AIM-3 Scalable Data Science – SS 2021 - Homework II

July 2, 2021

CANEVET Gaspard
g.canevet@campus.tu-berlin.de

Contents

1 Introduction

In the coming report, you will find some results regarding the second homework of AIM-3: Scalable Data Science: Systems and Methods course from TU Berlin. In the first part we used Apache Spark's ML library to build an email spam filter. Then, in the second part we used Apache Spark's ML library to cluster the MNIST dataset of handwritten digits. Finally, In the third part we used Apache Spark's ML library to build a movie recommendation system.

2 Classification

Below, we can find the architecture of the file A. It is made of

- All datasets: spam and nospam datasets for the training part and spam and nospam datasets for the testing part.
- One src folder with every source codes. The main code is in the python file "spam_filter.py". The PunctuationRemover is a user defined transformer for the Machine Learning pipeline that remove punctuation.
- A README.md file that explains how to run the code.

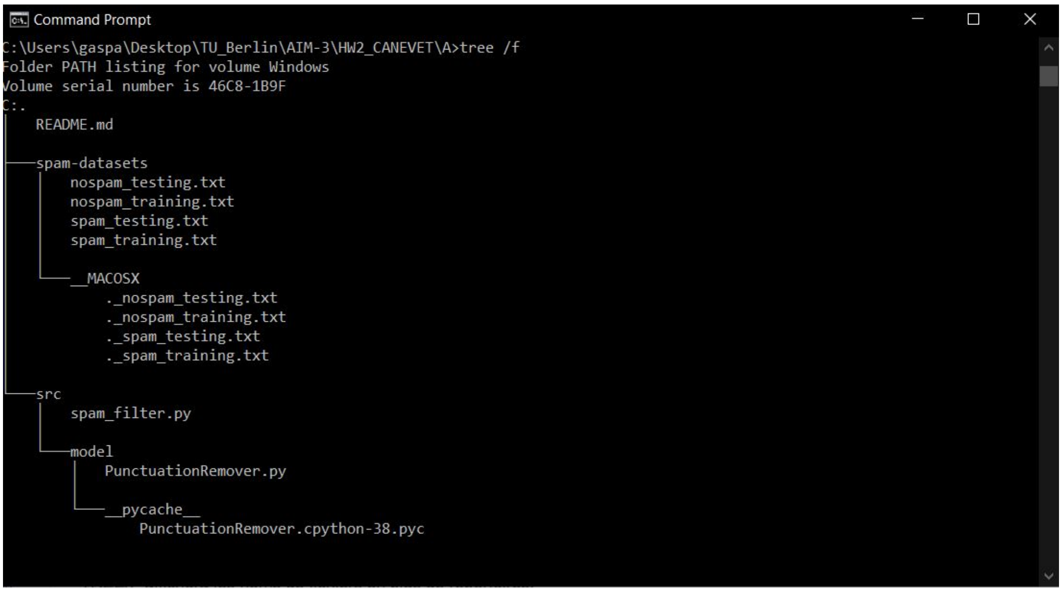
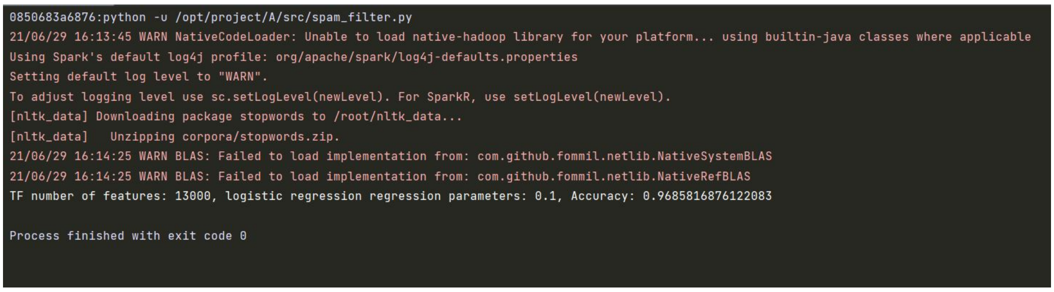


Figure 1: Folder A architecture

For this task we use a pipeline to first transform the input data and then perform a machine learning algorithm, a logistic regression in that case. On top of that we use an optimiser method (TrainValidationSplit function) provided by Apache Spark's ML library in order to define the best number of features for the HashingTF functions and the best regression parameter for the logistic regression regarding a binary classification evaluator. Find below a simple graph that summarise the pipeline.

