Predicting Spam from SMS Message Board

Group member: Winnie Shao, Yihui Zhang, Wenjia Dou

Summary of research problems

1. What information can we get to distinguish the difference between ham and spam messages if we only perform basic exploratory data analysis?

   In this part, we want to perform the Exploratory Data Analysis on our dataset. Because the dataset we used only has 2 columns, we want to find more information from those text messages. By doing that, we want to compare the length of two categories and also intend to explore the top 10-20 words used in spam and ham messages. We may plot a word cloud to visualize different word choices in spam and ham SMS.

   From our result, the ham and spam messages are different in total characters and also the word choice of different types of messages is very different.

2. How can we process the text message to make them machine-readable?

   In order to predict spam from the message board, we want our data machine-readable so that we need to transform our raw SMS messages into numerical. We plan to develop 5 features (word count / average word length / URLs count / special punctuation count / TF-IDF) that can define these messages. By converting the raw SMS messages into numerical language features, the messages will be machine-readable so that we can use the dataset to make predictions.

3. What model is most effective in predicting spam messages?

   For this problem, we will try some different models, such as Decision Tree Classifier, Naive Bayes Classifier, SVM Classifier, and XGBoost Classifier, to find the most optimized one. To do this, we divide the language feature generated data into training and testing sets and then try to find the minimum test mean squared error and the most accurate train method. Finally, we can plot out the internal decision tree of our model so that we can easily find out the most proper model and most important language features in defining spam messages.

   From our result, SVM Classifier is the most effective in predicting spam messages.

4. What features are most important in predicting spam messages?

   After we figure out which model is the most effective one, we can then choose which features are the most important. Since we have 5 features, including word count, average word length, URLs count, special punctuation count, TF-IDF, we choose each of them one by one as the selected feature, and then use the model we find to see which feature shows higher accuracy to predict whether the message is spam or ham. By doing these, we can finally figure out the most important features in finding spam messages.

   From our result, we identify TF-IDF as the most important factor in predicting spam messages.

Motivation and Backgrounds

Due to the rapid development of technology, companies or some criminals can immediately send pre-written text messages to thousands of personal mobile phones. According to ABC News, based on the data released in 2012 by the Pew Research Center, 69% of those who text said that they got unwanted spam messages. Additionally, 25 percent of them admitted to getting spam messages at least once a week. These spam emails usually include advertisements, unsafe links, fraudulent emails, etc. This not only makes it inconvenient for people to view important text messages but also puts people at risk of being scammed. Our team members are also annoyed by the frequent receipt of various types of spam messages, so our motivation is to help people reduce the trouble and the risk caused by spam messages. Also, we notice that although there are lots of spam message blocker apps available on phones, those apps cannot completely prevent spam messages and sometimes mistreat important messages as spam. Therefore, during the journey to find an efficient model for eliminating spam, we want to understand the decision factors and parameters in those predicting models. After this project, we can find the most efficient model to automatically identify spam messages and reduce the harassment caused by spam messages.

Dataset

URL: https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection
We obtained this dataset from the UCI Machine Learning Archive. In this dataset, we have 5574 text messages. The dataset only has 2 columns: the first column indicates if the text is spam by using string value ‘spam’ and ‘ham’, and the second column contains the text content.

Methodology

Our goal is to create several machine learning models to predict which message is spam or ham and figure out which model is the most effective one to use. Also, we want to find the most important features.

Step 1: Exploratory Data Analysis

From our original dataset, we need to grab some basic information about the messages. First, we drop three extra columns with no value in them from the dataset. Then, we add a column using a binary variable to represent the result of spam messages(1=spam, 0=ham). Finally, in order to do the exploratory analysis, we add a column recording the total characters in every text. We will compare the message length

distribution for ‘ham’ and spam messages to see if there is a difference between the mean, median, and standard deviation of different word lengths. Then we will use the Wordcloud package to create word cloud plots so that we could have a better understanding of the word-choice difference between ‘ham’ and spam messages.

Step 2: Natural Language Processing

After cleaning the dataset and grabbing basic information of messages, we will process our data to make them machine-readable by doing natural language processing. First, we will tokenize the messages in the SMS column and break the words. Next, we will create new columns to contain five numerical language features, including word count, average word length, URLs count, special punctuation count, and TF-IDF, to make the messages machine-readable. The word count column saves the total number of words for each message without all punctuations, url tokens which start with “http” (case non sensitive) and digits for each message; The average word length contains the value of total words length divided by the total word count; The URLs count is the number of the URL links for each message starting with “http”; The special punctuation count represents the number of the special punctuations except for digits, characters, periods and commas; The TF-IDF means the term frequency-inverse document frequency between each message and each term in the vocabulary.

Step 3: Model Fitting

Next step we need to figure out which model is the most effective one in predicting spam messages for our dataset and so we can build our machine learning algorithm with it. First, we choose some models, including Decision Tree Classifier, Naive Bayes Classifier, SVM Classifier, and Random Forest Classifier. Specifically, for Decision Tree and Random Forest, we test some different max depths and find the best one for them. And then we build a machine learning algorithm with each of them and see which one has the highest accuracy. In detail, we divide the language feature generated data into a 20% training set and an 80% testing set and then find the model with the most accurate score. We will use the best fitting model in the following step to find the most important feature and finally build our algorithm.

