Identifier: cs484
Rank & accuracy score: At the time of writing this report, the rank on the leaderboard is the first one with an accuracy of 0.84
Description:

Data Preprocessing

Data preprocessing is an essential step for efficient use of the data document. Procedures for removing unnecessary tags and words help to speed up and improve the good quality of training a model, thus the following detailed procedures are included and are used in the data_cleaning method:
Tokenization: This step is used three times in the data preprocessing stage: First, it divides review documents by splitting new line characters. Second, it separates labels and reviews. Third, reviews are separated into tokens while performing stemming, removing stop words, etc.
Remove HTML tags: This step is performed by substituting HTML tags with empty space characters.
Stop Words: Stops words, such as "the" and "an", are removed. Default TfidfVectorizer tokenizer has a default way to remove stop words that are used in this project.
Stemming: This step helps to derive the original form of words and transform them to the stemmed versions.
Remove Punctuations and numbers: After manually analyzing the review documents, I noticed that only alphabetic words are necessary for this program, thus numbers and other non-alphabetic characters are removed, such as "@" and "+".
Feature Selection: I choose to use the Tf-IDF rather than the bags of words technique in the project. It has advantages; it does similar tasks to bags of words, in addition, it calculates and qualifies words in a set of review documents. TfidfVectorizer is utilized in this stage and has several advantages: First, it provides fast term-frequency calculations and assigns weights to different words feature. Second, it converts the raw review documents to a matrix of TF-IDF features, which also is used to calculate cosine similarities in KNN.

Classification

This phase uses a machine learning-based technique- K nearest neighbors. It learns labeled data and produces an appropriate output by calculating similarities/distances between learned data and unseen data. It works great in this movie's review classification problem. Following procedures are performed:
Load data(fit): Instead of having and fit() method, I initialize and load all training data and k parameter value in the class of the KNN initial method. They perform the same functionalities as the fit() method but just have different method names. Note: all of the data has been preprocessed beforehand.
getNeighbors: Calculate cosine similarity distance between the training data and the unseen new data, then add the corresponding label and the distance to a collection. I select the highest score of neighbors and get their labels based on the k parameter value. In the end, a majority vote determines the corresponding label for the unseen new data, and the method is called getLabel in this case.
K parameter: Usually, a small value of k affects the KNN model decision easily, and a large value of k makes high computation cost. Initially, the k value is set to the square of the training document size, which is 117 in this case. I implement a loop for trying out a range of k values(k-100,k+200), and a graph presents at the end of this document, showing different k values affect model accuracy and F-score values. Finally, I notice that when k is around 150 for the given dataset size, the model has the highest accuracy and F-score (Approximately 86%)

Cosine similarity: To identify closet neighbors, we need to define a distance between two data points, but since this work is related to texting, which contains different word features, so general distance calculation,e.g Euclidean distance, does not work properly in this case and cosine similarity calculates distance for high dimensions data.
Predict: To make a prediction to a new unseen data, predict() method uses getNeighbors() and getLabel(). Firstly, it gets a group of neighbors based on k-value by using cosine similarity to calculate distances to training data, then ranks those distances to get the closest data points. Each of those data has pre-classified labels, which help to determine the new unseen data point label.

Cross-Validation

In my work, I choose to use **K-Fold cross-validation**. It provides advantages by splitting the k equal size datasets, and each new iteration produces a brand new dataset, which can fully utilize the data resources. Steps of the validation are performed as follows: after splitting the data into a training dataset and a test dataset, each iteration creates a new KNN model, learns a new training dataset, and tests its accuracy on the test dataset. Finally, I take a mean of performance on those KNN models. The performance calculation is used based on the performance metrics slides.
Evaluating performance: In this step, I use sklearn. metric built-in functions to evaluate an average accuracy in k-fold and accuracies for different k-value for KNN models; And an average F-score in k-fold and a list of F-score for different k values for KNN models. The built-in functions use mathematic formulas as listed below. Generally, either one of the calculations can help to determine the model performance. The reasons for using both in my project is that accuracy helps to focus on True Positive and True negative cases, which is helpful in balanced classes, and F-score focuses on False Negative and False Positive, which is helpful in imbalanced classes since it is difficult to tell the distributions of the data manually, so I come across to have both calculations presented.

$$\text{Recall}_{test}(h) = \frac{f_{++}}{\boxed{f_{++} + f_{-+}}} \qquad \text{Prec}_{test}(h) = \frac{f_{++}}{\boxed{f_{++} + f_{+-}}}$$

$$F_\beta(h) = \frac{(1+\beta^2)\text{Prec}(h)\text{Recall}(h)}{\text{Prec}(h) + \text{Recall}(h)}$$

