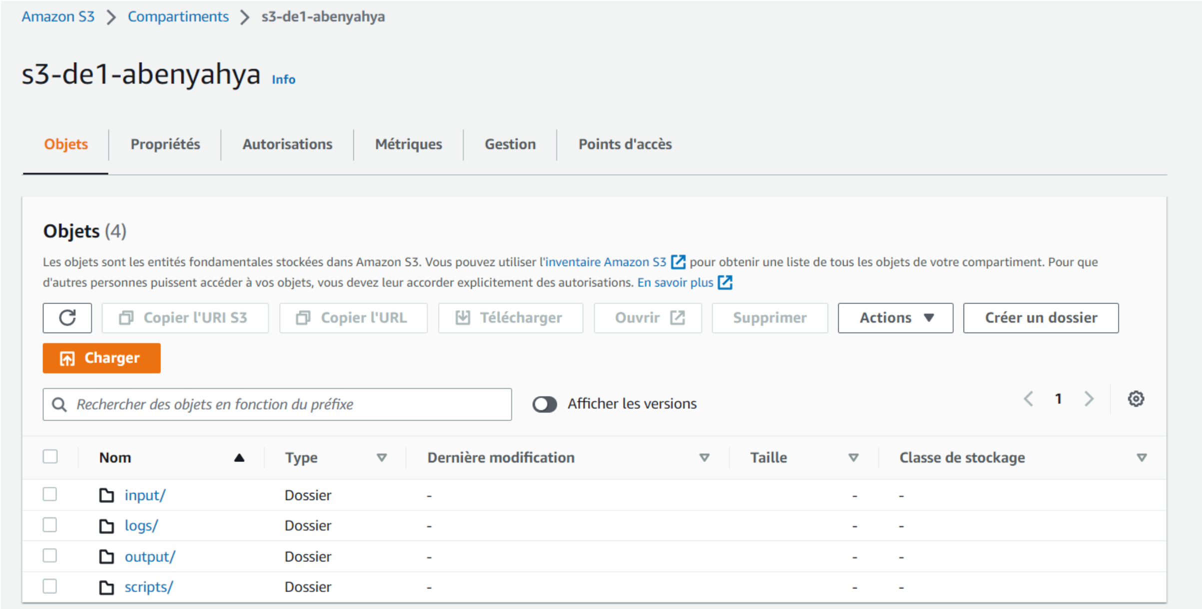
Achraf BEN YAHYA
Eranda MATHES

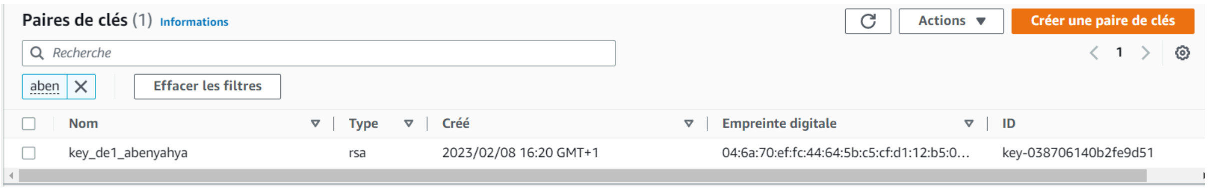
Rapport Big Data Frameworks

Movielens Data analysis : AWS EMR

S3 bucket :



Key-pair :



Cluster creation :

Achraf BEN YAHYA
Eranda MATHES

Créer un cluster | Afficher les détails | Cloner | Résilier

Filtre : Tous les clusters | Filtrer les clusters chargés... | 100 clusters chargés | charger d'autres éléments

		Nom	ID	Statut	Heure de création (UTC+1)	Temps écoulé	Heures d'instances normalisées
		cluster_de1_abenyahya	j-2EDIQIKF7BYQ1	Démarrage en cours	21-02-2023 15:27 (UTC+1)	3 minutes	0

Connection to master using ssh :

```
C:\Users\Achraf\Documents\data_frameworks>ssh -i key_de1_abenyahya.pem hadoop@ec2-3-238-15-188.compute-1.amazonaws.com
Last login: Tue Feb 21 15:01:36 2023

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
18 package(s) needed for security, out of 18 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-8-45 ~]$
```

```
[hadoop@ip-172-31-15-113 ~]$ pyspark
Python 3.7.16 (default, Dec 15 2022, 23:24:54)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-15)] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/02/24 13:13:17 WARN HiveConf: HiveConf of name hive.server2.thrift.url does not exist
23/02/24 13:13:19 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.8-amzn-2
      /_/

Using Python version 3.7.16 (default, Dec 15 2022 23:24:54)
SparkSession available as 'spark'.
>>>
```

Part 1 :
Create an ETL process with Spark :
Load the data into your Hadoop cluster and create the necessary dataframes :

Achraf BEN YAHYA
Eranda MATHES

```
>>> spark = SparkSession.builder.appName('abenyahyaSession').getOrCreate()
>>> movies = spark.read.csv("s3://nahle-bucket-datalake/emr/input/movielens/movies.csv", header = True, inferSchema = True)
>>> movies.show()
+-------+--------------------+--------------------+
|movieId|               title|              genres|
+-------+--------------------+--------------------+
|      1|    Toy Story (1995)|Adventure|Animati...|
|      2|      Jumanji (1995)|Adventure|Childre...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|Comedy|Drama|Romance|
|      5|Father of the Bri...|              Comedy|
|      6|         Heat (1995)|Action|Crime|Thri...|
|      7|      Sabrina (1995)|      Comedy|Romance|
|      8| Tom and Huck (1995)|  Adventure|Children|
|      9| Sudden Death (1995)|              Action|
|     10|    GoldenEye (1995)|Action|Adventure|...|
|     11|American Presiden...|Comedy|Drama|Romance|
|     12|Dracula: Dead and...|       Comedy|Horror|
|     13|        Balto (1995)|Adventure|Animati...|
|     14|        Nixon (1995)|               Drama|
|     15|Cutthroat Island ...|Action|Adventure|...|
|     16|       Casino (1995)|         Crime|Drama|
|     17|Sense and Sensibi...|       Drama|Romance|
|     18|   Four Rooms (1995)|              Comedy|
|     19|Ace Ventura: When...|              Comedy|
|     20|  Money Train (1995)|Action|Comedy|Cri...|
+-------+--------------------+--------------------+
only showing top 20 rows
```

```
>>> ratings  = spark.read.csv("s3://nahle-bucket-datalake/emr/input/movielens/ratings.csv", header = True, inferSchema = True)
>>> ratings.show()
+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|    307|   3.5|1256677221|
|     1|    481|   3.5|1256677456|
|     1|   1091|   1.5|1256677471|
|     1|   1257|   4.5|1256677460|
|     1|   1449|   4.5|1256677264|
|     1|   1590|   2.5|1256677236|
|     1|   1591|   1.5|1256677475|
|     1|   2134|   4.5|1256677464|
|     1|   2478|   4.0|1256677239|
|     1|   2840|   3.0|1256677500|
|     1|   2986|   2.5|1256677496|
|     1|   3020|   4.0|1256677260|
|     1|   3424|   4.5|1256677444|
|     1|   3698|   3.5|1256677243|
|     1|   3826|   2.0|1256677210|
|     1|   3893|   3.5|1256677486|
|     2|    170|   3.5|1192913581|
|     2|    849|   3.5|1192913537|
|     2|   1186|   3.5|1192913611|
|     2|   1235|   3.0|1192913585|
+------+-------+------+----------+
only showing top 20 rows
```

Achraf BEN YAHYA
Eranda MATHES

ADD year of release column to movie dataframe

```
>>> movie_with_year = movies.withColumn("year", regexp_extract(movies.title, "\((\d{4})\)", 1))
>>> movie_with_year.show()
+-------+--------------------+--------------------+----+
|movieId|               title|              genres|year|
+-------+--------------------+--------------------+----+
|      1|    Toy Story (1995)|Adventure|Animati...|1995|
|      2|      Jumanji (1995)|Adventure|Childre...|1995|
|      3|Grumpier Old Men ...|      Comedy|Romance|1995|
|      4|Waiting to Exhale...|Comedy|Drama|Romance|1995|
|      5|Father of the Bri...|              Comedy|1995|
|      6|         Heat (1995)|Action|Crime|Thri...|1995|
|      7|      Sabrina (1995)|      Comedy|Romance|1995|
|      8| Tom and Huck (1995)|  Adventure|Children|1995|
|      9| Sudden Death (1995)|              Action|1995|
|     10|    GoldenEye (1995)|Action|Adventure|...|1995|
|     11|American Presiden...|Comedy|Drama|Romance|1995|
|     12|Dracula: Dead and...|       Comedy|Horror|1995|
|     13|        Balto (1995)|Adventure|Animati...|1995|
|     14|        Nixon (1995)|               Drama|1995|
|     15|Cutthroat Island ...|Action|Adventure|...|1995|
|     16|       Casino (1995)|         Crime|Drama|1995|
|     17|Sense and Sensibi...|       Drama|Romance|1995|
|     18|   Four Rooms (1995)|              Comedy|1995|
|     19|Ace Ventura: When...|              Comedy|1995|
|     20|  Money Train (1995)|Action|Comedy|Cri...|1995|
+-------+--------------------+--------------------+----+
only showing top 20 rows
```

