EECS E6893 Big Data Analytics

Homework #2 Report

Name: Yi Zhang

Uni: yz4130

October 2022

Part I

Marcro Setting

In [ ]:
```python
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

In [ ]:
```python
spark = SparkSession.builder.appName("Adult Data Binary Classifier").getOrCreate()
```

1. Data loading

In [ ]:
```python
#Read csv file to dataframe
data = spark.read.csv("adult.csv")
data.show(5)
```

```
+---+-----------------+-------+----------+---+-------------------+------------------+--------------+------+-------+-----+----+----+--------------+-
-----+
|_c0|              _c1|    _c2|       _c3|_c4|                _c5|               _c6|           _c7|   _c8|    _c9| _c10|_c11|_c12|           _c13|
_c14|
+---+-----------------+-------+----------+---+-------------------+------------------+--------------+------+-------+-----+----+----+--------------+-
-----+
| 39|        State-gov|  77516| Bachelors| 13|      Never-married|      Adm-clerical| Not-in-family| White|   Male| 2174|   0|  40| United-States|
<=50K|
| 50| Self-emp-not-inc|  83311| Bachelors| 13| Married-civ-spouse|   Exec-managerial|       Husband| White|   Male|    0|   0|  13| United-States|
<=50K|
| 38|          Private| 215646|   HS-grad|  9|           Divorced| Handlers-cleaners| Not-in-family| White|   Male|    0|   0|  40| United-States|
<=50K|
| 53|          Private| 234721|      11th|  7| Married-civ-spouse| Handlers-cleaners|       Husband| Black|   Male|    0|   0|  40| United-States|
<=50K|
| 28|          Private| 338409| Bachelors| 13| Married-civ-spouse|    Prof-specialty|          Wife| Black| Female|    0|   0|  40|          Cuba|
<=50K|
+---+-----------------+-------+----------+---+-------------------+------------------+--------------+------+-------+-----+----+----+--------------+-
-----+
only showing top 5 rows
```

In [ ]:
```python
#change the column names of dataframe
df = data.withColumnRenamed('_c0','age').withColumnRenamed('_c1','workclass').withColumnRenamed('_c2','fnlwgt')\
.withColumnRenamed('_c3', 'education').withColumnRenamed('_c4', 'education_num')\
.withColumnRenamed('_c5','marital_status').withColumnRenamed('_c6', 'occupation').withColumnRenamed('_c7', 'relationship')\
.withColumnRenamed('_c8', 'race').withColumnRenamed('_c9', 'sex').withColumnRenamed('_c10', 'capital_gain')\
.withColumnRenamed('_c11', 'capital_loss').withColumnRenamed('_c12','hours_per_week')\
.withColumnRenamed('_c13', 'native_country').withColumnRenamed('_c14', 'income')

df.printSchema()
df.show(5)

dataset = df
```

```
root
 |-- age: string (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: string (nullable = true)
 |-- education: string (nullable = true)
 |-- education_num: string (nullable = true)
 |-- marital_status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital_gain: string (nullable = true)
 |-- capital_loss: string (nullable = true)
 |-- hours_per_week: string (nullable = true)
 |-- native_country: string (nullable = true)
 |-- income: string (nullable = true)

+---+-----------------+-------+----------+-------------+-------------------+------------------+--------------+------+-------+------------+---------
---+--------------+--------------+------+
|age|        workclass| fnlwgt| education|education_num|     marital_status|        occupation|  relationship|  race|    sex|capital_gain|capital_l
oss|hours_per_week|native_country|income|
+---+-----------------+-------+----------+-------------+-------------------+------------------+--------------+------+-------+------------+---------
---+--------------+--------------+------+
| 39|        State-gov|  77516| Bachelors|           13|      Never-married|      Adm-clerical| Not-in-family| White|   Male|        2174|
0|            40| United-States| <=50K|
| 50| Self-emp-not-inc|  83311| Bachelors|           13| Married-civ-spouse|   Exec-managerial|       Husband| White|   Male|           0|
0|            13| United-States| <=50K|
| 38|          Private| 215646|   HS-grad|            9|           Divorced| Handlers-cleaners| Not-in-family| White|   Male|           0|
0|            40| United-States| <=50K|
| 53|          Private| 234721|      11th|            7| Married-civ-spouse| Handlers-cleaners|       Husband| Black|   Male|           0|
0|            40| United-States| <=50K|
| 28|          Private| 338409| Bachelors|           13| Married-civ-spouse|    Prof-specialty|          Wife| Black| Female|           0|
0|            40|          Cuba| <=50K|
+---+-----------------+-------+----------+-------------+-------------------+------------------+--------------+------+-------+------------+---------
---+--------------+--------------+------+
only showing top 5 rows
```

2. Data preprocessing

In [ ]:
```python
from pyspark.ml import Pipeline
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler
```

In [ ]:
```python
#stages in our Pipeline
stages = []
categoricalColumns = ["workclass", "education", "marital_status", "occupation", "relationship", "race", "sex", "native_country"]
```

In [ ]:
```python
for categoricalCol in categoricalColumns:
    # Category Indexing with StringIndexer
    stringIndexer = StringIndexer(inputCol=categoricalCol, outputCol=categoricalCol + "Index")
    # Use OneHotEncoder to convert categorical variables into binary SparseVectors
```

```
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()], outputCols=[categoricalCol + "classVec"])
    # Add stages. These are not run here, but will run all at once later on.
    stages += [stringIndexer, encoder]
```

In [ ]:
```
# Convert label into label indices using the StringIndexer
label_stringIdx = StringIndexer(inputCol="income", outputCol="label")
stages += [label_stringIdx]
```

In [ ]:
```
# Convert values of numeric columns from string to integer
numericCols = ["age", "fnlwgt", "education_num", "capital_gain", "capital_loss", "hours_per_week"]
for nc in numericCols:
    dataset = dataset.withColumn(nc, df[nc].cast(IntegerType()))

# Transform all features into a vector using VectorAssembler
assemblerInputs = [c + "classVec" for c in categoricalColumns] + numericCols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
stages += [assembler]
```

In [ ]:
```
pipeline = Pipeline(stages=stages)
pipelineModel = pipeline.fit(dataset)
preppedDataDF = pipelineModel.transform(dataset)
```

In [ ]:
```
# Keep relevant columns
cols = dataset.columns
selectedcols = ["label", "features"] + cols
dataset = preppedDataDF.select(selectedcols)
display(dataset)
```

```
DataFrame[label: double, features: vector, age: int, workclass: string, fnlwgt: int, education: string, education_num: int, marital_status: string,
occupation: string, relationship: string, race: string, sex: string, capital_gain: int, capital_loss: int, hours_per_week: int, native_country: str
ing, income: string]
```

In [ ]:
```
# Randomly split data into training and test sets. set seed for reproducibility
trainingData, testData = dataset.randomSplit([0.7, 0.3], seed=10)
trainingData.cache()

print(trainingData.count())
print(testData.count())
```

```
22685
9876
```

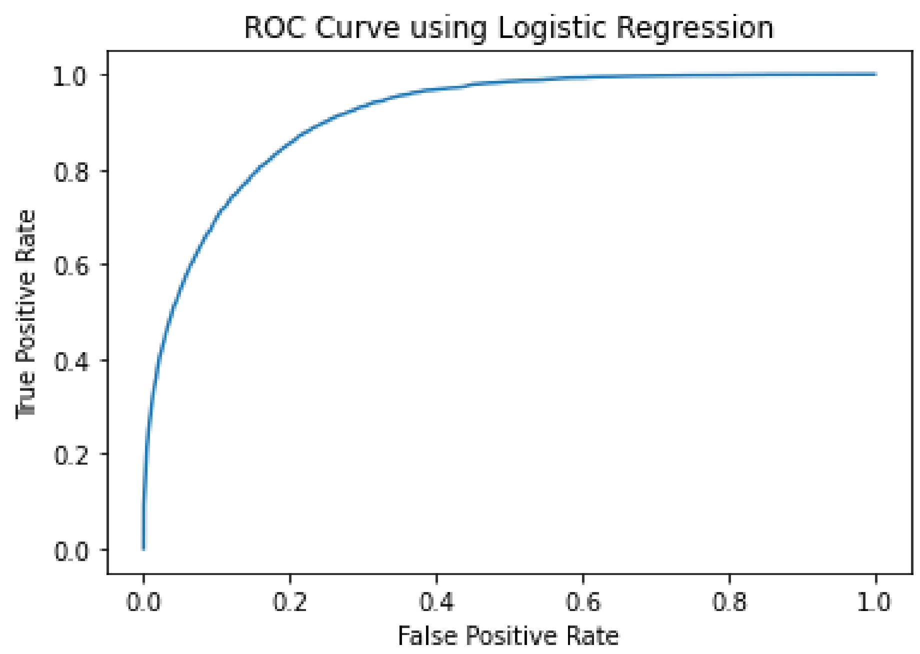
3. Modeling

In [ ]:
```
accuracy_dict = {}
```

In [ ]:
```
# LogisticRegression model, maxIter=10
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

lrModel = LogisticRegression(featuresCol="features", labelCol="label", maxIter=10).fit(trainingData)

