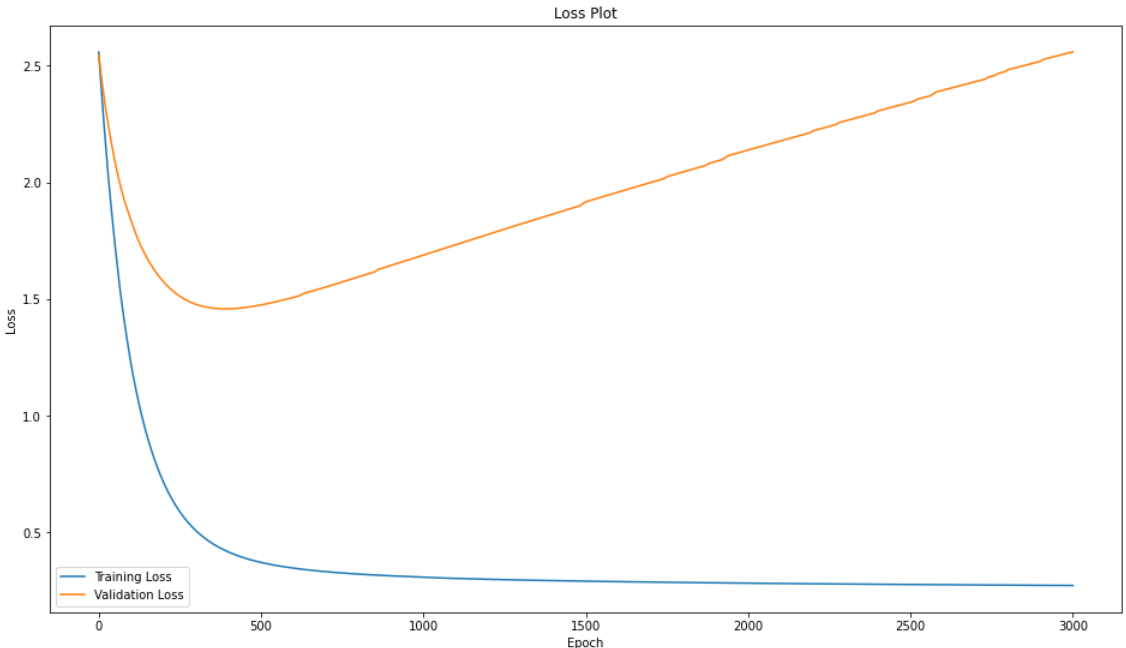
After splitting the rating dataset into two parts **ratings\_train.csv** and **ratings\_test.csv,** I use them as train dataset and validation dataset.

Using BCELoss and Adam Optimizer.