Figure 2: Graph of the pipeline use in the spam filter algorithm

The input is a dataset of spam and non spam emails. Each row is a sentence (String type) and a label (String Type: spam or nospam). The first part of the pipeline consist of removing every punctuation in sentences. We use String python library that provides a list of punctuation symbols. The second part of the pipline tokenize the sentence. In other words, it splits every sentences into list of words. The third part removes stop words like "a", "all". We use nltk python library that provides a list of english stop words. Then, in the forth and fifth parts we apply TF-IDF method to highlight the importance of specific words in the overall email corpus. After this method we have a features column made of a tuple (number of features in hashingTF, [wordsIndex],[TF-IDF words value]). The sixth stage of the pipeline is a string indexer that transform email category spam or nospam, into integer, 0 or 1. The last stage of the pipeline is a logistic regression method. After creating the pipeline we perform a TrainValidationSplit method that will fit the pipeline with the training set and different parameters: number of features for the hashingTF methods (np.arange(1000,20000,4000), regression parameters value ([0.1,0.05,0.01]). Finally, we print the best model parameters and transform the testing set in order to compute the accuracy. Find below a picture of an output of the algorithm.

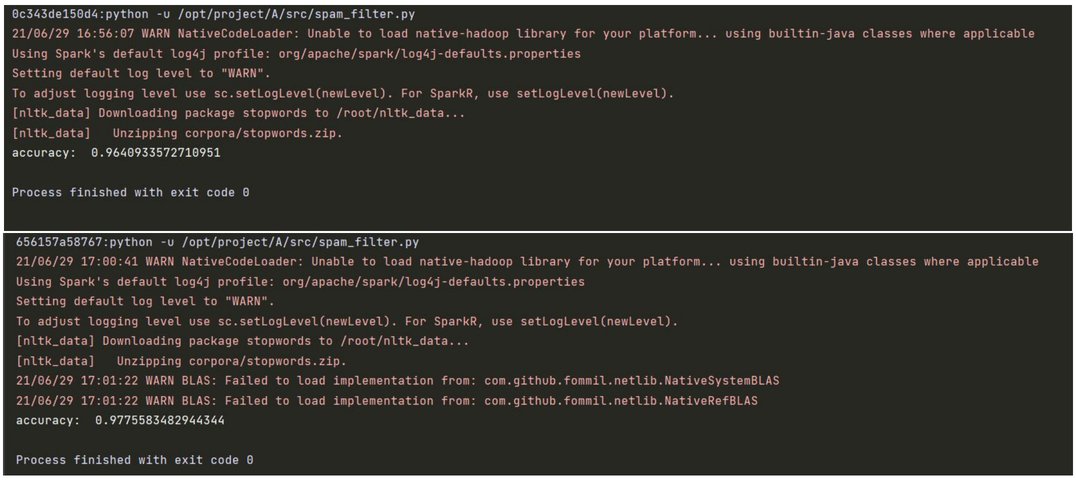


```
0850683a6876:python -u /opt/project/A/src/spam_filter.py
21/06/29 16:13:45 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
21/06/29 16:14:25 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
21/06/29 16:14:25 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
TF number of features: 13000, logistic regression regression parameters: 0.1, Accuracy: 0.9685816876122083

Process finished with exit code 0
```

Figure 3: Output of the spam filter algorithm

Here we can see that the best number of features is 13000 and the best regression parameter is 0.1. After transforming the test set we compute the accuracy and here our model has an accuracy of 0.97.
Regarding the model parameters, I ran the algorithm many times and most of the time a large number of features is compulsory. It seems quite logical the more features, the more words are used and thus the more precise our algorithm is. Nevertheless, we have to prevent too complicated model. This is why a more deep study on number of features could be really useful.
Regarding the accuracy, we have here a really good machine Learning model. Lets run other models in order to compare their accuracy.



```
0c343de150d4:python -u /opt/project/A/src/spam_filter.py
21/06/29 16:56:07 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
accuracy:  0.9640933572710951

Process finished with exit code 0
```

```
656157a58767:python -u /opt/project/A/src/spam_filter.py
21/06/29 17:00:41 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
21/06/29 17:01:22 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
21/06/29 17:01:22 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
accuracy:  0.9775583482944344

Process finished with exit code 0
```

Figure 4: accuracy for Naive Bayes method (top), Linear SVC (bottom)

Here we can see that the Linear SVC is the best one with an accuracy of 0.98. To conclude, we can say that all three methods provide good results. Over 0.95 accuracy in a machine learning model is really good. Nevertheless, we could also think about other ways of improvement. Indeed, our training dataset is skewed. It is made of 10% of spam emails and 90% of non spam emails. It could be really valuable to increase the number of spam emails. We could seek for other spam emails or we could implement data replication to curb data imbalanced impact.

3 Clustering

Below, we can find the architecture of the file B. It is made of:

- Mnist datasets: 28x28 pictures of hand-written digits, each pixel is between 0 and 255.
- One src folder with every source codes. The main code is in the python file "mnist_clustering.py". The "data_exploration.py" file is a python code to explore data. The "centroid_display_saving.py" file is a python code that displays centroids and save them into .csv file with .npy file as input.
- bisec and kmeans are folders that contain centroid images respectively for bisectiveKmeans and Kmeans method.
- A README.md file that explains how to run the code.