Add date of rating column (not used)

```
>>> ratings_date = ratings.withColumn("date", F.from_unixtime(ratings.timestamp))
>>> ratings_date.show()
+------+-------+------+----------+-------------------+
|userId|movieId|rating| timestamp|               date|
+------+-------+------+----------+-------------------+
|     1|    307|   3.5|1256677221|2009-10-27 21:00:21|
|     1|    481|   3.5|1256677456|2009-10-27 21:04:16|
|     1|   1091|   1.5|1256677471|2009-10-27 21:04:31|
|     1|   1257|   4.5|1256677460|2009-10-27 21:04:20|
|     1|   1449|   4.5|1256677264|2009-10-27 21:01:04|
|     1|   1590|   2.5|1256677236|2009-10-27 21:00:36|
|     1|   1591|   1.5|1256677475|2009-10-27 21:04:35|
|     1|   2134|   4.5|1256677464|2009-10-27 21:04:24|
|     1|   2478|   4.0|1256677239|2009-10-27 21:00:39|
|     1|   2840|   3.0|1256677500|2009-10-27 21:05:00|
|     1|   2986|   2.5|1256677496|2009-10-27 21:04:56|
|     1|   3020|   4.0|1256677260|2009-10-27 21:01:00|
|     1|   3424|   4.5|1256677444|2009-10-27 21:04:04|
|     1|   3698|   3.5|1256677243|2009-10-27 21:00:43|
|     1|   3826|   2.0|1256677210|2009-10-27 21:00:10|
|     1|   3893|   3.5|1256677486|2009-10-27 21:04:46|
|     2|    170|   3.5|1192913581|2007-10-20 20:53:01|
|     2|    849|   3.5|1192913537|2007-10-20 20:52:17|
|     2|   1186|   3.5|1192913611|2007-10-20 20:53:31|
|     2|   1235|   3.0|1192913585|2007-10-20 20:53:05|
+------+-------+------+----------+-------------------+
only showing top 20 rows
```

Achraf BEN YAHYA
Eranda MATHES

Dataframe with number of ratings and average of ratings

```
>>> movie_ratings = ratings.groupBy("movieId").agg(count("*").alias("num_ratings"), avg("rating").alias("avg_rating"))
>>> movie_ratings.show()
+-------+-----------+------------------+
|movieId|num_ratings|        avg_rating|
+-------+-----------+------------------+
|   1591|       6508|2.6466656422864165|
|   1088|      14100|3.2480141843971633|
|   2122|       2908|2.6475240715268225|
|   2366|       8252| 3.473642753271934|
|   4519|       2664|3.3402777777777777|
|   8638|       5134|3.9713673548889754|
|    833|       1562| 2.711587708066581|
|   1342|       4049|2.9728327982217833|
|   3918|       1501| 2.978014656895403|
|    148|        374| 2.907754010695187|
|   4101|         47|3.1914893617021276|
|   6357|        508|3.6663385826771653|
|  27760|         37|  3.77027027027027|
|  82529|         13| 2.923076923076923|
| 144522|         13|3.1538461538461537|
| 150604|         66|2.9696969696969697|
|    496|        424| 3.295990566037736|
| 142084|         61| 3.598360655737705|
| 104064|          8|             3.125|
|   7880|         37|3.3378378378378377|
+-------+-----------+------------------+
only showing top 20 rows
```

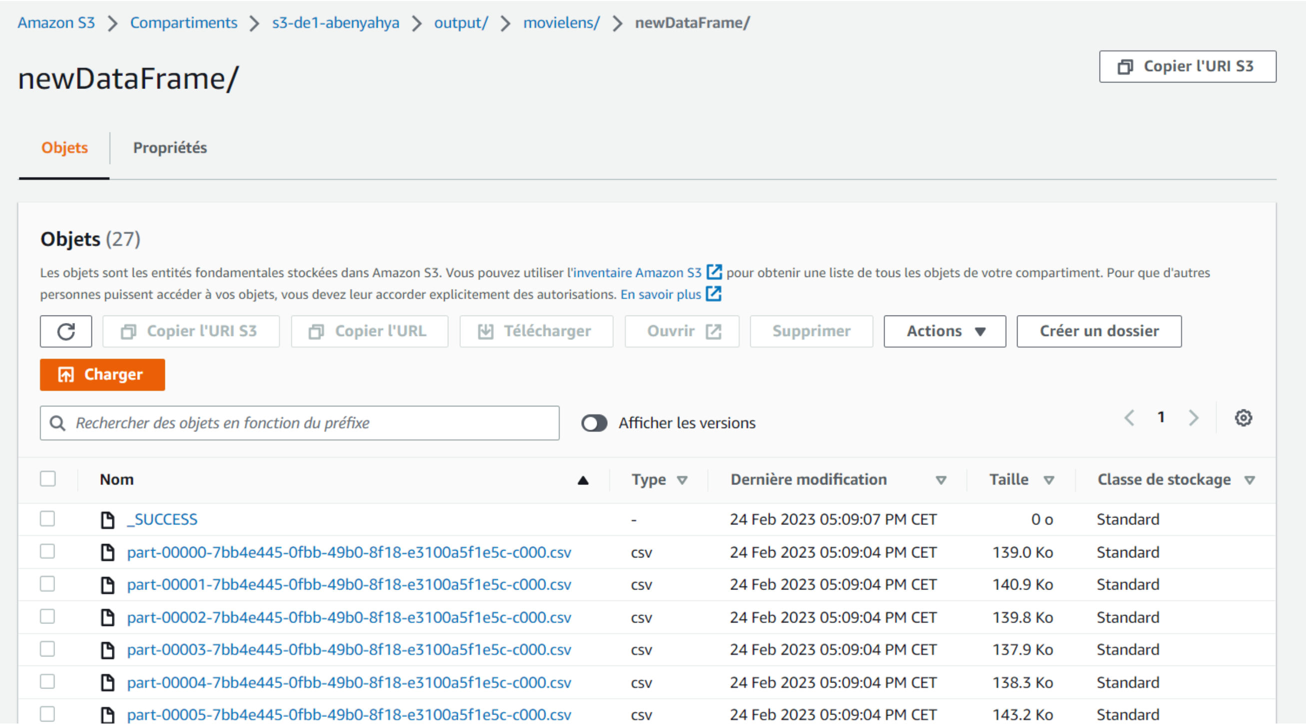
Join movies_with_year and movie_ratings to get final df

```
>>> joinedDf = movie_with_year.join(movie_ratings, movie_with_year.movieId==movie_ratings.movieId, "inner").drop(movie_ratings.movieId)
>>> joinedDf.show()
+-------+--------------------+--------------------+----+-----------+------------------+
|movieId|               title|              genres|year|num_ratings|        avg_rating|
+-------+--------------------+--------------------+----+-----------+------------------+
|   1591|        Spawn (1997)|Action|Adventure|...|1997|       6508|2.6466656422864165|
|   1088|Dirty Dancing (1987)|Drama|Musical|Rom...|1987|      14100|3.2480141843971633|
|   2122|Children of the C...|     Horror|Thriller|1984|       2908|2.6475240715268225|
|   2366|    King Kong (1933)|Action|Adventure|...|1933|       8252| 3.473642753271934|
|   4519|Land Before Time,...|Adventure|Animati...|1988|       2664|3.3402777777777777|
|   8638|Before Sunset (2004)|       Drama|Romance|2004|       5134|3.9713673548889754|
|    833|High School High ...|              Comedy|1996|       1562| 2.711587708066581|
|   1342|     Candyman (1992)|     Horror|Thriller|1992|       4049|2.9728327982217833|
|   3918|Hellbound: Hellra...|              Horror|1988|       1501| 2.978014656895403|
|    148|Awfully Big Adven...|               Drama|1995|        374| 2.907754010695187|
|   4101|Dogs in Space (1987)|               Drama|1987|         47|3.1914893617021276|
|   6357| High Society (1956)|Comedy|Musical|Ro...|1956|        508|3.6663385826771653|
|  27760|When the Last Swo...|               Drama|2003|         37|  3.77027027027027|
|  82529|Soo (Art of Reven...|Action|Crime|Dram...|2007|         13| 2.923076923076923|
| 144522|     Sky High (2003)|Action|Horror|Thr...|2003|         13|3.1538461538461537|
| 150604|  Moonwalkers (2015)|              Comedy|2015|         66|2.9696969696969697|
|    496|What Happened Was...|Comedy|Drama|Roma...|1994|        424| 3.295990566037736|
| 142084|Welcome to Leith ...|Documentary|Thriller|2015|         61| 3.598360655737705|
| 104064|Vares: The Path o...|         Crime|Drama|2012|          8|             3.125|
|   7880|Friday Night (Ven...|               Drama|2002|         37|3.3378378378378377|
+-------+--------------------+--------------------+----+-----------+------------------+
only showing top 20 rows
```

Load csv file into s3 bucket

```
>>>
>>> joinedDf.write.csv("s3://s3-de1-abenyahya/output/movielens/newDataFrame", header = True)
>>>
>>>
```

Achraf BEN YAHYA
Eranda MATHES


Amazon S3 > Compartiments > s3-de1-abenyahya > output/ > movielens/ > newDataFrame/

newDataFrame/

Copier l'URI S3

Objets | Propriétés

Objets (27)

