

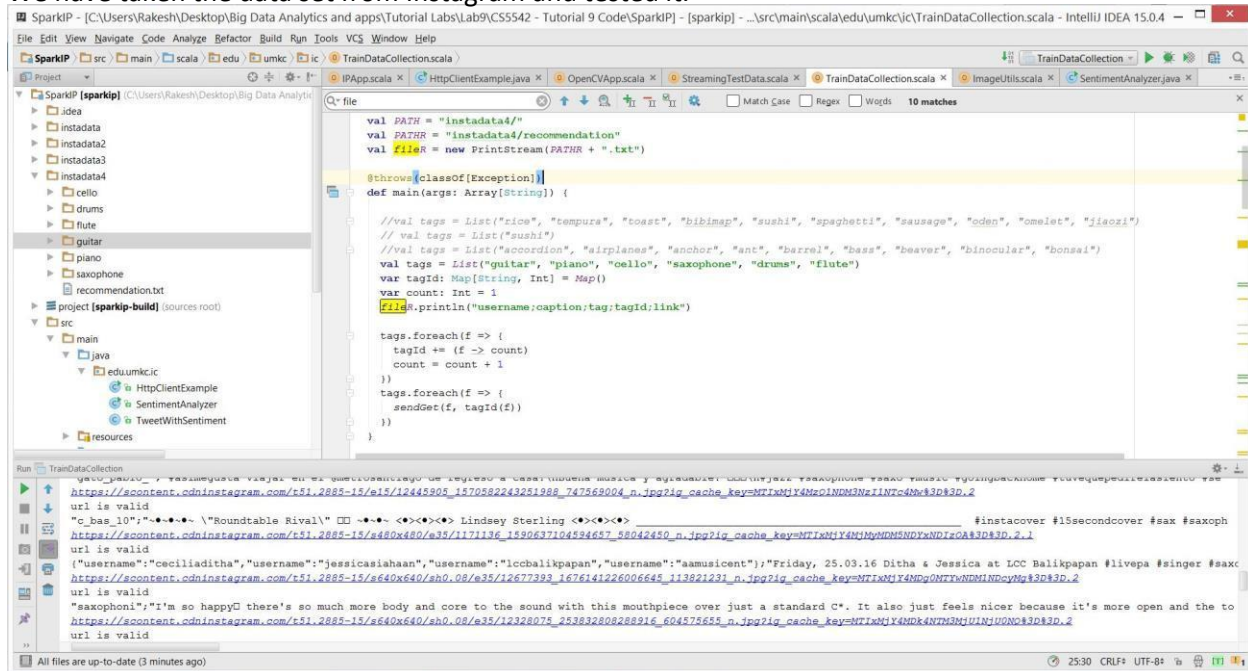
## Lab Assignment 9

## Spark ML Lib (with Instagram streaming and data from smart devices) and Smart Application

**Question 1: Image collection and sentimental analysis based on the image tags using Instagram streaming (related to your project)**

- a. Training Datasets: Instagram Streaming/Categorized Image (e.g., Static UEC Food Dataset) and metadata
- b. Testing Datasets e.g., Image, UserGroup, Category, Rating (Instagram streaming)

