DATA INTENSIVE COMPUTING
ASSIGNMENT 3
Predictive Analytics with Spark
Movie Genre Prediction

Prakhathi Murugesan- 5031 6846
Saranya Saravanan- 5031 4931
Vineeth Thayanithi- 5032 0197

OBJECTIVE:

The objective of the Assignment is to create a machine learning model that predicts to which genre a particular movie belongs to given the movie id using the Apache spark. It is a multilabel classification problem, each movie can be predicted to have been belonging to more than one genre.

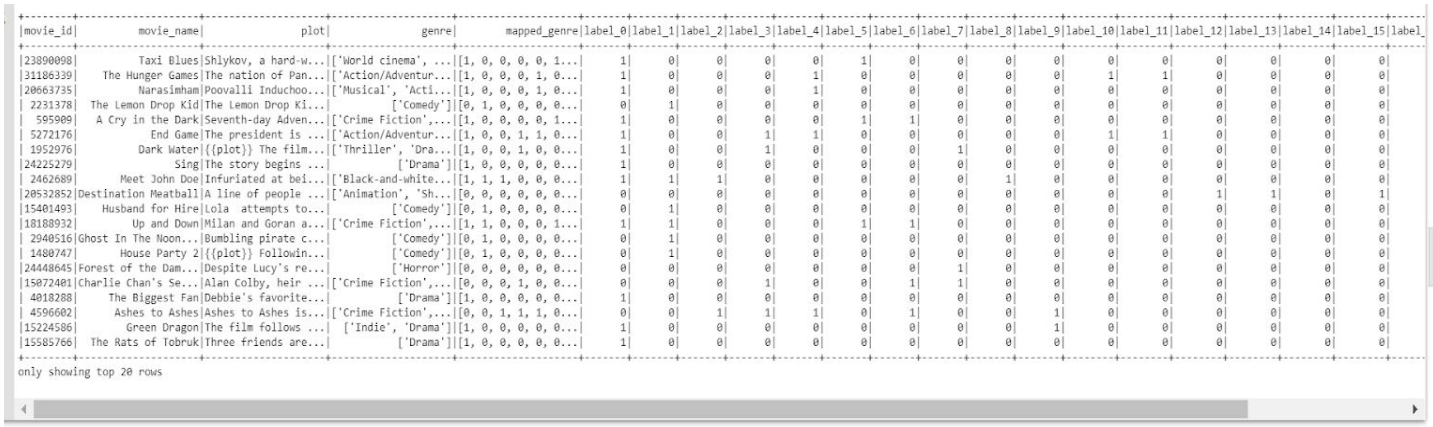
DATASET:

The dataset consists of four csv files train.csv, test.csv, mapping.csv and sample.csv.
Train.csv: This dataset is used to train our predictive model. It consists of the summary and genre for approximately 31K movies.
Test.csv: This contains the movie id and its summary only. The genres for the movie id in this dataset is to be predicted.
Mapping.csv: This contains the mapping of the genre to its string index in which the output is expected to be displayed.
Sample.csv: Contains the sample format of the expected prediction.

PART-1: BASIC MACHINE LEARNING MODEL

Data importing and preprocessing

- The python API Pyspark is imported to the program and used for implementation.
- Pyspark.sql is used for handling the data.
- The spark session is created to indicate the driver program and common Spark properties are set using the sparkcontext().
- The train and test data are imported using the pandas dataframe initially and then converted to spark dataframe using the sqlcontext. Mapping.csv is also imported into a spark dataframe.
- A user defined function is defined for creating a new dataframe genre_mapped which includes an additional column “mapped_genre” to the train dataframe. The “mapped_genre” is of array type with string values. Using the mapping dataset provided for each movie_id in train data it is mapped as 1 if the genre of the movie matches the genre in the mapping dataset and 0 if not matches.
- Each string in the array is split and all the twenty labels are represented as individual columns. These columns are added to the previously processed dataframe. The current dataframe to be used is “result” dataframe.The processed data is as shown below:



movie_id	movie_name	plot	genre	mapped_genre	label_0	label_1	label_2	label_3	label_4	label_5	label_6	label_7	label_8	label_9	label_10	label_11	label_12	label_13	label_14	label_15	label_
23890098	Taxi Blues	Shlykov, a hard-w...	['World cinema', ...	[1, 0, 0, 0, 0, 1...	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
31186339	The Hunger Games	The nation of Pan...	['Action/Adventur...	[1, 0, 0, 0, 1, 0...	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	
20663735	Narasimham	Poovalli Induchoo...	['Musical', 'Acti...	[1, 0, 0, 0, 1, 0...	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
2231378	The Lemon Drop Kid	The Lemon Drop Ki...	['Comedy']	[0, 1, 0, 0, 0, 0...	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
595909	A Cry in the Dark	Seventh-day Adven...	['Crime Fiction',...	[1, 0, 0, 0, 0, 1...	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	
5272176	End Game	The president is ...	['Action/Adventur...	[1, 0, 0, 1, 1, 0...	1	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	
1952976	Dark Water	{{plot}} The film...	['Thriller', 'Dra...	[1, 0, 0, 1, 1, 0...	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	
24225279	Sing	The story begins ...	['Drama']	[1, 0, 0, 0, 0, 0...	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2462689	Meet John Doe	Infuriated at bei...	['Black-and-white...	[1, 1, 1, 0, 0, 0...	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	
20532852	Destination Meatball	A line of people ...	['Animation', 'Sh...	[0, 0, 0, 0, 0, 0...	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	
15401493	Husband for Hire	Lola attempts to...	['Comedy']	[0, 1, 0, 0, 0, 0...	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18188932	Up and Down	Milan and Goran a...	['Crime Fiction',...	[1, 1, 0, 0, 0, 1...	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	
2940516	Ghost In The Noon...	Bumbling pirate c...	['Comedy']	[0, 1, 0, 0, 0, 0...	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1480747	House Party 2	{{plot}} Followin...	['Comedy']	[0, 1, 0, 0, 0, 0...	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
24448645	Forest of the Dam...	Despite Lucy's re...	['Horror']	[0, 0, 0, 0, 0, 0...	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
15072401	Charlie Chan's Se...	Alan Colby, heir ...	['Crime Fiction',...	[0, 0, 0, 1, 0, 0...	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	
4018288	The Biggest Fan	Debbie's favorite...	['Drama']	[1, 0, 0, 0, 0, 0...	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4596602	Ashes to Ashes	Ashes to Ashes is...	['Crime Fiction',...	[0, 0, 1, 1, 1, 0...	0	0	1	1	1	0	1	0	0	1	0	0	0	0	0	0	
15224586	Green Dragon	The film follows ...	['Indie', 'Drama']	[1, 0, 0, 0, 0, 0...	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	
15585766	The Rats of Tobruk	Three friends are...	['Drama']	[1, 0, 0, 0, 0, 0...	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

only showing top 20 rows

Fig: 1.1 Result dataframe from train data

- The “Plot” column containing raw data is preprocessed using a Spark NLP, a natural language processing library built on top of Apache Spark and Spark ML.
- **RegexTokenizer**: Takes “plot” as the input column and identifies tokens with tokenization open standards. The candidates are identified using basic regex rules with “\\w+” as the target pattern. The output column is “words”.
- **StopwordsRemover**: Drops all the stop words from the input column “words” and gives output as type tokens. The output column named “filtered”
- ML pipeline is used for combining the above-mentioned algorithms into a single workflow. The fit and transform are called with the preprocessed train dataframe “result”. Similarly for the test dataset as well.

Creating basic machine learning model

- From the Spark machine learning classification library import the package for logistic regression.
- For each label in the dataset we have selected only the required columns from the train dataframe. The selected columns are movie_id, movie_name, filtered, plot, label.
- The term document matrix here is created using the count vectorizer function. The output column here is “count_features”. This inturn is used as the feature column for generating the logistic regression model.
- The Logistic Regression model is used for multilabel classification. This function takes four arguments.

  Logistic Regression Parameters:

  maxIter:The maximum number of iterations to use = 500
  regParam: Regularization parameter = 0.001
  Featurescol: Features column name, as a length-one character vector = count_features
  labelCol: Label column name = label

- The model is fit with selected train data and transformed with test data to obtain the predictions.