Les objets sont les entités fondamentales stockées dans Amazon S3. Vous pouvez utiliser l'inventaire Amazon S3 pour obtenir une liste de tous les objets de votre compartiment. Pour que d'autres personnes puissent accéder à vos objets, vous devez leur accorder explicitement des autorisations. En savoir plus

Copier l'URI S3 | Copier l'URL | Télécharger | Ouvrir | Supprimer | Actions | Créer un dossier

Charger

Rechercher des objets en fonction du préfixe | Afficher les versions | 1

Nom	Type	Dernière modification	Taille	Classe de stockage
_SUCCESS	-	24 Feb 2023 05:09:07 PM CET	0 o	Standard
part-00000-7bb4e445-0fbb-49b0-8f18-e3100a5f1e5c-c000.csv	csv	24 Feb 2023 05:09:04 PM CET	139.0 Ko	Standard
part-00001-7bb4e445-0fbb-49b0-8f18-e3100a5f1e5c-c000.csv	csv	24 Feb 2023 05:09:04 PM CET	140.9 Ko	Standard
part-00002-7bb4e445-0fbb-49b0-8f18-e3100a5f1e5c-c000.csv	csv	24 Feb 2023 05:09:04 PM CET	139.8 Ko	Standard
part-00003-7bb4e445-0fbb-49b0-8f18-e3100a5f1e5c-c000.csv	csv	24 Feb 2023 05:09:04 PM CET	137.9 Ko	Standard
part-00004-7bb4e445-0fbb-49b0-8f18-e3100a5f1e5c-c000.csv	csv	24 Feb 2023 05:09:04 PM CET	138.3 Ko	Standard
part-00005-7bb4e445-0fbb-49b0-8f18-e3100a5f1e5c-c000.csv	csv	24 Feb 2023 05:09:04 PM CET	143.2 Ko	Standard

PySpark will partition the data into multiple parts based on the spark.sql.shuffle.partitions

Part 2 :
Read all partitions from s3 into one spark df

```
>>> df = spark.read.csv("s3://s3-de1-abenyahya/output/movielens/newDataFrame/*.csv", header = True, inferSchema = True)
>>> df.show()
+-------+--------------------+--------------------+----+-----------+------------------+
|movieId|               title|              genres|year|num_ratings|        avg_rating|
+-------+--------------------+--------------------+----+-----------+------------------+
|   2109|     Jerk, The (1979)|              Comedy|1979|       8488| 3.606915645617342|
| 175585|  Shot Caller (2017)|Action|Crime|Dram...|2017|        323| 3.674922600619195|
|   3498|Midnight Express ...|               Drama|1978|       3436| 3.890570430733411|
|   4756|Musketeer, The (2...|Action|Adventure|...|2001|        811|2.6017262638717633|
|   6946|Looney Tunes: Bac...|Action|Animation|...|2003|        601| 2.717970049916805|
|  61246|      Hamlet 2 (2008)|              Comedy|2008|        345| 3.163768115942029|
|  92938|Ghost Rider: Spir...|Action|Fantasy|Th...|2012|        770|2.2584415584415583|
|   1613|     Star Maps (1997)|               Drama|1997|        270|3.1592592592592594|
|   1899|Passion in the De...|     Adventure|Drama|1998|        100|             2.935|
|  99871|Jesse Stone: No R...|       Crime|Mystery|2010|         35|3.2142857142857144|
|   4507| Fresh Horses (1988)|               Drama|1988|         55|1.9363636363636363|
| 123107|The Phantom of th...|Drama|Mystery|Rom...|1990|         26| 3.519230769230769|
|  84996|Presumed Guilty (...|   Crime|Documentary|2008|          4|               3.5|
| 172715|        Platon (2008)|               Drama|2008|          8|               2.5|
|  68259|         Aamir (2008)|      Drama|Thriller|2008|         29| 3.586206896551724|
|  70862|It Might Get Loud...|         Documentary|2008|        341| 3.687683284457478|
| 117869|      Bastards (2014)|         Documentary|2014|          1|               3.5|
|  67896|Law and Order (1953)|Action|Romance|We...|1953|          3|3.3333333333333335|
| 118109|State of Emergenc...|     Sci-Fi|Thriller|2011|         16|           2.78125|
|  52189|Dark Horse (Voksn...|Comedy|Drama|Romance|2005|         30| 3.566666666666667|
+-------+--------------------+--------------------+----+-----------+------------------+
only showing top 20 rows
```

And create an SQL view

```
>>>
>>> df.createOrReplaceTempView("SQLView")
>>>
```

Achraf BEN YAHYA
Eranda MATHES

Query 1 : Best movie per year

```
>>> window_spec = Window.partitionBy("year").orderBy(desc("num_ratings"), desc("avg_rating"))
>>> ranked_movies = df.select("year", "title", "num_ratings", "avg_rating", rank().over(window_spec).alias("rank")).filter("rank=1").orderBy(desc("year")).drop("rank")
>>> ranked_movies.show()
+----+--------------------+-----------+------------------+
|year|               title|num_ratings|        avg_rating|
+----+--------------------+-----------+------------------+
|2018|Avengers: Infinit...|       2668|3.9567091454272862|
|2017|        Logan (2017)|       5209| 3.898924937607986|
|2016|     Deadpool (2016)|      13115| 3.859245139153641|
|2015|  The Martian (2015)|      16160| 4.043811881188119|
|2014| Interstellar (2014)|      23081| 4.092868593215199|
|2013|Wolf of Wall Stre...|      14748|3.8691347979387034|
|2012|Django Unchained ...|      20443| 4.002054492980482|
|2011| Intouchables (2011)|      13573| 4.127532601488249|
|2010|    Inception (2010)|      41475|4.1629897528631705|
|2009|           Up (2009)|      26143| 3.973415445817236|
|2008|Dark Knight, The ...|      44741| 4.173755615654545|
|2007|Bourne Ultimatum,...|      21677| 3.935553812796974|
|2006|V for Vendetta (2...|      27101| 3.913287332570754|
|2005|Batman Begins (2005)|      32027|3.9344303244137757|
|2004|Eternal Sunshine ...|      35064| 4.073479922427561|
|2003|Lord of the Rings...|      57378| 4.102853009864408|
|2002|Lord of the Rings...|      56696| 4.074705446592352|
|2001|Lord of the Rings...|      61883|4.0979428922321155|
|2000|    Gladiator (2000)|      48666|3.9563350182879216|
|1999|  Matrix, The (1999)|      84545| 4.149695428470046|
+----+--------------------+-----------+------------------+
only showing top 20 rows
```

```
>>> df.createOrReplaceTempView("movielens_View")
>>>
```

Check View with sql query

```
>>> query = spark.sql("SELECT * FROM movielens_View")
23/02/24 17:57:15 WARN HiveConf: HiveConf of name hive.server2.thrift.url does not exist
>>> query.show()
+-------+--------------------+--------------------+----+-----------+------------------+
|movieId|               title|              genres|year|num_ratings|        avg_rating|
+-------+--------------------+--------------------+----+-----------+------------------+
|   2109|    Jerk, The (1979)|              Comedy|1979|       8488| 3.606915645617342|
| 175585|  Shot Caller (2017)|Action|Crime|Dram...|2017|        323| 3.674922600619195|
|   3498|Midnight Express ...|               Drama|1978|       3436| 3.890570430733411|
|   4756|Musketeer, The (2...|Action|Adventure|...|2001|        811|2.6017262638717633|
|   6946|Looney Tunes: Bac...|Action|Animation|...|2003|        601| 2.717970049916805|
|  61246|     Hamlet 2 (2008)|              Comedy|2008|        345| 3.163768115942029|
|  92938|Ghost Rider: Spir...|Action|Fantasy|Th...|2012|        770|2.2584415584415583|
|   1613|    Star Maps (1997)|               Drama|1997|        270|3.1592592592592594|
|   1899|Passion in the De...|     Adventure|Drama|1998|        100|             2.935|
|  99871|Jesse Stone: No R...|       Crime|Mystery|2010|         35|3.2142857142857144|
|   4507| Fresh Horses (1988)|               Drama|1988|         55|1.9363636363636363|
| 123107|The Phantom of th...|Drama|Mystery|Rom...|1990|         26| 3.519230769230769|
|  84996|Presumed Guilty (...|   Crime|Documentary|2008|          4|               3.5|
| 172715|       Platon (2008)|               Drama|2008|          8|               2.5|
|  68259|        Aamir (2008)|      Drama|Thriller|2008|         29| 3.586206896551724|
|  70862|It Might Get Loud...|         Documentary|2008|        341| 3.687683284457478|
| 117869|     Bastards (2014)|         Documentary|2014|          1|               3.5|
|  67896|Law and Order (1953)|Action|Romance|We...|1953|          3|3.3333333333333335|
| 118109|State of Emergenc...|     Sci-Fi|Thriller|2011|         16|           2.78125|
|  52189|Dark Horse (Voksn...|Comedy|Drama|Romance|2005|         30| 3.566666666666667|
+-------+--------------------+--------------------+----+-----------+------------------+
only showing top 20 rows
```

