Documentation

Reza Marzban

December 2019

Introduction

The data that I chose for my final project is IMDb movie dataset. It is available online on https://www.imdb.com/interfaces/. IMDb data is updated continuously with all recent movies information. It is consisted of 7 tables. My program, automatically download and unzip 3 tables out of those 7 tables as 4 other tables is irrelevant to our problem. As a result, the program always runs on the latest version of data. The names of three tables that we use are as follows: **title.basics**, **title.crew** and **title.ratings**. We use inner join to merge them together using **tconst** column as the key. After doing so we get around 1 million rows of raw data, and each row represent a movie. The columns that we chose to include as our features are **Year, Genres, Title-Type, runtime, directors, average-rating**. Year is a 4-digit number that represent the year that the movie hit box office. Genres includes all genres of the movie (Comedy, action, animation, etc.), each movie has at least one genre. Title-Type clarify the type of the movie (movie, short, TV-series, etc.). runtime includes the duration of a movie in minutes. Directors specify director of a movie. Average-rating is a number between 1 to 10 which is the average of all ratings for current movie.

In this project my goal is to come up with a model that can predict a movie rating by checking the rest of above features. We have created 4 models, two of which are classification models (over 10 class), and the two remaining ones are regression models. We have used Logistic Regression and Neural Network for classification and Linear regression and Neural Network for regression.

Preprocess

In my project repository there is a dataset_downloader.py that is in charge of downloading and unzipping the latest version of data. After that data is read by Preprocess class and passed through several functions to be cleaned. Preprocessing techniques and steps are described below:

- Transforming all strings to lower case.
- Transforming all numerical values to float.
- Run time is normalized and converted from minutes to hours.
- Years are normalized by the following equation to be converted between 0 and 1:

$$\frac{year - minimum\ year}{maximum\ year - minimum\ year}$$

- All of the genres are split over comma, and transformed to one hot encoder representations, so if a movie belongs to both romantic and comedy, both columns would be set to 1 and rest to 0.
- Title types are transformed to one-hot encodings.
- Directors had some challenges, as we have over 100,000 unique directors, one hot encoder representation was out of discussion. I chose to use feature hashing (a.k.a. hashing trick).

The relation of some features and our label can be seen in figures 1 – 3.

