# Lab-9

## A-1

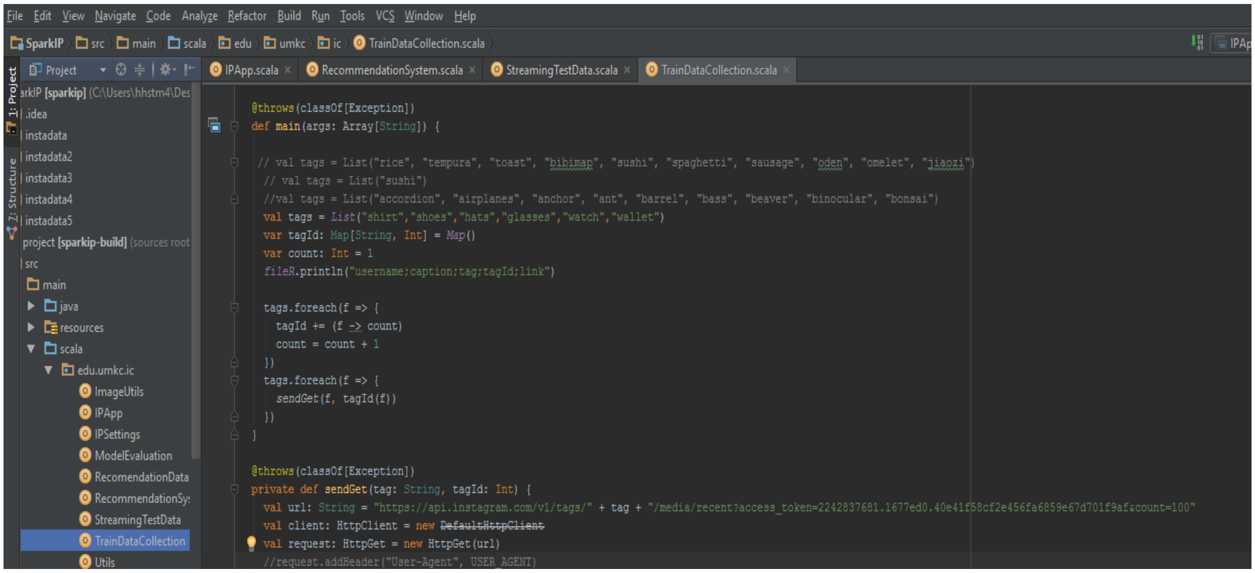
**1) Image collection and sentimental analysis based on the image tags using Instagram streaming (related to your project)**
**a. Training Datasets: Instagram Streaming/Categorized Image (e.g., Static UEC Food Dataset) and metadata**
**b. Testing Datasets e.g., Image, UserGroup, Category, Rating (Instagram streaming)**

- We have collected the image datasets from the Instagram. We have collected the Image data for the hashtags body wearing cosmetics.

- The list of tag categories are List("shirt","shoes","hats","glasses","watch","wallet")

- 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"

