**Introduction**

This project uses linear regression to predict the ratings for the new teleplays on the online platform PolyTube. This report will focus on the task to process the data and predict the ratings as mentioned in Task 1 and the Recommendations in the Task 2. Let us begin discussion with the analysis and the pre-processing.

**Data pre-processing/analytics of task 1:**

The data provided for this task contains a lot of uncertain entries that may have occurred due to typos in data entry or other reasons. Secondly, not all data is necessary for the predicting the ratings. So, we may need to create a modified data set to be predict the future ratings.

* As we will be running a linear regression to predict the ratings of the telefilms, we need to remove all the headers as they will generate error when taken in by the algorithm that is working on a different data type.
* There may be some outliers in the data due to errors in entry such as some of the genres have been input into the length. We will run the pre-processing first to swap them to their correct place.
* At the next step we must look for the non-numeric data types that maybe hard for the algorithm to calculate. For example, names, genre, and length will not be processed as strings, as string cannot be multiplied with an integer in minimizing the cost function. So, to avoid such problems we need to set some rules.

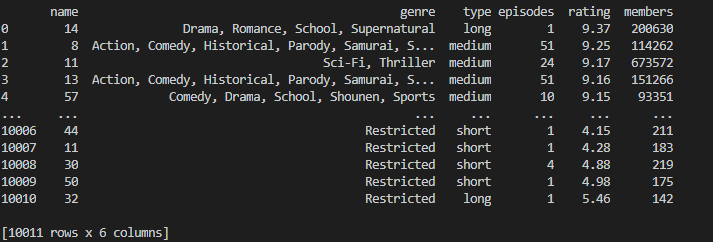
1. Name can be converted into an integer of length that will be easier to analyse. Longer names may be harder for people to remember hence less people watch and rate such shows.
2. Genre can be attributed as an array of multiple elements that can individually be identified as a separate independent variable. This adds more accuracy to the system and makes the calculations simpler. The genre can be individually extracted in form of an array and 1 or 0 can be added to each of the entry if they are part of that specific genre. The genre is as follows:

['Drama', ' Romance', ' School', ' Supernatural', 'Action', ' Comedy', ' Historical', ' Parody', ' Samurai', ' Sci-Fi', ' Shounen', 'Sci-Fi', ' Thriller', 'Comedy', ' Drama', ' Sports', ' Adventure', ' Super Power', ' Military', ' Space', ' Fantasy', ' Slice of Life', 'Adventure', ' Music', ' Mystery', ' Seinen', 'Fantasy', ' Vampire', ' Mecha', ' Shoujo', ' Horror', ' Police', ' Psychological', 'Mystery', 'Psychological', ' Demons', 'Josei', 'Military', 'Romance', ' Shounen Ai', ' Game', ' Josei', ' Magic', 'Ecchi', ' Martial Arts', 'Demons', ' Ecchi', 'Dementia', ' Harem', 'Music', 'Game', ' Cars', 'Cars', 'Mecha', 'Horror', 'School', ' Kids', 'Historical', ' Dementia', ' Shoujo Ai', 'Shoujo', 'Shounen', 'Magic', 'Harem', 'Martial Arts', 'Sports', 'Slice of Life', 'Kids', 'Parody', 'nan', 'Police', 'Seinen', 'Thriller', 'Supernatural', 'Samurai', 'Super Power', 'Vampire', 'Space', 'Restricted', ' Yaoi', ' Restricted', ' Yuri', 'Yaoi']

1. With quite a few values of episodes are marked “unknown”, they need to be replaced with 0. And rating is our target value so it will not be in our array of dependent variables. But there are a few values in our training data for rating that are blank and need to be dropped, otherwise they may interfere with our results.
2. Now we may have all the data sorted out we need to choose what is important and what is not. So, teleplay\_id can be dropped as its just an identifier and not required for our model.
3. Feature scaling needs to be used to increase the efficiency of the mathematical calculations. It will speed up the convergence of gradient decent algorithm. We can use the built-in function **.describe()** for pandas to get the max value of each item.

And this formula can be use for each of the independent variables.

The data looks like this after sorting and processing except the new columns that will be created in the future for better linear regression.



**Libraries used to data processing:**

1. Numpy
2. Pandas

**Video link**

https://youtu.be/0E5Misk985E

**Task 1**

As mentioned in the previous report. This task was supposed to be performed using a linear regression algorithm. But following the number of features and the size of the task some changes were made to the plan. The following paths were taken before finally arriving at the answer with Keras, Tensorflow.

1. Polynomial regression: As we have a lot features. Linear regression will not to justice to the data. So I chos polynomial regression.
2. Customized neural network: The File “NeuralNetworks.py” has the code for the neural network that was developed. This task took most part of the time give. I used sigmoid function after using leaky relu for having a smoother result as we only had one hidden layer.
3. Tensorflow: After the search Tensorflow was chosen as my final option for the algorithm as this was the most advanced and the most effective algorithm. I used 3 hidden layer all with relu as the activation function of choice. 2 with 64 neurons each and one output with 1 neuron. This was iterated for 1000 epochs.

The answers for the final results are saved in the file named “18086809d\_Task1.csv”

**Third party libraries:**

Numpy

Tensorflow

Keras

**Task 2**

This task required a recommender system. So as we know there are 2 types of recommender systems, we decided to go with the one that has more information. That was the Collaborative filtering. As we have a lot more data from multiple users compared to just 1 user. Therefore, collaborative filtering was the best choice.

I chose a third party module called pyspark for this task. It is a python module by apache. **This uses Hadoop on backend** as a way of managing tasks so it eased the task and fulfilled the requirements set for this project for mapreduce. The most useful thing for this recommender system is the future usefulness as it uses database framework that works similar to pandas dataframe but can provide multiple added features.

The model was trained with 25 iterations and it takes care of the regularization with the parameter set to 0.1 and all the tasks itself. The following is a snippet of the result generated.

Table

Description automatically generated

**Third party modules and libraries used:**

pyspark

**Summary and Future plan**

After preliminary analysis of data, it turned out that some of the variables like name had a really weak relation with rating. But according to my intuition it is because I am using the length of the name that maybe quite vague. So, I have decided to use map reduce to find the top repeated words in the names and use an array to see which of the names use those words and we can base the final linear model on that.

So our final data will use additional data analysis and will be using only the variable that have a strong corelation with the outcome. And the whole algorithm will be implemented using Numpy and other suitable libraries.

**Improvements**

The biggest problem with this task was the data processing so what can we do

1. We can use better models.
2. Identification of useful features
3. Using RNN as they can store information about the past neuron.
4. Different loss functions

**Reference:**

https://spark.apache.org/docs/latest/api/python/index.html