COSC2670: Practical Data Science

Assignment 2: YouTube Spam Classification

Thursday, 18 May 2017

Submitted by:

Name:	Max Yendall
Student Number:	s3436993
Email Address:	s3436993@student.rmit.edu.au

Table of Contents

Table of Tables

Table of Figures

1. Abstract / Executive Summary

The aim of this project is to utilize various techniques in data modelling, exploration, classification and evaluation in regards to classifying authored YouTube comments as spam or non-spam. The main research question being: what features determine whether an authored comment classes as spam or non-spam?

The corpus of authored comments is collected from various popular videos on YouTube from musicians: Eminem, Katy Perry, LMFAO, Psy and Shakira. These particular artists have incredibly high view counts on YouTube and in turn, have a large amount of spam due to the potential manipulation of YouTube users. The model and classifiers use break-down components of textual data in order to classify each comment accordingly with varying accuracies depending on the classifiers. The project concludes that Bayesian classification has a high success rate as opposed to K-Nearest-Neighbours, when classifying spam content and that a similar model is recommended for classifying spam to some satisfiable success rate.

2. Introduction

In order to classify the textual data contained in the YouTube Spam Collection, multiple techniques were used to split content variables into proportions, ratios, features and frequencies. Spam consists of many features which separate its content from typical, accepted forms of textual data. This includes alphabetic, numerical, sequence, special and spatial qualities.

In this project, modelling is focused on the spam content itself, rather than the authors and times the content was created, as spam can be classified solely on its semantics rather than its environment. Retrieval and exploratory processes involved parsing each variable into textual data frames and filtering the class variable of 1 and 0 to “Spam” and “Not Spam”. Upon retrieval, data was summarized and visualized to outline the type of data, relationships and statistical overviews for each frame. Once the data was prepared, each content row for spam and non-spam comments was split into TF-IDF frequencies in both 1-gram and 2-gram combinations, which was then split into training and test data which was used to train both a Multinomial Naïve Bayes and K-Nearest-Neighbours classification model.

3. Methodology

1. *Data Set Origin:*
   The data sourced for this project can be found at the following URL: http://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection.

   This data is a public set of comments collected for spam research. It has five datasets composed by 1,956 real messages extracted from five videos that were among the 10 most viewed on the collection period (2013 – 2015). The collection of data was achieved through the use of the YouTube Data API (v3) and consists of CSV files.

2. *Data Set Description:*

The five data sets included in the data collection have the following characteristics:

Data Set Characteristics:	Text	**Number of Instances:**	1956	**Area:**	Computer
Attribute Characteristics:	N/A	**Number of Attributes:**	5	**Date Donated**	2017-03-26
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	4286

Table 1: Data set characteristics

The data sets are balanced with an equal number of spam and non-spam comments, with each data set consisting of five textual variables, one being the class variable itself.

The data sets have the following variable characteristics:

Comment ID	Author	Date	Content	Tag
Text	Text	Text	Text	Text / Numeric

Table 2: Variable overview (columns) of Data Sets

The main variables used for classification are the *Content* and *Tag* variables, with the *Tag* being the class being predicted.

3. *Class for Prediction:*
The *Tag* variable is used for prediction, as it denotes whether an author comment is tagged as either “1” or “0” which resolves to “Spam” and “Not Spam”

4. *Data Cleansing*
Data cleansing was not necessary when parsing the initial data sets, however when extracting data for transformation, stop-word removal and punctuation filtering was used for certain measurements. Multiple dates were labelled as NaT but were dealt with appropriately when visualising the data.

5. *Data Transformation*
In preparation for classification and model training, the *Content* variable had multiple transformations applied which created new variables for training.

The first variable class utilised count vectorising, which extracted the vocabulary from the given collection and its data sets. This resulted in a vector of singular words with frequency tokens for each spam comment in the data set, giving insight into the frequency of terms in spam and non-spam

The second variable class utilised TF-IDF proportions, which extracted each singular word from the vectorised vocabulary which was then filtered through the following TF-IDF function:

$$tf(t,d) = \frac{t_f}{t_{d,f}}$$

$$idf(t,D) = \log\left(\frac{|D|}{(1 + |\{d \in D : t \in d\}|)}\right)$$

$$tfidf(t,D) = tf(t,d) * idf(t,D)$$

This creates a vector of proportionate values for each occurring term in the collection vocabulary. This also includes URLs and special character combinations.

The third variable is a combination of the first two variable techniques, but with added n-gram properties. An n-gram is a contiguous sequence of *n* items in a given sequence. In the case of this data, the contiguous sequence is textual vocabulary contained within comments, both spam and non-spam. This variable ranges from uni-grams, bi-grams, tri-grams and up to six-grams. This is then formulated as a proportion for each n-gram via TF-IDF as such:

$$tfidf(n_{gram}, D) = tf(n_{gram}, d) * idf(n_{gram}, D)$$

This now extends the variable characteristics as such:

Comment ID	Author	Date	Content	Vocab Count (Content)	TF-IDF Proportion (Content)	n-gram TF-IDF Proportion (Content)	Tag
Text	Text	Text	Text	Vector	Vector	Vector	Text / Numeric

Table 3: Additional Variable Overview

6. *Data Merging*
   Once the data sets have been filtered, cleaned and transformed, all sets are merged into a singular collection for training and classifying. This is achieved by simply merging the Pandas data frame objects into a singular data frame, joining on the column variables (including the newly created proportion variables).

7. *Classification:*
   Three classifiers were chosen for classification, both from different models of prediction: Probabilistic, Hyperplane-based and Distance-based.

   K-Nearest Neighbours:
   Using the K-Nearest Neighbour classifier; distances between comment features were computed using the Minkowski distance as a metric:

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

   The Minkowski metric is a generalised version of Manhattan and Euclidean distance metrics. Each feature of a comment is plotted as a point in normal vector space and is compared to features of comments in the training corpus. The classifier labels each comment based on the highest aggregation of spam or non-spam features within *k* neighbours of the test comment features. This was repeated for singular features, TF-IDF proportions and n-gram TF-IDF proportions.

   Multinomial Naïve Bayes:
   Using the Multinomial Naïve Bayes classifier, probabilities can be calculated based on the features of each comment as to whether they are attributed to spam or non-spam. The training of this classifier involves determining which features are determined to be spam or non-spam based on word frequencies and TF-IDF proportions calculated in transformation. Once determined, probability models are generated for each term, and compared against test data when features are inputted to the classifier.

