Public Opinion Analysis of Airlines

Team 1:

Fan Wu

Dayu Jia

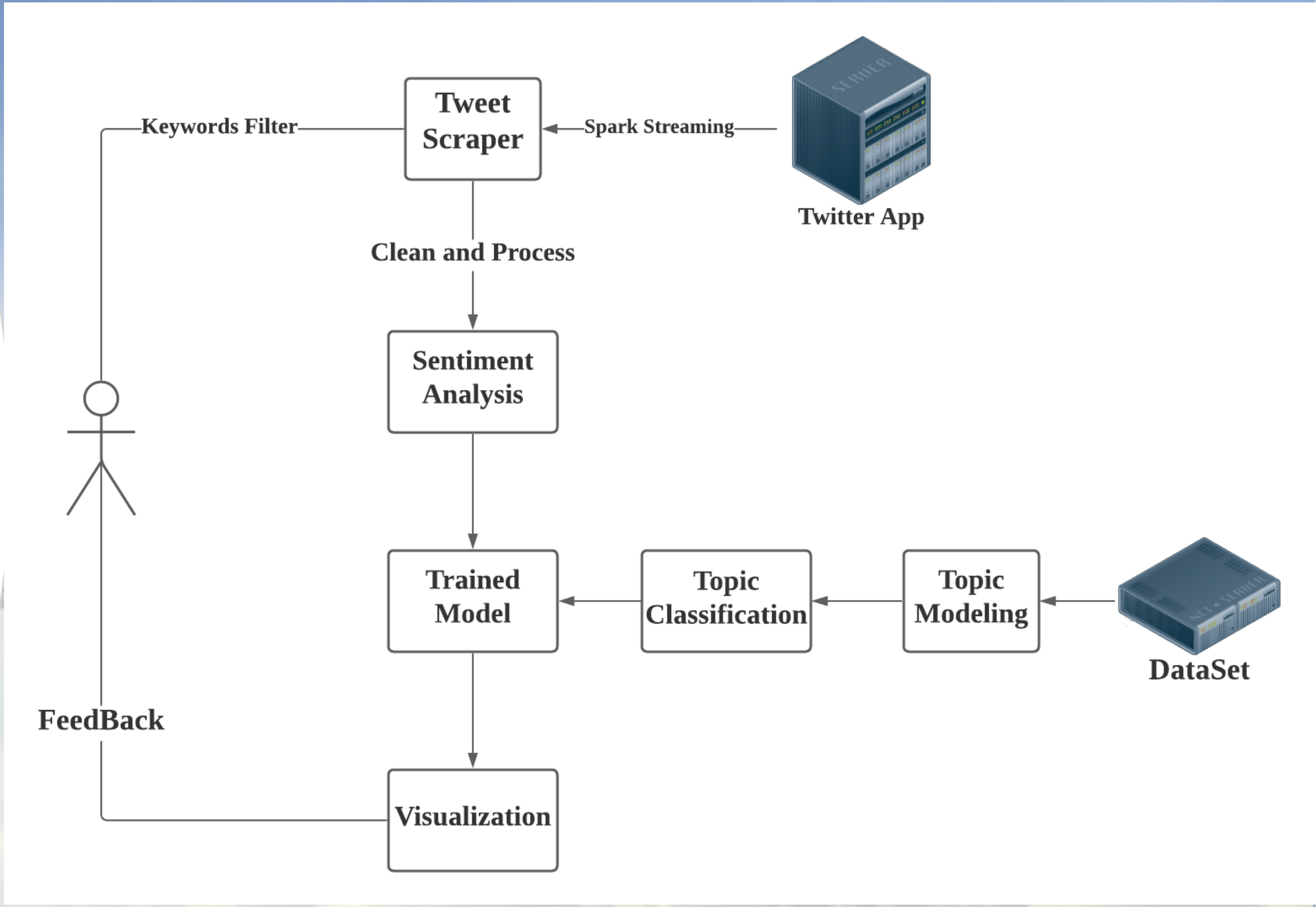
Bowen Jiang

Github link:

https://github.com/FanWu6/Spark-stream-twitter-analysis

Use Cases

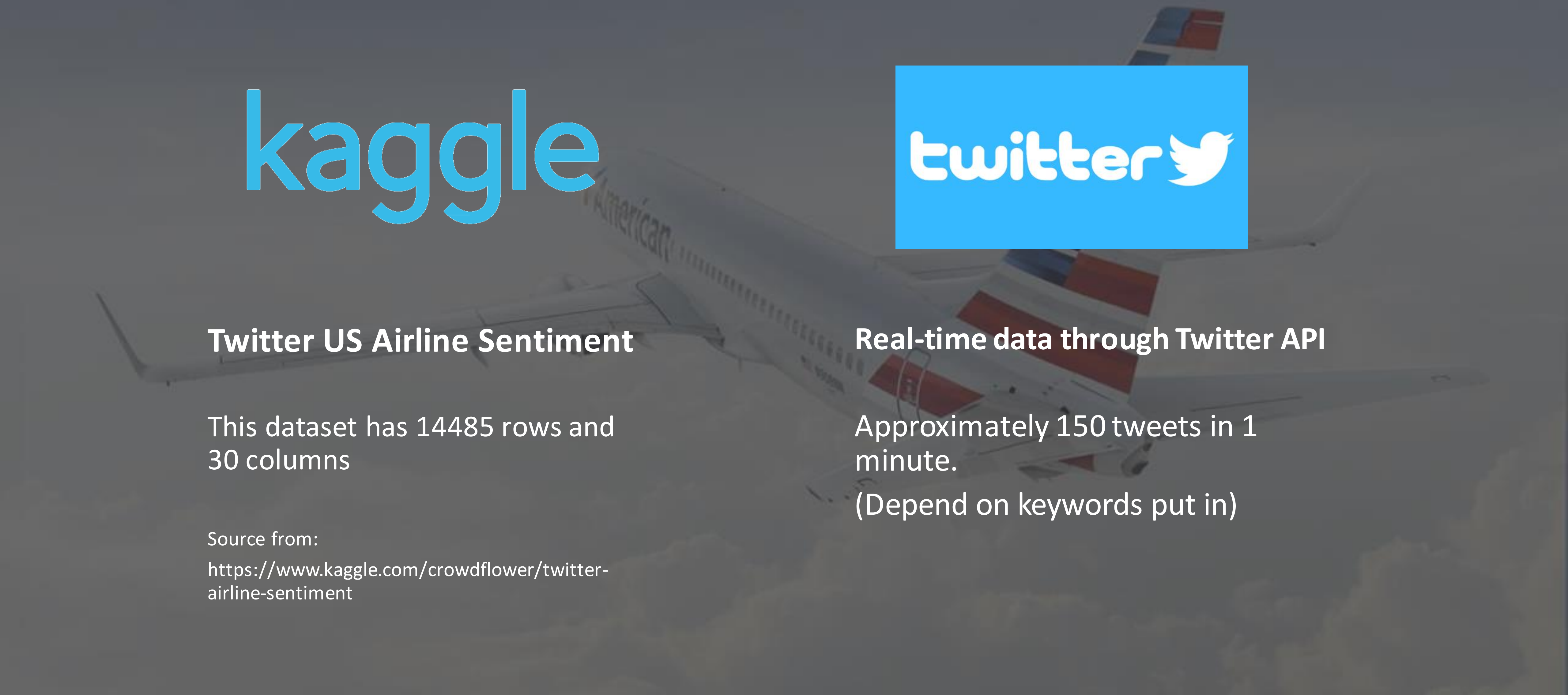
- User inputs the name of an airplane company
- and receives a comprehensive analysis on public opinion



Methodology

- `Tweets pulling->` Spark Streaming, `twitter4j`
- Data process -> `StopWordsRemover, tf-idf, SparkSQL,etc...`
- Topic modeling -> Latent Dirichlet Allocation(LDA)
- Topic classification -> Logistic Regression
- Sentiment analysis separately -> CoreNLP
- Visualization -> Elasticsearch, Kibana

Data Sources



Twitter US Airline Sentiment

This dataset has 14485 rows and 30 columns

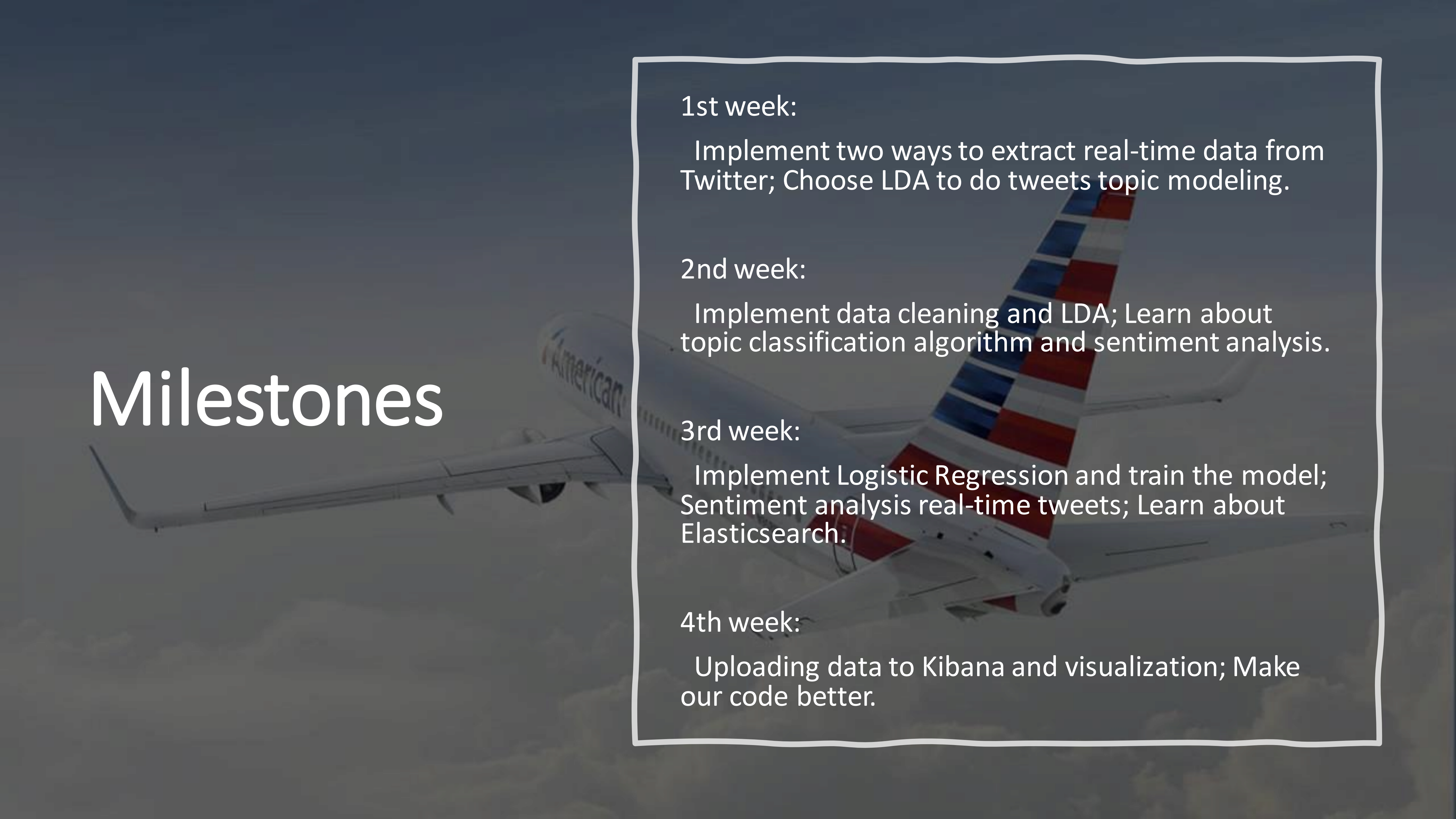
Source from:

https://www.kaggle.com/crowdflower/twitter-airline-sentiment

Real-time data through Twitter API

Approximately 150 tweets in 1 minute.

(Depend on keywords put in)



Milestones

1st week:

Implement two ways to extract real-time data from Twitter; Choose LDA to do tweets topic modeling.

2nd week:

Implement data cleaning and LDA; Learn about topic classification algorithm and sentiment analysis.

3rd week:

Implement Logistic Regression and train the model; Sentiment analysis real-time tweets; Learn about Elasticsearch.

4th week:

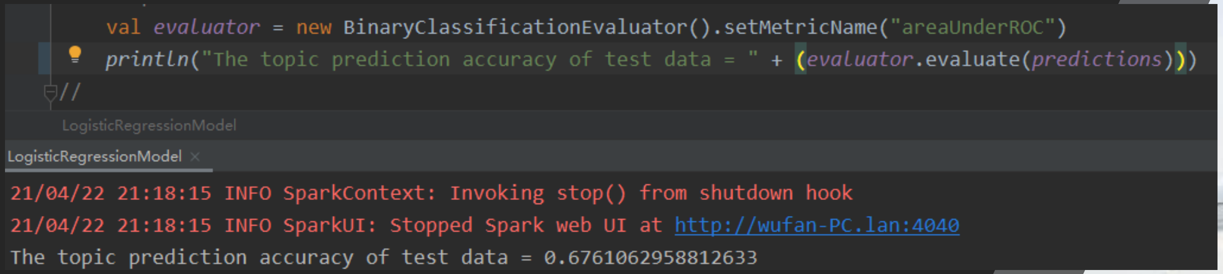
Uploading data to Kibana and visualization; Make our code better.