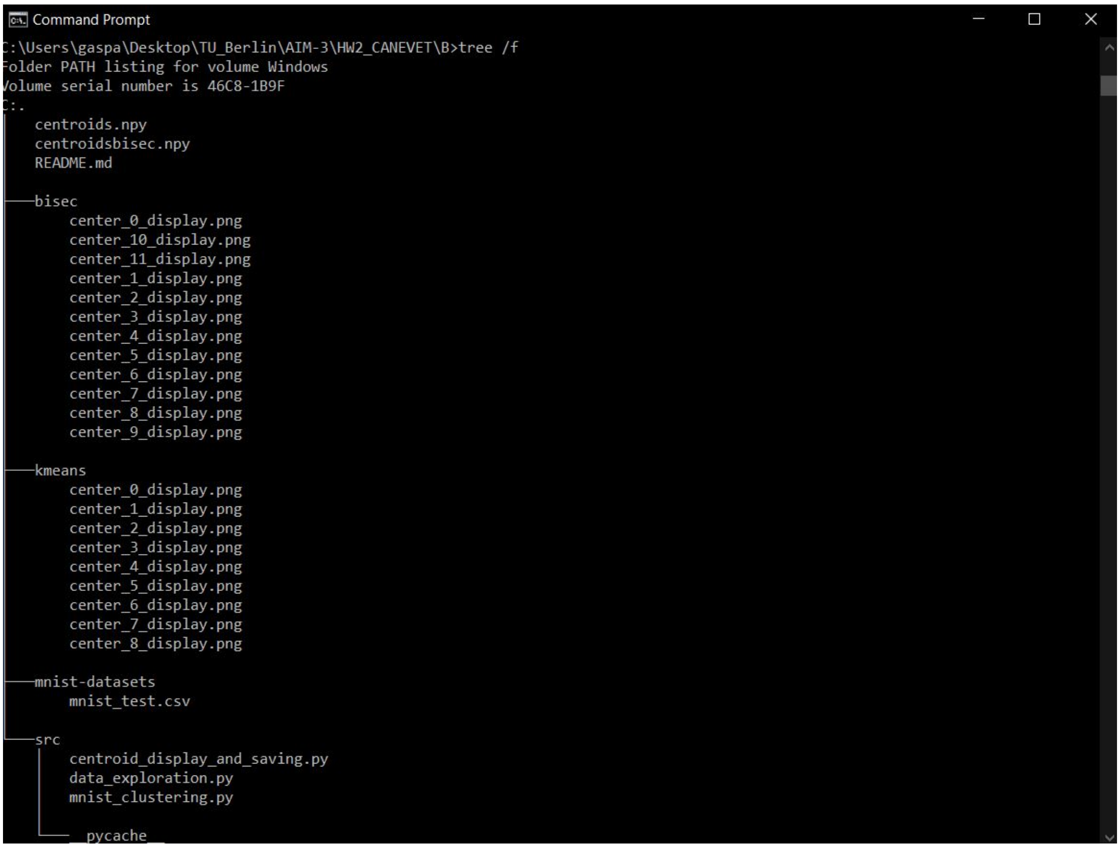


Figure 5: Folder B architecture

For this task we use a pipeline to first transform the input data and then perform a machine learning algorithm, two clustering methods in that case. The first clustering method is a classical k-mean algorithm. The second clustering algorithm is a sort of hierarchical clustering method named bisecting k-means. On top of that we use an optimiser method (TrainValidationSplit function) provided by Apache Spark's ML library in order to define the best number of clusters k regarding a clustering classification evaluator. Find below a simple graph that summarise the pipeline.

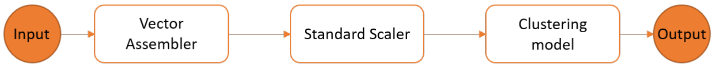
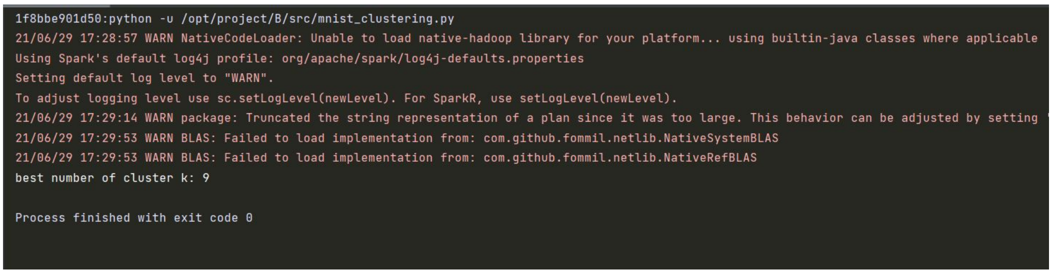