- Movie id with its prediction is selected and appended to a list.
- This process continues till all the twenty labels are executed.

Storing the output prediction in a csv file:

- A dictionary is created to append all the movie id and its prediction as a key value pair.
- To write the output in the specified format, a csv file “sample.csv” is imported and each of the key value pairs from the dictionary is read and written to the csv file using the for loop.

F1 Score(Part 1):

- This score was obtained on submitting the sample.csv output prediction file in kaggle
- F1 score = 0.96326

PART-2 (TF_IDF TO IMPROVE THE BASIC MODEL IN PART-1)

This method provides better accuracy than the basic model. The terms are mapped to indices using a Hash Function.

Data importing and preprocessing

- The data importing and preprocessing steps for this part are similar to that used for Part1 of the assignment. In addition to this hashingtf and idf is performed.
- Import the packages HashingTF and idf from the machine learning library for spark.
- **HashingTF**: All the terms of the document are matched to their frequencies using the HashingTF. The output column “filtered” from the stopwordsremover is passed as the input to the HashingTF. This produces the output column “count_features”.
- **IDF:**The idf conveys the importance of the word in the documents. This inputs the column “count_features” and produces the output column “features” which is later on used in the logistic regression model.
- ML pipeline is used for combining the above-mentioned algorithms into a single workflow. The fit and transform are called with the preprocessed train dataframe “result”. Similarly for the test dataset as well.

Creating improved machine learning model

- From the Spark machine learning classification library import the package for logistic regression.
- For each label in the dataset we have selected only the required columns from the train dataframe. The selected columns are movie_id, movie_name, features, plot, label.
- Output column “features” from idf is used here as the feature column for creating the logistic regression model.
- The program workflow is executed using the ML pipeline.
- The Logistic Regression model is used for multilabel classification. This function takes four arguments.

    Logistic Regression Parameters:

    maxIter:The maximum number of iterations to use = 100

    regParam: Regularization parameter = 0.001

    Featurescol: Features column name, as a length-one character vector = features

labelCol: Label column name = label

- The model is fit with selected train data and transformed with test data to obtain the predictions.
- Movie id with its prediction is selected and appended to a list.
- This process continues till all the twenty labels are executed.

Storing the output predictions is the same as the part1 of the assignment.

F1 Score(Part 2):

- This score was obtained on submitting the sample.csv output prediction file in kaggle
- F1 score = 0.97293

PART-3 (CUSTOM FEATURE ENGINEERING TO IMPROVE THE MODEL IN PART-2)

Data importing and preprocessing

- The data importing and preprocessing steps for this part are similar to that used for Part1 of the assignment except that only RegexTokenizer has been used.
- **RegexTokenizer**: Takes "plot" as the input column and identifies tokens with tokenization open standards. The candidates are identified using basic regex rules with "\\w+" as the target pattern. The output column is "words".
- **Word2Vec** :It takes "words" as the input column and converts word representation to a distributed vector. The output column here is "features".
- This workflow is executed using the machine learning pipeline.
- ML pipeline is used for combining the above-mentioned algorithms into a single workflow. The fit and transform are called with the preprocessed train dataframe "result". Similarly for the test dataset as well.

Creating improved machine learning model

- From the Spark machine learning classification library import the package for logistic regression.
- Import the package Word2Vec from the machine learning library for spark.
- For each label in the dataset we have selected only the required columns from the train dataframe. The selected columns are movie_id, movie_name, features, plot, label.
- The term document matrix here is generated using the Word2Vec.
- The program workflow is executed using the ML pipeline.
- The Logistic Regression model is used for multilabel classification. This function takes four arguments.

  Logistic Regression Parameters:

  maxIter:The maximum number of iterations to use = 1000
  regParam: Regularization parameter = 0.001
  Featurescol: Features column name, as a length-one character vector = features
  labelCol: Label column name = label
- The model is fit with selected train data and transformed with test data to obtain the predictions.
- Movie id with its prediction is selected and appended to a list.

- This process continues till all the twenty labels are executed.
- Storing the output predictions is the same as the part1 of the assignment.

F1 Score(Part 3):

- This score was obtained on submitting the sample.csv output prediction file in kaggle
- F1 score = 1.00