مدينة زويل للعلوم والتكنولوچيا
Zewail City of Science and Technology

Final Project - IMDB Analysis

CIE 427

Prepared For
Dr. Elsayed Hemayed

Prepared By

Omar Gaballah	201801697
Ahmed AbdelSalam	201801597
Mazen Hassan	201801897
Ibrahim Hamada	201800739

Table of content :

Objective

In this project, we are analyzing the IMDB reviews dataset. Moreover, we will be applying some Machine Learning models and sentiment analysis on reviews to predict some of the labels of the datasets such as: rating, and spoiler alert.

Movie recommender systems help people in narrowing their ranges of which movie to watch. However, people still look for other people's reviews to determine whether they would like to watch this movie or not. We believe that ranking people's reviews can be of great help for websites and viewers.

Data Collection

The dataset is obtained from IMDb Review Dataset - ebD | Kaggle. It is collected from IMDB with total records of 5, 571, 499 reviews for 453, 528 total shows. The reviews are written by 1, 699, 310 users. The data is split into 6 json files each of which has the following elements:

Content	Details
review_id	It is generated by IMBb and unique to each review
reviewer	Public identity or username of the reviewer
movie	It represents the name of the show (can be - movie, tv-series, etc.)
rating	Rating of movie out of 10, can be None for older reviews
review_summary	Plain summary of the review
review_date	Date of the posted review
spoiler_tag	If 1 = spoiler & 0 = not spoiler
review_detail	Details of the review
helpful	List of people find the review helpful.

Table1. Dataset.

We pulled data from Kaggle using Kaggle API, but AWS did not read the Kaggle CLI despite being put in the specified directory by Kaggle documentation. We overcome this issue by using Kaggle API inside the python environment.

On pulling the data, we made sure that we were on the EC2-user side and then pushed the data to the HDFS. In addition, we kept buck-up data in a separate S3. So that whenever the cluster terminates or fails, we will not have to bring it all again from Kaggle.

But right before pushing to the HDFS, we kept slicing the data so that when the data gets loaded on an RDD it won't fail. Doing slicing, we focused on maximizing the size of data that can be held in RDD, we finally agreed on using 100 mega per RDD as a way to parallelize the data while still keeping each RDD as an acceptable amount of data.

Data Analysis

We have defined multiple requirements to do on the data and since the data contains tv shows and movies, each requirement we made was being done on the tv shows alone and on the movies alone. The first step in most requirements is to filter out the nans present in the rating section when it was being used. Since it is one of the important sections it was being used in nearly all the requirements. To be able to do the requirements better, we defined a set of helpful functions:

1) **extract_year_Movie():**

The purpose of this function was to extract the year in the title of the movie using regex expressions and if it fails to extract the years in cases like the title provided is a title for a tv show, it would return None.

2) **extract_text_Movie:**

The purpose of this function was to extract the text in the title of the movie using regex expressions and if it fails to extract the text in cases like the title provided is a title for a tv show, it would return None. So in summary, it used to identify the title of a movie.

3) extract_years_Tv

The purpose of this function was to extract the year in the title of a Tv Show using regex expressions and if it fails to extract the years in cases like the title provided is a title for a movie , it would return None. So in summary, it used to identify the year of a Tv Show.

4) extract_text_Tv

The purpose of this function was to extract the text in the title of the Tv Show using regex expressions and if it fails to extract the text in cases like the title provided is a title for a movie, it would return None. So in summary, it used to identify the title of a Tv Show

```
## Helper Functions
def extract_year_Movie(string):
    match = re.search(r'\((\d{4})\)', string)
    if match:
        return match.group(1)
    else:
        return None


def extract_text_Movie(string):
    match = re.search(r'(.*)\s\(\d{4}\)', string)
    if match:
        return match.group(1)
    else:
        return None

def extract_years_Tv(string):
    match = re.search(r'\((\d{4})-\s?(\d{4})?\)', string)
    if match:
        start_year = match.group(1)
        end_year = match.group(2)
        if end_year:
            return "({}-{})".format(start_year, end_year)
        else:
            return "({}-)".format(start_year)
    else:
        return None
```

Figure 1. Helper Function.