Achraf BEN YAHYA
Eranda MATHES

Query 2 : Best movie per genre

```
query2 = spark.sql("SELECT title, genres, num_ratings, avg_rating FROM
    ( SELECT *, ROW_NUMBER() OVER (PARTITION BY genres ORDER BY avg_rating DESC, num_ratings DESC) AS rank
    FROM movielens_View
    ) WHERE rank = 1"
)
```

```
>>> query2 = spark.sql("SELECT title, genres, num_ratings, avg_rating FROM ( SELECT *, ROW_NUMBER(
= 1")
>>>
>>>
>>> query2.show()
+--------------------+--------------------+-----------+------------------+
|               title|              genres|num_ratings|        avg_rating|
+--------------------+--------------------+-----------+------------------+
|Detective Conan: ...|Action|Adventure|...|          3| 4.166666666666667|
|Teen Titans: Trou...|Action|Adventure|...|         70|3.5714285714285716|
|Davy Crockett and...|Action|Adventure|...|          2|              4.25|
|Under the Mountai...|Action|Adventure|...|         13| 2.230769230769231|
|  Justin Time (2010)|Action|Adventure|...|          1|               3.5|
|Kingsman: The Sec...|Action|Adventure|...|       8923|3.8067354028914044|
| The 39 Steps (1959)|Action|Adventure|...|          3|3.3333333333333335|
|Dragon Attack (1983)|Action|Adventure|...|         10|              2.65|
|Dragonheart 2: A ...|Action|Adventure|...|         22|               2.5|
|White Sun of the ...|Action|Adventure|...|        132| 3.837121212121212|
|First Strike (Pol...|Action|Adventure|...|       3863|3.2957545948744498|
|The Criminal Quar...|Action|Adventure|...|          3|3.8333333333333335|
|Blood Diamond (2006)|Action|Adventure|...|      12907| 3.855311071511583|
|When Eight Bells ...|Action|Adventure|...|          5|               2.8|
|    Sky Murder (1940)|Action|Adventure|...|          1|               4.0|
|Hercules Against ...|Action|Adventure|...|          1|               5.0|
|Aelita: The Queen...|Action|Adventure|...|         38|3.1052631578947367|
|Northwest Passage...|Action|Adventure|...|         97|3.5618556701030926|
|Wages of Fear, Th...|Action|Adventure|...|        998| 4.014529058116232|
|Star Wars: Episod...|Action|Adventure|...|      12747| 3.794736016317565|
+--------------------+--------------------+-----------+------------------+
only showing top 20 rows
```

Achraf BEN YAHYA
Eranda MATHES

Query 3 : Best action movie per year

```
SyntaxError: invalid syntax
>>> action_movies_df = df.filter(col("genres").contains("Action"))
>>>
>>>
>>> action_movies_df.show()
+-------+--------------------+--------------------+----+-----------+------------------+
|movieId|               title|              genres|year|num_ratings|        avg_rating|
+-------+--------------------+--------------------+----+-----------+------------------+
| 175585|  Shot Caller (2017)|Action|Crime|Dram...|2017|        323| 3.674922600619195|
|   4756|Musketeer, The (2...|Action|Adventure|...|2001|        811|2.6017262638717633|
|   6946|Looney Tunes: Bac...|Action|Animation|...|2003|        601| 2.717970049916805|
|  92938|Ghost Rider: Spir...|Action|Fantasy|Th...|2012|        770|2.2584415584415583|
|  67896|Law and Order (1953)|Action|Romance|We...|1953|          3|3.3333333333333335|
| 133802|    Slow West (2015)|Action|Thriller|W...|2015|        306| 3.560457516339869|
| 135270|Up, Up, and Away ...|     Action|Children|2000|         19|3.1578947368421053|
| 104841|      Gravity (2013)|  Action|Sci-Fi|IMAX|2013|      11982| 3.632740777833417|
|   3104| Midnight Run (1988)|Action|Comedy|Cri...|1988|       5819|3.7997078535830897|
| 130578|  Gunman, The (2015)|     Action|Thriller|2015|        137|2.9708029197080292|
| 190501|Fire on the Amazo...|Action|Adventure|...|1993|          2|              1.75|
| 157807|Labyrinth of Flam...|       Action|Comedy|2000|          1|               2.5|
| 160836|       Hazard (2005)|Action|Drama|Thri...|2005|          6|              2.75|
|    353|    Crow, The (1994)|Action|Crime|Fant...|1994|      18091|3.5139848543474654|
|   1586|    G.I. Jane (1997)|        Action|Drama|1997|       8491|2.9269226239547756|
|   2616|   Dick Tracy (1990)|        Action|Crime|1990|       7025|2.7298932384341636|
| 187109|Tremors: A Cold D...|Action|Horror|Sci-Fi|2018|         32|           2.78125|
| 128944|Honey, We Shrunk ...|Action|Adventure|...|1997|        114|2.6359649122807016|
| 187879|      Thimiru (2006)|      Action|Romance|2006|          1|               3.0|
|  79388|   Play Dirty (1969)|Action|Adventure|...|1969|         16|           3.28125|
+-------+--------------------+--------------------+----+-----------+------------------+
only showing top 20 rows
```

```
>>> window_spec = Window.partitionBy('year').orderBy(col('avg_rating').desc(), col('num_ratings').desc())
>>> best_action_movies = action_movies_df.select("year", "title","genres", "num_ratings", "avg_rating", rank().over(window_spec).alias("rank")).filter("rank=1").orderBy(desc("year")).drop("rank")
>>> best_action_movies.show()
+----+--------------------+--------------------+-----------+-----------------+
|year|               title|              genres|num_ratings|       avg_rating|
+----+--------------------+--------------------+-----------+-----------------+
|2018| Rangasthalam (2018)|        Action|Drama|          1|              5.0|
|2017|     Snowflake (2017)|Action|Comedy|Cri...|          2|             4.75|
|2016|Sherlock: The Abo...|Action|Crime|Dram...|       3076|4.013816644993498|
|2015|     Wild City (2015)|Action|Crime|Thri...|          1|              4.5|
|2014|        Yevadu (2014)|     Action|Thriller|          2|              4.5|
|2014|Lupin the Third: ...|Action|Adventure|...|          2|              4.5|
|2013|Ramaiya Vastavaiy...|Action|Comedy|Rom...|          1|              4.5|
|2013|R... Rajkumar (2013)|      Action|Romance|          1|              4.5|
|2012|CM Punk: Best in ...|        Action|Drama|          5|              4.7|
|2011|Mr. & Mrs. Incred...|Action|Comedy|Rom...|          2|              4.5|
|2010|      Baseline (2010)|  Action|Crime|Drama|          1|              5.0|
|2009|Chicago Overcoat ...|        Action|Drama|          2|             4.25|
|2009|Love Finds A Home...|Action|Children|D...|          2|             4.25|
|2008|The Accidental Ga...|Action|Adventure|...|          1|              4.5|
|2008|Best Place to be,...|Action|Drama|Thri...|          1|              4.5|
|2007|FB: Fighting Beat...|              Action|          1|              5.0|
|2006|The Baron Against...|Action|Adventure|...|          1|              5.0|
|2006|In Her Line of Fi...|Action|Drama|Thri...|          1|              5.0|
|2006|Snakes on a Train...|       Action|Horror|          1|              5.0|
|2005|Felicity: An Amer...|Action|Children|D...|          1|              4.5|
+----+--------------------+--------------------+-----------+-----------------+
only showing top 20 rows
```

Achraf BEN YAHYA
Eranda MATHES

Query 4 : Best romance movie per year

```
query4 = spark.sql( "SELECT title, year, genres, num_ratings, avg_rating FROM
    ( SELECT *, ROW_NUMBER()
        OVER (PARTITION BY year ORDER BY avg_rating DESC, num_ratings DESC ) AS rank FROM movielens_View)
        WHERE rank = 1 AND genres LIKE '%Romance%'"
    )
```

```
>>> query4 = spark.sql("SELECT title, year, genres, avg_rating, num_ratings FROM ( SELECT *, ROW_NUMBER() OVER (PARTITION BY
ank = 1 AND genres LIKE '%Romance%' ")
>>>
>>>
>>> query4.show()
+--------------------+----+--------------------+-----------------+-----------+
|               title|year|              genres|       avg_rating|num_ratings|
+--------------------+----+--------------------+-----------------+-----------+
|Gold Rush, The (1...|1925|Adventure|Comedy|...|4.052878965922444|       2553|
|      Geordie (1955)|1955|       Drama|Romance|              5.0|          1|
|Until They Sail (...|1957|   Drama|Romance|War|              5.0|          1|
| Four Mothers (1941)|1941|       Drama|Romance|              5.0|          1|
| Bed of Roses (1933)|1933|Comedy|Drama|Romance|              5.0|          1|
|The Diary of a Bi...|1988|      Comedy|Romance|              5.0|          2|
|Good Bye, Till To...|1960|       Drama|Romance|              5.0|          1|
|A Royal Winter (2...|2017|       Drama|Romance|              5.0|          2|
|       Awaken (2013)|2013|Drama|Romance|Sci-Fi|              5.0|          2|
|A Man To Remember...|1938|       Drama|Romance|              5.0|          1|
|The Memory Book (...|2014|       Drama|Romance|              5.0|          2|
|       Ramona (1936)|1936|       Drama|Romance|              5.0|          1|
|Love Finds You in...|2016|       Drama|Romance|              5.0|          2|
|Living on Love (1...|1937|      Comedy|Romance|              5.0|          1|
|    War Arrow (1954)|1954|Adventure|Drama|R...|              5.0|          2|
|Daughters Courage...|1939|       Drama|Romance|              4.5|          1|
| Ella Cinders (1926)|1926|      Comedy|Romance|             4.25|          2|
|The Most Wonderfu...|2008|       Drama|Romance|              5.0|          2|
+--------------------+----+--------------------+-----------------+-----------+
```

Then we can submit our spark application to the emr cluster using a spark-submit and we need to add the s3 path to our script.

