

CS5542 Big Data Apps and Analytics

LAB ASSIGNMENT-9

REPORT and SCREEN SHOTS

By Rakesh Reddy Bandi(2) & Mark J Schultz(26)

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

- 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)**
- 2) Image Classification based on the categories related to your project**
- 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)**
- 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)**
 - 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 collected the image datasets from the Instagram. We have collected the Image data for the hashtags musical instruments.
 - The list of tag categories for musical instruments are List("guitar", "piano", "cello", "saxophone", "drums", "flute")
 - Along with the Training data collection we have also collected the image tags based on user group and their image links and tags in Recommendation.txt file which is later used for the Recommendation. Format "username;caption;tag;tagId;link"
 - Testing Data is live streaming Data from the Instagram.

1. We have collected the image datasets from the Instagram. We have collected the Image data for the hashtags musical instruments. **Data Folder: Instadata4 - List("guitar", "piano", "cello", "saxophone", "drums", "flute")**

```

val PATH = "instadata4/"
val PATHR = "instadata4/recommendation"
val FILER = new PrintStream(PATHR + ".txt")

@throws(classOf[Exception])
def main(args: Array[String]) {
  //val tags = List("rice", "tempura", "toast", "bibimaps", "sushi", "spaghetti", "sausage", "oden", "omelet", "jiaozi")
  //val tags = List("sushi")
  //val tags = List("acordion", "airplanes", "anchor", "ant", "barrel", "bass", "beaver", "binocular", "bonsai")
  val tags = List("guitar", "piano", "cello", "saxophone", "drums", "flute")
  var tagId: Map[String, Int] = Map()
  var count: Int = 1
  FILER.println("username:caption;tag;tagId;link")

  tags.foreach(f => {
    tagId += (f -> count)
    count = count + 1
  })
  tags.foreach(f => {
    sendGec(f, tagId(f))
  })
}
  
```

2. For Testing and Validation of the model we create a live streaming data to predict the current live image for the particular tags. We open a socket to listen the request from the client i.e. our classification model.

```

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)

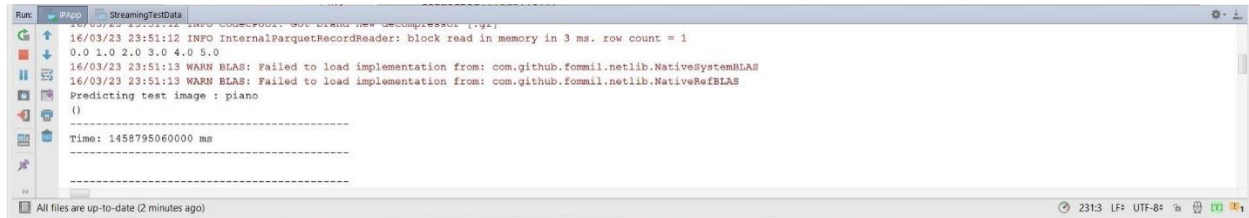
  if (saved) {
    val category = classifyImage(rdd.context, "newLabel.jpg")
    println(category)
  }
}

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

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

Image classification:

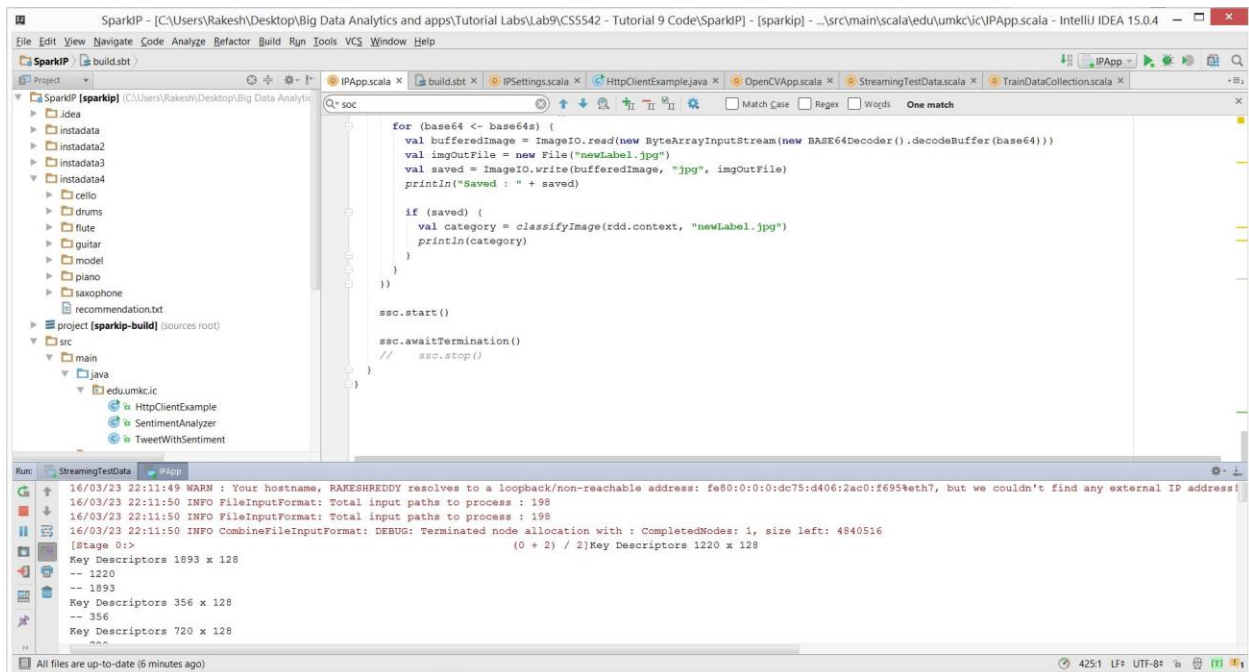
Predicting the tag for Live Image Data



