Movie Origin and Period Prediction Model Using Movie Plot Description

- Analysis of the Dataset: The movie dataset contains information about movies released in different parts of the world from the period of 1900-2017. The dataset contains the following fields about each movie:

  Release Year: Year of release of the movie.
  Title: Title of the movie.
  Origin/Ethnicity: The place where the movie is made and released.
  Director: Name of the Director of the movie.
  Cast: Lead Actors of the Movie.
  Genre: Genre category of the movie.
  Wiki Page: Wikipedia page of the movie.
  Plot: Description about the movie's storyline.

  The dataset is a very rich in terms or information and provides a lot of scope for performing various analysis.

- Main Problem Statement: Leveraging the movie's plot description to detect the origin and period of the movie.

- Approach:

  (1) Importing Packages and Libraries: First I started with importing the major libraries and packages that I had to use while working on the problem statement. Some important libraries and packages that have been used are Regex, NLTK and Scikit Learn

  (2) Reading the data: After importing the libraries I read the data. The data is available in csv format which makes it very easy for reading on python.

  (3) Data pre-processing and cleaning: After reading the data into Python, it was important for me to process the movie plot description because most of the analysis is based on it. After analysing the Movie Plots, I realized that it contains a lot of noise and information that might not help in the analysis and thus it was necessary to get rid of that. Thus, I first started with complete cleaning of the data. All the text cleaning was mainly performed using Regex. The steps followed for cleaning the text was as follows.

  Removing Special Characters (@, #, URL links if any): It is important to remove special characters from text data. It is not always possible to read the entire data and check whether there are any special characters or not. I specifically removed all the mentions, hashtags and URLs that might be present in the data.

  Expanding Contractions: There are always some words in text data which are written as "won't", "should've", "I'm". It is important to expand these words to "will not", "should have" and "I am" for clear understanding. This process of expanding these words is called as expanding contractions which I did after removing special characters.

  Removing Punctuations and Numbers: Punctuations and numbers do not add any value when we are analysing text data and usually act as noise in between the words. Eliminating them helps in easily analysing text data.

Upper case to lower case: Some words in the data usually repeat multiple times as “Friends” and “friends”. Although these words have the same spelling and the same meaning, they are considered as separate words because one starts with an upper-case letter and one with a lower-case letter. To avoid this issue, I converted all the upper-case words to lower-case.

Tokenization: Tokenization is the technique of separating a sentence into separate words. Tokenization is usually performed before a lot of other steps like Stemming, Lemmatization, Part of Speech tagging.

Lemmatization: Lemmatization converts words to its root form. So, if there are words like “operate”, “operating”, “operates”, “operation”, “operative”, “operatives”, “operational” it converts all of them to its root form “operate”

After performing the above steps for text clean, I got movie plot description completely without any noise which could then be used for analysis.

(4) Most Common Words used in Movie Plot Description: Before going to the main problem statement I decided to do some preliminary analysis of the movie plot description. First, I decided to find the most common words used in the movie plots. I used a counter to take the counts of the tokenized words. The top 20 used words that I got were:

```
[('the', 759140),
 ('to', 493655),
 ('and', 457955),
 ('a', 447408),
 ('is', 377435),
 ('of', 237569),
 ('in', 216382),
 ('his', 204670),
 ('he', 184499),
 ('her', 158096),
 ('that', 141328),
 ('with', 138944),
 ('him', 104989),
 ('for', 98146),
 ('she', 94422),
 ('by', 94184),
 ('on', 77429),
 ('but', 73544),
 ('who', 73044),
 ('they', 69701)]
```

Although this might not seem as an important step for the problem statement, but it helped me realize that the text data is still not completely clean because it contains a lot of stop words which need to be eliminated.

(5) Removing Stop Words: Words like “to”, “the”, “a”, “am” etc. are commonly used in text data and normal sentences and thus they form the major chunk of text. These words are not highly relevant for any kind of text analysis and thus it is important to eliminate them so that we can focus on more important words. I used the NLTK Stop Words library to eliminate all the stop words from the movie plot description.

After removing the stop words, I again found out the most common words from the movie plot description and this time the results were different. The top 20 used words that I got were:

```
[('ha', 56279),
 ('wa', 33346),
 ('find', 30558),
 ('one', 30538),
 ('get', 29756),
 ('go', 27568),
 ('take', 26869),
 ('father', 23903),
 ('tell', 23563),
 ('back', 22699),
 ('love', 22632),
 ('life', 22263),
 ('friend', 21268),
 ('two', 20304),
 ('time', 19006),
 ('day', 18884),
 ('family', 18584),
 ('man', 18257),
 ('home', 17895),
 ('also', 17539)]
```

This indicates that all the stop words have been successfully eliminated from the movie plots.

(6) Part of Speech Tagging: After removing stop words from the data I tagged all the words with their respective part of speech (noun, verb, adverb, adjective). Part of speech is a crucial step in any text mining application because it helps us in understanding the syntax and the meaning of the word correctly. There are some words which can be used in multiple different ways and part of speech tagging helps us in understanding its correct interpretation. I did Part of Speech tagging using the NLTK POS Tagger.

(7) Term Frequency – Inverse Document Frequency (TFIDF Vectorizer): TFIDF is a vectorization technique which is used to convert sentences into vectors based on the weight of the words used in those sentences. It uses a product of 2 different methods: Term Frequency (TF) and Inverse Document Frequency (IDF). TFIDF relatively assigns weights to each word based on its occurrence and importance.

The parameters that I used for TFIDFVectorizer were min_df = 1, max_df =0.99, n_gram range = (1,1) and max_features = 500

After doing TFIDF, I got 500 most **"important"** words in the movie plots which I can now use for my problem statement. I generated a csv of the TFIDF Vectors.

(8) Generalizing the Origin/ Ethnicity of the movies for analysis: The movie dataset contains movies from 24 different origins from various parts of the world and it is different to build a model with so many different target values. So, I decided to generalize the origin of the movies based on continents. I realized that there are a lot of different regional Indian movies which makes the data quite imbalanced. To balance out the data a bit I decided to have 6 general classes as follows:

African, American, Australian, East Asian, European and South Asian.

I combined the classes as follows:

south_asian = ['Assamese', 'Bangladeshi', 'Bengali', 'Bollywood', 'Kannada', 'Malayalam', 'Maldivian', 'Marathi', 'Punjabi', 'Tamil', 'Telugu']

australian = ['Australian']

american = ['American', 'Canadian']

european = ['British', 'Russian', 'Turkish']

african = ['Egyptian']

east_asian = ['Chinese', 'Filipino', 'Hong Kong', 'Japanese', 'Malaysian', 'South_Korean']

Although the classes were still imbalanced with a lot of movies from South Asia and East Asia as compared to others it helped quite a lot in generalizing the classes.

After generalizing the classes I added this list to the TFIDF Vectorizer and Count Vectorizer Data Frame so that it can be used for model building.

(9) Model Building (Classification based on Origin using Movie Plot Description): According to my assumptions, the imbalance in the data and classes would have led to issues like overfitting. I decided to select a robust algorithm like Support Vector Machines (SVM) which can avoid such issues. SVM is a non-parametric algorithm which does not make prior assumptions of the data and allows it to be robust. SVM also offers a high level of performance on the training data and high efficiency for classifying test data. It avoids issues like overfitting and thus I decided to go ahead with this algorithm.

I trained the classifier on TFIDF vectorizer dataset and tested the performance of the classifier.

The performance of SVM classifier on TFIDF Vectors was as follows:

Accuracy: 73.28%

Precision: 0.70

Recall: 0.73

F1 Score: 0.69

Confusion Matrix:

```
[    0    1    0    0    0    8]
[    0 4969    0  117  112  228]
[    0  159    0    2    3   13]
[    0  516    0  251    6  167]
[    0  897    0   37  168   95]
[    0  375    0   51    9 2282]
```

The entire data was divided into training dataset and testing dataset using Train-Test Split approach with 30% Test data to minimize the errors due to imbalance in the data.

(10) Generalizing the Release Year of the Movies to Quarters (25 years): I wanted to see how much the movie plot description helps in understanding the time in which a movie would have been made and that is why I decided to predict and classify the period of the movie based on the movie plot description. To do this I converted the years to periods as follows:

Movies released from 1900-1925 were converted to "First Quarter of the 20th Century".

Movies released from 1925-1950 were converted to "Second Quarter of the 20th Century".

Movies released from 1950-1975 were converted to "Third Quarter of the 20th Century".

Movies released from 1975-2000 were converted to "Fourth Quarter of the 20th Century".

Movies released from 2000-2025 were converted to "First Quarter of the 21st Century".

After converting the release years to their respective periods, I added this list also to the previously generated TFIDF Vectors.

(11) Model Building (Classification based on Period using Movie): I Implemented SVM classifier again on the data to predict period of the movie using the movie plot description.

The performance of SVM classifier on TFIDF Vectors was as follows:

Accuracy: 48.15%

Precision: 0.46

Recall: 0.48

F1 Score: 0.46

Confusion Matrix:

```
 [    0   31   25   61   48]
 [    0 2735  551  186  297]
 [    0 1286  769  226  370]
 [    0  271  173  822  428]
 [    0  664  358  451  714]
```

- **Conclusion:** Based on the results we observe that text data is highly important and can be considered as one of the most important resources for analysing various things. It is difficult to imagine predicting the origin of the movie from the plot description of the movie. But as we see from the results, movie plots can be used to understand and predict the origin of the movie. We built a classifier which classifies a movie into its origin just from the information of its movie plot. The accuracy that we get is close to 73% with TFIDF Vectors but this can certainly be improved if other features apart from text data are also leveraged.

  We also tried to understand the period of the movie from the movie plot description and build a classifier for that. From the results, I conclude that in case of the period the plot description is not a sufficient feature to predict the origin of the movie. I assume that they have very less correlation. Other features along with movie plots might help a lot in improving the results.