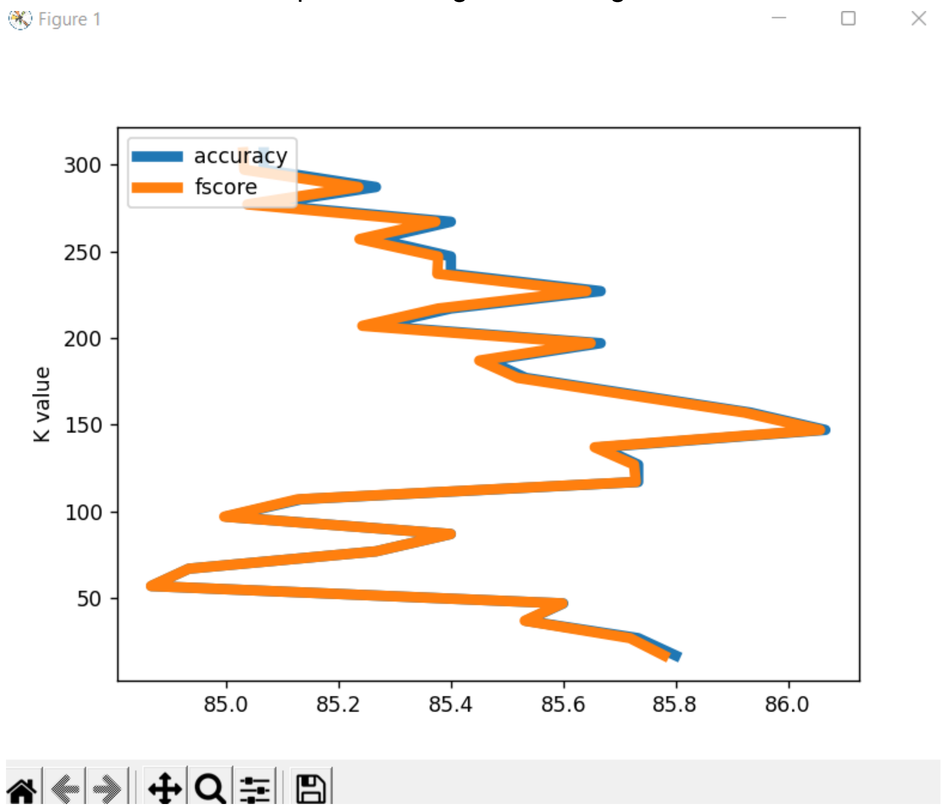
usually $\beta = 1$

Conclusion:

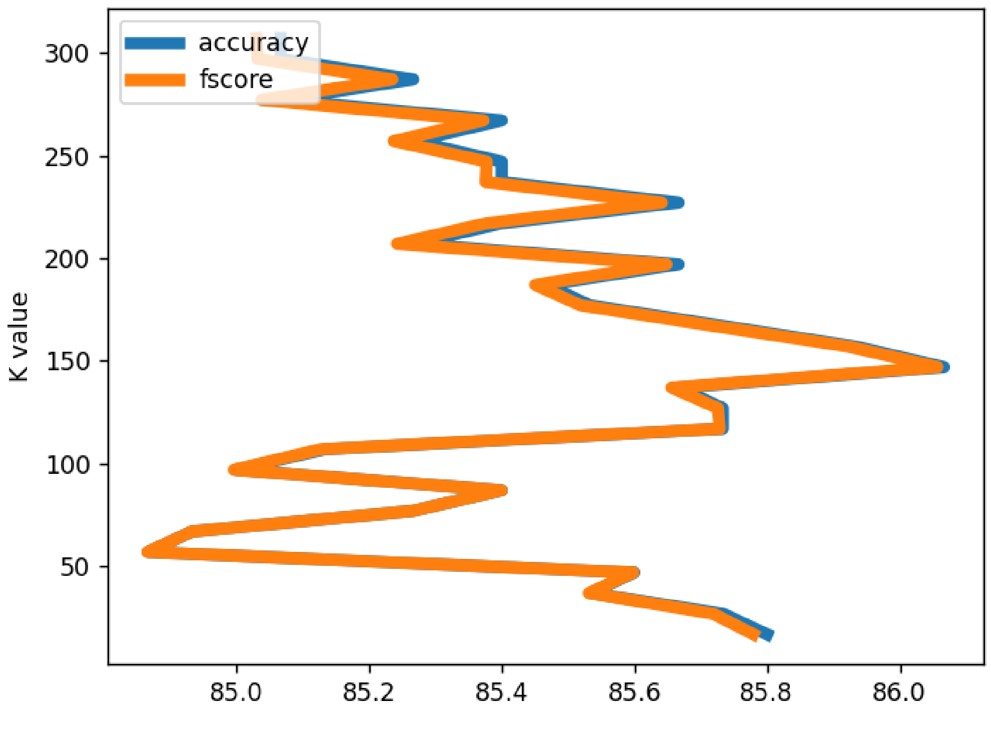
In summary, by implementing the procedures stated above, my KNN is able to make predictions with new unseen movie reviews with about 86% accuracy and F-score when having a k value for KNN for 150.

Graphs:

When K is set to the square of the given training data size.



When k is set to 150


accuracy
fscore
K value