Movie Review Classification Assignment

Name: Yaswanth Reddy Manukonda
Email: ymanukon@gmu.edu
Miner2 Id: ymanukon
Mason Id: ymanukon
Accuracy: 81
Rank: 42

Problem Statement: given a review, I need to develop a model to identify its polarity (either positive or negative)

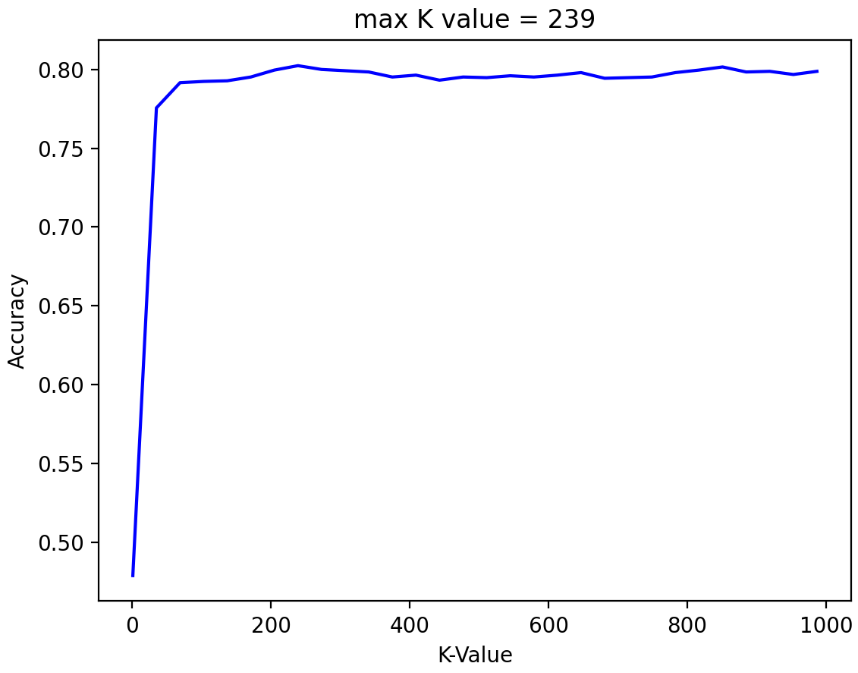
Solution:

1. Preprocess the given reviews
2. convert the given reviews to x_q.
   - where x_q is the vector representation of reviews
3. Now for the x_q, determine its y_q using K-Nearest-Neighbors algorithm.
   - where y_q is the polarity
4. Validate the accuracy using cross validation technique with the help of train data

Explanation:

- For preprocessing, I have followed the below steps and implemented the logic in the function named **preProcess()**
  - Removed HTML tags with the help of regular expressions (re library in python)
  - Removed punctuations or some special characters like ` , ~ , ! , @ , # , $, % , ^ , & , * , (,) , _ , - , + , = , { , , } , } , | , \\ , : , ; , " ," " , < , , , > , . , ? , /
  - Checked for the words not being alphanumeric and have only the English letters
  - Ensured the length of each word is not greater than 2 as there are no adjectives in English with size of word being less than 2
  - Converted the words to lowercase letters
  - Removed stop words ()
  - Finally used the snowball stemmer(porter2) for stemming
- Now that we have the preprocessed data, I have converted the preprocessed data into vectors using **term frequency-inverse document frequency** vectorizer from ***sklearn*** with the parameters being
  - ***min_df:*** it ignores the terms that have a document frequency strictly lower than the given threshold.
  - ***max_features:*** only considers the top max_features ordered by term frequency across the corpus
  - ***max_df:*** I have used default value which is 1. It basically ignores terms that have a document frequency strictly higher than the given threshold
- using TFIDF instead of count vectorizer has increased the accuracy and the efficiency of the program to execute fast.
- by using the cosine similarity from ***sklearn*** library in python on the given test and train data I can get the indices of ***k*** nearest neighbors and based on the polarities of the neighbors I got, I am able to classify the review either positive or negative

- Finally, for validating the algorithm and to get the K value with high accuracy for KNN I have used K-fold cross validation technique with the help of ***KFold*** class from ***sklearn*** and split the given train data with k=10 as it was found to provide good trade-off of low computational cost and low bias in an estimate of model performance
- I have tried cross validating with K in the KNN ranging from 1 to 1001 and found that for K=239 I have got the highest accuracy. Also, I have noticed that this high accuracy is repeated at K around 601
- During this assignment I have learned how cleaning the data impacts the accuracy. I have increased my accuracy by removing punctuations and by using the snowball stemmer(porter2) instead of regular porter algorithm which is known to have better performance and accuracy
- Below is the graph I have plotted for better understanding of the changes in the accuracy with respect to K value



References:

- Brownlee, J. (2020, August 26). *How to configure k-fold cross-validation*. Machine Learning Mastery. Retrieved September 14, 2021, from https://machinelearningmastery.com/how-to-configure-k-fold-cross-validation/.
- *sklearn.model_selection.KFold¶*. scikit. (n.d.). Retrieved September 15, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html.
- *sklearn.metrics.pairwise.cosine_similarity¶*. scikit. (n.d.). Retrieved September 15, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html.
- *snowballstemmer*. PyPI. (n.d.). Retrieved September 15, 2021, from https://pypi.org/project/snowballstemmer/.