Ben Noriega

CS777

2.22.23

Term Project

Project Repo: https://github.com/bennordi/cs-777-term-project-bennordi

Dataset

The dataset is a CSV file found on Kaggle under [Spam Text Message Classification](https://www.kaggle.com/datasets/team-ai/spam-text-message-classification). It contains 5572 rows of two columned data: Category and Message. Category can simply be described as the ‘class’ or ‘label’ of this dataset as it has a simple binary value: ham or spam. Spam represents the text in the ‘message’ column is from a text deemed to be spam. The opposite ‘ham’ represents the text was deemed to be, not spam.

Research Question

This is a simple classification dataset, which we will leverage PySpark and various libraries to analyze text data to build various machine learning models that will predict `ham or spam` based on the given text data. We will then review each model and its results. There are 5572 rows of data to analyze, and luckily the data was preformatted so that `ham/spam` are the only values in the category column, and the message column will need to be formatted into features columns for us to build models off of. The main need for preprocessing was removing non-letters from the text, lemmatizing the text, and then vectorizing the data (with MLlib’s pipeline using Tokenizer(), StopWordsRemover(), CountVectorizer(), IDF(). I was going to use ChiSqSelector() but we learned in Assignment 5 that with larger datasets this can be very computationally expensive, and knowing that the goal of this assignment is to be scalable, I opted to not perform this selection.

This data will be split into training and test sets (70/30 split) and will be seeded for consistency of the split. The classification models will be trained on the same exact training data and will be tested with the same test dataset. Then during the results, I can do some anecdotal tests for messages I create to test out our models.

Expectations

Based on the amount of balance of the classes in the dataset I imagine most text will be classified as ‘ham’. I wonder how this will take in data with long text, misspelled words, and other odd input.

Potentially more spam messages will have more ‘promotional’ elements to it, as the goal of most spam is to get users to click on certain things, potentially give money, etc.

Using MLlib, I’ll be able to take in the raw text, convert the text into tokens, and use other NLP elements in the MLlib and potentially other Python/Pyspark libraries to build multiple classification models.

Machine Learning Model

Used 6 different machine learning models from PySpark's MLlib library, and fit the model with our training dataset, predicted on our test dataset. Output of my local machine run can be found on output.md.

- Logistic Regression

- Linear SVC

- Naive Bayes classifier

- Decision Tree classifier

- Gradient-Boosted Tree classifier (GBT)

- Random Forest Classifier

Results

Timing of each step in the process:

* Loading data, preprocessing, vectorization: 25.456 secs
* Logistic Regression: 7.443 secs
* Linear SVC: 3.826 secs
* Naïve Bayes classifier: 2.440 secs
* Decision Tree classifier: 5.666 secs
* Gradient-Boosted Tree classifier: 14.998 secs
* Random Forest classifier: 3.856 secs
* Total time: 63.687 secs

**Logistic Regression:**

|  |  |  |
| --- | --- | --- |
|  | predicted(ham) | predicted(spam) |
| actual(ham) | 1431 – TN | 16 – FP |
| actual(spam) | 20 - FN | 201 – TP |

Accuracy: 0.978

Precision: 0.926

Recall: 0.910

F1: 0.918

**Linear SVC:**

|  |  |  |
| --- | --- | --- |
|  | predicted(ham) | predicted(spam) |
| actual(ham) | 1428 – TN | 19 – FP |
| actual(spam) | 15 - FN | 206 – TP |

Accuracy: 0.980

Precision: 0.916

Recall: 0.932

F1: 0.924

**Naïve Bayes:**

|  |  |  |
| --- | --- | --- |
|  | predicted(ham) | predicted(spam) |
| actual(ham) | 1372 – TN | 75 – FP |
| actual(spam) | 12 - FN | 209 – TP |

Accuracy: 0.948

Precision: 0.736

Recall: 0.946

F1: 0.828

**Decision Tree:**

|  |  |  |
| --- | --- | --- |
|  | predicted(ham) | predicted(spam) |
| actual(ham) | 1432 – TN | 16 – FP |
| actual(spam) | 73 - FN | 148 – TP |

Accuracy: 0.947

Precision: 0.908

Recall: 0.670

F1: 0.771

**Gradient-Boosted Tree:**

|  |  |  |
| --- | --- | --- |
|  | predicted(ham) | predicted(spam) |
| actual(ham) | 1431 – TN | 16 – FP |
| actual(spam) | 50 - FN | 171 – TP |

Accuracy: 0.960

Precision: 0.914

Recall: 0.774

F1: 0.838

**Random Forest:**

|  |  |  |
| --- | --- | --- |
|  | predicted(ham) | predicted(spam) |
| actual(ham) | 1447 – TN | 0 – FP |
| actual(spam) | 215 - FN | 6 – TP |

Accuracy: 0.871

Precision: 1.000

Recall: 0.027

F1: 0.053

**Local run output can be found on <output.md>**

Conclusion:

In terms of actual accuracy, Linear SVC and Logistic Regression were the most accurate at predicting the correct classification. In retrospect, based on the class distribution of ham/spam having the ‘spam’ class as positive makes the precision measure of this study a bit overrated. For example, at first look without seeing the confusion matrix, precision for the Random Forest model looks awesome at 100%, but to notice there were only 6 True Positives and 0 False Positives, the measure isn’t very valuable anymore.

Knowing that there were 1447 negative (ham) items and 221 positive (spam) items in our test dataset, this study could have used more of a closer distribution between classes. Had the split in training/test datasets been more even in terms of class the results would have been much more interesting but it really comes down to the goal of the analysis and prediction of the models. Do we want to be better at predicting spam or correctly predicting ham in terms of text messages? Obviously, the majority of text messages are considered ‘ham’, or legitimate, so the distribution of this ‘sample’ data is likely a good representation of what real world data looks like, and I’d hazard to guess that the 13% spam occurrence is even larger than the actual amount of spam coming to people’s text messages.

The results were very accurate for each model, each being above 94% accurate except for the random forest classifier (87%), which was my most disappointing model, having a precision measure of 1 (for only 6 True positives and 0 False positives) and a Recall measure of 0.027. The overall strongest model in this study was logistic regression, having an accuracy of 97.8%, Precision measure of 0.926, Recall of 0.910, and an F1-score of 0.918.

An effort to expand this study could be performing different feature extraction, or going a different route utilizing PySpark’s NLP libraries to break down the raw data to then build classifier machine learning models. Certain transformers could have been used such as Chi-Square Selection but as mentioned above the computational expense would make this use case difficult if the dataset was much larger.