After that, we started dealing with data to find the requirements as follows:

1) **Requirement 1:** (Lowest Rated Shows and Highest Rated Shows)
    a) Lowest Rated Shows :
        I. We firstly created a tuple which contains the rating and the title of show
        II. We then found the average rating of every title
        III. After, we use the extract_text_movie so we can have only the movies and extract_text_Tv another time so we can only have the tv shows.
        IV. Then, We filter the ratings lower than 3 to remove the outliers in the data
        V. Finally, we sort the rdd from the lowest to highest.

    b) Highest Rated Shows:
        I. We firstly created a tuple which contains the rating and the title of show
        II. We then found the average rating of every title
        III. After, we use the extract_text_movie so we can have only the movies and extract_text_Tv another time so we can only have the tv shows.
        IV. Then, We filter the ratings lower than 3 to remove the outliers in the data
        V. Finally, we sort the rdd from the highest to lowest.

2) **Requirement 2:** (Number of Spoiled Reviews)
        I. We firstly created a tuple which contains the spoiler tags and 1.
        II. We then reduce by addition.
        III. Finally, we have two tuples in a list, each containing the tag and the number of reviews.

3) **Requirement 3:** (Number of Helpful Reviews for shows)
   I. We firstly created a tuple which contains the show title and 1 if it is helpful.
   II. We then filter the non helpful reviews and After, we use the extract_text_movie so we can have only the movies and extract_text_Tv another time so we can only have the tv shows..
   III. Finally, we sort the two rdds and we have each one contains the helpful reviews, one for movies and one for tv shows

4) **Requirement 4:** (Number of reviews for shows)
   i) We firstly created a tuple which contains the show title and 1.
   ii) Then, We reduce by key using add
   iii) After, we use the extract_text_movie so we can have only the movies and extract_text_Tv another time so we can only have the tv shows.
   iv) Finally, we sort the two rdds and we have each one contains the number of reviews, one for movies and one for tv shows

5) **Requirement 5:** (The Highest Rated Show in each year)
   i) We firstly created a tuple which contains the show title and rating.
   ii) Then, We reduce by key using add
   iii) After, we use the extract_text_movie and extract_text_Tv to get the title of the movies and the tv shows.
   iv) We also filtered every rating that is under 3 and above 9.5.
   v) Then, we created a dictionary from the rdd which its key is the year and the value is the tv show or the movies which have the highest ratings and the ratings.

6) **Requirement 6:** (Years and number of movies and tv shows released)
   i) We firstly created a tuple which contains the title and 1.
   ii) Then we used the extract_year_movie on the title to element the tv shows
   iii) Then, We reduce by key using add.
   iv) So now, we have a rdd which contains the years and the number of movies released in it.
   v) Finally, we did the same but using extract_year_tv to element the movies.

Machine Learning Models (SparkML)

In this part, we used SparkML to apply some ML model on the dataset in order to predict some important information. We firstly chose the columns of interest to be predicted after training the model on the given dataset. As shown above, the dataset has 2 interesting features: Rating, and Spoiler_tag. We decided to set the previously mentioned columns as the output of the model. We used the Logistic Regression Model, the Naive Bayes Model, and the Support Vector Classifier (SVC) to predict the spoiler_tag given a particular input review. Additionally, we used Linear Regression Model and Decision Tree Regressor Model to predict the Rating of a given review.

Accordingly, we tried to perform sentiment analysis by using a set of labeled data on movie/series reviews. We used that dataset to try and train a best machine learning model. The steps and processes are discussed in detail below.

1) Data Extraction: We downloaded the 6 files of the dataset from Kaggle into our Google Drive. Then, we split the files into smaller files to overcome Connection Refused [Error 111].

```
import json
k = 0
for file_ in file_list:
  # Open the file to be splitted
  with open((file_), 'r') as f1:
    ll = json.load(f1)
    #total length size of the json file
    print(len(ll))
    #in here 50000 means we getting splits of 50000 reviews
    size_of_the_split=50000
    total = len(ll) // size_of_the_split
    # Number of splits
    print(total+1)
    for i in range(total+1):
      k = k + 1
      json.dump(ll[i * size_of_the_split:(i + 1) * size_of_the_split],
                open("/content/drive/MyDrive/BigData_FinalProject/Splits/part"+ str(k)+".json", 'w'))
      print(k)
```

Figure 2.Data Extraction.

