DATA ANALYTICS PIPELINE USING APACHE SPARK

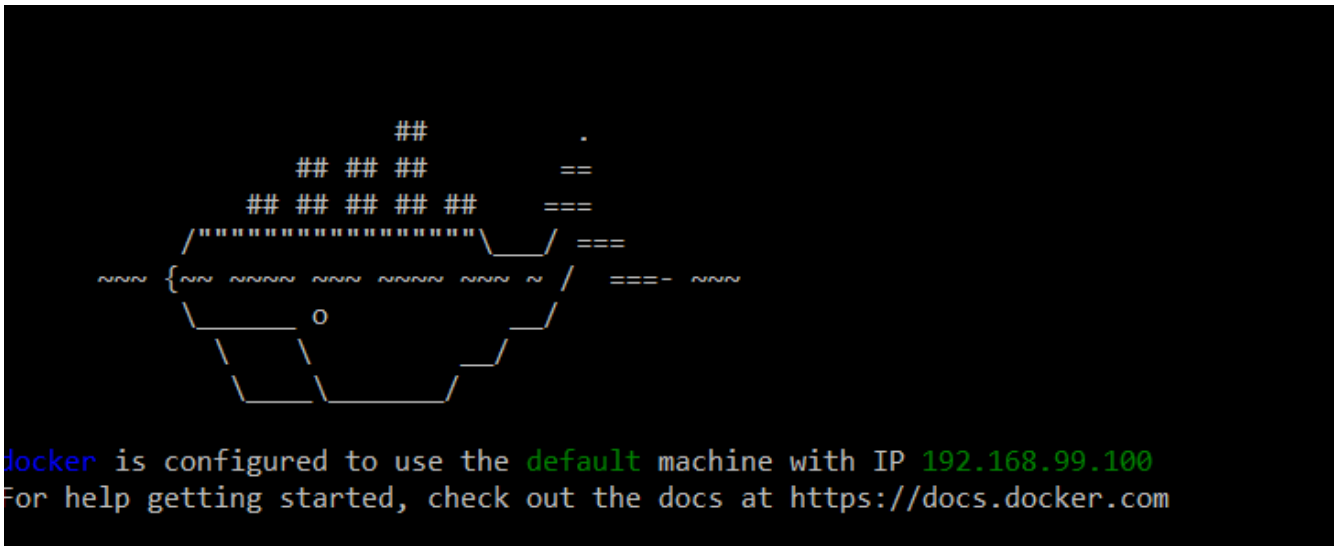
1. **Tejas Dhrangadharia (tejassha)**
2. **Karan Nisar (karankir)**

Environment:

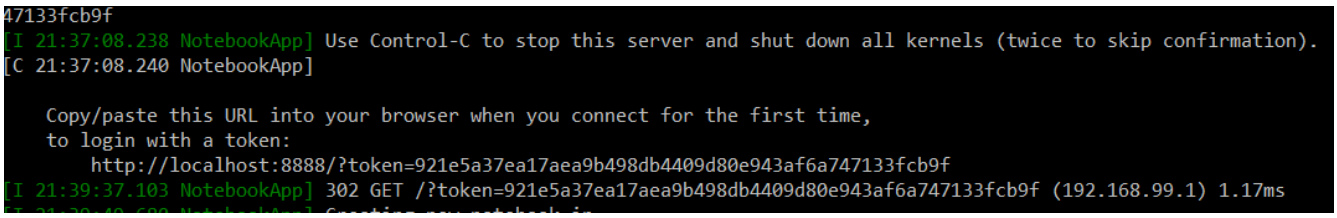
1. Python 3.6.4
2. PySpark 2.3.0
3. Jupyter Notebook 5.4.1
4. Docker 18.03.0-ce

Run this program:

1. Install Python 3.6.4 (1)
2. Install Docker (2)
3. Run Docker Quickstart Terminal
4. Install PySpark Jupyter Notebook (`docker run -it --rm -p 8888:8888 jupyter/pyspark-notebook`)
5. After successfully installation, navigate to the URL given in the terminal after changing the localhost with the IP address given
   IP Address:



   URL:



6. Upload the two notebooks and data
   a. dhrangadhariaLab3Part1.ipynb

b. dhrangadhariaLab3Part2.ipynb
c. train.csv (Titanic Data)
d. 4 folders (“Sports”,” Politics”,” Business”,” Movies”)

7. Part 1 is for understanding apache spark with Titanic data analysis.
8. Part 2 includes data collection, data cleaning, feature engineering, Multi-class classification and testing.

Explanation:

1. Understand Apache Spark with Titanic data analysis
   We collected the Titanic data from Kaggle (3). And we clear the data by dropping the null values from the data and then we divided the data into train(70%) and test(30%) randomly.
   We performed the feature engineering in which we index the labels and add the metadata to the label column. Train a Random Forest model with 20 trees and create a pipeline.
   Train the model using train data and test the model using testing data. Calculated the test error, Random Forest accuracy. And below is the output,

```
+----------+--------+--------------------+
|prediction|Survived|            features|
+----------+--------+--------------------+
|       1.0|       0|(7,[0,2,4],[3.0,2...|
|       1.0|       0|[2.0,0.0,26.0,1.0...|
|       1.0|       0|[2.0,0.0,38.0,0.0...|
|       1.0|       0|[3.0,0.0,2.0,0.0,...|
|       0.0|       0|[3.0,0.0,2.0,4.0,...|
+----------+--------+--------------------+
only showing top 5 rows

Test Error = 0.133005
RandomForestClassificationModel (uid=RandomForestClassifier_4dd98a44267f44f9bb6c) with 20 trees
Accuracy = 0.866995
f1 = 0.866846
weightedPrecision = 0.866741
weightedRecall = 0.866995
```

2. Collect and Clean data
   We collected articles URL from NYTimes using the API. And then wrote the scrapper for collecting the articles content. We collected the approximate 150 articles for each category (Sports, Business, Politics and Movies) so, total 600 articles. We stored each articles in separate text files in 4 different folders.
   For tokenizing and cleaning the data, we used Apache Spark. Removed the NA values from the data and using `RegexTokenizer` and `StopWordsRemover` we clean the data.

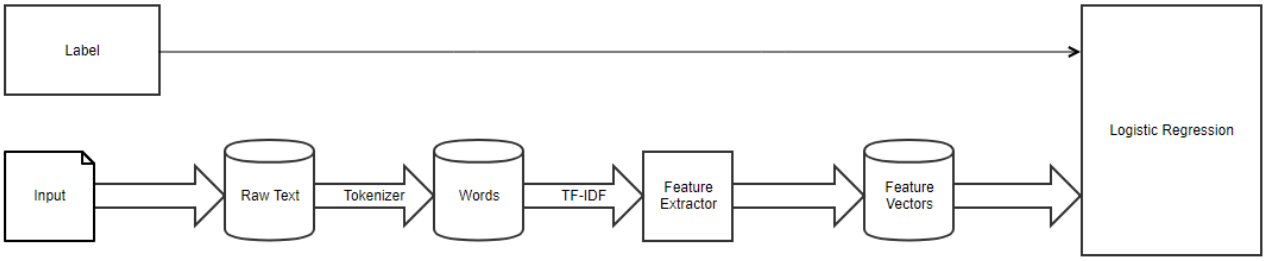
3. Feature Engineering
   We extracted the features characterizing the category. Using `HashingTF` and `IDF` we extracted the features. We build the pipeline and execute these stages sequentially. For better understanding see block diagram below.