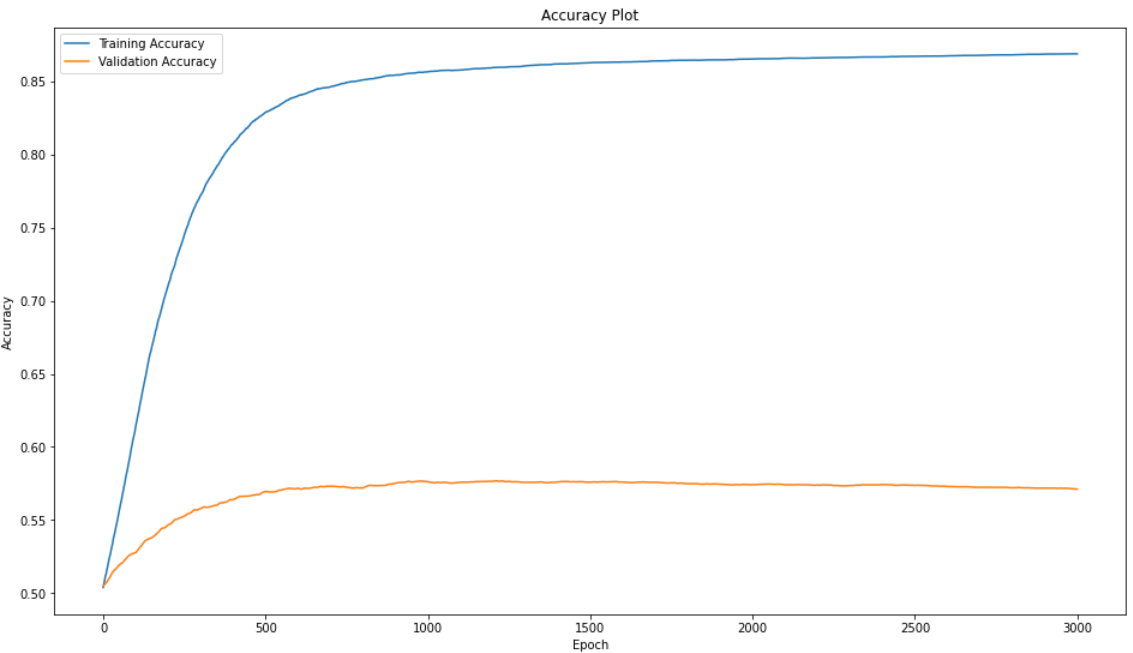
* Without Regularization



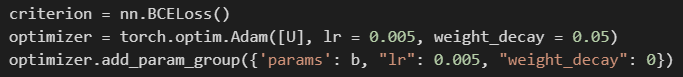
Take EPOCH of 3001, here is the loss and accuracy plot while training.



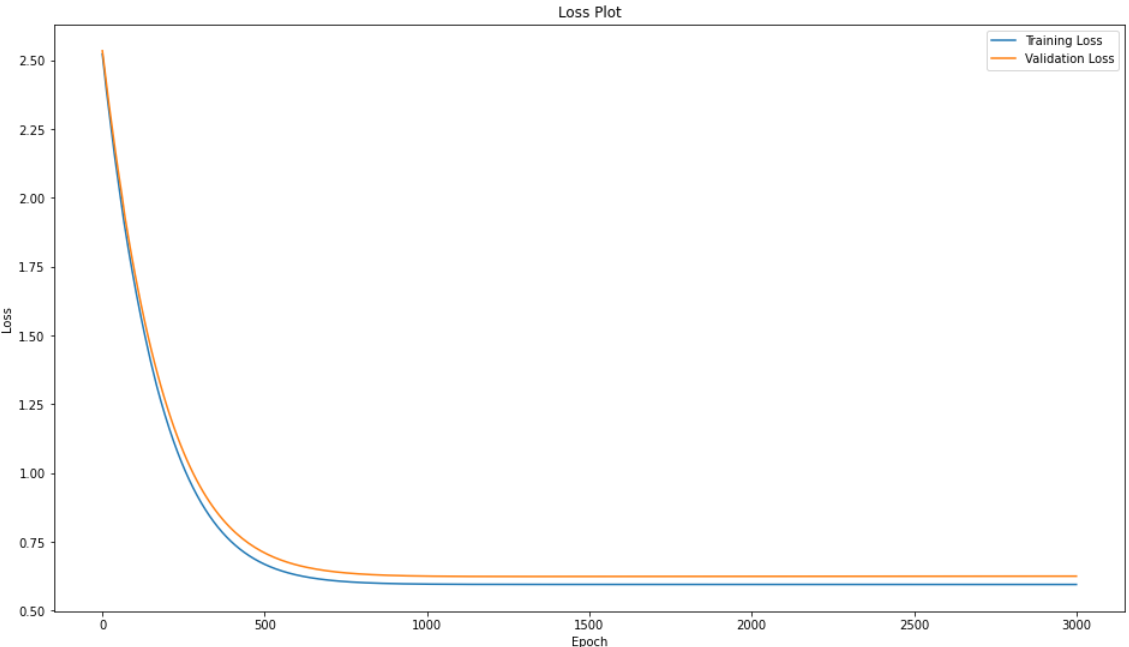
When I keep training after 500 epochs, training Loss decreases but validation loss increases rapidly. Overfitting appears here.