   This model is distinctly multinomial as a multinomial distribution is better fit to textual data due to raw counts and frequencies as metrics. Each probability model follows the Bayes Theorem[1] and is used as an aggregate to determine whether a newly given comment is spam or not-spam based on the extracted features of the comment. The Bayes Theorem is used as a basis for a Bayesian classification, for a given message feature x what class c produced it, where Spam is S and non-spam is L:

$$p(c|\,x) = \frac{P(x|c)P(c)}{P(x)} = \frac{P(x|c)P(c)}{P(x|S)P(S) + P(x|L)P(L)}$$ [2]

[1] http://ats.cs.ut.ee/u/kt/hw/spam/spam.pdf
[2] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Optional Extension: *Support Vector Machine:*
Using the Support Vector Machine classifier, spam and non-spam comments can be classified by finding an optimal hyperplane in vector space that successfully separates both classes of comments. In order to train the Support Vector Machine classifier, multiple specific features were extracted from comments and represented as numeric proportions. The following features were extracted from each comment:

- Number of Characters
- Proportion of Alphabetic Characters to all Characters
- Proportion of Numeric Characters to all Characters
- Proportion of Whitespace Characters to all Characters
- Proportion of Special Characters to all Characters
- Number of Words
- Average Word Length
- Proportion of Unique Words to all Words
- Proportion of Top 10 Spam Words to all Words
- Number of URLs

These features are then represented as a vector, similar to previous frequency vectors. Each feature is fitted in vector space and a Linear Kernel is used to find the maximal margin separating hyperplane. The hyperplane is fitted to the training data by finding the closest spam and non-spam vector. This linear separation boundary is then used to determine whether new comments fall into the spam and non-spam vector spaces when inputted to the classifier.

The idea of using a Support Vector Machine is to find a linear separation boundary ($w^T x + b = 0$) between the training set and the optimal hyperplane to define the regions where each spam feature vector is placed. Once given a feature vector from the test set, it is classified on one side of the $w^T x + b$ boundary.

4. Results

Exploration (Summaries):

Due to all variables being textual in this data collection, summary statistics were generated for each column:

Comment ID:

Count	Unique Count	Top Value	Top Frequency
1956	*1953*	*LneaDw26bFuH6iFsSrjlJLJIX3qD4R8-emuZ-aGUj0o*	*2*

Table 4: Comment ID Summary Statistics

Author:

Count	Unique Count	Top Value	Top Frequency
1956	*1792*	*M.E.S*	*8*

Table 5: Author Summary Statistics

Date:

Count	Unique Count	Top Value	First	Last	Top Frequency
1711	*1709*	*2014-11-07 19:33:46*	*2013-07-12 22:33:27.916000*	*2015-06-05 20:01:23*	*2*

Table 6: Date Summary Statistics

Content:

Count	Unique Count	Top Value	Top Frequency
1956	*1760*	*Check out this video on YouTube:*	*97*

Exploration (Visualisation of Relationships):

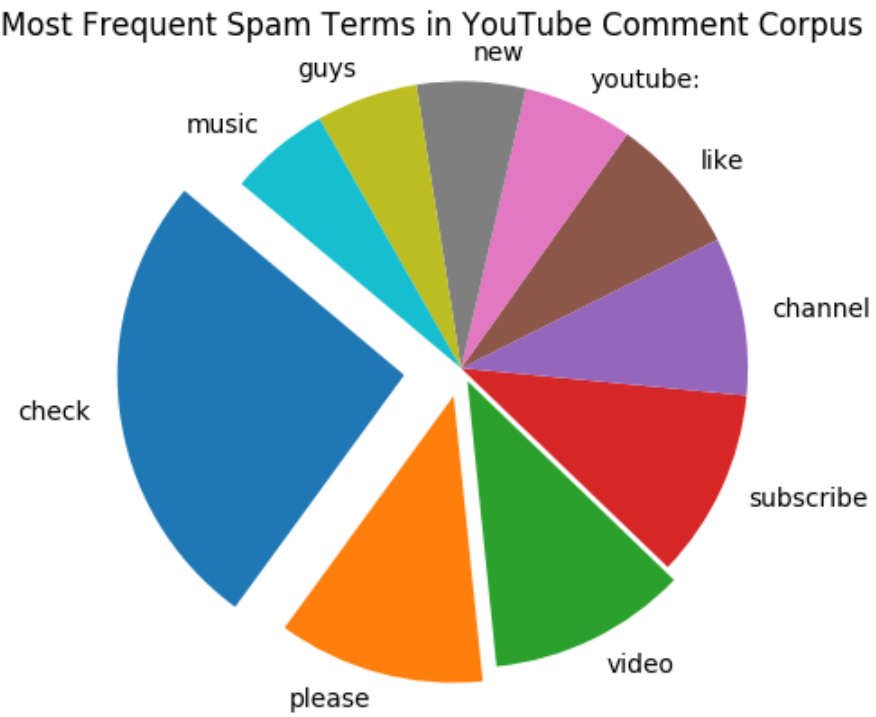

Figure 1: Pie Chart of Most Frequent Spam terms (Content by Class)

