Module8 – Deep Dive into Spark MLlib
Case Study : Email Analytics
Domain: IT Security Firm
IT International (ITI) is leading the development of new software that could revolutionize how computers support decision - makers.
The IT security team of ITI building this web -based tool for Enron to

- gain insights from the emails in case of fraud and
- identify any abnormal behavior in the email communication to prevent the unexpected.

The dataset contains data from about 150 users, mostly senior management of Enron, organized into folders. The corpus contains a total of about 0.5M messages.
Tasks: As part of the BigData consultant you are expected to implement the following use Cases:
Initially all the folders are scanned and a csv file is created which is loaded into HDFS

1. **Load data into Spark DataFrame**
   emailDF = spark.read.csv("/user/edureka_524533/Datasets/Emails.csv",inferSchema=True,header=True)

2. **Display the top 10 high-frequency users based on weekly numbers of emails sent.**
   gpBySenderDF = emailDF.groupby('From').count()
   top10EmailSenders = gpBySenderDF.orderBy(col('count').desc()).limit(10)
   top10EmailSenders.show(truncate=False)

In [6]:
```
#emailPD.head()
emailDF.printSchema()
```

```
root
 |-- ID: string (nullable = true)
 |-- Date: string (nullable = true)
 |-- From: string (nullable = true)
 |-- To: string (nullable = true)
 |-- Subject: string (nullable = true)
 |-- MimeVer: string (nullable = true)
 |-- ContentType: string (nullable = true)
 |-- ContentEncoding: string (nullable = true)
 |-- FromName: string (nullable = true)
 |-- ToName: string (nullable = true)
 |-- CC: string (nullable = true)
 |-- BCC: string (nullable = true)
 |-- Folder: string (nullable = true)
 |-- Origin: string (nullable = true)
 |-- FileName: string (nullable = true)
 |-- Body: string (nullable = true)
```

Mod8CaseStudy2-Copy2

3. **Extract top 20 keywords from the subject text for both for the top 10 high-frequency users and for the non - high frequency users.**
   gpBySenderDF = emailDF.groupby('From').count()
   top10EmailSenders = gpBySenderDF.orderBy(col('count').desc()).limit(10)

```
In [16]: top10EmailSenders.show(truncate=False)
```

```
+------------------------------+-----+
|From                          |count|
+------------------------------+-----+
|lynn.blair@enron.com          |1112 |
|outlook.team@enron.com        |1030 |
|no.address@enron.com          |106  |
|john.buchanan@enron.com       |73   |
|ava.garcia@enron.com          |52   |
|michael.bodnar@enron.com      |38   |
|special@flowgo.com            |38   |
|newsletter@quickinspirations.com|37   |
|updates@send4fun.com          |32   |
|shelley.corman@enron.com      |27   |
+------------------------------+-----+
```

#Create a DF containing subject line along with Sender
subjDF = emailDF.select('From','Subject')
top10UsersDF = subjDF.groupby('From').count().orderBy(col('count').desc()).limit(10)

```
In [21]: top10UsersDF.show()
```

```
+--------------------+-----+
|                From|count|
+--------------------+-----+
|lynn.blair@enron.com| 1112|
|outlook.team@enro...| 1030|
|no.address@enron.com|  106|
|john.buchanan@enr...|   73|
|ava.garcia@enron.com|   52|
|michael.bodnar@en...|   38|
|  special@flowgo.com|   38|
|newsletter@quicki...|   37|
|updates@send4fun.com|   32|
|shelley.corman@en...|   27|
+--------------------+-----+
```

top10NonHighUsersDF = subjDF.groupby('From').count().orderBy(col('count').asc()).limit(10)

In [23]:
```
top10NonHighUsersDF.show()
```

```
+--------------------+-----+
|                From|count|
+--------------------+-----+
|press.release@enr...|    1|
|newsletter@rigzon...|    1|
|elizabeth.bouldin...|    1|
|  sales@webdesin.com|    1|
|bob.d.johnson@mai...|    1|
|dhenderson1@pclie...|    1|
|thedesk@scudderpu...|    1|
|announcements.enr...|    1|
|ken.powers@enron.com|    1|
|patrick.brennan@e...|    1|
+--------------------+-----+
```

```
# Extract Subject line for each of the ten users
subjTopDF = top10UsersDF.join(subjDF,on=['From'],how='left_outer')
subNonHighTopDF = top10NonHighUsersDF.join(subjDF,on=['From'],how='left_outer')
```

