## **MOVIE RATING PREDICTION**

The document highlights the overall approach and the technical considerations made to solve the rating prediction problem. All code is encompassed in the zip

#### **What we have as Data:**

* User Ratings of movies by timestamp
* Movies and their genre
* Tags given by users to a movies and their corresponding relevance score
* Some external link ids of the movie

#### **Approach:**

The solution was attempted by creating a classification model on top features obtained from the user's movie ratings.

#### **How it was done:**

The entire exercise was divided into 3 different sections:

* **Data Sampling:** 
  + Given we had 138493 users and ~20 million ratings, building a model on the entire dataset was not the right approach and it was computationally expensive too.
  + So we tried to understand the distribution of movie ratings by users and realized that a long tail was there and hence we decided to take a stratified sampling approach
  + For stratification we bucketed all users by the total number of movies rated by them
  + Given we had such a large number of records we took 10% of the unique users from each bucket and their entire rating data to create a single batch.
  + We repeated it N (5 in our case) times to create a data set with 50% users
  + The aim was to bucket users into groups for the next stage of feature selection. This ensured that the feature selection is done only on 10% of data (each bucket) at a time rather than the entire data set at a whole, which is computationally expensive
  + A bucket contains 13849 users and ~1.7 million records
* **Feature Selection & Data Generation**
  + The approach towards feature selection was based on the concept of collaborative and content based filtering approach of any recommendation system
  + Initially we decided a blocking approach where we considered every user as a single entity which contains multiple user movie ratings sorted by timestamp of rating
  + Then for every user we created a two dataframe one with the all the records except the penultimate one and one with all the records except the final one

|  |  |  |
| --- | --- | --- |
| User Id Movie rating | Data Frame 1 | Data Frame 2 |
| First | Yes | Yes |
| Second | Yes | Yes |
| Third | Yes | Yes |
| Last-1 | Yes | No |
| Last | No | Yes |

* + The Data Frame 1 becomes our training data and Data Frame 2 becomes our testing data
  + The rationale behind this approach is:
    - Consider all users' latest movie rating as the one to be predicted depending on all the previous interactions he and his peers made on the platform. All interaction before the date of latest interaction is considered
    - Since we select Last-1 in training we remove it from the testing data frame to keep the baseline set of interactions the same for train and test without any scope of data leakage.
    - The aim is that given all interactions done till a certain time can we predict the movie rating of the upcoming movie
  + Now we build the features based on the above consideration
    - **User Genre Rating:** Average rating given by the user to movies of the same genre barring this movie, rated before this movie
    - **Peer Genre Rating:** Average rating given by users to the same genre of movies barring this movie, rated before this movie
    - **Peer movie Rating:** Average rating given by users to the same movie, before the movie was rated by the user
    - **Similar N movies Rating:** Average rating given by users to TopN (5 in our case) similar movies barring this movie. To get this we get the most relevant tags for all movies rated by various users before this movie was rated by the user. Then we take all the tags with >0.5 relevance score and match them to this movie’s tag. In case there are more than 4 (Hard Coded) tag matches we consider that movie as a similar movie. Then we get the average rating of the movies before this movie was rated
  + All the features are calculated considering only rating obtained before this movie was rated. In this way we ensure that there is no impact of future data to present rating. No time based data leakage
  + Once these considerations are made. We run the **UserMovie** class to get all the features for every User in our buckets of sample data obtained above.
  + Though we don’t use the entire rating data but every 10% user data batch is computationally cheap and also preserves the rating dependencies given the overall size of each batch

* **Model Building:**
  + After the data is created for test and training, since most of the values are continuous in nature we consider a linear classification model, LogisticRegression in this case
  + However to make it more robust we use a Bagging method to run 10 such LogisticRegression classifier and get an output based on the ensemble model
  + We ran 4 fold cross validation approach on top of the Bagging Logistic Classifier
  + Given all our features are time dependent a lot of them are **null.** We replaced all of them with the maximum rating of all the other features.
  + Once this data preparation was done we ran the model and got the output.
  + The best score obtained was accuracy of **~67%** with an F1\_score of **0.733.** We used the best score model to get our predictions on the test data
    - **pred:0 pred:1**

**true:0 13576 14927**

**true:1 7573 32384**

* + We use F1\_Score as our metric for model choice since F1\_score give importance to the model with the best balance between precision and recall
  + The Peer Movie Rating and User Genre rating has the highest impact on the model.

feature\_name feature\_importance\_coeff

0 peer\_movie\_rating 1.043513

1 peer\_genre\_rating -0.217966

2 user\_genre\_rating 0.532573

3 similar\_movie1 0.052378

4 similar\_movie2 0.248661

5 similar\_movie3 0.029987

6 similar\_movie4 -0.081546

7 similar\_movie4 -0.070867

**Conclusion:**

* The rating prediction was approached from the point of predicting the latest rating of a user.
* Given more time, we can try other approaches of data selection where instead of selecting blanket 10% we can probably select by binning users by number of movies rated
* Each word of genre can be separated and similar genres can be selected based on a number of matched words rather than exact match presently done
* Instead of a standard linear logistic classifier we select more advanced SVCs or Neural Network based models.
* An highly advanced algorithm can be a label propagation algorithm which can be used to label high or low rating in the movie
* The model suffers from low precision and hence more data augmentation can be done by balancing the majority and minority class to 50:50 instead of 60:40

**Running the Model:**

-- extract the zip file

-- create a virtual environment

-- activate the virtual env. **‘source venv/bin/activate’**

-- pip install -r requirements.txt

-- python main.py