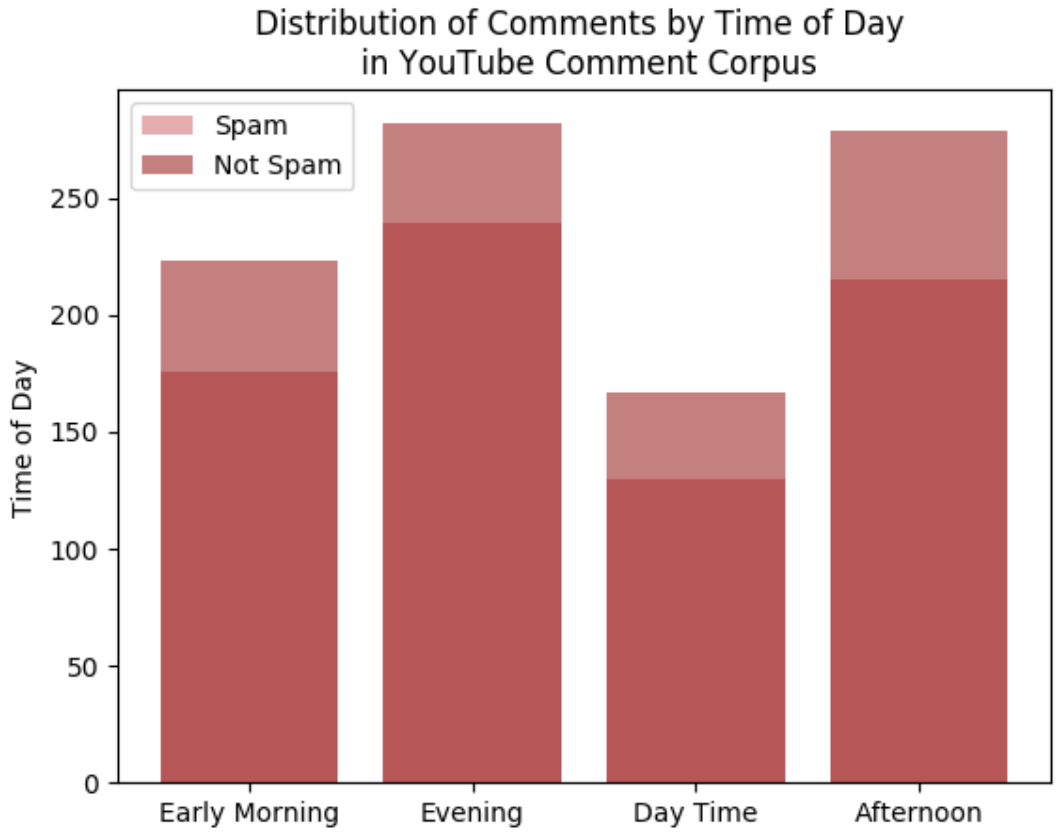

Figure 2: Stacked Bar Chart of Comments at Different Times of Day (Content by Date)

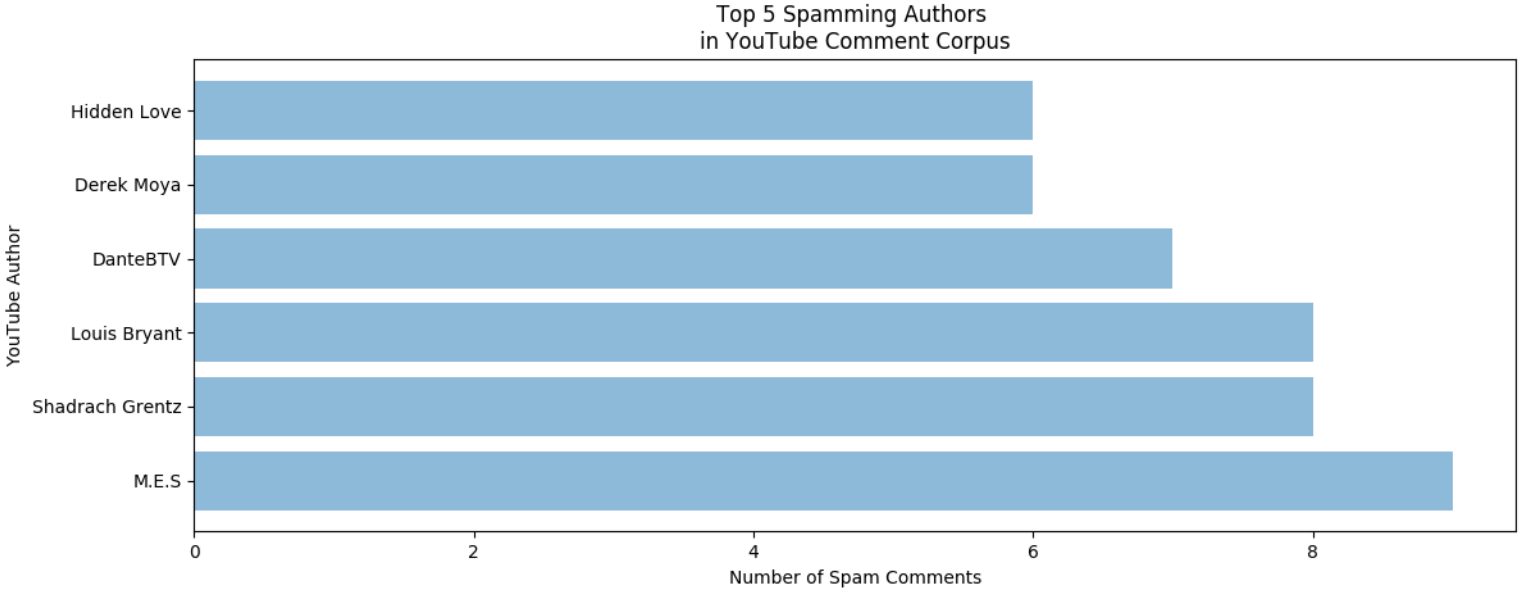

Figure 3: Horizontal Bar Chart of Top Spamming Authors (Author by Content Frequencies)

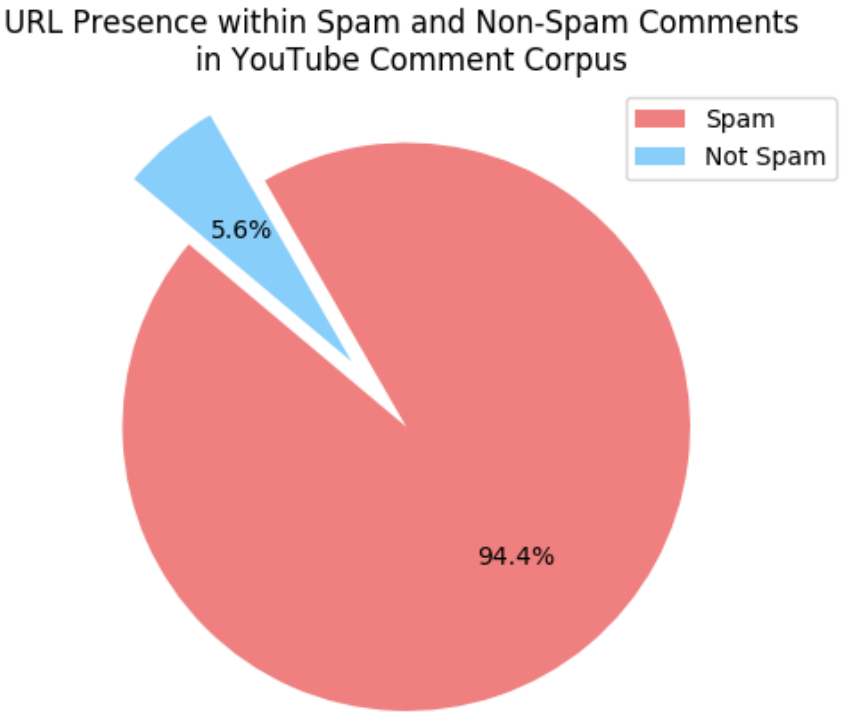

Figure 4: Pie Chart of URL Presence in Spam and Non-Spam Comments (Regex Content Frequency)

