# Multi-Class News Classification with Apache Spark

**Abhishek Vijay Bhave**
Graduate Student at
State University of New York at Buffalo
*abhave@buffalo.edu*
*UBIT Name:* ***abhave***
*UB Person No:* ***50289049***

**Omkar Sunil Thorat**
Graduate Student at
State University of New York at Buffalo
*omkarsun@buffalo.edu*
*UBIT Name:* ***omkarsun***
*UB Person No:* ***50290136***

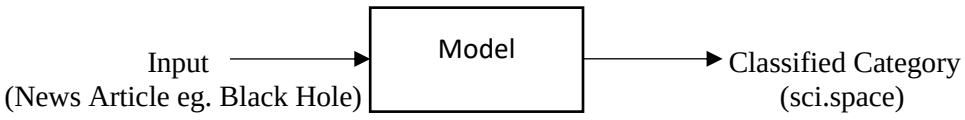
## Abstract:

In this age of Big Data where huge data is generated almost every second (289 TB to be precise), there is a growing need to harness this data to get information which would help an organization grow by developing personalizations, recommendation systems, and predictive insights using Machine Learning techniques. A prime example of this is Netflix, Amazon Prime and YouTube which gather and process data to recommend new videos to the user. Traditionally, these problems were solved using popular tools such as R and Python. However, these tools have their own limitations as they process data on a single machine which is generally not powerful enough to handle such huge volume of data. This is proven by the fact that the development in Machine Learning stalled in the early 90s due to the incapability of machines to perform computations. Spark provides a general machine learning library MLlib, that is designed for simplicity, scalability, and easy integration with other tools. With the scalability, language compatibility, and speed of Spark, data scientists can now solve and iterate through their data problems faster [1].

As part of this project, we are trying to classify a given news article into 20 pre-defined categories of "The 20 newsgroup text dataset" like politics, science, religion, recreational, etc. This dataset is available as part of the scikit-learn datasets. The dataset approximately has 18,000 news articles which are evenly partitioned across all the categories.

## Problem Statement:

Given a news article, we will assign it to one of the 20 categories. Here the underlying assumption is that each news article is assigned to only one category. We are trying to make this a multi-class text classification problem.



Apache Spark is one of the preferred frameworks for working with Big Data because of its scalability and ease of use while Python pips other programming languages when working with Machine Learning and Data Analysis. Combining these two together (PySpark) would provide us with a faster and efficient way to deal with huge datasets for Machine Learning. This is a major improvement over current/traditional way of using a single machine to work with huge datasets as movement of data becomes time consuming. Also, the analysis requires sampling which affects the accuracy.

We would be using MLib library provided by PySpark for our use case. After preprocessing our data, we will convert text articles into numerical feature vectors by using Bag of Words model. Next step would be to decide which model among various multi-class classification models provided by MLib to use. We can choose among Multinomial Logistic Regression, Naïve Bayes Classification, Decision Trees, Random Forest and Support Vector Machines. We will finally feed the data to the chosen model to perform multi-class text classification.

## Methodology:

Classification problems in Machine Learning require feature vectors as input. Our dataset is a collection text files which correspond to each news article. We cannot feed data in this form and need to extract features from the text files. To extract these features, we will perform the following operations:

### A. Tokenization:

"Tokenization is a task of breaking up natural language text into distinct meaning units (tokens)" (Kaplan, 2005), which usually consist of words and symbols. Thus, we tokenize the entire text file into various tokens to extract features.

### B. CountVectors:

Now that we have tokenized the data, the next step is to count all occurrences of each word and assign an integer to each word. All the distinct words will correspond to features of our dataset. We can do this using Bag of Words model. This can be done by using '*CountVectorizer*' method of the '*pyspark.ml.feature*' library also called as TF-IDF.

### C. Remove Stop Words:

CountVectors however has a small drawback in the sense that it gives more weightage to larger text files as compared to smaller text files. Also, we need to reduce the weightage of stop words like 'a', 'the', 'it', etc. This can be done by using '*StopWordsRemover* method of the '*pyspark.ml.feature*' library. Finally, we have the required input features needed to feed to our model.

## Pipeline and Architecture:

In machine learning, it is a common practice to run a sequence of algorithms to process and extract features from the data. We can consider our case where we have done various steps as part of processing workflow.