```
16/03/23 23:51:12 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 23:51:13 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
16/03/23 23:51:13 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
Predicting test image : piano
()
Time: 1458795060000 ms
```

Steps:

1. First we get the key Descriptors for the all images

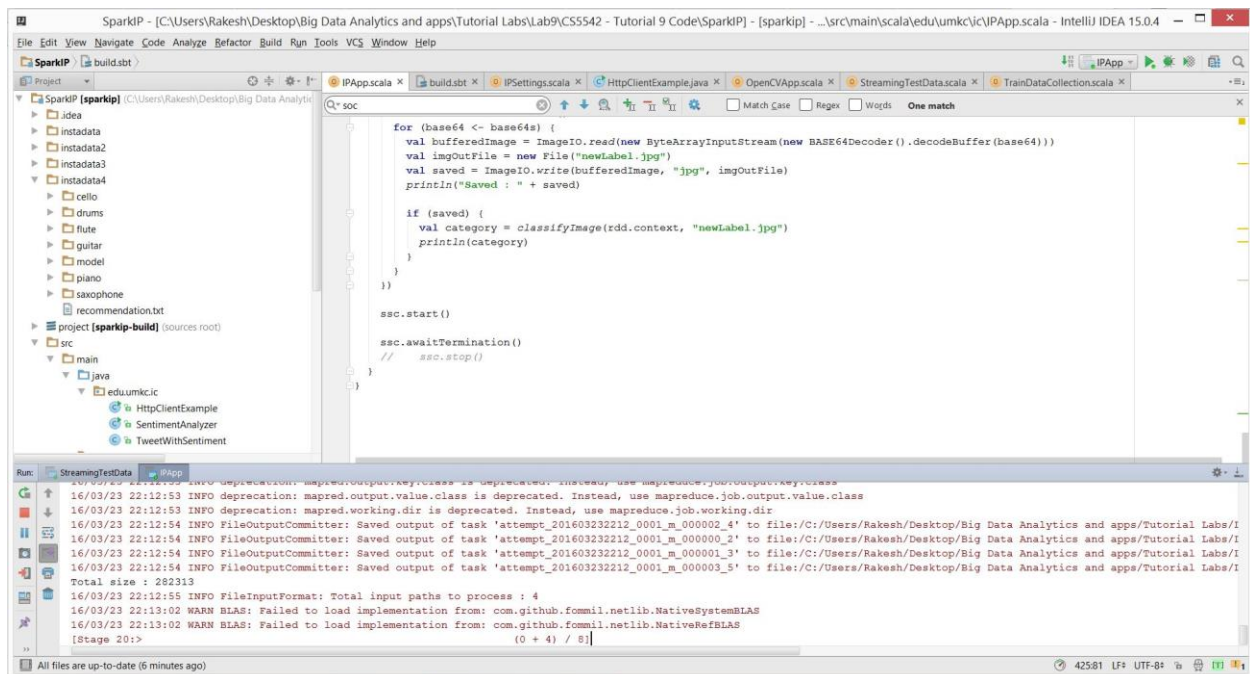