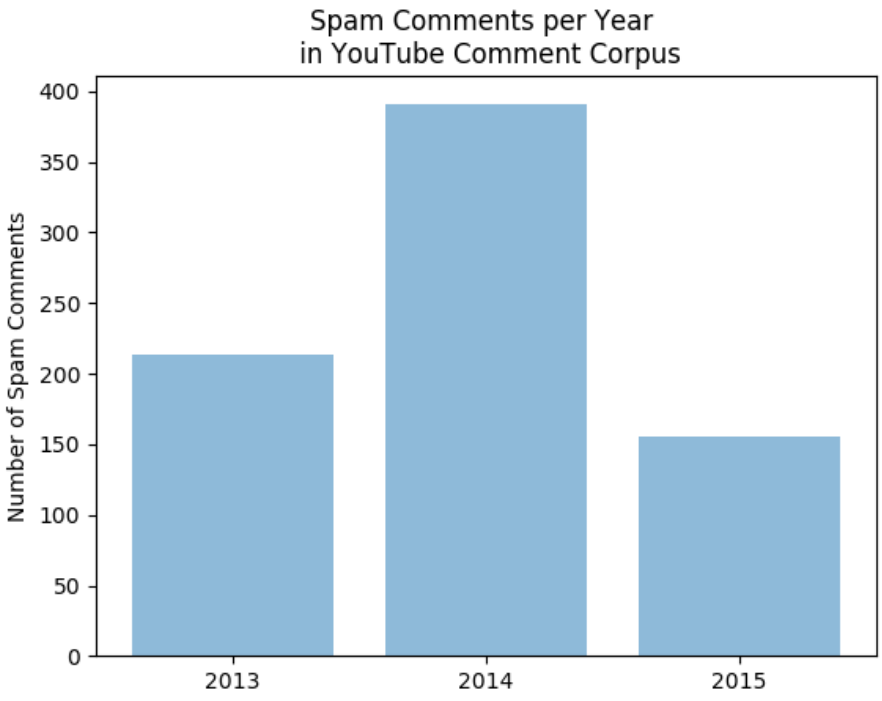

Figure 5: Bar Chart of Spam Comments per Year (Date by Content Frequency)

Classifier Results:

K-Nearest Neighbours (Direct Values):

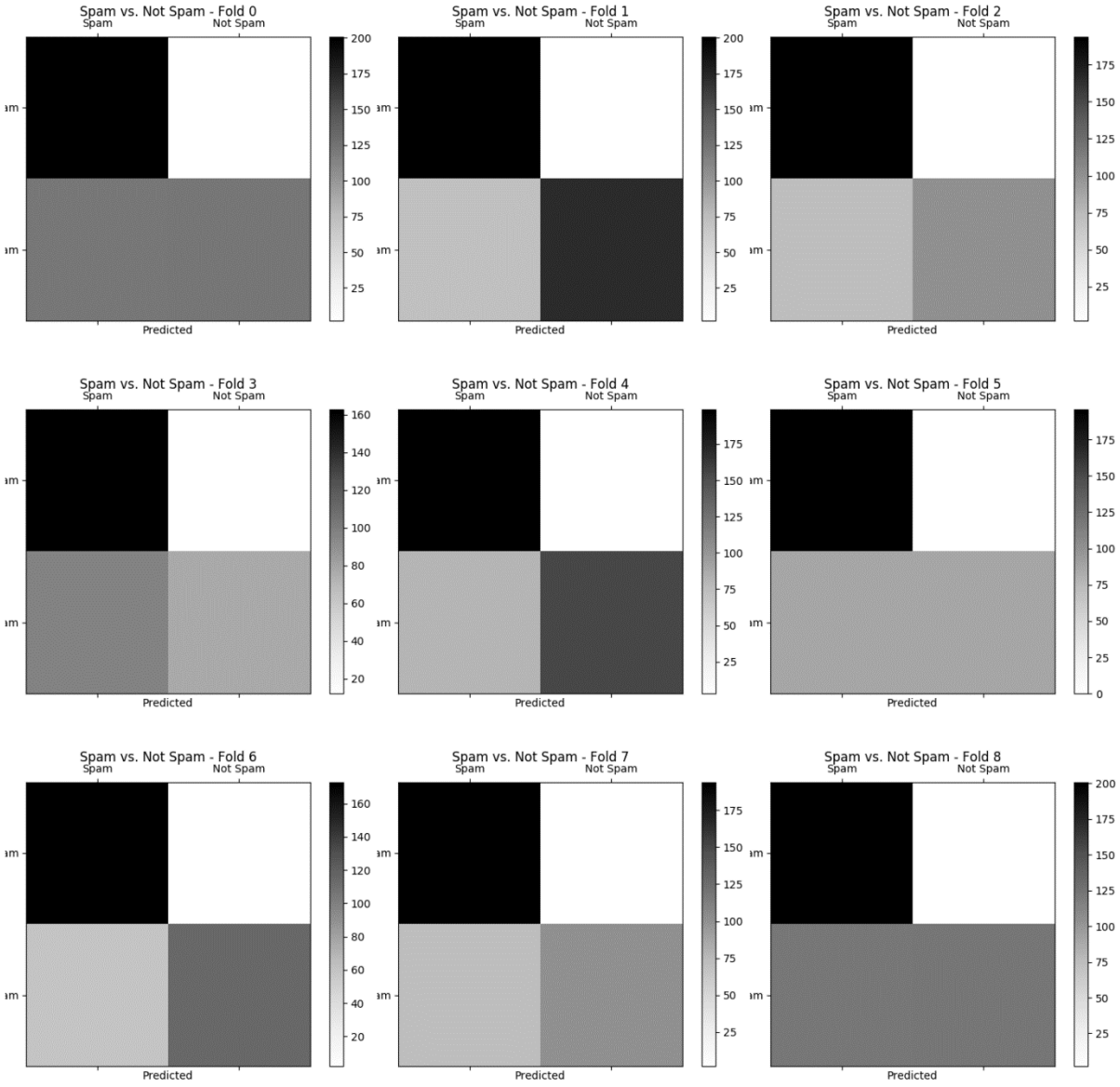

Figure 6: Confusion Matrix Cluster for KNN (Direct)

Average K-Fold Classification Error Rate	Average F1 Score	Average Precision	Average Recall
22.45%	71.69%	97.28%	57.07%

Table 7: Classification Results for KNN (Direct)

K-Nearest Neighbours (1-gram and 2-gram):

