# MID-TERM REPORT

**NAME**: ROHAN SAWANT

**ID**: U47249888

**KAGGLE USERNAME**: ROHAN SAWANT

## ABSTRACT:

In this report, I will outline my workflow for CS506's midterm competition, which involved analyzing Amazon Movie Reviews data. The large dataset primarily consisted of textual values, necessitating effective feature extraction to accurately characterize each review summary and text. This process ultimately determined the ratings given by users. The report is organized as follows:

1. Data Overview
2. Data Preprocessing
3. Modeling and Validation
4. Challenges

A comparative report on the performance of various machine learning models, employed after thorough feature extraction, engineering, dimensionality reduction, and selection, is also included in this report.

## 1.Data Overview

The original dataset consisted of two column-separated files: "train.csv" and "test.csv". The "test.csv" file only contained the Id and Score columns, with null values for scores. In order to make predictions using a machine learning model, it is crucial for the test data to have the same number of features as the training data. Therefore, the "test.csv" file was first matched with the "train.csv" file based on the Id column, and all feature columns except for "Score" were added to it.

Subsequently, all rows with null values for "Score" in the "train.csv" file were removed since they were already included in the "test.csv" file. Additionally, a new column called Helpfulness was created by dividing HelpfulnessNumerator by HelpfulnessDenominator. It is worth noting that this process was part of the template submission.

It was observed that both the Summary and Text columns had missing values in the training and testing sets. Considering the size of the entire dataset, removing these data points would be a more straightforward approach. However, eliminating data points from the test set due to null values could lead to issues during submission evaluation. To address this, the data type of Summary and Text columns was changed to "string", which converted all NaN values to "NaN" as a string. As a result, there were no actual null values, and these entries would appear the same as if a user had simply commented "NaN".

## 2. Data Preprocessing

In this code, I created a function called `process` that preprocesses the training and submission datasets for a machine learning task. I passed the `trainingSet`, `submissionSet`, and an optional `col` parameter to the function. The function first calculates the 'Helpfulness' ratio by dividing the 'HelpfulnessNumerator' by the 'HelpfulnessDenominator' and fills any missing values with 0.

Next, I filtered the training set to only include rows where 'HelpfulnessNumerator' is less than or equal to 'HelpfulnessDenominator' and removed any rows with missing values. Then, I split the cleaned training set into training and testing subsets using a 75/25 split.

Depending on the value of the `col` parameter, I removed unnecessary columns from the training and testing sets, as well as from the submission set. Finally, the function returns the processed training and testing sets, their corresponding target variables, and the processed submission set.

I first defined a function called `remove_char` that takes a string input and removes any non-alphabetic characters, trims any leading or trailing spaces, and converts the string to lowercase.

Next, I created another function called `clean_word` that processes a given dataset column by removing special characters and stopwords (common words that do not carry much meaning). I used the `remove_char` function in combination with the Natural Language Toolkit's (nltk) list of English stopwords to achieve this.

Then, I applied the `clean_word` function to the 'Text' column of the training, testing, and submission datasets. This resulted in cleaned text data, which I stored in new variables (X_train_text, X_test_text, and submission_text).

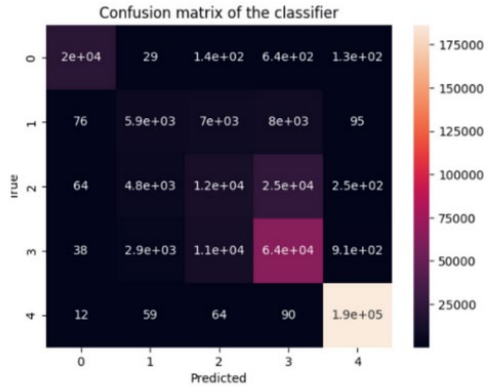
Following this, I used the `CountVectorizer` function from scikit-learn to create a count vector representation of the cleaned text data. The transformed data was stored in the variables X_training_vector_text, X_test_vectorr_text, and submission_vectorr_text.

Lastly, I used the `TfidfVectorizer` function from scikit-learn to create a Term Frequency-Inverse Document Frequency (TF-IDF) representation of the cleaned text data. The transformed data was stored in the variables X_training_tfidf_text, X_test_tfidf_text, and submission_tfidf_text.

## 3. Modeling and Validation

I evaluated three primary models: K-Nearest Neighbors (KNN), Logistic Regression (LR), and Linear Regression (LinR). The outcomes are presented below in the table. The confusion matrices for the Linear Regression can be seen in the figure below. Overall, LinR using Ridge Regression and alpha of 3.5 demonstrates the best performance.

| Model | RMSE |
|---|---|
| K-Nearest Neighbors | 1.42 |
| Logistic Regression | 0.97 |
| Linear Regression | 0.69 |




Confusion Matrix for LinR

## 4.Challenges and Future Work

Throughout the process of predicting and modeling Amazon movie reviews, I faced several challenges. Dealing with a large dataset that consisted mainly of textual data required meticulous feature extraction and engineering to accurately characterize each review summary and text. This allowed me to determine the rating a user might have given. Additionally, handling missing values and preprocessing the data to ensure consistency between the training and testing sets was a demanding task. Furthermore, selecting the most appropriate machine learning models and optimizing their parameters to achieve the best possible performance proved to be a complex undertaking. Overall, overcoming these challenges required a deep understanding of the data and various machine learning techniques, as well as extensive experimentation and fine-tuning.

For future work on this project, several avenues could be explored to improve the prediction and modeling of Amazon movie reviews:

1. **Deep learning models**: Implementing more advanced deep learning techniques, such as convolutional neural networks (CNN) or recurrent neural networks (RNN), specifically long short-term memory (LSTM) or Gated Recurrent Units (GRU), could potentially lead to better results by capturing complex patterns and dependencies in the textual data.
2. **Sentiment analysis**: Incorporating sentiment analysis as an additional feature can provide valuable insights into the underlying emotions expressed in the reviews. This could further improve the performance of the predictive models.
3. **Feature engineering**: Continued exploration of additional features, such as the time of the review or the frequency of certain keywords, could help enhance the predictive capabilities of the models.