CPTS 315

# COURSE PROJECT REPORT

## SMS Spam Classification


## Introduction

Imagine this - You get a sudden unsolicited text message saying you've won a $1000 gift card to Walmart. You get super excited because you've been eyeing that new iPhone X but your bank account always held you back so you click the link that says 'REDEEM'. Instead of the quick and easy money you were hoping for, you now find yourself caught in a rabbit hole of sign up pages, subscriptions, and other hoops they ask you to jump through. Eventually they ask you to take just a quick survey , possibly hand over your shipping address, credit card information and as much information they can squeeze out of you

before you realize the new iPhone probably isn't worth this hassle. So you close your browser on your phone and get on with your day. It sounds like a mild annoyance that you would never fall for right? Not quite.

The Federal Trade Commission (FTC) is issuing  warnings about a dangerous new trend in the Spam and Phishing arena rather quaintly named 'Smishing'.[1] They are to your phone what spam emails are to your computer. Text messages are the next logical avenue for spammers to target as people grow more and more suspicious of email and as spam filters in email grow more and more sophisticated[2]. While there are no empirical studies to show how much of the phone using population falls for these scams, it is safe to assume that these messages that are designed to prey on your impulse to panic or sense of urgency find a significant number of people that will take the bait. We can get a gauge by looking at Verizon's Data Breach report from 2015 that showed that in a study of 150,000 phishing emails, 23% of them opened the messages and 11% of them opened the attachment.[3] On average it was found that it takes 82 seconds after a phishing scam is initiated for the first person to take the bait.[4] Given how much of our personal data is stored in our phones, the need for securing our information from outside hostile entities is stronger than ever.

With this context in mind, I set out to answer some questions regarding what we were doing to combat this threat along with gaining a better understanding of how data mining could be a key tool in tipping the scales against the scammers. The questions I started out with were rather broad and the open ended nature of the project allowed me integrate this goal with my learning goals for the class. Some of my initial questions were

- What is the state-of-the-art in terms of spam filters for mobile?
- What libraries does python offer for data science?
- Which classifier allows for the most optimal classification of text messages?
- What words occur most frequently in Spam texts?

[1] Longley, R. (n.d.). How to Spot and Avoid Text Message Smishing Scams. Retrieved from https://www.thoughtco.com/text-message-scams-dont-text-back-3974548
[2] Here's How to Avoid 'Smishing,' the Latest Internet Scam. (n.d.). Retrieved from http://fortune.com/2017/07/07/smishing-scam/
[3] Verizon 2015 Data Breach Investigations Report. (2017, July 27). Retrieved from http://www.verizon.com/about/news/2015-data-breach-report-info/
[4] Aguilar, M. (2015, April 14). The Number of People Who Fall for Phishing Emails Is Staggering. Retrieved from https://gizmodo.com/the-number-of-people-who-fall-for-phishing-emails-is-st-1697725476

- What are some common pitfalls to be aware of when it comes to spam classifications?

I set about attempting to answer these questions through a series of trial and errors in terms of coding and by asking Google. When I first initiated the project, I had limited the scope to just me building a classifier from scratch and applying it on the data set. I then realized I had already explored the development of a binary classifier when we built the perceptron. I decided that a more interesting study would involve the classifiers we had discussed in class but hadn't programmatically implemented. Another learning goal I wanted to get out of the class was how I would realistically use data mining in an internship setting. We were just discussing frameworks and reusability in another class of mine and it got me thinking that the best possible thing I could do with my time was to familiarize myself with the most relevant libraries for this task rather than implement my own questionable classifier.  I discovered the capabilities of the sklearn python library and decided that I could use this to compare multiple classifiers on the dataset. I found a wordcloud library, the pandas library and plotting tools that allow me to visualize data in an otherwise mostly scripted and non-visual python interface. Some of my main challenges came from having changed the scope and intention of my project to an exploratory one. That meant my definitions for its success were a lot more implicit since learning is often not measured by a concrete result. So I decided to build an API for the final classifier that performed the best and had a couple of my friends try it out.

I chose to compare the four classifiers we had discussed in class -

- SVM - linear
- Random Forest
- Decision Tree
- K - Nearest Neighbors

I then found the different Vectorizers in sklearn and paired each of the above with

- TFIDF Vectorizer
- Count Vectorizer

The SVM with the TFIDF vectorizer performed the best and so I put it in a script and built a simple API that applied the trained classifier to predict the label of any input message. The details are elaborated in the sections that follow.

