ASSIGNMENT 3

CSE 587 - DATA INTENSIVE COMPUTING

Video Link: Assignment3.mp4
(https://drive.google.com/open?id=1KqrUl8HBnKLMs6kIF0xp2Mgaflvs1C4Z)

VERSION

Java - Java version 8

Pyspark - 2.4.0

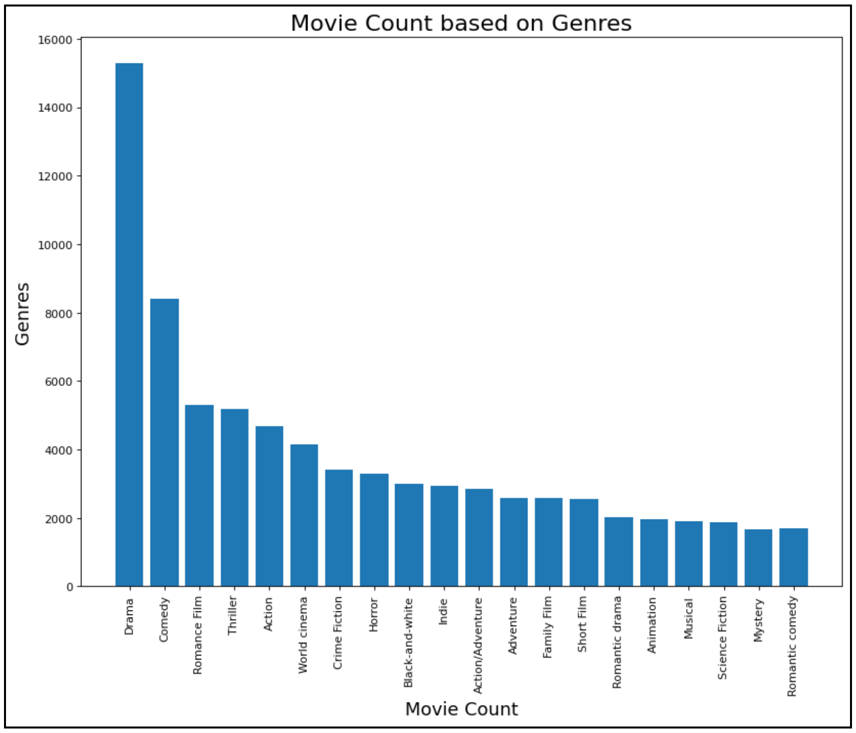
Hadoop - 2.7

Part 1 - Basic Model

Goal: To predict the genre associated with a movie from the corresponding movie plot**.**

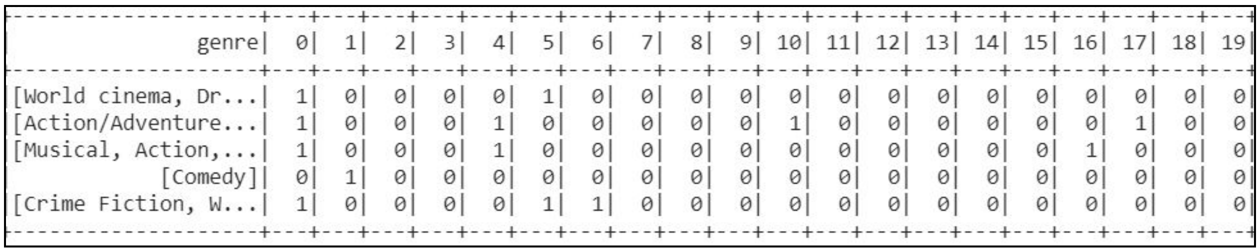
Dataset: Movie Genre Prediction

Language: Pyspark



Data Preprocessing steps:

1. We imported both train and test data using Pandas and we used SQLContext functionality to convert into dataframe(as per pyspark functionality).
2. We removed ‘NA’ values from both train and test dataset.
3. We removed unnecessary space and special characters from plot and genre variables.
4. We are assigning integer values to each genre and performing one-hot encoding.



```
+--------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               genre|  0|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12| 13| 14| 15| 16| 17| 18| 19|
+--------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|[World cinema, Dr...|  1|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|[Action/Adventure...|  1|  0|  0|  0|  1|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  1|  0|  0|
|[Musical, Action,...|  1|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|
|            [Comedy]|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
|[Crime Fiction, W...|  1|  0|  0|  0|  0|  1|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
+--------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

5. We have used logistic regression to train the model.
6. We have used the following functions to improve the accuracy of the model:
   6.1. Stringindexer-Converts a single column to an index column.

   6.2. Tokenizer- Tokenizer is the process of taking text (such as a sentence) and breaking it into individual terms (usually words). A simple Tokenizer class provides this functionality. The example below shows how to split sentences into sequences of words.

   6.3. Stopwordsremover-StopWordsRemover takes as input a sequence of strings (e.g. the output of a Tokenizer) and drops all the stop words from the input sequences. The list of stopwords is specified by the stopWords parameter. Default stop words for some languages are accessible by calling StopWordsRemover

   6.4. CountVectorizer- CountVectorizer converts text documents to vectors which give information of token counts.

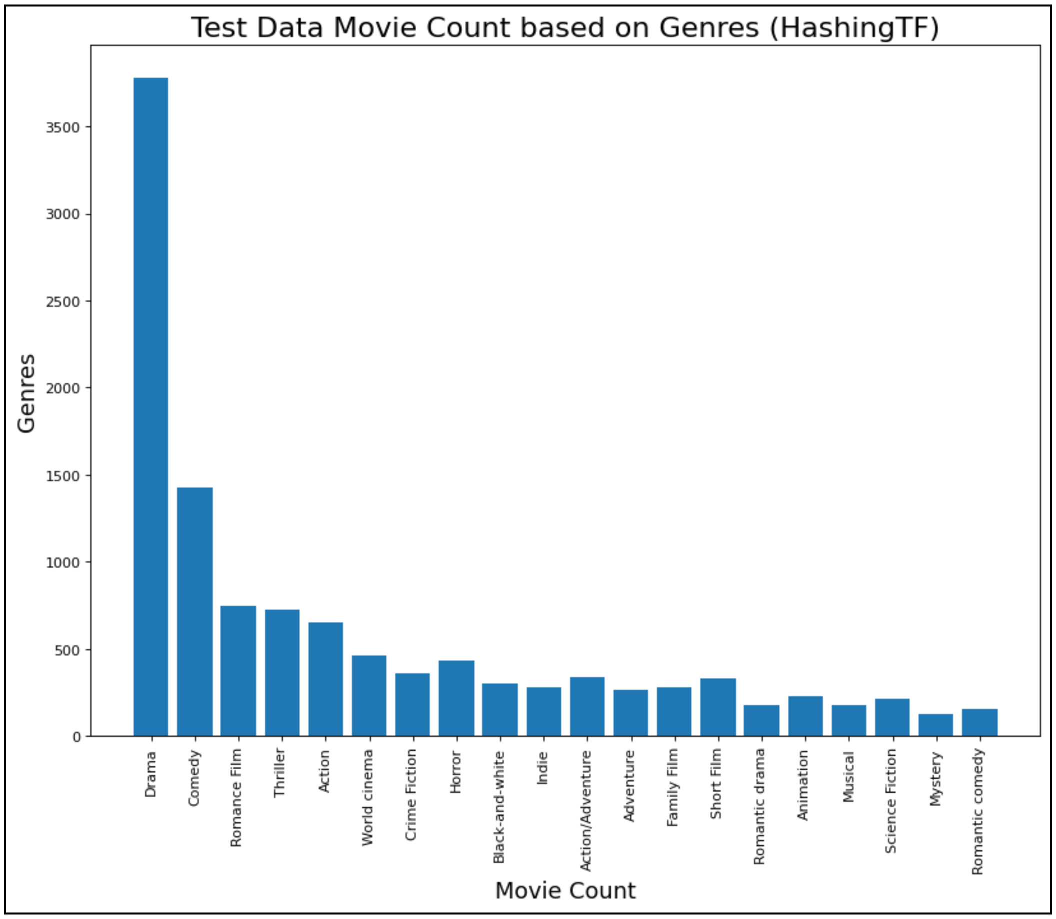
   6.5 HashTF- Maps a sequence of terms to their term frequencies using the hashing trick.

Model Building:

Logistic regression - Logistic regression is another technique borrowed by machine learning from the field of statistics.It is the go-to method for classification problems.

Steps:

1. After applying these functions we fitted the logistic regression and trained the model.
2. After training the model we predicted the values in the test data for each genre one by one and finally combined all genre prediction dataframe.



Part 2 - TF-IDF

Goal: To improve performance of the model using TF-IDF(Term Frequency-Inverse Document)

Term frequency-inverse document frequency (TF-IDF) is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus. Denote a term by t, a document by d, and the corpus by D. Term frequency TF(t,d) is the number of times that term t appears in document d, while document frequency DF(t,D) is the number of documents that contains term t. If we only use term frequency to measure the importance, it is very easy to over-emphasize terms that appear very often but carry little information about the