ASSIGNMENT 3
REPORT

CSE 587 : DATA INTENSIVE COMPUTING

Mrunal Inge (50337040)
Akash Malla (50336850)
Rohit (50336890)

PART 1 - Basic Model - 5 points ● Analyze the data and preprocess it if needed ● Create a machine learning model (use any algorithm) in spark to use the information provided in the train set to predict the genres associated with a movie. ● You should create a term-document matrix from the plots and use these as feature vectors for the machine learning model. ● Generate predictions for the test set and upload to Kaggle website ● Report macro F1 score obtained for your submission from the Kaggle website

Setting Up:
We have installed the Java Jdk 8 version and set the path for JAVA_Home and numpy and pandas in our VM.

Steps:
To create a basic model for movie genre prediction we have used a binary relevance method which treats each label as a separate single classification problem. The key assumption here though, is that there is no correlation among the various labels. Binary relevance is used along with logistic regression. The union of all classes that were predicted is then taken as the multi-label output

We have created two data frames using the spark framework which are data_spark_df to hold the training data and lables_spark_df to store the labels of the movie genre. A matrix is created with all the 20 genres present attributing it to the training data by iterating through a for loop. The two data frames are later merged and are sent into the **regexTokenizer**, with input as the plot and output as words. Stop words are removed using stopWordsRemover. HashingTF is used taking input as words and output columns as raw features, which has vector size, vector indices and values as it's attributes. The **regexTokenizer**, **stopWordsRemover**, **HashingTF** will later be put in a pipeline, which is further used to fit the training data to create our model, named as model. The same process will be applied to the test data as well, naming it as model2.

Logistic regression is performed on each label in labelCols and prediction of each label is added to dfList. dfList is then converted to temp_df which consists of movie_id and all the predictions against 20 labels. Then it is converted to a csv file i.e. here "predictions_part1.csv", which contains movie id and predictions (size 20) as specified in sample.csv. The macro f1 score for part1 is **0.97688**.

movie_id	movie_name	plot	genre	tokens	features	col0	col1	col2	col3	col4	col5	col6	col7	c
18549958	Love, Mary	Mary was a lonely...	['Drama']	[mary, was, a, lo...	[0.01836837841808...	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	
3150865	Yankee Doodle Daffy	Porky Pig is tryi...	['Animation', 'Co...	[porky, pig, is, ...	[0.01646497736710...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
4345210	Learning to Lie	Beginning in West...	['Romance Film']	[beginning, in, w...	[0.02494957366407...	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	
14072881	Boys' Reformatory	Seventeen-year-ol...	['Black-and-white...	[seventeen-year-o...	[0.05170378282405...	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	
4711636	The Last Flight o...	A jaded pilot nam...	['Adventure']	[a, jaded, pilot,...	[0.11341146605629...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
27181651	Esther	Hadassah , a beau...	['Drama', 'Crime ...	[hadassah, ,, a, ...	[0.00289057773415...	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	
26551541	Triple Trouble	The boys are on t...	['Crime Fiction',...	[the, boys, are, ...	[0.12233029621281...	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	
24070825	Summer Rain	The film takes pl...	['Romance Film', ...	[the, film, takes...	[0.06771354891592...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
30897463	Makaramanju	The film tells th...	['Romance Film']	[the, film, tells...	[0.07137968416809...	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
5349196	Arul	Arul ([[Vikram , ...	['Musical', 'Worl...	[arul, ([[vikram,...	[0.05471918750700...	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	
32751781	Bush Christmas	In the Australian...	['Adventure', 'Fa...	[in, the, austral...	[0.17701008167902...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
28733517	A Town Like Alice	In Post-WW2 Londo...	['Drama', 'Romanc...	[in, post-ww2, lo...	[0.07779930366627...	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	
11453926	Getting Acquainted	Charlie and his w...	['Black-and-white...	[charlie, and, hi...	[0.07277435439295...	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
16190026	Unstable Fables: ...	July 2012}} Goldi...	['Animation', 'Fa...	[july, 2012}}, go...	[0.07488096769944...	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	
11517297	The Law of the Range	Betty Dallas is a...	['Indie']	[betty, dallas, i...	[0.12247199819916...	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	
3492883	Kronk's New Groove	Emperor Kuzco na...	['Animation', 'Fa...	[emperor, kuzco, ...	[0.05642987596858...	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
2257809	Judas Kiss	Coco Chavez and ...	['Crime Fiction',...	[coco, chavez, , ...	[0.10230088605231...	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	
20318100	Breaking the Rules	Phil , reunites w...	['Drama']	[phil, ,, reunite...	[0.04354403735769...	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	
2918524	Secret Admirer	Michael Ryan is ...	['Indie', 'Comedy...	[michael, ryan, ,...	[-0.0228847870235...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
13607881	The Barbarians	The film is set i...	['Adventure', 'Ac...	[the, film, is, s...	[0.09912138784472...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	