# select example rows to display.
predictions = lrModel.transform(testData)
predictions.select(["label", "prediction"]).show(5)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
accuracy_dict['Logistic'] = round(accuracy, 4)

# draw the ROC curve
trainingSummary = lrModel.summary
roc = trainingSummary.roc.toPandas()
plt.plot(roc['FPR'], roc['TPR'])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve using Logistic Regression')
plt.show()
print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))
```

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       1.0|
|  0.0|       1.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
+-----+----------+
only showing top 5 rows
```

```
Test set accuracy = 0.8466990684487646
```



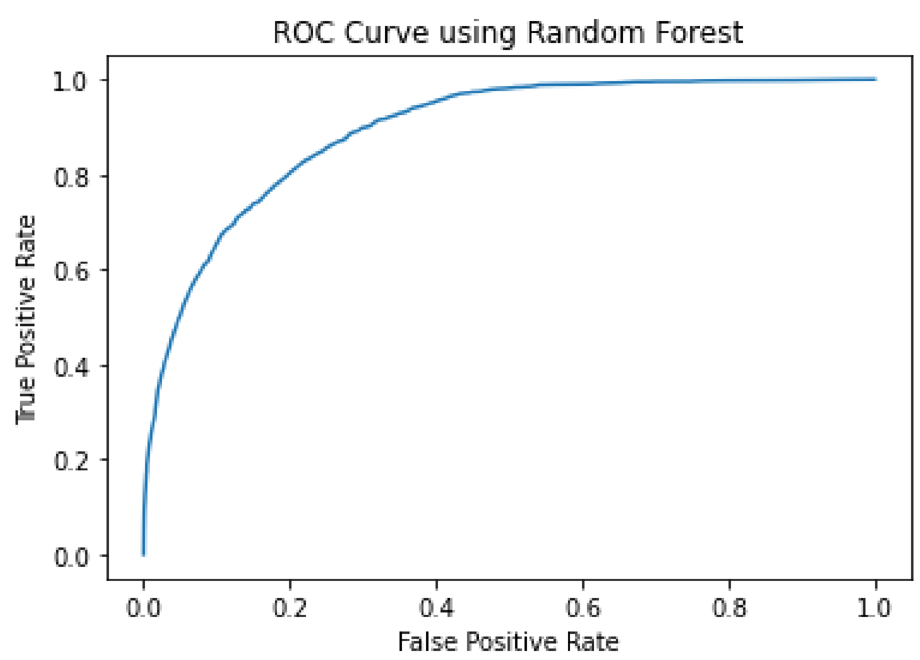

```
Training set areaUnderROC: 0.9107986021451718
```

In [ ]:

```python
# Random Forest
from pyspark.ml.classification import RandomForestClassifier

rfModel = RandomForestClassifier(featuresCol="features", labelCol="label").fit(trainingData)
predictions = rfModel.transform(testData)
predictions.select(["label", "prediction"]).show(5)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
accuracy_dict['Random Forest'] = round(accuracy, 4)

# draw the ROC curve
trainingSummary = rfModel.summary
roc = trainingSummary.roc.toPandas()
plt.plot(roc['FPR'], roc['TPR'])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve using Random Forest')
plt.show()
print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))
```

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
+-----+----------+
only showing top 5 rows
```

```
Test set accuracy = 0.8146010530579182
```




```
Training set areaUnderROC: 0.893372713336966
```

In [ ]:

```python
# NaiveBayes
from pyspark.ml.classification import NaiveBayes

nbModel = NaiveBayes(featuresCol="features", labelCol="label").fit(trainingData)

# select example rows to display.
predictions = nbModel.transform(testData)
predictions.select(["label", "prediction"]).show(5)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
accuracy_dict['NaiveBayes'] = round(accuracy, 4)
```

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       1.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
+-----+----------+
only showing top 5 rows

Test set accuracy = 0.7805791818550021
```

In [ ]:

```python
# Decision Tree
from pyspark.ml.classification import DecisionTreeClassifier
```

```python
dtree = DecisionTreeClassifier(featuresCol="features", labelCol="label").fit(trainingData)

# select example rows to display.
predictions = dtree.transform(testData)
predictions.select(["label", "prediction"]).show(5)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
accuracy_dict['DecisionTree'] = round(accuracy, 4)
```

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
+-----+----------+
only showing top 5 rows

[Stage 4081:>                                                       (0 + 1) / 1]
Test set accuracy = 0.83424463345484
```

In [ ]:

```python
# Gradient Boosting Trees
from pyspark.ml.classification import GBTClassifier

gbt = GBTClassifier(featuresCol="features", labelCol="label").fit(trainingData)

# select example rows to display.
predictions = gbt.transform(testData)
predictions.select(["label", "prediction"]).show(5)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
accuracy_dict['GBT'] = round(accuracy, 4)
```

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       1.0|
|  0.0|       1.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
+-----+----------+
only showing top 5 rows

Test set accuracy = 0.8522681247468611
```

In [ ]:

```python
# Multi-layer Perceptron
from pyspark.ml.classification import MultilayerPerceptronClassifier

layer_structure = [100, 40, 10, 2]
mlp = MultilayerPerceptronClassifier(featuresCol="features",
                                     labelCol="label",
                                     layers=layer_structure,
                                     seed=10).fit(trainingData)

# select example rows to display.
predictions = mlp.transform(testData)
predictions.show(5)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
accuracy_dict['MLP'] = round(accuracy, 4)
```

```
+-----+--------------------+---+---------+------+---------+-------------+------------------+--------------+------------+------+-----+------------
+------------+--------------+--------------+------+--------------------+--------------------+----------+
|label|            features|age|workclass|fnlwgt|education|education_num|    marital_status|    occupation|relationship|  race|  sex|capital_gain
|capital_loss|hours_per_week|native_country|income|       rawPrediction|         probability|prediction|
+-----+--------------------+---+---------+------+---------+-------------+------------------+--------------+------------+------+-----+------------
+------------+--------------+--------------+------+--------------------+--------------------+----------+
|  0.0|(100,[0,8,23,29,4...| 60|  Private|160625|  HS-grad|            9| Married-civ-spouse| Prof-specialty|     Husband| White| Male|        5013
|           0|            40| United-States| <=50K|[-1.1700061180216...|[0.33006698208408...|       1.0|
|  0.0|(100,[0,8,23,29,4...| 36|  Private|370767|  HS-grad|            9| Married-civ-spouse| Prof-specialty|     Husband| White| Male|           0
|        2377|            60| United-States| <=50K|[-0.1467829521576...|[0.79174805701775...|       0.0|
|  0.0|(100,[0,8,23,29,4...| 29|  Private| 40295|  HS-grad|            9| Married-civ-spouse| Prof-specialty|     Husband| White| Male|           0
|           0|            40| United-States| <=50K|[-0.1467829521576...|[0.79174805701775...|       0.0|
|  0.0|(100,[0,8,23,29,4...| 30|  Private| 83253|  HS-grad|            9| Married-civ-spouse| Prof-specialty|     Husband| White| Male|           0
|           0|            55| United-States| <=50K|[-0.1467829521576...|[0.79174805701775...|       0.0|
|  0.0|(100,[0,8,23,29,4...| 31|  Private| 62374|  HS-grad|            9| Married-civ-spouse| Prof-specialty|     Husband| White| Male|           0
|           0|            50| United-States| <=50K|[-0.1467829521576...|[0.79174805701775...|       0.0|
+-----+--------------------+---+---------+------+---------+-------------+------------------+--------------+------------+------+-----+------------
+------------+--------------+--------------+------+--------------------+--------------------+----------+
only showing top 5 rows

[Stage 4466:>                                                       (0 + 1) / 1]
Test set accuracy = 0.7918185500202511
```