In [29]:
```
subNonHighTopDF.show()
```

```
+--------------------+-----+--------------------+
|                From|count|             Subject|
+--------------------+-----+--------------------+
|shari.stack@enron...|    1|Confirmation Temp...|
|lorna.pennicooke@...|    1|Testing the Servi...|
|phil.lowry@enron.com|    1|    RE: OneOk Letter|
|  sales@webdesin.com|    1|                null|
|ld.stephens@enron...|    1|RE: Mt. Jesus Dri...|
|bob.d.johnson@mai...|    1|          Borderline|
|kkeuter@ftenergy.com|    1|Northern Natural Gas|
|james.saunders@en...|    1|RE: TW/ENA compre...|
|frankie.adams@enr...|    1|RE: Hartley IA TB...|
|tom.halpin@enron.com|    1|Follow up on the ...|
+--------------------+-----+--------------------+
```

```
wordDF = subNonHighTopDF.withColumn('word', explode(split(col('Subject'), ' ')))\
.groupBy('word')\
.count()\
.sort('count', ascending=False)\
.limit(20)\
```

.show()

word	count
Enron	2
and	2
RE:	2
Mt.	1
Jesus	1
Drip	1
Oneok	1
FW:	1
2001-20	1
	1
Hartley	1
payment	1
Follow	1
Borderline	1
Northern	1
Natural	1
Gas	1
IA	1
TBS	1
#1	1

Now to extract the Top 20 keywords in for Top 10 Users

In [39]:
```
wordDF = subjTopDF.withColumn('word', explode(split(col('Subject'), ' ')))\
    .groupBy('word')\
    .count()\
    .sort('count', ascending=False)\
    .limit(20)\
    .show()
```

```
+----------+-----+
|      word|count|
+----------+-----+
|       FW:|  454|
|         -|  454|
|       RE:|  397|
|      Mtg.|  283|
|      room|  273|
|Conference|  220|
|   Meeting|  178|
|    EB4102|  168|
|       for|  164|
|    Oncall|  119|
|     Staff|  112|
|          |  106|
|    Weekly|  103|
|        in|   97|
|        to|   96|
|       Re:|   89|
|      Team|   85|
|        TW|   83|
|        on|   82|
|conference|   80|
+----------+-----+
```

4. **Extract top 10 keywords by identifying removing the common stop words.**

```
from pyspark.ml.feature import StopWordsRemover
stopWords = StopWordsRemover.loadDefaultStopWords('english')
def remove_stopWords(message):
    if message is None:
        return ' '
    else:
        wordList = message.split(' ')
        messageEdit = [word for word in wordList if word not in stopWords]
        message = ' '.join(messageEdit)
        return message
udf_stopWEdit = udf(remove_stopWords)
subjCleanDF = (subjDF.select('*', udf_stopWEdit(subjDF['Subject']).alias('Subject_stopW')))
```

In [49]:
```
subjCleanDF.show(5)
```

```
+--------------------+--------------------+--------------------+
|                From|             Subject|       Subject_stopW|
+--------------------+--------------------+--------------------+
|amy.fitzpatrick@e...|Fitness Club Reim...|Fitness Club Reim...|
|office.chairman@e...|Organisational An...|Organisational An...|
|enron.announcemen...|2000 Chairman's A...|2000 Chairman's A...|
|casey@mercatorpar...|         GOOD ADVICE|         GOOD ADVICE|
|office.chairman@e...|      Code of Ethics|         Code Ethics|
+--------------------+--------------------+--------------------+
only showing top 5 rows
```

In [50]:
```
wordDF = subjCleanDF.withColumn('word', explode(split(col('Subject_stopW'), ' ')))\
    .groupBy('word')\
    .count()\
    .sort('count', ascending=False)\
    .limit(10)\
    .show()
```

```
+----------+-----+
|      word|count|
+----------+-----+
|       RE:|  590|
|       FW:|  567|
|         -|  536|
|      Mtg.|  284|
|      room|  273|
|          |  243|
|Conference|  231|
|   Meeting|  225|
|    EB4102|  168|
|    Oncall|  120|
+----------+-----+
```