```
for (base64 <- base64s) {
  val bufferedImage = ImageIO.read(new ByteArrayInputStream(new BASE64Decoder().decodeBuffer(base64)))
  val imgOutFile = new File("newLabel.jpg")
  val saved = ImageIO.write(bufferedImage, "jpg", imgOutFile)
  println("Saved : " + saved)

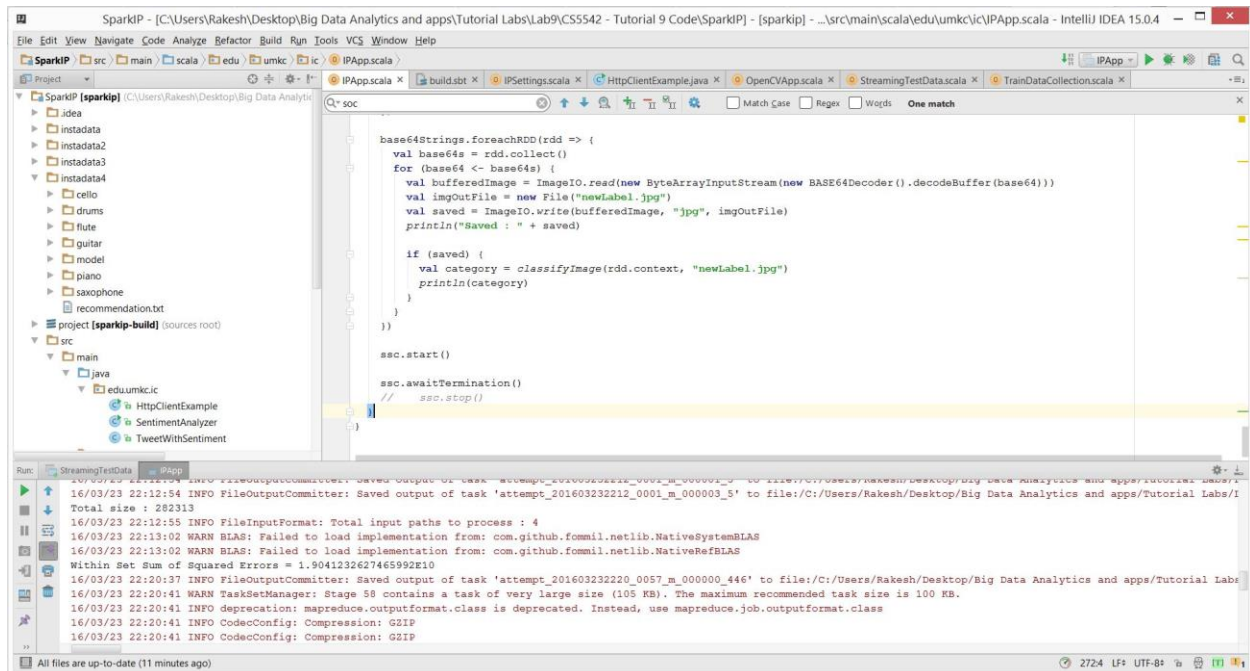
  if (saved) {
    val category = classifyImage(rdd.context, "newLabel.jpg")
    println(category)
  }
}

