SPARK DOCUMENTATION

DATA

- ***Data:*** 400 Articles urls are pulled from NYTimes using the NYTimes api. Articles from Multiple category like politics, buisness, sports, entertainment and science are chosen.

  Using Beautiful soup, we pull the text from the web urls and they are stored as dictionary with the category. Articles from other categories are dropped. Finally we have approximately 250 news articles with the text data and category.
- This data is split into training set and testing set in the ratio 65:35

FEATURE ENGINEERING

- SqlContext and SparkContext are created to initialize the spark

Tokenization: Text data we have is of string data type hence tokenization makes a list of words.

Remover: some common words are to be removed from the data as a part of preprocessing of data.

BUILDING PIPELINES USING IDF, HASHINGTF TO CHECK FOR ACCURACY

- Accuracy using HashingTF is more as compared to IDF. (Accuracy using HashingTF is 79.26% whereas using IDF is 65.62%). so HashingTF pipeline is further used in all the models and only retained in the downstream pipeline

  HashTF: This calculates the feature vectors, basically calculates the word frequency.

  LabelStrIndex: Each category is assigned a number

MULTI-CLASS CLASSIFICATION USING LOGISTIC REGRESSION AND CROSSVALIDATION

- ***Logistic Regression***: Logistic regression is performed on the training data and cross validation is done where we have 6 features to choose from.
- ***Testing:*** prediction is done on the testing data, where we predict the category of news article based on the text that the article contains.
- ***Evaluator:*** Evaluater verifies each category predicted with the actual category and we obtain the accuracy of the model.

MULTI-CLASS CLASSIFICATION USING RANDOM FORESTS

- ***Random Forest:*** RandomForestRegressor command is used to build a random forest model and training data is used to train the model. We obtain a 67.2789% accuracy on the training data.
- ***Testing:*** prediction of category is done for the test data. And evaluator gives the accuracy of the model.

RESULTS

Accuracies obtained for Logistic Regression

Using HashingTF:

The test accuracy obtained using Logistic regression: 0.7960420289784099

The training accuracy obtained using Logistic regression: 0.8105997920846342

Using IDF:

The test accuracy obtained using Logistic regression: 0.65.620640299

The training accuracy obtained using Logistic regression: 0.61.4768546342

Accuracies obtained for Random Forests:

Using HashingTF:

The training accuracy obtained using Random forests: 67.2789

The test accuracy obtained using Random Forests is: 64.284977

Using IDF:

The training accuracy obtained using Random forests: 55.8219

The test accuracy obtained using Random Forests is: 51.54724

Results:

The Logistic Regression has given the best results when compared with 2 different algorithms: Random Forests.

The HashingTF has given better accuracy values in the pipeline part and used the same for feature engineering while tuning as compared against IDF