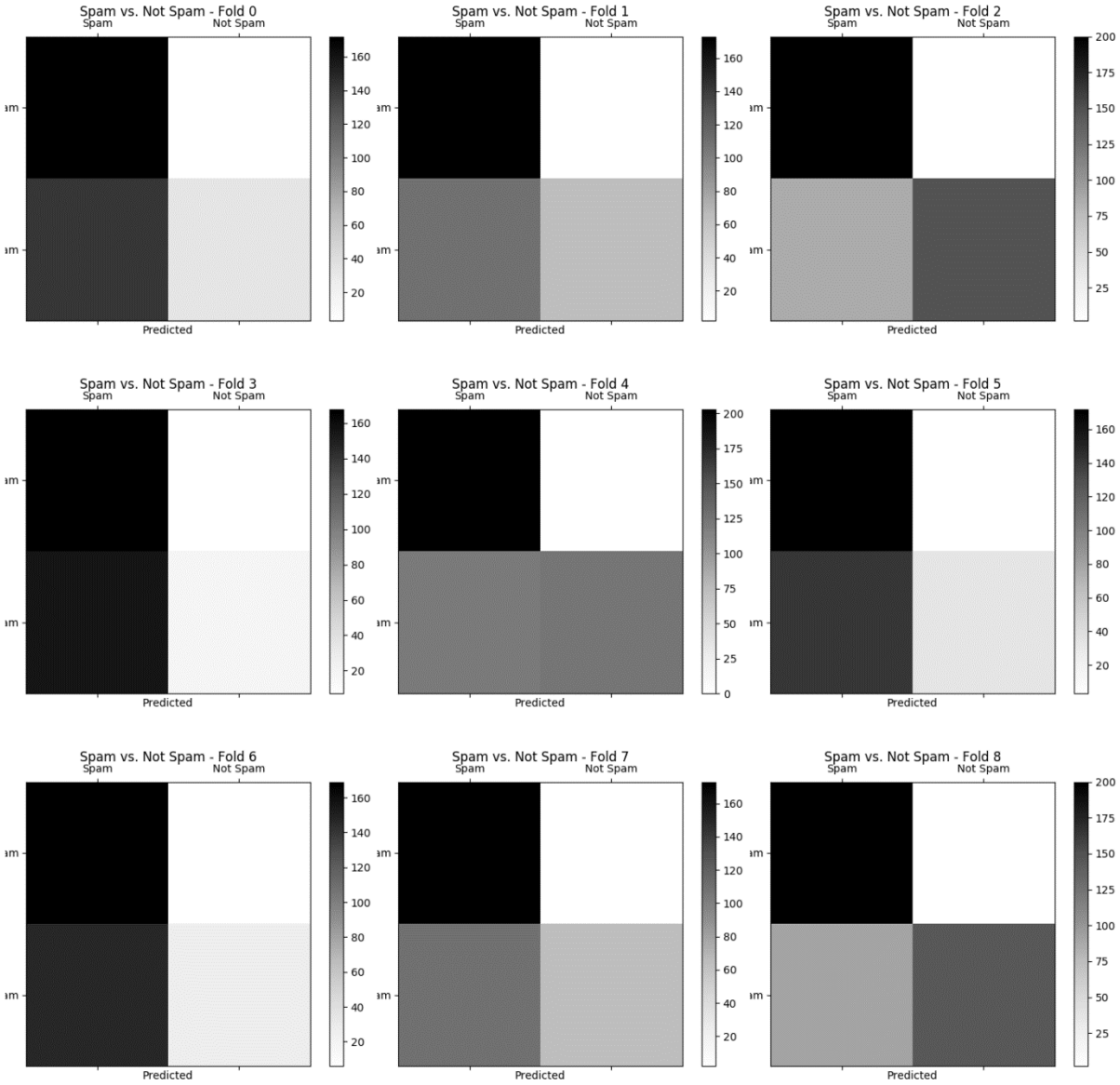

Figure 7: Confusion Matrix Cluster for KNN (1-gram and 2-gram)

Average K-Fold Classification Error Rate	Average F1 Score	Average Precision	Average Recall
37.61%	41.93%	93.94%	28.41%

Table 8: Classification Results for KNN (1-gram and 2-gram)

K-Nearest Neighbours (1-Gram, 2-Gram & TF-IDF):

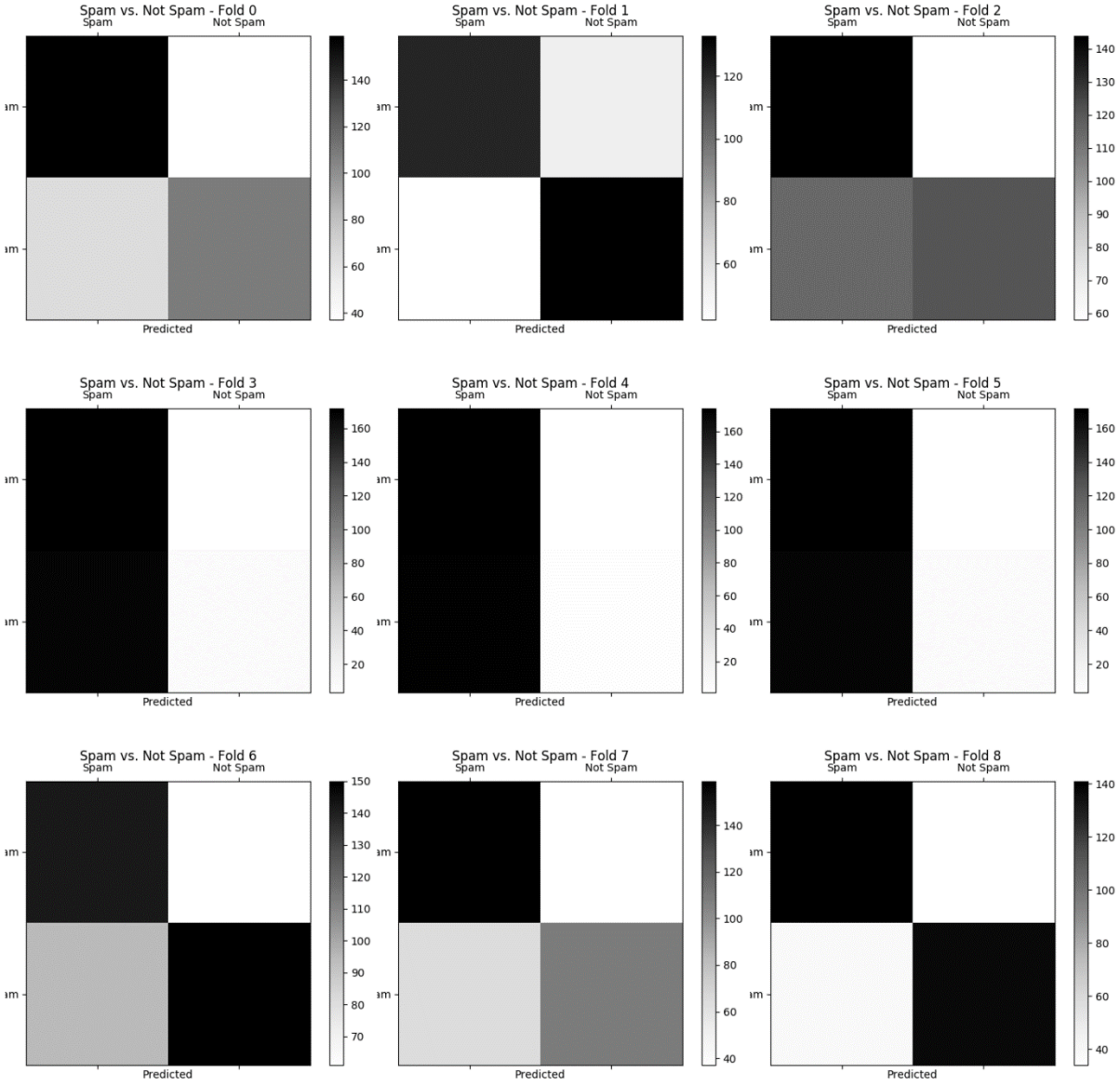

Figure 8: Confusion Matrix Cluster for KNN (1-gram, 2-gram and TF-IDF)