Fig: Final training data after join performed

col0	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14	col15	col16	col17	col18	col19
0.0	0.0	1.0	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	0.0	0.0	0.0
0.0	0.0	0.0	1.0	1.0	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	0.0
0.0	0.0	1.0	1.0	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	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	1.0	1.0	1.0	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	0.0	0.0
0.0	1.0	1.0	0.0	1.0	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	0.0
0.0	0.0	1.0	0.0	1.0	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	0.0
0.0	1.0	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	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	1.0	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
0.0	1.0	1.0	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	0.0	0.0	0.0
0.0	1.0	1.0	1.0	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	0.0	0.0
0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	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	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	1.0	0.0	1.0	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	0.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	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	0.0	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	1.0	0.0	1.0	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	0.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	1.0	1.0	1.0	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	0.0	0.0
0.0	1.0	0.0	1.0	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	0.0	0.0

only showing top 20 rows

Fig: Label Matrix

PART 2 - Use TF-IDF to improve the model - 7 points ● Focussing on the summary of the movie, implement Term Frequency-Inverse Document Frequency (TF-IDF) based feature engineering technique to improve the performance of the model ● Similar to part 1, generate predictions and upload to the Kaggle website ● Ideally, your model should improve performance from the previous step

Setting Up:
We have installed the Java Jdk 8 version and set the path for JAVA_Home and numpy and pandas in our VM.

There are two data frames created using the spark framework. data_spark_df, which holds the training data and lables_spark_df, which stores the labels of the movie genre. A matrix is created with all the 20 genres present attributing it to the training data by iterating through a for loop. The two data frames are later merged and are sent into the regexTokenizer, with input as the plot and output as words. HashingTF is used taking input as words and output columns as raw features, which has vector size, vector indices and values as it's attributes. Furthermore, an idf has been initialized taking input as the raw features and output as features. The **regexTokenizer, HashingTF, idf** will later be put in a pipeline, which is further used to fit the training data to create our model, named as model. The same process will be applied to the test data as well, naming it as model2.

The **Logistic Regression** model has been used to fit the training data set and obtain the predictions of test data using the transform function. A temp_df is created which consists of movie_id and all the predictions against 20 labels. Then it is converted to a csv file, which contains movie id and predictions (size 20) as specified in sample.csv. The macro f1 score for "predictions_2.csv" is **0.98280**

col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14	col15	col16	col17	col18	col19	col20
0.0	0.0	1.0	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	0.0	0.0	0.0
0.0	0.0	0.0	1.0	1.0	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	0.0
0.0	0.0	1.0	1.0	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	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	1.0	1.0	1.0	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	0.0	0.0
0.0	1.0	1.0	0.0	1.0	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	0.0
0.0	0.0	1.0	0.0	1.0	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	0.0
0.0	1.0	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	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	1.0	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
0.0	1.0	1.0	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	0.0	0.0	0.0
0.0	1.0	1.0	1.0	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	0.0	0.0
0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	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	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	1.0	0.0	1.0	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	0.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	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	0.0	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	1.0	0.0	1.0	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	0.0	0.0
0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	1.0	1.0	1.0	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	0.0	0.0
0.0	1.0	0.0	1.0	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	0.0	0.0