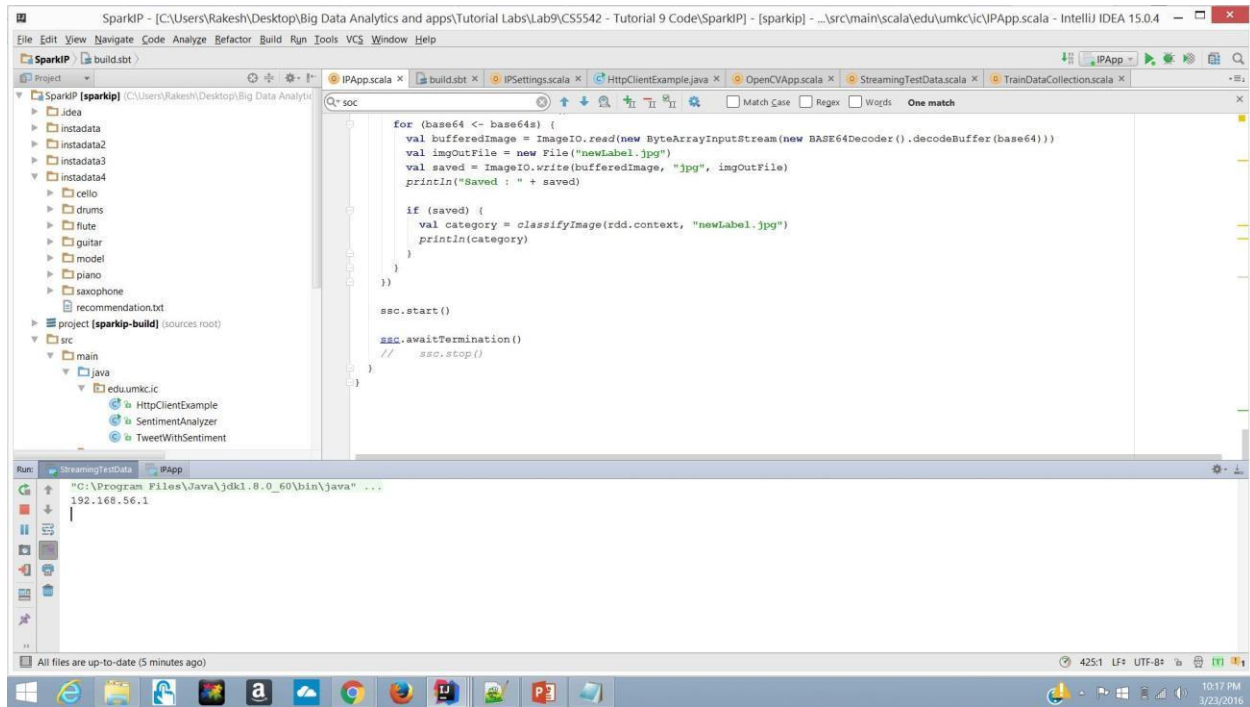
We have taken the data set from Instagram and tested it.



## 2) Image Classification based on the categories related to your project

**Description:**

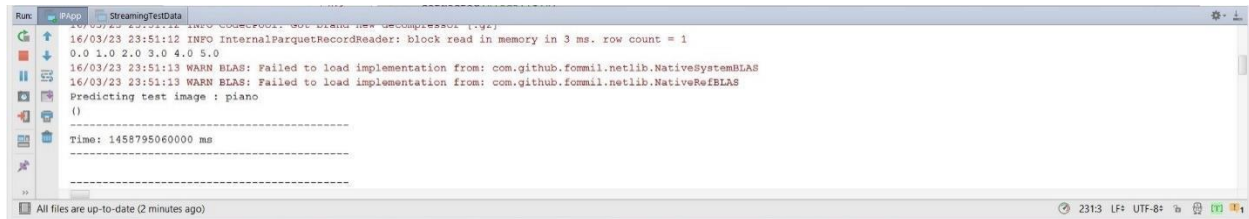
Testing and validation are done by creating a live streaming data to predict live image.