In [ ]:

```python
# Linear Support Vector Machine
from pyspark.ml.classification import LinearSVC

lsvc = LinearSVC(featuresCol="features", labelCol="label").fit(trainingData)
```

```
# select example rows to display.
predictions = lsvc.transform(testData)
predictions.select(["label", "prediction"]).show(5)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
accuracy_dict['LSVM'] = round(accuracy, 4)
```

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       1.0|
|  0.0|       1.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
+-----+----------+
only showing top 5 rows

[Stage 4701:>                                                       (0 + 1) / 1]
Test set accuracy = 0.8463953017415958
```

In [ ]:

```
# One-vs-Rest
from pyspark.ml.classification import OneVsRest

lr = LogisticRegression(featuresCol="features", labelCol="label", maxIter=10)
ovr = OneVsRest(classifier=lr).fit(trainingData)

# select example rows to display.
predictions = ovr.transform(testData)
predictions.select(["label", "prediction"]).show(5)

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
accuracy_dict['OVR'] = round(accuracy, 4)
```

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       1.0|
|  0.0|       1.0|
|  0.0|       0.0|
|  0.0|       0.0|
|  0.0|       0.0|
+-----+----------+
only showing top 5 rows

[Stage 4731:>                                                       (0 + 1) / 1]
Test set accuracy = 0.8466990684487646
```

4. Comparison and analysis

In [ ]:

```
# Rank models according to Test set accuracy
import pandas as pd
accuracy_df = pd.DataFrame.from_dict(accuracy_dict, orient='index', columns=['Accuracy'])
accuracy_df.sort_values(by='Accuracy', ascending=False, inplace=True)
accuracy_df
```

Out[ ]:

	Accuracy
GBT	0.8523
Logistic	0.8467
OVR	0.8467
LSVM	0.8464
DecisionTree	0.8342
Random Forest	0.8146
MLP	0.7918
NaiveBayes	0.7806

Result Analysis

The gradient boost tree method has the highest accuracy among all the methods, which is around 85%. Being an ensemble method, GBT uses boosting to put more effort on weak learners. Compared to the random forest, which simply uses bagging and train individual tree separately, GBT is is able to sequentially combine the tree. This helps to model complex structures but also may lead to overfitting very easily.

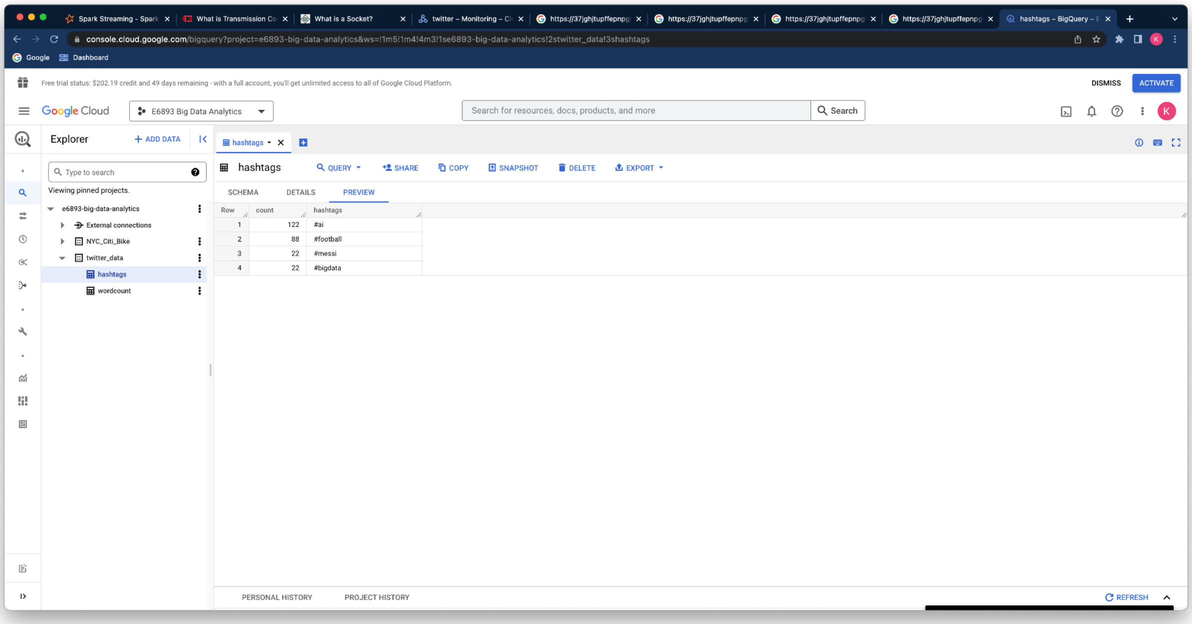
Other tree methods like OVR, decision treea and random forest also give decent outcomes because the problem's structure is suitable for tree classfication. One-versus-rest performs better than vanilla decision tree and random forest because it . And unexpectedly, the decision tree is better than random forest, which might indicate the data is noisy (or simply under the current random seed setting, the RF happens to be worse, which usually is not the case)

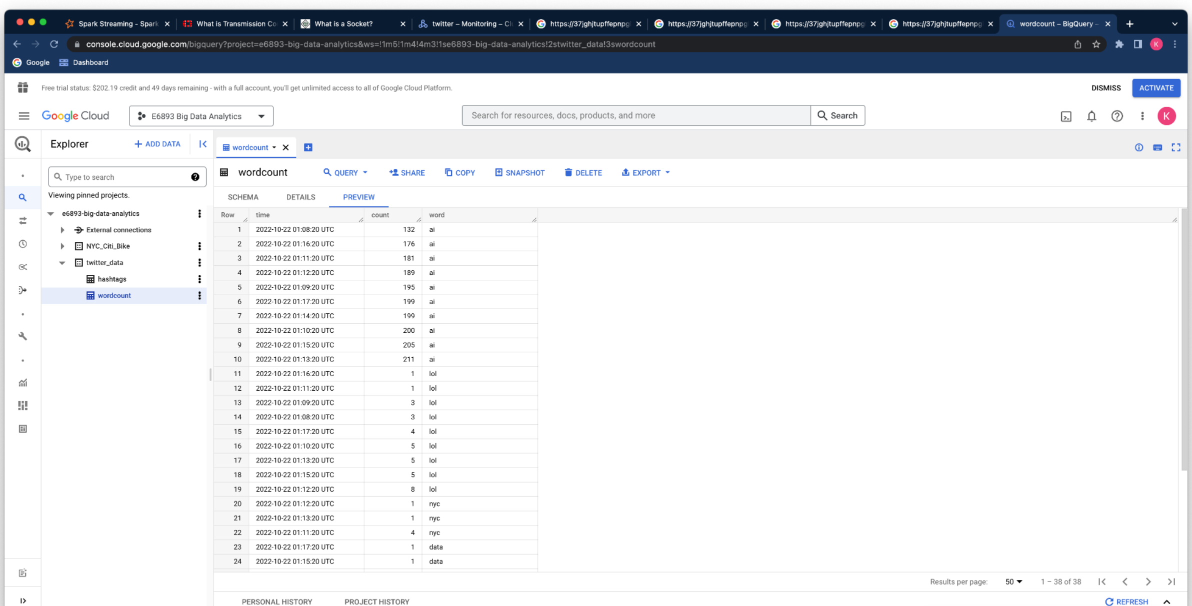
For logistic regression and linear support vector machine, they are all suitable for binary prediction, and both of their performance are great according to the ranking. Intuitively speaking, SVM handles outliers better than logistic regression, but since we are predicting income based on education, occupation etc, the dataset should not have many outliers. These two methods thus have similar performance.