Ajouter une étape | Cloner l'étape | Cancel step

Filtre : Toutes les étapes | Filtrer les étapes... | 3 étapes (toutes chargées)

ID	Nom	Statut	Heure de début (UTC+1)	Temps écoulé	Les fichiers journaux
s-39YUW8LDYMHFM	JAR personnalisé	En suspens		--	Aucun journal n'a été créé à ce stade

Emplacement JAR : command-runner.jar
Classe principale : Aucun
Arguments : spark-submit --deploy-mode client s3://s3-de1-abenyahya/scripts/script_movielens.py
Action sur échec : Continuer

Ajouter une étape | Cloner l'étape | Cancel step

Filtre : Toutes les étapes | Filtrer les étapes... | 3 étapes (toutes chargées)

ID	Nom	Statut	Heure de début (UTC+1)	Temps écoulé	Les fichiers journaux
s-39YUW8LDYMHFM	JAR personnalisé	Terminé	27-02-2023 20:20 (UTC+1)	1 minute	Afficher les journaux
s-27GIWH00T5A8E	JAR personnalisé	Échec	27-02-2023 20:13 (UTC+1)	2 secondes	contrôleur | syslog* | stderr | stdout*
s-3PJEZGHL3GJN6	JAR personnalisé	Échec	27-02-2023 20:03 (UTC+1)	58 secondes	contrôleur | syslog* | stderr | stdout

Then we can check if the step is finished and succeeded

```
INFO total process run time: 66 seconds
2023-02-27T19:21:23.001Z INFO Step created jobs:
2023-02-27T19:21:23.001Z INFO Step succeeded with exitCode 0 and took 66 seconds
```

And we have our script logs in the stdout

https://s3-de1-abenyahya.s3.eu-west

```
-----------Start ETL process --------------
-----------Start ETL process --------------
-----------ETL process finished--------------
-----------ETL process finished--------------
-----------PART 2 Starting queries--------------
-----------PART 2 Finished --------------
----------- Finished --------------
```

Achraf BEN YAHYA
Eranda MATHES

Bike Rental Data set : Build a predictive model to help Bike Rental companies in predicting the hourly and daily demands on bikes

Loading dataset into spark dataframe :

```
Cmd 2
1  rowData = spark.read.csv("/FileStore/tables/Bike_Rental_UCI_dataset-1.csv", inferSchema=True, header = True)
```

```
1  rowData.show(n=15)

▸ (1) Spark Jobs

+------+---+----+---+-------+----------+----------+----+----+---------+---------+----+------+
|season| yr|mnth| hr|holiday|workingday|weathersit|temp| hum|windspeed|dayOfWeek|days|demand|
+------+---+----+---+-------+----------+----------+----+----+---------+---------+----+------+
|     1|  0|   1|  0|      0|         0|         1|0.24|0.81|      0.0|      Sat|   0|    16|
|     1|  0|   1|  1|      0|         0|         1|0.22| 0.8|      0.0|      Sat|   0|    40|
|     1|  0|   1|  2|      0|         0|         1|0.22| 0.8|      0.0|      Sat|   0|    32|
|     1|  0|   1|  3|      0|         0|         1|0.24|0.75|      0.0|      Sat|   0|    13|
|     1|  0|   1|  4|      0|         0|         1|0.24|0.75|      0.0|      Sat|   0|     1|
|     1|  0|   1|  5|      0|         0|         2|0.24|0.75|   0.0896|      Sat|   0|     1|
|     1|  0|   1|  6|      0|         0|         1|0.22| 0.8|      0.0|      Sat|   0|     2|
|     1|  0|   1|  7|      0|         0|         1| 0.2|0.86|      0.0|      Sat|   0|     3|
|     1|  0|   1|  8|      0|         0|         1|0.24|0.75|      0.0|      Sat|   0|     8|
|     1|  0|   1|  9|      0|         0|         1|0.32|0.76|      0.0|      Sat|   0|    14|
|     1|  0|   1| 10|      0|         0|         1|0.38|0.76|   0.2537|      Sat|   0|    36|
|     1|  0|   1| 11|      0|         0|         1|0.36|0.81|   0.2836|      Sat|   0|    56|
|     1|  0|   1| 12|      0|         0|         1|0.42|0.77|   0.2836|      Sat|   0|    84|
|     1|  0|   1| 13|      0|         0|         2|0.46|0.72|   0.2985|      Sat|   0|    94|
|     1|  0|   1| 14|      0|         0|         2|0.46|0.72|   0.2836|      Sat|   0|   106|
+------+---+----+---+-------+----------+----------+----+----+---------+---------+----+------+
only showing top 15 rows
```

A first linear regression model was trained, but the evaluation of this first model's meanAbsoluteError and r2 was very far from being satisfactory :

```
Cmd 52
1  print ("r2=%g"%testResults.r2)    # my model explains x % of the variance of the data
2  print ("rootMeanSquaredError=%g"%testResults.rootMeanSquaredError)

r2=0.378508
rootMeanSquaredError=142.91

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
```

```
Cmd 53
1  print ("meanAbsoluteError=%g"%testResults.meanAbsoluteError)

meanAbsoluteError=107.091

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
```

Achraf BEN YAHYA
Eranda MATHES

Some insights from the results :

```python
from pyspark.sql.functions import format_number

# Grouping data by hr
pred_res.groupBy('hr').agg(format_number(avg('res_abs'), 2).alias('avg_abs_residual'),
                           format_number(avg('demand'), 2).alias('avg_demand'),
                           format_number(stddev('prediction'), 2).alias('stddev_prediction'),
                           format_number(stddev('demand'), 2).alias('stddev_demand')
                          ).sort('hr').show()
```

▸ (2) Spark Jobs

```
+---+----------------+----------+-----------------+-------------+
| hr|avg_abs_residual|avg_demand|stddev_prediction|stddev_demand|
+---+----------------+----------+-----------------+-------------+
|  0|           59.91|     53.90|            77.23|        42.31|
|  1|           71.37|     33.38|            76.14|        33.54|
|  2|           79.09|     22.87|            74.75|        26.58|
|  3|           89.08|     11.73|            72.42|        13.24|
|  4|           95.96|      6.35|            71.60|         4.14|
|  5|           86.72|     19.89|            72.30|        13.20|
|  6|           53.10|     76.04|            73.12|        55.08|
|  7|          142.46|    212.06|            76.31|       161.44|
|  8|          248.64|    359.01|            81.06|       235.19|
|  9|           76.97|    219.31|            83.90|        93.70|
| 10|           69.92|    173.67|            87.39|       102.21|
| 11|           80.86|    208.14|            89.08|       127.50|
| 12|           83.19|    253.32|            90.59|       145.08|
| 13|           89.22|    253.66|            91.05|       148.11|
| 14|           99.67|    240.95|            92.13|       147.27|
| 15|           95.17|    251.23|            92.64|       144.63|
| 16|           81.26|    311.98|            92.80|       148.68|
| 17|          207.30|    461.45|            92.95|       232.66|
```

Command took 1.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

```python
# Grouping data by season
pred_res.groupBy('season').agg(format_number(avg('res_abs'), 2).alias('avg_abs_residual'),
                               format_number(avg('demand'), 2).alias('avg_demand'),
                               format_number(stddev('prediction'), 2).alias('stddev_prediction'),
                               format_number(stddev('demand'), 2).alias('stddev_demand')
                              ).sort('season').show()
```

▸ (2) Spark Jobs

```
+------+----------------+----------+-----------------+-------------+
|season|avg_abs_residual|avg_demand|stddev_prediction|stddev_demand|
+------+----------------+----------+-----------------+-------------+
|     1|           77.99|    111.11|            98.63|       119.22|
|     2|          109.20|    208.34|           107.96|       188.36|
|     3|          127.83|    236.02|           101.51|       197.71|
|     4|          107.66|    198.87|            95.99|       182.97|
+------+----------------+----------+-----------------+-------------+
```

Achraf BEN YAHYA
Eranda MATHES

```
# Grouping data by weathersit
pred_res.groupBy('weathersit').agg(format_number(avg('res_abs'), 2).alias('avg_abs_residual'),
                            format_number(avg('demand'), 2).alias('avg_demand'),
                            format_number(stddev('prediction'), 2).alias('stddev_prediction'),
                            format_number(stddev('demand'), 2).alias('stddev_demand')
                           ).sort('weathersit').show()
```

▸ (2) Spark Jobs

```
+----------+----------------+----------+-----------------+-------------+
|weathersit|avg_abs_residual|avg_demand|stddev_prediction|stddev_demand|
+----------+----------------+----------+-----------------+-------------+
|         1|          110.83|    204.87|           113.27|       189.49|
|         2|           99.33|    175.17|           102.60|       165.43|
|         3|           88.94|    111.58|           101.10|       133.78|
|         4|           46.58|     74.33|            78.05|        77.93|
+----------+----------------+----------+-----------------+-------------+
```

```
# Grouping data by holiday
pred_res.groupBy('holiday').agg(format_number(avg('res_abs'), 2).alias('avg_abs_residual'),
                            format_number(avg('demand'), 2).alias('avg_demand'),
                            format_number(stddev('prediction'), 2).alias('stddev_prediction'),
                            format_number(stddev('demand'), 2).alias('stddev_demand')
                           ).sort('holiday').show()
```

▸ (2) Spark Jobs

```
+-------+----------------+----------+-----------------+-------------+
|holiday|avg_abs_residual|avg_demand|stddev_prediction|stddev_demand|
+-------+----------------+----------+-----------------+-------------+
|      0|          106.71|    190.43|           112.81|       181.98|
|      1|           82.86|    156.87|           107.64|       156.76|
+-------+----------------+----------+-----------------+-------------+
```

These results showed us that our model performs bad. So we decided to add dummy variables to our data and retrain the model.