Figure 6: Graph of the pipeline use in the clustering algorithm

The input is a dataset made of 10 000 28x28 hand written digits pictures. each column of our dataset is a pixel from our picture. We first have to transform this columns into a column named "features" composed of a vector that contains every pixels. We then center our data and finally we use a clustering method to create a machine learning model. After creating the pipeline we perform a TrainValidationSplit method that will fit the pipeline with the data set and different parameters: number of clusters for the clustering method ([7,8,9,10,11,12]). Finally, we print the best model parameters and display the cluster centers. find below the output of the K-mean algorithm.



```
1f8bbe901d50:python -u /opt/project/B/src/mnist_clustering.py
21/06/29 17:28:57 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/06/29 17:29:14 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting
21/06/29 17:29:53 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
21/06/29 17:29:53 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
best number of cluster k: 9

Process finished with exit code 0
```

Figure 7: K-mean output

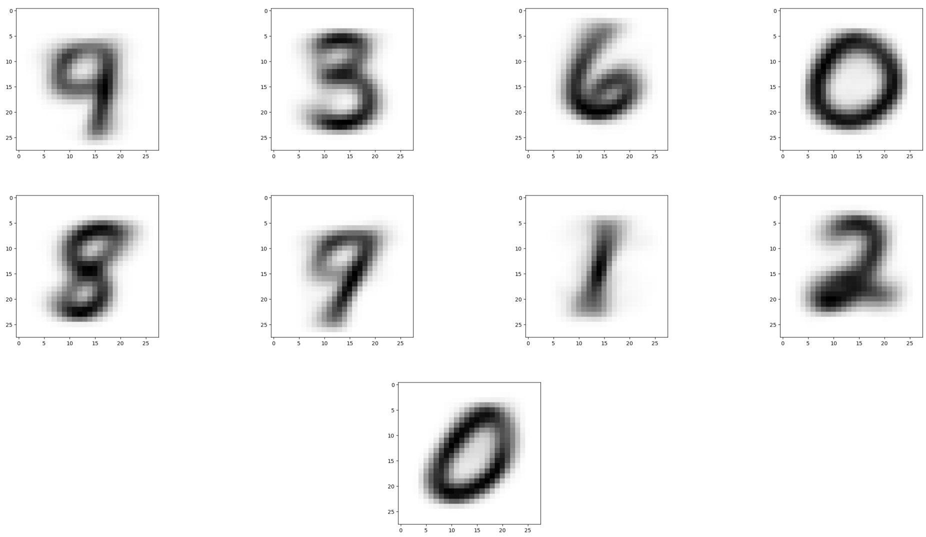


Figure 8: K-mean centroids

We can see here that the best number of clusters regarding the silhouette index and the K-mean method is nine. Our dataset is made of ten digits (0 to 9), thus, we have some digits that are gathered together in a cluster. Display clusters center is a good way to spot overlapping digits.

The figure 8 is the display of clusters center. There are two clusters for digit "0" (fourth picture of the first row and first picture of the third row). Then, we could assume that the digits "5" and "3" are gathered together (second picture of the first row) and digits "9", "4" and "7" are gathered together in two clusters (first picture of the first row and second picture of the second row). We could definitely improve our cluster method by performing some prepossessing transformations on training data. For instance, we could apply some convolution filters. It could help the model to learn more precise things regarding the digits in order to make a distinction between "7", "9" and "4" or between "3" and "5". Nevertheless we could also apply an other clustering method. Find below the output of a bisecting K-mean algorithm (based on a hierarchical clustering approach).

```
65b2059acc52:python -u /opt/project/B/src/mnist_clustering.py
21/06/29 17:37:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/06/29 17:37:38 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting
21/06/29 17:38:07 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
21/06/29 17:38:07 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
best number of cluster k: 12

Process finished with exit code 0
```