Programming in Scala

- Data extraction
- Data processing
- Model implement
- Unit tests
- Data uploading

Acceptance criteria

- 70% of data can be classified correctly
- The accuracy of sentimental analysis should reach 80%

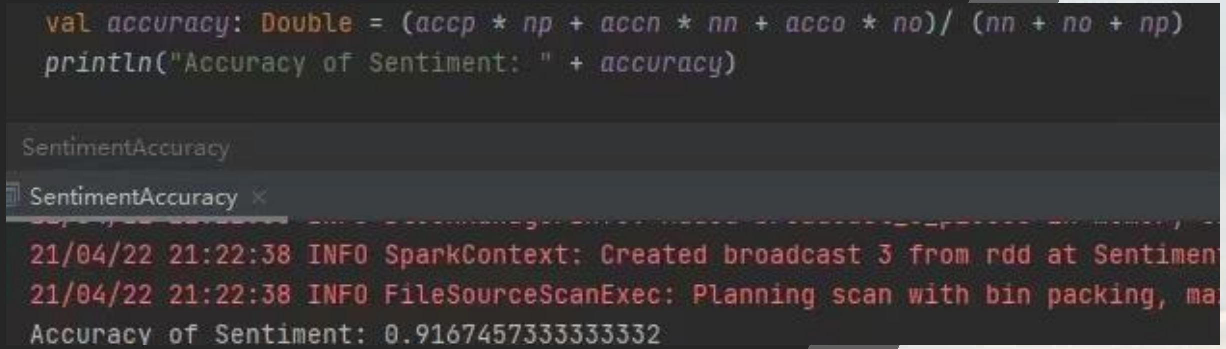


```
    val evaluator = new BinaryClassificationEvaluator().setMetricName("areaUnderROC")
    println("The topic prediction accuracy of test data = " + (evaluator.evaluate(predictions)))
//
```

LogisticRegressionModel

LogisticRegressionModel

```
21/04/22 21:18:15 INFO SparkContext: Invoking stop() from shutdown hook
21/04/22 21:18:15 INFO SparkUI: Stopped Spark web UI at http://wufan-PC.lan:4040
The topic prediction accuracy of test data = 0.6761062958812633
```



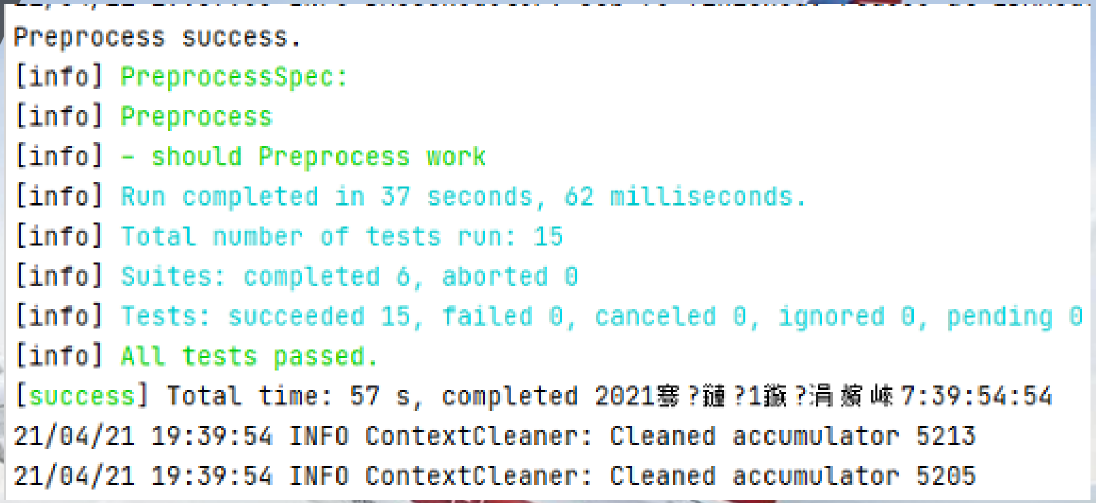
```
  val accuracy: Double = (accp * np + accn * nn + acco * no)/ (nn + no + np)
  println("Accuracy of Sentiment: " + accuracy)
```

SentimentAccuracy

SentimentAccuracy

```
21/04/22 21:22:38 INFO SparkContext: Created broadcast 3 from rdd at Sentimen
21/04/22 21:22:38 INFO FileSourceScanExec: Planning scan with bin packing, ma
Accuracy of Sentiment: 0.9167457333333332
```

Unit tests



```
Preprocess success.
[info] PreprocessSpec:
[info] Preprocess
[info] - should Preprocess work
[info] Run completed in 37 seconds, 62 milliseconds.
[info] Total number of tests run: 15
[info] Suites: completed 6, aborted 0
[info] Tests: succeeded 15, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 57 s, completed 2021骞?鏈?1鏃?涓嬪崍7:39:54:54
21/04/21 19:39:54 INFO ContextCleaner: Cleaned accumulator 5213
21/04/21 19:39:54 INFO ContextCleaner: Cleaned accumulator 5205
```

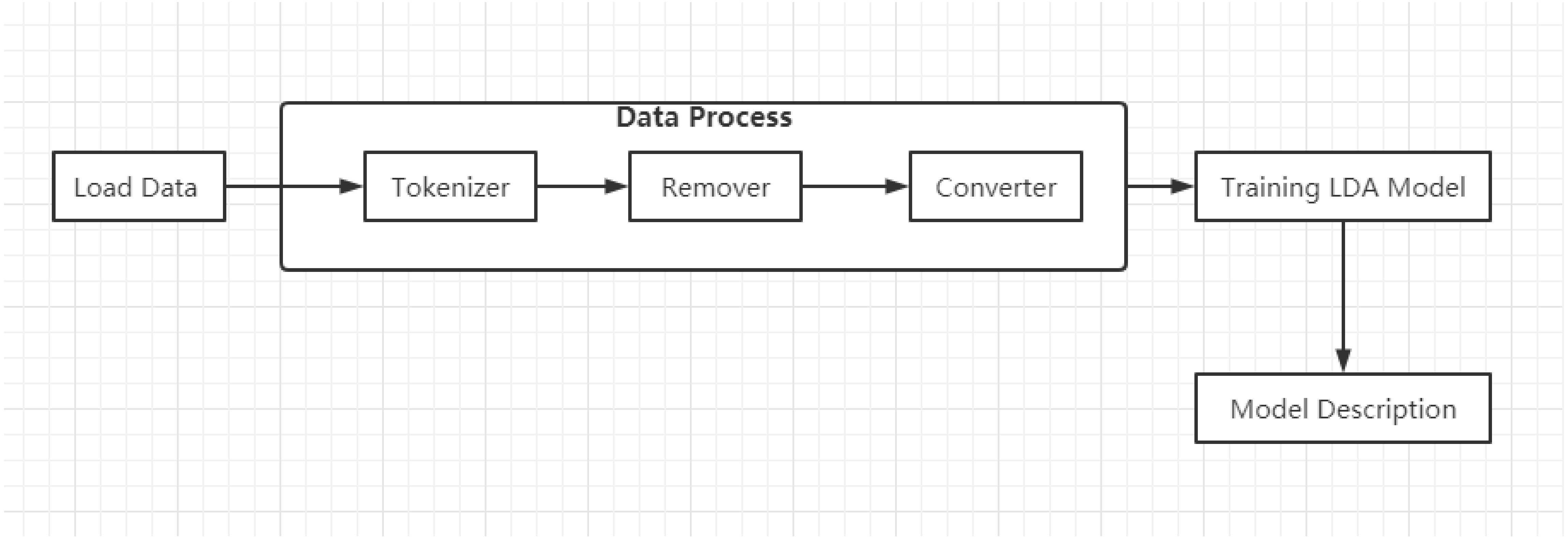
Goals

- For any airline company, we can offer a real–time analysis on public opinion based on our model.

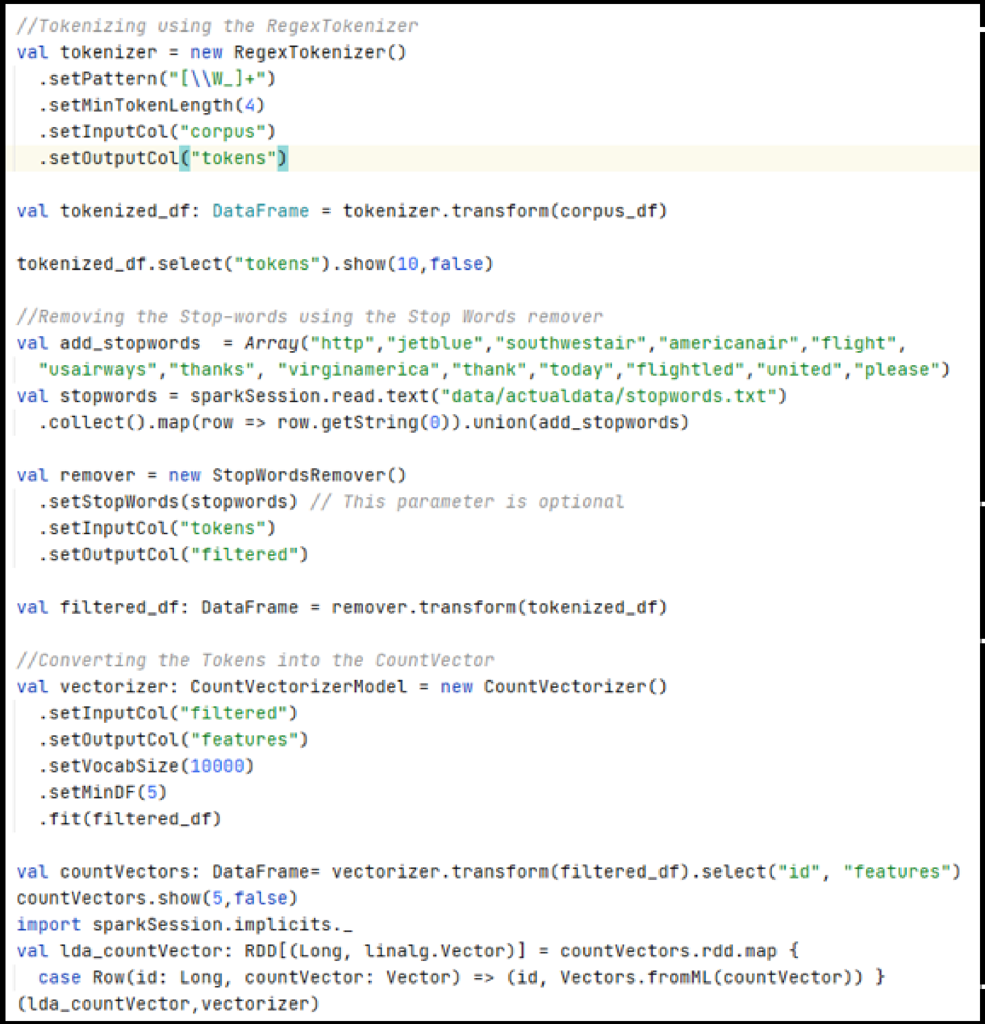
- For us, we want to learn the using of Scala and machine learning, and how to co-work on Github.

Latent Dirichlet Allocation

LDA Process Description

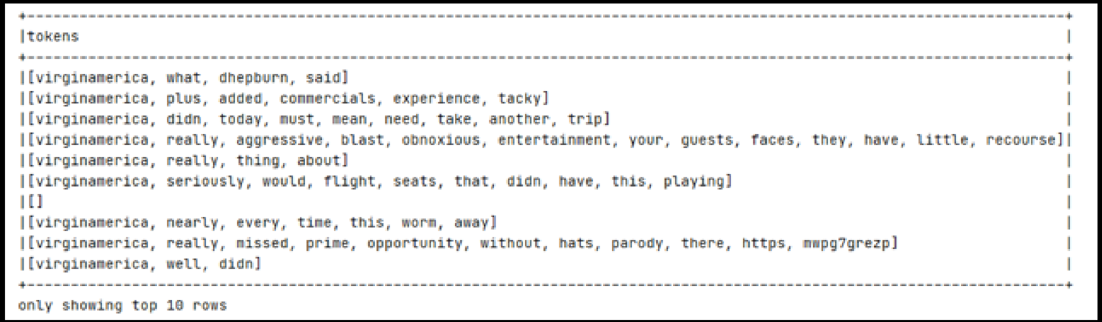


Data Preprocess Code Description



```scala
//Tokenizing using the RegexTokenizer
val tokenizer = new RegexTokenizer()
  .setPattern("[\\W_]+")
  .setMinTokenLength(4)
  .setInputCol("corpus")
  .setOutputCol("tokens")

val tokenized_df: DataFrame = tokenizer.transform(corpus_df)

tokenized_df.select("tokens").show(10,false)

//Removing the Stop-words using the Stop Words remover
val add_stopwords  = Array("http","jetblue","southwestair","americanair","flight",
  "usairways","thanks", "virginamerica","thank","today","flightled","united","please")
val stopwords = sparkSession.read.text("data/actualdata/stopwords.txt")
  .collect().map(row => row.getString(0)).union(add_stopwords)

val remover = new StopWordsRemover()
  .setStopWords(stopwords) // This parameter is optional
  .setInputCol("tokens")
  .setOutputCol("filtered")

val filtered_df: DataFrame = remover.transform(tokenized_df)

//Converting the Tokens into the CountVector
val vectorizer: CountVectorizerModel = new CountVectorizer()
  .setInputCol("filtered")
  .setOutputCol("features")
  .setVocabSize(10000)
  .setMinDF(5)
  .fit(filtered_df)

val countVectors: DataFrame= vectorizer.transform(filtered_df).select("id", "features")
countVectors.show(5,false)
import sparkSession.implicits._
val lda_countVector: RDD[(Long, linalg.Vector)] = countVectors.rdd.map {
  case Row(id: Long, countVector: Vector) => (id, Vectors.fromML(countVector)) }
(lda_countVector,vectorizer)
```

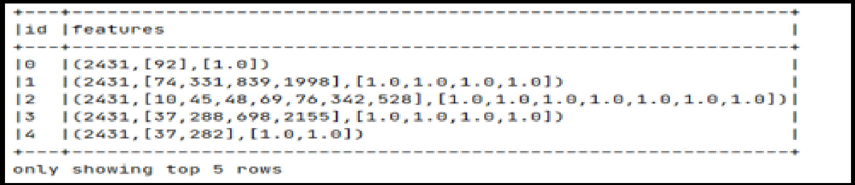
Using tokenizer filter away tokens with length <4



```
+----------------------------------------------------------------------------------------------------------------------------+
|tokens                                                                                                                      |
+----------------------------------------------------------------------------------------------------------------------------+
|[virginamerica, what, dhepburn, said]                                                                                       |
|[virginamerica, plus, added, commercials, experience, tacky]                                                                |
|[virginamerica, didn, today, must, mean, need, take, another, trip]                                                         |
|[virginamerica, really, aggressive, blast, obnoxious, entertainment, your, guests, faces, they, have, little, recourse]|
|[virginamerica, really, thing, about]                                                                                       |
|[virginamerica, seriously, would, flight, seats, that, didn, have, this, playing]                                           |
|[]                                                                                                                          |
|[virginamerica, nearly, every, time, this, worm, away]                                                                      |
|[virginamerica, really, missed, prime, opportunity, without, hats, parody, there, https, mwpg7grezp]                        |
|[virginamerica, well, didn]                                                                                                 |
+----------------------------------------------------------------------------------------------------------------------------+
only showing top 10 rows
```