We chose 6 categorical variables to convert to dummy variables : 'season', 'holiday', 'weathersit', 'dayOfWeek', 'hr' and 'mnth'.

First, we convert categorical variables to numerical values using StringIndexer :

```
# Identify the categorical variables in the dataset
categorical_cols = ['season', 'holiday', 'weathersit', 'dayOfWeek', 'hr', 'mnth']

# Convert categorical variables to numerical values using StringIndexer
indexed = [StringIndexer(inputCol = col, outputCol= col + '_idx')
           for col in categorical_cols]
```

Then we converted indexed categorical variables to dummy variables using OneHotEncoder :

```
# Convert indexed categorical variables to dummy variables using OneHotEncoder
# Create an instance of the one hot encoder
encoded = [OneHotEncoder(dropLast = False, inputCol = col + '_idx', outputCol = col + '_dum')
           for col in categorical_cols]
```

Achraf BEN YAHYA
Eranda MATHES

Then, we combined the dummy variables with our original data :

```
# Combine the dummy variables with the original dataset
assembler = VectorAssembler(
  inputCols = [
    'yr',
    'workingday',
    'temp',
    'hum',
    'windspeed'] + [col + '_dum' for col in categorical_cols],
  outputCol = 'features')
```

Then, we created the linear regression model, and defined the parameter grid to search over. We used regParam and elasticNetParam as the parameters to search over, which are regularization parameters for Linear Regression.

- **regParam: regularization parameter for L1 or L2 regularization**
- **elasticNetParam: the mixing parameter between L1 and L2 regularization.**

We used the r2 metric to evaluate the performance of our regression model, and then we created a cross validator. Finally, we Split our data into training and test sets.

```
# Create a LinearRegression model
lr = LinearRegression(featuresCol="features", labelCol="demand")

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
Cmd 77

# Define the parameter grid to search over
params = ParamGridBuilder() \
    .addGrid(lr.regParam, [0.01, 0.1, 1.0]) \
    .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \
    .build()

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
Cmd 78

# Create object to evaluate the regression model
evaluator = RegressionEvaluator(metricName="r2", labelCol=lr.getLabelCol(), predictionCol=lr.getPredictionCol())

Command took 0.09 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
Cmd 79

# Create a cross validator
cv = CrossValidator(estimator=lr, estimatorParamMaps=params, evaluator=evaluator)

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
Cmd 80

trainData, testData = rowData.randomSplit([0.7, 0.3])

trainData: pyspark.sql.dataframe.DataFrame = [season: integer, yr: integer ... 11 more fields]
testData: pyspark.sql.dataframe.DataFrame = [season: integer, yr: integer ... 11 more fields]
Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
```

Achraf BEN YAHYA
Eranda MATHES

We then create a pipeline and fit the pipeline to the training set. We use the fitted pipeline to make predictions on the test set.

```
# Construct a pipeline
pipelineLR = Pipeline(stages=indexed + encoded + [assembler, cv])

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
Cmd 82

# Train the Pipeline
pipelineModelLR = pipelineLR.fit(trainData)

▸ (95) Spark Jobs
Command took 30.77 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
Cmd 83

# Apply the fitted model on the test data to make predictions
predictionsLR = pipelineModelLR.transform(testData)

▸ predictionsLR: pyspark.sql.dataframe.DataFrame = [season: integer, yr: integer ... 25 more fields]
Command took 0.48 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
```

At this point, we could see that our model's performances were better this time, but not perfect.

```
rmse = evaluator.evaluate(predictionsLR)
print("RMSE on our test set: %g" % rmse)

# Evaluate the predictions using the RegressionEvaluator
r2 = evaluator.evaluate(predictionsLR, {evaluator.metricName: "r2"})
print("R^2 score on test set = %g" % r2)

mse = evaluator.evaluate(predictionsLR, {evaluator.metricName: "mse"})
print("MSE score on test set = %g" % mse)

▸ (3) Spark Jobs
RMSE on our test set: 0.676705
R^2 score on test set = 0.676705
MSE score on test set = 10473
Command took 2.29 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
```

So, we decided to train a new model using Random Forest Regressor. We repeated the same steps in creating indexes and dummy variables.

Achraf BEN YAHYA
Eranda MATHES

```
from pyspark.ml.regression import RandomForestRegressor
```

Command took 0.09 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 90

```
rowData.show()
```

▸ (1) Spark Jobs

```
+------+---+----+---+-------+----------+----------+----+----+---------+---------+----+------+
|season| yr|mnth| hr|holiday|workingday|weathersit|temp| hum|windspeed|dayOfWeek|days|demand|
+------+---+----+---+-------+----------+----------+----+----+---------+---------+----+------+
|     1|  0|   1|  0|      0|         0|         1|0.24|0.81|      0.0|      Sat|   0|    16|
|     1|  0|   1|  1|      0|         0|         1|0.22| 0.8|      0.0|      Sat|   0|    40|
|     1|  0|   1|  2|      0|         0|         1|0.22| 0.8|      0.0|      Sat|   0|    32|
|     1|  0|   1|  3|      0|         0|         1|0.24|0.75|      0.0|      Sat|   0|    13|
|     1|  0|   1|  4|      0|         0|         1|0.24|0.75|      0.0|      Sat|   0|     1|
|     1|  0|   1|  5|      0|         0|         2|0.24|0.75|   0.0896|      Sat|   0|     1|
|     1|  0|   1|  6|      0|         0|         1|0.22| 0.8|      0.0|      Sat|   0|     2|
|     1|  0|   1|  7|      0|         0|         1| 0.2|0.86|      0.0|      Sat|   0|     3|
|     1|  0|   1|  8|      0|         0|         1|0.24|0.75|      0.0|      Sat|   0|     8|
|     1|  0|   1|  9|      0|         0|         1|0.32|0.76|      0.0|      Sat|   0|    14|
|     1|  0|   1| 10|      0|         0|         1|0.38|0.76|   0.2537|      Sat|   0|    36|
|     1|  0|   1| 11|      0|         0|         1|0.36|0.81|   0.2836|      Sat|   0|    56|
|     1|  0|   1| 12|      0|         0|         1|0.42|0.77|   0.2836|      Sat|   0|    84|
|     1|  0|   1| 13|      0|         0|         2|0.46|0.72|   0.2985|      Sat|   0|    94|
|     1|  0|   1| 14|      0|         0|         2|0.46|0.72|   0.2836|      Sat|   0|   106|
|     1|  0|   1| 15|      0|         0|         2|0.44|0.77|   0.2985|      Sat|   0|   110|
|     1|  0|   1| 16|      0|         0|         2|0.42|0.82|   0.2985|      Sat|   0|    93|
|     1|  0|   1| 17|      0|         0|         2|0.44|0.82|   0.2836|      Sat|   0|    67|
```

Command took 0.29 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 91

```
# Identify the categorical variables in the dataset
categorical_vars = ['season', 'holiday', 'weathersit', 'dayOfWeek', 'hr', 'mnth']
```

```
# Convert categorical variables to numerical values using StringIndexer
indexers = [StringIndexer(inputCol = var, outputCol = var + '_idx')
            for var in categorical_vars]
```

Command took 0.20 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 93

```
# Convert indexed categorical variables to dummy variables using OneHotEncoder
encoders = [OneHotEncoder(dropLast = False, inputCol = var + '_idx', outputCol = var + '_dum')
            for var in categorical_vars]
```

Command took 0.20 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 94

```
# Combine the dummy variables with the original dataset
assembler = VectorAssembler(
  inputCols = [
    'yr',
    'workingday',
    'temp',
    'hum',
    'windspeed'] + [var + '_dum' for var in categorical_vars],
  outputCol = 'features')
```

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Then we defined our RandomForestRegressor model and then the parameter grid to search over. In this grid, we tuned two hyperparameters : maxDepth and numTrees.

- maxDepth refers to the maximum depth of each decision tree in the random forest. Increasing maxDepth may improve the model's performance on the training set.

- numTrees refers to the number of trees in the random forest. Increasing numTrees may improve the model's performance by reducing the variance of the model.