document, e.g. “a”, “the”, and “of”. If a term appears very often across the corpus, it means it doesn’t carry special information about a particular document. Inverse document frequency is a numerical measure of how much information a term provides:

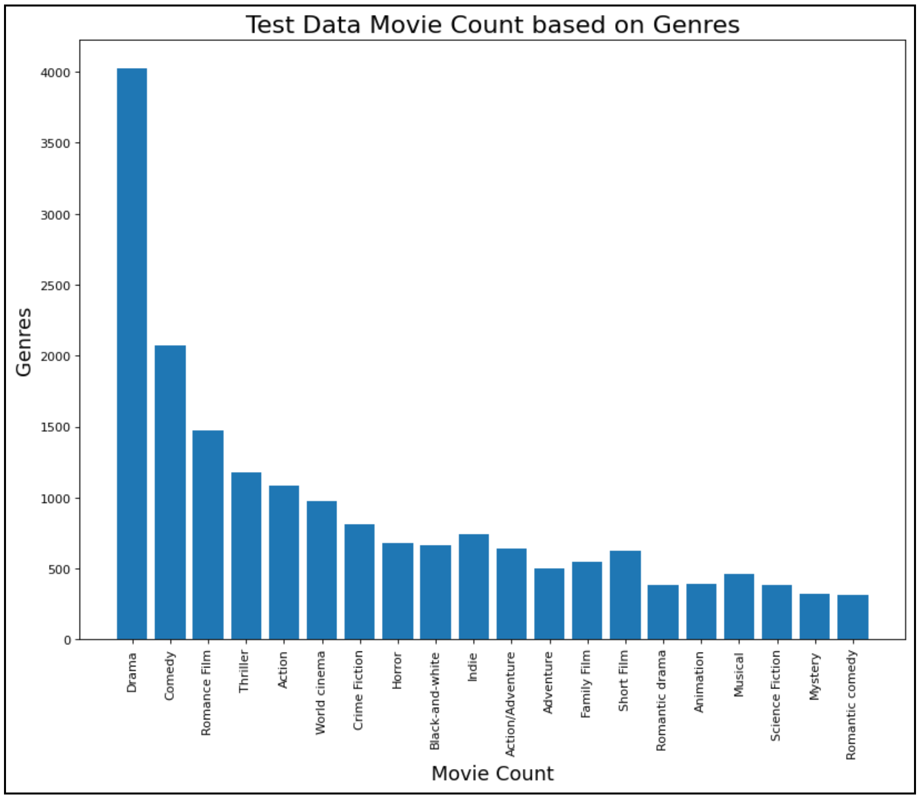
$$IDF(t,D) = \log \frac{|D|+1}{DF(t,D)+1},$$

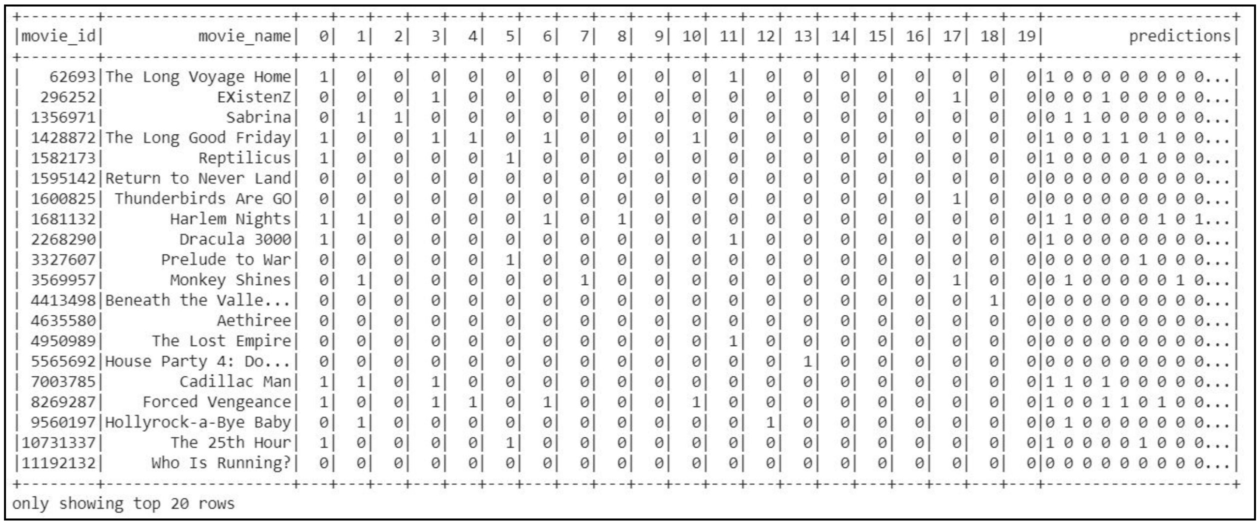
where |D| is the total number of documents in the corpus. Since logarithm is used, if a term appears in all documents, its IDF value becomes 0. Note that a smoothing term is applied to avoid dividing by zero for terms outside the corpus. The TF-IDF measure is simply the product of TF and IDF:

There are several variants on the definition of term frequency and document frequency. In MLlib, we separate TF and IDF to make them flexible.
Both HashingTF and CountVectorizer can be used to generate the term frequency vectors.

IDF:
IDF is an Estimator which is fit on a dataset and produces an IDFModel. The IDFModel takes feature vectors (generally created from HashingTF or CountVectorizer) and scales each feature. Intuitively, it down-weights features which appear frequently in a corpus.
From the logistic regression model which we did in the first part we used IDF function in this part in order to check if the performance of the model improves.

We have basically implemented the IDF function in this part in order to check if the performance of the model improves.
The performance of the model has improved from that of the previous part when using TF-IDF based on the F1 scored metric used in kaggle.