5. **Extend the stop words dictionary by adding your own stop words such as ' —'**

```
import string
def remove_punctuations(message):
    print(message)
    if message is None:
        return ' '
    else:
        messageEdit = [char for char in message if char not in string.punctuation]
        message = ''.join(messageEdit)
        return message
udf_puncEdit = udf(remove_punctuations)
subjNoPuncDF = (subjCleanDF.select('*',
udf_puncEdit(subjCleanDF['Subject_stopW']).alias('Subject_punc')))
```

In [56]:
```
wordDF = subjNoPuncDF.withColumn('word', explode(split(col('Subject_punc'), ' ')))\
    .groupBy('word')\
    .count()\
    .sort('count', ascending=False)\
    .limit(10)\
    .show()
```

```
+----------+-----+
|      word|count|
+----------+-----+
|           |  974|
|        RE|  590|
|        FW|  571|
|       Mtg|  325|
|      room|  273|
|   Meeting|  236|
|Conference|  232|
|    EB4102|  169|
|       Gas|  125|
|    Oncall|  121|
+----------+-----+
```

6. **Introduce a new column label to identify new, replied, and forwarded messages.**

```
forwarded = 'fwd:'
forward = 'fw:'
replied = 're:'

def checkMessageType(message):
    if message is None:
        return 'n'
    message = message.lower()
    messageList = message.split(' ')
    if (forwarded in messageList) or (forward in messageList):
        return 'f'
    elif replied in messageList:
        return 'r'
    else:
        return 'n'

udf_emailType = udf(checkMessageType)
emailDF = (emailDF.select('*', udf_emailType(emailDF['Subject']).alias('Email_Type')))

def checkNewEmail(emailType):
    if emailType == 'n':#if new set indicator to 'Y
        result = 'Y'
    else:
        result = 'N'
    return result

def checkForwardedEmail(emailType):
    if emailType == 'f':
        result = 'Y'
```

```
    else:
        result = 'N'
    return result

def checkRepliedEmail(emailType):
    if emailType == 'r':
        result = 'Y'
    else:
        result = 'N'
    return result

udf_newEmail = udf(checkNewEmail)
udf_forwardedEmail = udf(checkForwardedEmail)
udf_repliedEmail = udf(checkRepliedEmail)

emailDF = (emailDF.select('*', udf_newEmail(emailDF['Email_Type']).alias('NewEmail')))
emailDF = (emailDF.select('*',
udf_forwardedEmail(emailDF['Email_Type']).alias('ForwardedEmail')))
emailDF = (emailDF.select('*', udf_repliedEmail(emailDF['Email_Type']).alias('RepliedEmail')))
```

7. **Get the trend of the over mail activity using the pivot table from spark itself.**

```
from pyspark.sql.functions import regexp_extract, col
day_pattern='\w{3},'
emailDF = emailDF.withColumn('Day', regexp_extract(col('Date'), day_pattern, 0))
month_pattern=' \w{3} '
emailDF = emailDF.withColumn('Month', regexp_extract(col('Date'), month_pattern, 0))
year_pattern = '[1-2]\d{3}'
emailDF = emailDF.withColumn('Year', regexp_extract(col('Date'), year_pattern, 0))
```

In [110]:
```
emailDF.printSchema()
```

```
root
 |-- ID: string (nullable = true)
 |-- Date: string (nullable = true)
 |-- From: string (nullable = true)
 |-- To: string (nullable = true)
 |-- Subject: string (nullable = true)
 |-- MimeVer: string (nullable = true)
 |-- ContentType: string (nullable = true)
 |-- ContentEncoding: string (nullable = true)
 |-- FromName: string (nullable = true)
 |-- ToName: string (nullable = true)
 |-- CC: string (nullable = true)
 |-- BCC: string (nullable = true)
 |-- Folder: string (nullable = true)
 |-- Origin: string (nullable = true)
 |-- FileName: string (nullable = true)
 |-- Body: string (nullable = true)
 |-- Email_Type: string (nullable = true)
 |-- NewEmail: string (nullable = true)
 |-- ForwardedEmail: string (nullable = true)
 |-- RepliedEmail: string (nullable = true)
 |-- Day: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- Year: string (nullable = true)
```

emailDF.groupBy('From').pivot('Email_Type').count().show()

