CMPUT 466 Final Project

Arun Woosaree

December 16, 2021

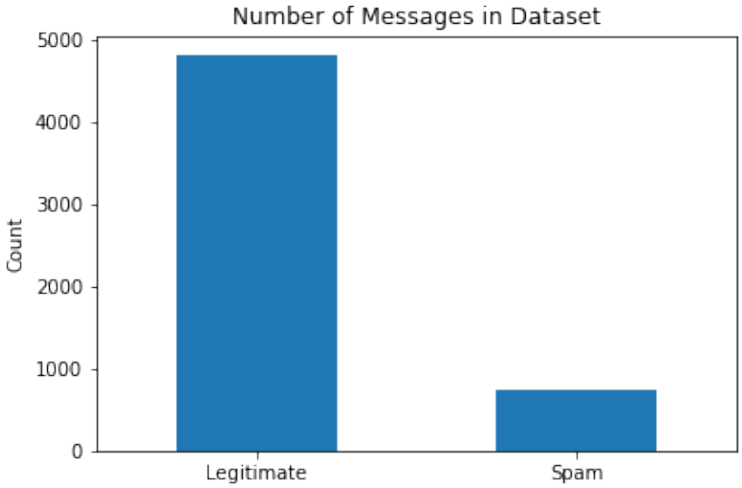
Introduction

This project focuses on the task of classifying text as "spam" or "not spam" using different machine learning algorithms. Spam detectors are important tools used everyday by people, like when checking emails. Receiving spam is not fun, but with the help of machine learning we can learn a classifier that can do the work of filtering out junk messages for us. In this final project for CMPUT 466, we attempt training/tuning the following types of models: Linear Regression, Logistic Regression, Naive Bayes, and a Multilayer Perceptron neural network.

Problem Formulation

The inputs to the models are text messages and the output is "1" if the message is spam, or "0" if the message is not spam. The dataset used for this project is originally from `https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection` and `https://www.dt.fee.unicamp.br/~tiago/smsspamcollection/`. It was conveniently converted to a comma-separated value format obtained from `https://www.kaggle.com/uciml/sms-spam-collection-dataset`. The messages from the dataset are from a collection of free and for research sources from the internet with a total of about 5572 messages. Parts of the data come from Grumbletext, NUS SMS Corpus, Caroline Tag's PhD thesis, and SMS Spam Corpus v.0.1. More details about the dataset can be found from the above links.

There are more legitimate messages in the dataset than spam messages. The bar chart below illustrates the relative difference in the number of spam messages versus the number of legitimate messages:



Approaches and baselines

Features were extracted from input text using the TF-IDF measure. Term Frequency Inverse Document Frequency is a statistical measure that reflects how important a word is to a document in a collection of documents. It increases proportionally to the number of times a word appears in a document, but is also offset by the number documents that also contain that word. This helps to offset the effect of common words like "the" appearing frequently.

For all models, the defaults in scikit-learn were used as a baseline. A grid search was used to select the best combination of parameters. This was combined with 5-fold cross validation to validate the performance of the model.

Linear Regression

The linear regression classifier likely would not be useful, but the results might be interesting. There are no hyperparameters to tune in this approach.

Logistic Regression

Baseline

For logistic regression, the following hyperparameters were used as a baseline:

- penalty: l2 (can also use none, l1, or elasticnet)
- tolerance for stopping criteria: 1e-4

Other Approaches:

20 different cominations of hyperparameters were chosen. These are the hyperparameters and the values tried out.

```
  params = {
    'solver': ('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'),
    'penalty': ('l1', 'l2', 'elasticnet', 'none'),
}
```

Multinomial Naive Bayes

For naive Bayes, no hyperparameters were tuned.

