Predictive Analytics with Spark Report

By: Krishna Menon, Thankam Abish, Lucas Beitzel

NOTE: we used NumPy just for storing our predictions and for storing our models. We did this purely for the ease of access, no computations were done with NumPy throughout the project.
NOTE: we used Java version 8 instead of the default Java version.

Part1: Basic Model

The first thing we do after loading in the train and test CSV files as a PySpark dataframes is clean the data of invalid rows. We did this by checking the validity of every row, and recording those outputs in a new column called “validity”. We then use this to filter out and drop every row with a validity of “False”, so we are left with only valid rows.
Next we clean our plot column data by removing all non-alphabetical characters, and replacing all occurrences of multiple spaces with a single space. Then, we split each plot on the space character so all of our plots are converted into lists. Finally, we remove all stop words in each of our plots.
Now that our plot data is cleaned, we now pass that column into a CountVectorizer to convert our collection of plots into a collection of token lists and their counts. This also gets our plot output data into a format that can be used later on in our Logistic Regression.
Once that’s done, we can start parsing the “genre” column. We did this by creating another UDF(user-defined function), which created an array the size of every single genre, which had a 1 if that genre was attributed to that row’s particular movie, and a 0 otherwise. We then took this array and created 20 new columns, one for each genre, with the array’s information split across appropriately.
Now that all of our data is set up correctly, we finally pass it into our Logistic Regression models. We create a separate model for each genre, for a total of 20 models. For each row, we get the prediction for each of the genre models, and append each of the predictions in a new column called “predictions”, just like the format is outlined in the sample.csv file.

F1 Accuracy Score: 0.97364

Part2: TF-IDF Improvements

For this part, we used a lot of the same procedure as we did in Part 1. We first loaded the train and test.csv files into a PySpark dataframe. Then the dataframe was procedurally cleaned by first removing any invalid rows. Afterward, we cleaned and reformatted the text in the plot column, and removed all of the stop words.

Now just like the last part we pass our plot data into a CountVectorizer and get our raw feature data, but then we pass that raw feature data into an IDF model to get our end result features. This completes the TF-IDF part.
Now just like in Part 1, we parse the genre column for a total of 20 additional columns (one per each of the genres), and we indicated if that row contained a specific genre with a 1 in that respective column, and a 0 otherwise.
Finally, just like the last part, we pass all of our predictions into our 20 separate Logistic Regression models. Each model gives its own prediction for each of the rows, and all of these predictions are appended to each other in the “predictions” column, with the same format as the sample.csv file.

F1 Accuracy Score: 0.97364

Part3: Custom Feature Engineering w/ Word2Vec

For this part, we started out with the same cleaning procedures as the last two parts. We loaded the CSV files into spark dataframes, cleaned them by first removing any invalid rows, then reformatted the text in the plot column, and removed all of the stop words.
Now we pass our plot data into a Word2Vec model, and get our features.
Now just like the last parts, we parse the genre column for a total of 20 additional columns (one per each of the genres), and we indicated if that row contained a specific genre with a 1 in that respective column, and a 0 otherwise.
Finally, just like the last parts, we pass all of our predictions into our 20 separate Logistic Regression models. Each model gives its own prediction for each of the rows, and all of these predictions are appended to each other in the “predictions” column, with the same format as the sample.csv file.

F1 Accuracy Score: 0.94134