CSE 487/587 ASSIGNMENT 3

BIG DATA PROCESSING WITH SPARK

Implementation of Machine Learning Model for movie genre prediction over the feature space created by the below feature engineering techniques from the given plot of each movie.

Part1: Using Feature Engineering technique : Term-Document Matrix

Part2: Using Feature Engineering technique : Term Frequency-Inverse Document Frequency (TF- IDF)

Part3: Using Feature Engineering technique : Word2Vec

Part4: Obtained trained models from above three parts are used to predict movie genre of the given test data and results obtained from each part are uploaded to Kaggle competition to get F1 score in order to compete.

In this Assignment we have used Apache Spark for implementing end to end predictive Analytics pipeline. We have used several libraries to tackle the given tasks, these libraries are mentioned in later parts of the report.

Java Setup:

We have changed the Java version in the VM from **11 to 8** as this is required for our spark operations.

Part-1:

Libraries loaded:

The following libraries were loaded for this part:

Pandas

RegexTokenizer

StopWordsRemover

CountVectorizer

StringIndexer

IndexToString

RandomForestClassificationModel

RandomForestClassifier

Pipeline

MultiClassClassificationEvaluator

Udf pyspark.sql

Implementation:

Spark session has been initialized and above necessary libraries were loaded.

Data Loading:

Training data has been loaded using pandas and then changed to pyspark data frame (as it is not splitting and reading the training data correctly when tried loading using pyspark).

Data cleaning:

From the training data loaded, we have to change the text information in plot column to feature space. So, firstly the words from the text in plot column as converted to lower and special characters were removed using Regex Tokenizer. Later StopWordsRemover has been used to remove inbuilt and separately defined stop words. By now the text information in plot has been changed to refined list of words.

Feature Engineering using Term-Document Matrix:

These lists of words are now converted into Term-Document matrix using CountVectorizer with minimum DF of value 120 resulting in 4969 length feature space for each document. This term document matrix created consisted the list of occurrences of each word which has a particular order which is consistent across all the documents.

Generating unique labels:

From the list of genres in the 'genre' column of the training data frame, a unique set of labels were assigned for each combination of genre using StringIndexer.

Labels for each combination were stored into a variable in order to restore the combinations from predictions.

Model Fitting:

These features and labels of the modified data frame are used to fit the machine learning model. Random Forest model with 20 number of trees and 100 max Bins was built by fitting on training data.

Tuning Hyper Parameters:

Parameters are tuning based on the accuracies obtained from the model when applied on the validation test set.

Saving the Model:

This random forest model is stored into a file in order to use it later.

Predicting the genre for test data:

This is done in Part4 clearly in a user-friendly way. Same pre-processing methods are applied on the training data and obtain pre-processing model is then fit to the testing data. Then saved model is fit on the modified test data frame.

Extracting genre from Predicted labels:

These predicted labels are converted to genres using IndexToString and the labels saved above.

Converting to Output format:

Mapping.csv file has been loaded into a data frame. These above extracted genres of the test data have been mapped and to respective index value using mapping data frame and the predicted are changed to a binary string.

Save and Submit:

This final predicted data frame of test data is saved into a csv format. This saved result is then submitted to Kaggle.

F1 score of part1:

F1 score obtained to part1 was **1**

Part-2:

Libraries loaded:

The following libraries were loaded for this part:

Pandas

Tokenizer

HashingTF

IDF

StringIndexer

IndexToString

RandomForestClassificationModel

RandomForestClassifier

MultiClassClassificationEvaluator

Udf pyspark.sql

Implementation:

Spark session has been initialized and above necessary libraries were loaded.

Data Loading:

Training data has been loaded using pandas and then changed to pyspark data frame (as it is not splitting and reading the training data correctly when tried loading using pyspark).

Data cleaning and Feature Engineering using TD-IDF:

From the training data loaded, we have to change the text information in plot column to feature space. **Stop words are not removed in this part** because TD IDF implicitly ignores the most repeated words across all documents according to its algorithm. With the help of Tokenizer, HashingTF and IDF, feature engineering has been done.

Generating unique labels:

From the list of genres in the 'genre' column of the training data frame, a unique set of labels were assigned for each combination of genre using StringIndexer.

Labels for each combination were stored into a variable in order to restore the combinations from predictions.

Model Fitting:

These features and labels of the modified data frame are used to fit the machine learning model. Random Forest model with 20 number of trees and 100 max Bins was built by fitting on training data.