Fig: Label Matrix

PART 3 - Custom Feature Engineering - 8 Points ● Implement any one of the modern text-based feature engineering methodology to improve the performance of the model ● Some of the methods to consider are (But not limited to) ○ Word2vec ○ Glove ○ Doc2vec ○ Topic Modelling ○ etc… ● This is an open-ended part of the assignment, where you are free to explore any new text processing methodology to create custom features ● Custom feature engineering would be deemed successful only if the model performs better than the model of part 2

Setting Up:
We have installed the Java Jdk 8 version and set the path for JAVA_Home and numpy and pandas in our VM.

Steps:
To further improve the performance of the model, we have used **word2vec** as our modern text-based feature methodology. The flow continues the same as that of part-2 with few additional changes in the code. After the two data frames being created using the smart framework: data_spark_df and lables_spark_df, and after the matrix being created with all the 20 genres present attributing it to the training data by iterating through a for loop, the data frames are later merged and is sent into the Tokenizer. Instead of using HashingTF, idf, it has been replaced by word2Vec, which accepts words as input column

and output column as features. The **Tokenizer and word2Vec** will later be put in a pipeline, which is further used to fit the training data to create our model, named as model. The same process will be applied to the test data as well, naming it as model2.

A **Logistic Regression** model has been used to fit the training data set and obtain the predictions of test data using the transform function. In the final csv file, a list named dfList has been created to append the movie id and predictions alongside the genre labels.

The macro f1 score observed for word2Vec is **1.00000**.

BONUS:- 5 Points ● Kaggle maintains a leaderboard of all the submissions ● There are two leaderboards, public, and a private leaderboard ● You would only be able to see the public leaderboard. ● Generally, the ranking in the public leaderboard also reflects in the private leaderboard ● If the performance of your model ranks in the top 10 of the entire class in the private leaderboard, you will get the bonus 5 points

Steps:

To further improve the performance of the model, we have used word2vec as our modern text-based feature methodology. The flow continues the same as that of part-2 with few additional changes in the code. After the two data frames being created using the smart framework: data_spark_df and lables_spark_df, and after the matrix being created with all the 20 genres present attributing it to the training data by iterating through a for loop, the data frames are later merged and is sent into the Tokenizer. Instead of using HashingTF, idf, it has been replaced by **word2Vec**, which accepts words as input column and output column as features. The **Tokenizer** and **word2Vec** will later be put in a pipeline, which is further used to fit the training data to create our model, named as model. The same process will be applied to the test data as well, naming it as model2.

A **Logistic Regression** model has been used to fit the training data set and obtain the predictions of test data using the transform function. In the final csv file, a list named dfList has been created to append the movie id and predictions alongside the genre labels.

The macro f1 score observed after improvising the model to word2Vec is **1.00000**.

External Libraries used:

We used numpy and pandas to perform the initial read operation of the train.csv, test.csv, and mapping.csv files.

Installations to VM:

Installed java version 8 to the VM for spark.
Installed numpy, pandas.

Link of Video -
https://drive.google.com/drive/folders/1RUO4vTE0NoKAMRMZrF-3duG0cbIZ67Ph?usp=sharing

CITATIONS AND REFERENCES:

In Solving this assignment we took help of various tools and resources which are listed below

1.https://spark.apache.org/docs/latest/ml-classification-regression.html
2.https://spark.apache.org/docs/latest/mllib-data-types.html
3.https://spark.apache.org/docs/latest/api/java/org/apache/spark/mllib/linalg/DenseMatrix.html
4.https://spark.apache.org/docs/2.1.1/api/java/org/apache/spark/mllib/feature/Word2Vec.html