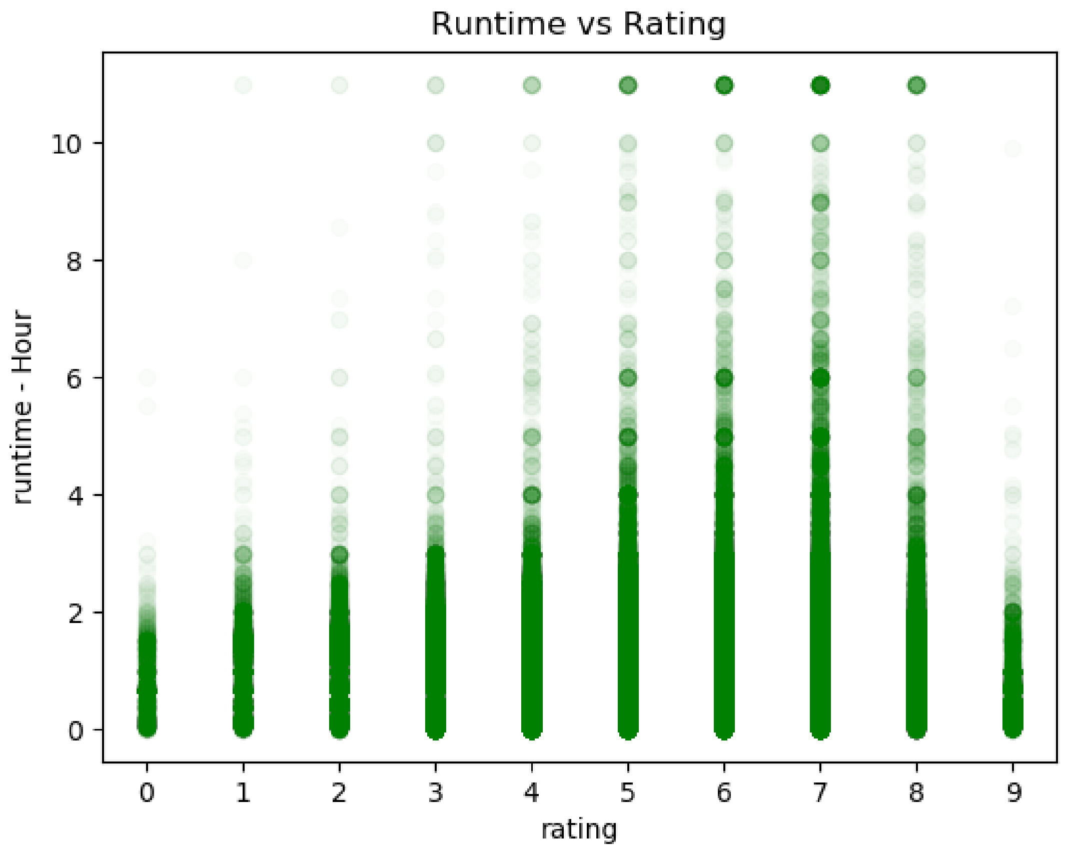

Figure 1 – runtime vs rating

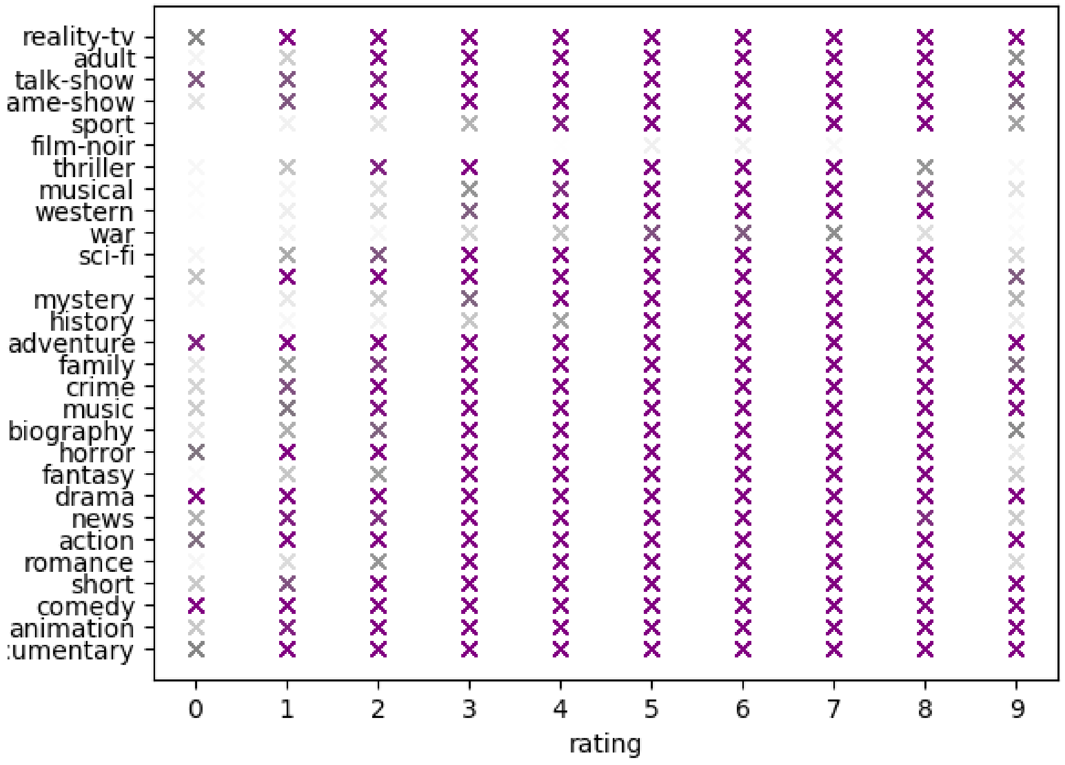

Figure 2 – Genres vs rating

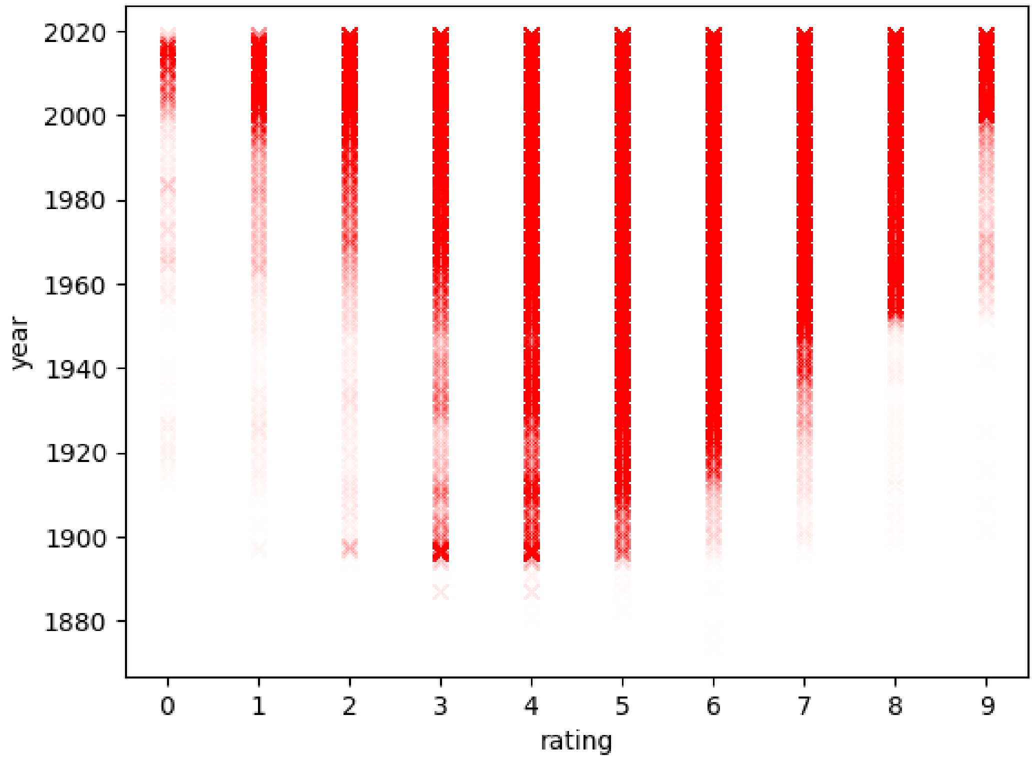

Figure 3 – Year vs rating

After cleaning data, we had 100 features (columns) for each movie. The next problem that we handled was that the data was not balanced (figure 4), and it caused our models to be biased toward specific classes. As a result, we balancer our data by randomly selecting 15,000 rows from each class, and it gave us around 150,000 randomly chosen balanced data points.