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

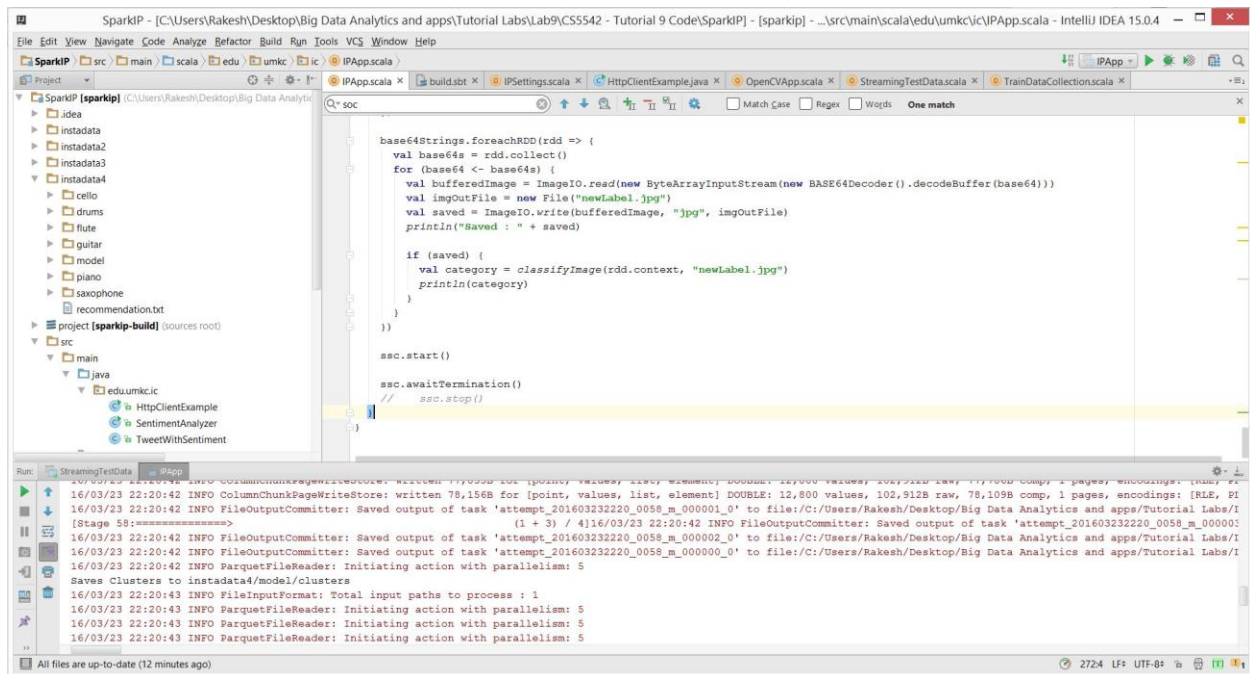
```
16/03/23 22:11:49 WARN : Your hostname, RAKESHREDDY resolves to a loopback/non-reachable address: fe80:0:0:0:dc75:d406:2ac0:f695%eth7, but we couldn't find any external IP address:
16/03/23 22:11:50 INFO FileInputFormat: Total input paths to process : 198
16/03/23 22:11:50 INFO FileInputFormat: Total input paths to process : 198
16/03/23 22:11:50 INFO CombineFileInputFormat: DEBUG: Terminated node allocation with : CompletedNodes: 1, size left: 4840516
(0 + 2) / 2]Key Descriptors 1220 x 128
[Stage 0:]
Key Descriptors 1893 x 128
-- 1220
-- 1893
Key Descriptors 356 x 128
-- 356
Key Descriptors 720 x 128
--
```