Figure 9: bisecting K-mean output

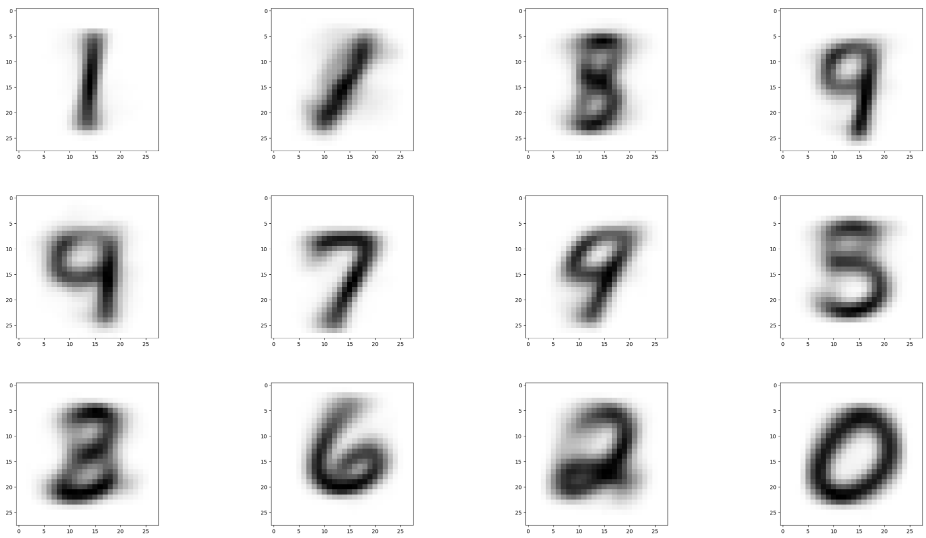


Figure 10: Bisecting K-mean centroids

We can see here that the best number of clusters regarding the silhouette index and the bisecting K-mean method is twelve. Our dataset is made of ten digits (0 to 9). It means that our model has maybe learned different way to write a same digit. For instance in the k-mean implementation, our model learned two different ways to write the digit "0". Display cluster center is a good way to spot cluster design.
Our clustering method learned two ways to write digit "1" (first and second picture of the first row) and three ways to write digits "9" and "4" (fourth picture of the first row, first and third picture of the second row). Even if it is still hard for the model to make a distinction between "4" and "9" or between "3" and "8" (third picture of the first row and first picture of the third row seems to be a mix between "8" and "3"), it has learned how to properly write digits "7" and "5". As I said before, prepossessing methods with convolution filtering could be really valuable to improve our clustering method.

4 Recommendation

Below, we can find the architecture of the file C. It is made of

- All datasets: movies.csv file made of movies information (movieId, title, genre) and ratings.csv file made of users ratings information (userId,movieId,rating,timestamp).
- One src folder with every source code. The main code is in the python file "recommendation_system.py" and the code to explore the both datasets is in "data_exploration.py" file.
- A README.md file that explains how to run the code.

```
C:\Windows\System32\cmd.exe
C:\Users\gaspa\Desktop\TU_Berlin\AIM-3\HW2_CANEVET\C>Tree /F
Folder PATH listing for volume Windows
Volume serial number is 46C8-1B9F
C:.
│   README.md
│
├───recommendation-datasets
│       movies.csv
│       ratings.csv
│
└───src
        data-exploration.py
        recommendation_system.py


C:\Users\gaspa\Desktop\TU_Berlin\AIM-3\HW2_CANEVET\C>
```

Figure 11: Folder C architecture

4.a Task i

For this task, we have to print the name of the top-10 movies with the largest number of ratings. Thus, we do a inner-join with the two datasets regarding "movieId" and group by "title" to compute ratings count and ratings average. Then we display the top ten and ordering by count. We can find below the output of the algorithm.

```
part i
+--------------------+-------------+-----------------+
|               title|count_ratings|   average_rating|
+--------------------+-------------+-----------------+
| Forrest Gump (1994)|          329|4.164133738601824|
|Shawshank Redempt...|          317|4.429022082018927|
| Pulp Fiction (1994)|          307|4.197068403908795|
|Silence of the La...|          279|4.161290322580645|
|  Matrix, The (1999)|          278|4.192446043165468|
|Star Wars: Episod...|          251|4.231075697211155|
|Jurassic Park (1993)|          238|             3.75|
|   Braveheart (1995)|          237|4.031645569620253|
|Terminator 2: Jud...|          224|3.970982142857143|
|Schindler's List ...|          220|            4.225|
+--------------------+-------------+-----------------+
```

Figure 12: Top-10 movies with the largest number of ratings

4.b Task ii

For this task, we have to print the names of the top-10 movies with the highest average for each genre. We first need to find every genre. Thus, we split the genre column for every movies (flatmap function), transform every new line into couples (movieGenre, 1) (map function) and finally reducing by key. At the end we have a RDD made of every genre and the count of each genre. To perform the task, we join the two datasets regarding "movieId", filter by genre and group by "title". Finally, we compute ratings count and ratings average. Then, we display the top ten and ordering by ratings average. We can find below the outputs of the algorithm.

Action

title	count_ratings	average_rating
Maniac Cop 2 (1990)	1	5.0
Max Manus (2008)	1	5.0
Love Exposure (Ai...	1	5.0
Priklyucheniya Ka...	1	5.0
Battle Royale 2: ...	1	5.0
Reform School Gir...	1	5.0
Supercop 2 (Proje...	1	5.0
Sonatine (Sonachi...	1	5.0
Shogun Assassin (...	1	5.0
The Big Bus (1976)	1	5.0

only showing top 10 rows

Adventure

title	count_ratings	average_rating
Palindromes (2004)	1	5.0
Crippled Avengers...	1	5.0
12 Chairs (1976)	1	5.0
Monster Squad, Th...	1	5.0
Into the Woods (1...	1	5.0
Priklyucheniya Ka...	1	5.0
Shogun Assassin (...	1	5.0
Lamerica (1994)	2	5.0
Light Years (Gand...	1	5.0
Return to Treasur...	1	5.0

