# EGR 598: Machine Learning and Artificial Intelligence - Final Project Report

**TOPIC -** Predict the rating for a User / Item pair using Jaccard Similarity

**TEAM MEMBERS -** Abhishek Kemia, Aman Mehul Patel, Ezhilan Veluchami, Kirthik Roshan Nagaraj, Shiva Sam Kumar

## 1. Problem Statement

The aim of rating prediction is to determine how each user will rate the new books they read in the future.

This can be achieved by various Machine Learning Algorithms like Linear Regression, Logistic Regression, Bag of words and Jaccard Similarity. So, the problem that is going to be investigated in the project is as follow:

*Which machine learning approach performs better in terms of accuracy and Mean Square Error to predict the user rating based on his/her rating and reviews for the previous books?*

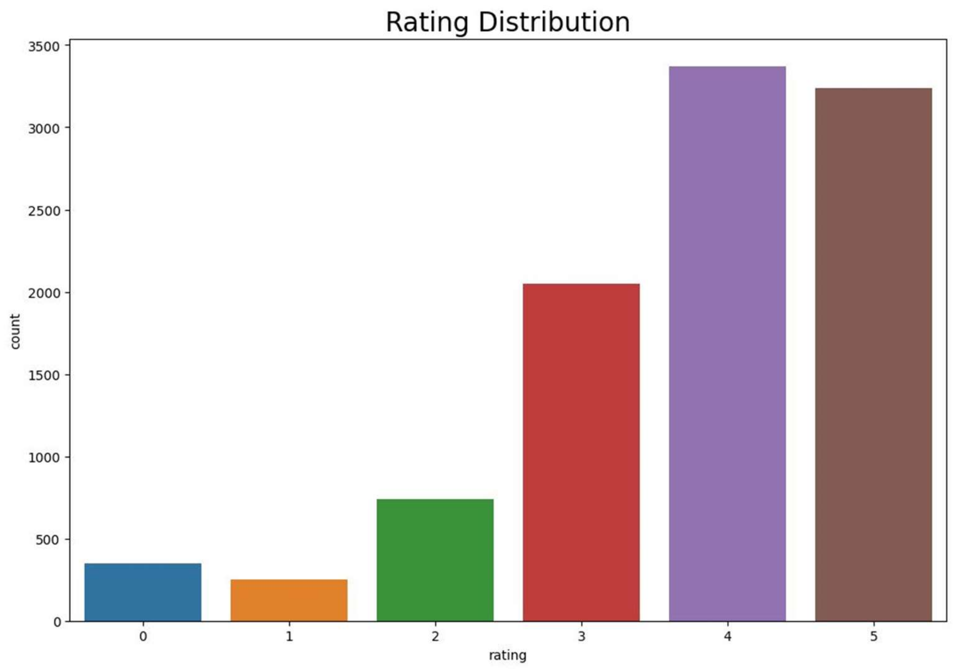
## 2. Analysis

### Data Exploration

Following steps were followed for Data preprocessing -

1. Checked data for null/empty values and removed these empty values as they are not of any use for the computation of the book rating.
2. Encoding the user_id and book_id field form hashed string to integers
3. Removing primary key " review_id " as it is a unique value for all data points and have no correlation with the rating
4. Removed the time stamps fields "date_added","date_ read_at " and started_at " as they do not add any value to the Model
5. Removed fields " n_votes " and n_comments " as they were null/empty for more than 95% of all data points

After cleaning the data, we have done the data visualization and put the rating distribution on the bar chart. On the x axis are put the ratings given by the user and on the y axis we put the count. As we can see from the bar chart, rating 4 has the highest count after that rating 5 with the lowest number of counts is for rating 1.



## 3. Algorithms and Techniques

### Jaccard Similarity

Jaccard Similarity is a common proximity measurement used to compute the similarity between two objects, such as two text documents. Jaccard similarity can be used to find the similarity between two asymmetric binary vectors or to find the similarity between two sets. In literature, Jaccard similarity, symbolized by, can also be referred to as Jaccard Index, Jaccard Coefficient, Jaccard Dissimilarity, and Jaccard Distance.

Jaccard Similarity is frequently used in data science applications. Example use cases for Jaccard Similarity:

**Text mining:** find the similarity between two text documents using the number of terms used in both documents
**E-Commerce:** from a market database of thousands of customers and millions of items, find similar customers via their purchase history
**Recommendation System:** Movie recommendation algorithms employ the Jaccard Coefficient to find similar customers if they rented or highly rated many of the same movies.