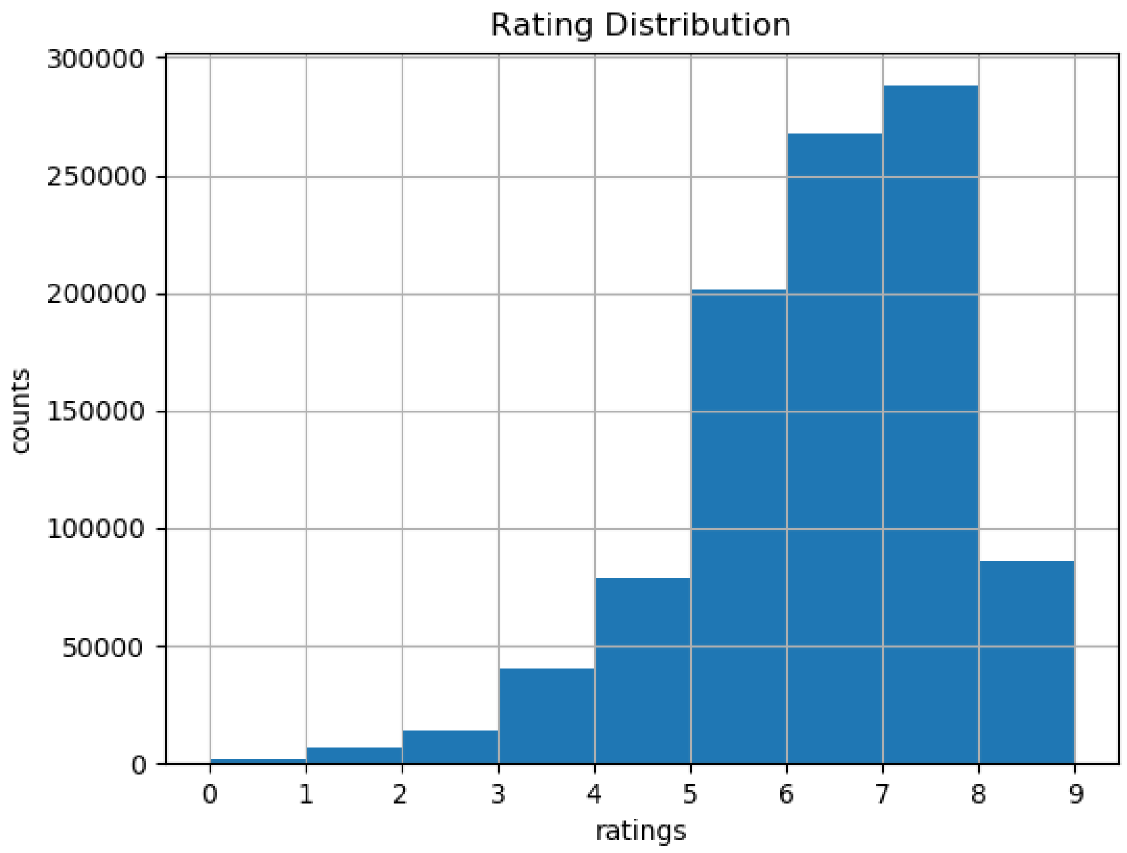

Figure 4 – Rating distribution

The final step in our preprocessing was to split data into training set and testing set. The ratio that we used is 80:20. The preprocess gave us 4 final numpy arrays that consisted training and testing features and labels.

Models

Due to the type of the problem in this project, we could use both classification and regression for our purpose, we chose two classification algorithm and two regressions to compare the results. You can check the details of each model in table 1.

	Linear Regression	**Logistic Regression**	**Neural Network for Classification**	**Neural Network for Regression**
Training Time (min)	0.0160	0.1851	2.1050	2.0840
epochs	20	20	20	20
Loss function	RMSE	RMSE	Categorical Cross Entropy	MSE
Test Loss value at last epoch	-	-	1.8701	3.4307
Performance metrics	R squared - R2	Accuracy	Accuracy	R squared - R2
Train Performance value at last epoch	25.25 %	25.52 %	40.43 %	51.92 %
Test Performance value at last epoch	24.16 %	25.41 %	30.41 %	39.34 %

Table1 – Model Comparison

For linear Regression and Logistic regression, I used sklearn to implement them, and for Neural networks I used tensorflow.

Neural Network details: Both neural Networks have same architecture with a little difference. In ANN for classification last layer has 10 nodes, but in regression it has just one node. The loss function in each are different as well (Categorical Cross Entropy, Mean Squared Error respectively). See table 2 for details.

Layer (Type)	**Output Shape**	**Number of Parameters**

Input	(None, 100)	-
Dense (Fully Connected)	(None, 1024)	103,424
Dense (Fully Connected)	(None, 512)	524,800
Dense (Fully Connected)	(None, 256)	131,328
Dense (Fully Connected)	(None, 128)	32,896
Dense (Fully Connected)	(None, 64)	8,256
Dense (Fully Connected)	(None, 32)	2,080
Output	(None, 10) or (None,1)	330 or 33
Total:		**803,114**

Table 2 – Neural Network architecture

Performance: The chance (random) accuracy is 10% as we have 10 possible classes.