Then, we used cross-validation to test different combinations of these hyperparameters and choose the combination that gives the best performance on the validation set.

```
# Define the RandomForestRegressor model
rf = RandomForestRegressor(featuresCol = 'features', labelCol = 'demand')
```

Command took 0.09 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 96

```
# Define the parameter grid to search over
paramGrid = ParamGridBuilder() \
    .addGrid(rf.maxDepth, [5, 10, 15]) \
    .addGrid(rf.numTrees, [10, 20, 30]) \
    .build()
```

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 97

```
# Define the evaluator to use for the model
evaluator = RegressionEvaluator(metricName="rmse", labelCol=rf.getLabelCol(), predictionCol=rf.getPredictionCol())
```

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 98

```
# Define the cross-validator
cv = CrossValidator(estimator = rf, estimatorParamMaps = paramGrid, evaluator = evaluator)
```

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 99

```
# Split the dataset randomly into 70% for training and 30% for testing.
train, test = rowData.randomSplit([0.7, 0.3])
```

- train: pyspark.sql.dataframe.DataFrame = [season: integer, yr: integer ... 11 more fields]
- test: pyspark.sql.dataframe.DataFrame = [season: integer, yr: integer ... 11 more fields]

Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Then, we created a pipeline and fit the pipeline to the training set. We use the fitted pipeline to make predictions on the test set.

Achraf BEN YAHYA
Eranda MATHES

```
# Construct a pipeline
pipeline = Pipeline(stages=indexers + encoders + [assembler, cv])
```

```
Command took 0.10 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
Cmd 101
```

```
# Train the Pipeline
pipelineModel = pipeline.fit(train)
```

```
▸ (53) Spark Jobs
Command took 7.33 minutes -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
Cmd 102
```

```
# Apply the fitted model on the test data to make predictions
predictions = pipelineModel.transform(test)
```

```
▸ predictions: pyspark.sql.dataframe.DataFrame = [season: integer, yr: integer ... 25 more fields]
Command took 2.73 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData
```

We could see that our new model performs better than the previous one.

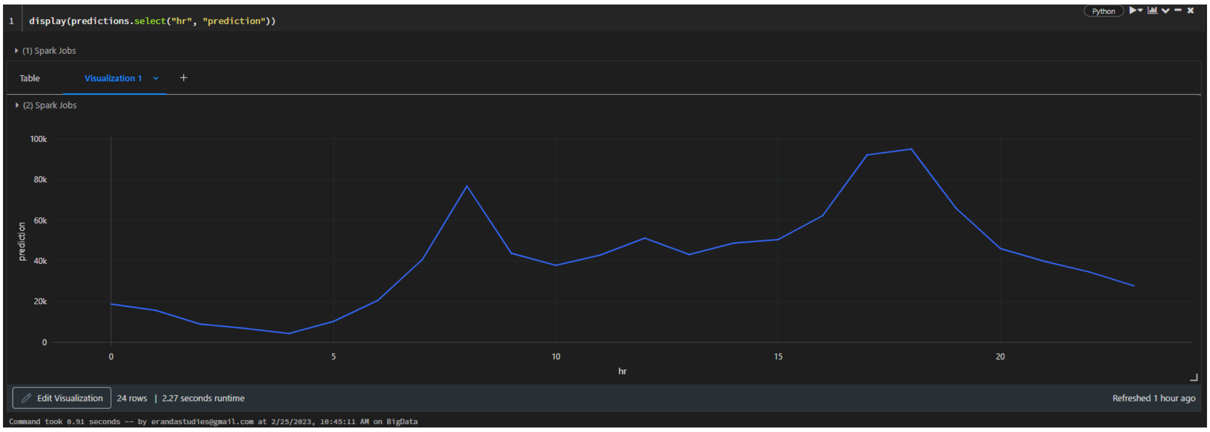
```
# Evaluate the predictions using the RegressionEvaluator
rmse = evaluator.evaluate(predictions)
print("RMSE on our test set: %g" % rmse)

r2 = evaluator.evaluate(predictions, {evaluator.metricName: "r2"})
print("R^2 score on test set = %g" % r2)

mse = evaluator.evaluate(predictions, {evaluator.metricName: "mse"})
print("MSE score on test set = %g" % mse)
```

```
▸ (3) Spark Jobs
RMSE on our test set: 68.0505
R^2 score on test set = 0.856748
MSE score on test set = 4630.87
```

The following plot shows the number of bicycle rentals during each hour of the day. As we expect, rentals are low during the night, and peak at commute hours.



Achraf BEN YAHYA
Eranda MATHES

When compared, we can clearly see that the second model which uses Random Forest Regressor has a better r2 score than the model which uses Linear Regression.

Compare results

Cmd 108

```
results.append({'model': "RandomForestRegressor", 'r2': evaluator.evaluate(predictions, {evaluator.metricName: "r2"})})
```

▸ (1) Spark Jobs

Command took 0.97 seconds -- by erandastudies@gmail.com at 2/25/2023, 10:16:36 AM on BigData

Cmd 109

```
results_df = spark.createDataFrame(results)
results_df.select("model", "r2").orderBy(results_df.r2.desc()).show(truncate=False)
```

▸ (1) Spark Jobs

▸ results_df: pyspark.sql.dataframe.DataFrame = [model: string, r2: double]

```
+---------------------+------------------+
|model                |r2                |
+---------------------+------------------+
|RandomForestRegressor|0.8567482320683689|
|LinearRegression     |0.6767048651364174|
+---------------------+------------------+
```

SMS Spam Collection : Build a predictive model to classify an incoming sms as Spam or Safe.

Part 1 : change the text features into numeric using the suitable classes (StringIndexer, Tokenizer, StopWordsRemover, CountVectorizer, IDF, VectorAssembler).

Loading dataset into spark dataframe

Cmd 3

```
rowData = spark.read.csv("/FileStore/shared_uploads/achraf.ben.yahya@efrei.net/SMSSpamCollection", sep="\t", inferSchema=True, header = False)
# Rename columns to type (spam or ham) and text for the sms content
rowData = rowData.withColumnRenamed("_c0", "type").withColumnRenamed("_c1", "text")
```

▸ (2) Spark Jobs

```
rowData.display()
```

▸ (1) Spark Jobs

Table

	type	text
1	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
2	ham	Ok lar... Joking wif u oni...
3	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
4	ham	U dun say so early hor... U c already then say...
5	ham	Nah I don't think he goes to usf, he lives around here though

Achraf BEN YAHYA
Eranda MATHES

Start transformation of text data

```
# create a Tokenizer to split the text into words
tokenizer = RegexTokenizer(inputCol='text', outputCol='words', pattern='\\W')
tokenizer.setMinTokenLength(3)

Out[8]: RegexTokenizer_17f46a3451cb

Command took 0.36 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:43:44 on SMS_SPAM
```

words
["until", "jurong", "point", "crazy", "available", "only", "bugis", "great", "world", "buffet", "cine", "there", "got", "amore", "wat"]
["lar", "joking", "wif", "oni"]
["free", "entry", "wkly", "comp", "win", "cup", "final", "tkts", "21st", "may", "2005", "text", "87121", "receive", "entry", "question", "std", "txt", "rate", "apply", "08452810075over18"]

```
Cmd 8

# create a StopWordsRemover to Remove stop words from the words feature
stopwords_remover = StopWordsRemover(inputCol="words", outputCol="filtered_words")
```

When we apply the StopWordsRemover class to a text feature, it removes any stop words from the sequence of words in the text feature, leaving only the "important" words that we want to analyse.

filtered_words
["jurong", "point", "crazy", "available", "bugis", "great", "world", "buffet", "cine", "got", "amore", "wat"]
["lar", "joking", "wif", "oni"]
["free", "entry", "wkly", "comp", "win", "cup", "final", "tkts", "21st", "may", "2005", "text", "87121", "receive "std", "txt", "rate", "apply", "08452810075over18"]

```
Cmd 9

# create a CountVectorizer to Convert each sms into a sparse vector of word counts
count_vectorizer = CountVectorizer(inputCol="filtered_words", outputCol="raw_features")

Command took 0.12 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:43:50 on SMS_SPAM
```

Achraf BEN YAHYA
Eranda MATHES

The result of CountVectorizer is a sparse vector, where each entry represents the count of a particular word in the text, and the index of the entry corresponds to the index of the word in the vocabulary.

raw_features

{"vectorType": "sparse", "length": 8309, "indices": [52, 63, 64, 145, 174, 276, 283, 654, 735, 928, 1032, 2395], "values": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}

```python
# create an IDF to weight the bag of words features
idf = IDF(inputCol="raw_features", outputCol="features")
```

IDF class is used to transform the raw features obtained from CountVectorizer into a new set of features that take into account the importance of each word in the text feature.

features

object
vectorType: "sparse"
length: 8309
indices: [52, 63, 64, 145, 174, 276, 283, 654, 735, 928, 1032, 2395]
values:
[4.020877410340228, 4.195230797485006, 4.231598441655881, 4.754846585420428, 4.841857962410058, 5.160311693528593, 5.581525158604896, 6.141140946540319, 6.141140946540319, 6.323462503334274, 6.4288230189921, 7.52743530766021]

Cmd 11

```python
# create a StringIndexer to convert the text column into a numerical index
indexer = StringIndexer(inputCol="type", outputCol="label")
```

Command took 0.24 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:43:40 on SMS_SPAM

Cmd 12

```python
# create a VectorAssembler to combine the label and features columns into a single vector
vector_assembler = VectorAssembler(inputCols=["label", "features"], outputCol="final_features")
```

Command took 0.11 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:44:04 on SMS_SPAM

We define labels from the type column : [0 ; 1]
And then, we combine the features with the label column to contain the final_features.