5. Displaying for squared errors



6. Clustering the image Bag of Visual words

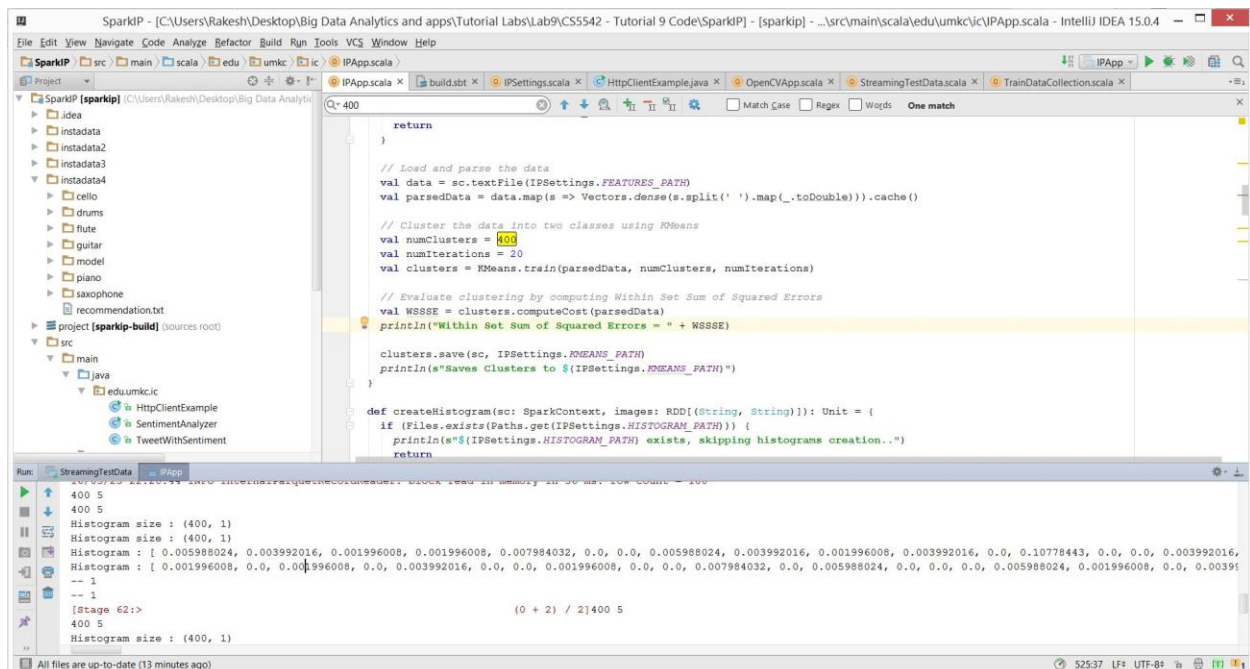


The screenshot shows the IntelliJ IDEA interface with the SparkIP project. The main editor displays the `IPApp.scala` file, which contains the following code:

```
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()
```

The bottom panel shows the Spark console output, which includes logs for `ColumnChunkPageWriteStore`, `FileOutputCommitter`, and `ParquetFileReader`.

7. Histograms for features of images



The screenshot shows the IntelliJ IDEA interface with the SparkIP project. The main editor displays the `IPApp.scala` file, which contains the following code:

```
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  
  }  
}
```

The bottom panel shows the Spark console output, which includes logs for `Histogram` and `Stage`.

8. Confusion Matrix Almost we got accuracy of 45%

```
Run: StreamingTestData #App
16/03/23 23:24:51 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 23:24:51 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
```