Surprisingly the Multi Layer Perception method performance is not great. Since MLP heavily relies on the structure of the network, adjusting the number of layers and number of neurons within each layer might help to improve the prediction accuracy。

Part II

The codes are attached at the end of this section. Below are the previews of *hashtags* and *worcdcount* tables





In [ ]:
```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Columbia EECS E6893 Big Data Analytics
"""
This module is used to pull data from twitter API and send data to
Spark Streaming process using socket. It acts like a client of
twitter API and a server of spark streaming. It opens a listening TCP
server socket, and listens to any connection from TCP client. After
a connection established, it send streaming data to it.


Usage:
  If used with dataproc:
    gcloud dataproc jobs submit pyspark --cluster <Cluster Name> twitterHTTPClient.py

  Make sure that you run this module before you run spark streaming process.
  Please remember stop the job on dataproc if you no longer want to stream data.

"""

import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
import socket
import re

# credentials
ACCESS_TOKEN = "1578524023033667586-nYVmtbLtnhKqa40jEoMirSRmeCJ6vS"
ACCESS_SECRET = "oG98avuyMnjsX8SuVF6cfzY40Pg6AvcwSuDwnIOwaaCLJ"
CONSUMER_KEY = "xIDvSjDs3ZjREUM1WvXlScY0W"
CONSUMER_SECRET = "OV3p8VhuWPtjQnxAo32D0edi35YteQOuNJvljbH6UmHy3WWnG0"
BEARER_TOKEN = "AAAAAAAAAAAAAAAAAAAAAKfdiAEAAAAAZWSgd%2BmDUoyZU07jPDbdgNfTQ6w%3DvbWWUbmbLIsr5yVcWXjo0LKSf7CHh52oEJlgUTFmx0l7ozG4w7"

# the tags to track
tags = ['messi', 'bigdata', 'football', 'ai', 'fcbarcelona']

class MyStream(tweepy.StreamingClient):

    global client_socket
    def on_tweet(self, tweet):
        try:
            msg = tweet
#             print('TEXT:{}\n'.format(msg.text))
            # Remove some non-English tweets, reference only
            temp = re.sub('[^\u0000-\u05C0\u2100-\u214F]+', '', msg.text)
            temp = re.sub(r'http\S+', '', temp)
            client_socket.send(msg.text.encode('utf-8'))
            return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
            self.disconnect()
            return False

    def on_error(self, status):
        print(status)
        return False

def sendData(c_socket, tags):
    """
    send data to socket
    """
    global client_socket
    client_socket = c_socket
    stream = MyStream(BEARER_TOKEN)

    for tag in tags:
        stream.add_rules(tweepy.StreamRule(tag))

    stream.filter()


class twitter_client:
    def __init__(self, TCP_IP, TCP_PORT):
      self.s = s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
      self.s.bind((TCP_IP, TCP_PORT))

    def run_client(self, tags):
      try:
        self.s.listen(1)
        while True:
          print("Waiting for TCP connection...")
          conn, addr = self.s.accept()
          print("Connected... Starting getting tweets.")
          sendData(conn,tags)
          conn.close()
      except KeyboardInterrupt:
        exit


if __name__ == '__main__':
    client = twitter_client("localhost", 9001)
    client.run_client(tags)
```

```
Waiting for TCP connection...
Connected... Starting getting tweets.

Stream connection closed by Twitter

Error on_data: [Errno 32] Broken pipe
Waiting for TCP connection...
```

In [ ]:

In [1]:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Columbia EECS E6893 Big Data Analytics
"""
This module is the spark streaming analysis process.

Usage:
    If used with dataproc:
        gcloud dataproc jobs submit pyspark --cluster <Cluster Name> twitterHTTPClient.py

    Create a dataset in BigQurey first using
        bq mk bigdata_sparkStreaming

"""

from pyspark import SparkConf, SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row, SQLContext
import sys
import requests
import time
import subprocess
import re
from google.cloud import bigquery

# global variables
bucket = "6893_course_data"
output_directory_hashtags = 'gs://{}/hadoop/tmp/bigquery/pyspark_output/hashtagsCount'.format(bucket)
output_directory_wordcount = 'gs://{}/hadoop/tmp/bigquery/pyspark_output/wordcount'.format(bucket)

# output table and columns name
output_dataset = 'twitter_data'
output_table_hashtags = 'hashtags'
columns_name_hashtags = ['hashtags', 'count']
output_table_wordcount = 'wordcount'
columns_name_wordcount = ['word', 'count', 'time']

# parameter
IP = 'localhost'    # ip port
PORT = 9001       # port

# time that the streaming process runs
STREAMTIME = 600

# the tags to track
tags = ['messi', 'bigdata', 'football', 'ai', 'nyc']
tags_regex = ["^#" + tag + "$" for tag in tags]
PATTERN = "|".join(tags_regex)

# the words you should filter and do word count
WORD = ['data', 'lol', 'ai', 'news', 'nyc']
```

In [2]:
```python
# Helper functions
def saveToStorage(rdd, output_directory, columns_name, mode):
    """
    Save each RDD in this DStream to google storage
    Args:
        rdd: input rdd
        output_directory: output directory in google storage
        columns_name: columns name of dataframe
        mode: mode = "overwirte", overwirte the file
              mode = "append", append data to the end of file
    """
    if not rdd.isEmpty():
        (rdd.toDF(columns_name).write.save(output_directory, format="json", mode=mode))



def saveToBigQuery(sc, output_dataset, output_table, directory):
    """
    Put temp streaming json files in google storage to google BigQuery
    and clean the output files in google storage
    """
    files = directory + '/part-*'
    subprocess.check_call(
        'bq load --source_format NEWLINE_DELIMITED_JSON '
        '--replace '
        '--autodetect '
        '{dataset}.{table} {files}'.format(
            dataset=output_dataset, table=output_table, files=files
        ).split())
    output_path = sc._jvm.org.apache.hadoop.fs.Path(directory)
    output_path.getFileSystem(sc._jsc.hadoopConfiguration()).delete(
        output_path, True)


def hashtagCount(words):
    """
    Calculate the accumulated hashtags count sum from the beginning of the stream
    and sort it by descending order of the count.
    Ignore case sensitivity when counting the hashtags:
        "#Ab" and "#ab" is considered to be a same hashtag
    You have to:
    1. Filter out the word that is hashtags.
       Hashtag usually start with "#" and followed by a serious of alphanumeric
    2. map (hashtag) to (hashtag, 1)
    3. sum the count of current DStream state and previous state
    4. transform unordered DStream to a ordered Dstream

    Hints:
        you may use regular expression to filter the words
        You can take a look at updateStateByKey and transform transformations
    Args:
        dstream(DStream): stream of real time tweets
    Returns:
        DStream Object with inner structure (hashtag, count)
    """

    def update(newVal, runningCnt):
        if runningCnt is None:
            return sum(newVal)
        return sum(newVal) + runningCnt

    hashtag = words.map(lambda w: w.lower())\
                   .filter(lambda w: True if re.match(PATTERN, w) else False)\
                   .map(lambda tag: (tag, 1))\
                   .reduceByKey(lambda v1, v2: v1 + v2)\
                   .updateStateByKey(update)\
                   .transform(lambda rdd: rdd.sortBy(lambda x: -x[1]))
    return hashtag


def wordCount(words):
    """
    Calculte the count of 5 sepcial words for every 60 seconds (window no overlap)
    You can choose your own words.
    Your should:
    1. filter the words
    2. count the word during a special window size
    3. add a time related mark to the output of each window, ex: a datetime type
    Hints:
        You can take a look at reduceByKeyAndWindow transformation
        Dstream is a series of rdd, each RDD in a DStream contains data from a certain interval
        You may want to take a look of transform transformation of DStream when trying to add a time
    Args:
        dstream(DStream): stream of real time tweets
    Returns:
        DStream Object with inner structure (word, count, time)
    """

    window_count = words.map(lambda w: w.lower())\
                        .filter(lambda w: w in WORD)\
                        .map(lambda w: (w, 1))\
                        .reduceByKeyAndWindow(lambda x, y: x + y,
                                              lambda x, y: x - y,
                                              60,
                                              60)\
                        .transform(lambda time, rdd: rdd.map(lambda w: (w[0],
                                                                         w[1],
                                                                         time.strftime("%Y-%m-%d %H:%M:%S"))))
    window_count_time = window_count
    return window_count
```

In [3]:
```
if __name__ == '__main__':
    # Spark settings
    /gateway/default/node/conf?host&port = SparkConf()
    /gateway/default/node/conf?host&port.setMaster('local[2]')
    /gateway/default/node/conf?host&port.setAppName("TwitterStreamApp")