- **Linear Regression**

In this model our training R squared was 25.25% and test R squared was 24.16%. In figure 5, you can see that Y-axis represent True ratings and X-axis represent predicted rating by this model.

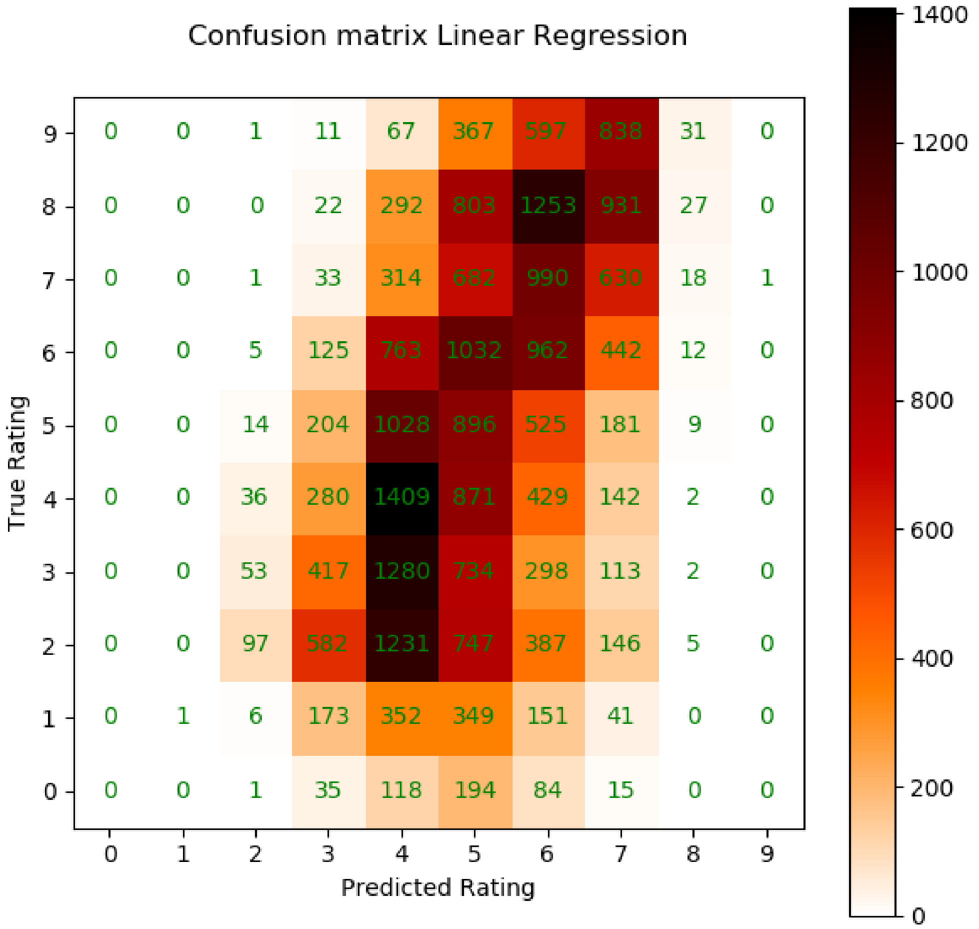

Figure 5 – Linear regression evaluation

❖ Logistic Regression

In this model our training accuracy was 25.52% and test accuracy was 25.41%. In figure 6, you can see that Y-axis represent True ratings and X-axis represent predicted rating by this model.