only showing top 10 rows

Animation

title	count_ratings	average_rating
Light Years (Gand...	1	5.0
Return to Treasur...	1	5.0
Scooby-Doo! Abrac...	1	5.0
Alesha Popovich a...	1	5.0
Winnie the Pooh G...	1	5.0
In the blue sea, ...	1	5.0
Vacations in Pros...	1	5.0
Winnie the Pooh a...	1	5.0
Cosmic Scrat-tast...	1	5.0
Stuart Little 3: ...	1	5.0

only showing top 10 rows

Children

title	count_ratings	average_rating
Ice Age: The Grea...	1	5.0
Scooby-Doo! and t...	1	5.0
There Once Was a ...	1	5.0
Scooby-Doo! and t...	1	5.0
Happy Feet Two (2...	1	5.0
Stuart Little 3: ...	1	5.0
In the blue sea, ...	1	5.0
Cosmic Scrat-tast...	1	5.0
A Flintstones Chr...	1	5.0
Scooby-Doo! Abrac...	1	5.0

only showing top 10 rows

Comedy

title	count_ratings	average_rating
Paper Birds (Pája...	1	5.0
Radio Day (2008)	1	5.0
Tom Segura: Mostl...	1	5.0
George Carlin: Yo...	1	5.0
Chinese Puzzle (C...	1	5.0
Stuart Little 3: ...	1	5.0
English Vinglish ...	1	5.0
National Lampoon'...	1	5.0
Bill Hicks: Revel...	1	5.0
Martin Lawrence L...	1	5.0

only showing top 10 rows

Crime

title	count_ratings	average_rating
Little Murders (1...	1	5.0
Mother (Madeo) (2...	1	5.0
Sherlock Holmes a...	1	5.0
American Friend, ...	1	5.0
Ex Drummer (2007)	1	5.0
12 Angry Men (1997)	1	5.0
Decalogue, The (D...	1	5.0
Supercop 2 (Proje...	1	5.0
Sonatine (Sonachi...	1	5.0
Villain (1971)	1	5.0

only showing top 10 rows

Documentary

title	count_ratings	average_rating
The Jinx: The Lif...	1	5.0
Black Tar Heroin:...	1	5.0
Dylan Moran: Mons...	1	5.0
Heidi Fleiss: Hol...	2	5.0
Bitter Lake (2015)	1	5.0
Zeitgeist: Moving...	1	5.0
Connections (1978)	1	5.0
Tickling Giants (...	1	5.0
Louis Theroux: La...	1	5.0
Martin Lawrence L...	1	5.0

only showing top 10 rows

Drama

title	count_ratings	average_rating
Sonatine (Sonachi...	1	5.0
Human Condition I...	1	5.0
Paper Birds (Pája...	1	5.0
'Salem's Lot (2004)	1	5.0
Lady Jane (1986)	1	5.0
Rain (2001)	1	5.0
Villain (1971)	1	5.0
Reform School Gir...	1	5.0
English Vinglish ...	1	5.0
Awfully Big Adven...	1	5.0

only showing top 10 rows

Fantasy

title	count_ratings	average_rating
L.A. Slasher (2015)	1	5.0
Nine Lives of Tom...	1	5.0
Into the Forest o...	1	5.0
Lesson Faust (1994)	2	5.0
Holy Motors (2012)	1	5.0
Into the Woods (1...	1	5.0
Stuart Little 3: ...	1	5.0
Light Years (Gand...	1	5.0
In the blue sea, ...	1	5.0
Goodbye Charlie (...	1	5.0

only showing top 10 rows

Film-Noir

title	count_ratings	average_rating
Rififi (Du rififi...	2	4.75
Long Goodbye, The...	3	4.666666666666667
You Only Live Onc...	1	4.5
13 Tzameti (2005)	1	4.5
Sunset Blvd. (a.k...	27	4.333333333333333
Laura (1944)	6	4.333333333333333
Double Indemnity ...	17	4.323529411764706
Touch of Evil (1958)	17	4.264705882352941
Notorious (1946)	20	4.25
Third Man, The (1...	24	4.229166666666667

only showing top 10 rows

Horror

title	count_ratings	average_rating
The Editor (2015)	1	5.0
Lumberjack Man (2...	1	5.0
Buzzard (2015)	1	5.0
Ex Drummer (2007)	1	5.0
The Eye: Infinity...	1	5.0
Alien Contaminati...	1	5.0
Hellbenders (2012)	1	5.0
'Salem's Lot (2004)	1	5.0
Maniac Cop 2 (1990)	1	5.0
Monster Squad, Th...	1	5.0