    # create spark context with the above configuration
    sc = SparkContext(/gateway/default/node/conf?host&port=/gateway/default/node/conf?host&port)
    sc.setLogLevel("ERROR")

    # create sql context, used for saving rdd
    sql_context = SQLContext(sc)

    # create the Streaming Context from the above spark context with batch interval size 5 seconds
    ssc = StreamingContext(sc, 5)
    # setting a checkpoint to allow RDD recovery
    ssc.checkpoint("~/checkpoint_TwitterApp")

    # read data from port 9001
    dataStream = ssc.socketTextStream(IP, PORT)
    words = dataStream.flatMap(lambda line: line.split(" "))

    # calculate the accumulated hashtags count sum from the beginning of the stream
    topTags = hashtagCount(words)
    topTags.pprint()

    # Calculte the word count during each time period 60s
    wordCount = wordCount(words)
    wordCount.pprint()

    # save hashtags count and word count to google storage
    # used to save to google BigQuery
    # You should:
    #   1. topTags: only save the lastest rdd in DStream
    #   2. wordCount: save each rdd in DStream
    # Hints:
    #   1. You can take a look at foreachRDD transformation
    #   2. You may want to use helper function saveToStorage
    #   3. You should use save output to output_directory_hashtags, output_directory_wordcount,
    #       and have output columns name columns_name_hashtags and columns_name_wordcount.

    topTags.foreachRDD(lambda rdd: saveToStorage(rdd,
                                            output_directory_hashtags,
                                            columns_name_hashtags,
                                            "overwrite"))
    wordCount.foreachRDD(lambda rdd: saveToStorage(rdd,
                                            output_directory_wordcount,
                                            columns_name_wordcount,
                                            "append"))

    # start streaming process, wait for 600s and then stop.
    ssc.start()
    time.sleep(STREAMTIME)
    ssc.stop(stopSparkContext=False, stopGraceFully=True)
    print("STREAMING END")

    # put the temp result in google storage to google BigQuery
    saveToBigQuery(sc, output_dataset, output_table_hashtags, output_directory_hashtags)
    saveToBigQuery(sc, output_dataset, output_table_wordcount, output_directory_wordcount)
    print("DONE SAVING")
```

```
-------------------------------------------
Time: 2022-10-22 01:07:25
-------------------------------------------

-------------------------------------------
Time: 2022-10-22 01:07:30
-------------------------------------------

-------------------------------------------
Time: 2022-10-22 01:07:35
-------------------------------------------
('#messi', 1)

-------------------------------------------
Time: 2022-10-22 01:07:40
-------------------------------------------
('#messi', 1)

-------------------------------------------
Time: 2022-10-22 01:07:45
-------------------------------------------
('#football', 4)
('#messi', 1)
('#bigdata', 1)
('#ai', 1)

-------------------------------------------
Time: 2022-10-22 01:07:50
-------------------------------------------
('#football', 7)
('#messi', 3)
('#bigdata', 1)
('#ai', 1)

-------------------------------------------
Time: 2022-10-22 01:07:55
-------------------------------------------
('#football', 9)
('#messi', 4)
('#ai', 3)
('#bigdata', 1)

-------------------------------------------
Time: 2022-10-22 01:08:00
-------------------------------------------
('#football', 10)
('#messi', 4)
('#ai', 3)
('#bigdata', 1)

-------------------------------------------
Time: 2022-10-22 01:08:05
-------------------------------------------
('#football', 14)
('#messi', 5)
('#ai', 3)
('#bigdata', 1)

-------------------------------------------
Time: 2022-10-22 01:08:10
-------------------------------------------
('#football', 18)
('#messi', 5)
('#ai', 4)
('#bigdata', 2)

-------------------------------------------
Time: 2022-10-22 01:08:15
-------------------------------------------
('#football', 21)
('#messi', 5)
('#ai', 4)
('#bigdata', 2)

-------------------------------------------
Time: 2022-10-22 01:08:20
-------------------------------------------
('#football', 23)
('#messi', 6)
('#ai', 6)
('#bigdata', 2)

-------------------------------------------
Time: 2022-10-22 01:08:20
-------------------------------------------
('ai', 132, '2022-10-22 01:08:20')
('news', 1, '2022-10-22 01:08:20')
('lol', 3, '2022-10-22 01:08:20')

-------------------------------------------
Time: 2022-10-22 01:08:25
-------------------------------------------
('#football', 25)
('#messi', 7)
('#ai', 7)
('#bigdata', 2)

-------------------------------------------
Time: 2022-10-22 01:08:30
-------------------------------------------
('#football', 26)
('#ai', 19)
('#messi', 7)
('#bigdata', 2)
```

```
-------------------------------------------
Time: 2022-10-22 01:08:35
-------------------------------------------
('#football', 26)
('#ai', 20)
('#messi', 8)
('#bigdata', 2)

-------------------------------------------
Time: 2022-10-22 01:08:40
-------------------------------------------
('#football', 26)
('#ai', 20)
('#messi', 8)
('#bigdata', 2)

-------------------------------------------
Time: 2022-10-22 01:08:45
-------------------------------------------
('#football', 26)
('#ai', 21)
('#messi', 8)
('#bigdata', 3)

-------------------------------------------
Time: 2022-10-22 01:08:50
-------------------------------------------
('#football', 26)
('#ai', 21)
('#messi', 8)
('#bigdata', 3)

-------------------------------------------
Time: 2022-10-22 01:08:55
-------------------------------------------
('#football', 27)
('#ai', 22)
('#messi', 8)
('#bigdata', 3)

-------------------------------------------
Time: 2022-10-22 01:09:00
-------------------------------------------
('#football', 27)
('#ai', 22)
('#messi', 8)
('#bigdata', 3)

-------------------------------------------
Time: 2022-10-22 01:09:05
-------------------------------------------
('#football', 30)
('#ai', 22)
('#messi', 8)
('#bigdata', 3)

-------------------------------------------
Time: 2022-10-22 01:09:10
-------------------------------------------
('#football', 30)
('#ai', 22)
('#messi', 8)
('#bigdata', 3)

-------------------------------------------
Time: 2022-10-22 01:09:15
-------------------------------------------
('#football', 30)
('#ai', 22)
('#messi', 8)
('#bigdata', 5)

-------------------------------------------
Time: 2022-10-22 01:09:20
-------------------------------------------
('#football', 30)
('#ai', 22)
('#messi', 9)
('#bigdata', 5)

-------------------------------------------
Time: 2022-10-22 01:09:20
-------------------------------------------
('ai', 195, '2022-10-22 01:09:20')
('news', 2, '2022-10-22 01:09:20')
('data', 2, '2022-10-22 01:09:20')
('lol', 3, '2022-10-22 01:09:20')

-------------------------------------------
Time: 2022-10-22 01:09:25
-------------------------------------------
('#football', 31)
('#ai', 23)
('#messi', 9)
('#bigdata', 5)

-------------------------------------------
Time: 2022-10-22 01:09:30
-------------------------------------------
('#football', 31)
('#ai', 24)
('#messi', 9)
('#bigdata', 6)

-------------------------------------------
Time: 2022-10-22 01:09:35
```

```
-------------------------------------------
('#football', 35)
('#ai', 24)
('#messi', 9)
('#bigdata', 6)

-------------------------------------------
Time: 2022-10-22 01:09:40
-------------------------------------------
('#football', 36)
('#ai', 24)
('#messi', 9)
('#bigdata', 6)

-------------------------------------------
Time: 2022-10-22 01:09:45
-------------------------------------------
('#football', 36)
('#ai', 24)
('#messi', 9)
('#bigdata', 6)

-------------------------------------------
Time: 2022-10-22 01:09:50
-------------------------------------------
('#football', 36)
('#ai', 25)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:09:55
-------------------------------------------
('#football', 36)
('#ai', 26)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:00
-------------------------------------------
('#football', 36)
('#ai', 26)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:05
-------------------------------------------
('#football', 36)
('#ai', 27)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:10
-------------------------------------------
('#football', 36)
('#ai', 28)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:15
-------------------------------------------
('#football', 36)
('#ai', 30)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:20
-------------------------------------------
('#football', 37)
('#ai', 32)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:20
-------------------------------------------
('ai', 200, '2022-10-22 01:10:20')
('news', 3, '2022-10-22 01:10:20')
('lol', 5, '2022-10-22 01:10:20')