# Data Mining Task

## Input Data

It consists of a series of text messages in a csv file that contains data records in the form [label text - - -] . The messages are labeled either ‘spam’ or ‘ham’. I split the data into 80:20 in order to leave some test data for accuracy scoring and validation.

## Task Details

The initial task was to build a working classifier with a high accuracy for the dataset. That objective then evolved into a comparison study of the sklearn library’s inbuilt classifiers and vectorizer combinations and building a simple application for real time classification. Many discuss content based filtering as necessary to counteract the spam threats.

The first stage of the final project plan was me reading through the literature about this particular problem and industry verified approaches to solving it. I referred to papers describing their use of data mining to filter spam text messages to understand the state of the industry today, recommended algorithms and challenges that are faced when trying to efficiently classify texts.

Based of what I knew about the problem space, I split the entire project’s code into these quantifiable parts:

- Figure out what words are most commonly in spam text by building word clouds
- Compare the classifiers and vectorizer combinations
- Chart their accuracy scores
- Choose the best performing one and output the predicted labels for the test data
- Build API for this classifier

The output was then analyzed and I tried to gain a better understanding of the results of each of these tasks. Some of the answers I found to address my questions are discussed in the Results section.

## Questions

Some of the broader questions are detailed in the introduction section. Some additional questions did arise as I went through the coding for this project. Why was a particular classifier the best and is this the best for this particular binary classification problem or for all word based problems of this nature? I also wondered - How do we know how cautious a particular classifier is being when it comes to declaring a message as spam? How do libraries optimize their training to be so quick ? What data structures are underneath the black box of the classifiers in sklearn ? Can email classifiers be used as well ? Most of these questions were what arose from trying to figure out how to code and accomplish the tasks mentioned above. While they weren't related directly to the result of my project, it offered me some insight into what thread I could follow to improve my understanding of practical implementations of the algorithms.

## Challenges

Initially, when I planned to implement my own classifier from scratch, I realized that I fell into my usual antipattern of reinventing the wheel so I switched goals. As it is with any project that relies heavily on a framework that you don't quite understand, I found myself struggling a lot in trying to understand and read through the documentations of the libraries I considered in order to help me accomplish my project goals. Many of python's packages are open source and have very few usage examples to base my implementation on. So finding the holy grail of a library that could both do what I wanted and had documentation explaining how to use it was a challenge. Sklearn is well documented but the vocabulary needed to properly manipulate the functions the way you intend them was something I had to learn. Another challenge was trying to do more complex things with the dataset when the result accuracy was already quite high. Any additional improvements could only be meaningful when evaluated against a larger test set.

## Technical Approach

The programming section consists of three different scripts I wrote.

The first one compares the different classifiers along with the two vectorizers. First I prepped the data by renaming the columns to text and label and dropping the unneeded columns. I discovered the nltk library that already had a built in way to remove stop words so I used that on the entire data frame. To address my earlier question about what words best constituted a spam or a ham message, I made two word cloud images of the respective messages. This exercise helped generate examples to test my online classifier with. It also showed me how much people hate the letter ‘e’ in ‘let’ and ‘get’!

I then chose my classifiers from the sklearn documentation and configured multiple of them to different parameter values. I then trained them on the training set and calculated the accuracy scores on the test set. I familiarized myself with the matplotlib library and plotted the scores to have a visual representation.

As for the classifiers, I chose

- Random Forest - n_trees = 50 , max_depth = 10
- Random Forest - n_trees = 100 , max_depth = 5
- Random Forest - n_trees = 10, max_depth = 15
- Decision Tree
- SVM - Linear Kernel
- KNN - k = 50
- KNN - k =10
- KNN - k=1

In order to tune parameter of the random forest and k-nn classifiers I used multiple parameter values to measure performance. I used these paired with the two vectorizers that we had talked about in class in order to compare the simplistic Bag-of Words vectorizer with the TF-IDF methodology-