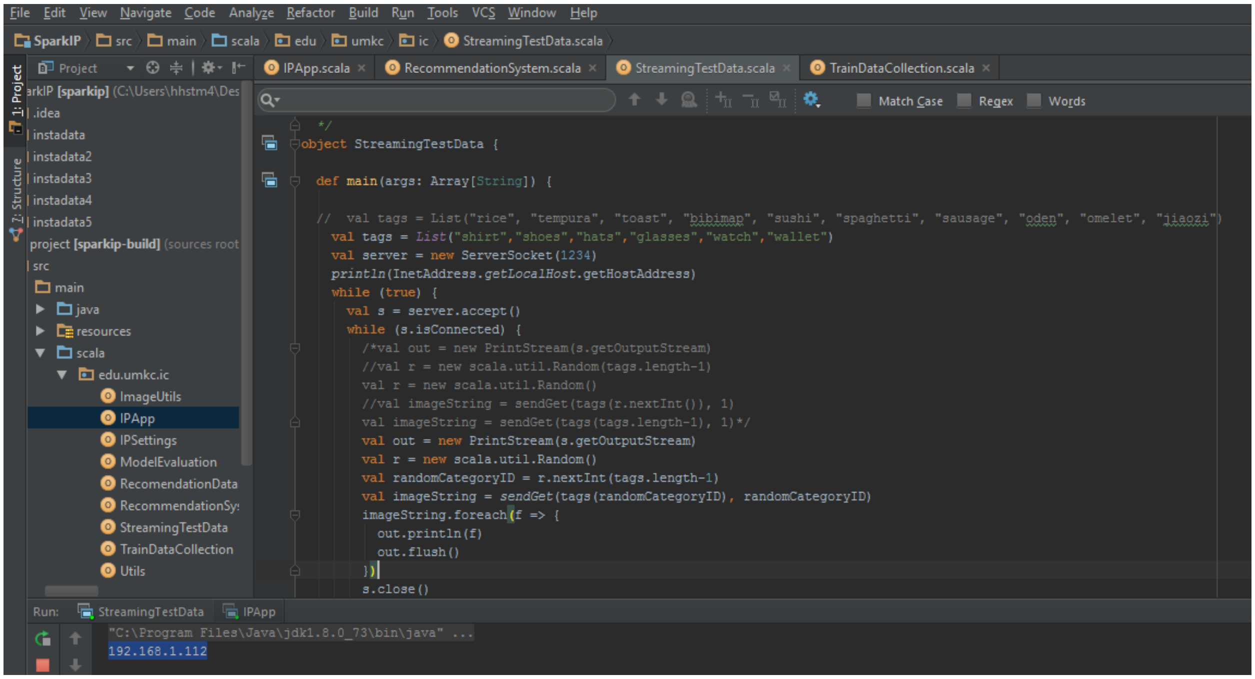
1. 1. We have collected the image datasets from the Instagram. We have collected the Image data for the hashtags.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SparkIP › src › main › scala › edu › umkc › ic › TrainDataCollection.scala
IPAp
Project
IPApp.scala ×
RecommendationSystem.scala ×
StreamingTestData.scala ×
TrainDataCollection.scala ×
1: Project
7: Structure

```
arkIP [sparkip] (C:\Users\hhstm4\Des
 .idea
 instadata
 instadata2
 instadata3
 instadata4
 instadata5
project [sparkip-build] (sources root
 src
  main
   java
   resources
   scala
    edu.umkc.ic
     ImageUtils
     IPApp
     IPSettings
     ModelEvaluation
     RecomendationData
     RecommendationSy:
     StreamingTestData
     TrainDataCollection
     Utils
```

```
@throws(classOf[Exception])
def main(args: Array[String]) {

 // val tags = List("rice", "tempura", "toast", "bibimap", "sushi", "spaghetti", "sausage", "oden", "omelet", "jiaozi")
  // val tags = List("sushi")
  //val tags = List("accordion", "airplanes", "anchor", "ant", "barrel", "bass", "beaver", "binocular", "bonsai")
  val tags = List("shirt","shoes","hats","glasses","watch","wallet")
  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 => {
    sendGet(f, tagId(f))
  })
}

@throws(classOf[Exception])
private def sendGet(tag: String, tagId: Int) {
  val url: String = "https://api.instagram.com/v1/tags/" + tag + "/media/recent?access_token=2242837681.1677ed0.40e41f58cf2e456fa6859e67d701f9af&count=100"
  val client: HttpClient = new ~~DefaultHttpClient~~
  val request: HttpGet = new HttpGet(url)
  //request.addHeader("User-Agent", USER_AGENT)
```

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.



