**HW1 - Movie Review Classification**

Sahil Shrikrishna Zele (G01465963)

**Objective:**

To implement the Nearest Neighbor Classification Algorithm on movie reviews to predict the sentiments.

**Introduction:**

Movie reviews are a great way to determine the performance of a movie. Although the reviews have numerical indicators which can give you a basic insight in a quantitative sense. The written reviews dive deeper into a lot of emotional aspects and in general if the movie met the expectations of the reviewer or not. We will Perform Sentiment Analysis on the Movie reviews to understand and determine in a binary manner if the reviewer was happy (+1) or sad (-1).

**Dataset:**

The dataset provided consists of written movie reviews and a numerical value along with them which tells us about the sentiments associated. Since the movie review has traces of html tags, we can presume that the data was gathered from movie review websites such as ‘imdb’ or ‘rotten tomatoes’.

The Dataset provided has 3 files mentioned below:

‘Trainset.txt’ – Consist of 25k rows of movie reviews having a written reviews and a sentiment value.

‘Testset.txt’ – Consist of 25k rows of movie reviews having only the written review but not the sentiment values which we will be predicting in this HW.

‘Format.txt; - Sample format file which we will refer to create our own format (Output) file. This file will consist of the sentiment values which we will predict using our knn model.

**Approach:**

Since we will be performing sentiment analysis on movie reviews, firstly we will split the sentiment score from the review text and then we will work on review column to prepare an acceptable input for our training model.

we will be following the below approach:

1. **Data Cleaning and Pre-processing:** Since the dataset was gathered directly from website, what we have now is a raw data and will have a lot of noise and anomalies. For our model to get trained well and provide accurate results, we will have to perform data cleaning process such as:
2. **Removing HTML Tags:** Since we got our data from webpages, we need to remove all the tags from our review data. We will be using regex for the same,
3. **Removing Stop words:** We will remove all the unnecessary words, conjunctions such as ‘is’, ‘the’, ‘get’, ‘and’, etc. we will use Stop words api from NLTK Library.
4. **Removing Special characters/numbers:** since special characters and numbers will be noise for our training, we will remove those using regex.
5. **Lower case:** we will change all the review text to lower case to reduce any possible errors and it will be useful in future for TF-IDF. We will use regex for this as well.
6. **Stemming:** Perform Stemming process on the datasets to remove the present participles such as ‘ing’, ‘ed’, etc, we will be using PorterStemmer api from NLTK library to perform this operation.
7. **TF-IDF Vectorization:** We will calculate the term frequencies and inverse document frequencies which will assign a weight to each word, this numerical value will define the importance of that word. Once we calculate the numerical value, we will create a training and testing vector and provide this as an input to our Training model. We will be using TFidfVectorizer from sklearn kit.

Once all the above operations are completed, we will have a result as cleaned dataset. We will perform the same steps for test data.

1. **Hyper parameter tuning:** Hyper parameter tuning finds us correct values and parameters to be applied for our model to give more accurate results. It will help us figure parameters such as no. of neighbors, weights, and metric etc.
2. **RandomSearchCV:** We will be using RandomSearchCV from sklearn kit, it performs randomized search over parameters to find the best accuracy for our model. It is mostly used when we have a large dataset, and any exhaustive search or brute force is not feasible in terms of run time as well as computing resources.
3. **Cross validation:** Once we find the best parameters using hyper parameter tuning, we will now create a new knn model object and push all parameters we gathered from tuning and calculate a cross validation score. We will use cross\_val\_score method from sklearn kit, this will run our model process onto several subsets of our training dataset and provide an estimate on our model quality and accuracy score.

Once we get our accuracy score from cross validation and hyper parameter tuning, we can manually update the values such as tf-idf max\_features, n\_neighbours, distance, metric, etc. to increase accuracy.

**Classification model:** Once we found correct parameters, we can create our model object and fit the training dataset vector into the model, next we will predict the y test values using the x test vector and store it into a list. We will then move that list into our output file.

**End analysis:** Once we get the predicted y test result, we can check the accuracy of our model and tune the model to improve our accuracy. Knn model is a simple classification algorithm. Using the below parameters, we have received accuracy of around 81%.

Max\_features: 2500

Ngram\_rnage: (1,2)

weights: uniform

metric: Euclidean

Miner Username: sahilzele

Accuracy on miner: 0.81

Below plotted graph is a representation from selecting n\_neighbors and the accuracy we received using those parameters.

A graph with a line

Description automatically generated