Big Data Programming Assignment 8

Varaprasad Kurra **Panther ID - 002430487**

Source Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.StringTokenizer;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;
import org.apache.spark.broadcast.Broadcast;
import org.apache.spark.ml.classification.NaiveBayes;
import org.apache.spark.ml.classification.NaiveBayesModel;
import org.apache.spark.ml.feature.CountVectorizer;
import org.apache.spark.ml.feature.CountVectorizerModel;
import org.apache.spark.ml.feature.StopWordsRemover;
import org.apache.spark.ml.feature.Tokenizer;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.RowFactory;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.types.DataTypes;
import org.apache.spark.sql.types.StructField;
import org.apache.spark.sql.types.StructType;
import scala.Tuple2;
import shapeless.Tuple;
import scala.Tuple1;

public class NaiveBayesClassification {

	private static final String LABEL_SEPARATOR = "|";
	private static final String TRAINING_URI   =
"file:///C:/Users/VaraPrasad/Desktop/Summer_Semester/NB_Training_Files";
	private static final String CATEGORIES     =
"C:/Users/VaraPrasad/Desktop/Summer_Semester/NB_Categories.txt";
	private static final String TESTING_URI    =
"file:///C:/Users/VaraPrasad/Desktop/Summer_Semester/NB_Testing_Files";

	public static void main(String[] args) throws IOException
	{
//---------------------------------------------------------------------------
-------------------------------
		// initializing spark
		SparkSession spark =
SparkSession.builder().config("spark.master","local[*]").getOrCreate();
		JavaSparkContext sc = new JavaSparkContext(spark.sparkContext());
		sc.setLogLevel("WARN");

		// read the categories file that maps text categories to
numerical ones
		HashMap<String, Integer> categories = getCategoryMap(CATEGORIES);
```

```
            Broadcast<HashMap> allCategories = sc.broadcast(categories);
            System.out.println("Hash Values of the allCatogries are");
            System.out.println(allCategories.getValue());

            // read the training documents
            JavaPairRDD<String,String> documents =
sc.wholeTextFiles(TRAINING_URI);

      System.out.println(documents.take((int)documents.count()).toString());

            //read the testing documents
            JavaPairRDD<String, String> tdocuments =
sc.wholeTextFiles(TESTING_URI);

      System.out.println(tdocuments.take((int)tdocuments.count()).toString())
;

//-------------------------------------------------------------------------
----------------------------------

            // each training document starts with the label &&  get the
label, and change it to an integer
            JavaPairRDD<String, Tuple2<Integer,String>> trainingDocs =
documents.mapValues(
                        new Function<String,Tuple2<Integer,String>>()
                        {
                              public Tuple2<Integer,String> call(String line)
throws Exception
                              {
                                    if ( line == null || line.length() == 0 )
return null;
                                    if ( line.indexOf(LABEL_SEPARATOR) < 0 )
return null;
                                    String label = line.substring(0,
line.indexOf(LABEL_SEPARATOR));
                                    if (
allCategories.getValue().containsKey(label) == false )
                                    {
                                          // missing label
                                          return null;
                                    }
                                    String content =
line.substring(line.indexOf(LABEL_SEPARATOR)+1);
                                    return new
Tuple2(allCategories.getValue().get(label),content);
                  }
            });
      System.out.println(trainingDocs.take((int)trainingDocs.count()).toStrin
g());

//-------------------------------------------------------------------------
----------------------------------