Remove stopwords

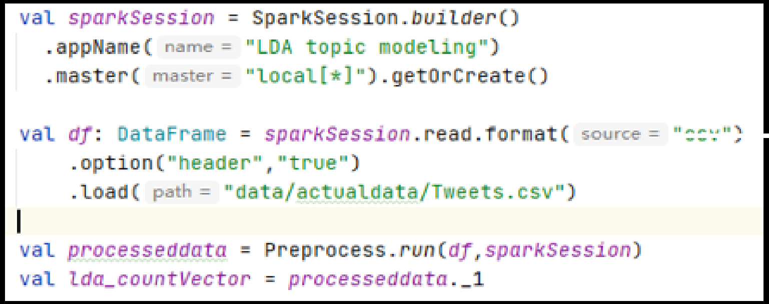
Set vocabulary size to 10000, and let the word show 5 times in each vocabulary



```
+---+-------------------------------------------------------------+
|id |features                                                     |
+---+-------------------------------------------------------------+
|0  |(2431,[92],[1.0])                                            |
|1  |(2431,[74,331,839,1998],[1.0,1.0,1.0,1.0])                   |
|2  |(2431,[10,45,48,69,76,342,528],[1.0,1.0,1.0,1.0,1.0,1.0,1.0])|
|3  |(2431,[37,288,698,2155],[1.0,1.0,1.0,1.0])                   |
|4  |(2431,[37,282],[1.0,1.0])                                    |
+---+-------------------------------------------------------------+
only showing top 5 rows
```

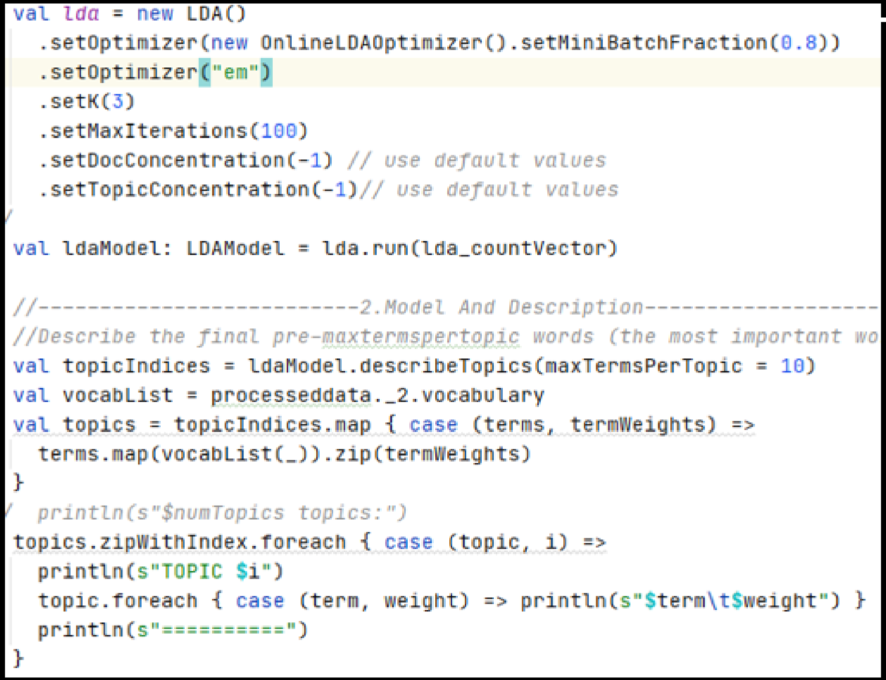
Completes the process of converting the documents into a vector of word counts

LDA Code Description



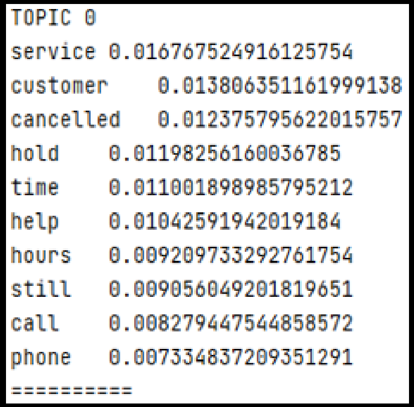
```scala
val sparkSession = SparkSession.builder()
  .appName( name = "LDA topic modeling")
  .master( master = "local[*]").getOrCreate()

val df: DataFrame = sparkSession.read.format( source = "csv")
    .option("header","true")
    .load( path = "data/actualdata/Tweets.csv")

val processeddata = Preprocess.run(df,sparkSession)
val lda_countVector = processeddata._1
```

Load Data.

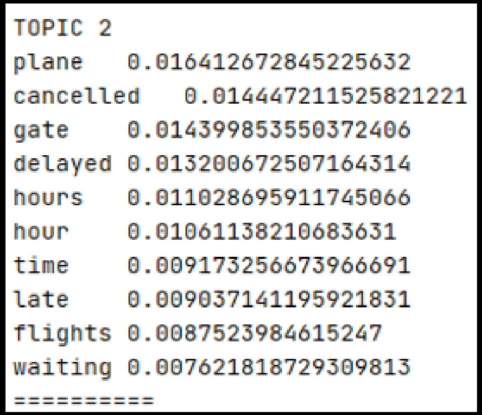


```scala
val lda = new LDA()
   .setOptimizer(new OnlineLDAOptimizer().setMiniBatchFraction(0.8))
   .setOptimizer("em")
   .setK(3)
   .setMaxIterations(100)
   .setDocConcentration(-1) // use default values
   .setTopicConcentration(-1)// use default values

val ldaModel: LDAModel = lda.run(lda_countVector)

//-------------------------2.Model And Description------------------
//Describe the final pre-maxtermspertopic words (the most important wo
val topicIndices = ldaModel.describeTopics(maxTermsPerTopic = 10)
val vocabList = processeddata._2.vocabulary
val topics = topicIndices.map { case (terms, termWeights) =>
  terms.map(vocabList(_)).zip(termWeights)
}
  println(s"$numTopics topics:")
topics.zipWithIndex.foreach { case (topic, i) =>
  println(s"TOPIC $i")
  topic.foreach { case (term, weight) => println(s"$term\t$weight") }
  println(s"==========")
}
```

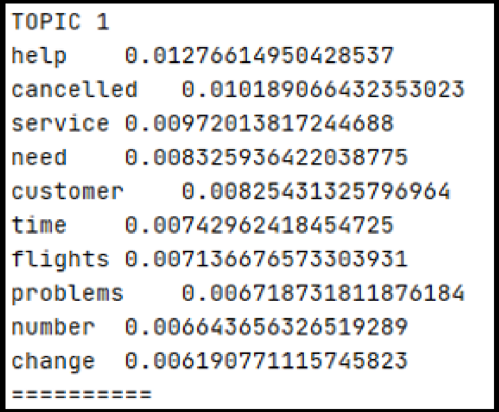
Builds LDA Model generate 3 topic after 100 iterations.



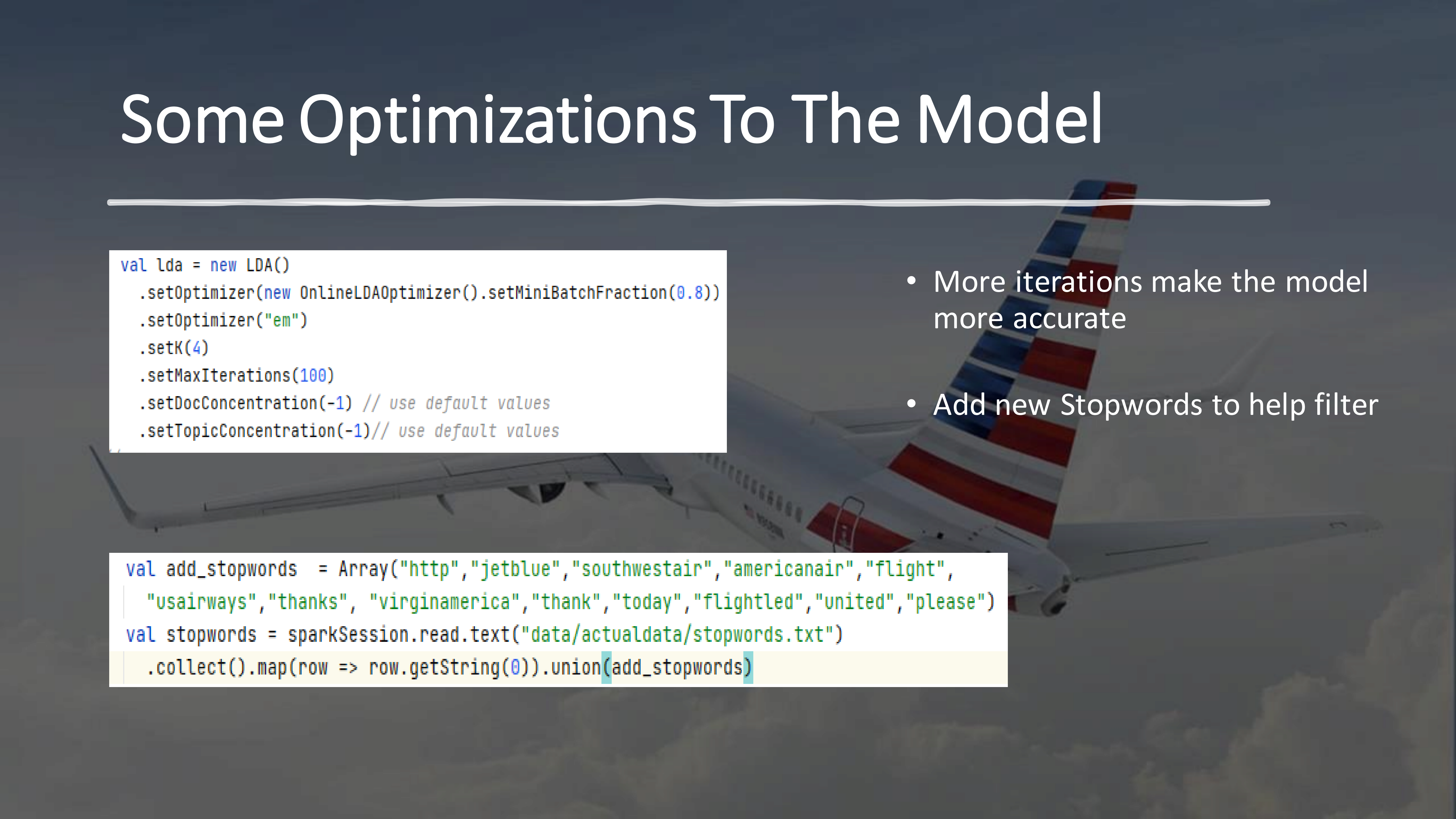
```
TOPIC 0
service 0.016767524916125754
customer    0.013806351161999138
cancelled   0.012375795622015757
hold    0.01198256160036785
time    0.011001898985795212
help    0.01042591942019184
hours   0.009209733292761754
still   0.009056049201819651
call    0.008279447544858572
phone   0.007334837209351291
==========
```



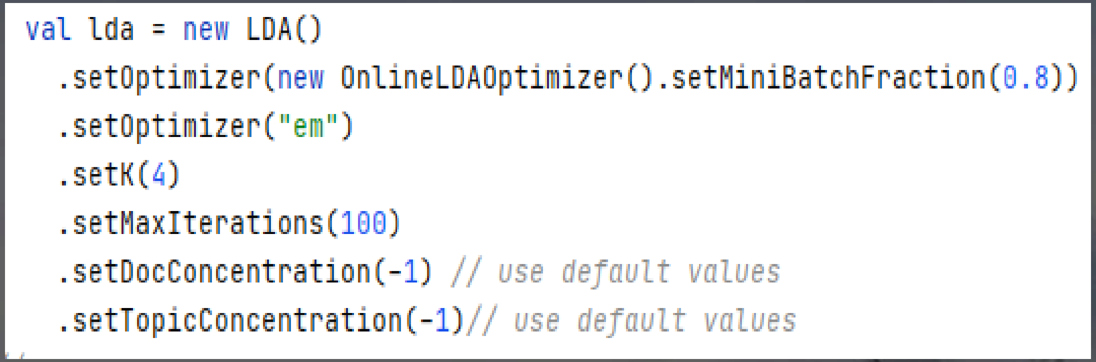
```
TOPIC 2
plane   0.016412672845225632
cancelled   0.014447211525821221
gate    0.014399853550372406
delayed 0.013200672507164314
hours   0.011028695911745066
hour    0.01061138210683631
time    0.009173256673966691
late    0.009037141195921831
flights 0.0087523984615247
waiting 0.007621818729309813
==========
```



```
TOPIC 1
help    0.01276614950428537
cancelled   0.010189066432353023
service 0.00972013817244688
need    0.008325936422038775
customer    0.00825431325796964
time    0.00742962418454725
flights 0.007136676573303931
problems    0.006718731811876184
number  0.006643656326519289
change  0.006190771115745823
==========
```

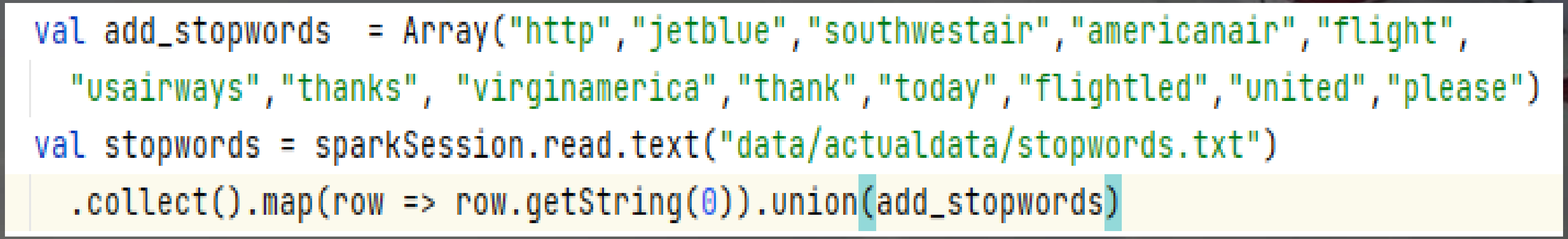


Some Optimizations To The Model



```scala
val lda = new LDA()
  .setOptimizer(new OnlineLDAOptimizer().setMiniBatchFraction(0.8))
  .setOptimizer("em")
  .setK(4)
  .setMaxIterations(100)
  .setDocConcentration(-1) // use default values
  .setTopicConcentration(-1)// use default values
```