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SparkIP › src › main › scala › edu › umkc › ic › StreamingTestData.scala
IPApp.scala
RecommendationSystem.scala
StreamingTestData.scala
TrainDataCollection.scala
Match Case
Regex
Words
Project
SparkIP [sparkip] (C:\Users\hhstm4\Des
.idea
instadata
instadata2
instadata3
instadata4
instadata5
project [sparkip-build] (sources root
src
main
java
resources
scala
edu.umkc.ic
ImageUtils
IPApp
IPSettings
ModelEvaluation
RecomendationData
RecommendationSy:
StreamingTestData
TrainDataCollection
Utils
Run: StreamingTestData IPApp

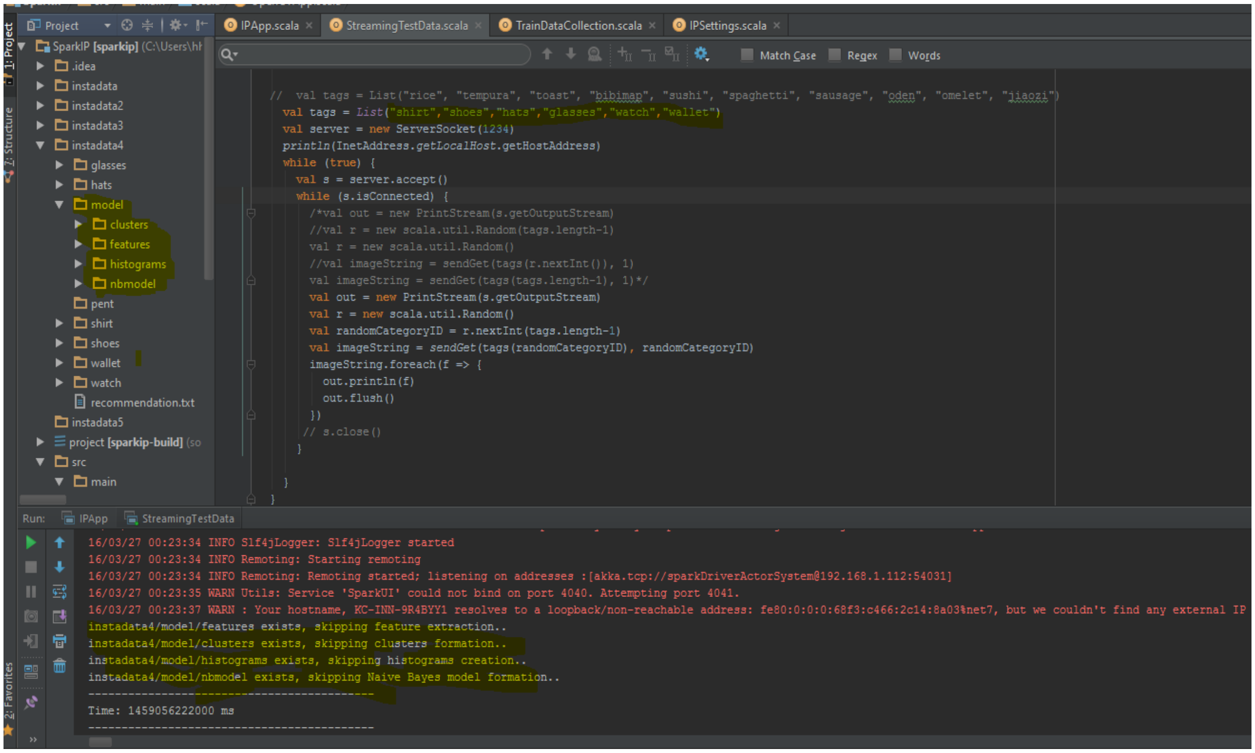
```
 */
object StreamingTestData {

  def main(args: Array[String]) {

  //  val tags = List("rice", "tempura", "toast", "bibimap", "sushi", "spaghetti", "sausage", "oden", "omelet", "jiaozi")
    val tags = List("shirt","shoes","hats","glasses","watch","wallet")
    val server = new ServerSocket(1234)
    println(InetAddress.getLocalHost.getHostAddress)
    while (true) {
      val s = server.accept()
      while (s.isConnected) {
        /*val out = new PrintStream(s.getOutputStream)
        //val r = new scala.util.Random(tags.length-1)
        val r = new scala.util.Random()
        //val imageString = sendGet(tags(r.nextInt()), 1)
        val imageString = sendGet(tags(tags.length-1), 1)*/
        val out = new PrintStream(s.getOutputStream)
        val r = new scala.util.Random()
        val randomCategoryID = r.nextInt(tags.length-1)
        val imageString = sendGet(tags(randomCategoryID), randomCategoryID)
        imageString.foreach(f => {
          out.println(f)
          out.flush()
        })
        s.close()
```

```
"C:\Program Files\Java\jdk1.8.0_73\bin\java" ...
192.168.1.112
```