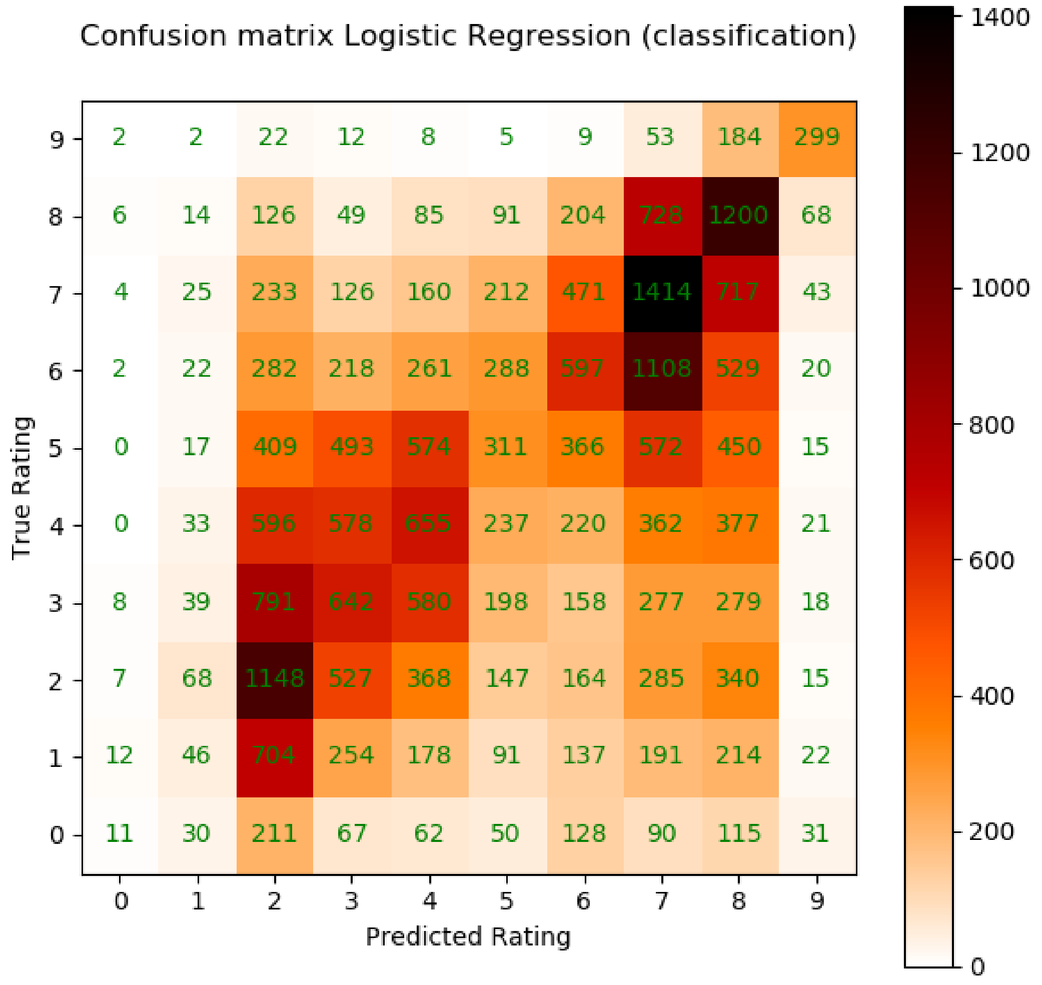

Figure 6 – Logistic regression evaluation

❖ First Artificial Neural Network for Classification

In this model our training accuracy was 40.43% and test accuracy was 30.41%. In figure 7, you can see that Y-axis represent True ratings and X-axis represent predicted rating by this model. The accuracy after each epoch is visualized in figure 8, and the loss function is represented in figure 9.