-------------------------------------------
Time: 2022-10-22 01:10:25
-------------------------------------------
('#football', 37)
('#ai', 33)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:30
-------------------------------------------
('#football', 37)
('#ai', 33)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:35
-------------------------------------------
('#football', 38)
('#ai', 33)
```

```
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:40
-------------------------------------------
('#football', 38)
('#ai', 33)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:45
-------------------------------------------
('#football', 39)
('#ai', 33)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:50
-------------------------------------------
('#football', 39)
('#ai', 34)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:10:55
-------------------------------------------
('#football', 40)
('#ai', 34)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:11:00
-------------------------------------------
('#football', 40)
('#ai', 36)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:11:05
-------------------------------------------
('#football', 41)
('#ai', 37)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:11:10
-------------------------------------------
('#football', 41)
('#ai', 39)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:11:15
-------------------------------------------
('#football', 42)
('#ai', 40)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:11:20
-------------------------------------------
('#football', 42)
('#ai', 40)
('#messi', 9)
('#bigdata', 7)

-------------------------------------------
Time: 2022-10-22 01:11:20
-------------------------------------------
('ai', 181, '2022-10-22 01:11:20')
('news', 4, '2022-10-22 01:11:20')
('nyc', 4, '2022-10-22 01:11:20')
('data', 3, '2022-10-22 01:11:20')
('lol', 1, '2022-10-22 01:11:20')

-------------------------------------------
Time: 2022-10-22 01:11:25
-------------------------------------------
('#football', 42)
('#ai', 40)
('#messi', 9)
('#bigdata', 8)

-------------------------------------------
Time: 2022-10-22 01:11:30
-------------------------------------------
('#football', 43)
('#ai', 40)
('#messi', 9)
('#bigdata', 8)

-------------------------------------------
Time: 2022-10-22 01:11:35
-------------------------------------------
('#football', 44)
('#ai', 41)
('#messi', 9)
```

```
('#bigdata', 8)

-------------------------------------------
Time: 2022-10-22 01:11:40
-------------------------------------------
('#football', 45)
('#ai', 42)
('#messi', 9)
('#bigdata', 8)

-------------------------------------------
Time: 2022-10-22 01:11:45
-------------------------------------------
('#football', 46)
('#ai', 43)
('#messi', 9)
('#bigdata', 9)

-------------------------------------------
Time: 2022-10-22 01:11:50
-------------------------------------------
('#football', 48)
('#ai', 46)
('#messi', 9)
('#bigdata', 9)

-------------------------------------------
Time: 2022-10-22 01:11:55
-------------------------------------------
('#football', 48)
('#ai', 48)
('#bigdata', 10)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:00
-------------------------------------------
('#football', 49)
('#ai', 49)
('#bigdata', 10)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:05
-------------------------------------------
('#football', 51)
('#ai', 49)
('#bigdata', 10)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:10
-------------------------------------------
('#football', 52)
('#ai', 51)
('#bigdata', 10)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:15
-------------------------------------------
('#football', 53)
('#ai', 51)
('#bigdata', 10)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:20
-------------------------------------------
('#ai', 57)
('#football', 54)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:20
-------------------------------------------
('ai', 189, '2022-10-22 01:12:20')
('nyc', 1, '2022-10-22 01:12:20')
('news', 1, '2022-10-22 01:12:20')
('lol', 8, '2022-10-22 01:12:20')
('data', 3, '2022-10-22 01:12:20')

-------------------------------------------
Time: 2022-10-22 01:12:25
-------------------------------------------
('#ai', 57)
('#football', 54)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:30
-------------------------------------------
('#ai', 58)
('#football', 54)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:35
-------------------------------------------
('#ai', 58)
('#football', 57)
('#bigdata', 12)
('#messi', 9)
```

```
-------------------------------------------
Time: 2022-10-22 01:12:40
-------------------------------------------
('#ai', 59)
('#football', 57)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:45
-------------------------------------------
('#ai', 60)
('#football', 57)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:50
-------------------------------------------
('#ai', 60)
('#football', 58)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:12:55
-------------------------------------------
('#ai', 60)
('#football', 58)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:13:00
-------------------------------------------
('#ai', 60)
('#football', 59)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:13:05
-------------------------------------------
('#ai', 61)
('#football', 60)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:13:10
-------------------------------------------
('#ai', 63)
('#football', 61)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:13:15
-------------------------------------------
('#ai', 65)
('#football', 62)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:13:20
-------------------------------------------
('#ai', 65)
('#football', 63)
('#bigdata', 12)
('#messi', 9)

-------------------------------------------
Time: 2022-10-22 01:13:20
-------------------------------------------
('ai', 211, '2022-10-22 01:13:20')
('news', 4, '2022-10-22 01:13:20')
('nyc', 1, '2022-10-22 01:13:20')
('lol', 5, '2022-10-22 01:13:20')

-------------------------------------------
Time: 2022-10-22 01:13:25
-------------------------------------------
('#ai', 65)
('#football', 64)
('#bigdata', 12)
('#messi', 10)

-------------------------------------------
Time: 2022-10-22 01:13:30
-------------------------------------------
('#ai', 66)
('#football', 65)
('#bigdata', 13)
('#messi', 10)

-------------------------------------------
Time: 2022-10-22 01:13:35
-------------------------------------------
('#ai', 68)
('#football', 66)
('#bigdata', 13)
('#messi', 10)

-------------------------------------------
```

```
Time: 2022-10-22 01:13:40
-------------------------------------------
('#ai', 68)
('#football', 67)
('#bigdata', 13)
('#messi', 10)

-------------------------------------------
Time: 2022-10-22 01:13:45
-------------------------------------------
('#football', 68)
('#ai', 68)
('#bigdata', 13)
('#messi', 10)

-------------------------------------------
Time: 2022-10-22 01:13:50
-------------------------------------------
('#football', 69)
('#ai', 68)
('#bigdata', 13)
('#messi', 10)

-------------------------------------------
Time: 2022-10-22 01:13:55
-------------------------------------------
('#football', 70)
('#ai', 69)
('#bigdata', 13)
('#messi', 10)

-------------------------------------------
Time: 2022-10-22 01:14:00
-------------------------------------------
('#football', 71)
('#ai', 70)
('#bigdata', 13)
('#messi', 10)

-------------------------------------------
Time: 2022-10-22 01:14:05
-------------------------------------------
('#football', 72)
('#ai', 71)
('#bigdata', 13)
('#messi', 11)

-------------------------------------------
Time: 2022-10-22 01:14:10
-------------------------------------------
('#football', 74)
('#ai', 72)
('#bigdata', 14)
('#messi', 11)

-------------------------------------------
Time: 2022-10-22 01:14:15
-------------------------------------------
('#football', 75)
('#ai', 72)
('#bigdata', 14)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:14:20
-------------------------------------------
('#football', 75)
('#ai', 75)
('#bigdata', 16)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:14:20
-------------------------------------------
('ai', 199, '2022-10-22 01:14:20')
('news', 4, '2022-10-22 01:14:20')
('data', 2, '2022-10-22 01:14:20')

-------------------------------------------
Time: 2022-10-22 01:14:25
-------------------------------------------
('#football', 75)
('#ai', 75)
('#bigdata', 16)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:14:30
-------------------------------------------
('#ai', 79)
('#football', 76)
('#bigdata', 17)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:14:35
-------------------------------------------
('#ai', 79)
('#football', 76)
('#bigdata', 17)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:14:40
-------------------------------------------
('#ai', 80)
```

```
('#football', 76)
('#bigdata', 17)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:14:45
-------------------------------------------
('#ai', 80)
('#football', 76)
('#bigdata', 17)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:14:50
-------------------------------------------
('#ai', 82)
('#football', 76)
('#bigdata', 18)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:14:55
-------------------------------------------
('#ai', 83)
('#football', 77)
('#bigdata', 18)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:15:00
-------------------------------------------
('#ai', 86)
('#football', 78)
('#bigdata', 18)
('#messi', 12)

-------------------------------------------
Time: 2022-10-22 01:15:05
-------------------------------------------
('#ai', 88)
('#football', 78)
('#bigdata', 18)
('#messi', 13)

-------------------------------------------
Time: 2022-10-22 01:15:10
-------------------------------------------
('#ai', 91)
('#football', 79)
('#bigdata', 18)
('#messi', 14)