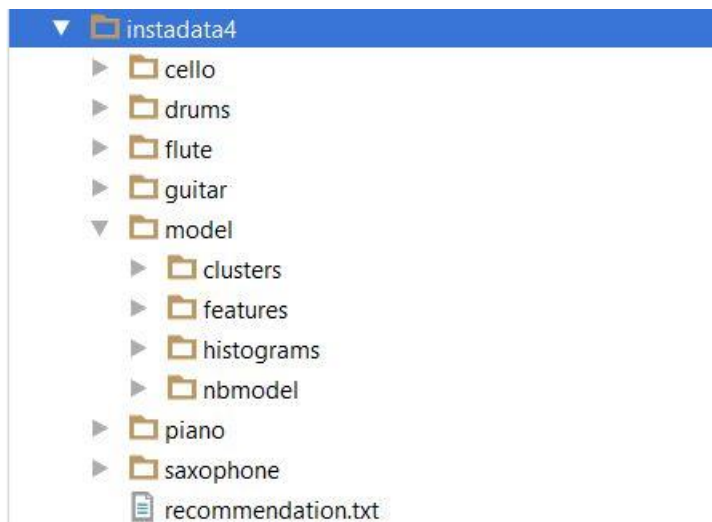
9. Built the Naïve Bayes Model

```
Run: StreamingTestData #App
16/03/23 23:24:56 INFO FileOutputCommitter: Saved output of task 'attempt_201603232324_0013_m_000000_0' 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: #App StreamingTestData
16/03/23 23:51:12 INFO CodecPool: Got brand new decompressor [org]
16/03/23 23:51:12 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 23:51:13 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
16/03/23 23:51:13 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

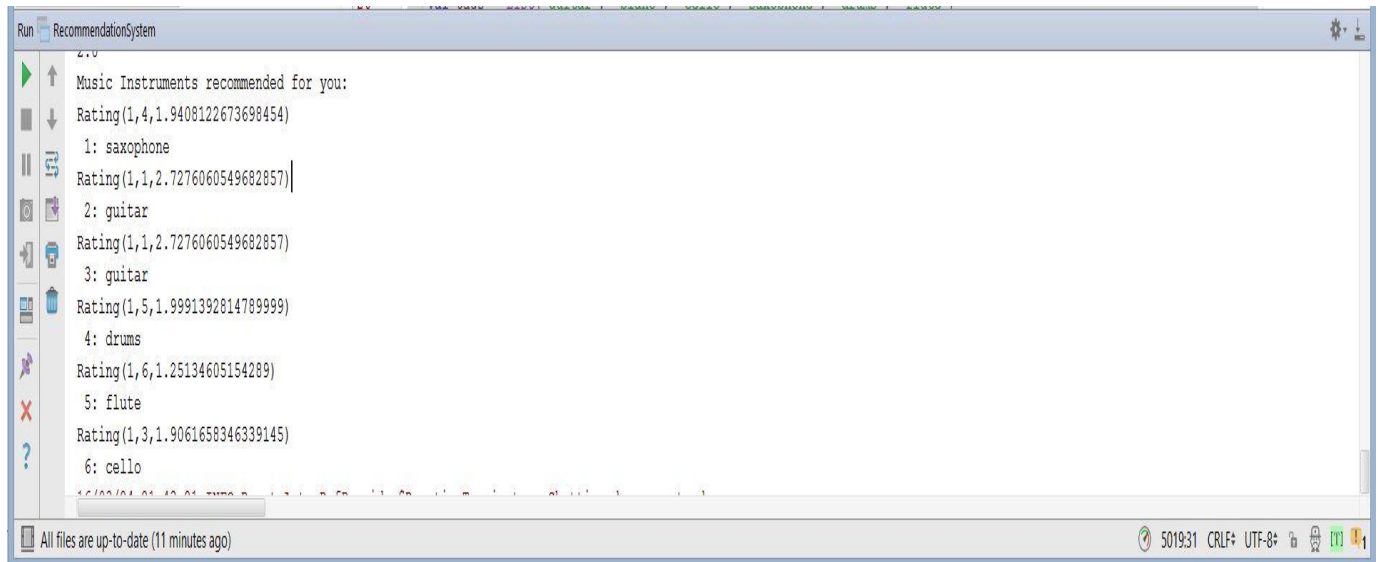


3) Image-based Recommendation system (related to your own project)

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
2.0
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+