only showing top 10 rows

IMAX

title	count_ratings	average_rating
Happy Feet Two (2...	1	5.0
More (1998)	1	5.0
Journey to the We...	2	4.75
Final Destination...	1	4.5
Everest (1998)	6	4.25
Dark Knight, The ...	149	4.238255033557047
Toy Story 3 (2010)	55	4.109090909090909
Inception (2010)	143	4.066433566433567
Captain Phillips ...	21	4.0476190476190474
Legend of the Gua...	2	4.0

only showing top 10 rows

Musical

title	count_ratings	average_rating
Very Potter Seque...	1	5.0
Satin Rouge (2002)	1	5.0
Woman Is a Woman,...	1	5.0
Babes in Toyland ...	1	5.0
Into the Woods (1...	1	5.0
Holy Motors (2012)	1	5.0
True Stories (1986)	1	5.0
Reefer Madness: T...	2	4.75
Chorus Line, A (1...	3	4.666666666666667
Marat/Sade (1966)	2	4.5

only showing top 10 rows

Mystery

title	count_ratings	average_rating
Get Low (2009)	1	5.0
The Adventures of...	1	5.0
The Adventures of...	1	5.0
Holy Motors (2012)	1	5.0
The Editor (2015)	1	5.0
Scooby-Doo! Abrac...	1	5.0
Scooby-Doo! Curse...	1	5.0
American Friend, ...	1	5.0
Mother (Madeo) (2...	1	5.0
'Salem's Lot (2004)	1	5.0

only showing top 10 rows

(no genres listed)

title	count_ratings	average_rating
Death Note: Desu ...	1	5.0
The Adventures of...	1	5.0
Black Mirror	1	5.0
The Godfather Tri...	2	4.75
Whiplash (2013)	2	4.75
Paterson	1	4.5
The Adventures of...	1	4.5
Too Funny to Fail...	1	4.5
Lemonade (2016)	1	4.5
Cosmos	2	4.5

only showing top 10 rows

Romance

title	count_ratings	average_rating
What Love Is (2007)	1	5.0
Raise Your Voice ...	1	5.0
Belle époque (1992)	2	5.0
Love Exposure (Ai...	1	5.0
Down Argentine Wa...	1	5.0
Thousand Clowns, ...	1	5.0
Rain (2001)	1	5.0
Lady Jane (1986)	1	5.0
Goodbye Charlie (...	1	5.0
Chinese Puzzle (C...	1	5.0

only showing top 10 rows

Sci-Fi

title	count_ratings	average_rating
The Girl with All...	1	5.0
Galaxy of Terror ...	1	5.0
Stand, The (1994)	1	5.0
SORI: Voice from ...	1	5.0
Alien Contaminati...	1	5.0
Mystery of the Th...	1	5.0
A Detective Story...	1	5.0
Holy Motors (2012)	1	5.0
Light Years (Gand...	1	5.0
When Worlds Colli...	1	5.0

only showing top 10 rows

Thriller

title	count_ratings	average_rating
What Happened Was...	1	5.0
Hellbenders (2012)	1	5.0
Maniac Cop 2 (1990)	1	5.0
Cherish (2002)	1	5.0
Battle Royale 2: ...	1	5.0
'Salem's Lot (2004)	1	5.0
American Friend, ...	1	5.0
Mother (Madeo) (2...	1	5.0
Supercop 2 (Proje...	1	5.0
Villain (1971)	1	5.0

only showing top 10 rows

War

title	count_ratings	average_rating
Battle For Sevast...	1	5.0
King of Hearts (1...	1	5.0
Red Sorghum (Hong...	1	5.0
Max Manus (2008)	1	5.0
Come and See (Idi...	2	5.0
Human Condition I...	1	5.0
Che: Part Two (2008)	1	5.0
Battle Royale 2: ...	1	5.0
Guy X (2005)	1	5.0
Eichmann (2007)	1	5.0

only showing top 10 rows

Western

title	count_ratings	average_rating
7 Faces of Dr. La...	1	5.0
Trinity and Sarta...	1	5.0
Duel in the Sun (...	1	5.0
Oklahoma! (1955)	2	4.5
Ox-Bow Incident, ...	1	4.5
Texas - Doc Snyde...	1	4.5
Scalphunters, The...	1	4.5
Bone Tomahawk (2015)	2	4.5
Sweetgrass (2009)	1	4.5
Man Who Shot Libe...	5	4.4

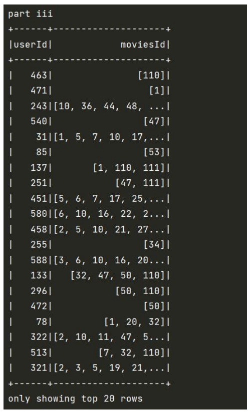
only showing top 10 rows

Figure 13: Top-10 movies movies with the highest average for each genre

I really think that it could have been more interesting to take into account the number of ratings for each film. This is why I added this possibility in the source code.