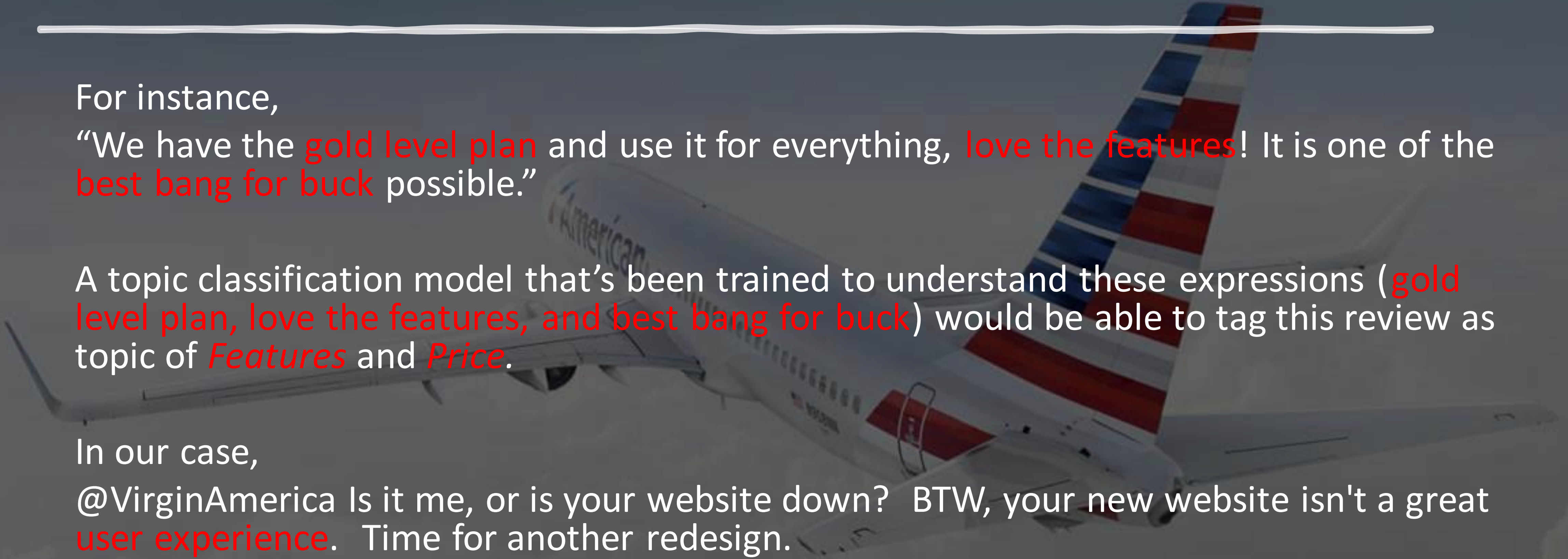
- More iterations make the model more accurate
- Add new Stopwords to help filter



```scala
val add_stopwords  = Array("http","jetblue","southwestair","americanair","flight",
  "usairways","thanks", "virginamerica","thank","today","flightled","united","please")
val stopwords = sparkSession.read.text("data/actualdata/stopwords.txt")
  .collect().map(row => row.getString(0)).union(add_stopwords)
```

Topic Classification

Topic classification



For instance,

"We have the gold level plan and use it for everything, love the features! It is one of the best bang for buck possible."

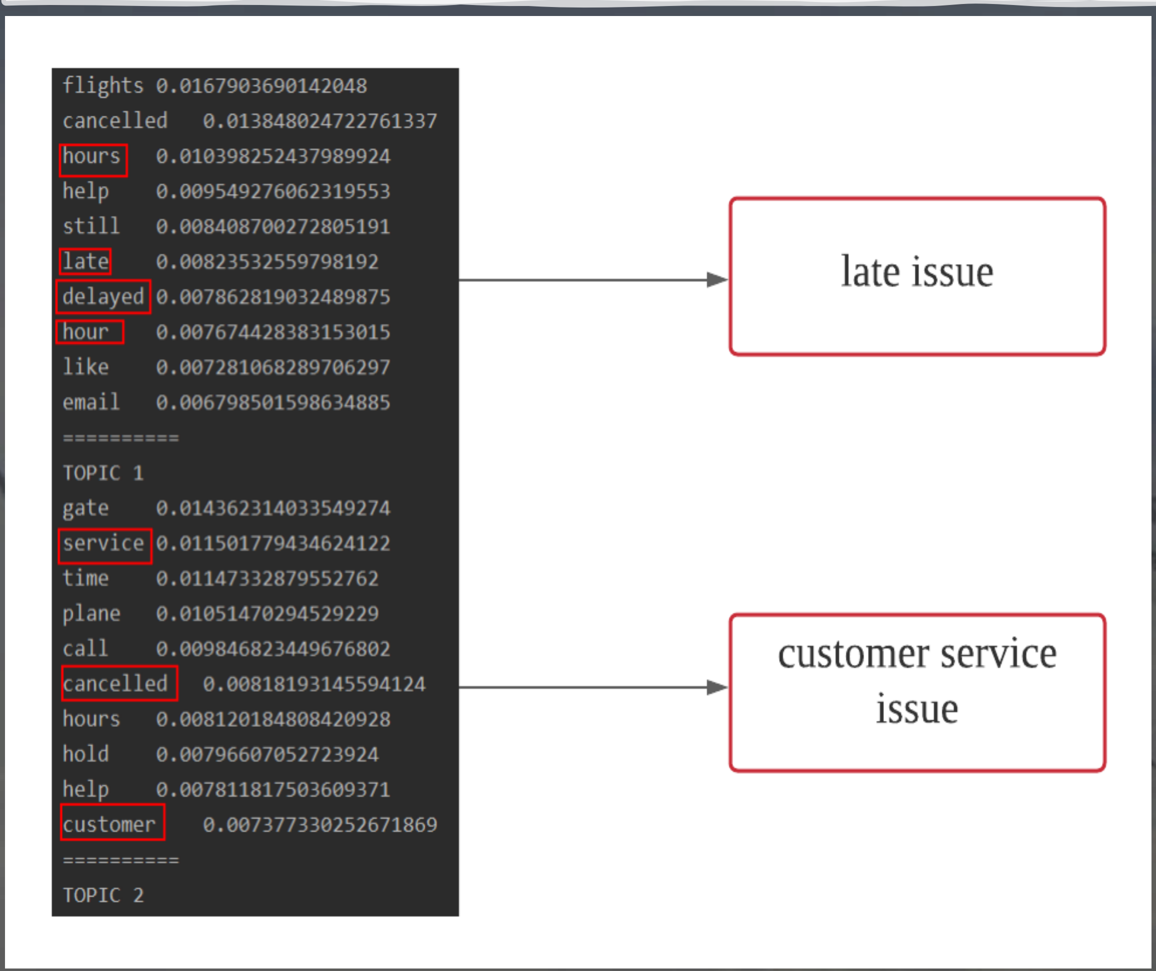
A topic classification model that's been trained to understand these expressions (gold level plan, love the features, and best bang for buck) would be able to tag this review as topic of *Features* and *Price*.

In our case,

@VirginAmerica Is it me, or is your website down? BTW, your new website isn't a great user experience. Time for another redesign.

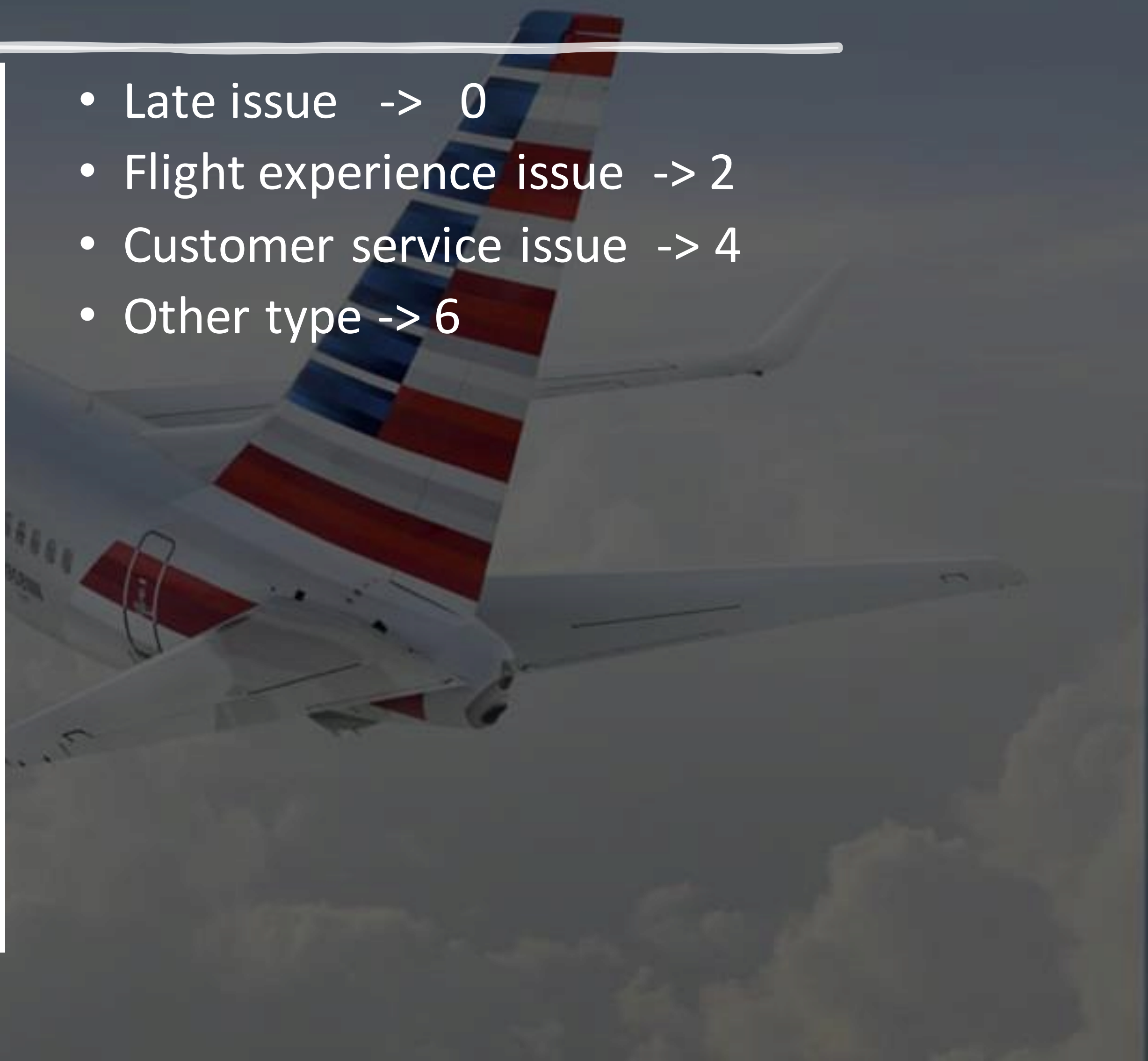
Customer Service Issue

Data for training

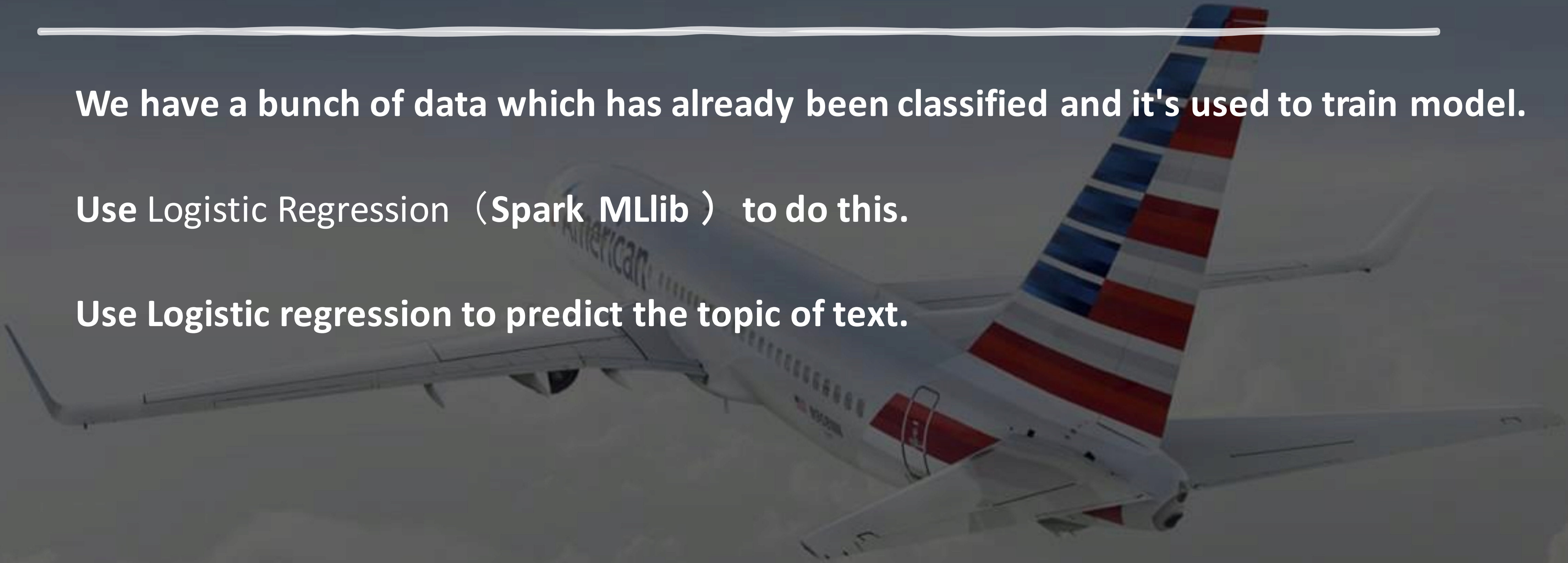


```
flights 0.0167903690142048
cancelled    0.013848024722761337
hours    0.010398252437989924
help     0.009549276062319553
still    0.008408700272805191
late     0.00823532559798192
delayed 0.007862819032489875
hour     0.007674428383153015
like     0.007281068289706297
email    0.006798501598634885
==========
TOPIC 1
gate     0.014362314033549274
service 0.011501779434624122
time     0.01147332879552762
plane    0.01051470294529229
call     0.009846823449676802
cancelled    0.00818193145594124
hours    0.008120184808420928
hold     0.00796607052723924
help     0.007811817503609371
customer     0.007377330252671869
==========
TOPIC 2
```

- Late issue -> 0
- Flight experience issue -> 2
- Customer service issue -> 4
- Other type -> 6