-------------------------------------------
Time: 2022-10-22 01:15:15
-------------------------------------------
('#ai', 92)
('#football', 81)
('#bigdata', 19)
('#messi', 14)

-------------------------------------------
Time: 2022-10-22 01:15:20
-------------------------------------------
('#ai', 93)
('#football', 81)
('#bigdata', 19)
('#messi', 15)

-------------------------------------------
Time: 2022-10-22 01:15:20
-------------------------------------------
('ai', 205, '2022-10-22 01:15:20')
('news', 2, '2022-10-22 01:15:20')
('lol', 5, '2022-10-22 01:15:20')
('data', 1, '2022-10-22 01:15:20')

-------------------------------------------
Time: 2022-10-22 01:15:25
-------------------------------------------
('#ai', 93)
('#football', 81)
('#bigdata', 19)
('#messi', 15)

-------------------------------------------
Time: 2022-10-22 01:15:30
-------------------------------------------
('#ai', 96)
('#football', 81)
('#bigdata', 19)
('#messi', 15)

-------------------------------------------
Time: 2022-10-22 01:15:35
-------------------------------------------
('#ai', 97)
('#football', 81)
('#bigdata', 19)
('#messi', 17)

-------------------------------------------
Time: 2022-10-22 01:15:40
-------------------------------------------
('#ai', 98)
('#football', 81)
('#bigdata', 19)
```

```
('#messi', 17)

-------------------------------------------
Time: 2022-10-22 01:15:45
-------------------------------------------
('#ai', 101)
('#football', 82)
('#bigdata', 19)
('#messi', 17)

-------------------------------------------
Time: 2022-10-22 01:15:50
-------------------------------------------
('#ai', 102)
('#football', 82)
('#bigdata', 19)
('#messi', 18)

-------------------------------------------
Time: 2022-10-22 01:15:55
-------------------------------------------
('#ai', 103)
('#football', 82)
('#bigdata', 19)
('#messi', 18)

-------------------------------------------
Time: 2022-10-22 01:16:00
-------------------------------------------
('#ai', 103)
('#football', 82)
('#messi', 19)
('#bigdata', 19)

-------------------------------------------
Time: 2022-10-22 01:16:05
-------------------------------------------
('#ai', 106)
('#football', 83)
('#messi', 19)
('#bigdata', 19)

-------------------------------------------
Time: 2022-10-22 01:16:10
-------------------------------------------
('#ai', 107)
('#football', 84)
('#messi', 19)
('#bigdata', 19)

-------------------------------------------
Time: 2022-10-22 01:16:15
-------------------------------------------
('#ai', 107)
('#football', 86)
('#messi', 20)
('#bigdata', 19)

-------------------------------------------
Time: 2022-10-22 01:16:20
-------------------------------------------
('#ai', 110)
('#football', 86)
('#messi', 20)
('#bigdata', 19)

-------------------------------------------
Time: 2022-10-22 01:16:20
-------------------------------------------
('ai', 176, '2022-10-22 01:16:20')
('news', 5, '2022-10-22 01:16:20')
('lol', 1, '2022-10-22 01:16:20')

-------------------------------------------
Time: 2022-10-22 01:16:25
-------------------------------------------
('#ai', 112)
('#football', 86)
('#messi', 21)
('#bigdata', 19)

-------------------------------------------
Time: 2022-10-22 01:16:30
-------------------------------------------
('#ai', 113)
('#football', 86)
('#messi', 21)
('#bigdata', 20)

-------------------------------------------
Time: 2022-10-22 01:16:35
-------------------------------------------
('#ai', 114)
('#football', 86)
('#messi', 21)
('#bigdata', 20)

-------------------------------------------
Time: 2022-10-22 01:16:40
-------------------------------------------
('#ai', 114)
('#football', 86)
('#messi', 21)
('#bigdata', 20)

-------------------------------------------
```

```
Time: 2022-10-22 01:16:45
-------------------------------------------
('#ai', 115)
('#football', 86)
('#messi', 21)
('#bigdata', 20)

-------------------------------------------
Time: 2022-10-22 01:16:50
-------------------------------------------
('#ai', 115)
('#football', 86)
('#messi', 21)
('#bigdata', 20)

-------------------------------------------
Time: 2022-10-22 01:16:55
-------------------------------------------
('#ai', 116)
('#football', 88)
('#messi', 21)
('#bigdata', 20)

-------------------------------------------
Time: 2022-10-22 01:17:00
-------------------------------------------
('#ai', 116)
('#football', 88)
('#messi', 21)
('#bigdata', 20)

-------------------------------------------
Time: 2022-10-22 01:17:05
-------------------------------------------
('#ai', 116)
('#football', 88)
('#messi', 21)
('#bigdata', 20)

-------------------------------------------
Time: 2022-10-22 01:17:10
-------------------------------------------
('#ai', 116)
('#football', 88)
('#messi', 21)
('#bigdata', 21)

-------------------------------------------
Time: 2022-10-22 01:17:15
-------------------------------------------
('#ai', 120)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:17:20
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:17:20
-------------------------------------------
('ai', 199, '2022-10-22 01:17:20')
('news', 1, '2022-10-22 01:17:20')
('lol', 4, '2022-10-22 01:17:20')
('data', 1, '2022-10-22 01:17:20')

-------------------------------------------
Time: 2022-10-22 01:17:25
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:17:30
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:17:35
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:17:40
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:17:45
-------------------------------------------
```

```
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:17:50
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:17:55
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:18:00
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:18:05
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:18:10
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:18:15
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:18:20
-------------------------------------------
('#ai', 122)
('#football', 88)
('#messi', 22)
('#bigdata', 22)

-------------------------------------------
Time: 2022-10-22 01:18:20
-------------------------------------------

STREAMING END
DONE SAVING
```

In [ ]:

In [1]:
```
from pyspark import SparkConf, SparkContext, SQLContext
from pyspark.sql import SparkSession
from pyspark.ml.feature import Word2Vec, CountVectorizer
from pyspark.ml.clustering import LDA, LDAModel
from pyspark.sql.functions import col, udf
from pyspark.sql.types import IntegerType, ArrayType, StringType
import pylab as pl
```

In [2]:
```
def to_word(termIndices):
    words = []
    for termID in termIndices:
        words.append(vocab_broadcast.value[termID])
    return words
```

In [3]:
```
# Load document dataframe (provided by the TA)
PATH = "gs://6893_course_data/twitter_data/stream_data.csv"
spark = SparkSession.builder.appName("LDA").getOrCreate()
spark_df = spark.read.csv(PATH)

spark_df.show()
```

```
+--------------------+
|                 _c0|
+--------------------+
|I absolutely ADOR...|
|Java Vs Python Fo...|
|voulu un grec pui...|
|Pareil Il pris de...|
|Music Academy Blo...|
|Tarps, tents, and...|
|voulu un grec pui...|
|We drive efficien...|
|Check out my Gig ...|
|Hey, nice bones y...|
|lembro como sofri...|
|WHO WITH A DEEP T...|
|@Tina69911364 @As...|
|alguem cria um ap...|
|@Neptvn08 Comment...|
|une dinguerie de ...|
|Y a une grosse mo...|
|Je te cache pas q...|
|@JAPANFESS setauk...|
|Femme recherchant...|
+--------------------+
only showing top 20 rows
```

In [4]:
```
# dataframe preprocessing
from pyspark.sql.functions import col, split
spark_df = spark_df.withColumnRenamed('_c0', 'words')
spark_df = spark_df.withColumn("input", split(col("words"),"\s+"))
spark_df.show()
```

