Course Code: AIML427

Course Title: Big Data

Assignment 3 – Individual Part 2

George Wiles

Student ID: 300586465

Due Date: 8 June 2021, 17:00 pm

Date of Submission: 8 June 2021

1. Individual Machine Learning Task – AG News Classification Dataset

a) Task Description and Overview

According to Anand [1], AG is a collection of more than 1 million new articles gathered by the academic news search engine ComeToMyHead. The dataset contains 3 columns: the class index or response variable, and two columns of text representing the article title and the article description.

The AG News Classification dataset is an example of a multiclass classification problem, where the response variable ('Class Index) is a value that represents 4 news category classifiers (1-World, 2-Sports, 3-Business, 4-Science and Technology). Penchikala [2] states machine learning steps as: Data Ingestion, Data Cleaning, Feature Extraction, Model Training, Model Validation, Model Selection and Model Deployment. This assignment applies the Data Ingestion through to the Model Validation stages, creating two parameterized programs for Logistic Regression and Decision Tree classification using Spark ML (machine learning libraries) executed in a Spark Hadoop Cluster and varying a PCA dimensionality reduction technique to the large TF-IDF feature space for the article Title and Description original data features. The Data Cleaning and Feature Extraction stages uses TF-IDF to vectorize both the article Title and Description text fields to evaluate how important each word (or feature) is in the AIG News Corpus to predict the news category response variable class.

Original Dataset Size

The AG News training set contains 120,00 training dataset instances (29 MB), and the test set contains 7600 testing samples (1.8 MB).

Number of Features

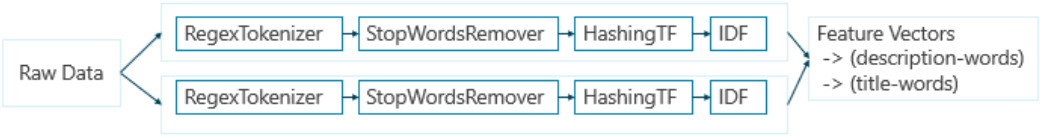
The Kaggle dataset [1] retrieved at https://www.kaggle.com/amananandrai/ag-news-classification-dataset has been pre-processed to ensure that there are no missing columns, the text and description columns contain quotes, commas and new line characters that have been escaped prior to the Data Ingestion step discussed by Penchikala [2]. It is assumed a typical text article can contain 500-800 words, which when applied to a TF-IDF (term frequency-inverse document frequency) numerical statistic, this implies a vectorized feature model containing both title and description fields can contain up to 1500 unique words or features during the Data Cleaning and Feature Extraction pipeline stages.

Expected Output

As stated this assignment uses two algorithms, logistic regression and decision tree classification models, executed on a spark Hadoop cluster to predict the multiclass response variable of the test set against the trained model using the training data set.

b) Pre-Processing Step

As noted in section (a), the AG News dataset selected [1] has been pre-processed as part of the Data Ingestion step ensuring there are no missing data instances in the column data and the textual data for Title and Description has been largely escaped so it can be successfully read as a CSV file. The Data Cleaning and Feature Extraction stages as discussed by Penchikala [2], contains the following Spark ML pipeline transform steps of the original features to word feature vectors for the AIG News Dataset.



While the original dataset contains 2 features being the Title and Description columns, the decision tree and linear regression programs take a feature count as a parameter, this was used to limit the word vectorization of Title and Description respectively represent these text fields as 500-800 features each respectively. The respective word features are cleaned using regex, stop words transformers to reduce the non-word/irrelevant features dimensionality and combining the word features using a Spark vector assembler.

a. 1. Regex Tokenizer to
    i. Remove white space characters
    ii. Set all word/features as lower case to remove duplication
    iii. Set a minimum length token of 1 character to reduce punctuation.
    |Reuters - Stocks …| - to - |[reuters, stocks,...|
b. Stop words remover, by default the Spark ML StopWordsRemover contains a list of common stop words (and, of, to et al).
    |stocks, assets, of| - to -.|[stocks, assets, nati...|
c. Hashing term frequency (HashingTF vs CountVectorizer, both were investigated with as a pre-fitting processing data transform step.
d. NOTE: Title word vectors and sentence word vectors are given the same weighting by the Spark ML transformers above. Intuitively it would make sense to apply different weights to relevant words in the title compared to the description field and also test relevance counts of words the appear in both the sentence and title description as more predictive then words that only appears in a single word vector.

c) Program Description for both Linear Regression and Decision Tree

The Linear Regression and Decision Tree classification programs follow the same pattern for the AIG New Classification problem.

