PREDICTION OF MOVIE RATINGS AMONG YOUNG ADULTS USING LINEAR REGRESSION AND DEEP LEARNING

Kate Xu
Massachusetts Institute of Technology
Cambridge, MA 02139
katexu@mit.edu

May 12, 2020

1 Introduction

Watching movies is a popular activity for people of all ages and backgrounds. After watching movies, people can rate their experiences on websites like IMDb, which collects movie information in an online database. This vast amount of movie data has led me to wonder about how I could use this data. The IMDb dataset is a popular choice for data scientists, and there exist projects that investigate the correlation between movies and their overall ratings. However, I am curious about the movie preferences of young adults like myself. My project focuses on the prediction of movie ratings among people who are aged between 18 and 30 using methods discusses in class like linear regression and deep learning in the form of neural networks. I also limit my dataset to include only movies published after the year 2000. A Jupyter notebook that includes the code for my project is also available on GitHub, and snippets of the code will be available in this project report [1].

2 Methods

2.1 IMDb dataset

I used the movies dataset and ratings dataset from the IMDb movies extensive dataset on Kaggle [2]. The movies dataset contains information on 81,273 movies with 22 features such as movie description, genre, and director. The ratings dataset contains information on these 81,273 movies with ratings based on demographics. Because these datasets contain many attributes that may not all be useful for the question of interest, data preprocessing techniques are used to select only the relevant data for this project.

2.2 Data preprocessing

I would like to consider only the columns that contain information relevant to the research question, so I removed the unnecessary columns from the datasets. Of the remaining columns, I encoded the information in a different way using one-hot encoding or standardization before we apply our models. This process of cleaning the data may be subjective because it can be difficult to define the types of relevant information for our project.

2.2.1 Movie and rating datasets

The movie dataset originally has the following columns: imdb_title_id, title, original_title, year, date_published, genre, duration, country, language, director, writer, production_company, actors, description, avg_vote, votes, budget, usa_gross_income, worlwide_gross_income, metascore, reviews_from_users, and reviews_from_critics. I want features that may correlate with the movie rating, so I keep only the following columns:

- year
- genre

- duration
- country
- language
- director
- writer
- production_company
- actors

The title and description columns were removed because there may not be common words for these features, and a model might overestimate the importance of individual words and lead to overfitting. The budget column was dropped because the amounts varied in currency. The votes, gross income, and review columns were also removed because I want information that I have before knowing whether a movie will be successful, and these removed columns would obviously correlate with higher ratings but would not be helpful for prediction.

The rating dataset originally has columns with data for the total weighted average rating, total votes received, number of votes corresponding to each rating, average rating from each age group, and number of votes from each age group. I want to predict the movie preferences of young adults, so the following columns are relevant:

- allgenders_18age_avg_vote
- allgenders_18age_votes
- males_18age_avg_vote
- males_18age_votes
- females_18age_avg_vote
- females_18age_votes

These columns refer to the average and number of ratings by all genders, males, and females between the ages of 18 and 30. Afterwards, I concatenated the two datasets for movie data and rating data into one dataset because it can be hard to work with separate datasets once I start removing some movie entries to further clean our data.

2.2.2 Remove some entries

Before I move on to encoding the values of each feature, I need to drop the movie entries that were published before the year 2000 or do not contain sufficient information. First, I am focusing on movies that were published in 2000 or later, so any movies published before then are not relevant to the research. Second, when I look at the column for number of ratings, I notice that some movies have few votes, which may have skewed the rating and not be a good representation of preferences for that movie. I handle this situation by keeping only movies that have at least 100 ratings by young adults for both males and females. I have 81273 movie entries in total, so it is possible to drop some rows without worrying about not having enough data left.

Once I remove the movie entries that lack enough ratings, I drop the following columns:

- allgenders_18age_votes
- males_18age_votes
- females_18age_votes

This is because I want to predict only the movie ratings. Moreover, I notice that there are missing values for features of some movie entries, and I will deal with them by removing those rows from the dataset. After these necessary steps to remove some of the data, I am left with 9859 movie entries.