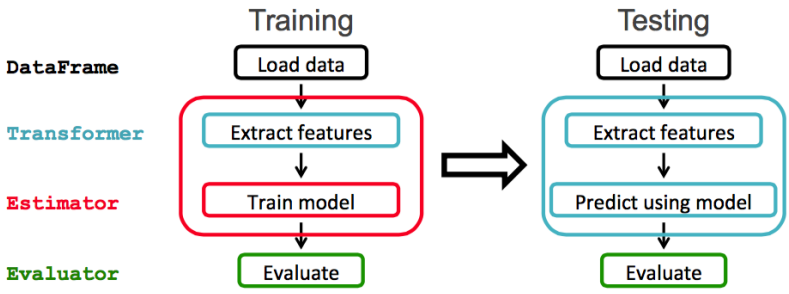



**Fig. 1 ML Pipeline**

Such a workflow is represented as a Pipeline in MLib. A Pipeline is a sequence of steps called as PipelineStages (Transformers and Estimators) which are executed in a predefined order where input data is modeled as a Dataframe which gets transformed at every stage in the Pipeline. For Transformer stages, the transform() method is called on the DataFrame whereas for Estimator stages, the fit() method is called to produce a Transformer.

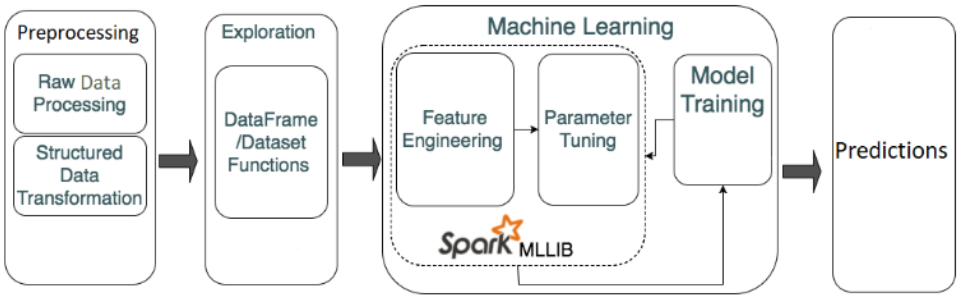



**Fig. 2 Spark ML Architecture**

Apache Spark is fast, easy-to-use and general engine for big data processing that has built-in modules for streaming, SQL, Machine Learning (ML) and graph processing. This makes the processing of data faster which helps in training the model as the data is distributed.

## Data:

We are using “The 20-newsgroup” text dataset which has news articles classified into 20 pre-defined categories like politics, science, religion, recreational, etc. This dataset is available as part of the scikit-learn datasets. The dataset approximately has 18,000 news articles which are evenly partitioned across all the categories. The following table displays all the 20 categories.

| | | |
|---|---|---|
| comp.graphics<br>comp.os.ms-windows.misc<br>comp.sys.ibm.pc.hardware<br>comp.sys.mac.hardware<br>comp.windows.x | rec.autos<br>rec.motorcycles<br>rec.sport.baseball<br>rec.sport.hockey | sci.crypt<br>sci.electronics<br>sci.med<br>sci.space |
| misc.forsale | talk.politics.misc<br>talk.politics.guns<br>talk.politics.mideast | talk.religion.misc<br>alt.atheism<br>soc.religion.christian |

**Fig. 3 Table showing 20 categories**

Some of the newsgroups categories are very closely related to each other while there are many which are completely unrelated to each other. To access the dataset, we can either download it from the source or we can get it as a part of the sklearn dataset.

## Algorithm and Implementation:

Classification is a technique for determining the class that the dependent belongs to based on the one or more independent variables. There are various Classification Algorithms in the Machine Learning Domain that can work well with our dataset.

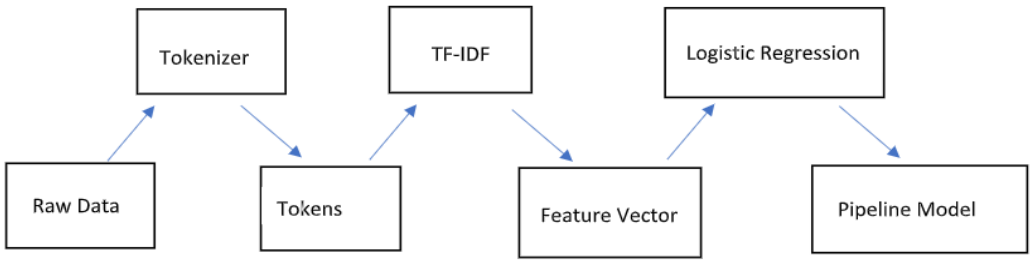



**Fig. 4 Flowchart**

### A. Multinomial Logistic Regression

Logistic regression is a probabilistic statistical classification model. It includes all the points within a dataset to find the separating hyperplane. It works well when there are lesser number of input features. Softmax regression is a method in machine learning which allows for the classification of an input into discrete classes. Softmax allows for classification into multiple number of possible classes.

We train the model based on the extracted feature vectors using the *.fit* method. Once the model is trained, we use the *.transform* method to test our model. This method will classify the given news article in one of the 20 news groups. The accuracy of the model is obtained using the *.evaluate* method.

