# logo_bw.eps *Worcester Polytechnic Institute*

# *Data Science Program*

# Case Study 3

**Review Sentiment Analysis**

# Submitted By

# Sirshendu Ganguly Enbo Tian Dang Tran

## **Date Submitted :** 4/06/2022

## **Date Completed :** 4/06/2022

## **Course Instructor : Prof.** Ngan

# Motivation and Background

Before this case study, we have already talked about the scores of movies from 1995 to 2000. However, we can not decide if the score is effective or not. People may give a careless review to a movie when they don’t pay attention, and whatever score they give to a movie, they don’t have the duty on the score. Thus, if we want to drop the illusory score to movies from people, it is important to consider the written reviews of movies from them.

Furthermore , when we consider the reviews of movies, we noticed that people may have a lack of consideration for the real quality of a movie, but go with the tide. It is also important to drop these kind of written reviews to make the review of a movie more objectively.

# Data Sources

# In this case study, we used the v2.0 polarity movie\_reviews dataset from <http://www.cs.cornell.edu/people/pabo/movie-review-data>. This dataset contains 2000 movie reviews in a “txt\_sentoken” folder with them being separated by sentiment. The reviews with a positive sentiment are placed in the “pos” subdirectory and the ones with negative sentiment were placed in the “neg” subdirectory.

# Various methods were used to determine if a review was positive or negative. With a 5-star system, 3.5 stars and above are considered positive while 2 stars and below are considered negative. With a 4-star system, 3 stars and up are considered positive while 1.5 stars and below are considered negative. With a letter grade system, B or above is considered negative and C- or below is considered negative.

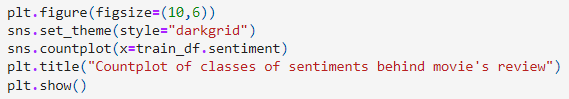
In the dataframe of the dataset, we have review and sentiment. “review” is the raw html text of a review and sentiment can be either 1 or 0 which corresponds to a positive or negative sentiment accordingly.

# Methodology

# Problem 1:

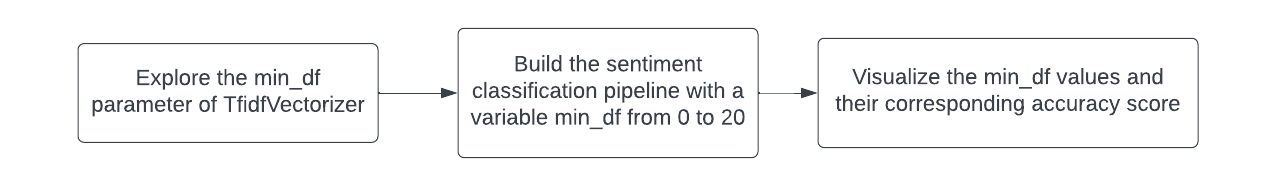
# 

* To begin, we needed to import some necessary libraries. These include nltk, sys, re, numpy, pandas, matplotlib.pyplot, seaborn, and sklearn.
* Downloaded the movie review into a “txt\_token” folder using the provided python script.
  + Loaded the dataset using sklearn load\_files then put it into a panda DataFrame.
  + Set the columns to be “Review” and “Sentiment”.
* Looking at a preview of the dataset, it’s clear pre-processing is necessary.
  + Created a function to clean to clean the text
    - Removed backslash-apostrophe.
    - Removed everything except the letters in the alphabets.
    - Removed unnecessary whitespaces.
    - Converted the text to lowercase.
  + Created a function to remove the stopwords
    - Imported a list of english stopwords from nltk.corpus.
  + Created a function to lemmatize the dataset
    - Used the lemmatizer from nltk.stem.
  + Created a function to stem the dataset
    - Used the stemmer from nltk.stem.
* Visualized the word frequency of the dataset with a horizontal bar graph.
* Develop sentiment classification pipeline.
  + Split the dataset into training and testing data sets with sklearn train\_test\_split.
  + Create a panda dataframe for the training set for data exploration.
    - Made a countplot of the sentiments behind movies’ reviews using seaborn and matplotlib.