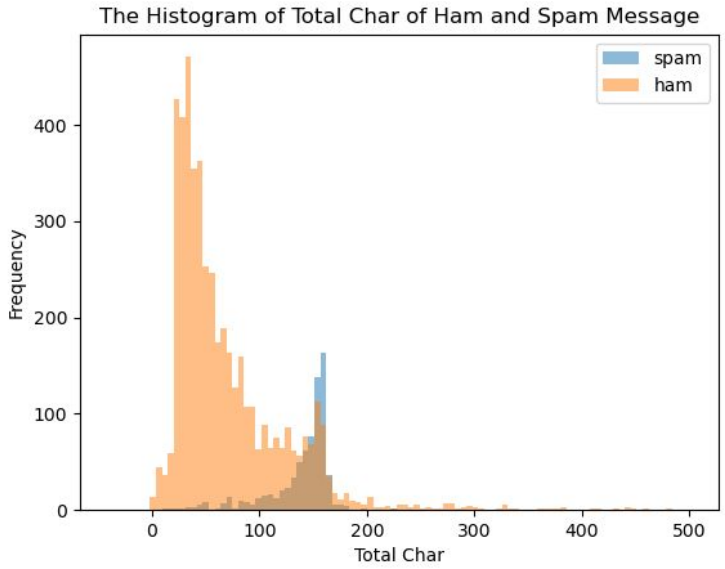
Step 4: Finding the Most Important Feature

In this step, we will focus on finding the most significant predictor for spam messages by using the 5 features one by one with the most effective model we obtained from the last steps. For testing each feature, we are planning on building the model with only this feature and then calculate the accuracy of the model prediction. We assume the most accurate model will use the most important feature.

Result

1. **What information can we get to distinguish the difference between ham and spam messages if we only perform basic exploratory data analysis?**

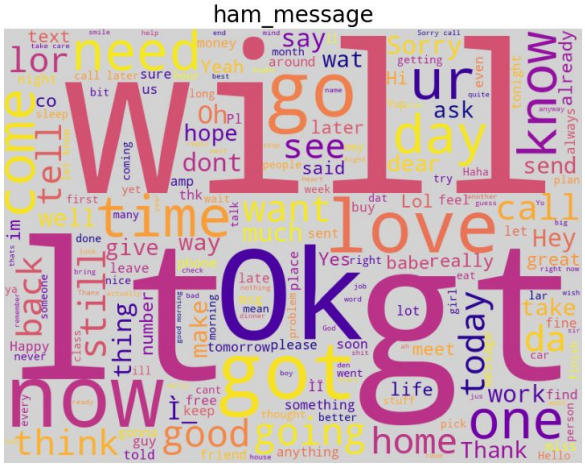
   In this part, we first perform a rough data cleaning. We add binary labels for spam and ham messages and also calculate the total characters of every message. Since for this research problem, we want to do an exploratory analysis to see if there is any difference between spam and ham messages, we plot the histogram of the total characters numbers of both types of messages.



From the plot we can clearly see that the distributions of ham and spam messages are significantly different. For ham messages, the distribution is right skewed meaning that most ham messages will be within 100 characters. In contrast, the distribution for spam messages is left skewed meaning that most spam messages will be in 100 to 200 charactors. Moreover, from the plot we can see that in the dataset, we have ham messages way more than spam messages. If we predict spam messages by only guessing ham messages, we will get 86.6% accuracy. Therefore, in the later part, our most effective model must have accuracy higher than 86.6%.

Besides the distribution of total characters of different types of messages, we are interested in how word choice could be different between ham and spam messages. In order to have a straight forward view, we decided to plot word clouds of different types. In this part, we utilized the Wordcloud package.

ham_message



spam_message

From these two word clouds we can observe a big difference between the word choice of different types of message. Especially in spam messages, the frequency of words such as “free”, “call”, “call now”, and “text” is extremely high. It is understandable because spam messages always try to trick people into their traps, they would always use “free prize” as bait. On the other hand, the ham message contains more diverse word choice.

Therefore, we can conclude that by only using exploratory analysis, the ham and spam messages are different in total characters and word choice.

2. **How can we process the text message to make them machine-readable?**

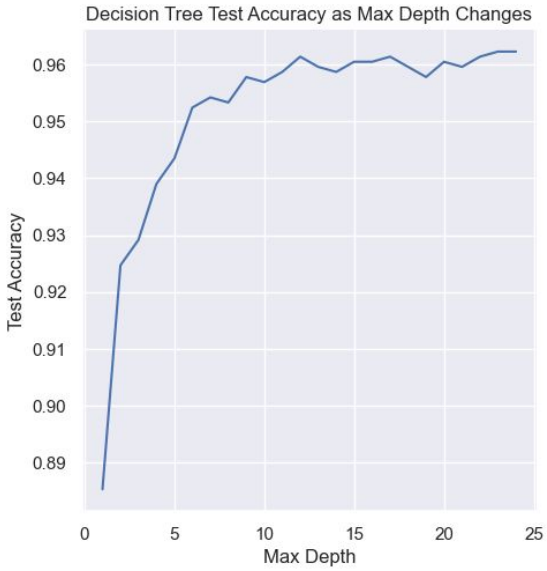
To make text messages in our dataset machine-readable, we use two python files, preprocess.py and analysis.py, to process the dataset we cleaned in the first step.