## 2) Image Classification based on the categories related to your project

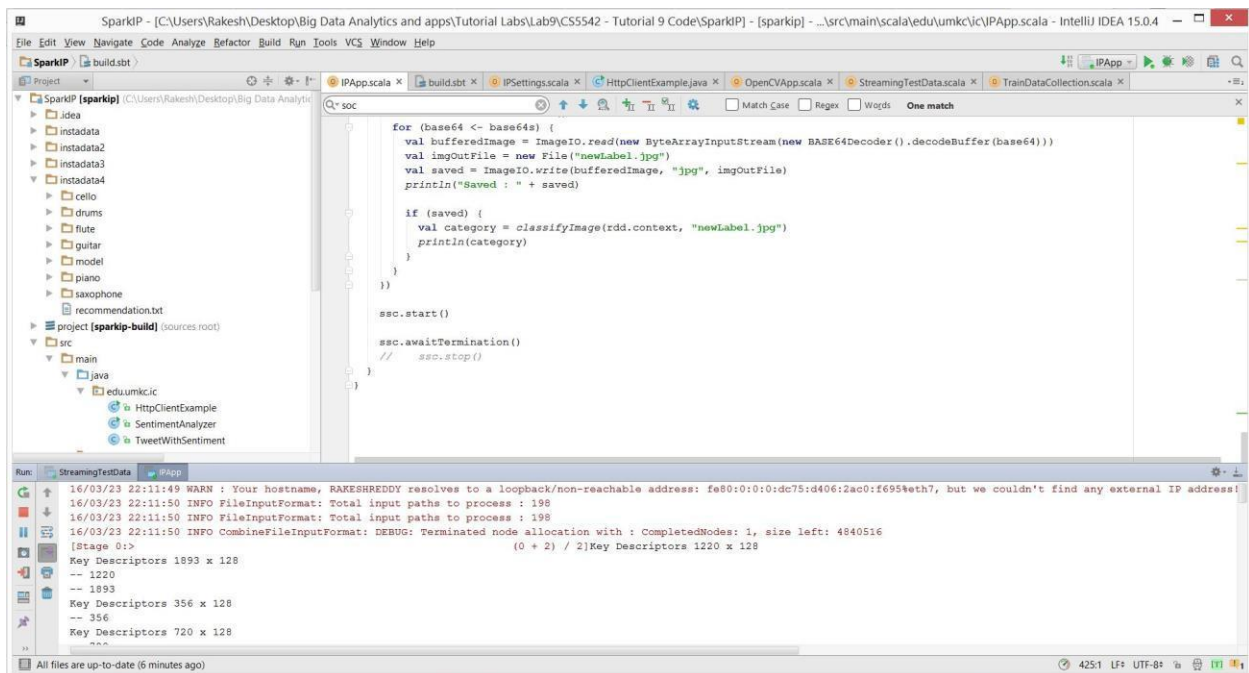
Image classification:

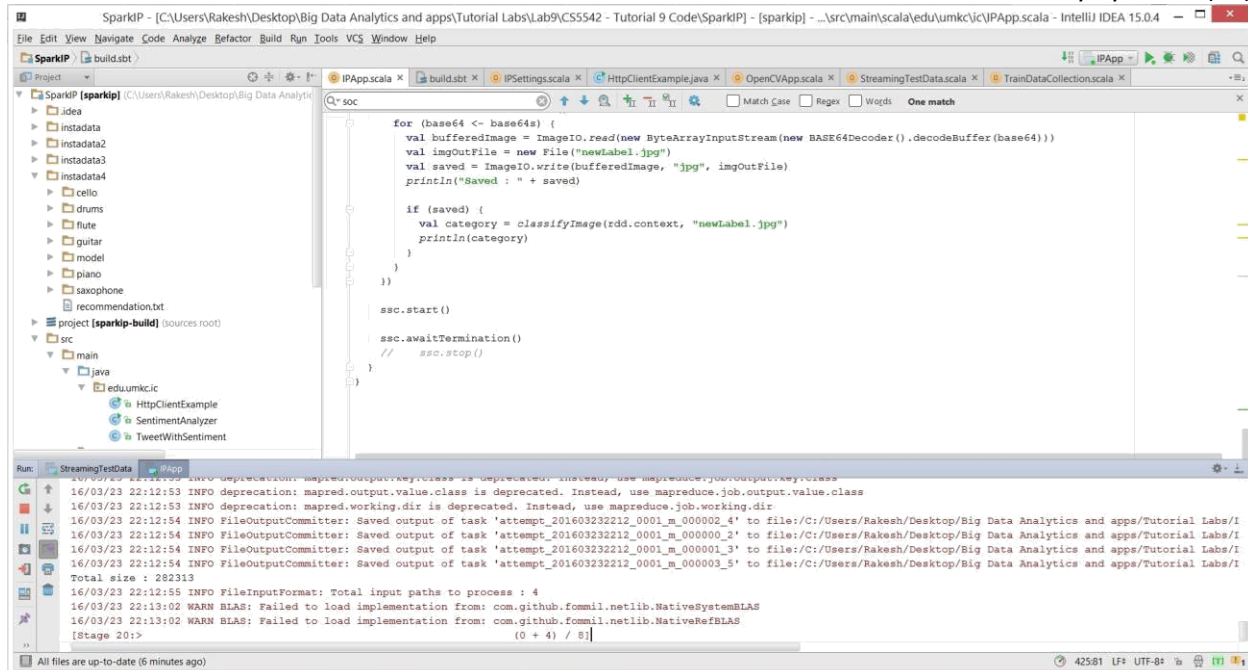
Prediction of the live tag



Steps:

1. key descriptors of the images are obtained.





### 3) Image-based Recommendation system (related to your own project)

- The rating based on sentiment analysis of Instagram metadata
- Expected outcome is to make a recommendation based on user image input or profile (e.g., preferences, location, gender, age)

### 4) Instagram trend notification to smartphone/smartwatch

### 5) Mobile Recommendation through smartphone/smartwatch using your ML application

### 1) Image collection and sentimental analysis based on the image tags using Instagram streaming (related to your project)

- Training Datasets: Instagram Streaming/Categorized Image (e.g., Static UEC Food Dataset) and metadata
- Testing Datasets e.g., Image, UserGroup, Category, Rating (Instagram streaming)

## Lab Assignment 9

Varun Chavakula (4)

Sowmya yelmati (32)

### 5. Displaying for squared errors

```
base64Strings.foreachRDD(rdd => {
    val base64s = rdd.collect()
    for (base64 <- base64s) {
        val bufferedImage = ImageIO.read(new ByteArrayInputStream(new BASE64Decoder().decodeBuffer(base64)))
        val imgOutFile = new File("newLabel.jpg")
        val saved = ImageIO.write(bufferedImage, "jpg", imgOutFile)
        println("Saved : " + saved)

        if (saved) {
            val category = classifyImage(rdd.context, "newLabel.jpg")
            println(category)
        }
    }
})

ssc.start()

ssc.awaitTermination()
// ssc.stop()
```

Run: StreamingTestData - IPApp

16/03/23 22:12:54 INFO FileOutputCommitter: Saved output of task 'attempt\_201603232212\_0001\_m\_000003\_5' to file:/C:/Users/Rakesh/Desktop/Big Data Analytics and apps/Tutorial Labs/1  
Total size : 282313  
16/03/23 22:12:55 INFO FileInputFormat: Total input paths to process : 4  
16/03/23 22:13:02 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS  
16/03/23 22:13:02 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS  
Within Net: Sum of Squared Errors = 1.9041232627465992E10  
16/03/23 22:20:37 INFO FileOutputCommitter: Saved output of task 'attempt\_201603232220\_0057\_m\_000000\_446' to file:/C:/Users/Rakesh/Desktop/Big Data Analytics and apps/Tutorial Labs/1  
16/03/23 22:20:41 WARN TaskSetManager: Stage 58 contains a task of very large size (105 KB). The maximum recommended task size is 100 KB.  
16/03/23 22:20:41 INFO deprecation: mapreduce.outputformat.class is deprecated. Instead, use mapreduce.job.outputformat.class  
16/03/23 22:20:41 INFO CodecConfig: Compression: GZIP  
16/03/23 22:20:41 INFO CodecConfig: Compression: GZIP

All files are up-to-date (11 minutes ago)