Tuning Hyper Parameters:

Parameters are tuning based on the accuracies obtained from the model when applied on the validation test set.

Saving the Model:

This random forest model is stored into a file in order to use it later.

Predicting the genre for test data:

This is done in Part4 clearly in a user-friendly way. Same pre-processing methods are applied on the training data and obtain pre-processing model is then fit to the testing data. Then saved model is fit on the modified test data frame.

Extracting genre from Predicted labels:

These predicted labels are converted to genres using IndexToString and the labels saved above.

Converting to Output format:

Mapping.csv file has been loaded into a data frame. These above extracted genres of the test data have been mapped and to respective index value using mapping data frame and the predicted are changed to a binary string.

Save and Submit:

This final predicted data frame of test data is saved into a csv format. This saved result is then submitted to Kaggle.

F1 score of part2:

F1 score obtained to part2 was **1** **Part-3:**

Libraries loaded:

The following libraries were loaded for this part:

Pandas

RegexTokenizer

StopWordsRemover

Word2Vec

StringIndexer

IndexToString

RandomForestClassificationModel

RandomForestClassifier

MultiClassClassificationEvaluator

Udf pyspark.sql

Implementation:

Spark session has been initialized and above necessary libraries were loaded.

Data Loading:

Training data has been loaded using pandas and then changed to pyspark data frame (as it is not splitting and reading the training data correctly when tried loading using pyspark).

Data cleaning and Feature Engineering using Word2Vec:

From the training data loaded, we have to change the text information in plot column to feature space. **Even Stop words are not removed in this part** because Word2Vec assigns a score of each word based on its preceding and successive words. RegexTokenizer has been used to convert the text into a list of words. By now the text information in plot has been changed to refined list of

words. Later Word2Vec has been applied with vector size of 100 and min count 5 on the obtained list of words. Feature vector space of 100 has been generated for each document.

Generating unique labels:

From the list of genres in the 'genre' column of the training data frame, a unique set of labels were assigned for each combination of genre using StringIndexer.

Labels for each combination were stored into a variable in order to restore the combinations from predictions.

Model Fitting:

These features and labels of the modified data frame are used to fit the machine learning model. Random Forest model with 20 number of trees and 100 max Bins was built by fitting on training data.

Tuning Hyper Parameters:

Parameters are tuning based on the accuracies obtained from the model when applied on the validation test set.

Saving the Model:

This random forest model is stored into a file in order to use it later.

Predicting the genre for test data:

This is done in Part4 clearly in a user-friendly way. Same pre-processing methods are applied on the training data and obtain pre-processing model is then fit to the testing data. Then saved model is fit on the modified test data frame.

Extracting genre from Predicted labels:

These predicted labels are converted to genres using IndexToString and the labels saved above.

Converting to Output format:

Mapping.csv file has been loaded into a data frame. These above extracted genres of the test data have been mapped and to respective index value using mapping data frame and the predicted are changed to a binary string.

Save and Submit:

This final predicted data frame of test data is saved into a csv format. This saved result is then submitted to Kaggle.

F1 score of part3:

F1 score obtained to part3 was **1** **Part-4**

(Bonus part):

Models which are saved earlier are loaded in this part. This enables users to get the saved final predicted csv file by giving path of test data.

Libraries loaded:

All of the above-mentioned libraries are loaded for this part.

Implementation:

Spark session has been initialized and above necessary libraries were loaded.

Functions defined:

Three functions are defined which takes training, testing and saved models as input and returns a data frame with predicted labels. Inside these functions, the test data is pre-processing according to respective part and feature engineering. Then loaded model has been fit on the modified test data.

Mapping functions:

These obtained predicted data frame is then passed to map and extract function where mapping.csv file has been loaded into a data frame. These above extracted genres of the test data have been mapped and to respective index value using mapping data frame and the predicted are changed to a binary string.

Data Loading:

Training data has been loaded using pandas and then changed to pyspark data frame (as it is not splitting and reading the training data correctly when tried loading using pyspark). Similarly testing data has been loaded using pandas then changed to pyspark data frame.

User have to give the path of the test data in order to run all models and csv file with predictions are stored in the specified locations.

Loading the Models:

The Trained random forest models from each part are loaded and passed into the above functions along with training and testing data to get the desired results.