Logistic Regression

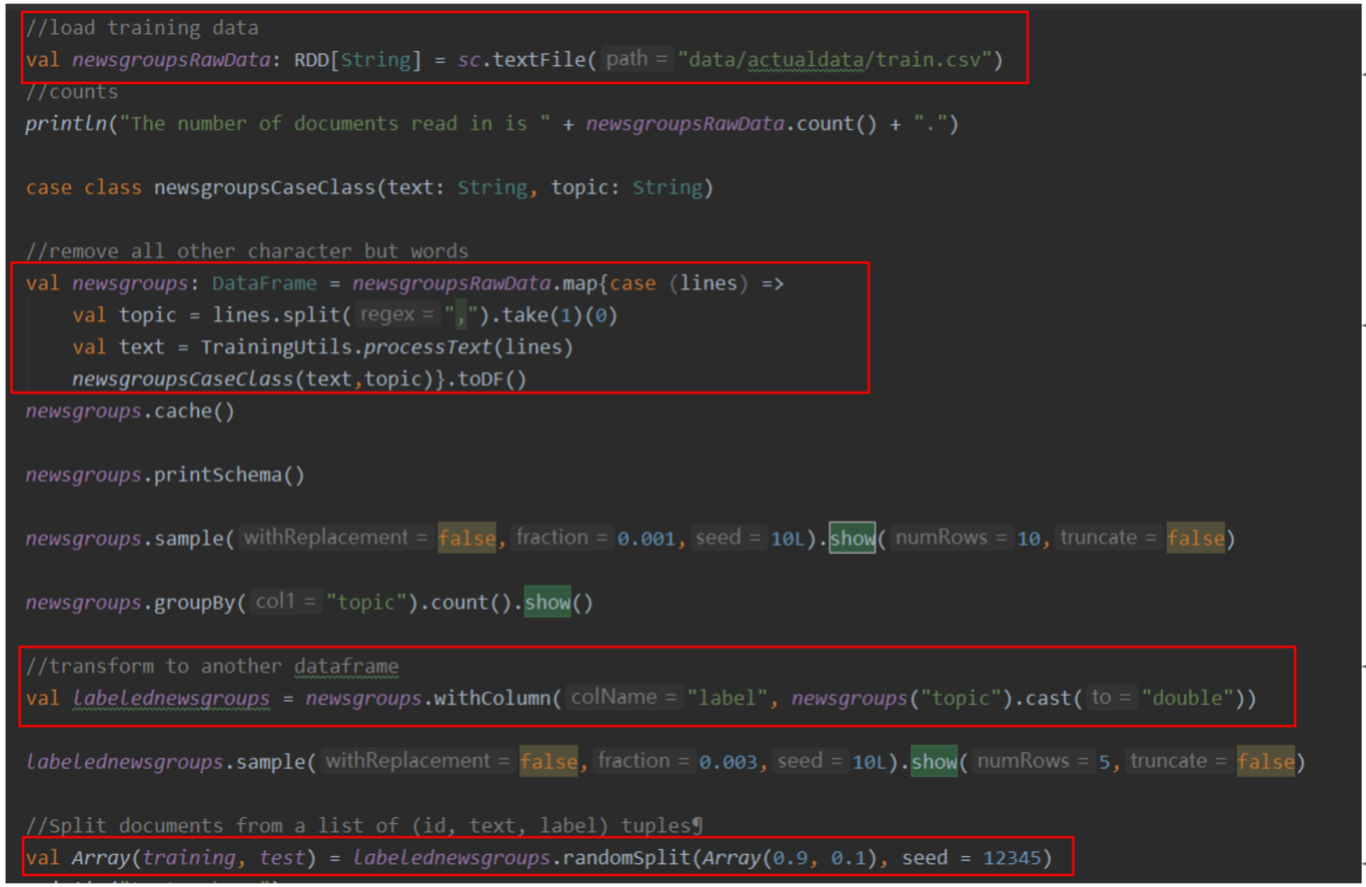


We have a bunch of data which has already been classified and it's used to train model.

Use Logistic Regression （**Spark MLlib** ） **to do this.**

Use Logistic regression to predict the topic of text.

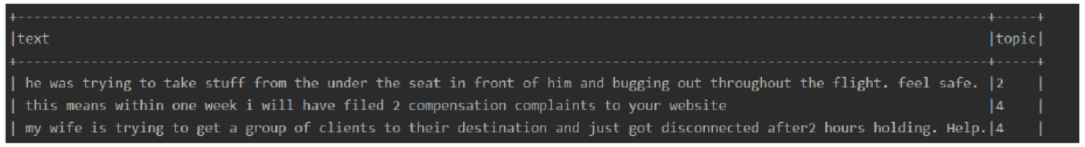
Pre-process



```
//load training data
val newsgroupsRawData: RDD[String] = sc.textFile( path = "data/actualdata/train.csv")
//counts
println("The number of documents read in is " + newsgroupsRawData.count() + ".")

case class newsgroupsCaseClass(text: String, topic: String)

//remove all other character but words
val newsgroups: DataFrame = newsgroupsRawData.map{case (lines) =>
    val topic = lines.split( regex = ",").take(1)(0)
    val text = TrainingUtils.processText(lines)
    newsgroupsCaseClass(text,topic)}.toDF()
newsgroups.cache()

newsgroups.printSchema()

newsgroups.sample( withReplacement = false, fraction = 0.001, seed = 10L).show( numRows = 10, truncate = false)

newsgroups.groupBy( col1 = "topic").count().show()

//transform to another dataframe
val labelednewsgroups = newsgroups.withColumn( colName = "label", newsgroups("topic").cast( to = "double"))

labelednewsgroups.sample( withReplacement = false, fraction = 0.003, seed = 10L).show( numRows = 5, truncate = false)

//Split documents from a list of (id, text, label) tuples¶
val Array(training, test) = labelednewsgroups.randomSplit(Array(0.9, 0.1), seed = 12345)
```

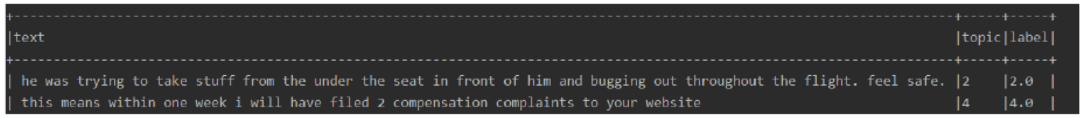
1. load data

2. extract column text, and remove all other charactor but only leave words



```
+-----------------------------------------------------------------------------------------------------+-----+
|text                                                                                                 |topic|
+-----------------------------------------------------------------------------------------------------+-----+
| he was trying to take stuff from the under the seat in front of him and bugging out throughout the flight. feel safe. |2    |
| this means within one week i will have filed 2 compensation complaints to your website             |4    |
| my wife is trying to get a group of clients to their destination and just got disconnected after2 hours holding. Help.|4    |
```

3.cast topic to double

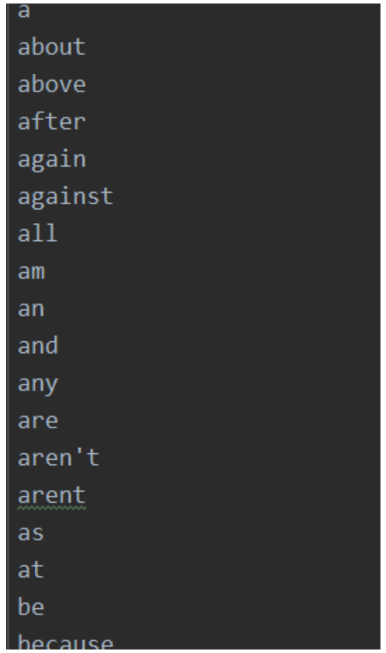


```
+----------------------------------------------------------------------------------------------------+-----+-----+
|text                                                                                                |topic|label|
+----------------------------------------------------------------------------------------------------+-----+-----+
| he was trying to take stuff from the under the seat in front of him and bugging out throughout the flight. feel safe. |2    |2.0  |
| this means within one week i will have filed 2 compensation complaints to your website            |4    |4.0  |
```

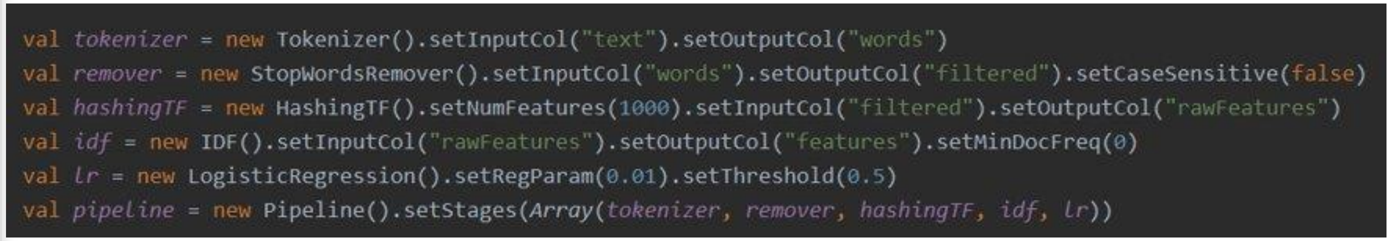
4.split data to training and test

Main stages

- Tokenizer
- StopWordsRemover
- TF-IDF
- Logistic Regression

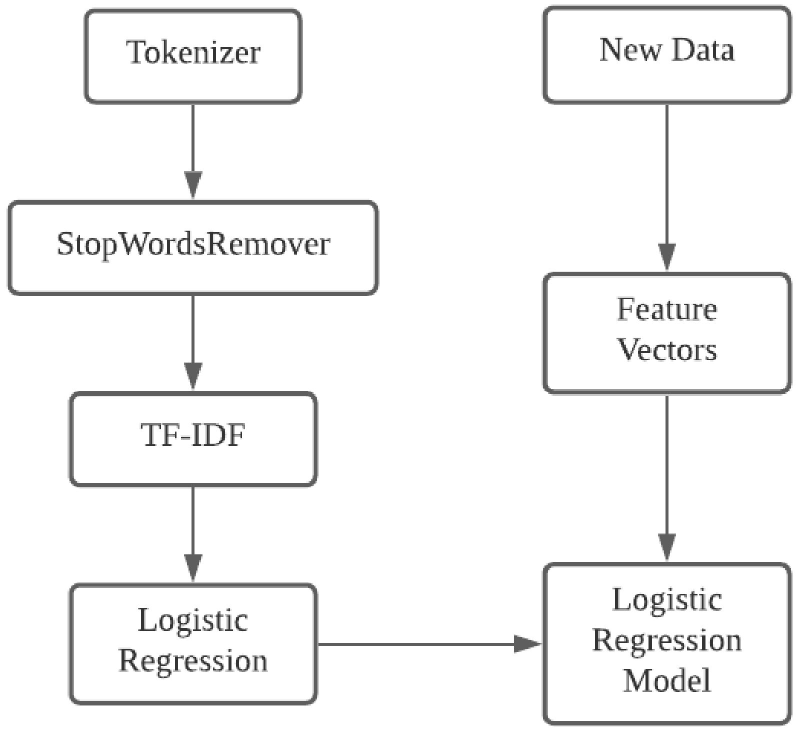


```
a
about
above
after
again
against
all
am
an
and
any
are
aren't
arent
as
at
be
because
```

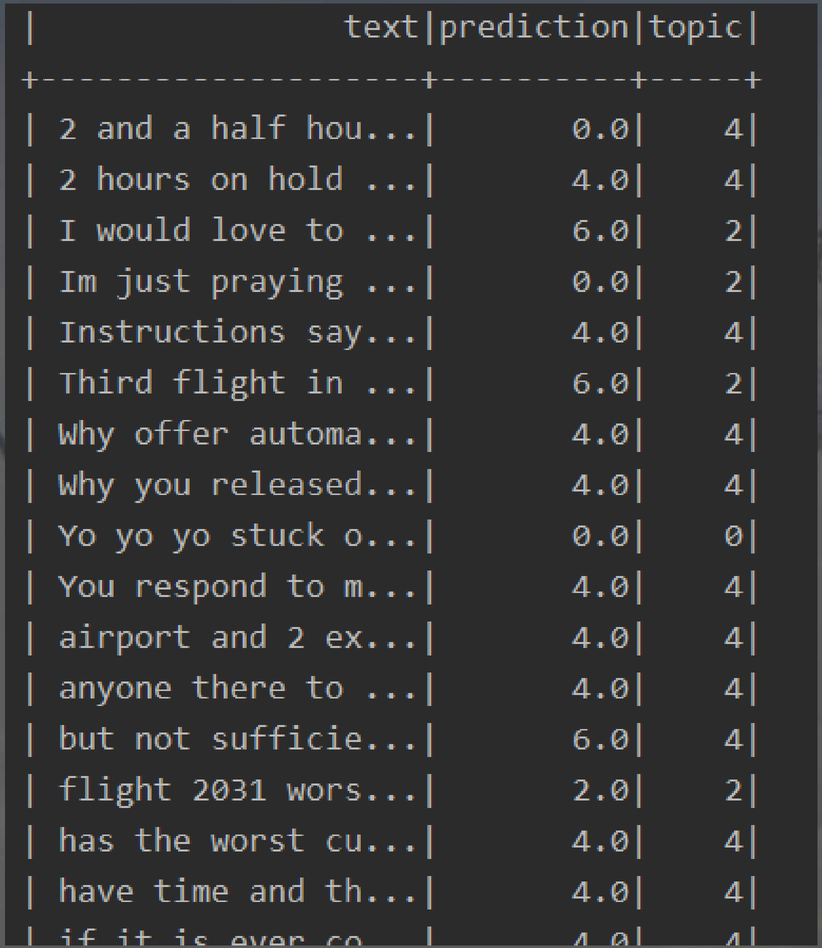


```
val tokenizer = new Tokenizer().setInputCol("text").setOutputCol("words")
val remover = new StopWordsRemover().setInputCol("words").setOutputCol("filtered").setCaseSensitive(false)
val hashingTF = new HashingTF().setNumFeatures(1000).setInputCol("filtered").setOutputCol("rawFeatures")
val idf = new IDF().setInputCol("rawFeatures").setOutputCol("features").setMinDocFreq(0)
val lr = new LogisticRegression().setRegParam(0.01).setThreshold(0.5)
val pipeline = new Pipeline().setStages(Array(tokenizer, remover, hashingTF, idf, lr))
```

Pipeline


Tokenizer
StopWordsRemover
TF-IDF
Logistic Regression
New Data
Feature Vectors
Logistic Regression Model

How to Optimize?



```
|                 text|prediction|topic|
+--------------------+----------+-----+
| 2 and a half hou...|       0.0|    4|
| 2 hours on hold ...|       4.0|    4|
| I would love to ...|       6.0|    2|
| Im just praying ...|       0.0|    2|
| Instructions say...|       4.0|    4|
| Third flight in ...|       6.0|    2|
| Why offer automa...|       4.0|    4|
| Why you released...|       4.0|    4|
| Yo yo yo stuck o...|       0.0|    0|
| You respond to m...|       4.0|    4|
| airport and 2 ex...|       4.0|    4|
| anyone there to ...|       4.0|    4|
| but not sufficie...|       6.0|    4|
| flight 2031 wors...|       2.0|    2|
| has the worst cu...|       4.0|    4|
| have time and th...|       4.0|    4|
| if it is ever co...|       4.0|    4|
```

01

1.Tune the hyperparameters of model.