The pseudo code for predicting rating using Jaccard similarity algorithm is shown below:

```
def predictRating(user,item):
    ratings = [] # Collect ratings over which to average
    sims = [] # and similarity scores
    for d in reviewsPerUser[user]:
        j = d['product_id']
        if j == item: continue # Skip the query item
        ratings.append(d['star_rating'] - itemAverages[j])
        sims.append(Jaccard(usersPerItem[item],usersPerItem[
            j]))
    if (sum(sims) > 0):
        weightedRatings = [(x*y) for x,y in zip(ratings,sims
            )]
        return itemAverages[item] + sum(weightedRatings) /
            sum(sims)
    else:
        # User hasn't rated any similar items
        return ratingMean
```

## Linear regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.
This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a "least squares" method to discover the best-fit line for a set of paired data. You then estimate the value of X (dependent variable) from Y (independent variable).

## Logistic regression

This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas:

Logit(pi) = 1/(1+ exp(-pi))

ln(pi/(1-pi)) = Beta_0 + Beta_1*X_1 + … + B_k*K_k

In this logistic regression equation, logit(pi) is the dependent or response variable and x is the independent variable. The beta parameter, or coefficient, in this model is commonly estimated via maximum likelihood estimation (MLE). This method tests different values of beta through multiple iterations to optimize for the best fit of log odds. All of these iterations produce the log likelihood function, and logistic regression seeks to maximize this function to find the best parameter estimate. Once the optimal coefficient (or coefficients if there is more than one independent variable) is found, the conditional probabilities for each observation can be calculated, logged, and summed together to yield a predicted probability. For binary classification, a probability less than .5 will predict 0 while a probability greater than 0 will predict 1. After the model has been computed, it's best practice to evaluate how well the model predicts the dependent variable, which is called goodness of fit.

## Bag of words

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

1. A vocabulary of known words.
2. A measure of the presence of known words.

The Bag of words consists of the words present in the text input which then is added to a matrix. The words from the sentence are trimmed using 2 methods - Lemmatization and stemming. Stemming involves just trimming the last part of the word and storing the most informational part of the word (example - Jumps after stemming becomes Jump).
Lemmatization involves running the words through the vocabulary library and storing the common / informational word.
It is called a "bag" of words because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

Bag of Words limitations -

1. Vocabulary: The vocabulary needs careful design, especially to control the size, which affects how sparsely it is represented in the page.
2. Sparsity: Sparse representations are more difficult to model for computational and informational reasons.
3. Meaning: By disregarding word order, the context and subsequent meaning of the document's words are disregarded. The model could recognize the difference between the identical words differently organized.

# 4. Methodology

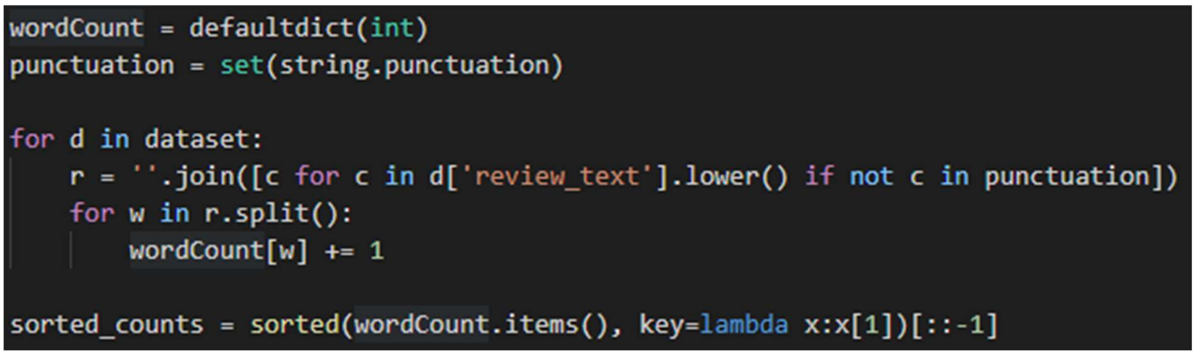
## 4.1 Implementation

### Bag of Words Approach

The BoW approach is used to predict rating based on the text present in the dataset. In the current dataset <review_text> is used to predict rating.
Code Variables for Bag of Words approach:

1. Word Count Keeps count of repeated unique words in the dataset.
2. words - Stores the most common occurring and frequently used words in the dataset.
3. function <feature> - Generates a list of the word's matrix with the usedwords per data input.