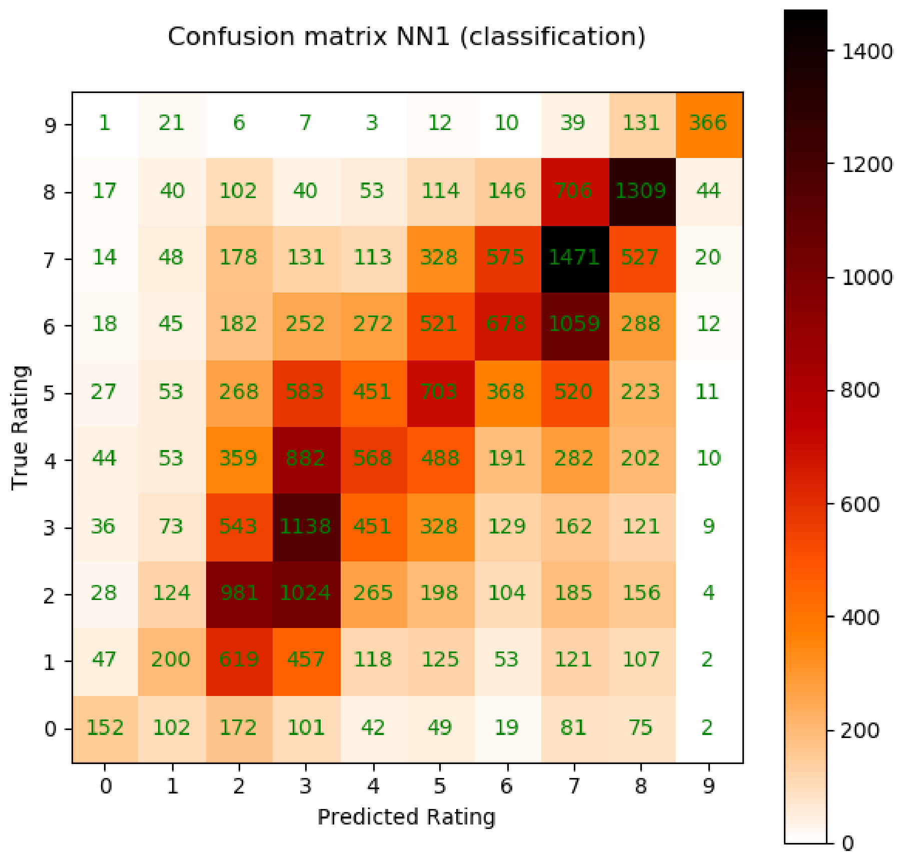

Figure 7 – NN1 Classification evaluation

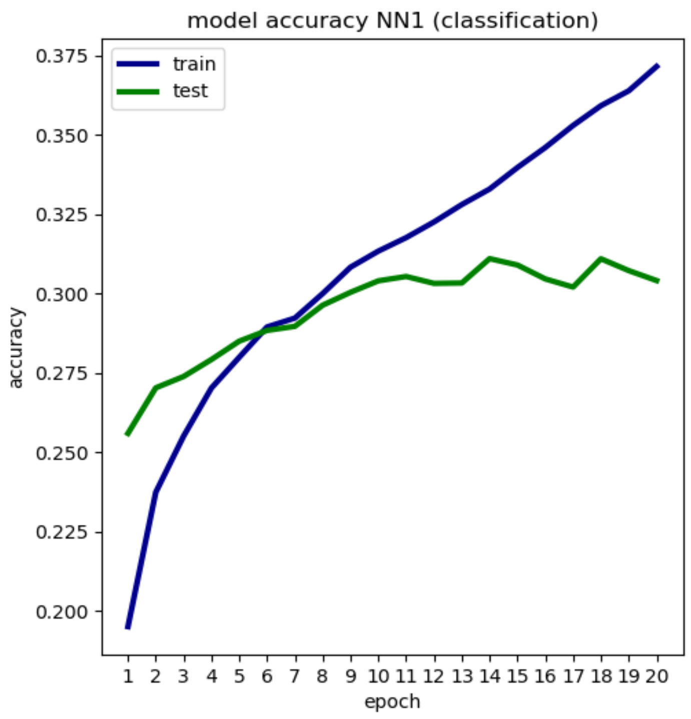

Figure 8 – NN1 accuracy plot

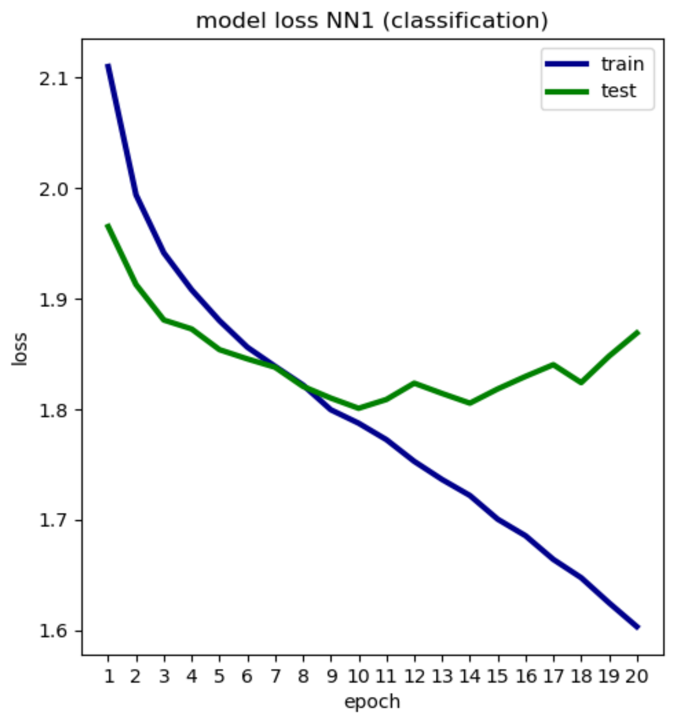

Figure 9 – NN1 accuracy plot

❖ Second Artificial Neural Network for Regression

In this model our training R squared was 51.92% and test R squared was 39.34%. In figure 10, you can see that Y-axis represent True ratings and X-axis represent predicted rating by this model. The R Squared after each epoch is visualized in figure 11, and the loss function is represented in figure 12.