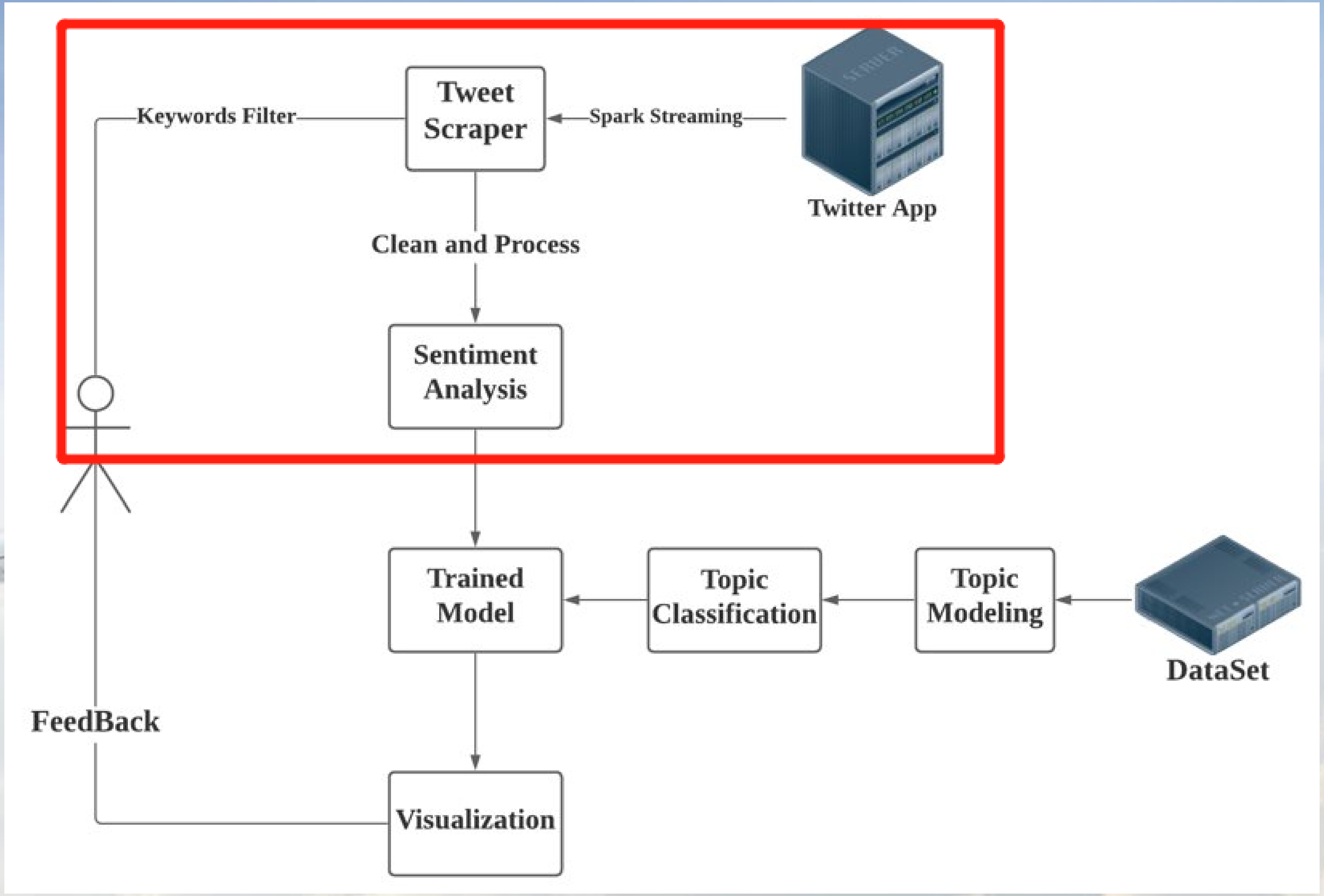
02

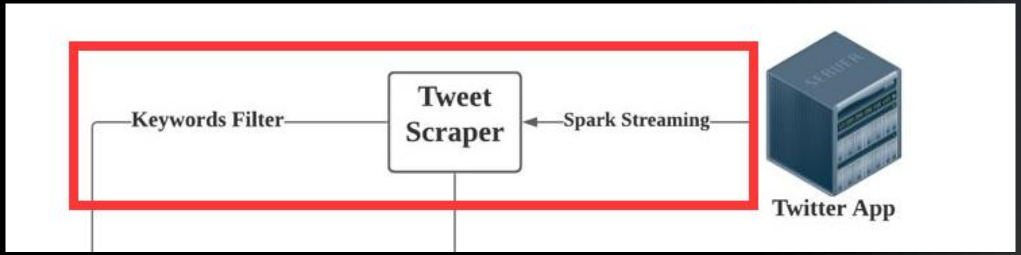
2.Make data more accurate.

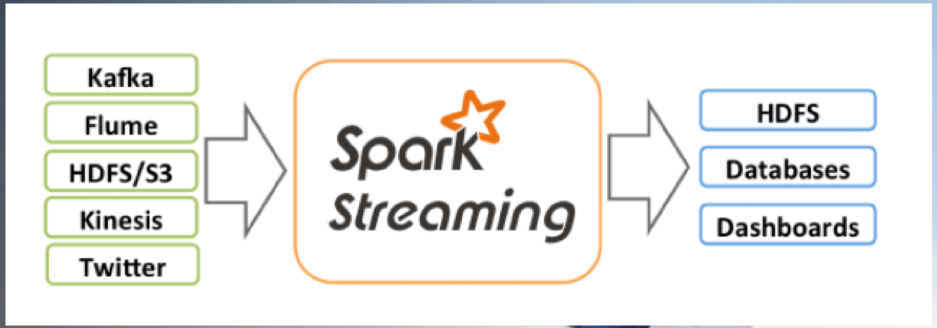
03

3.Training with more data...