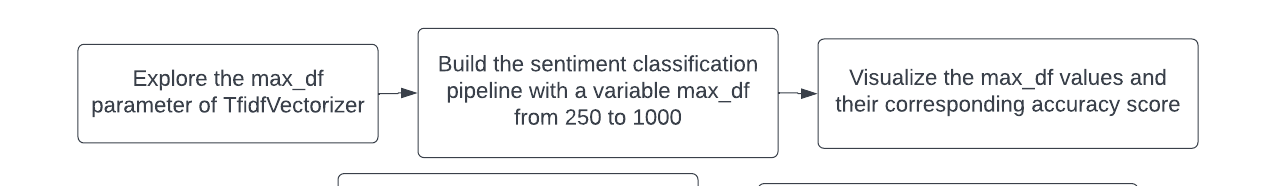


* + - Created a new feature “Length” of the number of characters in each review.
      * Plotted a histogram of the reviews’ length.
  + Build a pipeline with sklearn Pipeline.
  + Define the n\_gram parameter.
  + Used grid search to find the optimal parameters.
  + Predict the test values with grid\_search.predict().
  + Print out the classification report.
  + Plot out the confusion matrix.
  + Print out the accuracy score and F1 score of our pipeline.

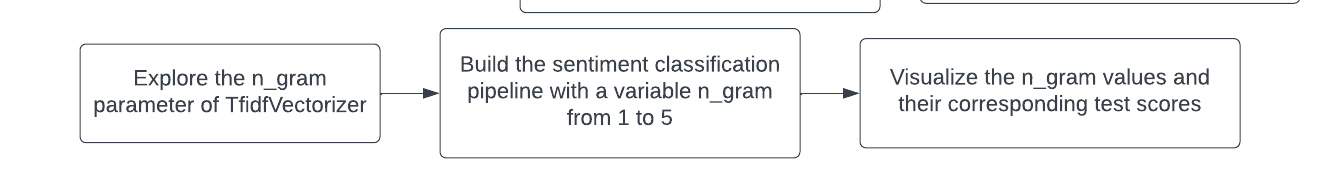
**Problem 2:**



* Used a for loop to build the sentiment classification pipeline with various min\_df values.
  + Record the min\_df value and the corresponding accuracy score into separate arrays in each iteration.
* Create a panda DataFrame of min\_df values and their corresponding accuracy scores.
  + Create a line plot of min\_df values vs accuracy scores using matplotlib and seaborn.
  + Get the min\_df value with the highest accuracy score.

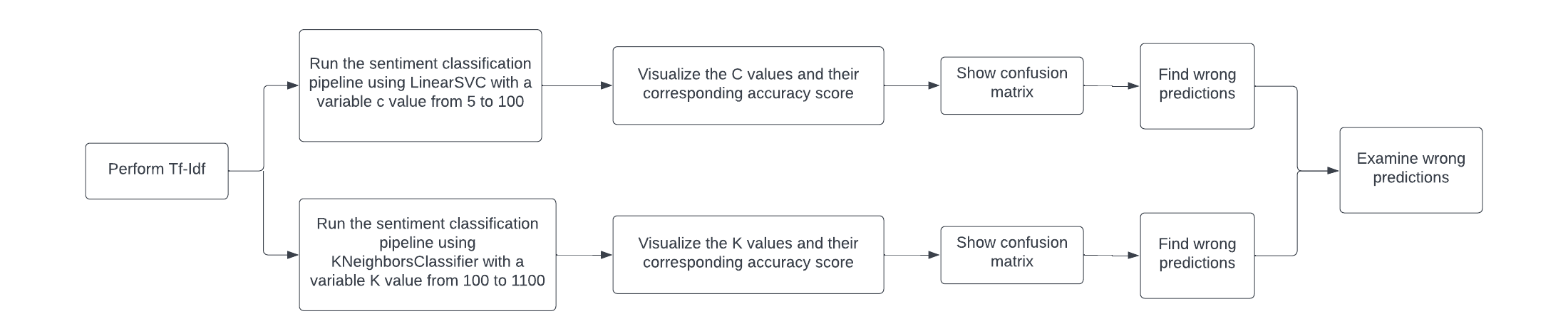


* Used the same approaches above to explore the max\_df parameter.