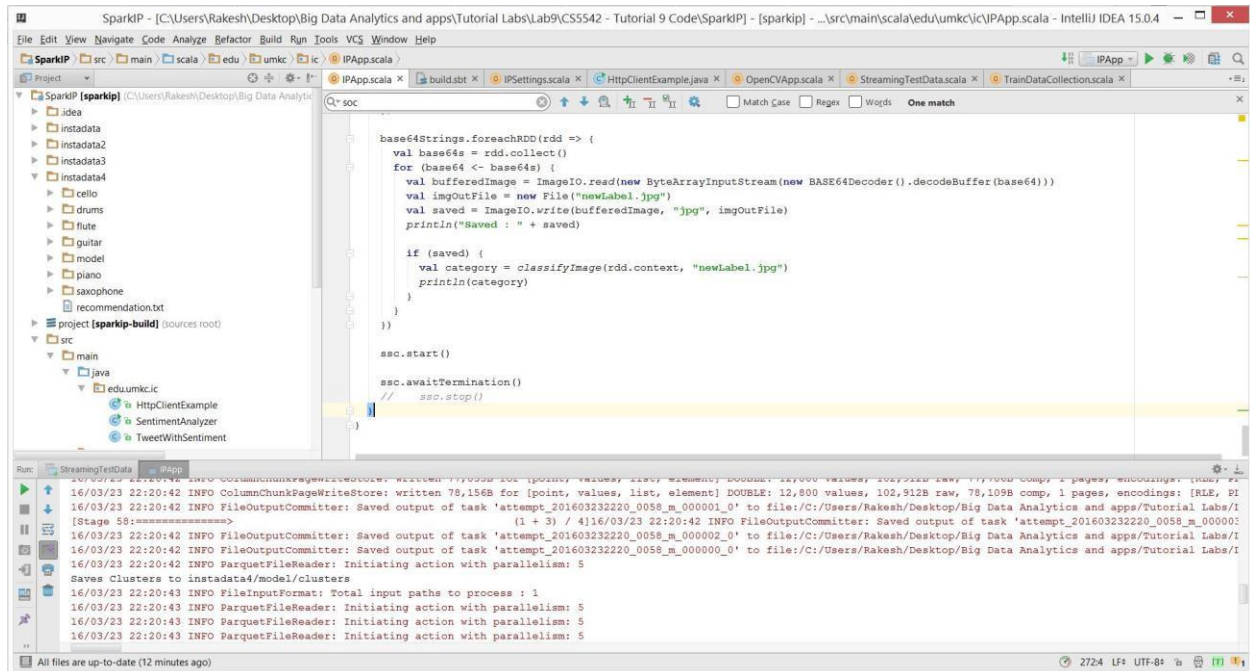


## Lab Assignment 9

Varun Chavakula (4)

Sowmya yelmati (32)

### 6. Clustering the image Bag of Visual words



```
base64Strings.foreachRDD(rdd => {
  val base64s = rdd.collect()
  for (base64 <- base64s) {
    val bufferedImage = ImageIO.read(new ByteArrayInputStream(new BASE64Decoder().decodeBuffer(base64)))
    val imgOutFile = new File("newLabel.jpg")
    val saved = ImageIO.write(bufferedImage, "jpg", imgOutFile)
    println("Saved : " + saved)

    if (saved) {
      val category = classifyImage(rdd.context, "newLabel.jpg")
      println(category)
    }
  }
})

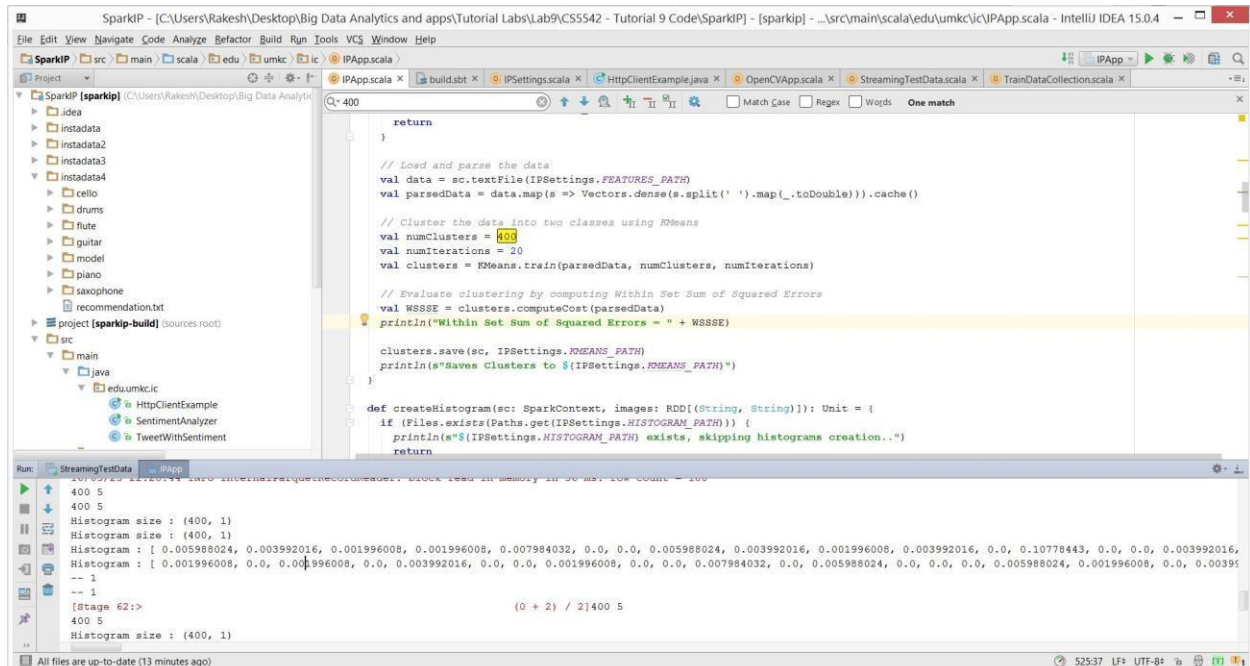
ssc.start()
ssc.awaitTermination()
// ssc.stop()
```

