**Website Classification using Spark Mlib**

We have used Apache Spark as they claim it to be up to a 100 times faster by leveraging the distributed memory of the cluster and by not being tied to the multi stage execution of Map/Reduce.

MLlib is Spark’s scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives Algorithms.

MLlib 1.1 contains the following algorithms:

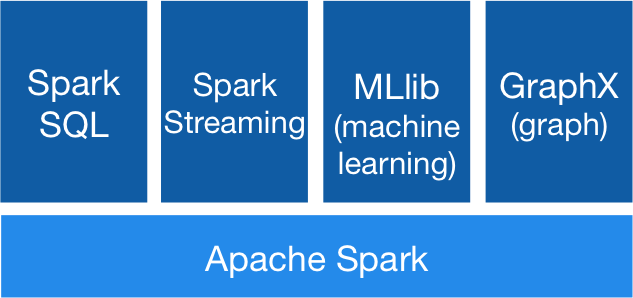
* linear SVM and logistic regression
* classification and regression tree
* k-means clustering
* recommendation via alternating least squares
* singular value decomposition
* linear regression with L1- and L2-regularization
* multinomial naive Bayes
* basic statistics
* feature transformations

We have used Naïve Bayes Algorithm to classify the websites into predetermined categories.

[**MLlib**](https://spark.apache.org/docs/latest/mllib-guide.html)**- Naive Bayes:**

[Naive Bayes](http://en.wikipedia.org/wiki/Naive_Bayes_classifier) is a simple multiclass classification algorithm with the assumption of independence between every pair of features. Naive Bayes can be trained very efficiently. Within a single pass to the training data, it computes the conditional probability distribution of each feature given label, and then it applies Bayes’ theorem to compute the conditional probability distribution of label given an observation and use it for prediction.

MLlib supports [multinomial naive Bayes](http://en.wikipedia.org/wiki/Naive_Bayes_classifier#Multinomial_naive_Bayes), which is typically used for [document classification](http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html). Within that context, each observation is a document and each feature represents a term whose value is the frequency of the term. Feature values must be nonnegative to represent term frequencies. [Additive smoothing](http://en.wikipedia.org/wiki/Lidstone_smoothing) can be used by setting the parameter λ (default to 1.0). For document classification, the input feature vectors are usually sparse, and sparse vectors should be supplied as input to take advantage of sparsity. Since the training data is only used once, it is not necessary to cache it.



**Training Data**

The popular Reuters 21578 collection of documents that appeared on Reuters newswire in 1987 as a training set. The Reuters collection can be obtained from http://archive.ics.uci.edu/ml/machine-learning-databases/reuters21578-mld/.

The collection has 123 categories. Each of the 22 files that the collection is made of, begins with a document type declaration line:

<! DOCTYPE lewis SYSTEM "lewis.dtd">

The DTD file lewis.dtd is included in the distribution. Following the document type declaration

Line are individual Reuters articles marked up with SGML tags, as described below.

Each article starts with an "open tag" of the form

<REUTERS TOPICS=?? LEWISSPLIT=?? CGISPLIT=?? OLDID=?? NEWID=??>

Where the ‘??’ are filled in an appropriate fashion. Each article ends with a "close tag" of the form:

</REUTERS>

In all cases the <REUTERS> and </REUTERS> tags are the only items on their line.

Each REUTERS tag contains explicit specifications of the values of five attributes, TOPICS, LEWISSPLIT, CGISPLIT, OLDID, and NEWID. But the only important attributes are TOPICS, NEWID and BODY these attributes are meant to identify documents and groups of documents, and have the following meanings:

**1. TOPICS**: The possible values are YES, NO, and BYPASS:

a. YES indicates that \*in the original data\* there was at least one entry in the TOPICS fields.

b. NO indicates that \*in the original data\* the story had no entries in the TOPICS field.

c. BYPASS indicates that \*in the original data\* the story was marked with the string "bypass" (or a typographical variant on that string).

**2. NEWID**: The identification number (ID) the story has in the Reuters-21578, Distribution 1.0 collection. These IDs are assigned to the stories in chronological order.

**3. BODY**: Contains the body of the html page which is being classified.

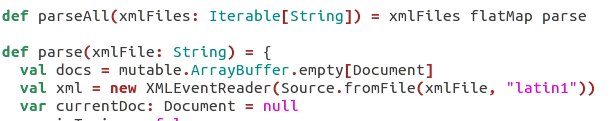
Then we added custom training set from following websites

* University
  + Wikipedia page
  + [www.utdallas.edu](http://www.utdallas.edu/)
  + [www.gtu.ac.in](http://www.gtu.ac.in/)
* Shopping Websites
  + Amazon
  + Ebay
  + Aliexpress
* Social Network websites
  + Facebook
  + Twitter
  + Instagram

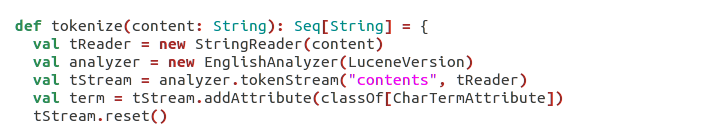
And it has the same format as the Reuters training dataset.

**Training the Naïve Bayes classifier:**

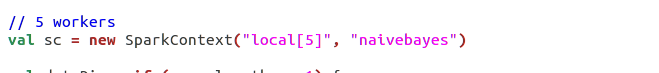
* Parsed the training XML data into labels and body
  + Simple API for XML parser (SAX parser)



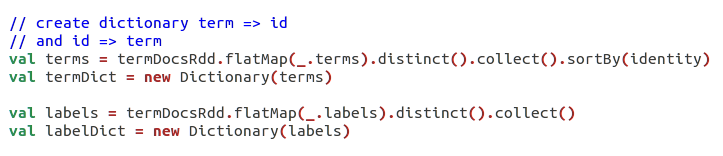
* Tokenized the training data using Apache Lucene
* And stemmed it using English Stemmer
  + Running and ran -> run



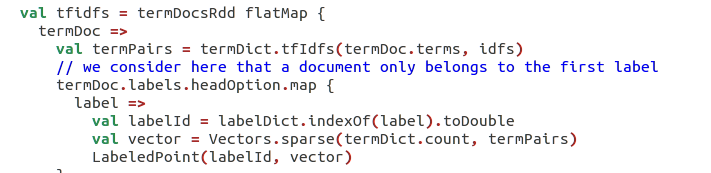
* Initialized the Spark context using 5 workers



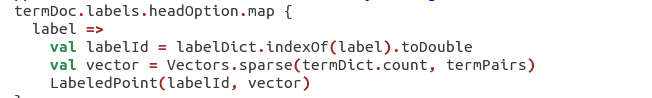
* Create term and label dictionary



* Based on dictionary calculated IDF based on each term
  + idf(term, docs) = (number of documents) / (number of documents containing term)
* Vectorize each document using TF-IDF score



* Convert into label points which represents label id and sparse vector using Mlib regression library
* Train Naïve Bayesian classifier





**Classification**

* Extract the web page content using Goose developed by gravity labs
* Read console input using jline console reader
* classify the web page content to a particular label using naïve Bayesian model that we train



* Input is url of website to classified



* Output label to which website is classified based on Naïve Bayesian model that we trained.