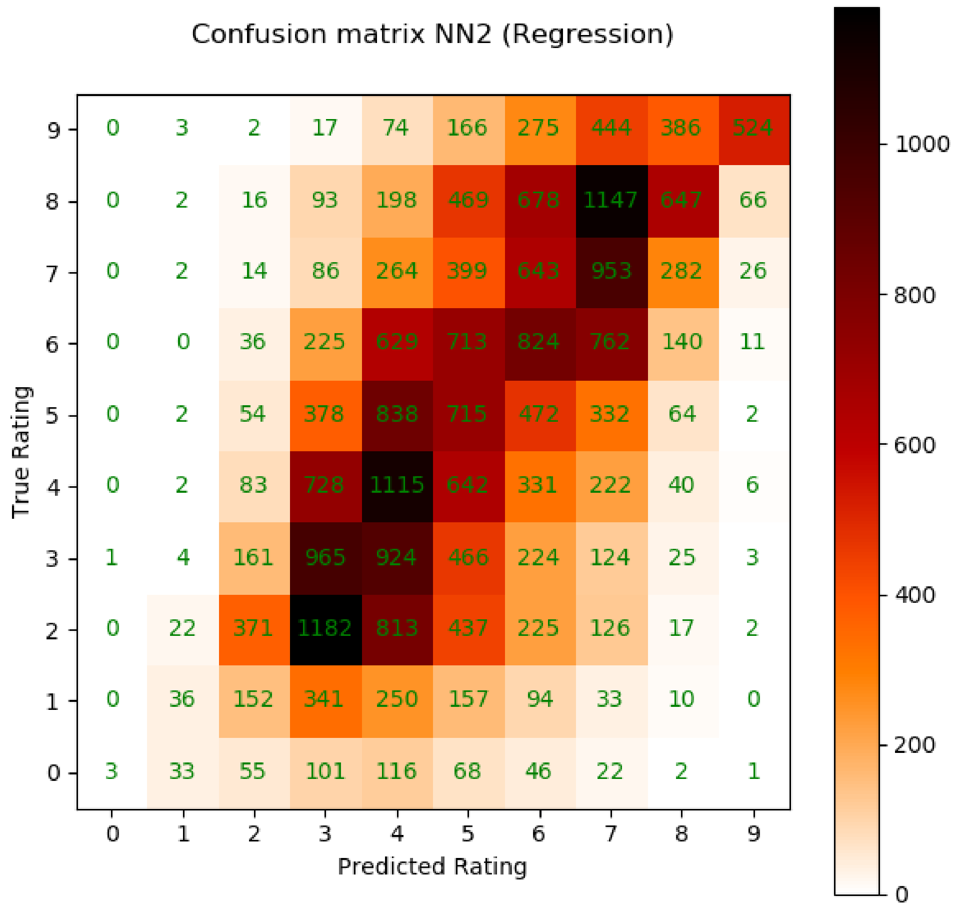

Figure 10 – NN2 Regression evaluation

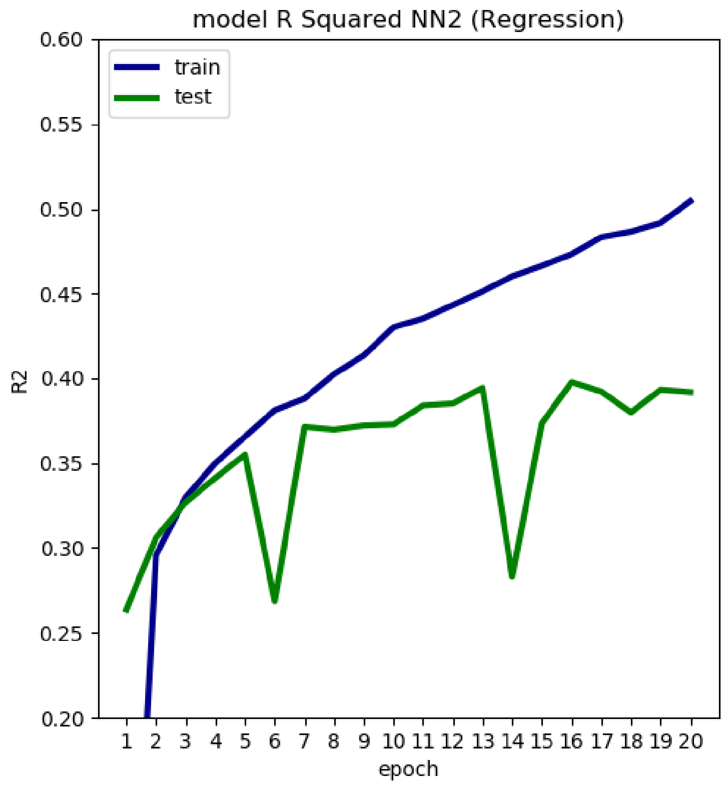

Figure 11 – NN2 R Squared plot

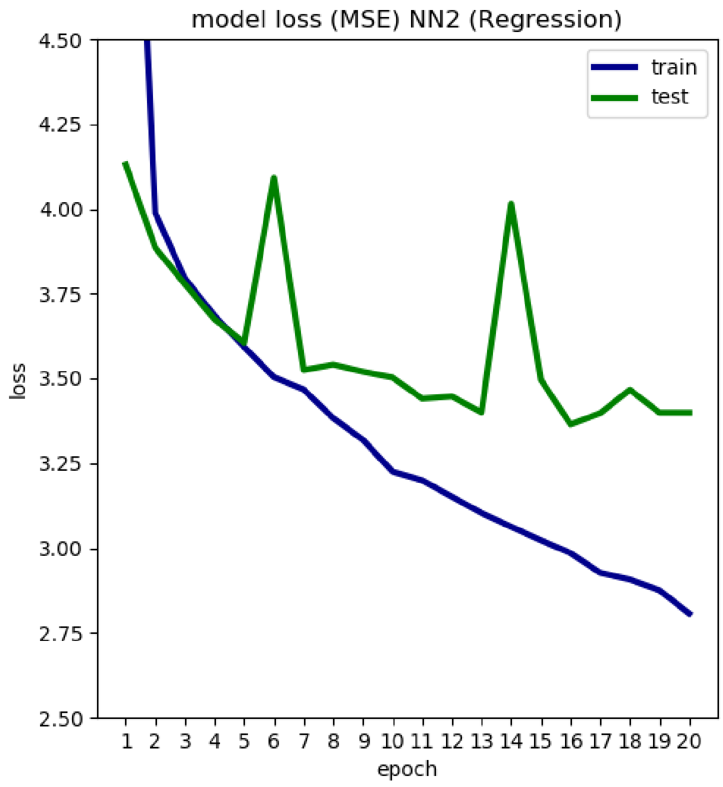

Figure 12 – NN1 accuracy plot

❖ Overall Performance

In figure 13, you can compare the performance of all 4 models. Notice that the value for classification models are accuracy and in regression models are R squared.