Average K-Fold Classification Error Rate	Average F1 Score	Average Precision	Average Recall
35.54%	49.50%	73.19%	45.55%

Table 9: Classification Results for KNN (1-gram, 2-gram and TF-IDF)

Multinomial Naïve Bayes (Direct Values):

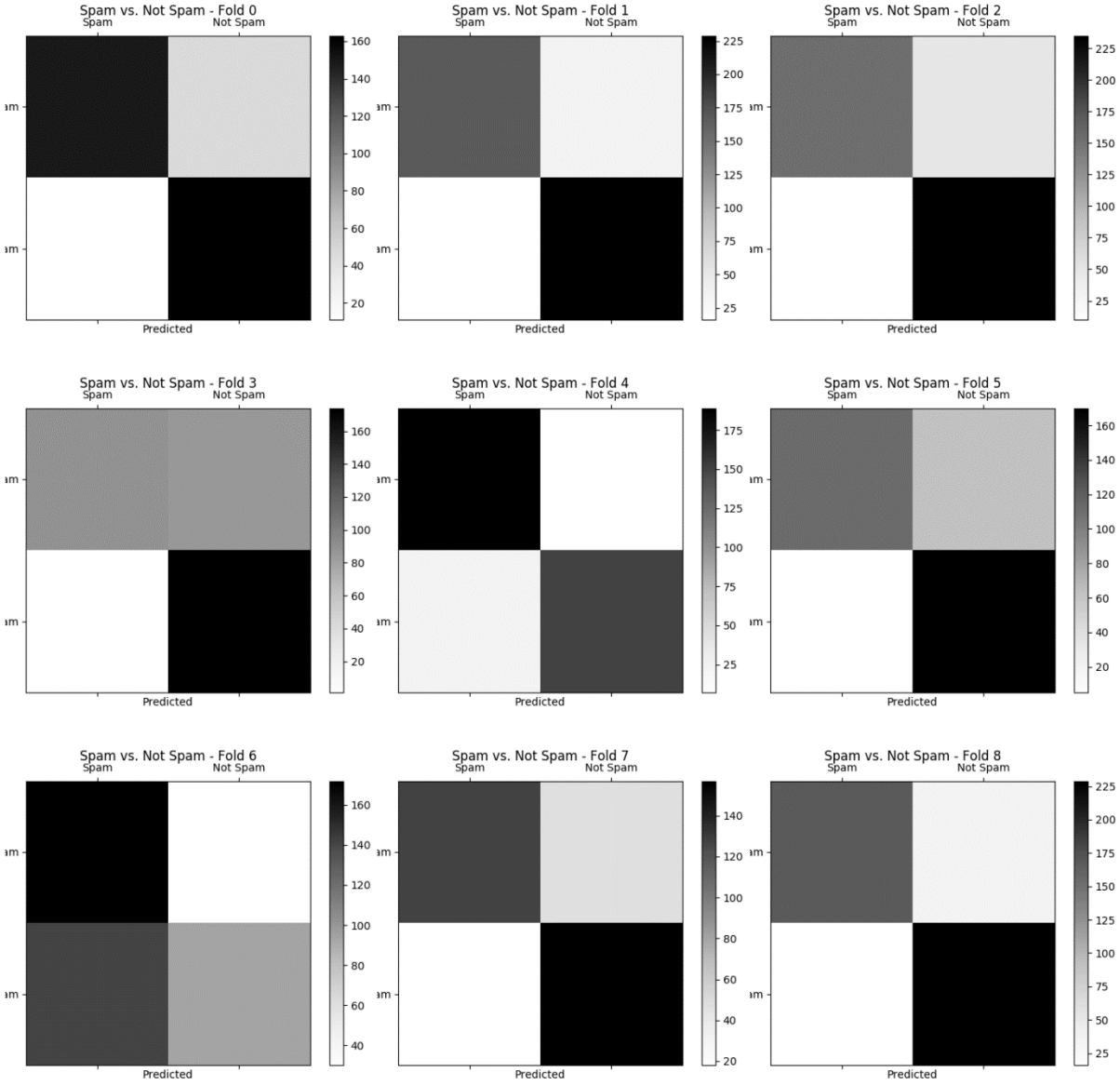

Figure 9: Confusion Matrix Cluster for Naive Bayes (Direct)

Average K-Fold Classification Error Rate	Average F1 Score	Average Precision	Average Recall
17.50%	83.19%	80.48%	88.53%

Table 10: Classification Results for Naive Bayes (Direct Values)

Multinomial Naïve Bayes (1-gram and 2-gram):