Run: StreamingTestData - 9App

16/03/23 22:20:42 INFO ColumnChunkPageWriteStore: written 78,156B for [point, values, list, element] DOUBLE: 12,800 values, 102,912B raw, 78,109B comp, 1 pages, encodings: [RLE, PI  
16/03/23 22:20:42 INFO FileOutputCommitter: Saved output of task 'attempt\_201603232220\_0058\_m\_000001\_0' to file:/C:/Users/Rakesh/Desktop/Big Data Analytics and apps/Tutorial Labs/I  
[Stage 58:=====]  
16/03/23 22:20:42 INFO FileOutputCommitter: Saved output of task 'attempt\_201603232220\_0058\_m\_000002\_0' to file:/C:/Users/Rakesh/Desktop/Big Data Analytics and apps/Tutorial Labs/I  
16/03/23 22:20:42 INFO FileOutputCommitter: Saved output of task 'attempt\_201603232220\_0058\_m\_000003\_0' to file:/C:/Users/Rakesh/Desktop/Big Data Analytics and apps/Tutorial Labs/I  
16/03/23 22:20:42 INFO ParquetFileReader: Initiating action with parallelism: 5  
Saves clusters to instadata4/model/clusters  
16/03/23 22:20:43 INFO FileInputFormat: Total input paths to process : 1  
16/03/23 22:20:43 INFO ParquetFileReader: Initiating action with parallelism: 5  
16/03/23 22:20:43 INFO ParquetFileReader: Initiating action with parallelism: 5  
16/03/23 22:20:43 INFO ParquetFileReader: Initiating action with parallelism: 5

All files are up-to-date (12 minutes ago)

### 7. Histograms for features of images



```
return
}

// Load and parse the data
val data = sc.textFile(IPSettings.FEATURES_PATH)
val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_toDouble))).cache()

// Cluster the data into two classes using KMeans
val numClusters = 400
val numIterations = 20
val clusters = KMeans.train(parsedData, numClusters, numIterations)

// Evaluate clustering by computing Within Set Sum of Squared Errors
val WSSSE = clusters.computeCost(parsedData)
println("Within Set Sum of Squared Errors = " + WSSSE)

clusters.save(sc, IPSettings.KMEANS_PATH)
println("Saves Clusters to " + IPSettings.KMEANS_PATH)

def createHistogram(sc: SparkContext, images: RDD[(String, String)]): Unit = {
  if (Files.exists(Paths.get(IPSettings.HISTOGRAM_PATH))) {
    println(s"${IPSettings.HISTOGRAM_PATH} exists, skipping histograms creation..")
    return
  }
}
```