In [111]:
```
emailDF.groupBy('From').pivot('Email_Type').count().show()
```

```
+--------------------+----+----+----+
|                From|   f|   n|   r|
+--------------------+----+----+----+
|alice.johnson@enr...|   1|   3|null|
| rick.kile@enron.com|   1|null|   1|
|announcements.enr...|null|   1|null|
|thedesk@scudderpu...|null|   1|null|
|dhenderson1@pclie...|null|   1|null|
|integrated.soluti...|null|   3|null|
|nancy.bagot@enron...|null|   3|null|
|   40enron@enron.com|null|   8|   1|
|ken.powers@enron.com|null|   1|null|
|maggie.matheson@e...|null|   4|null|
|    ipayit@enron.com|null|   2|null|
|  sales@webdesin.com|null|   1|null|
|michele.winckowsk...|null|   4|   7|
|elizabeth.bouldin...|null|   1|null|
|bob.d.johnson@mai...|null|   1|null|
|raetta.zadow@enro...|  10|  11|   6|
|e..anderson@enron...|   2|   1|   5|
|press.release@enr...|null|   1|null|
|newsletter@rigzon...|null|   1|null|
|   news@real-net.net|null|   3|null|
+--------------------+----+----+----+
only showing top 20 rows
```

emailDF.filter(emailDF['Year'].isNotNull()).groupBy('From').pivot('Email_Type').count().show()

```
+--------------------+----+----+----+
|                From|   f|   n|   r|
+--------------------+----+----+----+
|alice.johnson@enr...|   1|   3|null|
|thedesk@scudderpu...|null|   1|null|
| rick.kile@enron.com|   1|null|   1|
|announcements.enr...|null|   1|null|
|integrated.soluti...|null|   3|null|
|dhenderson1@pclie...|null|   1|null|
|nancy.bagot@enron...|null|   3|null|
|   40enron@enron.com|null|   8|   1|
|maggie.matheson@e...|null|   4|null|
|ken.powers@enron.com|null|   1|null|
|    ipayit@enron.com|null|   2|null|
|  sales@webdesin.com|null|   1|null|
|elizabeth.bouldin...|null|   1|null|
|michele.winckowsk...|null|   4|   7|
|raetta.zadow@enro...|  10|  11|   6|
|bob.d.johnson@mai...|null|   1|null|
|e..anderson@enron...|   2|   1|   5|
|press.release@enr...|null|   1|null|
|newsletter@rigzon...|null|   1|null|
|   news@real-net.net|null|   3|null|
+--------------------+----+----+----+
only showing top 20 rows
```

8. **Use k-means clustering to create 4 clusters from the extracted keywords.**

```
from pyspark.ml.clustering import KMeans,KMeansModel
from pyspark.ml.feature import VectorAssembler
#Extract all words from subject line and order them in descending order
allWordsDF = subjNoPuncDF.withColumn('word', explode(split(col('Subject_punc'), ' ')))\
  .groupBy('word')\
  .count()\
  .sort('count', ascending=False)\
  .collect()
```

```
AllWordsDF= spark.createDataFrame(allWordsDF)
vc = VectorAssembler(inputCols=['count'],outputCol='features')
newVecDF = vc.transform(AllWordsDF)
```

```
In [65]: newVecDF.show(5)
```

```
+----+-----+--------+
|word|count|features|
+----+-----+--------+
|    |  974| [974.0]|
|  RE|  590| [590.0]|
|  FW|  571| [571.0]|
| Mtg|  325| [325.0]|
|room|  273| [273.0]|
+----+-----+--------+
only showing top 5 rows
```

```
kmeans = KMeans(k=4,seed=1)
model = kmeans.fit(newVecDF.select('features'))
transformed = model.transform(newVecDF)
```

```
In [69]: transformed.show(5)
```

```
+----+-----+--------+----------+
|word|count|features|prediction|
+----+-----+--------+----------+
|    |  974| [974.0]|         1|
|  RE|  590| [590.0]|         3|
|  FW|  571| [571.0]|         3|
| Mtg|  325| [325.0]|         3|
|room|  273| [273.0]|         3|
+----+-----+--------+----------+
only showing top 5 rows
```

```
centers = model.clusterCenters()
```

```
In [74]: for center in centers:
             print(center)
```

```
[3.22195775]
[974.]
[67.34883721]
[371.16666667]
```