- The simple CountVectorizer
- TFIDFvectorizer

The SVM classifier paired with the TFIDF vectorizer came out on top. I built this into a script and took a look at all that it go right or wrong. I realized this gave me no way to answer the question regarding cost sensitivity on how conservative it was regarding spam or non-spam. I found a parameter called prediction probability that showed how likely each label was according to the classifier. It can be used at a certain threshold probability by an actual filtering application to decide to accept the model's prediction of 'spam'. I used the same classifier and vectorizer in the third script that is a mini application. It uses the Flask microframework to open a simple GET request for a classification on a given text message. The reason for the inclusion of this portion was to explore ways of implementing a practical portal for the classifier to perform predictions in real time.


```
{
  "prediction": "spam",
  "probability_correct": [
    [
      0.019223509114073445,
      0.9807764908859266
    ]
  ],
  "text": "CALL 4352 for FREE Ringtones"
}
```


Fig 1. Classifier on Localhost:5000

## Evaluation Methodology

The input data for this project came from the SMS Spam Collection Dataset in Kaggle.The entire dataset contains a total of 5574 messages extracted from various research sources compiled on the internet. The output of the data mining task happened to be my

application which I tested on both the test data segment on based off of some actual spam messages my friends had received over time. The accuracy of my application was quantitatively evaluated by the test data set's accuracy scores and also verified as a usable solution to some limited capacity. The number of false positives was also a metric given how the cost of misclassifying a ham message as spam can be. I also evaluated my learning goals based on what I could find answers to for the questions I asked, either through the programming or my research in the problem space. The details of some of the answers I got for my questions are addressed in the result section.

## Results and Discussion

The results of the word cloud building are shown below. These show what words occur most often in the spam labeled data set and the ham labeled one.


Fig 2. Spam Word cloud

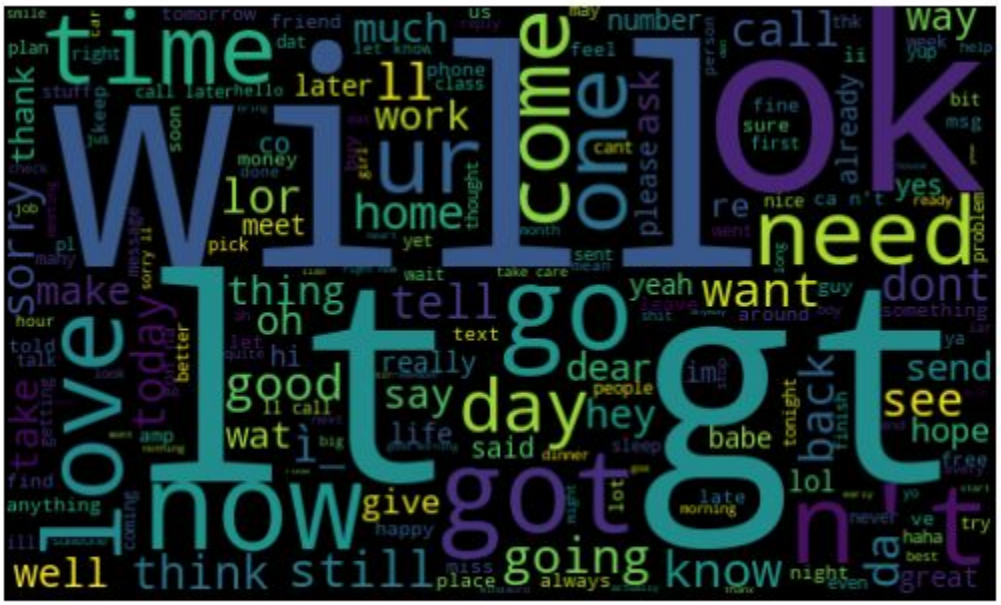



Fig 3. Ham word cloud

The outcome of the classifier comparison is shown in the graph below in Fig 4.