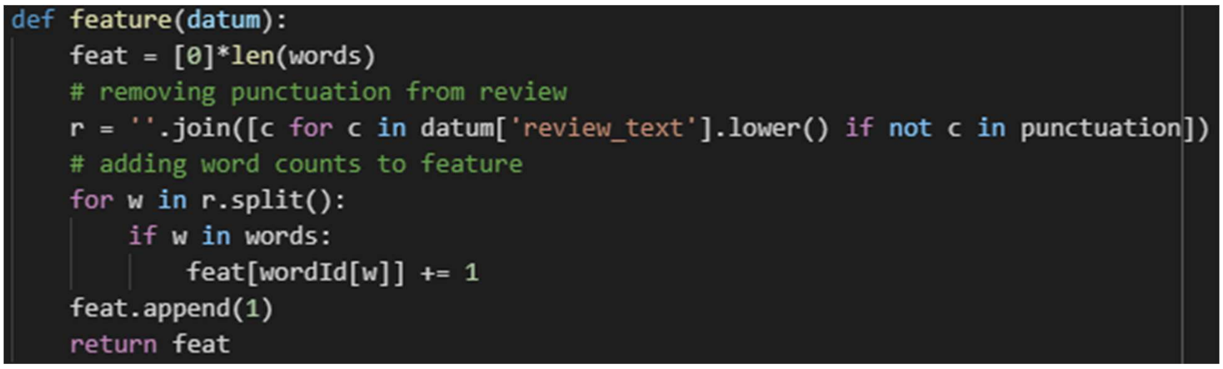
The BoW created a list of the top 1000 words which are most used in the review_text field. This list is then run across each data entry of the dataset. The output of the function <feature> will be a strength matrix with int values where if the word from the list matches the word, we are checking then it adds a count and if not it enters as 0. Here multiple use of words is just added in the count.
These generated values of the matrix are considered as X and rating as Y which is then split into 90% training and 10% testing for Linear and Logistic regression.
The following are the snapshots of the code where Bag of Words is used.



```
wordCount = defaultdict(int)
punctuation = set(string.punctuation)

for d in dataset:
    r = ''.join([c for c in d['review_text'].lower() if not c in punctuation])
    for w in r.split():
        wordCount[w] += 1

sorted_counts = sorted(wordCount.items(), key=lambda x:x[1])[::-1]
```



```
def feature(datum):
    feat = [0]*len(words)
    # removing punctuation from review
    r = ''.join([c for c in datum['review_text'].lower() if not c in punctuation])
    # adding word counts to feature
    for w in r.split():
        if w in words:
            feat[wordId[w]] += 1
    feat.append(1)
    return feat
```

## Jaccard Similarity

Even though the Bag of Approach gives a good result, it still has limitations and the predicted rating is not very accurate as the approach ignores the meaning of the sentence and just takes the consideration of the number of words and repeated words. Hence, for a more robust approach to predict the rating we have used a similarity-based measurement approach - Jaccard similarity. Jaccard Similarity is a weight-based approach which gives the user's past ratings the highest weights. This helps in finding the most relevant future ratings.