10. **Use LDA to generate 4 topics from the extracted keywords. Can you identify top keywords in the spam messages across the organization?**

```
# Trains a LDA model.
lda = LDA(k=10, maxIter=10)
model1 = lda.fit(newVecDF)

# Describe topics.
topics = model.describeTopics(3)
print("The topics described by their top-weighted terms:")
topics.show(truncate=False)
```

```
The topics described by their top-weighted terms:
+-----+-----------+-----------+
|topic|termIndices|termWeights|
+-----+-----------+-----------+
|0    |[0]        |[1.0]      |
|1    |[0]        |[1.0]      |
|2    |[0]        |[1.0]      |
|3    |[0]        |[1.0]      |
|4    |[0]        |[1.0]      |
|5    |[0]        |[1.0]      |
|6    |[0]        |[1.0]      |
|7    |[0]        |[1.0]      |
|8    |[0]        |[1.0]      |
|9    |[0]        |[1.0]      |
+-----+-----------+-----------+
```

```
# Shows the result
transformed = model1.transform(newVecDF)
transformed.show(truncate=False)
```

```
+----------+-----+--------+-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------+
|word      |count|features|topicDistribution
|
+----------+-----+--------+-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------
-----------+
|          |974  |[974.0] |[1.0114433398680371E-4,1.0078016005676634E-4,1.0081246419437194E-4,1.0115524146413965E-4,
1.0063162334282231E-4,1.0140054313467611E-4,1.0074216062689085E-4,1.0052439231682387E-4,1.005975471431424E-4,0.999092
2115337335] |
|RE        |590  |[590.0] |[1.668643364495297E-4,1.662635348231376E-4,1.663145840494268E-4,1.6688233123074775E-4,1.66
01848406411847E-4,1.672870214988523E-4,1.662008446712144 8E-4,1.6584185924967823E-4,0.998501512650346,1.6681435361716
985E-4]    |
|FW        |571  |[571.0] |[1.7240719488028895E-4,0.9984520368305188,1.7183918090431523E-4,1.7242578740891 64E-4,1.715
3324512366984E-4,1.7285492349736076E-4,1.717216633521327E-4,1.7135046263552835E-4,1.7147515995176736E-4,1.72355551727
19059E-4]  |
|Mtg       |325  |[325.0] |[3.025125386336153E-4,3.014233288446393E-4,3.0151587738880574E-4,3.0254516186805264E-4,3.0
0988056647075 94E-4,3.032788360119027 5E-4,3.013096761472055E-4,3.0065835176238976E-4,3.0087715091465356E-4,0.997284891
0217816]   |
|room      |273  |[273.0] |[3.5992717628415363E-4,3.5863124180023634E-4,3.5874135543488594E-4,3.599911447084365E-4,3.
5810266514415477E-4,3.608389117680548E-4,3.5849601858638507E-4,3.577210772257907E-4,3.579814029806149E-4,0.9967695690
060673]    |
|Meeting   |236  |[236.0] |[4.1612229675341E-4,4.1462473673463905E-4,4.1475133397951113E-4,4.161671718210437E-4,4.140
129251361809E-4,4.171763812949966E-4,0.996263703013492,4.1357175999624774E-4,4.1387273034262E-4,4.15997650449298E-4]
|
|Conference|232  |[232.0] |[4.232665390333677E-4,4.2174254731619184E-4,0.9961998450240434,4.2331218454683074E-4,4.211
209522861899E-4,4.2433872081957993E-4,4.215835276603728E-4,4.2067240104447766E-4,4.2097835053308115E-4,4.231397527163
3864E-4]   |
|EB4102    |169  |[169.0] |[5.801395456725673E-4,5.780512895757818E-4,0.994791413681798,5.802021086366695E-4,5.771987
470828566E-4,5.816091067905659E-4,5.778327652464445E-4,5.765836929352121E-4,5.7700329331302 73E-4,5.799657689489519E-
4]         |
|Gas       |125  |[125.0] |[7.827548065407828E-4,7.799364498446207E-4,7.801759214257081E-4,7.828392199565428E-4,7.787
86918420313E-4,0.9929768633197081,7.796423707643883E-4,7.779570533908592E-4,7.785232013961377E-4,7.825207385523664E-
4]          |
|Oncall    |121  |[121.0] |[8.084224130662976E-4,8.055116377775191E-4,8.057608404114338E-4,8.085095945384456E-4,8.043
```