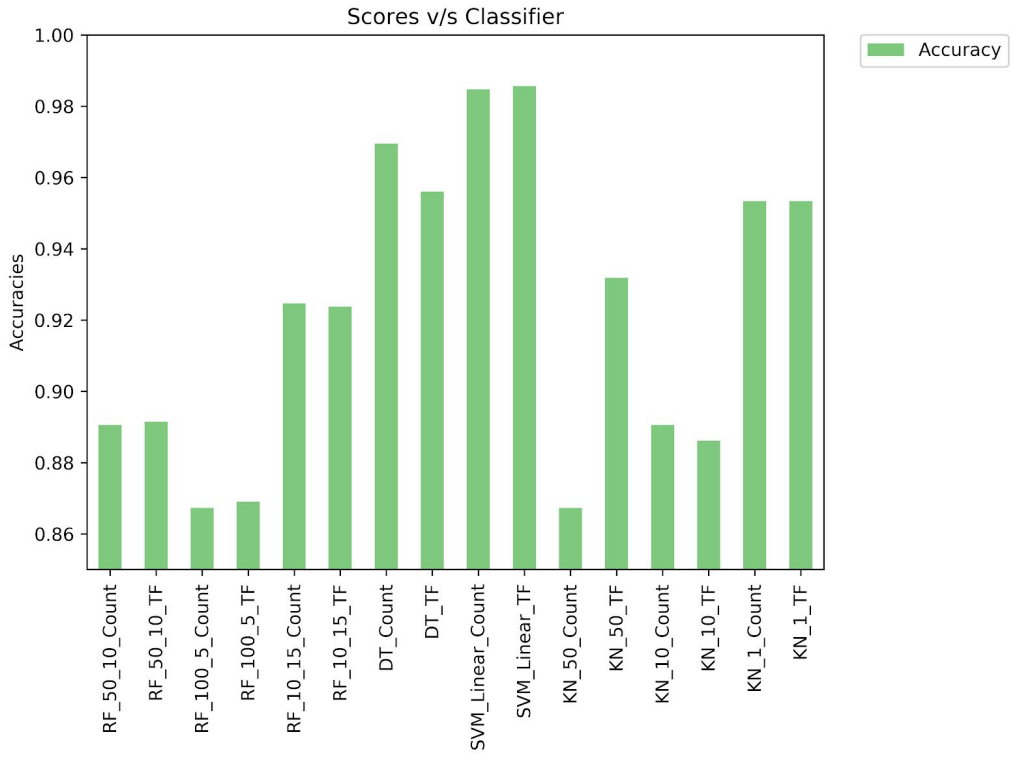
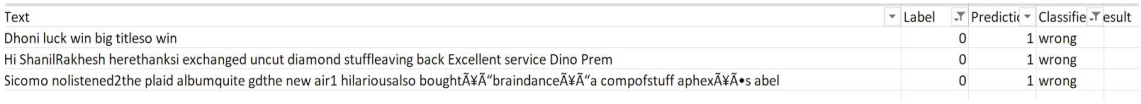


Fig 4. Scores V Classifier Bar graph

Observations:

- The TFIDF vectorizer fared better for each of the classifiers when compared to the count vectorizer. This was an expected outcome as TF - IDF looks at a normalized count instead.[5]
- The SVM with the linear kernel came out with an accuracy score of 98.565 %
- For the SVM , out of a test set of 1114 examples
    - The number of correctly classified test examples were 1098
    - The number of wrongly classified test examples were 16
    - The number of false positives were 3



| Text | Label | Predicti | Classifie | esult |
|---|---|---|---|---|
| Dhoni luck win big titleso win | 0 | 1 | wrong | |
| Hi ShanilRakhesh herethanksi exchanged uncut diamond stuffleaving back Excellent service Dino Prem | 0 | 1 | wrong | |
| Sicomo nolistened2the plaid albumquite gdthe new air1 hilariousalso boughtÃ¥Ã"braindanceÃ¥Ã"a compofstuff aphexÃ¥Ã•s abel | 0 | 1 | wrong | |

Fig. 5  False Positives from test set

Some of the questions I researched were :

1. Why is a linear kernel SVM most suited for text classification ?

Most of text classification problems are linearly separable, have a high dimensional input space and they often have very few irrelevant feature, thereby limiting the feature pruning normally required of other high dimensional problems.Therefore it can be empirically shown to be suited for text classification problems, particularly of a binary nature[6]

2. What is the state-of-the-art in terms of spam filters for mobile and what techniques do they use ?

Simple filtering methods like using traffic analysis to identify high volumes of messages from individual subscribers are employed. But the sophistication of the scammers are one

[5] Casari, A., & Zheng, A. (n.d.). Feature Engineering for Machine Learning. Retrieved from https://www.safaribooksonline.com/library/view/feature-engineering-for/9781491953235/ch04.html
[6] http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

the rise and required more content-based filtering techniques.[7] A multitude of data mining algorithms have been proposed over the years based on SVMs, Naive Bayes, the Winnow algorithm, index models. The study by Almeida, Gómez Hidalgo, and Yamakami(2011) had similar results to mine where they concluded that the SVM was the most optimal with alphanumeric tokenization.[8]

3. What are the challenges to keep in mind when considering text spam?

The size of text messages doesn't allow for much material to use for content based filtering. Client side filtering done on a mobile is also limited by the phones minimal resources as opposed to a filtering system hosted on a large web server. The strange lingo might lead to a higher number of false positives as building dictionaries is limited. A single word may be written in multiple lexically wrong spellings. The paper - Text Normalization and semantic indexing [...][9] has an interesting solution to the content filtering and word format limitations, involved the extraction of concepts and normalizing the text lingo.