2) Data Preprocessing: After reading the data, we firstly checked the data types and set the rating column to be float instead of string. Then, we found the number of null values and they are contained in the rating columns. Therefore, we imputed the null values using the mean rating of the movie. The dataset schema is as follows:

```
root
 |-- helpful: array (nullable = true)
 |     |-- element: string (containsNull = true)
 |-- movie: string (nullable = true)
 |-- rating: float (nullable = true)
 |-- review_date: string (nullable = true)
 |-- review_detail: string (nullable = true)
 |-- review_id: string (nullable = true)
 |-- review_summary: string (nullable = true)
 |-- reviewer: string (nullable = true)
 |-- spoiler_tag: float (nullable = true)
```

Figure 3. Data Preprocessing.

After selecting the columns of interest, we tokenized our data so as to separate each word of the reviews as individual values. We used the standard stop-word removing to remove and clean our words. We then used hashing-TF on our tokenized data to transform the words into vector values so we can work with the data. Therefore, the pipeline is defined to be:
[label_indexer, Tokenizer, stopwords remove, hashingTF, ML Model]

helpful	movie	review_date	review_detail	review_id	review_summary	reviewer	spoiler_tag	rating-Out	words	Wordsfiltered	hashedValues
[1, 1]	After Life (2019-)	3 May 2020	I enjoyed the fir...	rw5704482	Very Strong Season 2	raeldor-96879	0.0	9.0	[i, enjoyed, the,...	[enjoyed, first, ...	(262144,[2437,125...
[2, 2]	The Valhalla Murd...	3 May 2020	I know Iceland is...	rw5704483	Icelandic detecti...	dosleeb	0.0	6.0	[i, know, iceland...	[know, iceland, s...	(262144,[16794,21...
[0, 0]	Special OPS (2020-)	3 May 2020	Except K K , no o...	rw5704484	Nothing special	brightconscious	0.0	7.0	[except, k, k, ,,...	[except, k, k, ,,...	(262144,[29550,68...
[5, 9]	#BlackAF (2020-)	3 May 2020	I'm guessing that...	rw5704485	Good but	gasconyway	0.0	8.0	[i'm, guessing, t...	[guessing, 62, ye...	(262144,[5675,853...
[26, 41]	The Droving (2020)	3 May 2020	Here's the truth....	rw5704487	An honest review	mmason-15867	0.0	2.0	[here's, the, tru...	[truth., much, mo...	(262144,[991,1889...
[0, 1]	All About Eve (1950)	3 May 2020	Having seen this ...	rw5704488	Amazing	schroederagustavo	0.0	10.0	[having, seen, th...	[seen, film, firs...	(262144,[645,3889...
[0, 1]	Runaway Train (1985)	3 May 2020	The movie had som...	rw5704489	Impressive action...	welhof1	0.0	7.0	[the, movie, had,...	[movie, impressiv...	(262144,[5561,833...
[7, 9]	Iron Fist (2017–2...	3 May 2020	I loved it from t...	rw5704490	Another great Net...	Evastar	0.0	9.0	[i, loved, it, fr...	[loved, first, ep...	(262144,[56844,62...
[16, 26]	The Half of It (I...	3 May 2020	I see that Netfli...	rw5704491	Needed the other ...	tioeta	0.0	4.0	[i, see, that, ne...	[see, netflix, te...	(262144,[1904,243...
[1, 5]	This Is Us (2016-)	3 May 2020	This is the show ...	rw5704492	All the Pearsons ...	stephenrifkin	0.0	2.0	[this, is, the, s...	[show, people, no...	(262144,[13644,29...
[2, 2]	Closure (I) (2018)	3 May 2020	This is a fun and...	rw5704494	Fun and intriguing	andrewtschroeder	0.0	9.0	[this, is, a, fun...	[fun, intriguing,...	(262144,[5370,952...
[3, 4]	Unstoppable (2010)	3 May 2020	A suspenseful thr...	rw5704493	Excellent last fi...	UniqueParticle	0.0	8.0	[a, suspenseful, ...	[suspenseful, thr...	(262144,[4235,125...
[2, 3]	Dangerous Lies (2...	3 May 2020	Highlight was Cam...	rw5704496	Not bad	Hellooo1234321	0.0	6.745999	[highlight, was, ...	[highlight, camil...	(262144,[2257,934...
[8, 20]	Beastie Boys Stor...	3 May 2020	A lot of excuses ...	rw5704497	The apology tour	flippereight	0.0	3.0	[a, lot, of, excu...	[lot, excuses, ap...	(262144,[19352,20...
[0, 1]	Ruben Brandt, Col...	3 May 2020	A fenomel animati...	rw5704500	Magnificent art-a...	ovandoreyna	0.0	10.0	[a, fenomel, anim...	[fenomel, animati...	(262144,[47435,50...
[1, 2]	Some Kind of Hate...	3 May 2020	Some Kind Of Hate...	rw5704499	Vengeful Spirit	Pairic	0.0	7.0	[some, kind, of, ...	[kind, hate:, lin...	(262144,[956,4757...
[0, 2]	Cube Zero (2004)	3 May 2020	I actually liked ...	rw5704501	NOT FOR KIDS!	driftingintime	0.0	10.0	[i, actually, lik...	[actually, liked,...	(262144,[3158,270...
[4, 4]	Carne (1991)	3 May 2020	Well, I just fini...	rw5704502	Like watching Noe...	Stay_away_from_th...	0.0	8.0	[well,, i, just, ...	[well,, finished,...	(262144,[2432,927...
[0, 1]	500 Days of Summe...	3 May 2020	Ah Indies done by...	rw5704504	Cutie indie, just...	vostf	0.0	5.0	[ah, indies, done...	[ah, indies, done...	(262144,[6561,978...
[2, 3]	8½ (1963)	3 May 2020	Everybody should ...	rw5704503	Maybe the biggest...	sharonrota	0.0	10.0	[everybody, shoul...	[everybody, watch...	(262144,[6512,100...