```
+--------+--------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--------------------+
|movie_id|          movie_name|  0|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12| 13| 14| 15| 16| 17| 18| 19|         predictions|
+--------+--------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--------------------+
|   62693|The Long Voyage Home|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|1 0 0 0 0 0 0 0 0...|
|  296252|            EXistenZ|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|0 0 0 1 0 0 0 0 0...|
| 1356971|             Sabrina|  0|  1|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|0 1 1 0 0 0 0 0 0...|
| 1428872|The Long Good Friday|  1|  0|  0|  1|  1|  0|  1|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|1 0 0 1 1 0 1 0 0...|
| 1582173|          Reptilicus|  1|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|1 0 0 0 0 1 0 0 0...|
| 1595142|Return to Never Land|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|0 0 0 0 0 0 0 0 0...|
| 1600825| Thunderbirds Are GO|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|0 0 0 0 0 0 0 0 0...|
| 1681132|       Harlem Nights|  1|  1|  0|  0|  0|  0|  1|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|1 1 0 0 0 0 1 0 1...|
| 2268290|        Dracula 3000|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|1 0 0 0 0 0 0 0 0...|
| 3327607|      Prelude to War|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|0 0 0 0 0 1 0 0 0...|
| 3569957|       Monkey Shines|  0|  1|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|0 1 0 0 0 0 0 1 0...|
| 4413498|Beneath the Valle...|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|0 0 0 0 0 0 0 0 0...|
| 4635580|            Aethiree|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|0 0 0 0 0 0 0 0 0...|
| 4950989|     The Lost Empire|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|0 0 0 0 0 0 0 0 0...|
| 5565692|House Party 4: Do...|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|0 0 0 0 0 0 0 0 0...|
| 7003785|        Cadillac Man|  1|  1|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|1 1 0 1 0 0 0 0 0...|
| 8269287|    Forced Vengeance|  1|  0|  0|  1|  1|  0|  1|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|1 0 0 1 1 0 1 0 0...|
| 9560197|Hollyrock-a-Bye Baby|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|0 1 0 0 0 0 0 0 0...|
|10731337|       The 25th Hour|  1|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|1 0 0 0 0 1 0 0 0...|
|11192132|     Who Is Running?|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|0 0 0 0 0 0 0 0 0...|
+--------+--------------------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--------------------+
only showing top 20 rows
```

Part 3 - Custom Feature Engineering

Goal: To implement any one of the modern text-based feature engineering methodologies to improve the performance of the model.

We have implemented word2vec function to validate if the model performance has improved.

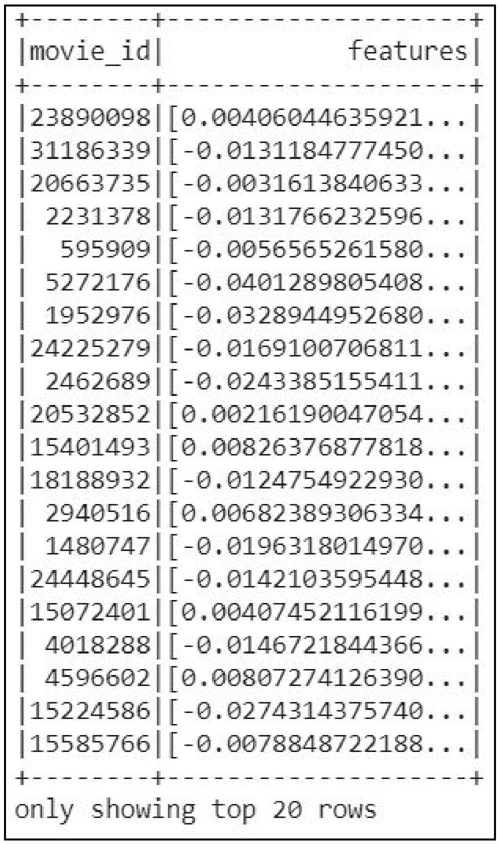
Word2vec:

Word2Vec is an Estimator which takes sequences of words representing documents and trains a Word2VecModel. The model maps each word to a unique fixed-size vector. The Word2VecModel transforms each document into a vector using the average of all words in the document; this vector can then be used as features for prediction, document similarity calculations, etc.

From the logistic regression model which we did in the first part we used Word2vec feature engineering technique to check if the performance of our model improves from that of using TF-IDF.

Steps:

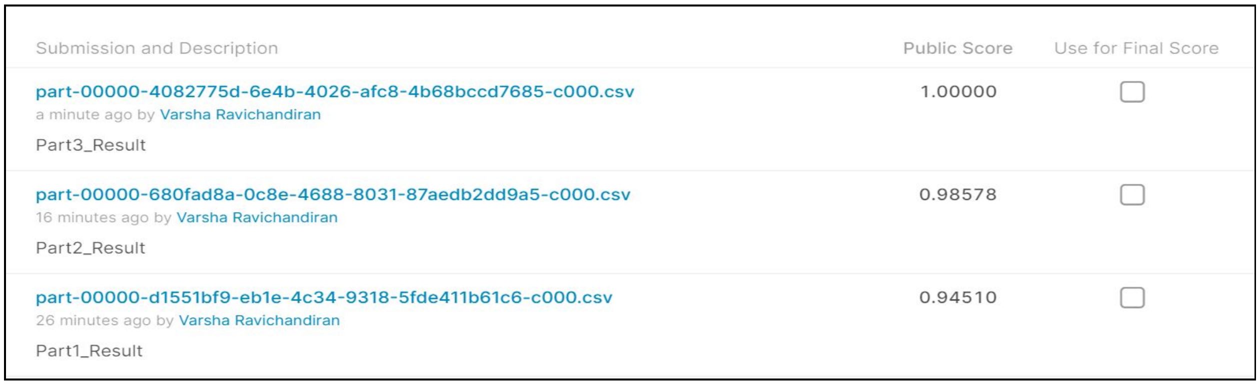
1. We tokenized the features and performed word2vec custom feature engineering and fitted logistic regression into it.