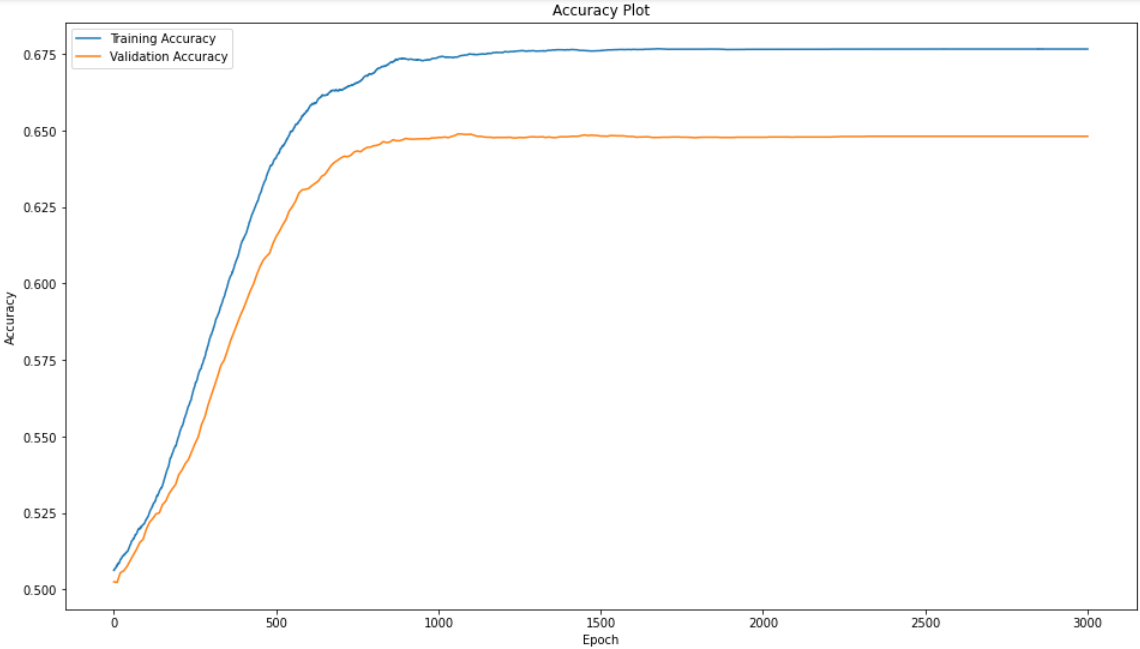


* With Regularization of lambda 0.05 on U



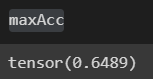
Take EPOCH of 3001, here is the loss and accuracy plot while training.



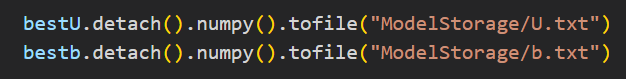


The loss plot looks better compared to the case of not Regularization. And the accuracy in the Validation Test increased by 10%.

I save the model with the highest accuracy on the Validation Set.