      // create a dataframe for training documents
            StructType docSchema = new StructType(
            new StructField[] {
```

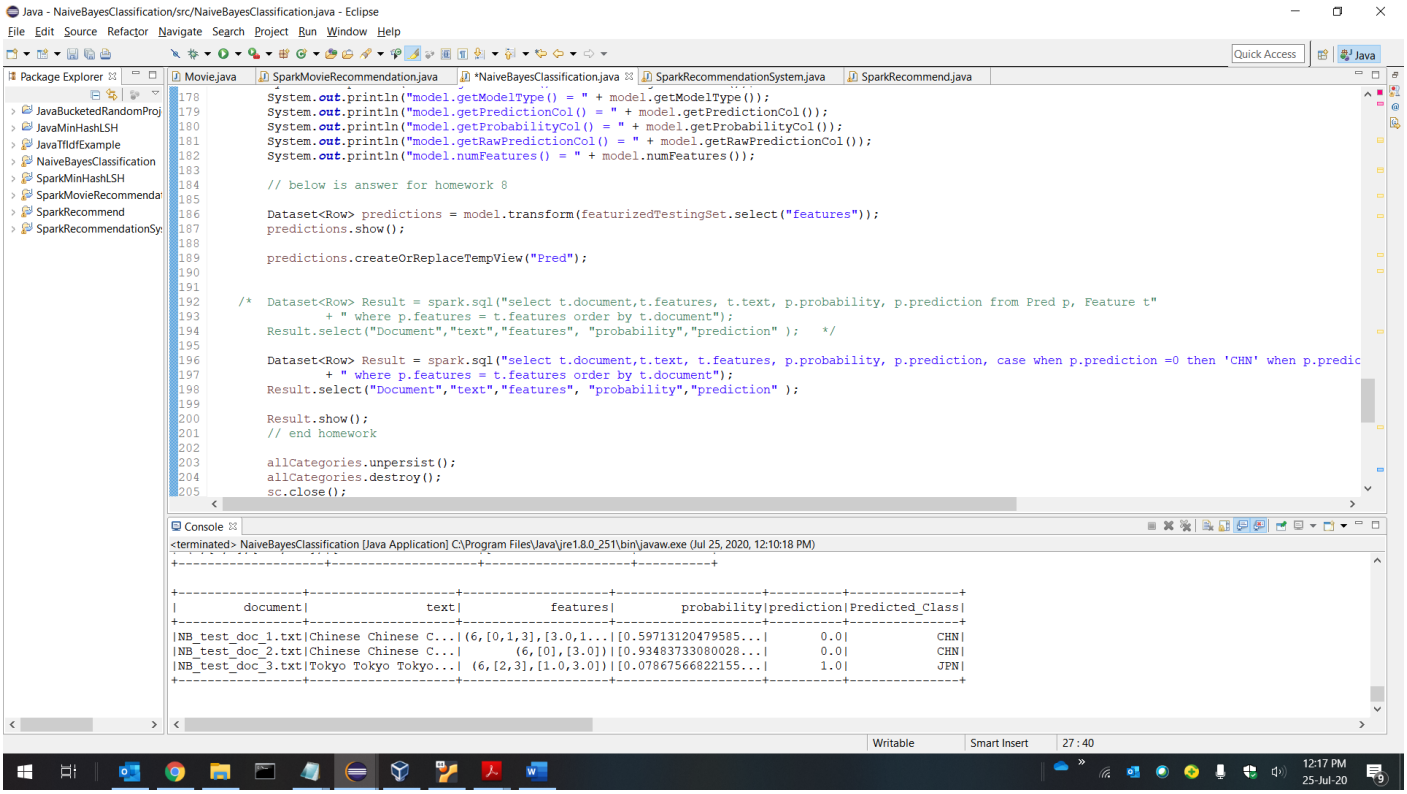
```
                                DataTypes.createStructField("label",
DataTypes.IntegerType, false),
                                DataTypes.createStructField("text",
DataTypes.StringType, false)
                }
      );


                //Create a dataFrame for testing documents
                StructType tdocSchema = new StructType(
                            new StructField[]{
                                        DataTypes.createStructField("Document",
DataTypes.StringType, false),
                                        DataTypes.createStructField("text",
DataTypes.StringType, false)});

//-------------------------------------------------------------------------
-----------------------------------

                Dataset<Row> trainingSet = spark.createDataFrame(
                            trainingDocs.map( new Function<Tuple2<String,
Tuple2<Integer,String>>, Row> () {
                @Override
                            public Row call(Tuple2<String,
Tuple2<Integer,String>> record) {
                                        return RowFactory.create(record._2()._1(),
record._2()._2());
                            }
                      } ), docSchema);
                trainingSet.show(true);


                Dataset<Row> testingSet = spark.createDataFrame(
                   tdocuments.map( new Function<Tuple2<String, String>, Row> ()
                   {
                         public Row call(Tuple2<String, String> trecord){
                               return
RowFactory.create(trecord._1().substring(trecord._1().lastIndexOf("/")+1),tre
cord._2());

                         }}
                   ),tdocSchema);
          testingSet.show(true);

//-------------------------------------------------------------------------
-----------------------------------

          // tokenizer the training set

                Tokenizer tokenizer = new
Tokenizer().setInputCol("text").setOutputCol("words");
                Dataset<Row> trainingSetTokenized =
tokenizer.transform(trainingSet);
                trainingSetTokenized.show(true);

            // tokenizer of the testing Set
```

```
            Tokenizer tokenizer1 = new
Tokenizer().setInputCol("text").setOutputCol("words");
            Dataset<Row> testingSetTokenized =
tokenizer1.transform(testingSet);
            testingSetTokenized.show(true);

//--------------------------------------------------------------------------
-----------------------------------
            // remove stopwords etc, can use Stanford NLP library if needed
            StopWordsRemover remover = new
StopWordsRemover().setInputCol("words").setOutputCol("filtered");
            Dataset<Row> trainingSetStopWordsRemoved =
remover.transform(trainingSetTokenized);
            trainingSetStopWordsRemoved.show(true);

            StopWordsRemover remover1 = new
StopWordsRemover().setInputCol("words").setOutputCol("filtered");
            Dataset<Row> testingSetStopWordsRemoved =
remover1.transform(testingSetTokenized);
            testingSetStopWordsRemoved.show(true);