```
+--------+--------------------+
|movie_id|            features|
+--------+--------------------+
|23890098|[0.00406044635921...|
|31186339|[-0.0131184777450...|
|20663735|[-0.0031613840633...|
| 2231378|[-0.0131766232596...|
|  595909|[-0.0056565261580...|
| 5272176|[-0.0401289805408...|
| 1952976|[-0.0328944952680...|
|24225279|[-0.0169100706811...|
| 2462689|[-0.0243385155411...|
|20532852|[0.00216190047054...|
|15401493|[0.00826376877818...|
|18188932|[-0.0124754922930...|
| 2940516|[0.00682389306334...|
| 1480747|[-0.0196318014970...|
|24448645|[-0.0142103595448...|
|15072401|[0.00407452116199...|
| 4018288|[-0.0146721844366...|
| 4596602|[0.00807274126390...|
|15224586|[-0.0274314375740...|
|15585766|[-0.0078848722188...|
+--------+--------------------+
only showing top 20 rows
```

Results:

Team Name: Movie Blazers



Submission and Description	Public Score	Use for Final Score
part-00000-4082775d-6e4b-4026-afc8-4b68bccd7685-c000.csv a minute ago by Varsha Ravichandiran Part3_Result	1.00000	☐
part-00000-680fad8a-0c8e-4688-8031-87aedb2dd9a5-c000.csv 16 minutes ago by Varsha Ravichandiran Part2_Result	0.98578	☐
part-00000-d1551bf9-eb1e-4c34-9318-5fde411b61c6-c000.csv 26 minutes ago by Varsha Ravichandiran Part1_Result	0.94510	☐

The above submission of our results in kaggle.

1. In part-1 we used HashTF and fitted with logistic regression so the F1 Score is 0.94510.
2. In part-2 we used TF-IDF and observed that the F1 score has increased from 0.98578.So the performance of the logistic regression is improved when we used the IDF function in spark.
3. In part-3 we used word2vec custom feature engineering and we see that the performance of the model has improved as we observe that the F1 score is 1.

Team Members:

Team Members	UB IT Name	UB Email
Kundan Kumar	kkumar5	kkumar5@buffalo.edu
Varsha Ravichandiran	varshara	varshara@buffalo.edu
Viswanathan Gopalakrishnan	vgopalak	vgopalak@buffalo.edu