The parameters to save are U and b



### 

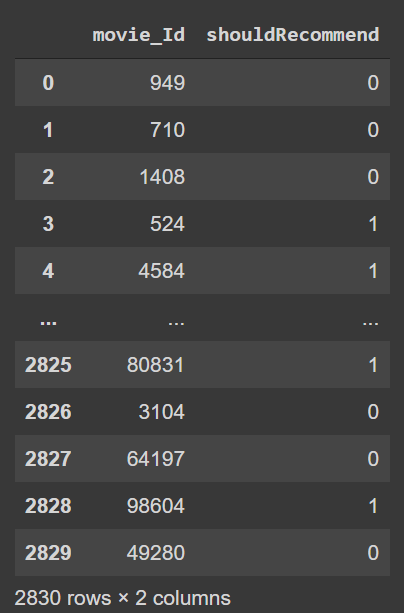
### 

### 

### Recommendation Phase

Specify **user = 10**

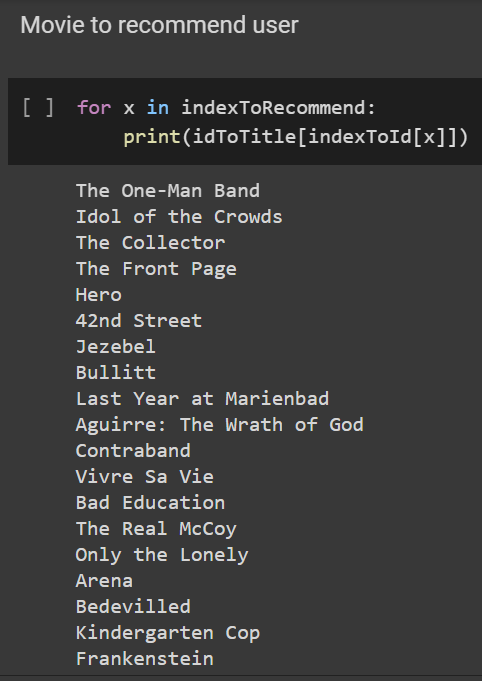
Predict the variable shouldRecommend (rating >=4.0) we have the dataframe.



After that remove from that list the movie that the user has seen.



Take the first 20 movies and recommend them to the user.

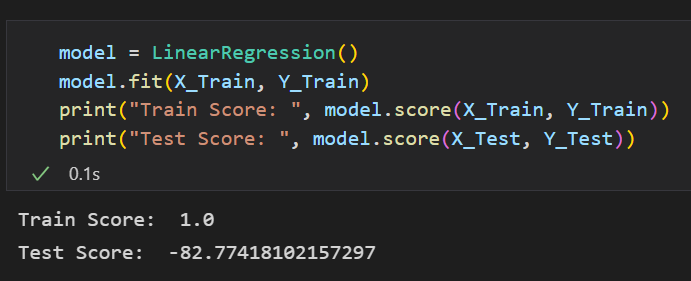


# Issues and Overcome Solution

In this part, our code is in ***Machine Learning/Old Model,*** it will not be deployed because that is only for experimenting.

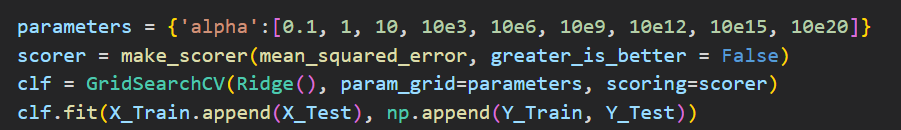
## Learning one user’s model at a time

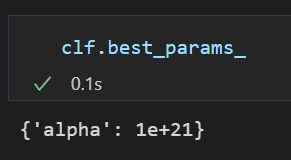
When trying to use **Linear Regression** on learning one user’s model at a time.



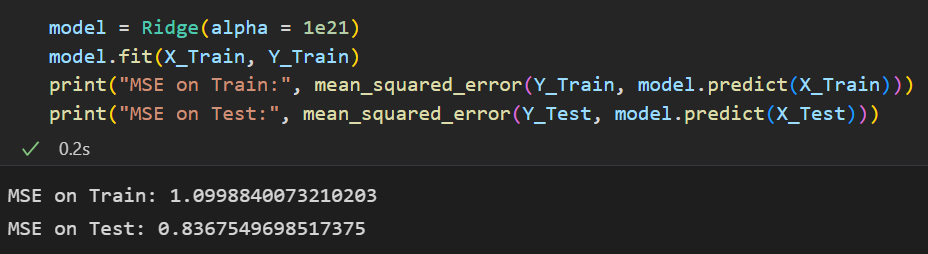
The model dramatically overfitted. Reason for that is each user only rates for a few movies but the movie profile has 38 features. We considered reducing the features but that is not a good idea.