4.c Task iii

For this task we have to find the common support for all pair of the first 100 movies (Given two items, the common support is the number of users who rated both items). I did not manage to finish it but I started. Thus I will share my idea to perform this task. First we create a new dataframe made of the first 100 movies. Then we perform a inner-join between the ratings dataframe and the first 100 movies dataframe. Then we group row by "userId" and concatenate "movieId". We have now two columns: the userId and a list made of movies rated by the user. Find Below a picture of the current dataframe



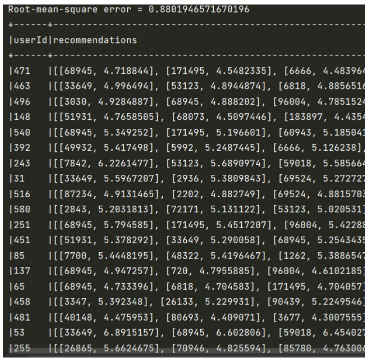
```
part iii
+------+--------------------+
|userId|            moviesId|
+------+--------------------+
|   463|               [110]|
|   471|                 [1]|
|   243|[10, 36, 44, 48, ...|
|   540|                [47]|
|    31|[1, 5, 7, 10, 17,...|
|    85|                [53]|
|   137|       [1, 110, 111]|
|   251|           [47, 111]|
|   451|[5, 6, 7, 17, 25,...|
|   580|[6, 10, 16, 22, 2...|
|   458|[2, 5, 10, 21, 27...|
|   255|                [34]|
|   588|[3, 6, 10, 16, 20...|
|   133|   [32, 47, 50, 110]|
|   296|           [50, 110]|
|   472|                [50]|
|    78|         [1, 20, 32]|
|   322|[2, 10, 11, 47, 5...|
|   513|        [7, 32, 110]|
|   321|[2, 3, 5, 19, 21,...|
+------+--------------------+
only showing top 20 rows
```

Figure 14: Top-10 movies with the largest number of ratings

What I wanted to do after was to split the moviesId column and create for each pair of rated movies by the user a key, value pair alike ((movie1,movie2),1). Finally I would have reduce by key and sum the value. Nevertheless, I faced a problem with the splitting part with a flatMap method.

4.d Recommendation System

Finally we implement a recommendation system. Unfortunately I did not have enough time to implement the method using the baseline predictor discussed during the lecture but I implemented the one using collaborative filtering and Apache Spark ALS function. The source code is in the python file "recommendation_system.py". I used the TrainValidationSplit method to optimise the regression parameter in the ALS function. Then I used the best model to display the root mean squared and the 10 best movie recommendation for each user. Find below a picture of the output of the recommendation system. The recommendation column is composed of ordered list [movieId, recommendationScore].



```
Root-mean-square error = 0.8801946571670196
+------+------------------------------------------------------------
|userId|recommendations
+------+------------------------------------------------------------
|471   |[[68945, 4.718844], [171495, 4.5482335], [6666, 4.48396
|463   |[[33649, 4.996494], [53123, 4.8944874], [6818, 4.885651
|496   |[[3030, 4.9284887], [68945, 4.888202], [96004, 4.785152
|148   |[[51931, 4.7658505], [68073, 4.5097446], [183897, 4.435
|540   |[[68945, 5.349252], [171495, 5.196601], [60943, 5.18504
|392   |[[49932, 5.417498], [5992, 5.2487445], [6666, 5.126238],
|243   |[[7842, 6.2261477], [53123, 5.6890974], [59018, 5.58566
|31    |[[33649, 5.5967207], [2936, 5.3809843], [69524, 5.27272
|516   |[[87234, 4.9131465], [2202, 4.882749], [69524, 4.881570
|580   |[[2843, 5.2031813], [72171, 5.131122], [53123, 5.020531
|251   |[[68945, 5.794585], [171495, 5.4517207], [96004, 5.4228
|451   |[[51931, 5.378292], [33649, 5.290058], [68945, 5.254343
|85    |[[7700, 5.4448195], [48322, 5.4196467], [1262, 5.388654
|137   |[[68945, 4.947257], [720, 4.7955885], [96004, 4.6102185
|65    |[[68945, 4.733396], [6818, 4.704583], [171495, 4.704057
|458   |[[3347, 5.392348], [26133, 5.229931], [90439, 5.2249546
|481   |[[40148, 4.475953], [80693, 4.409071], [3677, 4.3007555
|53    |[[33649, 6.8915157], [68945, 6.602806], [59018, 6.45402
|255   |[[26865, 5.6624675], [70946, 4.825594], [85780, 4.76300
```

Figure 15: Output of the recommendation system

5 Conclusion

To conclude, I will give my point of view. I did not manage to do everything but I really find this assignment interesting and helpful to learn the Apache Spark's ML library. I find the studied topics well define but I really think that it could have been valuable to focus on only two tasks, the mnist digits clustering and the recommendation system for instance. It could have allowed us to dive deeper in some specific task. For instance, implementing a convolution filter for the clustering task could have been really challenging.