## Lessons Learned

Many times, in industry and in academics, the parameters and specifications are handed to you and you meet them in the best way you know how. I approached this problem the same way and think I limited myself in terms of its scope as my first thought was to build a working classifier. I wish I had followed and implemented one of the reference papers I found later in the discovery process - Text normalization and semantic indexing to enhance Instant Messaging and SMS spam filtering. I was unable to do so due to time constraints and a lot of time wasted trying to reinvent the wheel when it came to building a classifier. It was an interesting idea to first normalize the text, generate a concept and identify the concept that made sense in context in order to validate the message. I wish I had taken

[7] SMS spam filtering: Methods and data. (2012, February 22). Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417412002977
[8] SMS spam filtering: Methods and data. (2012, February 22). Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417412002977
[9] Text normalization and semantic indexing to enhance ... (n.d.). Retrieved from http://www.bing.com/cr?IG=28FA5934F1BE4CFEAE0484E4A92CEE9A&CID=017AF544B9C565BA06F3FEA6B86A64C5&rd=1&h=p5_hxDXKG7F0XsJsqC-7vua3H0PZl1S-0Q2ifjaSNLk&v=1&r=http://daneshyari.com/article/preview/571784.pdf&p=DevEx.LB.1,5066.1

more note of how I could enhance the techniques like so. The lesson learned there was to plan better!

I got a little carried away experimenting with the functionality of the sklearn library that I understood but didn't take the time to modify and modulate the data much. It would have been nice to try out the classifier on a larger data set. For the application part of it, in hindsight, I wish I had implemented a way of having the user send back a response to the prediction and have the model use it as a training example. That would have been fun!

I learned how complicated it can be to formulate your own guidelines and measures of success for a project. For someone who has never done any research based project in computer science, the process of asking the right questions and implementing a series of steps to get to a valid answer that can be tested and evaluated based on metrics was challenging. The way I went about it felt frivolous and tangential at times. Next time onwards, I will make sure the programming, the questions and the research are all geared towards a coherent objective. Although I did enjoy the freedom the project offered in terms of exploration. From a pure technical standpoint, I am glad of all the technical aspects of the project that I learned as well as how to make use of in built classifiers.

Overall I think the project gave me an experience I can draw on at my summer internship. As I was working on it and most of the projects for this class, many of the discussions I had last year with a data science intern at Expedia were being put into context. I don't feel so dumb anymore! Now that almost every company is pivoting toward a paradigm of data driven decision making, understanding what data is telling us is an invaluable skill to have.

# Bibliography

Aguilar, M. (2015, April 14). The Number of People Who Fall for Phishing Emails Is Staggering. Retrieved from https://gizmodo.com/the-number-of-people-who-fall-for-phishing-emails-is-st-1697725476

Casari, A., & Zheng, A. (n.d.). Feature Engineering for Machine Learning. Retrieved from https://www.safaribooksonline.com/library/view/feature-engineering-for/9781491953235/ch04.html

Here's How to Avoid 'Smishing,' the Latest Internet Scam. (n.d.). Retrieved from http://fortune.com/2017/07/07/smishing-scam/

Longley, R. (n.d.). How to Spot and Avoid Text Message Smishing Scams. Retrieved from https://www.thoughtco.com/text-message-scams-dont-text-back-3974548

SMS spam filtering: Methods and data. (2012, February 22). Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417412002977

Text normalization and semantic indexing to enhance ... (n.d.). Retrieved from http://www.bing.com/cr?IG=28FA5934F1BE4CFEAE0484E4A92CEE9A&CID=017AF544B9C565BA06F3FEA6B86A64C5&rd=1&h=p5_hxDXKG7F0XsJsqC-7vua3H0PZl1S-0Q2ifjaSNLk&v=1&r=http://daneshyari.com/article/preview/571784.pdf&p=DevEx.LB.1,5066.1

Verizon 2015 Data Breach Investigations Report. (2017, July 27). Retrieved from http://www.verizon.com/about/news/2015-data-breach-report-info/