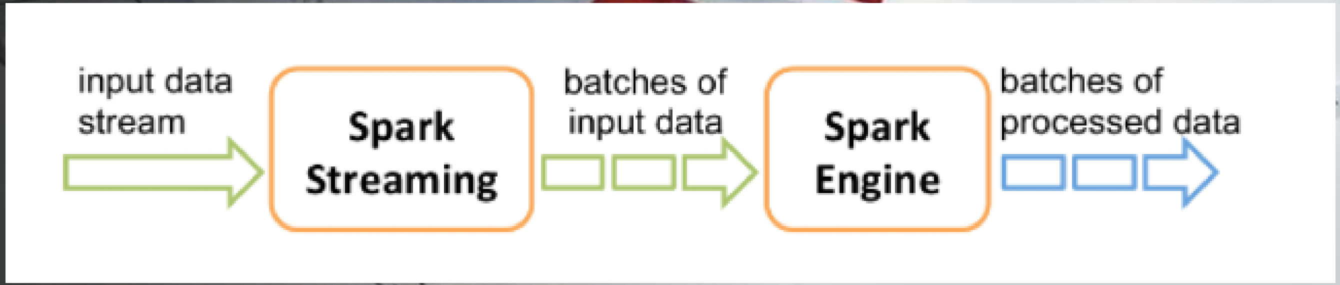
Twitter Real-time Data



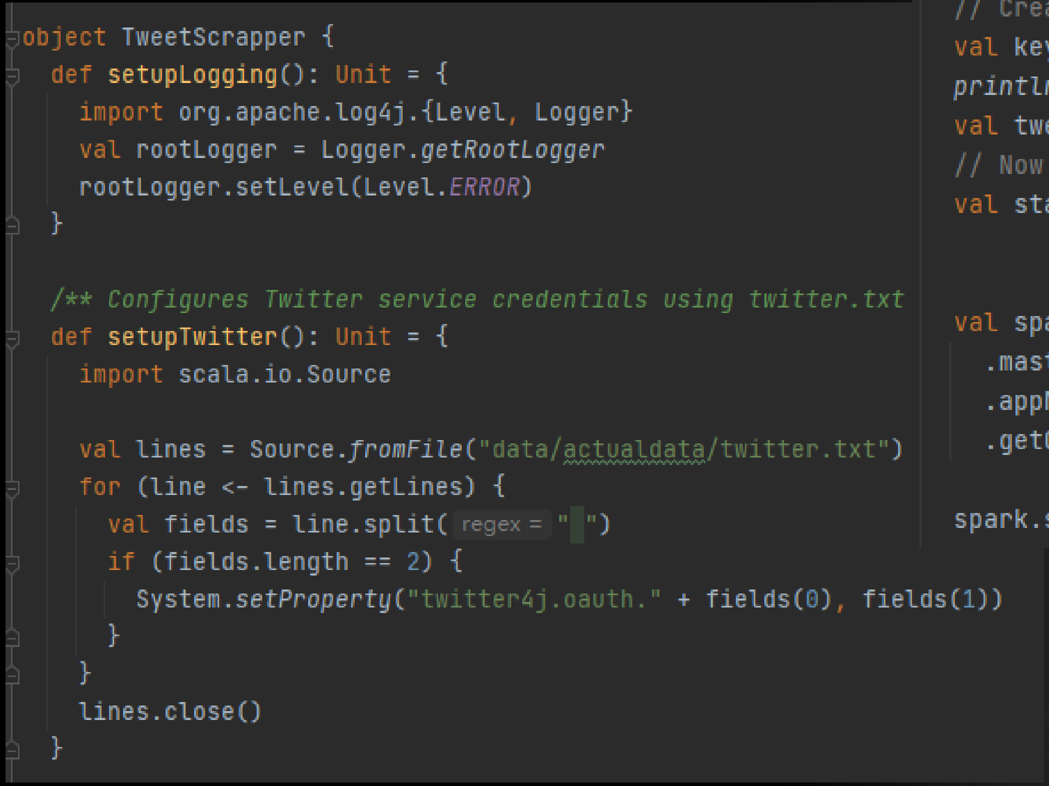




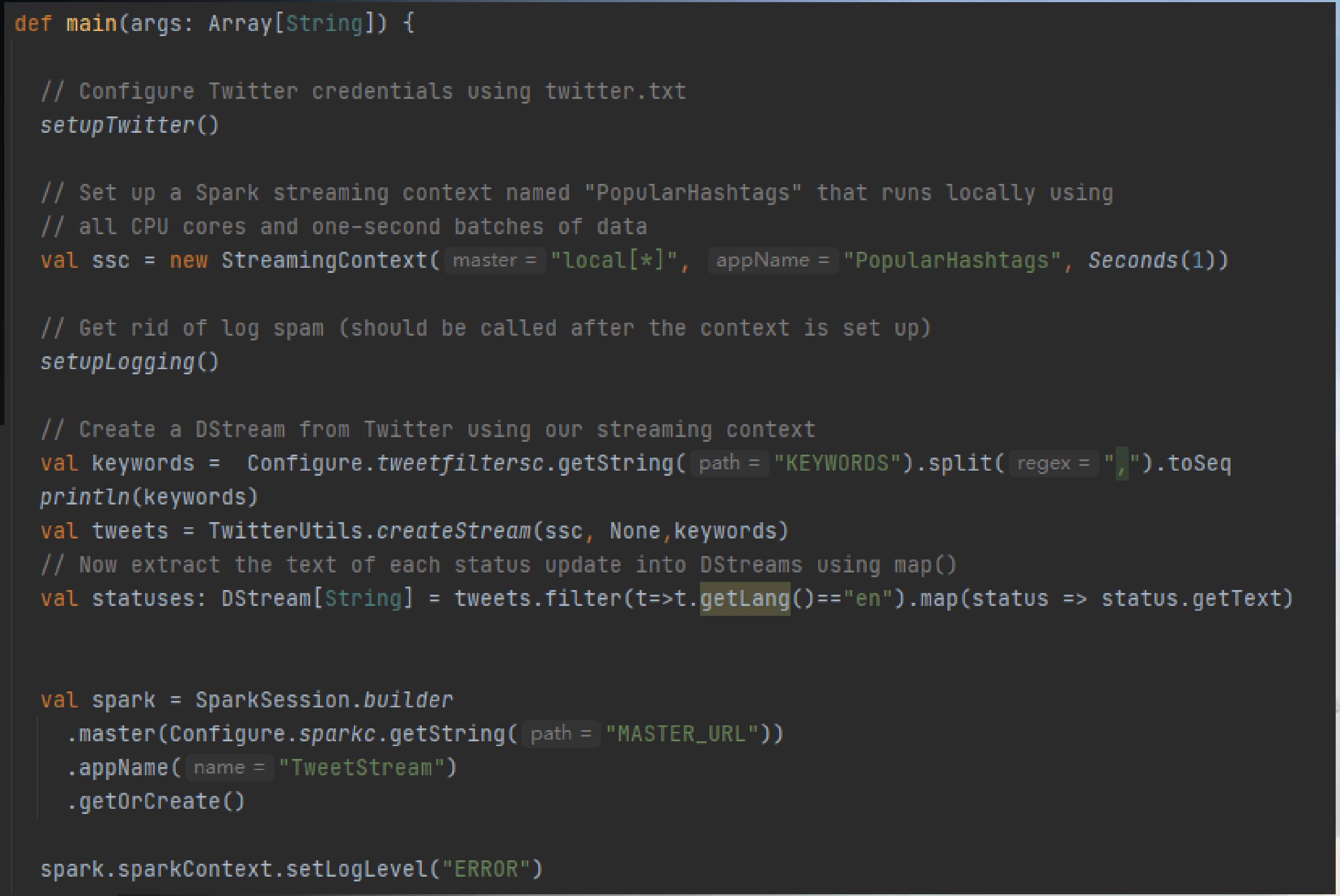
Twitter Scraper



Twitter Scraper

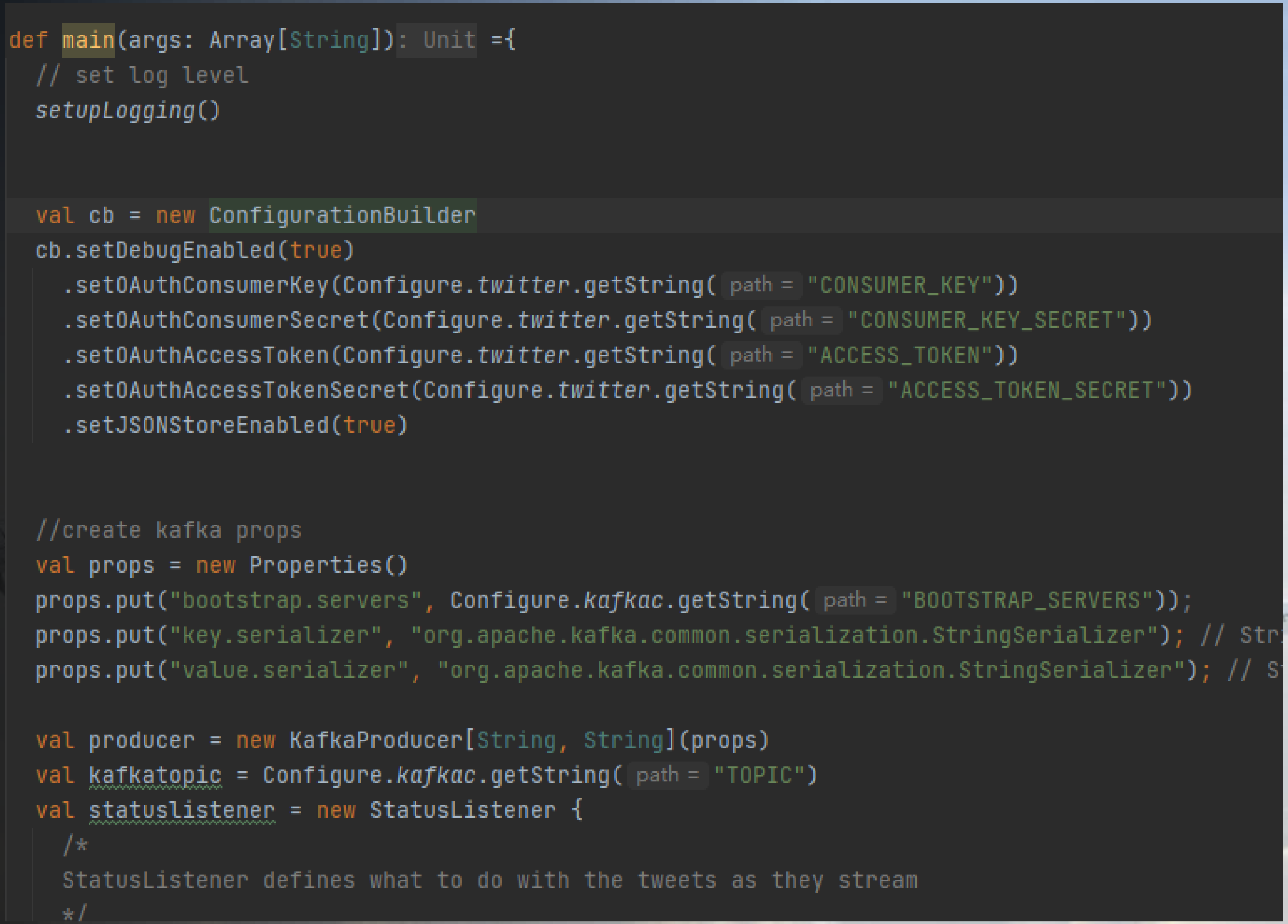


```scala
object TweetScrapper {
  def setupLogging(): Unit = {
    import org.apache.log4j.{Level, Logger}
    val rootLogger = Logger.getRootLogger
    rootLogger.setLevel(Level.ERROR)
  }

  /** Configures Twitter service credentials using twitter.txt
  def setupTwitter(): Unit = {
    import scala.io.Source

    val lines = Source.fromFile("data/actualdata/twitter.txt")
    for (line <- lines.getLines) {
      val fields = line.split( regex = " ")
      if (fields.length == 2) {
        System.setProperty("twitter4j.oauth." + fields(0), fields(1))
      }
    }
    lines.close()
  }
```



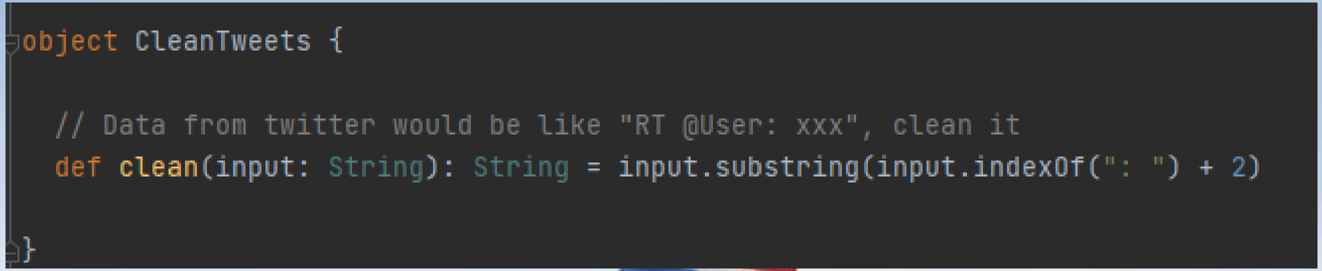
```scala
def main(args: Array[String]) {

  // Configure Twitter credentials using twitter.txt
  setupTwitter()

  // Set up a Spark streaming context named "PopularHashtags" that runs locally using
  // all CPU cores and one-second batches of data
  val ssc = new StreamingContext( master = "local[*]",  appName = "PopularHashtags", Seconds(1))

  // Get rid of log spam (should be called after the context is set up)
  setupLogging()

  // Create a DStream from Twitter using our streaming context
  val keywords =  Configure.tweetfiltersc.getString( path = "KEYWORDS").split( regex = ",").toSeq
  println(keywords)
  val tweets = TwitterUtils.createStream(ssc, None,keywords)
  // Now extract the text of each status update into DStreams using map()
  val statuses: DStream[String] = tweets.filter(t=>t.getLang()=="en").map(status => status.getText)


  val spark = SparkSession.builder
    .master(Configure.sparkc.getString( path = "MASTER_URL"))
    .appName( name = "TweetStream")
    .getOrCreate()

  spark.sparkContext.setLogLevel("ERROR")
```

Twitter Scraper with Kafka

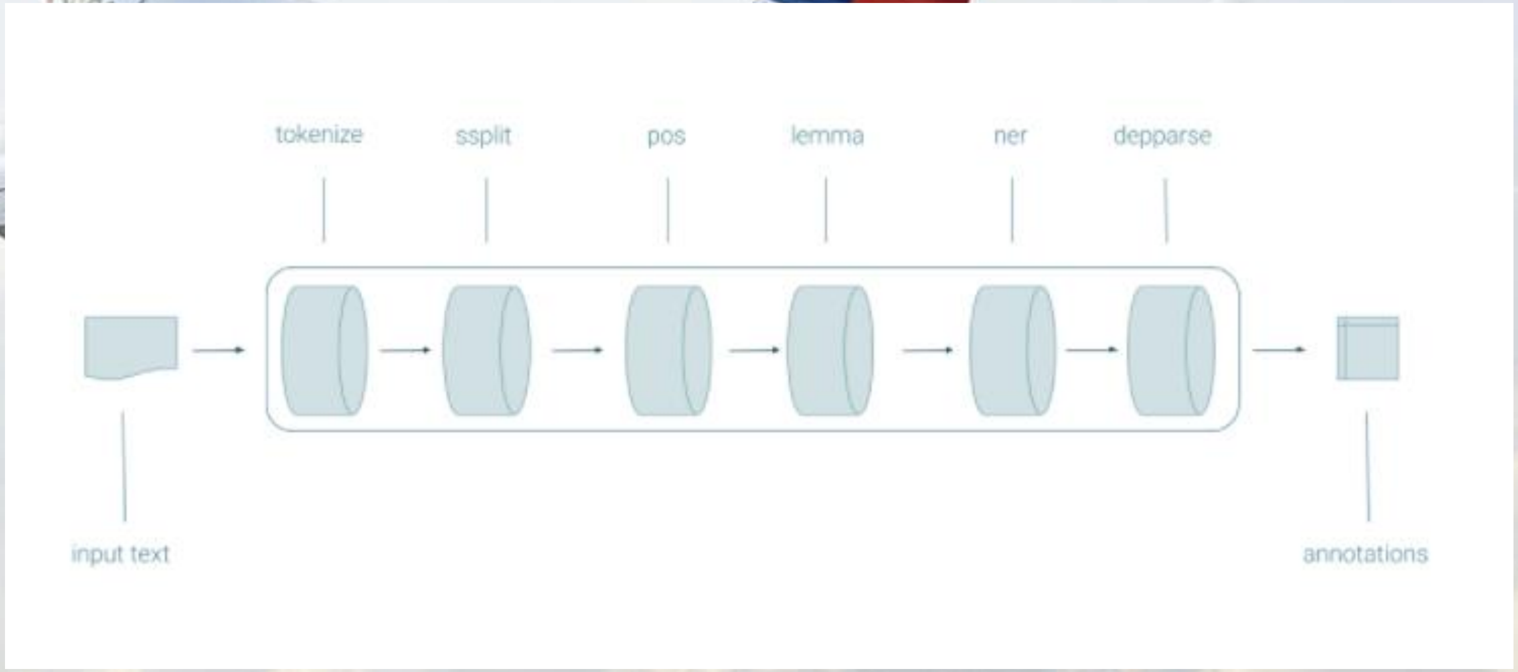


```
def main(args: Array[String]): Unit ={
  // set log level
  setupLogging()


  val cb = new ConfigurationBuilder
  cb.setDebugEnabled(true)
    .setOAuthConsumerKey(Configure.twitter.getString( path = "CONSUMER_KEY"))
    .setOAuthConsumerSecret(Configure.twitter.getString( path = "CONSUMER_KEY_SECRET"))
    .setOAuthAccessToken(Configure.twitter.getString( path = "ACCESS_TOKEN"))
    .setOAuthAccessTokenSecret(Configure.twitter.getString( path = "ACCESS_TOKEN_SECRET"))
    .setJSONStoreEnabled(true)


  //create kafka props
  val props = new Properties()
  props.put("bootstrap.servers", Configure.kafkac.getString( path = "BOOTSTRAP_SERVERS"));
  props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer"); // Str
  props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer"); // S

  val producer = new KafkaProducer[String, String](props)
  val kafkatopic = Configure.kafkac.getString( path = "TOPIC")
  val statuslistener = new StatusListener {
    /*
    StatusListener defines what to do with the tweets as they stream
    */
```

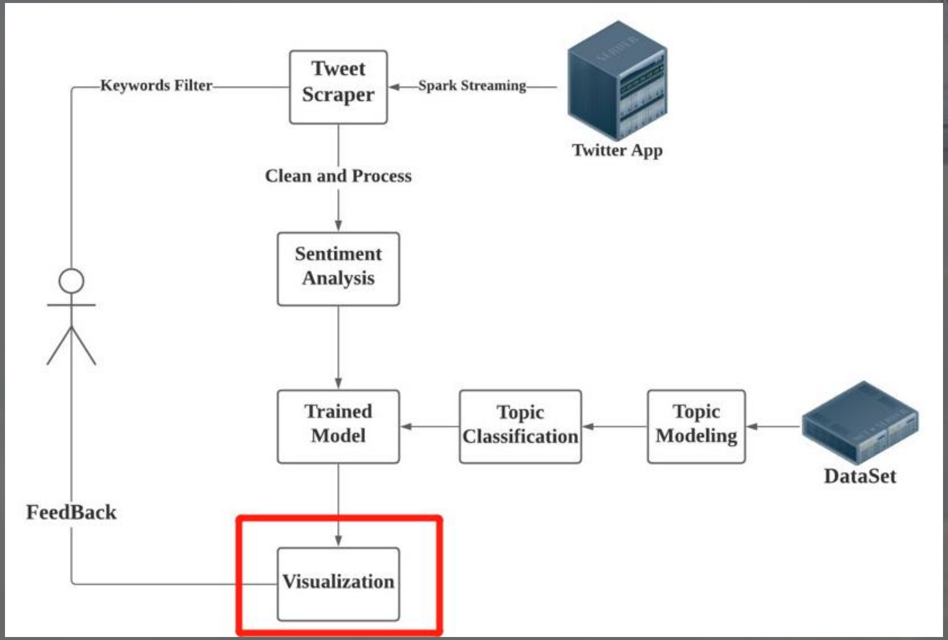
Sentimental analysis with CoreNLP

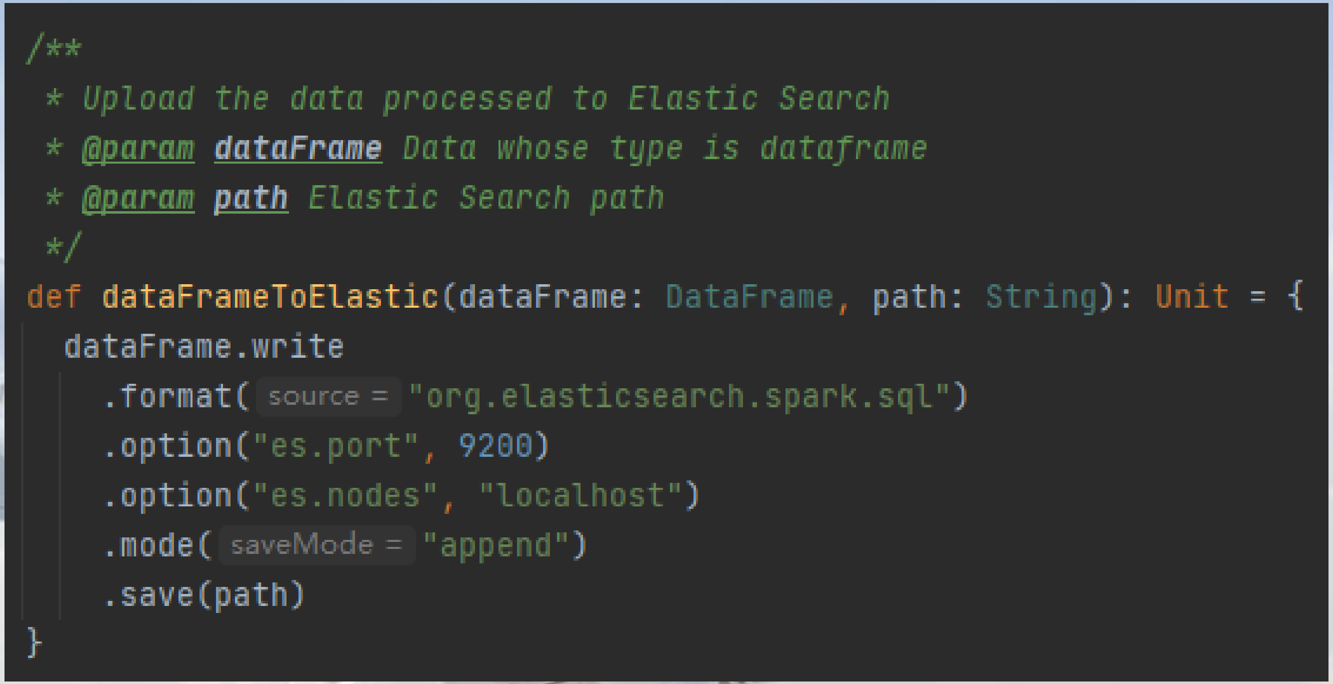


```
object CleanTweets {

  // Data from twitter would be like "RT @User: xxx", clean it
  def clean(input: String): String = input.substring(input.indexOf(": ") + 2)

}
```



