

|  |
| --- |
| Module 11: Apache Spark Streaming –  Data Sources |
| Case Study I Solution |



© Brain4ce Education Solutions Pvt. Ltd.

Solution:

**Step 1:** Modify the model application to train the model and persist it Add following line in ***src/main/scala/SpamMessage.scala***

import org.apache.spark.sql.SparkSession import org.apache.spark.ml.feature.Tokenizer

import org.apache.spark.ml.feature.StopWordsRemover

import org.apache.spark.ml.feature.{CountVectorizer, CountVectorizerModel} import org.apache.spark.ml.{Pipeline, PipelineModel}

import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator import org.apache.spark.ml.classification.LogisticRegression

import org.apache.spark.ml.feature.{OneHotEncoder, StringIndexer} object SpamMessage {

def main(args: Array[String]) {

val spark = SparkSession.builder.appName("Spam Messages").getOrCreate() spark.sparkContext.setLogLevel("ERROR")

// Load data in spark val raw =

spark.read.option("delimiter","\t").csv("use\_cases/SMSSpamCollection").toDF("s pam","message")

// Extract words from the SMS message

val tokenizer = new Tokenizer().setInputCol("message").setOutputCol("words")

// Modify the stop words to include your custom words such as â-â

val stopwords = new StopWordsRemover().getStopWords ++ Array("-") val remover = new

StopWordsRemover().setStopWords(stopwords).setInputCol("words").setOutput

Col("filtered")

// Create the features from SMS message using CountVectorizer

val cvmodel = new CountVectorizer().setInputCol("filtered").setOutputCol("features")

val indexer = new StringIndexer().setInputCol("spam").setOutputCol("label") val lr = new

LogisticRegression().setMaxIter(10).setRegParam(0.3).setElasticNetParam(0.8)

val pipeline = new Pipeline().setStages(Array(tokenizer, remover, cvmodel, indexer, lr))

val model = pipeline.fit(raw) model.save("spam\_message.model") val output = model.transform(raw) println("--->", output.show()) spark.stop()

}

}

**Step 2:** Create table in Mysql

create table spam\_message(message varchar(140),prediction double);

**Step 3:** Create a new spark streaming application to predict the spam messages

import org.apache.spark.sql.SparkSession

//import org.apache.spark.ml.feature.QuantileDiscretizer

//import org.apache.spark.sql.types.\_

//import org.apache.spark.ml.feature.VectorAssembler

//import org.apache.spark.ml.linalg.Vectors

//import org.apache.spark.ml.Pipeline import org.apache.spark.sql.functions.\_

//import org.apache.spark.ml.evaluation.RegressionEvaluator

//import org.apache.spark.ml.regression.{RandomForestRegressionModel, RandomForestRegressor}

import org.apache.spark.\_

import org.apache.spark.streaming.\_ import org.apache.spark.sql.Encoders import org.apache.spark.ml.\_

case class Message(message:String) object SpamStreaming {

def main(args: Array[String]) {

val conf = new SparkConf().setAppName("SpamStreaming") val ssc = new StreamingContext(conf, Seconds(10))

val lines = ssc.textFileStream("tmp/kafka/spam\_message") lines.foreachRDD { rdd =>

val spark=SparkSession.builder().getOrCreate() import spark.implicits.\_

val rawRdd = rdd.map(Message(\_))

val raw = spark.createDataFrame(rawRdd)

val pipeline = PipelineModel.read.load("spam\_message.model") val predictions = pipeline.transform(raw)

val prop = new java.util.Properties prop.put("driver", "com.mysql.jdbc.Driver");

prop.put("url", "jdbc:mysql://mysqldb.edu.cloudlab.com/use\_cases"); prop.put("user", "labuser");

prop.put("password", "edureka"); predictions.select("message","prediction").write.mode("append").jdbc( prop.getProperty("url"), "spam\_message", prop)

}

ssc.start() ssc.awaitTermination()

}