I also tried to use Ridge Regression which uses L2 regularization term to force the weight under threshold and GridSearchCV to find lambda term of L2.

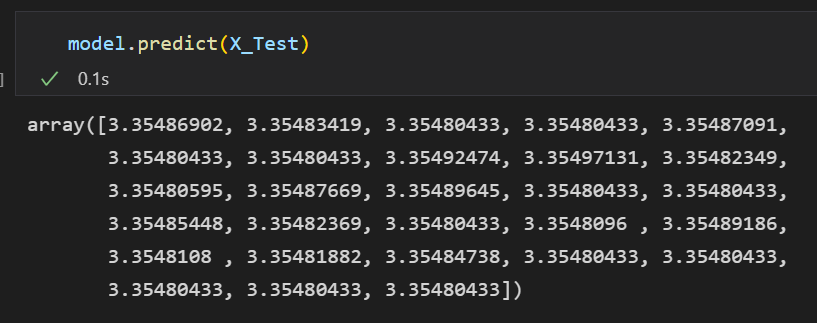




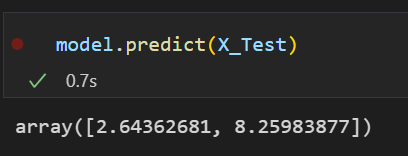
The found alpha best param is 1e21.



The result of MSE seems good but when I use that to predict the ratings. The results are somewhat constant.

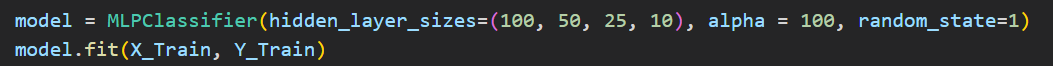


## Problem of Real Value

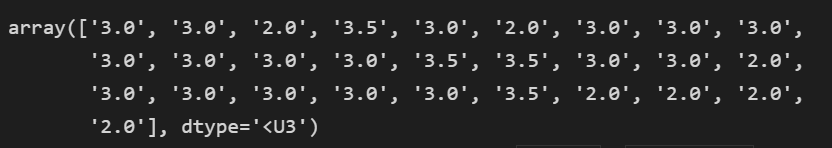


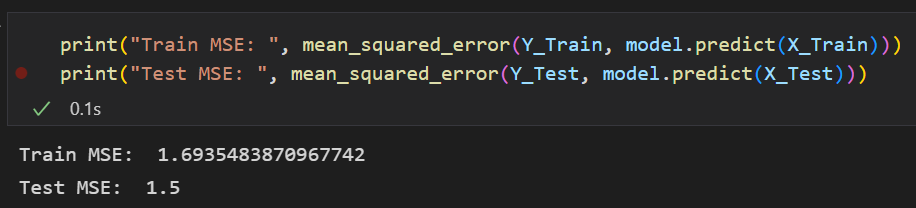
Ratings can only be between 0 to 5 but the predicted one is over **5**. Because I use real value for the output.

I try to use the MLPClassifier. MultiLayer Perceptron Classifier is a Neural Network with a softmax layer at the end when used for multiclass classification.









That is reasonable. But this model can only train one user at a time.

# Future Improvement

## Matrix Factorization

Content-Based Recommendation System has advantages and also some disadvantages.

One of them is that movie profiles are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features. Later I may use Matrix Factorization instead of Content-Based as currently working.

## Multiclass Classification Model

The output of our model is a real number so it is hard to evaluate the predicted ratings. Metrics may be ambiguous or used in the wrong case. Later I will research a classification model.

## Try Big Data

Due to memory limitation, I can use only the ratings\_small.csv dataset (not the ratings.csv one). Later I may try to use high efficient computing on working with big datasets like that.

# 

# References

The dataset that I use in this project.

* https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset

And the reference I refer to when experimenting with models.

* Recommender Systems | Content Based Recommendations — [ Andrew Ng ]
* Content-based Recommendation Systems - Machine Learning Cơ Bản