Figure 4. Data Preprocessing output.

We used a random split to split our data into train and test. We split our data in 70-30 split so we have enough data to train and test the models.

3) Model Training: After processing the data and splitting the train and test, we started with our machine learning models. We decided to test three models for our data:
   A) Logistic Regression
   B) Naïve- Bayes
   C) Support Vector Classifier

   In order to predict the spoiler flag and determine whether a given review spoils the movie/series or not.
   Additionally, we decided to use Linear Regression and Decision Tree Regressor to predict the rating of a given review.

4) Model Performances:
   A) **Logistic Regression:** As logistic regression is one of the widely used classification models and has high accuracy and low runtime, we decided to first go with it.
   The Logistic Regression model achieved an accuracy of **0.792** and F1 score of **0.782**.

```
Logistic Regression F1 Score:  0.78184782015050 26
Logistic Regression Accuracy:  0.79199325710675 56
```

Figure 5. Logistic Report.

   B) **Naïve Bayes:** we went ahead with the most preferred model for NLP, we got an accuracy and F1 score of **0.782** and **0.784** respectively. This model seems to perform slightly higher than logistic regression model.

```
Naive Bayes F1 Score:  0.7840851494732808
Naive Bayes Accuracy:  0.7821342851322747
```

Figure 6. Naive Bayes Report.

C) **Support Vector Machine:** For our final model that we wanted to test we went with SVC since it can work with multidimensional data and binary classification like our dataset. We had the best accuracy and F1 score with this model of 0.801 and 0.785 respectively.

```
SVM F1 Score:  0.7851842590516469
SVM Accuracy:  0.8011909295686497
```

Figure 7. Logistic Report.

D) Cross Validation: To reduce the chances of overfitting and to get the best model with considerable accuracy, we decided to go ahead with K-Fold cross validation. We decided to do cross validation on the top two models with the best accuracy score. Logistic Regression and Support Vector Classifier.

Results of the classification models are summarized in the following table:

Model	Logistic Regression	Naïve Bayes	Support Vector Machine
Accuracy	0.7919	0.7840	0.8011
F1-score	0.7181	0.7821	0.7851

Table 2. Results of the models.