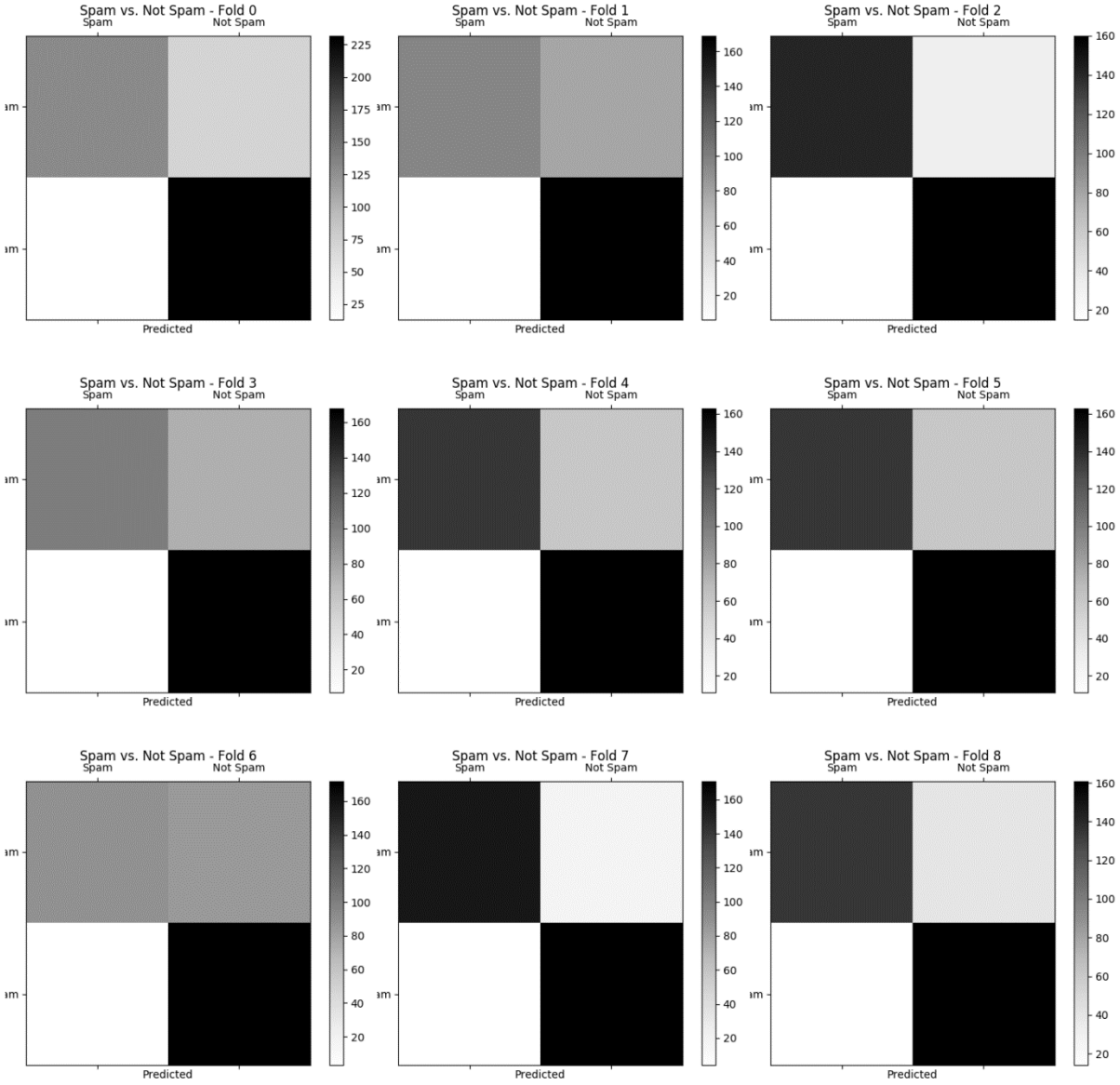

Figure 10: Confusion Matrix Cluster for Naive Bayes (1-gram and 2-gram)

Average K-Fold Classification Error Rate	Average F1 Score	Average Precision	Average Recall
18.56%	83.87%	75.49%	94.85%

Table 11: Classification Results for Naive Bayes (1-gram and 2-gram)

Multinomial Naïve Bayes (1-gram, 2-gram and TF-IDF):

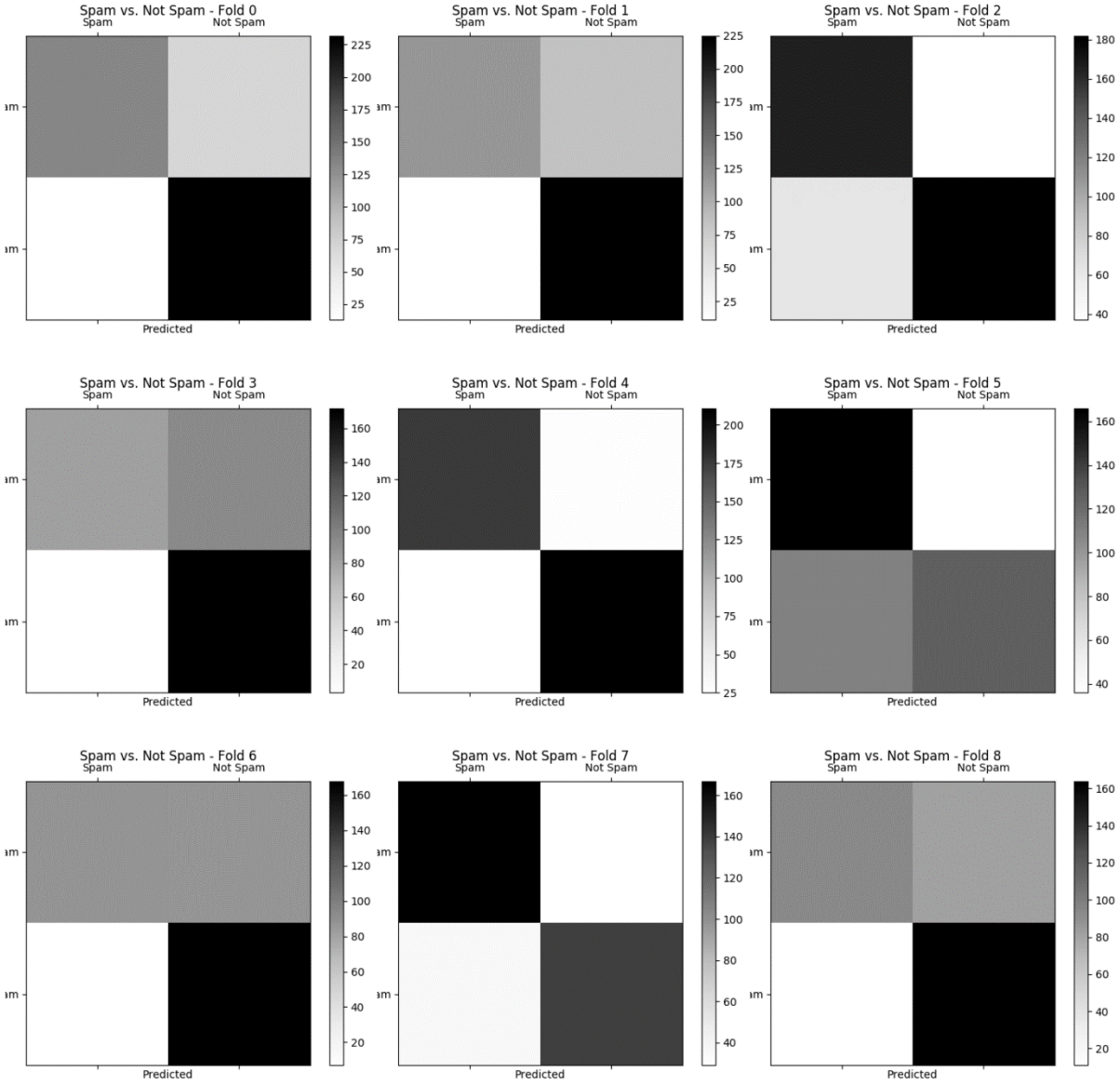

Figure 11: Confusion Matrix Cluster for Naive Bayes (1-gram, 2-gram and TF-IDF)