DIAGRAMS

Figure: Data Extraction

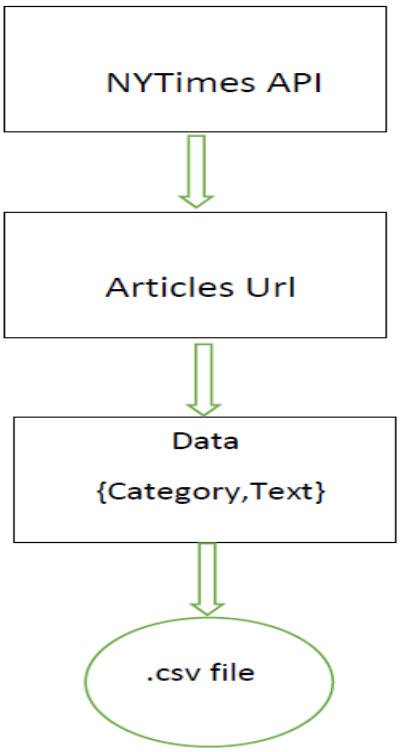
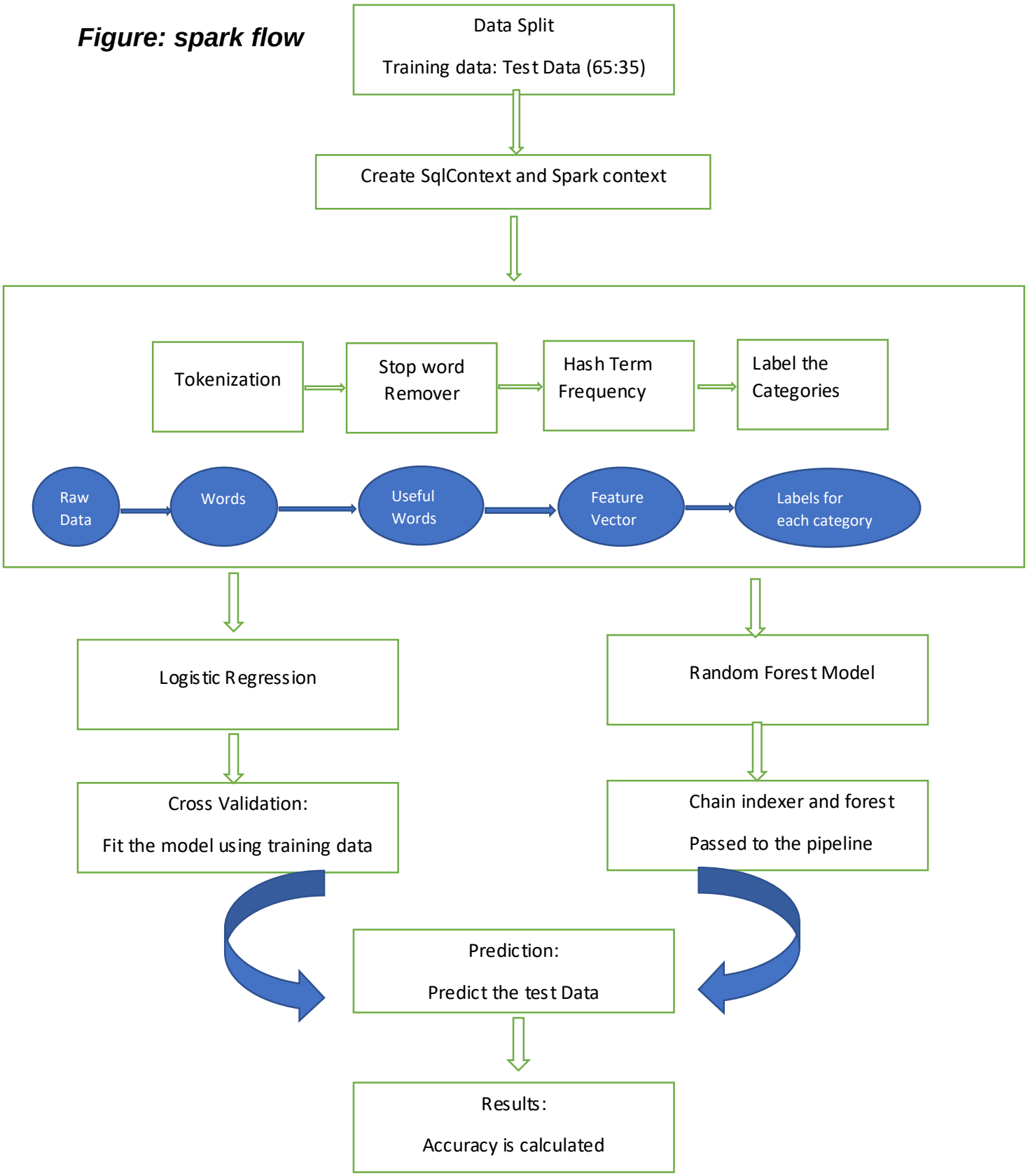


figure: Data Extraction

Figure: spark flow



Results ScreenShots

Part 1: Titanic

```
print("weightedRecall = %g" % wr)
```

```
+----------+--------+--------------------+
|prediction|Survived|            features|
+----------+--------+--------------------+
|       1.0|       0|[3.0,0.0,9.0,1.0,...|
|       1.0|       0|[3.0,0.0,18.0,0.0...|
|       1.0|       0|(7,[0,2,4],[3.0,2...|
|       1.0|       0|(7,[0,2,4],[3.0,3...|
|       1.0|       0|[3.0,0.0,32.0,1.0...|
+----------+--------+--------------------+
only showing top 5 rows

Test Error = 0.194313
RandomForestClassificationModel (uid=RandomForestClassifier_497db5b82de985a1a769) with 20 trees
Accuracy = 0.805687
f1 = 0.804944
weightedPrecision = 0.806001
weightedRecall = 0.805687
```

In [ ]:

Part 2: Results Screenshot

```
print('The test accuracy obtained using Logistic regression is: ',test_error)
```

```
The test accuracy obtained using Logistic regression:  0.7960420289784099
```

In [133]:

```
prediction = cvModel.transform(train)
selected = prediction.select( "label", "prediction")
evaluator=MulticlassClassificationEvaluator(predictionCol='prediction')
train_error=evaluator.evaluate(prediction)
print('The training accuracy obtained using Logistic regression: ',train_error)
```

```
The training accuracy obtained using Logistic regression:  0.8105997920846342
```