### B. Naïve Bayes

This method is based on Bayes rule. It relies on Bag of Words representation of the document. To use this model, we first process the document (using Bag of Words model). For each label y (20 news group), we construct a probabilistic model using conditional probability. The class that gives the highest probability, is the class to which the news article belongs to.

### C. Random Forest

Random forests are an ensemble learning method for classification. They operate by constructing a large number of decision trees at training time and outputting the class that is the mode of the classes of the individual trees as the final prediction. spark.mllib supports random forests for binary and multiclass classification and for regression, using both continuous and categorical features.

### D. Support Vector Machine:

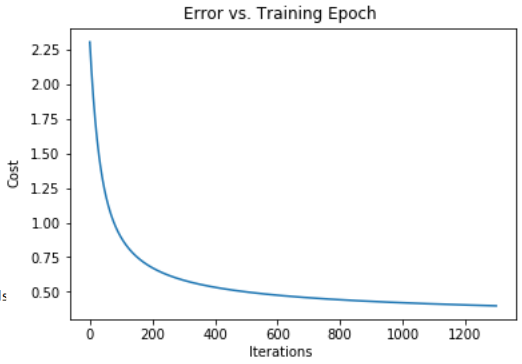
Support vector machine constructs a set of hyper planes in an infinite dimensional space. SVM works well when the number of input features are high. A good classification is achieved by the hyperplane that has the largest distance to the nearest training-data points of any class. This can be shown by the fact that the larger the margin, the lower is the generalization error of the classifier. LinearSVC in the Spark MLib supports binary classification. LinearSVC is used iteratively over each category to make one-vs-Rest classifier.

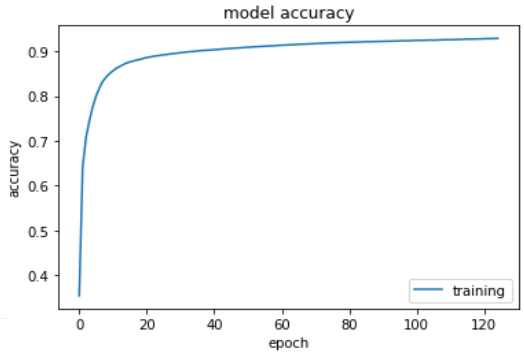
### Analysis of Algorithms:

Our dataset has comparatively less features, and Logistic Regression and Naïve Bayes implementations work well with lesser features. Thus, Logistic Regression and Naïve Bayes classification work best for our use case of multinomial text classification.

## Code Snippets and Output:

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer
from sklearn.datasets import fetch_20newsgroups
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer

trainingData = fetch_20newsgroups(subset='train', shuffle=True)
testData = fetch_20newsgroups(subset='test', shuffle=True)


regexTokenizer = RegexTokenizer(inputCol="Descript", outputCol="words", pattern="\\W")

add_stopwords = ["and","is","this","at","the"]
stopwordsRemover = StopWordsRemover(inputCol="words", outputCol="filtered").setStopWords(add_stopwords

countVectors = CountVectorizer(inputCol="filtered", outputCol="features", vocabSize=10000, minDF=5)


label_stringIdx = StringIndexer(inputCol = "Category", outputCol = "label")
pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors, label_stringIdx])
# Fit the pipeline to training documents.
pipelineFit = pipeline.fit(trainingData)
dataset = pipelineFit.transform(trainingData)

lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
lrModel = lr.fit(trainingData)
predictions = lrModel.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("Descript","Category","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)


evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator.evaluate(predictions)
```




As we can see from the "error vs training" graph, loss decreased as we trained the model for more epochs and as a result of which the accuracy increased as seen in "accuracy" graph. Our model can classify a given news article to the correct category with about 94.65% accuracy.

## Summary:

**a.** The Spark framework makes the task of Machine Learning easier as we have to focus on solving the problem and not worry about platform integrations and putting everything together.

**b.** By making use of Spark, we can run all of our processes like data processing, machine learning and distributed computing in one framework which makes Spark more convenient.

**c.** In the Apache Spark framework, we bring the models to our data which gives us a better performance as well as scalability and allows for processing of large volume of data with ease.

**d**. Working on Spark helped us learn new facets of working with huge datasets.

## References:


[1] https://www.infoworld.com/article/3031690/why-you-should-use-spark-for-machine-learning.html

[2] https://spark.apache.org/docs/latest/ml-pipeline.html

[3]https://scikitlearn.org/0.19/modules/generated/sklearn.datasets.fetch_20newsgroups.html#sklearn.datasets.fetch_20newsgroups

[4] http://qwone.com/~jason/20Newsgroups/

[5] https://towardsdatascience.com/multi-class-text-classification-with-pyspark-7d78d022ed35