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

Image classification:
Predicting the tag for Live Image Data

First, we created the model for training data as shown in below figure.



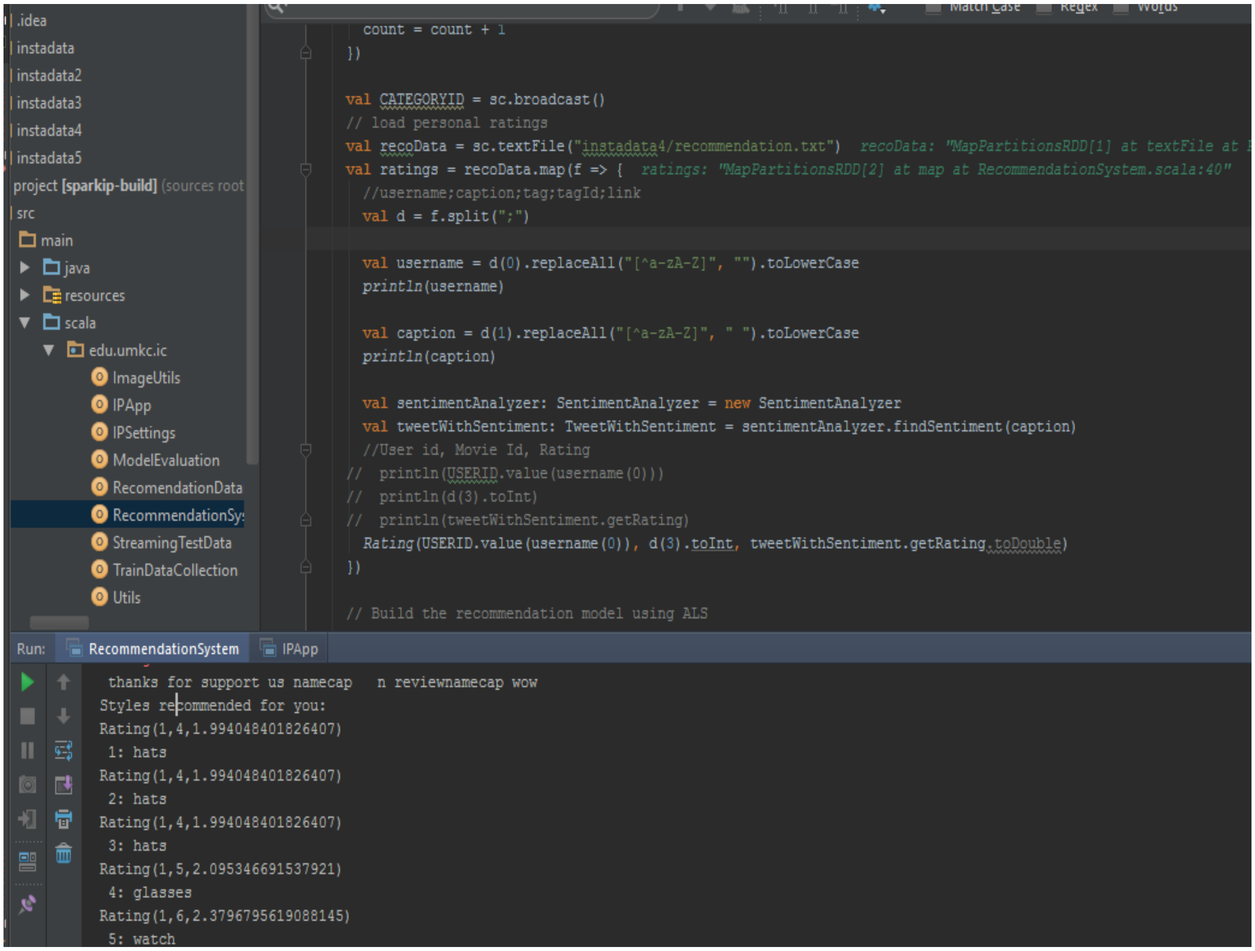
```
//  val tags = List("rice", "tempura", "toast", "bibimap", "sushi", "spaghetti", "sausage", "oden", "omelet", "jiaozi")
  val tags = List("shirt","shoes","hats","glasses","watch","wallet")
  val server = new ServerSocket(1234)
  println(InetAddress.getLocalHost.getHostAddress)
  while (true) {
    val s = server.accept()
    while (s.isConnected) {
      /*val out = new PrintStream(s.getOutputStream)
      //val r = new scala.util.Random(tags.length-1)
      val r = new scala.util.Random()
      //val imageString = sendGet(tags(r.nextInt()), 1)
      val imageString = sendGet(tags(tags.length-1), 1)*/
      val out = new PrintStream(s.getOutputStream)
      val r = new scala.util.Random()
      val randomCategoryID = r.nextInt(tags.length-1)
      val imageString = sendGet(tags(randomCategoryID), randomCategoryID)
      imageString.foreach(f => {
        out.println(f)
        out.flush()
      })
     // s.close()
    }

  }
}
```

```
16/03/27 00:23:34 INFO Slf4jLogger: Slf4jLogger started
16/03/27 00:23:34 INFO Remoting: Starting remoting
16/03/27 00:23:34 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriverActorSystem@192.168.1.112:54031]
16/03/27 00:23:35 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
16/03/27 00:23:37 WARN : Your hostname, KC-INN-9R4BYY1 resolves to a loopback/non-reachable address: fe80:0:0:0:68f3:c466:2c14:8a03%net7, but we couldn't find any external IP
instadata4/model/features exists, skipping feature extraction..
instadata4/model/clusters exists, skipping clusters formation..
instadata4/model/histograms exists, skipping histograms creation..
instadata4/model/nbmodel exists, skipping Naive Bayes model formation..
-------------------------------------------
Time: 1459056222000 ms
-------------------------------------------
```