4. Multi-class Classification
   We used Logistic regression and Naïve Bayes machine learning methods for classification. We fit the model using train data. Using that trained model, we evaluate it on the test data (Note:

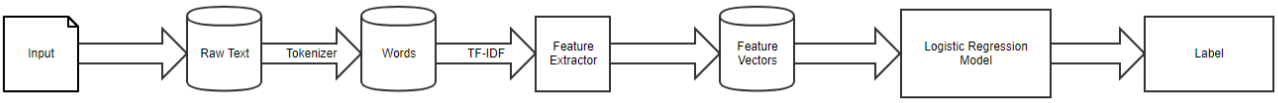
test data doesn't have any labels). We calculate the accuracy by comparing the predicted values and actual values. And below is the complete pipeline for the train and test data set for both the machine learning algorithms.

Block Diagram - Logistic Regression

(a) Training



(b) Prediction



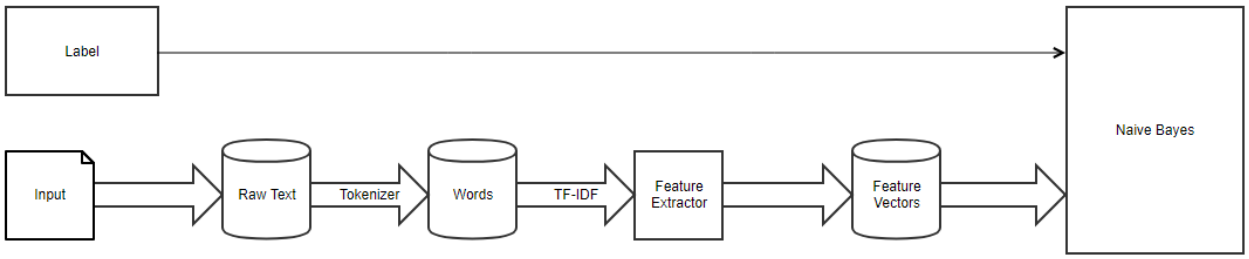
Logistic Regression ¶

```
In [372]: # Fit logistic regression model
          lr=LogisticRegression(featuresCol="features",labelCol="label",maxIter=20)
          lx=lr.fit(train)

In [373]: # Test the model using test data
          ly=lx.transform(test)

In [374]: evaluator=MulticlassClassificationEvaluator()

In [375]: # Evaluate the model
          evaluator.evaluate(ly)

Out[375]: 0.7156237232501345
```

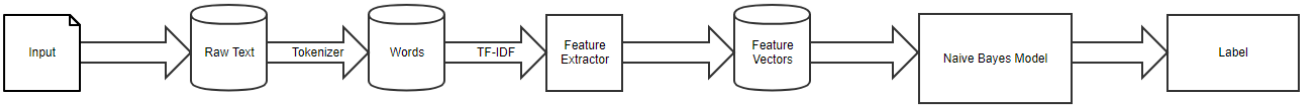
Accuracy: 71.56%

Block Diagram - Naïve Bayes

(a) Training



(b) Prediction




```
In [377]: # Fit logistic regression model
          nb=NaiveBayes(smoothing=1.0)
          mod=nb.fit(train)

In [379]: # Test the model using test data
          model = mod.transform(test)

In [380]: # Evaluate the model
          evaluator=MulticlassClassificationEvaluator()
          evaluator.evaluate(model)

Out[380]: 0.7260387782876818
```

Accuracy: 72.60%

5. Testing

We scrap 20 new articles (not test set) and then give that data as an input to our both the trained models (Logistic Regression and Naïve Bayes) and evaluate the performance of both the machine learning methods.

Logistic Regression

```
In [459]: ly=lx.transform(inx)
          evaluator=MulticlassClassificationEvaluator()
          evaluator.evaluate(ly)

Out[459]: 0.5500619233315466
```

Accuracy: 55.00%

Naive Bayes

```
In [460]: model = mod.transform(inx)
          evaluator=MulticlassClassificationEvaluator()
          evaluator.evaluate(model)

Out[460]: 0.5572045986833497
```

Accuracy: 55.72%