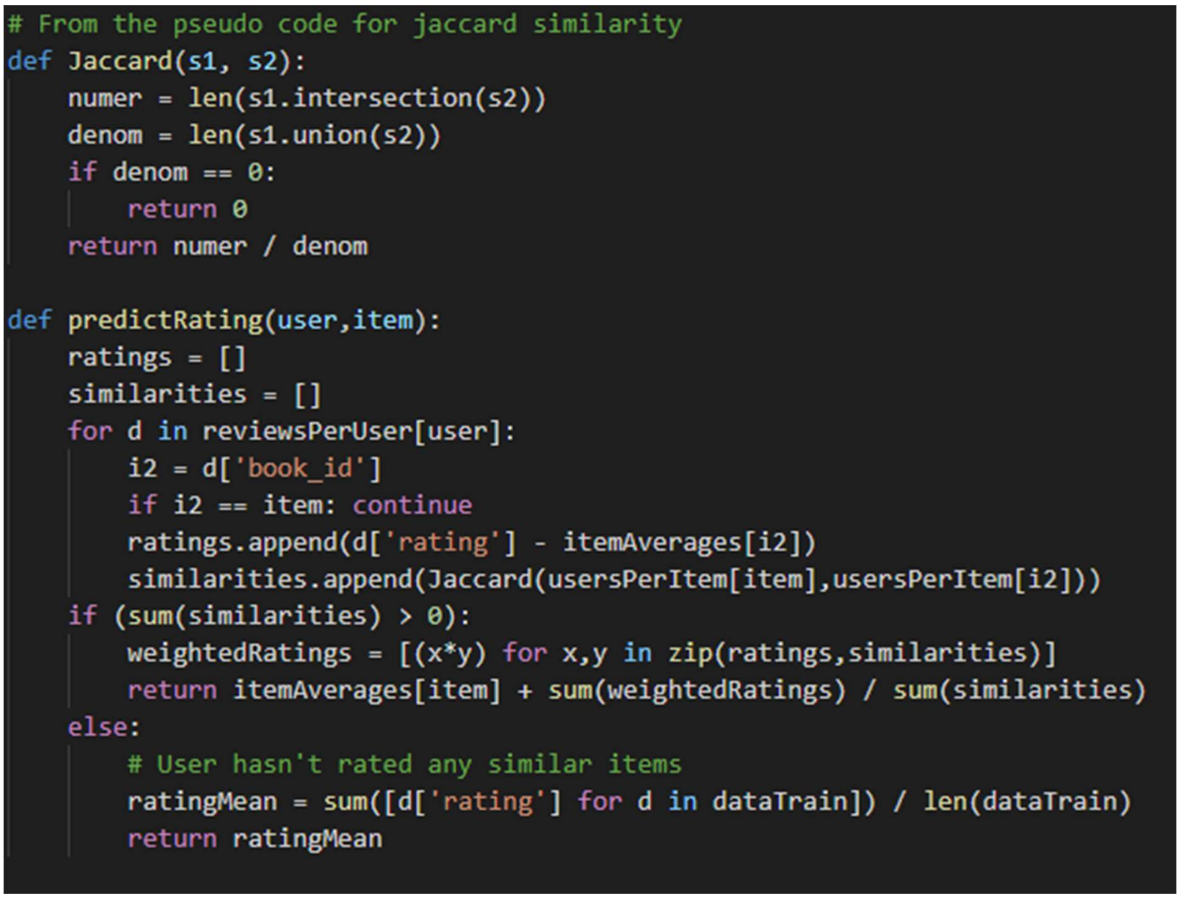
Code variables for Jaccard similarity

1. Users Per Item - Dict for unique item as values and different users as key.
2. Item Per Users Dict for unique users as values and different item as key.
3. Similarities - Jaccard similarities generated from the function <Jaccard>

In Jaccard similarity we use a User/Item pair, and the algorithm compares the User/Item pair of the item whose rating has to be predicted and the items which are rated previously. This generated a similarity score for the user and predicts the rating for a new item for the given user.

The major advantage of using similarity-based measurement methods is that it considers the previous experience and ratings of the user and using this information predicts the future rating of any new items presented to the user. This approach is more reliable compared to the Bag of Words approach and hence is widely used in rating prediction problems.

The following are the snapshots from the code with the Jaccard algorithm function.



```python
# From the pseudo code for jaccard similarity
def Jaccard(s1, s2):
    numer = len(s1.intersection(s2))
    denom = len(s1.union(s2))
    if denom == 0:
        return 0
    return numer / denom

def predictRating(user,item):
    ratings = []
    similarities = []
    for d in reviewsPerUser[user]:
        i2 = d['book_id']
        if i2 == item: continue
        ratings.append(d['rating'] - itemAverages[i2])
        similarities.append(Jaccard(usersPerItem[item],usersPerItem[i2]))
    if (sum(similarities) > 0):
        weightedRatings = [(x*y) for x,y in zip(ratings,similarities)]
        return itemAverages[item] + sum(weightedRatings) / sum(similarities)
    else:
        # User hasn't rated any similar items
        ratingMean = sum([d['rating'] for d in dataTrain]) / len(dataTrain)
        return ratingMean
```

## 5. Results

### Metrics

#### Mean Square Error

The Mean Squared Error measures how close a regression line is to a set of data points. It is a risk function corresponding to the expected value of the squared error loss.
Mean square error is calculated by taking the average, specifically the mean, of errors squared from data as it relates to a function.
A larger MSE indicates that the data points are dispersed widely around its central moment (mean), whereas a smaller MSE suggests the opposite. A smaller MSE is preferred because it indicates that your data points are dispersed closely around its central moment (mean). It reflects the centralized distribution of your data values, the fact that it is not skewed, and, most importantly, it has fewer errors (errors measured by the dispersion of the data points from its mean).

### Accuracy

Accuracy classification score.

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

## Model Evaluation and Validation

### Bag of Words Approach

#### Logistic Regression

• Accuracy - 45.5%

#### Linear Regression

• Mean squared error - 1.575
• Mean absolute percentage error - 36.61%

### Jaccard Similarity

Mean squared error - 1.837
Mean absolute error - 31.01%

# 6. Conclusion

After implementing different Machine learning algorithms, we found out that Jaccard Similarity is better than the Bag of Words approach. This same kind of approach can be used for different other machine learning tasks like -

This machine learning model can be used for predicting the ratings of reviews of hotels.

The model can also be used to give the product recommendation based on the rating of the user on previous purchases and the rating of the user with similar purchase pattern