Run: StreamingTestData - 9App

400 5  
400 5  
Histogram size : (400, 1)  
Histogram : [ 0.005988024, 0.003992016, 0.001996008, 0.001996008, 0.007984032, 0.0, 0.0, 0.005988024, 0.003992016, 0.001996008, 0.003992016, 0.0, 0.10778443, 0.0, 0.0, 0.003992016,  
Histogram : [ 0.001996008, 0.0, 0.001996008, 0.0, 0.003992016, 0.0, 0.0, 0.001996008, 0.0, 0.0, 0.007984032, 0.0, 0.005988024, 0.0, 0.0, 0.0, 0.005988024, 0.001996008, 0.0, 0.00399  
-- 1  
[Stage 62:-->  
400 5  
Histogram size : (400, 1)  
(0 + 2) / 2] 400 5

All files are up-to-date (13 minutes ago)

## Lab Assignment 9

Varun Chavakula (4)

Sowmya yelmati (32)

### 8. Confusion Matrix Almost we got accuracy of 45%

```
Run: StreamingTestData #App
16/03/23 23:24:51 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
16/03/23 23:24:51 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
===== Confusion matrix =====
0.0 3.0 0.0 6.0 0.0 0.0
0.0 9.0 0.0 1.0 0.0 0.0
0.0 6.0 0.0 1.0 0.0 0.0
0.0 2.0 0.0 14.0 0.0 0.0
0.0 3.0 0.0 4.0 0.0 0.0
0.0 8.0 0.0 4.0 0.0 0.0
16/03/23 23:24:55 INFO FileOutputCommitter: Saved output of task 'attempt_201603232324_0013_m_000000_47' to file:/C:/Users/Rakesh/Desktop/Big Data Analytics and apps/Tutorial Labs/
16/03/23 23:24:55 INFO deprecation: mapreduce.outputformat.class is deprecated. Instead, use mapreduce.job.outputformat.class
```

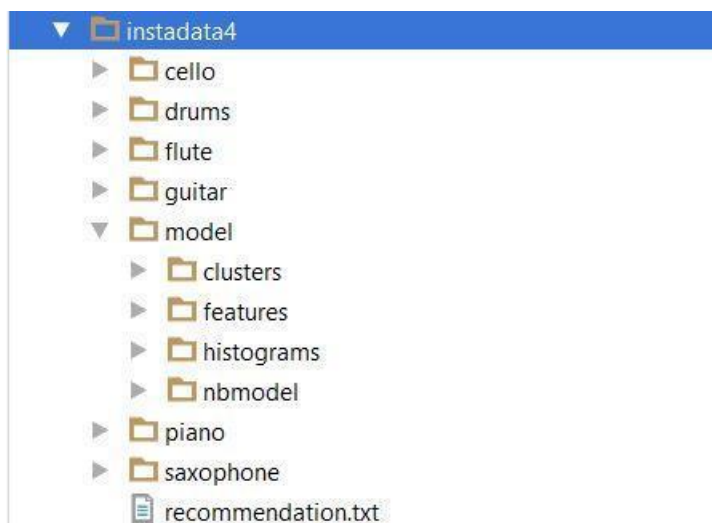
### 9. Built the Naïve Bayes Model

```
Run: StreamingTestData #App
16/03/23 23:24:56 INFO FileOutputCommitter: Saved output of task 'attempt_201603232324_0013_m_000000_47' to file:/C:/Users/Rakesh/Desktop/Big Data Analytics and apps/Tutorial Labs/
16/03/23 23:24:56 INFO ParquetFileReader: Initiating action with parallelism: 5
Naive Bayes Model generated
Time: 1458793498000 ms
-----
16/03/23 23:24:58 WARN ReceiverSupervisorImpl: Restarting receiver with delay 2000 ms: Socket data stream had no more data
16/03/23 23:24:58 ERROR ReceiverTracker: Deregistered receiver for stream 0: Restarting receiver with delay 2000ms: Socket data stream had no more data
16/03/23 23:24:58 WARN BlockManager: Block input-0-1458793498000 replicated to only 0 peer(s) instead of 1 peers
Time: 1458793500000 ms
```