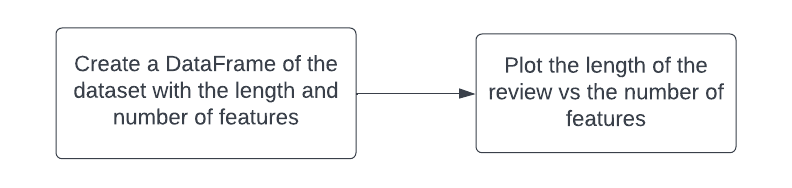


* Manually set 5 different n\_gram parameters from 1, 1 to 1,5.
* Do a gridsearch with all the different parameters.
  + Print out all the n\_gram parameters and their corresponding test scores.
* Print out the accuracy score using the best n\_gram param.

**Problem 3:**



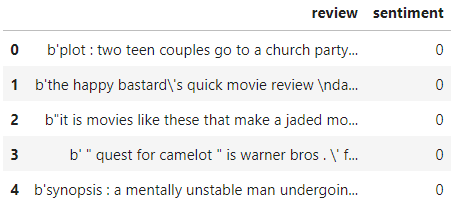
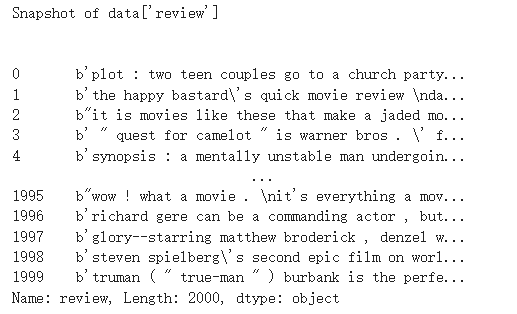
* Performed Tf-Idf
  + Used the best parameters found in Problem 2.
  + Computed Xtrain using fit\_transform on docs\_train.
  + Computed Xtest using transform on docs\_test.
* Used the same methods in Problem 2 when exploring the min\_df and max\_df values on the C value when running the pipeline with LinearSVC.
  + Shows the accuracy and F1 score along with the confusion matrix for LinearSVC.
  + Find wrong predictions by getting the indices where the test dataset doesn’t match with the prediction dataset
* Used the same method above for KNeighborsClassifier with different K values
* Examine wrong predictions
  + Shows the number of wrong predictions using SVC and KNN.
  + Find records where both classifiers failed to make a prediction.

**Problem 4:**

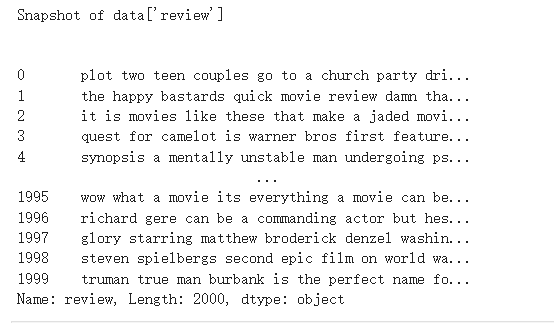
* Create a function that calculates the number of features each reviews
* Create a new array for the number of features for all reviews
  + Iterates through all the rows of the dataset and calculates the number of features using the function created above.
* Add this array as a new column to the DataFrame.
* Used seaborn and matplotlib to make a scatterplot of the length vs number of features.