//--------------------------------------------------------------------------
-----------------------------------
            // fit a CountVectorizerModel from the corpus
            CountVectorizer vectorizer = new
CountVectorizer().setInputCol("filtered").setOutputCol("features");
            CountVectorizerModel cvm =
vectorizer.fit(trainingSetStopWordsRemoved);

            // fit a CountVectorizerModel from the corpus
            CountVectorizer vectorizer1 = new
CountVectorizer().setInputCol("filtered").setOutputCol("features");
            CountVectorizerModel cvm1 =
vectorizer1.fit(testingSetStopWordsRemoved);


//--------------------------------------------------------------------------
-----------------------------------
            System.out.println("vocab size = " + cvm.vocabulary().length);
            for (int i = 0; i < cvm.vocabulary().length; i ++ ) {
                  System.out.print(cvm.vocabulary()[i] + "(" + i + ") ");
                  }
            System.out.println();
            Dataset<Row> featurizedTrainingSet =
cvm.transform(trainingSetStopWordsRemoved);
            System.out.println("===> final featured testing set");
            featurizedTrainingSet.show(true);
            System.out.println("vocab size = " + cvm.vocabulary().length);
            for (int i = 0; i < cvm.vocabulary().length; i ++ ) {
                  System.out.print(cvm.vocabulary()[i] + "(" + i + ") ");
            }
            System.out.println();
            Dataset<Row> featurizedTestingSet =
cvm.transform(testingSetStopWordsRemoved);
            System.out.println("===> final featured testing set");
            featurizedTestingSet.show(true);
            featurizedTestingSet.createOrReplaceTempView("Feature");
```

```
//-------------------------------------------------------------------------
------------------------

            // create naive bayes model and train it
            NaiveBayes nb = new NaiveBayes();
            NaiveBayesModel model =
nb.fit(featurizedTrainingSet.select("label", "features"));
            //NaiveBayesModel model1 =
nb.train(featurizedTrainingSet.select("label", "features"));
            // study the model
            System.out.println("model.getFeaturesCol() = " +
model.getFeaturesCol());
            System.out.println("model.getLabelCol() = " +
model.getLabelCol());
            System.out.println("model.getModelType() = " +
model.getModelType());
            System.out.println("model.getPredictionCol() = " +
model.getPredictionCol());
            System.out.println("model.getProbabilityCol() = " +
model.getProbabilityCol());
            System.out.println("model.getRawPredictionCol() = " +
model.getRawPredictionCol());
            System.out.println("model.numFeatures() = " +
model.numFeatures());

            // below is answer for homework 8

            Dataset<Row> predictions =
model.transform(featurizedTestingSet.select("features"));
            predictions.show();

            predictions.createOrReplaceTempView("Pred");


      /*    Dataset<Row> Result = spark.sql("select t.document,t.features,
t.text, p.probability, p.prediction from Pred p, Feature t"
                        + " where p.features = t.features order by
t.document");
            Result.select("Document","text","features",
"probability","prediction" );   */

            Dataset<Row> Result = spark.sql("select t.document,t.text,
t.features, p.probability, p.prediction, case when p.prediction =0 then 'CHN'
when p.prediction =1 then 'JPN' end as Predicted_Class from Pred p, Feature
t"
                        + " where p.features = t.features order by
t.document");
            Result.select("Document","text","features",
"probability","prediction" );

            Result.show();
            // end homework

            allCategories.unpersist();
            allCategories.destroy();
            sc.close();
```

```
	}

	private static HashMap getCategoryMap(String filePath) {

		HashMap<String, Integer> categories = new
HashMap<String,Integer>();
		BufferedReader br = null;

		try {
			br = new BufferedReader(new FileReader(CATEGORIES));
			String line = br.readLine();

			while (line != null) {

				StringTokenizer st = new StringTokenizer(line);
				String categoryText = st.nextToken();
				Integer categoryIndex = new Integer(st.nextToken());
				categories.put(categoryText, categoryIndex);

				line = br.readLine();

			}

		} catch(Exception e) { // handle it the way you want
			System.out.println(e.getMessage());
		} finally {
			if ( br != null ) {
				try {
					br.close();
				} catch (IOException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
		}
		return categories;
	}
}
```

Output:



document	text	features	probability	prediction	Predicted_Class
NB_test_doc_1.txt	Chinese Chinese C...	(6,[0,1,3],[3.0,1...	[0.59713120479585...	0.0	CHN
NB_test_doc_2.txt	Chinese Chinese C...	(6,[0],[3.0])	[0.93483733080028...	0.0	CHN
NB_test_doc_3.txt	Tokyo Tokyo Tokyo...	(6,[2,3],[1.0,3.0])	[0.07867566822155...	1.0	JPN