Now we fit and transform the data using the preprocessing pipeline and the different classes already defined.

Achraf BEN YAHYA
Eranda MATHES

Create the pipeline, fit it to the data and transform the model

Cmd 14

```
# fit and transform the data using the pre-processing pipeline
preprocessing_pipeline = Pipeline(stages=[tokenizer, stopwords_remover, count_vectorizer, idf, indexer, vector_assembler])
df = preprocessing_pipeline.fit(rowData).transform(rowData)
```

▸ (5) Spark Jobs

Command took 12.52 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:44:10 on SMS_SPAM

Cmd 15

```
df.display()
```

Part 2 : Train 4 classifiers and compare them (LogisticRegression,DecisionTree, RandomForestClassifier, NaiveBayes)

A- LogisticRegression

Cmd 18

```
results = []
# split data with transformed test features to numeric into train and test sets
train, test = df.randomSplit([0.7, 0.3])
train.cache()
test.cache()
```

Out[16]: DataFrame[type: string, text: string, label: double, words: array<string>,

Command took 0.75 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:44:46 on SMS_SPAM

Cmd 19

```
lr = LogisticRegression(featuresCol='final_features', labelCol='label')
```

Command took 0.26 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:47:44 on SMS_SPAM

Cmd 20

```
lr_model = lr.fit(train)
```

```
evaluation_result = lr_model.evaluate(test)
```

```
Command took 0.45 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:48:16 on SMS_SPAM
Cmd 22
```

```
print("Accuracy:", evaluation_result.accuracy)
print("Precision:", evaluation_result.weightedPrecision)
print("Recall:", evaluation_result.weightedRecall)
print("F1 score:", evaluation_result.weightedFMeasure())
```

```
(1) Spark Jobs
Accuracy: 0.9878048780487805
Precision: 0.9878380185970961
Recall: 0.9878048780487805
F1 score: 0.9876086232409186
```

B- DecisionTree

```
# create a DecisionTreeClassifier model
dt = DecisionTreeClassifier(featuresCol="final_features", labelCol="label")
```

```
Command took 0.09 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:48:37 on SMS_SPAM
Cmd 26
```

```
# fit the DecisionTreeClassifier model on the training data
dt_model = dt.fit(train)
```

```
(4) Spark Jobs
Command took 10.00 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:48:41 on SMS_SPAM
Cmd 27
```

```
# make predictions on the test data
predictions  = dt_model.transform(test)
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# evaluate the model on the test data
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="label", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Accuracy:", accuracy)
```

```
(1) Spark Jobs
Accuracy: 1.0
```

We remarked that the model is maybe too complex and fits the training data too closely since we had an accuracy of one, this may indicate overfitting. We tried to limit the depth of the tree and run crossValidation on this model to be sure there is no overfitting.

```
# create the parameter grid for cross-validation
paramGrid = ParamGridBuilder() \
    .addGrid(dt.maxDepth, [2, 5, 10]) \
    .addGrid(dt.maxBins, [10, 20, 30]) \
    .build()

Command took 0.09 seconds -- by achraf.ben.yahya@efrei.net at 25/02/2023 13:25:21 on My Cluster
```

Cmd 30

```
cv = CrossValidator(estimator=dt, estimatorParamMaps=paramGrid, evaluator=evaluator, numFolds=5)

Command took 0.10 seconds -- by achraf.ben.yahya@efrei.net at 25/02/2023 13:25:21 on My Cluster
```

Cmd 31

```
cvModel = cv.fit(train)

Running command...
▸ (11) Spark Jobs
```

Cmd 32

```
# make predictions on the test data using the best model found by cross-validation
predictions_cv = cvModel.transform(test)

Waiting to run...
```

We got the same accuracy after cross validation, we can deduce that this model is not adequate for this dataset.

Cmd 33

```
# evaluate the model on the test data
accuracy2 = evaluator.evaluate(predictions_cv)
# print the accuracy score
print("Accuracy: %.4f" % accuracy2)

▸ (1) Spark Jobs
Accuracy: 1.0000
Command took 0.70 seconds -- by achraf.ben.yahya@efrei.net at 25/02/2023 13:25:21 on My Cluster
```

Achraf BEN YAHYA
Eranda MATHES

C- RandomForestClassifier

```
# create a RandomForestClassifier model
rf = RandomForestClassifier(featuresCol="final_features", labelCol="label")
```

Command took 0.13 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:50:39 on SMS_SPAM

Cmd 32

```
# fit the RandomForestClassifier model on the training data
rf_model = rf.fit(train)
```

(8) Spark Jobs

Command took 11.67 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:50:42 on SMS_SPAM

Cmd 33

```
# make predictions on the test data
predictions = rf_model.transform(test)
```

```
# evaluate the model on the test data
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="label", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Accuracy:", accuracy)
```

(1) Spark Jobs

Accuracy: 0.8722415795586528

D- NaiveBayes

```
# create a NaiveBayes model
nb = NaiveBayes(featuresCol="final_features", labelCol="label")
```

Command took 0.15 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:51:17 on SMS_SPAM

md 38

```
# fit the NaiveBayes model on the training data
nb_model = nb.fit(train)
```

(2) Spark Jobs

Command took 2.00 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:51:20 on SMS_SPAM

md 39

```
# make predictions on the test data
predictions = nb_model.transform(test)
```

Command took 0.34 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:51:25 on SMS_SPAM

md 40

```
# evaluate the model on the test data
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="label", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Accuracy:", accuracy)
```

(1) Spark Jobs

Accuracy: 0.9390243902439024

Achraf BEN YAHYA
Eranda MATHES

Part 3 : Tune at least one important hyper parameter using ParamGridBuilder and CrossValidator to improve model performance.

We chose the model with the worst accuracy (random forest) and we are going to tune numTrees and maxDepth.

Then we applied cross validation which performs k-fold cross-validation by training and evaluating the model on k different subsets of the data, each time using a different combination of hyperparameters from the parameter grid.

```
rf = RandomForestClassifier(featuresCol="final_features", labelCol="label")

Command took 0.15 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:52:26 on SMS_SPAM
Cmd 44

# create a parameter grid to search over
param_grid = ParamGridBuilder() \
    .addGrid(rf.numTrees, [10, 20, 30]) \
    .addGrid(rf.maxDepth, [5, 10, 15]) \
    .build()

Command took 0.08 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:52:37 on SMS_SPAM
Cmd 45

# create a cross-validator with the param_grid and evaluation metric
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="label", metricName="accuracy")
crossval = CrossValidator(estimator=rf,
                          estimatorParamMaps=param_grid,
                          evaluator=evaluator,
                          numFolds=3)

Command took 0.11 seconds -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:52:41 on SMS_SPAM
```

Finally we fit the crossval to our training data to obtain the model :

```
# fit the cross-validator on the training data
cv_model = crossval.fit(train)

▸ (53) Spark Jobs
Command took 6.34 minutes -- by achraf.ben.yahya@efrei.net at 24/02/2023 21:52:46 on SMS_SPAM
Cmd 47

# make predictions on the test data using the best model from the cross-validation
best_rf_model = cv_model.bestModel
predictions = best_rf_model.transform(test)

Command complete
Cmd 48

# evaluate the model on the test data
accuracy = evaluator.evaluate(predictions)
print("Accuracy:", accuracy)

Accuracy: 0.9383440986494421
```

```
1  # print the best model hyperparameters
2  print("Best numTrees:", best_rf_model.getNumTrees)
3  print("Best maxDepth:", best_rf_model.getOrDefault("maxDepth"))

Best numTrees: 10
Best maxDepth: 15
Command took 0.10 seconds -- by achraf.ben.yahya@efrei.net at 25/02/2023 13:25:21 on My Cluster
```

Part 4 : compare results

```
1  results_df = spark.createDataFrame(results)

▸ results_df: pyspark.sql.dataframe.DataFrame = [accuracy: double, model: string]
Command took 0.10 seconds -- by achraf.ben.yahya@efrei.net at 25/02/2023 13:25:21 on My Cluster
Cmd 60

1  results_df.select("model", "accuracy").orderBy(results_df.accuracy.desc()).show(truncate=False)

▸ (1) Spark Jobs
+---------------------------+------------------+
|model                      |accuracy          |
+---------------------------+------------------+
|DecisionTreeClassifier     |1.0               |
|DecisionTreeWithCrossVal   |1.0               |
|LogisticRegression         |0.9923394225103123|
|NaiveBayes                 |0.9398939304655274|
|TunedRandomForestClassifier|0.929876252209782 |
|RandomForestClassifier     |0.8727165586328816|
+---------------------------+------------------+

Command took 0.70 seconds -- by achraf.ben.yahya@efrei.net at 25/02/2023 13:25:21 on My Cluster
```

When compared, we can clearly see that the tuned random forest model shows improvements but we can also eliminate the Decision Tree for the moment since it shows a 100% accuracy which is very rare. Maybe the model is not adequate for this dataset (limited dataset - too small) and we need to test it on other sets to validate its efficiency. Finally the best models are logistic regression and Naive Bayes.