Unfortunately, due to the limited capacity of my computer, I can use only 1000 of the 9859 available movie entries for this project. This issue is mainly due to the one-hot feature encoding that I will need to perform in the next step, and my computer does not have enough memory to safely handle more data. As a result, I will need to select a random sample of 1000 movie entries to use for the analysis.

2.2.3 Feature encoding

Of the kept columns, I use the following encoding schemes for each variable:

- year: standardize
- genre: one-hot
- duration: standardize
- country: one-hot
- language: one-hot
- director: one-hot
- writer: one-hot
- production_company: one-hot
- actors: one-hot

I start by standardizing the values in the ‘year‘ and ‘duration‘ columns. I will one-hot encode the values in the other columns. Many of these columns contain multiple values separated by commas, so I need to extract each of them when I apply one-hot encoding. I also scale the values of the output ratings from a range of 0.0 to 10.0 to a range of 0.0 to 1.0.

2.3 Machine learning models

I have a regression problem because the possible movie ratings are between 1.0 and 10.0. I will use a linear regression model and a neural network model to predict movie ratings among young adults for males, females, and both genders. I trained, tested, and evaluated my models on the data five times for each model and category. The code for the following models are available in Section 5.

2.3.1 Linear regression

Linear regression models the linear relationship between one or more features and the output. I will implement our linear regression model using Scikit-learn. I will use 70% of our data to train and 30% to test the model, and I evaluate the model using mean squared error (MSE).

2.3.2 Deep learning

Deep learning is a popular approach for machine learning and data science problems, and it includes neural networks that have multiple layers between the input and output layers. I will implement a deep neural network model using TensorFlow and keras. I will use 70% of the data to train, 15% to test the model, and 15% for a validation set. I use a learning rate of 0.001 and 10 epochs for the neural network.

3 Results

3.1 Linear regression

Table 1: Linear Regression MSE

Trial	**All genders**	**Males**	**Females**
1	0.7690085327	0.8530062387	0.770439379
2	0.9571185079	0.85150219	0.8963243093
3	0.9370875061	0.9455832923	0.7764646447
4	1.140171457	0.8895355497	0.8826786854
5	0.7436397747	0.7838265389	0.7643921582
Avg	**0.9094051557**	**0.8646907619**	**0.8180598353**

As in Table 1 with the linear regression model, I found that the mean squared error (MSE) for trials on data from people from all genders, males, and females were mostly between 0.7 and 1.2. The lowest average MSE was for the movie ratings from females, and the highest average MSE was for the movie ratings from all genders that includes the males and females. There was some variation in the MSE depending on the trial, but the errors stayed within the same range.

3.2 Deep learning

Table 2: Neural network MSE

Trial	**All genders**	**Males**	**Females**
1	1.684515366	1.613799886	1.247506284
2	1.313865411	1.870941078	1.38943791
3	1.448695457	1.591628293	1.379074999
4	1.334519084	1.848452113	1.537997001
5	1.531961884	1.231837908	1.259409356
Avg	**1.462711441**	**1.631331856**	**1.36268511**

In Table 2 with the neural network model, we see higher average and overall mean squared error for all three gendered groups. The movie ratings from females still had the lowest average MSE, while the ratings from males now has the highest average MSE. The mean squared error for all three groups fall between the range of 1.2 and 1.9, which higher than that from the linear regression model above.

4 Discussion and Conclusion

From the results from our data, we see that our linear regression model showed better performance than our neural network in terms of mean squared error. The average MSE for the linear regression varies between 0.81 and 0.91, while that for the deep learning neural network varies between 1.36 and 1.63. These findings remain consistent for the rating data for all genders, males, and females. However, we note that the deep learning model could have been improved. With additional time to work on this project, I would expand my use of deep neural networks by trying different combinations of hidden layers and nodes. This would allow me to find the optimal deep learning model by selecting the one that decreases mean squared error.

We also note that the research problem for this project is very similar to some other problems relating to recommender systems, such as proposing the next movie for a user to watch on Netflix. We could expand this project to include other approaches like content-based recommendation and collaborative filtering, including alternating least squares and stochastic gradient descent for this type of problem. Furthermore, the use of mean squared error for this project serves as a relative metric to measure the performance of linear regression and deep learning models. We could use other performance metrics like accuracy to provide another perspective on our models.