E) Linear Regression: This model was used to predict the value (1-10) of the rating given a specific review. After training the model, we obtained an R-squared value of nearly 0, which indicates no linear relationship between the review and ratings. So, we tried other models.

F) Decision Tree Regressor: This model somehow was able to drive some of the relationship between the review and ratings. It satisfies an RMSE of 0.21, which is slightly high, but still better than Linear Regression.

G) GBT Regressor: It is a popular method for solving prediction problems in regression domains. The approach improves the learning process by simplifying the objective and reducing the number of iterations to get to a sufficiently optimal solution. After training the model, we achieved an RMSE of 0.17, which made it the model with the best performance.

AWS Learner Lab

There were some problems related to the data such as:

1. The learner lab was timed limited as the session only lasted for 4 hours. The cluster would terminate once the session is over so if we were doing any work on the cluster, we would need to restart again.
2. We faced many difficulties uploading the data but as mentioned before we used S3s to remedy the situation.
3. We had faced a terrible mistake, which is that we thought the default fetching of the data is done on the ec2 user side. But we found it was done on a hadoop user, such a mistake resulted in crashing the cluster multiple times and trying to fetch the data from the S3 multiple times.
4. To be able to read the data in spark, we had to split the files after downloading it from kaggle which took many times.
5. We faced an error called “Service currently not reachable, please retry emr” which took a long time to solve.

The overall experience of AWS had also bright sides, for instance we managed to access each others’ clusters using cloud9 and consequently managed to have easier job coordination.

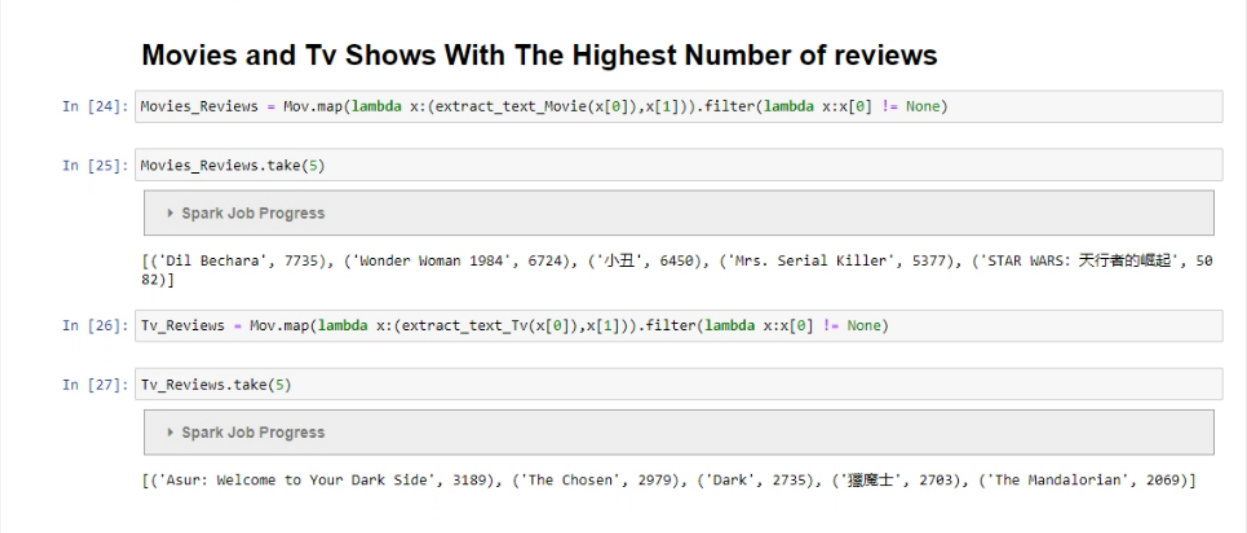
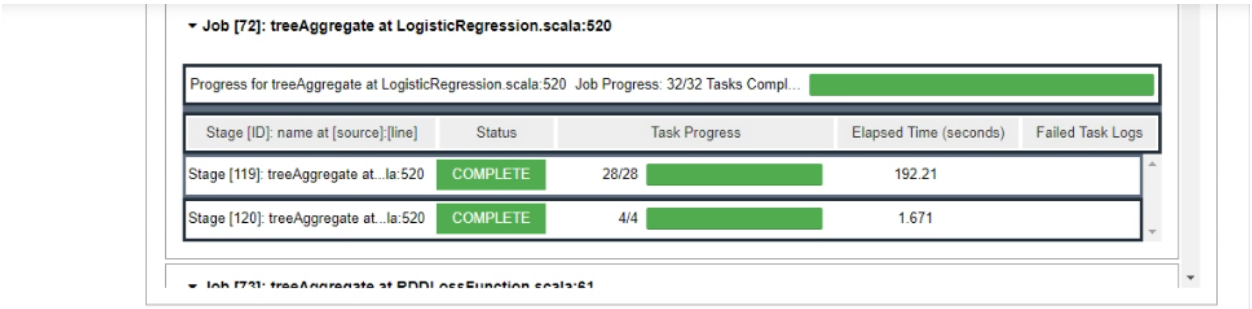