Pseudo code/execution flow of the programs are as follows:

- Both programs accept a set of command line variables allowing the variance of different aspects of the Data Cleaning and Feature Extraction stages [trainingFile, testFile, randomSeed, numFeatures, pcaFlag, numPCAFeatures, treeMaxDepth (for Decision Tree)]
- The full training and test file are read in using the spark session csv reader.
- The Data Cleansing and Feature Transformation modules are created for the raw Title and Description Colums respectively and addes as stages into the Spark ML Pipeline.

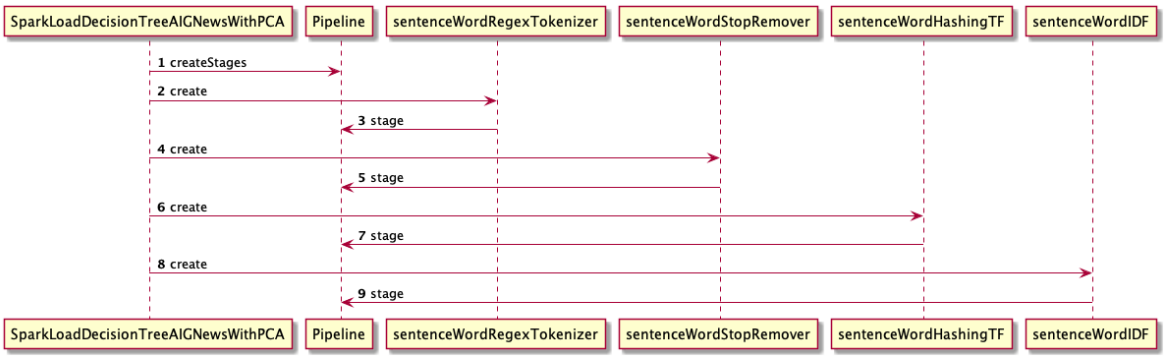
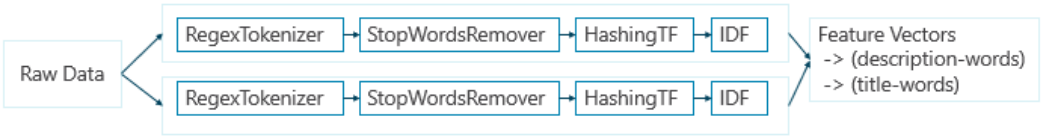


Diagram 1 Decision Tree Feature Transform and Data Cleansing Spark ML libraries

- The Data Cleansing and Feature Transformation modules are created for the raw Title and Description Colums respectively and added as stages into the Spark ML Pipeline