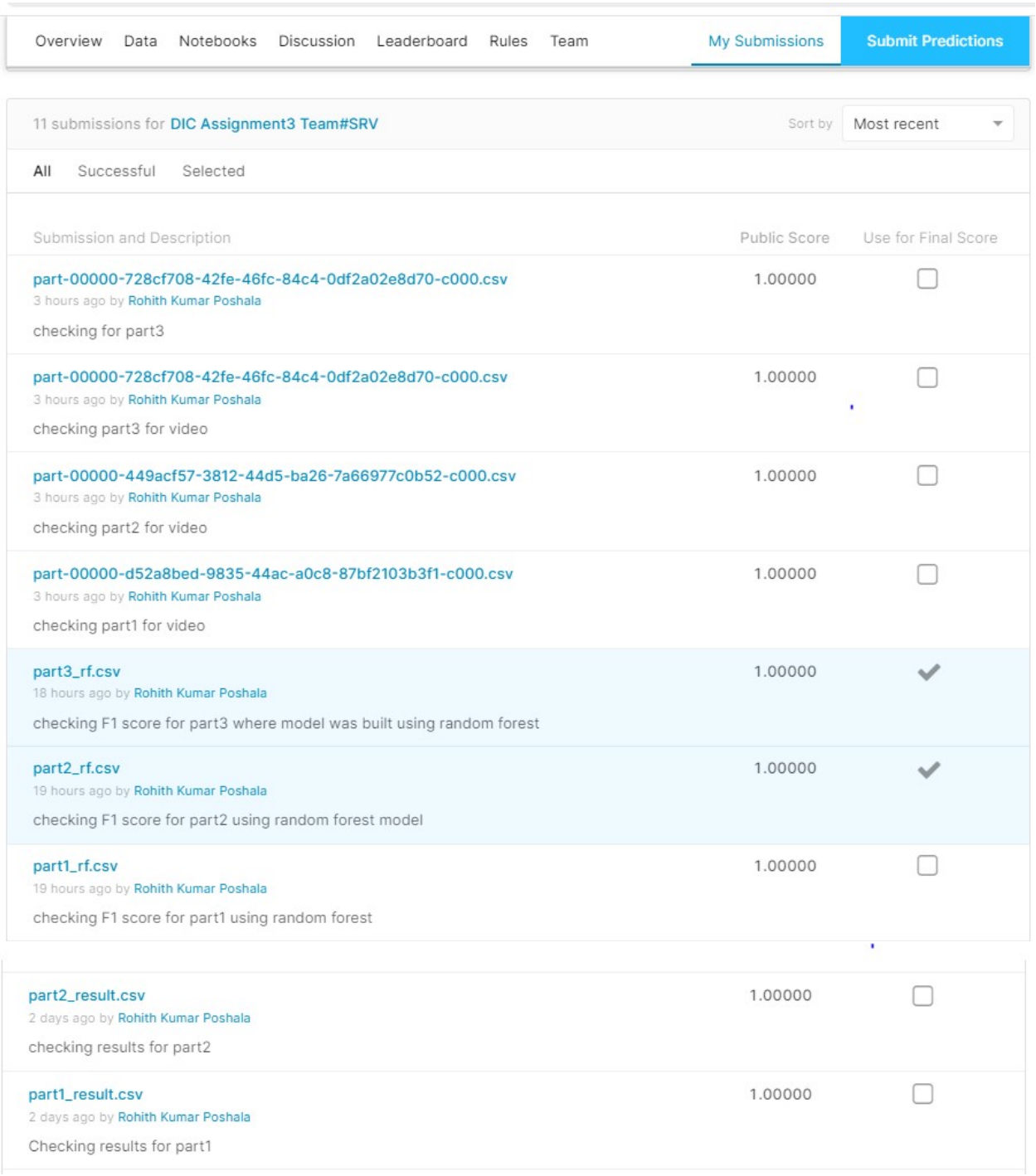
Save and Submit:

This final predicted data frame of test data is saved into a csv format. This saved result is then submitted to Kaggle.

F1 score:

F1 score of 1 has been obtained for all the 9 submissions done by the team.

F1 scores obtained for all three parts are observed to be **1.**



Video link: (explaining our implementation)

https://www.youtube.com/watch?v=46BnQxF2H4Q&feature=youtu.be **References**:

https://spark.apache.org/docs/latest/ml-features https://towardsdatascience.com/multi-class-text-classification-with-pyspark-7d78d022ed35 https://spark.apache.org/docs/latest/mllib-feature-extraction.html