Figure 8. Example for a requirement running in AWS.




```
+------------------------------+------------------------------+------------------------------+------------------------------+----
-+----------+
|                review_detail|                Wordsfiltered|                      features|                   probability|labe
l|prediction|
+------------------------------+------------------------------+------------------------------+------------------------------+----
-+----------+
|Hay que felicitar a todos l...|[hay, que, felicitar, a, to...|(262144,[2243,4220,8744,887...|[1.0,1.0147523803513819E-16]|  0.
0|       0.0|
|#FinalVerdictCourage is not...|[#finalverdictcourage, is, ...|(262144,[619,633,711,1711,1...| [1.0,9.431133197591836E-17]|  0.
0|       0.0|
|Jon Stevenson's directional...|[jon, stevenson's, directio...|(262144,[14,6446,6943,7182,...| [1.0,8.729460791393196E-17]|  0.
0|       0.0|
|Claro que más que inquietar...|[claro, que, más, que, inqu...|(262144,[2733,3636,4631,673...| [1.0,8.120925660206758E-17]|  0.
0|       0.0|
|Im Rahmen der "Operation Mo...|[im, rahmen, der, "operatio...|(262144,[1369,2196,3150,323...| [1.0,7.955611589648392E-17]|  0.
0|       0.0|
|Fue estrenado en la Argenti...|[fue, estrenado, en, la, ar...|(262144,[492,929,3674,3965,...| [1.0,7.859240727789956E-17]|  0.
0|       0.0|
|Dieser französische Thrille...|[dieser, französische, thri...|(262144,[4,1727,2196,3151,3...| [1.0,7.076305014814043E-17]|  0.
0|       0.0|
|Tras del armisticio del 8 d...|[tras, del, armisticio, del...|(262144,[1053,4695,5135,634...| [1.0,6.603690322083478E-17]|  0.
0|       0.0|
|A sensational M G M Picture...|[a, sensational, m, g, m, p...|(262144,[919,991,1536,3533,...| [1.0,6.281163898169253E-17]|  0.
0|       0.0|
|This film is outstanding in...|[this, film, is, outstandin...|(262144,[1466,1889,2977,302...| [1.0,5.928934389472381E-17]|  0.
0|       0.0|
+------------------------------+------------------------------+------------------------------+------------------------------+----
-+----------+
only showing top 10 rows

Logistic Regression F1 Score:  0.7792807474829343
Logistic Regression Accuracy:  0.7867455501370735
```

Figure 9. Example for a Machine learning model running in AWS.

Data Problems and Challenges

There were some problems related to the data such as:

1) We couldn't read the dataset until we splitted the files into smaller files (less than 100 MB).
2) After reading the dataset, there were some uncorrect data types. We changed the data types of these columns to the correct types.
3) There were around 450,000 null values in the rating column of the dataset, so we imputed the null values using the mean of rating of that movie.
4) Some movies had only few reviews, so we considered all the movies that had less than 4 reviews as outliers.
5) From our knowledge on IMDB, we know that no movie has a rating higher than 9.2, so we did not consider any movies that had higher ratings than 9.2.