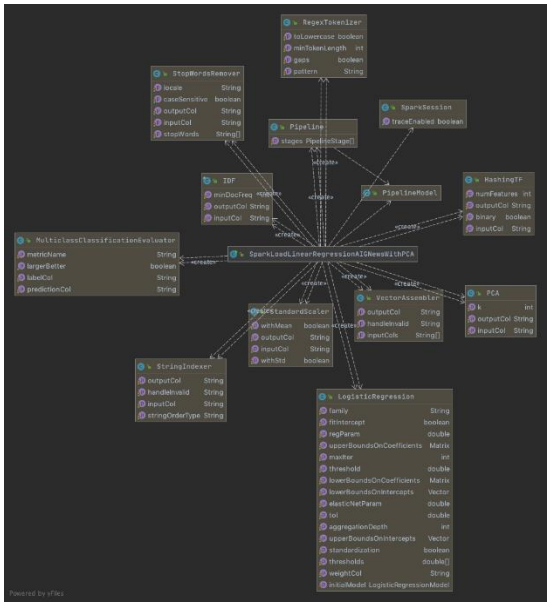


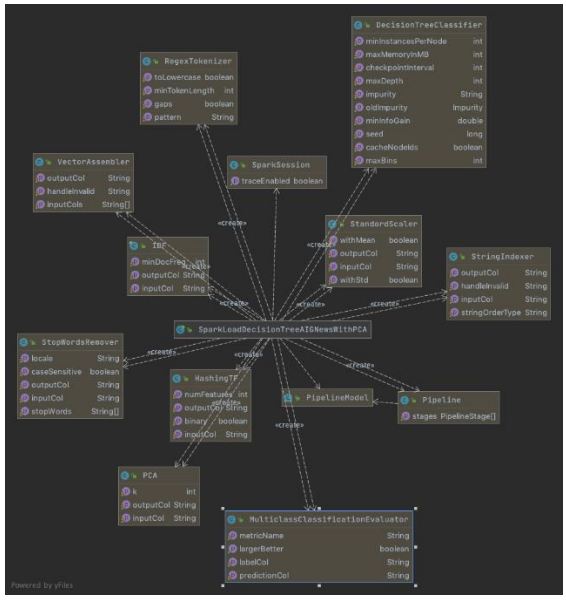
- Once the Title and Description Raw Data is cleaned and represented as output IDF feature vectors of words, they are combined into a single feature vector set by the Spark ML VectorAssembler.

```
VectorAssembler vectorAssembler = new VectorAssembler()
    .setInputCols(new String[] {"tw_idf_features", "sw_idf_features"})
    .setOutputCol("features");
```

  NOTE: If the PCA flag and the number of PCA features parameters are set as arguments to the program then PCA is used to reduce the feature dimensionality prior to combining the title and sentence word vectors
- A standard scalar is then set to rescale the word vector feature values, before being applied to the Logistic Regression and DecisionTreeClassifier Spark ML classifiers.
- At this point all the cleaning and tranform steps are set on the pipeline and the LogisticRegression or DecisionTreeClassifier classifiers are set on the Pipeline and the pipeline stages are ready to be fitted against the training data set and evaluated against both the training data and test data set.

Class Diagrams for Linear Regression and Decision Tree Classifier Programs





d) Installing and Running the Programs

The steps to install and run the two programs can be found in the Appendix. In addition the same readme is provided as part of Assignment 3 on the ECS submission site for AIML427.

e) Compare and Discuss Results

As discussed in Section a) above the AG News training set contains 120,00 training dataset instances (29 MB), and the test set contains 7600 testing samples (1.8 MB). The following results were obtained running both the Logistic Regression and Decision Tree programs.

Logistic Regression - Results varying normalization of feature values with ML StandarScalar

1000 feature count – for HashingTF for both Title and Description columns to feature vectors
NO_PCA flag, with a principle component count of 0, to ensure PCA is not used.

As the number of words in the AG News classification dataset for Title and Description is typically less the 800, setting the HashingTF to a greater number than expected word vector features means there is likely to be less hash clashes resulting in different tokens residing in the same bucket. Initial tests comparing HashingTF with CountVectorizer gave similar test errors, whereas the HashingTF had much better performance on the Hadoop Cluster.

In the first run the pipeline stages were executed without the StandardScalar.

Logistic Regression without Standard Scalar	
Command	*spark-submit* *--class q2.SparkLoadLinearRegressionAIGNewsWithPCA --master yarn --deploy-mode cluster* *skdd_lr_pca.jar resources/kaggle-news-train.csv resources/kaggle-news-test.csv 1 1000 NO_PCA 0 0*
Results	Training Error = 0.28485000000000005 Test Error = 0.2896052631578947 Elapsed Time = PT54.63S
Arguments	algorithm=LinearRegression, train=resources/kaggle-news-train.csv, test=resources/kaggle-news-test.csv, seed=1, features=1000, pca=NO_PCA, pcaFeature=0 useScalar=0
Logistic Regression with Standard Scalar	
Command	*spark-submit* *--class q2.SparkLoadLinearRegressionAIGNewsWithPCA --master yarn --deploy-mode cluster* *skdd_lr_pca.jar resources/kaggle-news-train.csv resources/kaggle-news-test.csv 1 1000 NO_PCA 0 1*
Results	Training Error = 0.16840833333333338 Test Error = 0.1919736842105263 Elapsed Time = PT5M33.932S
Arguments	train=resources/kaggle-news-train.csv, test=resources/kaggle-news-test.csv, seed=1, features=1000, pca=NO_PCA, pcaFeature=0 useScalar=1