Average K-Fold Classification Error Rate	Average F1 Score	Average Precision	Average Recall
21.73%	80.31%	76.88%	86.44%

Table 12: Classification Results for Naive Bayes (1-gram,2-gram and TF-IDF)

Optional Extension: *Support Vector Machine:*

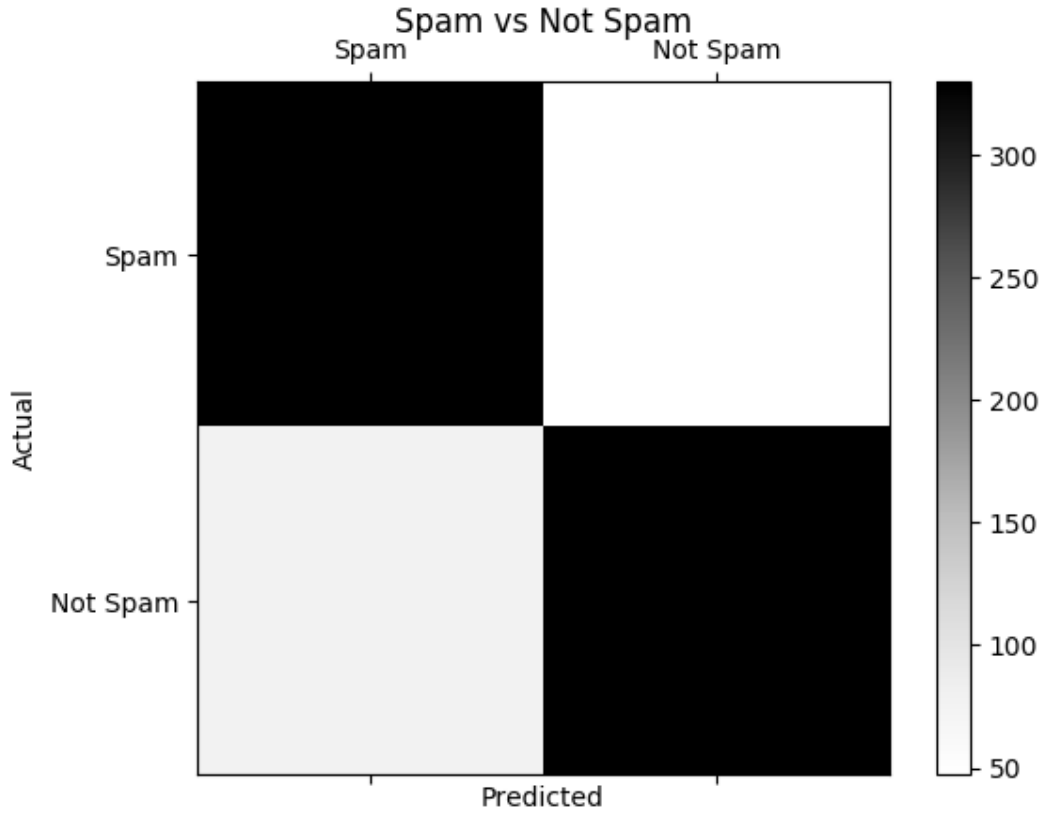

Figure 12: Confusion Matrix for Support Vector Machine

Classification Error Rate	F1 Score	Precision	Recall
15.45%	84.55%	86.72%	82.63%

Table 13: Classification Results for Support Vector Machine

5. Discussion

The classification of spam and non-spam comments involved the use of data transformation, feature extraction and data filtering. The difficulty in classifying textual data is the necessity for numeric representations of features, including the vectorisation of frequency and proportion. Alongside this, the creation of additional features for a Support Vector Machine proved to be a recursive process, testing and finding the correct mix of features to extract.

In regards to classification, overall the Support Vector Machine and Multinomial Naïve Bayes classifiers performed well. The K-Nearest-Neighbours classifier did not perform well under the multiple conditions it was presented with (direct values, n-grams, TF-IDF transforms).

K-Nearest-Neighbours returned very high False-Positive rates which would result in a lot of unnecessary filters being applied to spam comments. This indicates that fetching the distance between features isn't necessarily the most effective way of classifying content. Because of a high False-Positive (Low Recall) alarm it would seem features are losing contextual meaning.

Multinomial Naïve Bayes performed well due to its probabilistic nature of computation. By utilising probabilities for each feature, it can scale well in regards to the impact of a certain feature on whether it is spam or non-spam. By having a well-defined metric for each feature through the use of Bayes Theorem, aggregates can be predict quite well on new features to a reasonable degree. Given the probability calculations of spam messages in particular, it dramatically reduces the number of False-Positives (High Recall), as spam messages have very distinct features. Generally an above 80% rate for classification with low False-Positives (High Recall) indicates a suitable classifier for spam detection.

The Support Vector Machine performed very well also, having consistent accuracy above 80% when tested recursively. This is due to a well-chosen set of features, which is paramount for Support Vector Machines to work. The feature extraction can be improved in this implementation, although the set of features tends to work well. Given the selected features, the Support Vector Machine tended to create an effective linear hyperplane that happened to label a large amount of test data successfully, with little False-Positives.

6. Conclusion

Overall, the exploration and analysis of the YouTube Spam Collection was a challenge in regards to vectorising textual data and using it for classification, with many variations and features to be considered. Exploring various classifiers and classification paradigms resulted in clear positive and negative aspects of choosing one over another, showing the variability in the nature of data.

Possible future developments would include:

- More intrinsic feature extraction of textual data, including semantics of natural flow
- Various classifier tuning in regards to Bayesian models and Support Vector Machines
- Utilising author features to classify recurring spamming user accounts
- Acquiring a larger corpus for a real-world test (50,000+ comments)
- Training a Neural Network for comparison

Many research questions have opened up, including the use of Deep Neural Networks and abstract approaches to a machine-based understand of textual trends, and a more thorough understand of Bayesian models and Support Vector Machine implementations

7. References

[1] Tretyakov, KT, 2004. Machine Learning Techniques in Spam Filtering. 1st ed. Institute of Computer Science, University of Tartu: Institute of Computer Science, University of Tartu.

[2] Analytics Vidhya. 2017. Understanding Support Vector Machine algorithm from examples (along with code) . [ONLINE] Available at: https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/. [Accessed 15 May 2017]

[3] Kibriya, AK, EF, BP, GH, 2017. Multinomial Naive Bayes for Text Categorisation Revisited. 1st ed. University of Waikato: University of Waikato.