### 10. Predicting the tag for Live Image Data – Piano tag

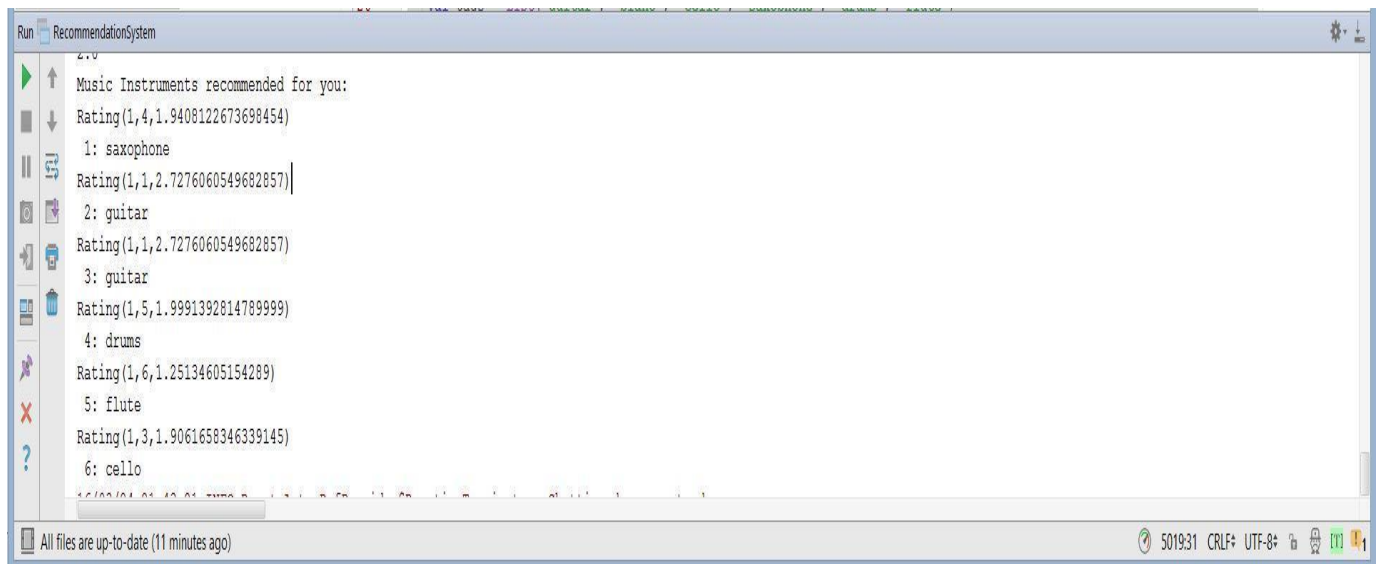
```
Run: StreamingTestData #App
16/03/23 23:25:12 INFO CodeGenerator: Generating new decompressor (r1yz)
16/03/23 23:25:12 INFO InternalParquetRecordReader: block read in memory in 3 ms. row count = 1
0.0 1.0 2.0 3.0 4.0 5.0
16/03/23 23:25:13 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
16/03/23 23:25:13 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
Predicting test image : piano
()
Time: 1458795060000 ms
-----
```

### 11. Trained model stored in Data folder



**3) Image-based Recommendation system (related to your own project)****a. The rating based on sentiment analysis of Instagram metadata****b. Expected outcome is to make a recommendation based on user image input or profile (e.g., preferences, location, gender, age)**

- We have recommended the top Music Instruments for the user.
- We have assigned userId based on the alphabets.
- We have collected recommendation.txt with "username;caption;tag;tagId;link"
- We have used the user preference with UserId,TagId,SentimentRating



```
Run RecommendationSystem
Music Instruments recommended for you:
Rating(1,4,1.9408122673698454)
1: saxophone
Rating(1,1,2.7276060549682857)
2: guitar
Rating(1,1,2.7276060549682857)
3: guitar
Rating(1,5,1.9991392814789999)
4: drums
Rating(1,6,1.25134605154289)
5: flute
Rating(1,3,1.9061658346339145)
6: cello
```

All files are up-to-date (11 minutes ago) 5019:31 CRLF UTF-8