The results from the Logistic Regression Model shows that using the StandardScalar to normalize the word vector feature coefficients shows a distinct improvement in the Test Error from 0.2896052631578947 (without normalization) to 0.1919736842105263 (with normalization). The training error variance indicates slight overfitting with data which is more pronounced using the

standard scalar. For Logistical Regression if the variance of some features are larger than others then these features can dominate the learning model.

The elapsed time for the Logistic Regression model shows a significant increase in processing time of PT5M33.932S (5 minutes 33.932 seconds) for logistic regression with scalar compared to PT54.63S (54 seconds) without scaling. Running the test several times shows that this difference in performance time is not directly due to the load on the machines at different times as the processing times of multiple runs is consistent.

Decision Tree – Results varying normalization of feature values with ML StandardScalar

Decision Tree without Standard Scalar	
Command	*spark-submit* *--class q2.SparkLoadDecisionTreeAIGNewsWithPCA --master yarn --deploy-mode cluster* *skdd_dt_pca.jar resources/kaggle-news-train.csv resources/kaggle-news-test.csv 1 1000 NO_PCA 0 0*
Results	Training Error = 0.5137 Test Error = 0.5255263157894736 Elapsed Time = PT57.291S
Arguments	algorithm=DecisionTree, train=resources/kaggle-news-train.csv, test=resources/kaggle-news-test.csv, seed=123, features=1000, pca=NO_PCA, pcaFeatures=0 treeDepth=12 useScalar=0
Decision Tree with Standard Scalar	
Command	*spark-submit* *--class q2. SparkLoadDecisionTreeAIGNewsWithPCA --master yarn --deploy-mode cluster* *skdd_dt_pca.jar resources/kaggle-news-train.csv resources/kaggle-news-test.csv 1 1000 NO_PCA 0 1*
Results	Training Error = 0.5137 Test Error = 0.5255263157894736 Elapsed Time = PT1M18.79S
Arguments	algorithm=DecisionTree, train=resources/kaggle-news-train.csv, test=resources/kaggle-news-test.csv, seed=123, features=1000, pca=NO_PCA, pcaFeatures=0 treeDepth=12 useScalar=1

The results from the Decision Tree Model shows that using the StandardScalar to normalize the word vector feature coefficients does not demonstrate any improvement in the Test Error of 0.5255263157894736 (for both scaled and non-scaled/normalized features). For Logistical Regression as discussed above scaling can have a dramatic effect on the performance accuracy, however decision trees do not suffer from scaling in the same way as they split based on an entropy/gini/gain values, therefor the scale of the numbers do not matter as much when compared to Logistic Regression.

The elapsed time for the Decision Tree model shows an increase in processing time of PT1M18.79S (1 minute 18.79 seconds) with scalar compared to PT57.291S (57.291 seconds) without scaling. Running the test several times shows that this difference in performance time is not directly due to the load on the machines at different times as the processing times of multiple runs is consistent, (see full logs of multiple runs in submission logs folder).

f) Compare and Discuss Results using PCA for dimensionality reduction

Logistic Regression - Results varying normalization of feature values with ML StandarScalar