Perceptron

Baseline

For the perceptron, the following hyperparameters were used as a baseline:

- penalty: none (can also use l1, l2, elasticnet)
- max iterations: 1000
- tolerance for stopping criteria: 1e-3

Other Approaches

60 different cominations of hyperparameters were chosen. These are the hyperparameters and the values tried out.

```
params = {
    'penalty': ('l1', 'l2', 'elasticnet', 'none'),
    'max_iter': (500, 1000, 2000)
}
```

Multilayer Perceptron

Baseline

For multilayer perceptron, the following hyperparameters were used as a baseline:

- activation: relu (can choose identity, logistic, tanh)
- hidden layer sizes: (100,)
- solver: adam (can also use lbfgs, sgd)
- learning rate: constant (can also use invscaling, adaptive)
- learning rate: 0.001
- max iterations: 200

Other Approaches

108 different cominations of hyperparameters were chosen. These are the hyperparameters and the values tried out.

```
params = {
    'activation': ('relu', 'identity', 'logistic', 'tanh'),
    'hidden_layer_sizes': ((100), (100, 50, 25), (100, 80, 60, 40, 20, 10)),
    'learning_rate': ('invscaling', 'adaptive', 'constant'),
    'max_iter': (400, 800, 1600)
}
```

Evaluation metric

Because there are more legitimate messages than spam messages in the dataset, a trivial classifier would be to label all messages as legitimate. This yields an accuracy of 0.8659368269921034. Thus, the metric to beat is about 87% accuracy. If a learned model does not meet this threshold of accuracy, it is useless.

To measure the effectiveness of the classifier selected, 5-fold cross-validation was used for each classifier. An average was taken of the five accuracy scores, and compared to the 0.8659368269921034 baseline. If the accuracy was higher than this threshold, it is considered useful for the task of detecting spam.

Results

Linear Regression

As predicted, the accuracy for linear regression was low. The trivial classifier marking everything as spam was better than linear regression.

Logistic Regression

Out of the 20 possible combinations of hyperparameters selected, the best one was with no penalty and the newton-cg solver. The best mean score obtained was 0.9824118636835706. This seems like a pretty good result and beats the trivial classifier that marks everything as spam. It only took about 33.7 seconds to do training with 5-fold validation.

Multinomial Naive Bayes

Using Multinomial Naive Bayes, a mean accuracy of 0.9587220133482541 was obtained. This seems like a pretty good result and beats the trivial classifier that marks everything as spam. It only took about 4.4 seconds to train and do 5-fold cross-validation. Even though the accuracy was worse than logistic regression, it was much faster to train.

Perceptron

Out of the 12 possible combinations of hyperparameters selected, the best one was actually the baseline parameters: The best mean score obtained was 0.9807967088261103. This seems like a pretty good result and beats the trivial classifier that marks everything as spam. It only took about 2.6 seconds to do training with 5-fold validation. Even though this classifier is the fastest one to train, it has a really high accuracy.

Multilayer Perceptron

Out of the 108 possible combinations of hyperparameters selected, the best one was with the relu activation function, hidden layer sizes of (100, 50, 25), 800 iterations and an adaptive learning rate. The adaptive learning rate keeps the learning rate constant as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss, the current learning rate is divided by 5. The best mean score obtained was 0.9856415293331509. This seems like a pretty good result and beats the trivial classifier that marks everything as spam. It took about 1 minute to train with 5-fold cross-validation. This method took the longest time to train. This made the process of tuning hyperparameters with a grid search take a really long time. Although this method had the highest mean accuracy, the downside is that it took a long time to obtain this result.

Conclusion

The model with the highest accuracy was unsurprisingly the multilayer perceptron with the relu activation function, hidden layer sizes of (100, 50, 25), 800 iterations and an adaptive learning rate. The mean accuracy with 5-fold cross validation was 0.9856415293331509. As predicted, linear regression was not useful for the task of identifying spam as it is not a suitable approach for this type of problem. A surprising result was that the perceptron which was the fastest model to train, also had a good mean accuracy of 0.9807967088261103. If the goal is to have quick training, but also decent accuracy, the perceptron has a nice balance for the goal of identifying spam in texts. If the goal is to have the best accuracy at the cost of long training time, the multilayer perceptron classifier is the best. Using machine learning techniques is a good way to automate the identification of spam in texts. It is a useful technology we already use every day when checking our emails. This report shows how easy it is to learn a classifier with pretty good accuracy.