Name: Rachana Thota
G-Id: G01237600
Username on miner: noddy
Accuracy: 0.80. **Rank**: 307

Report: Movie Review Classification

Steps Followed to develop KNN classifier:

1. **Pre-processing the data:**

   - To pre-process the data, I have defined a pre-processor function. I have used nltk for word tokenizing and removing stop words and re packages for regular expression support.
   - Initially pre-processor takes each sentence and uses re.sub (regular expressions) package to remove special characters.
   - After that, I have converted the data into lower text and split into individual words
   - Then, I have converted stop words to a set because searching a set is much faster than searching a list.
   - I have removed stop words and 3 letter words and then did stemming using PorterStemmer and lemmatize the review using WordNetLemmatizer
   - Finally, I appended the lemmatized words to processed_features separated by space and return the result.

2. **Vectorizing using TF-IDF vectorizer:**

   - After pre-processing the train data and test data, I have used tf-idf vectorizer class provided by sklearn. Since the data is in text format, tf-idf vectorizers helps to vectorize the documents. It internally uses count vectorizer which produces bag-of-words encoding to enumerate the frequencies of bag words.
   - I have done the vectorizing using tf-idf in TFIDFVectorizing() method. I have fit the train_data in train_matrix using TF-IDF vectorizer. After that, parameters generated from fit () method on train_data is applied upon model to generate transformed data set of test data.

3. **Calculating cosine similarity between vectors:**

   - After generation of vectors, I have used cosine similarity between train_matrix (vectorized train_data) and test_matrix (vectorized test data) using cosine similarity from sklearn metrics. It calculates the angle between each vector and gives efficient distance as NumPy array.

4. **Prediction: K fold Cross Validation:**

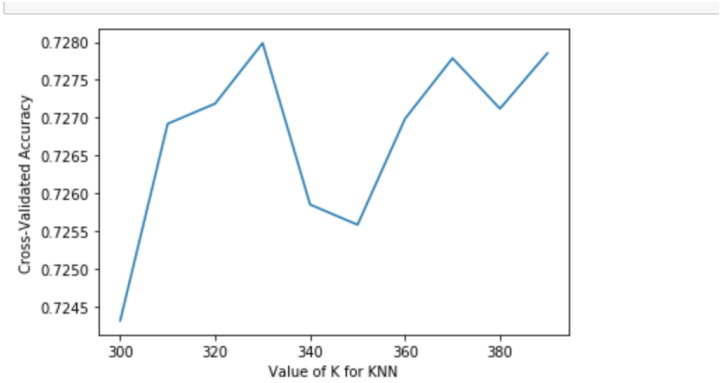
   - I have used k-fold cross validation for model selection.
   - I have used K Fold provided by sklearn. model selection.
   - Firstly, I split the train data into X_train, X_test, Y_train and Y_test using K-Fold.Split() function.
   - X_train and X_test data are the reviews of train_data which is pre-processed. Y_train and Y_test are the sentiments of train_data.
   - In get_score() method, for every split I have done TF-IDF vectorizer on train data and calculated cosine distance between vectors. After that, calculated k neighbors for every similarity vector.
   - After that, I have iterated each similarity to predict the sentiment of each review using predict () method. Appended ‘1’ for positive reviews and ‘-1’ for negative reviews.
   - Then, I have calculated the score between predicted sentiments of train_data and actual sentiments of train_data to get the accuracy for each k-value.
   - Finally, I calculated the average score of all k-values and determined accuracy for each k-fold.
   - As I kept the range 300-400 with increment of 10, it does 10 iterations and gives 10 k-fold avg mean scores. After multiple trials, I got accurate scores in range from 300 to 400

Name: Rachana Thota
G-Id: G01237600
Username on miner: noddy
Accuracy: 0.80. **Rank**: 307

```
For value of k= 370 the accuracy is : 0.815
For value of k= 370 the accuracy is : 0.8002667555851951
For value of k= 370 the kfold accuracy is : [0.724320106702234, 0.7269201733911304, 0.7271866177614761, 0.72798655107
25798, 0.7258531510503501, 0.725586373235523, 0.7269866177614761, 0.7277866844503723]
For value of k= 380 the accuracy is : 0.5186666666666667
For value of k= 380 the accuracy is : 0.7286666666666667
For value of k= 380 the accuracy is : 0.7736666666666666
For value of k= 380 the accuracy is : 0.814
For value of k= 380 the accuracy is : 0.8006002000666889
For value of k= 380 the kfold accuracy is : [0.724320106702234, 0.7269201733911304, 0.7271866177614761, 0.72798655107
25798, 0.7258531510503501, 0.725586373235523, 0.7269866177614761, 0.7277866844503723, 0.7271200400133377]
For value of k= 390 the accuracy is : 0.5136666666666667
For value of k= 390 the accuracy is : 0.7313333333333333
For value of k= 390 the accuracy is : 0.7753333333333333
For value of k= 390 the accuracy is : 0.8153333333333334
For value of k= 390 the accuracy is : 0.8036012004001334
For value of k= 390 the kfold accuracy is : [0.724320106702234, 0.7269201733911304, 0.7271866177614761, 0.72798655107
25798, 0.7258531510503501, 0.725586373235523, 0.7269866177614761, 0.7277866844503723, 0.7271200400133377, 0.727853573
41336]
```

From the above picture, we can see the average values each k-fold. This tells me for which value of k in KNN, I can get the highest accuracy.

I have also plotted graph between value of k for KNN and mean score accuracy for every k-fold to get the clear picture



K value for KNN	Accuracy
330	0.7279865510725798
350	0.725586373235523
370	0.7277866844503723
390	0.72785357341336

From the above data, we can see that highest accuracy is obtained from **k=370** value.

5. KNN Classifier:

- After k-fold cross-validation, I have determined the k value. With **k=370**, I have done the KNN classification. Firstly, I calculated tf-idf for vectorization. and computed cosine distance between vectors and stored as an array. For each similarity, I calculated KNearest using **np.argsort()** method and predicted sentiment for each review by appending '1' for positive review and '-1' for negative review.
- Once it is done, it will print 'Finished'.
- Finally, loaded the result in 'output.txt' file.