All in all, I enjoyed 18.065 and learned a lot about matrix methods, especially singular value decomposition, different types of matrices, and eigenvalues and eigenvectors. I am glad to be able to apply the concepts that I learned in this class, as well as in other classes, towards this final project.

5 Code

The following images are snippets of my code used for this project, including the linear regression and neural network models for analyzing the movie rating dataset [1].

```
import numpy as np
import pandas as pd
import random
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from tensorflow import keras
from tensorflow.keras import layers

movie_data = pd.read_csv('imdb-extensive-dataset/IMDb movies.csv')
rating_data = pd.read_csv('imdb-extensive-dataset/IMDb ratings.csv')

movie_data = movie_data[['year', 'genre', 'duration', 'country', 'language', 'director', 'writer',
                         'production_company', 'actors']]
rating_data = rating_data[['allgenders_18age_avg_vote', 'allgenders_18age_votes', 'males_18age_avg_vote',
                           'males_18age_votes', 'females_18age_avg_vote', 'females_18age_votes']]
data = pd.concat([movie_data, rating_data], axis=1, sort=False)

data = data.loc[data['year'] >= 2000]
data = data.loc[data['males_18age_votes'] >= 100.0]
data = data.loc[data['females_18age_votes'] >= 100.0]
data = data.drop(columns=['allgenders_18age_votes', 'males_18age_votes', 'females_18age_votes'])

for col in data.columns:
    data[col].replace('', np.nan, inplace=True)
    data.dropna(subset=[col], inplace=True)

row_indices = []
for index, row in data.iterrows():
    row_indices.append(index)
```

```
rand_indices = random.sample(row_indices, 1000)
data = data.loc[rand_indices]

data = data.apply(lambda x : (x - x.mean()) / x.std() if (x.name == 'year' or x.name == 'duration') else x)
data = pd.concat([data.drop('genre', axis=1), data['genre'].str.get_dummies(sep=", ")], 1)
data = pd.concat([data.drop('country', axis=1), data['country'].str.get_dummies(sep=", ")], 1)
data = pd.concat([data.drop('language', axis=1), data['language'].str.get_dummies(sep=", ")], 1)
data = pd.concat([data.drop('director', axis=1), data['director'].str.get_dummies(sep=", ")], 1)
data = pd.concat([data.drop('writer', axis=1), data['writer'].str.get_dummies(sep=", ")], 1)
data = pd.concat([data.drop('production_company', axis=1), data['production_company'].str.get_dummies(sep=", ")], 1)
data = pd.concat([data.drop('actors', axis=1), data['actors'].str.get_dummies(sep=", ")], 1)

X = data.drop(columns=['allgenders_18age_avg_vote', 'males_18age_avg_vote', 'females_18age_avg_vote'])
y_all = data[['allgenders_18age_avg_vote']]
y_male = data[['males_18age_avg_vote']]
y_female = data[['females_18age_avg_vote']]

def linear_regression(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
    model = LinearRegression()
    model.fit(X_train, y_train)
    prediction = model.predict(X_test)

    mse = mean_squared_error(prediction, y_test)
    return mse

def nn(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)

    model = keras.Sequential([layers.Dense(16, activation="relu",
                              input_shape=[len(X_train.keys())]),
                              layers.Dense(16, activation="relu"),
                              layers.Dense(1)])
```

```
    model.compile(optimizer=keras.optimizers.SGD(0.001),
                  loss=keras.losses.MeanSquaredError(),
                  metrics=["mse"])

    model.fit(X_train, y_train, epochs=10, validation_split=0.15)
    prediction = model.predict(X_test)

    mse = mean_squared_error(prediction, y_test)
    return mse

linear_regression(X, y_all)
linear_regression(X, y_male)
linear_regression(X, y_female)

nn(X, y_all)
nn(X, y_male)
nn(X, y_female)
```

6 References

[1] https://github.com/k8xu/movie-rating-prediction/blob/master/18.065_Final_Project_KX.ipynb

[2] https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset