HOMEWORK 2.2

IMDb Movie Review Dataset

Homework 2.2

Using the IMDb Movie Review Dataset, complete the following tasks:

- **Load the Dataset :** Use PySpark to read the CSV file into a DataFrame.
- **Explore the Data**
  - Print the schema of the DataFrame.
  - Display the first 5 rows.
- **Clean the Data:** Remove any rows with null or empty reviews.
- **Analyze the Data**
  - Count the total number of reviews.
  - Count how many reviews are labeled positive and negative.
- **Transform the Data**
  - Create a new column that stores the length of each review (number of characters).
  - Filter and display reviews where the review length is greater than 500 characters.
- **Save Your Results:** Save the cleaned and transformed DataFrame as a new CSV file.

The following are the sections of the solution notebook. It has detailed analysis and documentation of all the steps performed and the results obtained. Here's a list of the same:

1. Setup

2. Load the data

3. Explore the Data
    - 3.1. Print the schema
    - 3.2. Display first 5 rows
    - 3.3. Check data quality and distribution

4. Clean the data
    - 4.1. Remove HTML tags
    - 4.2. Remove URLs
    - 4.3. Remove special characters, handle line breaks & tab spaces
    - 4.4. Convert all text to lowercase
    - 4.5. Tokenisation
    - 4.6. Removing stopwords
    - 4.7. Remove null or empty reviews

5. Analyse the Data
    - 5.1. Total number of reviews
    - 5.2. Count reviews by sentiment
    - 5.3. Display the rows with highest no of stopwords, URLs, HTML tags, special characters and unwanted characters

6. Transform and Feature Engg
    - 6.1. Column for length of each review
    - 6.2. Column for word count for each review
    - 6.3. Text Lemmatisation
    - 6.4. Vectorisation - TF-IDF

7. Some interesting analysis
    7.1. Analyze reviews by sentiment
    7.2. Total unique words in vocabulary
    7.3. Showing the imnportance of data cleaning
    7.4. Top 20 Most Frequent Words
    7.5. Gram analysis

8. Filter and Display data

9. Save cleaned data
    9.1. Saving data as a CSV
    9.2. Saving as a Parquet

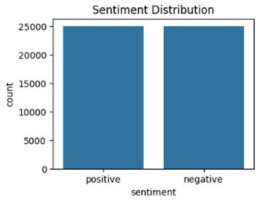
10. Data Visualisation
    10.1. Review Length Distribution
    10.2. Word Cloud
    10.2.1. Word cloud for positive review words
    10.2.2. Word cloud for negative review words

11. PySpark Modelling
    11.1. Preparing feature vector in the format for PySpark models
    11.2. Model

Summary of Inferences

[The detailed analysis can be seen in the .ipynb solution notebook.]

- The dataset is perfectly balanced, with an equal number of positive and negative reviews



- Analysis of review length shows that positive reviews are, on average, slightly longer (844 chars) and contain more words (122 words) than negative reviews (811 chars / 119 words).

summary	review_length	words_count
count	50000	50000
mean	827.29614	120.24452
stddev	639.6723162483631	90.61886164332086
min	17	3
max	9206	1428

Average review length and word count by sentiment:

sentiment	avg_review_length	avg_words_count
positive	843.7912	121.62704
negative	810.80108	118.862

- N-gram analysis is highly effective at finding predictive phrases.
    - Some interesting frequent trigrams of "positive" reviews include: "well worth watching", "seen long time", "one best movie", "one best film", "best movie ever", "based true story", "highly recommend movie"
    - Some interesting frequent trigrams of "negative" reviews include: "worst movie ever", "movie ever seen", "dont waste time", "one worst movie", "movie ive ever", "worst film ever", "doesnt make sense", "bad acting bad", "complete waste time"

Top 20 POSITIVE Tri-Grams

trigram	count
ive ever seen	363
new york city	193
film ever made	165
film ive seen	164
world war ii	157
one best movie	157
one best film	149
movie ever seen	148
movie ive seen	135
based true story	132
movie ever made	121
first time saw	116
best movie ever	105
first saw movie	94
movie ive ever	91
dont get wrong	91
well worth watching	90
seen long time	90
highly recommend movie	83
international film festival	82

Top 20 NEGATIVE Tri-Grams

trigram	count
ive ever seen	673
worst movie ever	563
movie ever seen	394
dont waste time	357
one worst movie	322
movie ive ever	274
worst film ever	266
movie ever made	236
worst movie ive	233
film ever seen	195
one worst film	179
dont get wrong	178
waste time money	178
movie ive seen	168
film ever made	153
bad acting bad	125
film ive ever	125
complete waste time	122
worst film ive	108
doesnt make sense	107

- Importance of data cleaning & pre-processing:

Top 20 Words: Original vs. Filtered vs. Lemmatized (Entire Corpus)

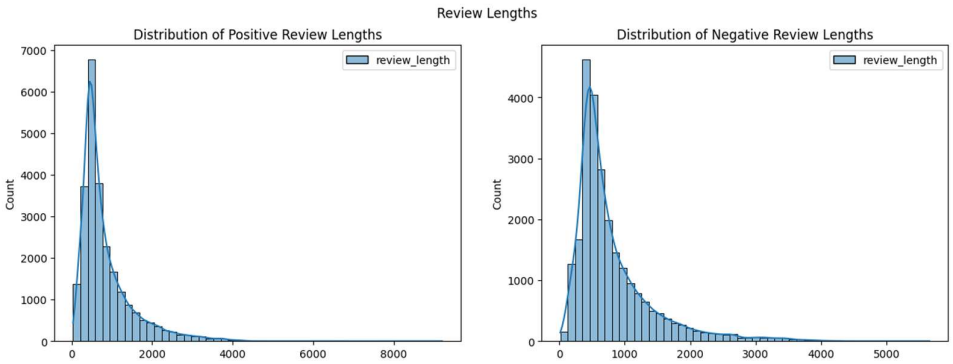
rank	original_word	original_count	filtered_word	filtered_count	lemmatized_word	lemmatized_count
1	the	664007	movie	85233	movie	100952
2	and	320725	film	76033	film	91571
3	a	320606	one	51479	one	53810
4	of	288488	like	39063	like	40019
5	to	266936	good	28913	time	30247
6	is	210515	even	24584	good	29030
7	in	185075	time	23969	character	27981
8	it	154921	really	23013	story	24738
9	i	152148	see	22640	even	24585
10	this	149907	story	22519	get	24516
11	that	136538	well	19225	see	23699
12	was	95405	much	19108	make	23622
13	as	91248	get	18244	really	23013
14	with	87005	bad	18002	scene	21086
15	for	86838	great	17936	well	19390
16	movie	85233	also	17859	much	19108
17	but	81790	people	17709	people	18210
18	film	76033	first	17207	great	18017
19	on	67004	dont	16975	bad	18003
20	not	60055	movies	15719	also	17859

Inference Note:

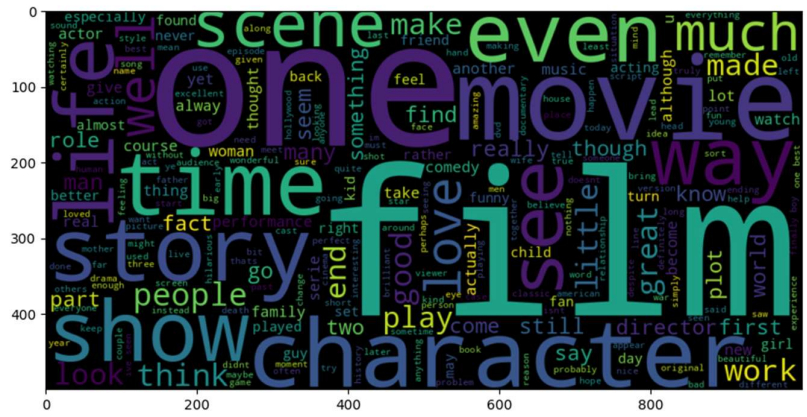
- In original words ranking: all unnecessary words (stopwords) are listed, removing these in data preprocessing has made a drastic significance by reducing the corpus size and increasing attention to useful words
- In the filtered words ranking (after removing HTML tags, URLs, stopwords, etc): the word "film" takes second place because the word "films" (count=15538) and "film" (count=76033) are treated as separate words. This problem is solved in lemmatised ranking. Same with "movie" and "movies" (ranked 1 and 20 in filtered words ranking) are combined into the base lemma "movie" in the lemmatised ranking.

Some interesting visualisations:

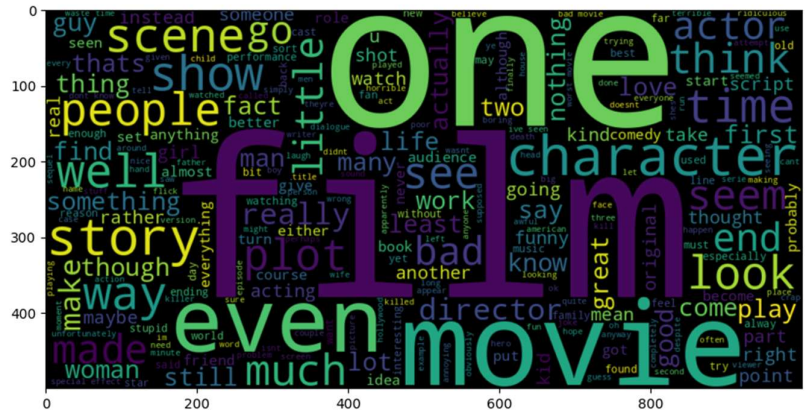
- Distribution of review length by sentiment



- Word cloud for positive review lemmas



- Word cloud for negative lemmas



Lemmatization: This is a crucial feature engineering step. Its importance is:

- Reduces vocabulary size: It consolidates words (e.g., "movie," "movies") into one base form ("movie"), making the feature vector smaller and more efficient.
- More accurate TF-IDF scores: By consolidating all variations of the word, it increases the frequency and TF-IDF score of the core lemma.
- Improves Accuracy: It helps the model generalize by treating different forms of a word as the same, strengthening the pattern between a phrase and its sentiment.

TF-IDF vectorisation:

- To Feed the model: PySpark ML model cannot process words/lemmas directly. It requires numerical representation.
- To give importance to the signal, not the noise: TF-IDF automatically punishes the noise (common) words (low IDF score) and boosts the signal (rare/unique) words (high IDF score).

PySpark Modelling:

Test Set Area Under ROC (AUC): 87.57%

```
Model Predictions:
+-----+----------+--------------------+
|label|prediction|         probability|
+-----+----------+--------------------+
|  0.0|       1.0|[1.95321693541597...|
|  1.0|       1.0|[1.16355879607933...|
|  0.0|       0.0|           [1.0,0.0]|
|  1.0|       0.0|           [1.0,0.0]|
|  1.0|       1.0|[6.61239504574131...|
|  1.0|       1.0|           [0.0,1.0]|
|  1.0|       0.0|           [1.0,0.0]|
|  0.0|       0.0|           [1.0,0.0]|
|  0.0|       0.0|           [1.0,0.0]|
|  0.0|       0.0|[0.99900731490708...|
+-----+----------+--------------------+
only showing top 10 rows
```

- Nithya (1RV22CS099)