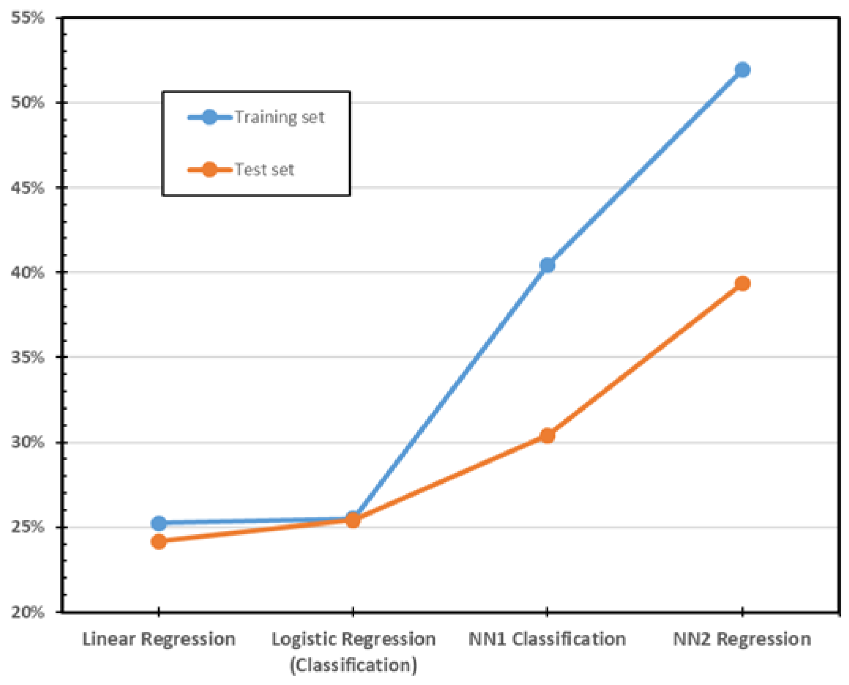

Figure 13 – Models comparison

❖ **Dropout**

In our first Neural Network for classification, we tried to increase the accuracy of test set by using dropout, in order to hyper-tune the dropout rate, we tested different values. The performance is visualized in figure 14.

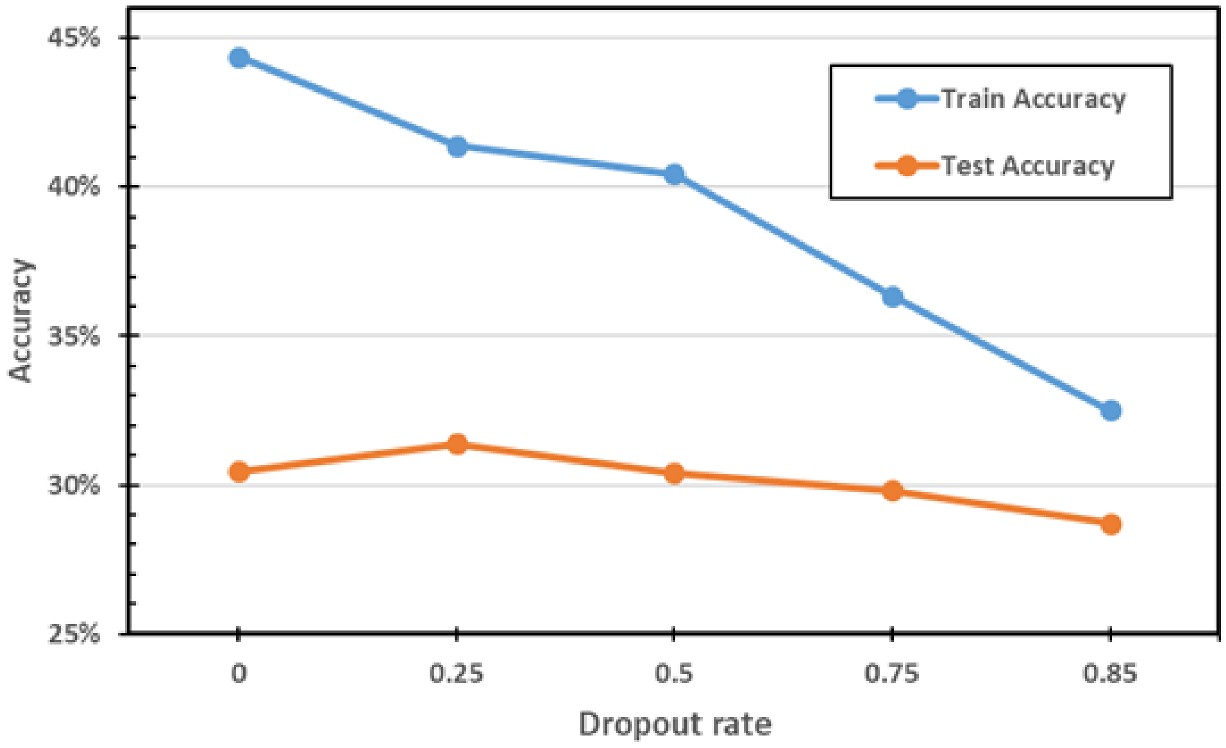

Figure 14 – Dropout rate hyper-tuning

As It can be seen, as the dropout increase, training accuracy decreases, but the testing accuracy does not have any significant change (The test set accuracy is always equal to **30% ±1%**).

Conclusion

It is believed that deep learning algorithms are better just for unstructured data (e.g. images, texts, voices, etc.), but when we have tabular structured data, machine learning algorithms is believed to be more than enough. In this project we saw that even in a nicely shaped tabular data, a deep learning algorithm can do much better that traditional ML algorithms in both Classification and Regression.

In our specific problem, the best test accuracy that we get was 30.41%, and the best test R squared that we get was 39.34%. As the chance (random) performance start from 10% due to the fact that we have 10 possible classes for each movie, it seems that our model is finding some patterns, but these patterns are not strong enough to support a stronger and more accurate model.

As it can be seen in the heat maps that we included (figures 7 and 10), most of the data are concentrated around opposite diagonal (which include all accurate predictions) which shows that, we may have low accuracy, but the predicted values are not far from true ratings.