Then we have run the same program to predict the upcoming stream to classify it into the category.

**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 wearing style 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



```
      count = count + 1
    })

    val CATEGORYID = sc.broadcast()
    // load personal ratings
    val recoData = sc.textFile("instadata4/recommendation.txt")  recoData: "MapPartitionsRDD[1] at textFile at 
    val ratings = recoData.map(f => {  ratings: "MapPartitionsRDD[2] at map at RecommendationSystem.scala:40"
      //username;caption;tag;tagId;link
      val d = f.split(";")

      val username = d(0).replaceAll("[^a-zA-Z]", "").toLowerCase
      println(username)

      val caption = d(1).replaceAll("[^a-zA-Z]", " ").toLowerCase
      println(caption)

      val sentimentAnalyzer: SentimentAnalyzer = new SentimentAnalyzer
      val tweetWithSentiment: TweetWithSentiment = sentimentAnalyzer.findSentiment(caption)
      //User id, Movie Id, Rating
    //  println(USERID.value(username(0)))
    //  println(d(3).toInt)
    //  println(tweetWithSentiment.getRating)
      Rating(USERID.value(username(0)), d(3).toInt, tweetWithSentiment.getRating.toDouble)
    })

    // Build the recommendation model using ALS
```

```
 thanks for support us namecap   n reviewnamecap wow
Styles recommended for you:
Rating(1,4,1.994048401826407)
 1: hats
Rating(1,4,1.994048401826407)
 2: hats
Rating(1,4,1.994048401826407)
 3: hats
Rating(1,5,2.095346691537921)
 4: glasses
Rating(1,6,2.3796795619088145)
 5: watch
```