Logistic Regression without Standard Scalar with PCA	
Command	*spark-submit* *--class q2.SparkLoadLinearRegressionAIGNewsWithPCA --master yarn --deploy-mode cluster* *skdd_lr_pca.jar resources/kaggle-news-train.csv resources/kaggle-news-test.csv 1 1000 PCA 12 0*
Results	Training Error = 0.28485000000000005 Test Error = 0.2896052631578947 Elapsed Time = PT54.63S
Arguments	train=resources/kaggle-news-train.csv, test=resources/kaggle-news-test.csv, seed=1, features=1000, pca=PCA, pcaFeature=12 useScalar=0
Logistic Regression with Standard Scalar	
Command	*spark-submit* *--class q2.SparkLoadLinearRegressionAIGNewsWithPCA --master yarn --deploy-mode cluster* *skdd_lr_pca.jar resources/kaggle-news-train.csv resources/kaggle-news-test.csv 1 1000 NO_PCA 0 1*
Results	Training Error = 0.28485000000000005 Test Error = 0.2896052631578947 Elapsed Time = PT53.949S
Arguments	train=resources/kaggle-news-train.csv, test=resources/kaggle-news-test.csv, seed=1, features=1000, pca=PCA, pcaFeature=12 useScalar=1

The results from the Logistic Regression Model shows that using the StandardScalar to normalize the word vector feature coefficients now difference to the test error when PCA is used. Both the Title and Sentence feature vectors are dimensionally reduced by PCA resulting in a simpler overall model. This seems to affect both the training and test performance. The Logistic Regression model performs best without PCA and with Scaling applied to normalize feature where large scale values effect the model accuracy.

The elapsed time for the Logistic Regression model shows a significant decrease in processing time when compared to the Logistic Regression model without PCA and with scaling applied. This is likely due to the large number of feature dimensionality in the non PCA model and is particular impacted by the affect of applying a scalar to the non PCA scaled/normalized model.

Decision Tree – Results varying normalization of feature values with ML StandardScalar with PCA

The Decision Tree classification program was run over a number of initial test runs to determine the best number of principal components for the large feature dimensionality of the AIG News dataset. The test runs determining the value of the number of principle components for PCA is outside the scope of this report.

Decision Tree with Standard Scalar with PCA	
Command	*spark-submit* *--class q2.SparkLoadDecisionTreeAIGNewsWithPCA --master yarn --deploy-mode cluster* *skdd_dt_pca.jar resources/kaggle-news-train.csv resources/kaggle-news-test.csv 1 1000 PCA 12 1*
Results	Training Error = 0.2509166666666667 Test Error = 0.3102631578947368 Elapsed Time = PT1M4.08S
Arguments	algorithm=DecisionTree, train=resources/kaggle-news-train.csv, test=resources/kaggle-news-test.csv, seed=123, features=1000, pca=PCA, pcaFeatures=12 treeDepth=12 useScalar=1
Decision Tree without Standard Scalar with PCA	
Command	*spark-submit* *--class q2. SparkLoadDecisionTreeAIGNewsWithPCA --master yarn --deploy-mode cluster* *skdd_dt_pca.jar resources/kaggle-news-train.csv resources/kaggle-news-test.csv 1 1000 PCA 12 0*
Results	Training Error = 0.2509166666666667 Test Error = 0.3102631578947368 Elapsed Time = PT56.347S
Arguments	algorithm=DecisionTree, train=resources/kaggle-news-train.csv, test=resources/kaggle-news-test.csv, seed=123, features=1000, pca=PCA, pcaFeatures=12 treeDepth=12 useScalar=0

The results from the Decision Tree Model shows the performance of the model provides a lower test error result of 0.3102631578947368 when using PCA for dimensionality reduction then the Decision Tree Model without PCA which has a test error result of 0.5255263157894736. \

The processing times for the Decision Tree model does not show any significant variance between the PCA and non PCA model runs, though this could be tested further.

In conclusion the Logistic Regression with Scaling and without PCA performed the best overall test performance. The accuracy overall for a complex multiclass large dataset was relatively promising Training Error = 0.16840833333333338 and Test Error = 0.1919736842105263 , given the time constraints of the assignment and would be worth deeper investigation.

References

[1] Aman Anand: AG's News Topic Classification Set. Retrieved 7 June 2021: https://www.kaggle.com/amananandrai/ag-news-classification-dataset

[2] Srini Penchikala: Big Data Processing with Apache Spark - Part 5: Spark ML Data Pipelines. Retrieved 7 June 2021: https://www.infoq.com/articles/apache-sparkml-data-pipelines/