# Results

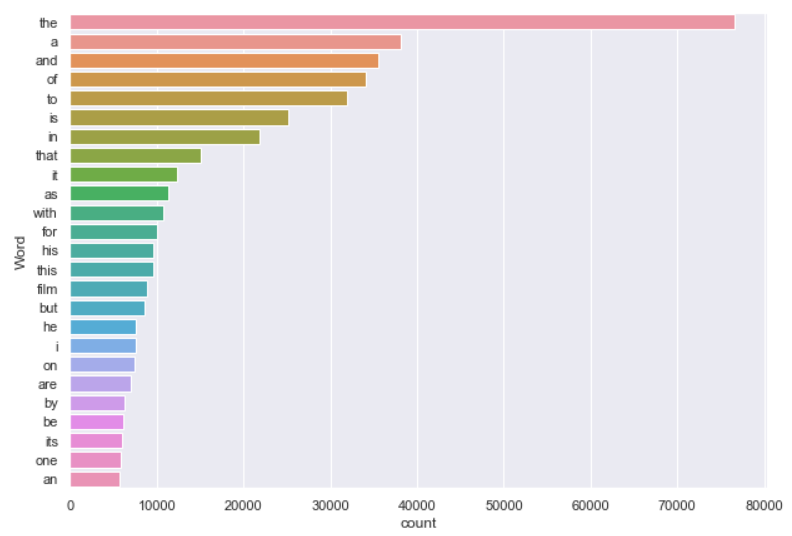
**Problem 1:**

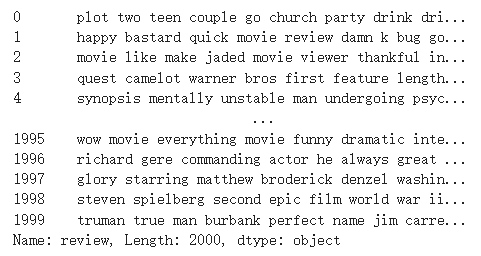
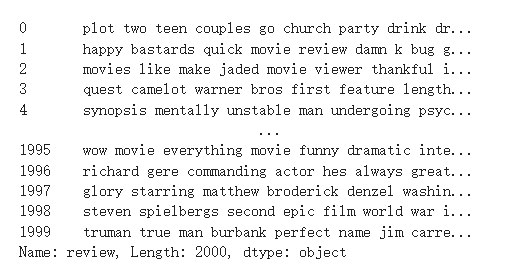
The original review data we connected:

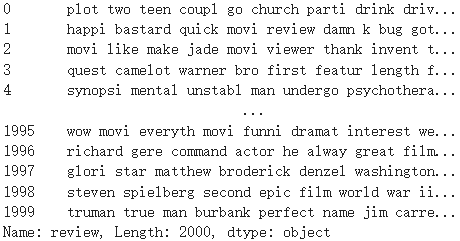
After cleaning the text:



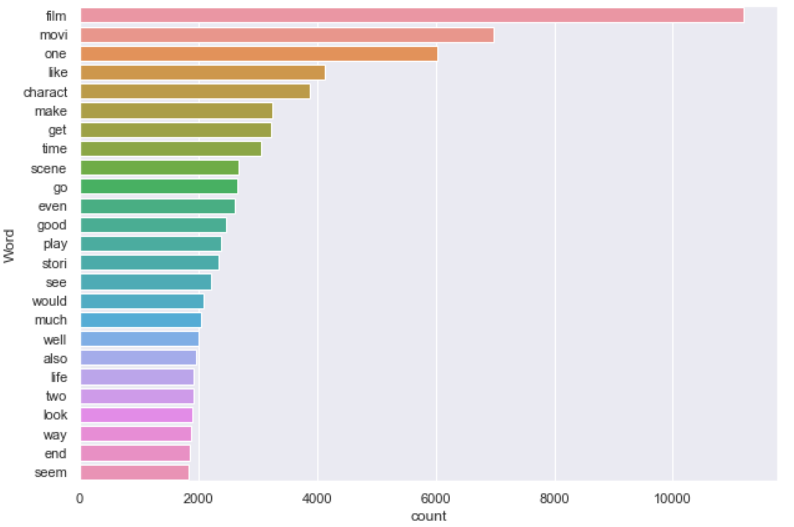
Generate a graph to visualize the words and frequency in data's review:



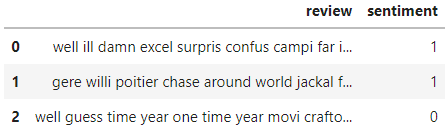
Removing stop words: Lemmatization on Review:

Stemming on Review:

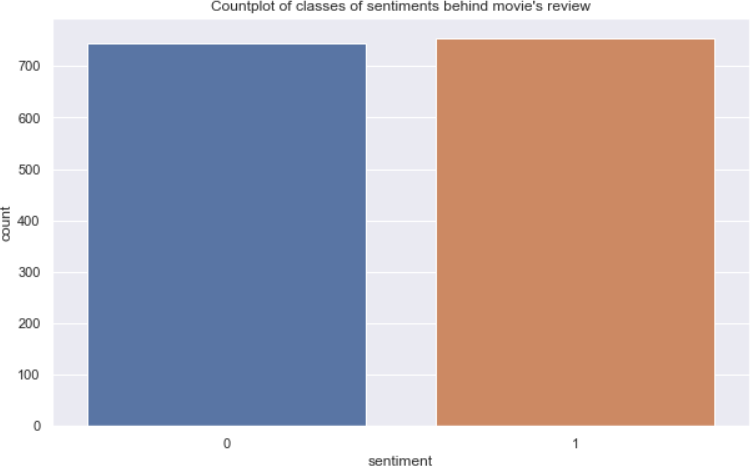
Generate a graph to visualize the words and frequency in data's review:



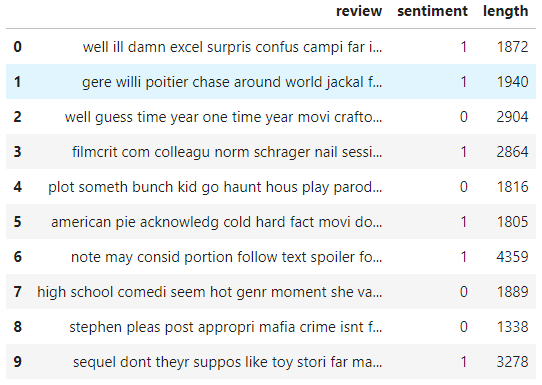
We split the data into training and test set, and shows the first three review in the training set:



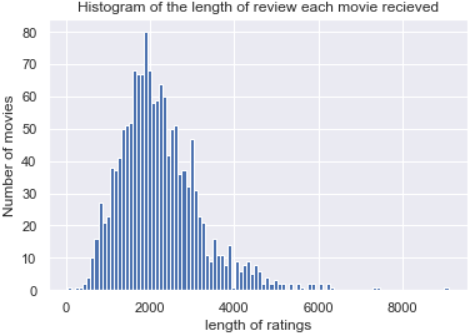
We plot the count plot of the different classes of reviews present:



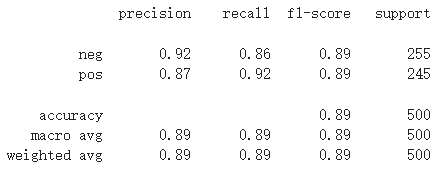
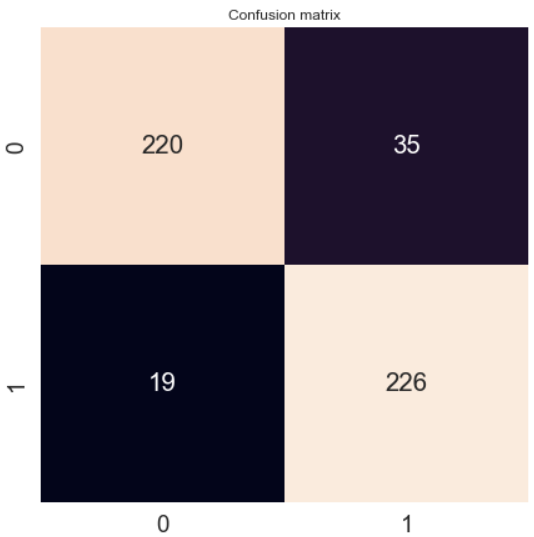
We created a new feature 'length' which is the length of each movie review:



Plotting the histogram of the length of the review each movie received:



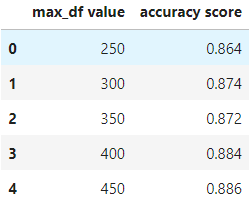
We then predicted the test values and showed the confusion matrix:

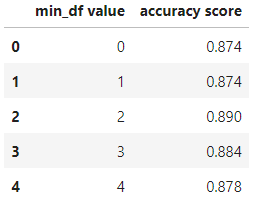


The accuracy score is: 89.2 %

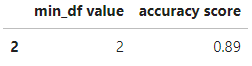
The F1 score is: 89.32806324110672 %

**Problem 2:**

min\_df: max\_df:

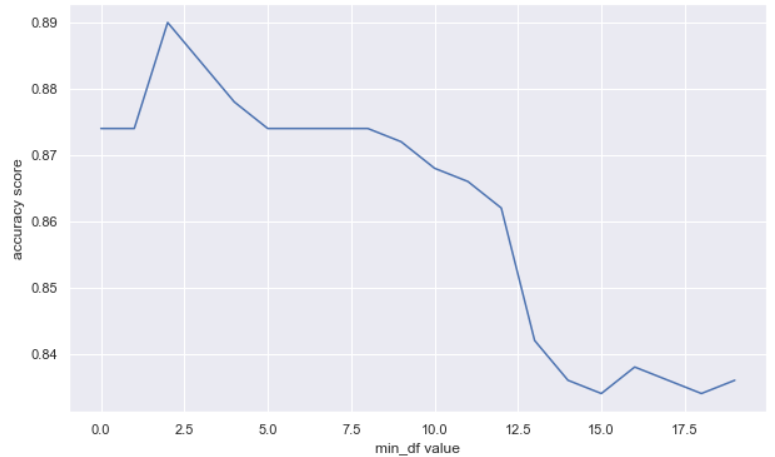


Finding the max value of accuracy for a given min\_df value

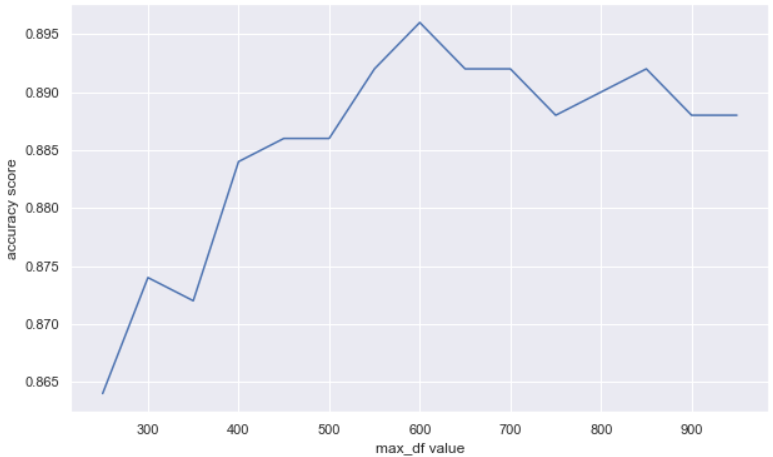


Finding the max value of accuracy for a given max\_df value

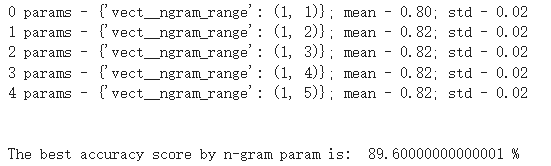


Line plot of min\_df vs accuracy score:

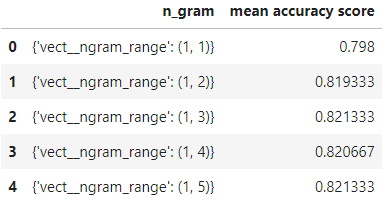
Line plot of min\_df vs accuracy score:



Exploring the n\_gram parameter of TfidfVectorizer:



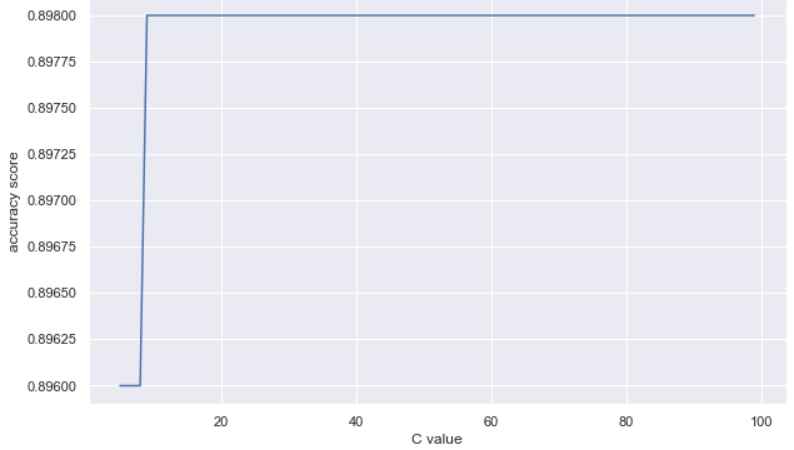
The first 5 n\_gram are:



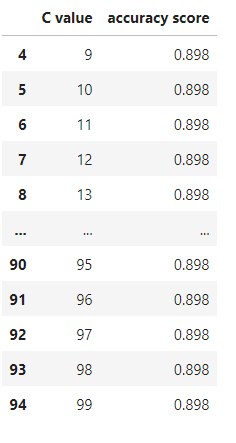
**Problem 3:**

**For LinearSVC**

Plotting a line plot of k value vs accuracy score:



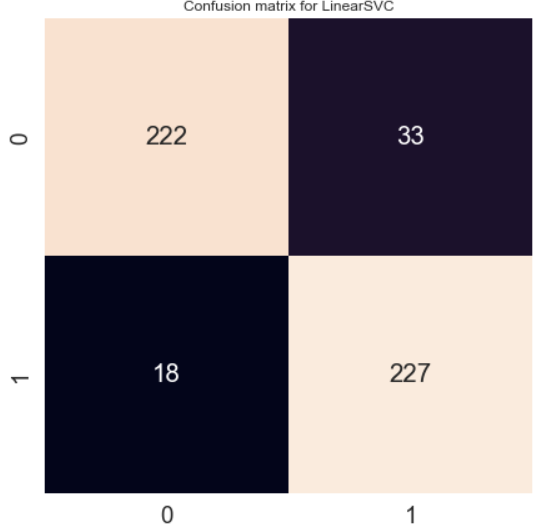
Finding the max accuracy score of k value:



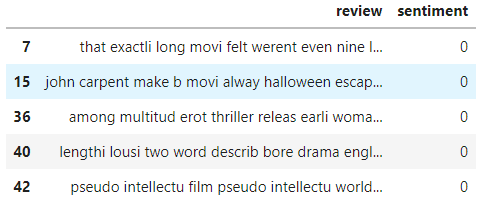
The accuracy score is: 89.8 %.

The F1 score is: 89.72332015810277 %.

The confusion matrix for LinearSVC is:

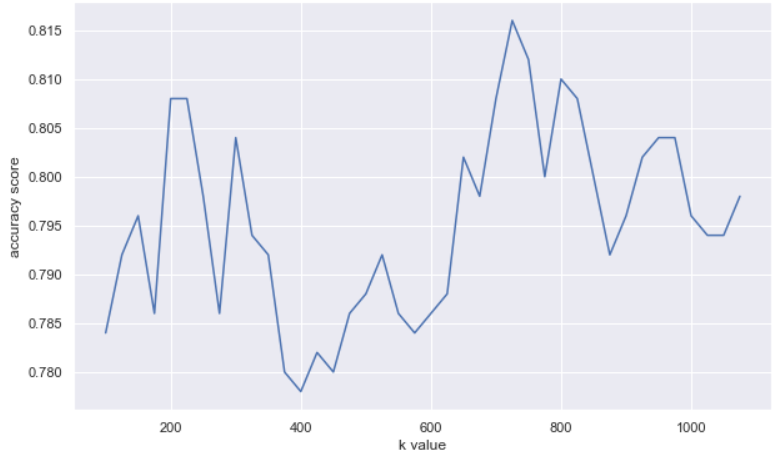


We find the wrong prediction:

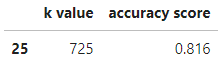


**For KNeighborsClassifier**

Plotting a line plot of k value vs accuracy score:



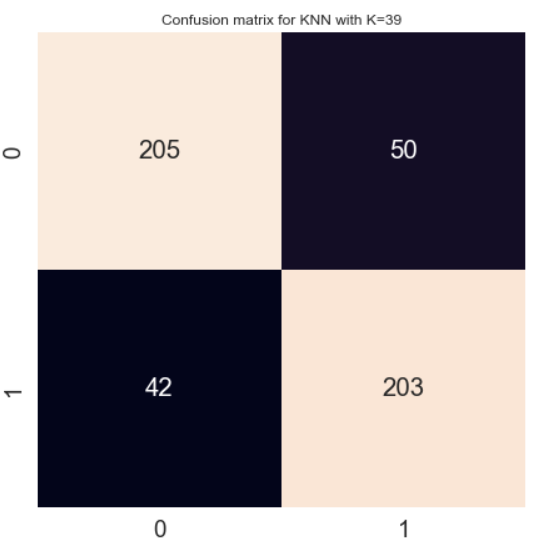
Finding the max accuracy score of k value:



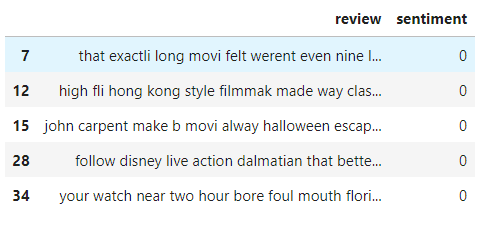
The accuracy score is: 81.6 %.

The F1 score is: 81.52610441767068 %

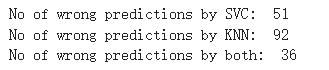
The confusion matrix for KNeighborsClassifier is:



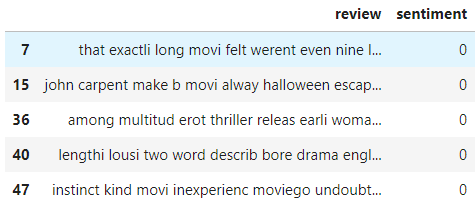
The wrong prediction:



We Find records where both classifiers failed to make prediction:

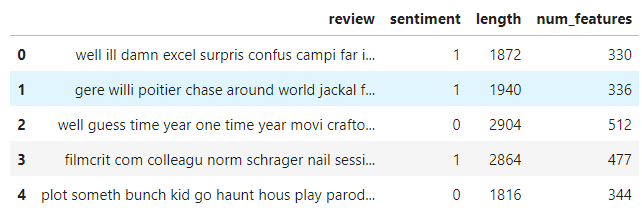


We can see that for all of these records where sentiment = 0 the predictions failed.

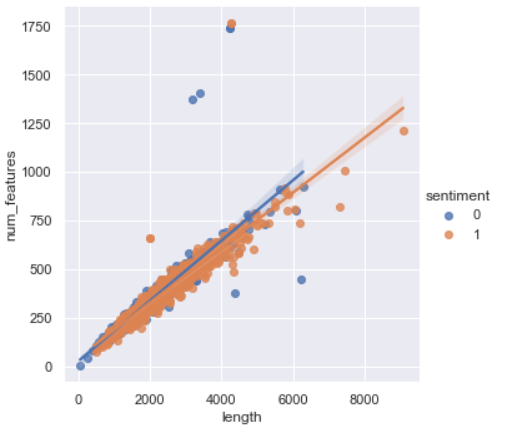


**Problem 4:**

Creating a dataframe of features and length of review:



Plotting the length of the review versus the number of features in that review:



# Conclusion

In conclusion for problem 2, min\_df vs accuracy score is a decreasing plot, which means the word not related to the review because of too infrequently is decreasing. The max\_df vs accuracy is an increasing plot, which means the word not related to the review because of too frequently is increasing. The n-gram is a contiguous sequence of *n* items from the review. Since the accuracy is about 0.89, Using N-gram to keep multiplicity is useful to the reviews. In conclusion for problem 3, We get a larger value of accuracy score and F1 score on LinearSVC than K Neighbors Classifier. Also we have less type I and II error in LinearSVC than K Neighbors Classifier. Thus the method of LinearSVC is better than K Neighbors Classifier. The mistake of predictions might be because of Type II error, since all of the predictions are negative. In conclusion for problem 4, the plot of length and number of features are a very linear plot. We can say that they have a high relationship so we’re unable to separate them using this method.