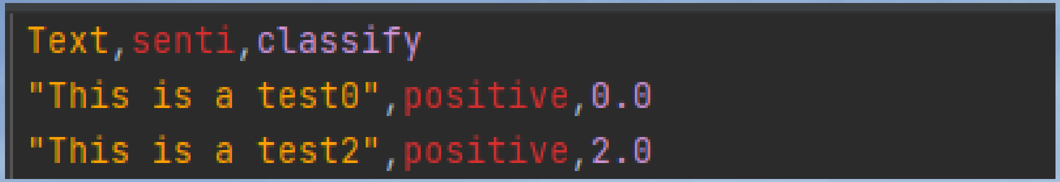
Upload to Elastic



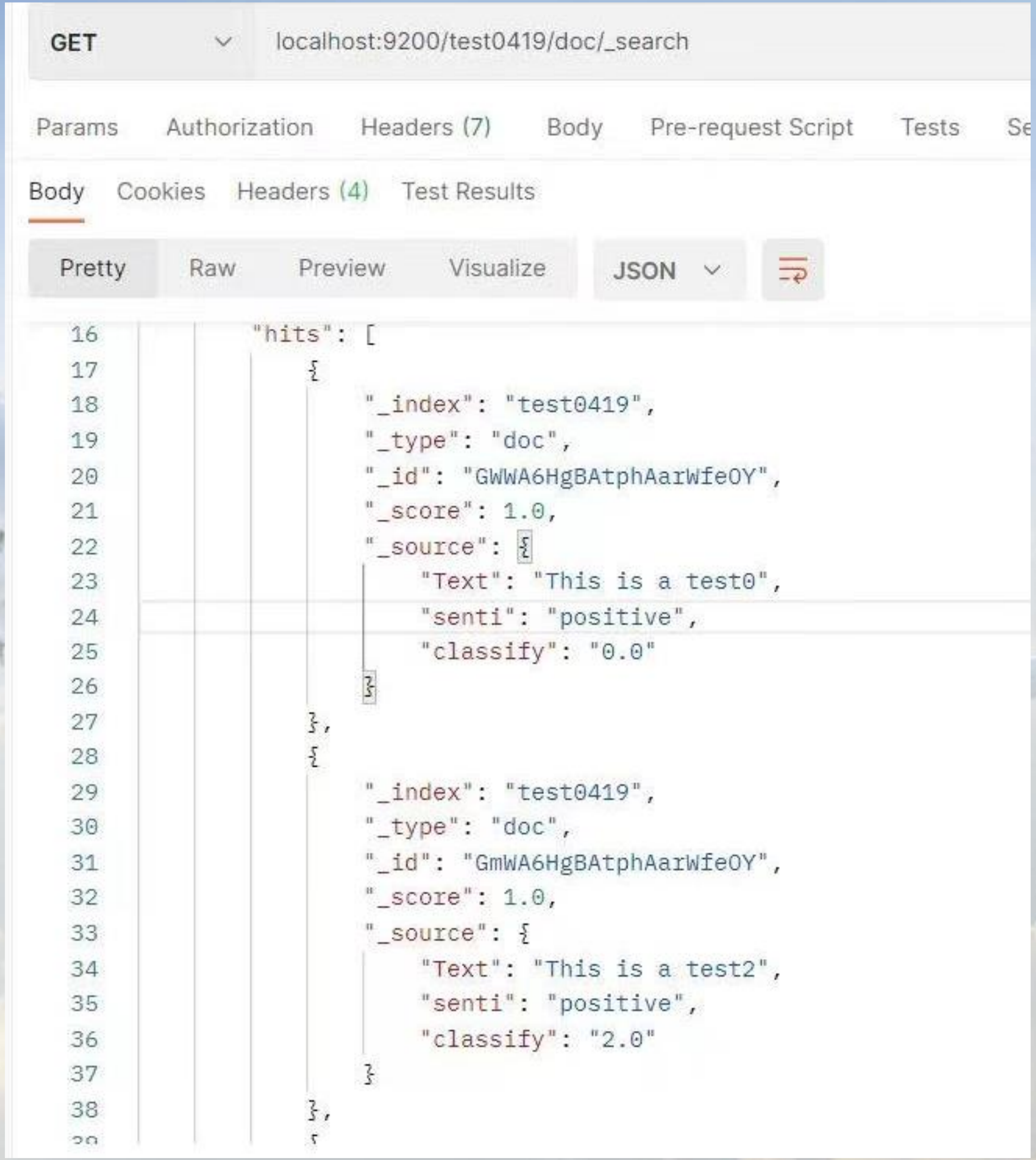


```
/**
 * Upload the data processed to Elastic Search
 * @param dataFrame Data whose type is dataframe
 * @param path Elastic Search path
 */
def dataFrameToElastic(dataFrame: DataFrame, path: String): Unit = {
  dataFrame.write
    .format( source = "org.elasticsearch.spark.sql")
    .option("es.port", 9200)
    .option("es.nodes", "localhost")
    .mode( saveMode = "append")
    .save(path)
}
```

Upload to Elastic

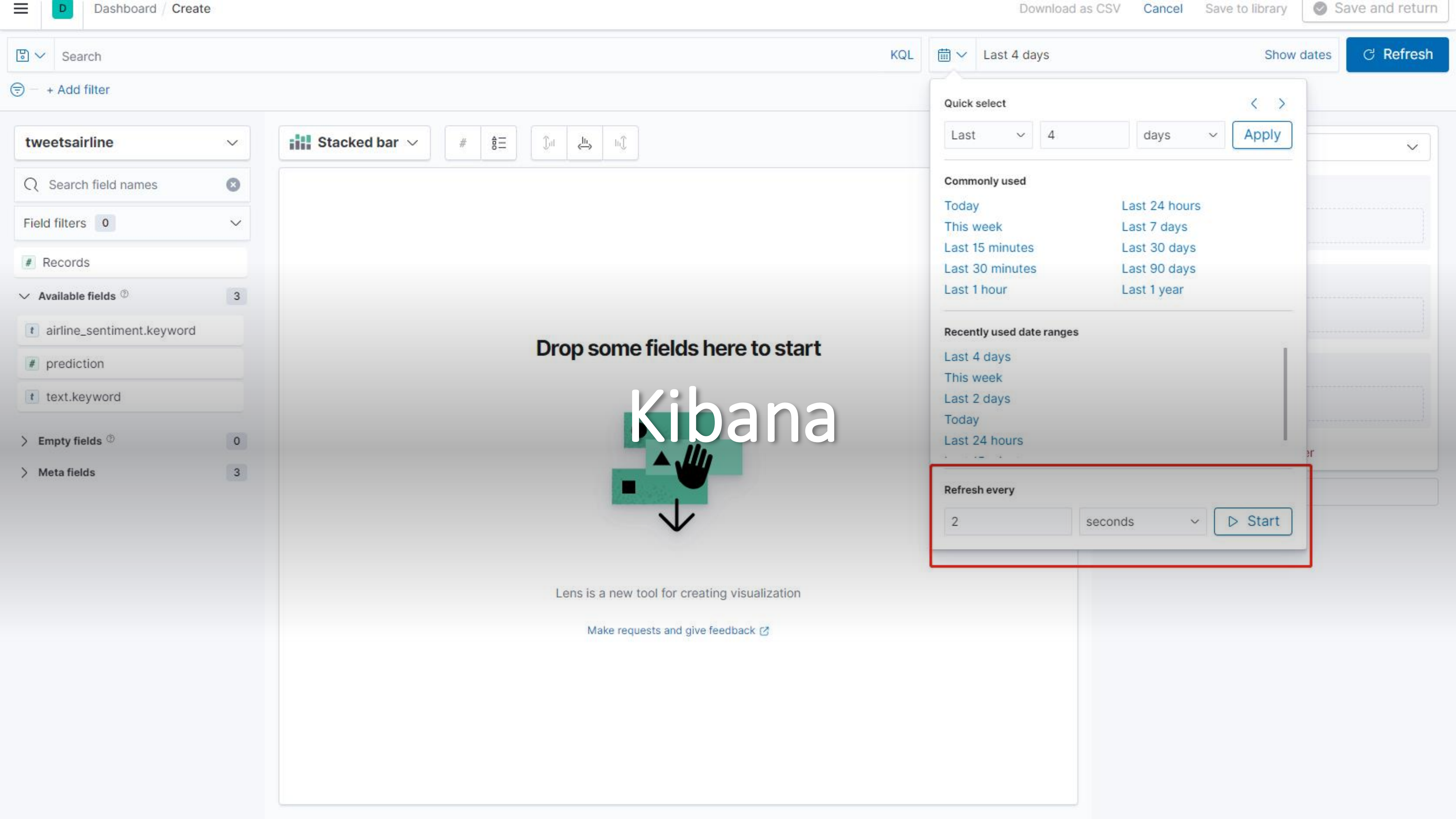


```
Text,senti,classify
"This is a test0",positive,0.0
"This is a test2",positive,2.0
```



GET localhost:9200/test0419/doc/_search

```
"hits": [
    {
        "_index": "test0419",
        "_type": "doc",
        "_id": "GWWA6HgBAtphAarWfeOY",
        "_score": 1.0,
        "_source": {
            "Text": "This is a test0",
            "senti": "positive",
            "classify": "0.0"
        }
    },
    {
        "_index": "test0419",
        "_type": "doc",
        "_id": "GmWA6HgBAtphAarWfeOY",
        "_score": 1.0,
        "_source": {
            "Text": "This is a test2",
            "senti": "positive",
            "classify": "2.0"
        }
    },
```



Kibana

Dashboard / Create

Download as CSV   Cancel   Save to library   Save and return

Search   KQL   Last 4 days   Show dates   Refresh

+ Add filter

tweetsairline

Search field names

Field filters 0

Records

Available fields 3

airline_sentiment.keyword

prediction

text.keyword

Empty fields 0

Meta fields 3

Stacked bar

Drop some fields here to start

Lens is a new tool for creating visualization

Make requests and give feedback

Quick select

Last   4   days   Apply

Commonly used

Today	Last 24 hours
This week	Last 7 days
Last 15 minutes	Last 30 days
Last 30 minutes	Last 90 days
Last 1 hour	Last 1 year

Recently used date ranges

Last 4 days

This week

Last 2 days

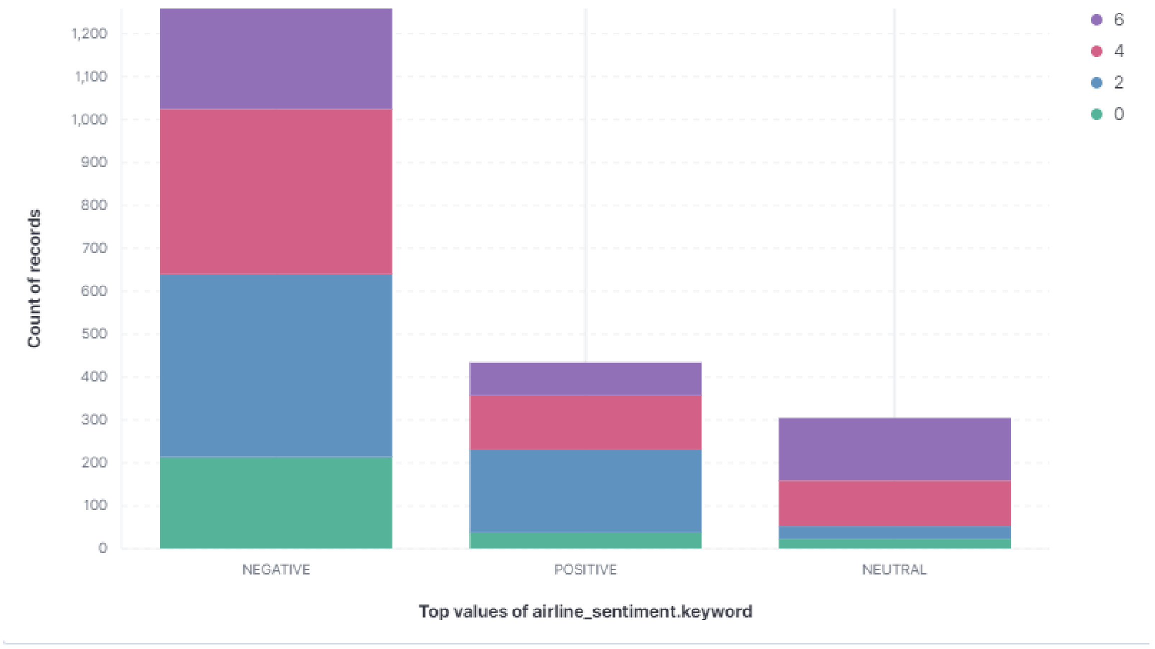
Today

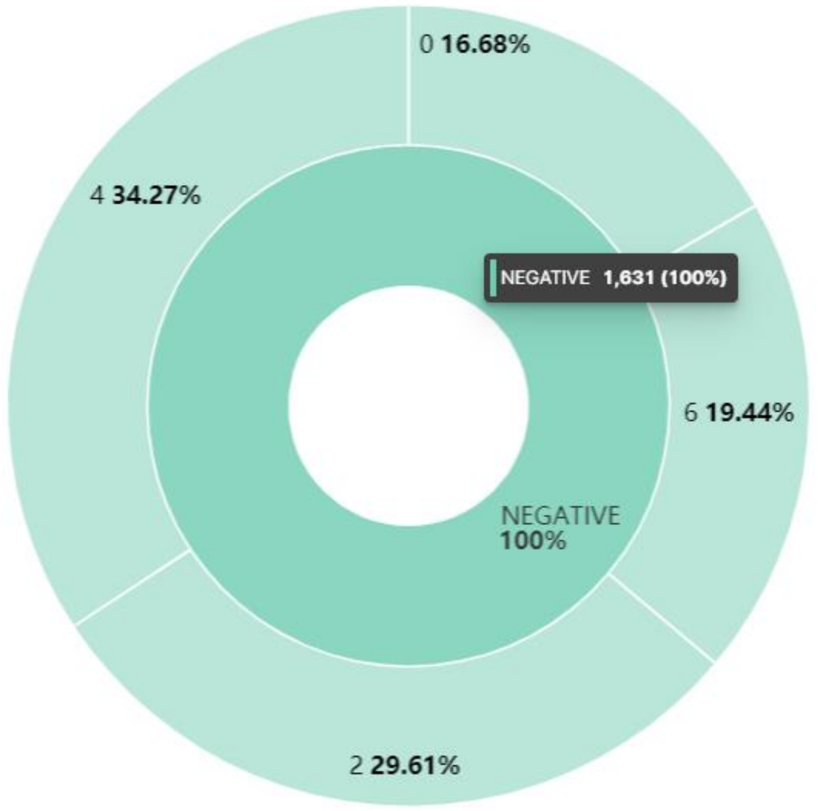
Last 24 hours

Refresh every

2   seconds   Start

Example


1,200
1,100
1,000
900
800
700
600
500
400
300
200
100
0
Count of records
NEGATIVE
POSITIVE
NEUTRAL
Top values of airline_sentiment.keyword
● 6
● 4
● 2
● 0


0 16.68%
4 34.27%
NEGATIVE 1,631 (100%)
6 19.44%
NEGATIVE
100%
2 29.61%

Thank you for watching !