```
+--------------------+--------------------+
|               words|               input|
+--------------------+--------------------+
|I absolutely ADOR...|[I, absolutely, A...|
|Java Vs Python Fo...|[Java, Vs, Python...|
|voulu un grec pui...|[voulu, un, grec,...|
|Pareil Il pris de...|[Pareil, Il, pris...|
|Music Academy Blo...|[Music, Academy, ...|
|Tarps, tents, and...|[Tarps,, tents,, ...|
|voulu un grec pui...|[voulu, un, grec,...|
|We drive efficien...|[We, drive, effic...|
|Check out my Gig ...|[Check, out, my, ...|
|Hey, nice bones y...|[Hey,, nice, bone...|
|lembro como sofri...|[lembro, como, so...|
|WHO WITH A DEEP T...|[WHO, WITH, A, DE...|
|@Tina69911364 @As...|[@Tina69911364, @...|
|alguem cria um ap...|[alguem, cria, um...|
|@Neptvn08 Comment...|[@Neptvn08, Comme...|
|une dinguerie de ...|[une, dinguerie, ...|
|Y a une grosse mo...|[Y, a, une, gross...|
|Je te cache pas q...|[Je, te, cache, p...|
|@JAPANFESS setauk...|[@JAPANFESS, seta...|
|Femme recherchant...|[Femme, rechercha...|
+--------------------+--------------------+
only showing top 20 rows
```

In [5]:
```
# CountVectorizer
cv = CountVectorizer(inputCol="input", outputCol="features")
model = cv.fit(spark_df)
cvResult = model.transform(spark_df)
cvResult.show(5)
```

```
+--------------------+--------------------+--------------------+
|               words|               input|            features|
+--------------------+--------------------+--------------------+
|I absolutely ADOR...|[I, absolutely, A...|(4475,[0,9,12,62,...|
|Java Vs Python Fo...|[Java, Vs, Python...|(4475,[241,398,71...|
|voulu un grec pui...|[voulu, un, grec,...|(4475,[8,14,15,55...|
|Pareil Il pris de...|[Pareil, Il, pris...|(4475,[2,13,15,21...|
|Music Academy Blo...|[Music, Academy, ...|(4475,[0,3,4,30,1...|
+--------------------+--------------------+--------------------+
only showing top 5 rows
```

In [6]:
```
# train LDA model, cluster the documents into 10 topics
ldaModel = LDA(featuresCol="features").setK(10).fit(cvResult)
```

In [7]:
```python
transformed = ldaModel.transform(cvResult).select("topicDistribution")
#show the weight of every topic Distribution
transformed.show(truncate=False)
```

```
+-------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
|topicDistribution
|
+-------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
|[0.005186164216874515,0.003960160476691635,0.003956312707033753,0.003934193713513273,0.003934219359069845,0.003934507045586802,0.962
9163772715503,0.004071676643242233 6,0.0039378703542221295,0.004168518212215463]   |
|[0.007799073061056249,0.9418017624877411,0.005949954723848018,0.005916704936338126,0.005916747034640611,0.005917171856976 65,0.008383
900180722291,0.006123461481739278,0.005922236727356218,0.006268987509581555]       |
|[0.9436470323598692,0.005955713934380984 5,0.005949909186700705,0.005916661523791629,0.005916698285244 34,0.005917129790119107,0.00838
2368015919312,0.006123399724134416,0.005922191400707887,0.006268895779132182]      |
|[0.9536550689576917,0.004897957061174042,0.004893177872296634 5,0.004865831164086 37,0.004865866670514 05,0.004866217930864015,0.006894
118395051148,0.005035861912325537,0.004870379093801928,0.005155520942194619]       |
|[0.006057495012731839 4,0.004624384862931298,0.004619891300247162,0.004594043698171925 6,0.00459410613815166,0.004594421039009235,0.95
66949820658037,0.004754626289906503,0.0045983438445779,0.004867705748468703]       |
|[0.008404860455611665,0.006417818674753064,0.006411558651371602,0.006375745815758958,0.006375775518467542,0.006376235276310497,0.366
04517638977097,0.006598539563966586,0.006381690980193671,0.5806125986737956]       |
|[0.9436470323598692,0.005955713934380984 5,0.005949909186700705,0.005916661523791629,0.005916698285244 34,0.005917129790119107,0.00838
2368015919312,0.006123399724134416,0.005922191400707887,0.006268895779132182]      |
|[0.007801270400857256,0.005955939362672 46,0.005950136191427075,0.005916895956584153,0.7620980170657207,0.005917359032440619,0.188044
84003716423,0.006123742248290633 4,0.005922403279368215,0.006269396425474574]      |
|[0.006057977677200088,0.004624389387720662 5,0.004619878618766 66,0.004594052238250576,0.004594230867509453,0.004594433119314479 5,0.95
66942091602015,0.004754686302647543,0.004598367994717345,0.004867774633671651]     |
|[0.007281311095216425,0.005556029560773472,0.005550633049020 38,0.005519588670424111,0.005519645752300187,0.005520022994663279,0.9479
67104804293,0.005712525314545051,0.005524721861486869 5,0.005848416897277269]      |
|[0.00573570248124609,0.004379419980987283 4,0.004375144655380202 4,0.004350697309650789,0.004350728770987631,0.004351045407472334,0.95
89899681443459,0.004502794310345266,0.0043547656720875 5,0.004609733267496938]     |
|[0.004733038936727789,0.003614172954162737 2,0.003610654342886255 4,0.003590475477526231,0.003590504026523226 3,0.003590759097359575,0.
9661563207066061,0.003715944956293797,0.003593829429412116 7,0.003804300072502154 5]|
|[0.008404503425467393,0.006417785925041724,0.006411527617476171,0.006375703612792339,0.006375747350987522,0.006376206899185855,0.009
03998868325186,0.006598503995542425,0.006381665599391005 5,0.9376183668908636]     |
|[0.007803199636703919,0.005955906308488051,0.005950068058683891,0.005916811065958371,0.005916875133244223,0.005917293939035641 5,0.51
21368840098761,0.43821143312845695,0.005922340426761541,0.006269188292791376]      |
|[0.9474273442432625,0.005555882602923386,0.005550473711243357,0.005519464393656279,0.005519500764727213,0.005519895459695997,0.00782
2166560598687,0.005712410802207438,0.005524682614590265,0.00584817884709508]       |
|[0.006059656566684649,0.004624167462850101,0.004619662627856 73,0.004593843735915511,0.00459387488065969,0.004594206496118 98,0.956694
7530799059,0.004754372622805529,0.004598136797790713,0.004867325729412239]         |
|[0.9436366740589319,0.005955999721701497,0.005950187627745641,0.005916929227470806 5,0.005916953467225662,0.005917449632176018,0.0083
90136535726239,0.006123814679523933,0.005922453934881723 5,0.006269401114616579]   |
|[0.9474250026351184,0.005555943097069845,0.005550516003196437,0.005519499118297 14,0.005519530188958094,0.005519917021080304,0.007824
410227391599,0.005712397056635345,0.005524639594754359,0.005848145057498606]       |
|[0.009111281561980 15,0.006957492920547816,0.006950724315800845,0.006911870994411143,0.006911904782800206,0.006912413514777064,0.0097
96179662681438,0.007153390930546373,0.006918321696266507,0.9323764196201885]       |
|[0.9436443127016174,0.005955790992779921 5,0.005949979531867567 6,0.005916728337001588 5,0.005916773076453810 5,0.005917195452316469,0.0
08384455766608265,0.006123486927938401,0.005922261629643798,0.006269015583772888] |
+-------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------+
only showing top 20 rows
```

In [8]:
```python
#The higher ll is, the lower lp is, the better model is.
ll = ldaModel.logLikelihood(cvResult)
lp = ldaModel.logPerplexity(cvResult)
print("ll: ", ll)
print("lp: ", lp)
```

```
ll:  -123598.70801748236
lp:  11.011020758795757
```

In [9]:
```python
# Output topics. Each is a distribution over words (matching word count vectors)
print("Learned topics (as distributions over vocab of " + str(ldaModel.vocabSize())+ " words):")
topics = ldaModel.topicsMatrix()
print(topics)
```

```
Learned topics (as distributions over vocab of 4475 words):
DenseMatrix([[24.1896685 ,  1.18594903,  0.56720227, ...,  0.83336772,
               1.54761506,  0.68031076],
             [ 0.59859061,  1.24843117,  0.69348586, ...,  0.57424243,
               0.56883441, 12.28921633],
             [76.35448255,  0.63714835,  1.31674193, ...,  4.21452495,
               0.54687653,  0.56760941],
             ...,
             [ 0.76212978,  0.63671608,  0.54815412, ...,  0.74086032,
               0.6261614 ,  0.65348755],
             [ 0.54197475,  0.68193242,  0.61783876, ...,  0.6347725 ,
               0.59419805,  0.87816495],
             [ 0.57097721,  0.64350979,  0.7138692 , ...,  0.59503576,
               0.60042012,  0.51342917]])
```

In [ ]:
```python

```