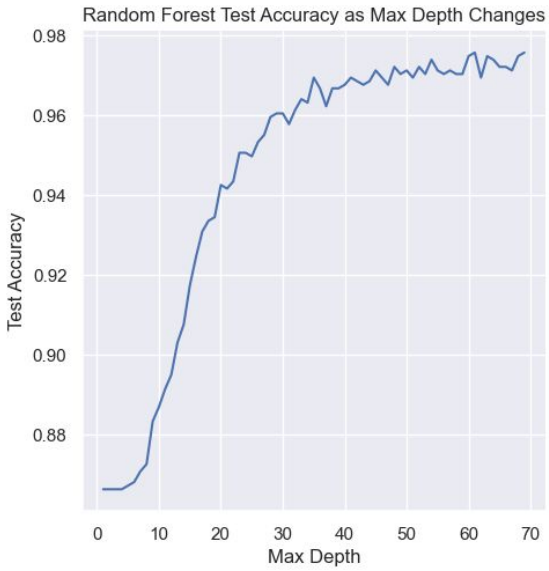
In the preprocess.py, we implement five methods and one helper method in order to add five numerical language features into our dataset. The feature_word_count takes the message DataFrame as input, and add a new column called word count to save the total number of words of each message without all punctuations, url tokens which start with “http” (case non-sensitive) and digit numbers; The feature_average_word_length takes the message DataFrame as input, and calculate the total length of words divided by the total number of words to adds a new column called average word length; The feature_url_count takes the message DataFrame as input and adds a new column called url count to contain the total number of tokens that start with “http” (case non-sensitive) for each message; The feature_special_punc_count takes the message DataFrame as input and adds a new column called spc which contains total numbers of special punctuations with all characters, digit numbers, and periods and commas removed; The feature_tfidf takes the message DataFrame as input, and generates the TF-IDF feature as a sparse matrix.

In the analysis.py, we import the preprocess.py. We read the CSV file and save it as the original DataFrame called message. We first clean the dataset by using the function clean_data in preprocess.py. Then function question2_report takes this cleaned up message DataFrame as input, calls methods in preprocess.py, returns the message DataFrame with four new numerical feature columns, and also generates a matrix that contains the TF-IDF scores between each message and each term in the vocabulary. We convert the raw text messages into numeric features, so this processed message DataFrame can be understood and operated by the Python program. Finally, we save the returned message DataFrame as message and TF-IDF matrix as message_tfidf for later use.

3. **What model is most effective in predicting spam messages?**

In this step, we want to figure out which model among the Decision Tree Classifier, the Naive Bayes Classifier, the SVM Classifier, and the Random Forest Classifier can best predict spam messages. We build each algorithm with these models and get the accuracy of these algorithms to see which one has the highest accuracy. For the Decision Tree and the Random Forest, we find the best max depth for these two models so that the result will not be underfitting or overfitting. Based on our plots (see below), the best max depth for Decision Tree is 12 and the best max depth for Random Forest is 60. Then we build our algorithms of the four models, every time we split our data into a 80% train set and a 20% test set, and we use the same two sets to build the four algorithms. Everytime, the SVM model works the best. The below table is the average accuracy in our 5 times running results. Therefore, we decide the SVM is the best fitting model for predicting spam messages, and we will use this model in the following finding the most important feature part.

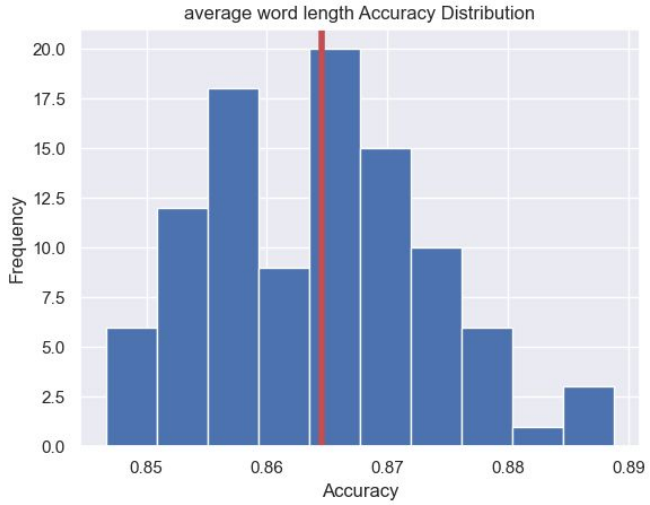


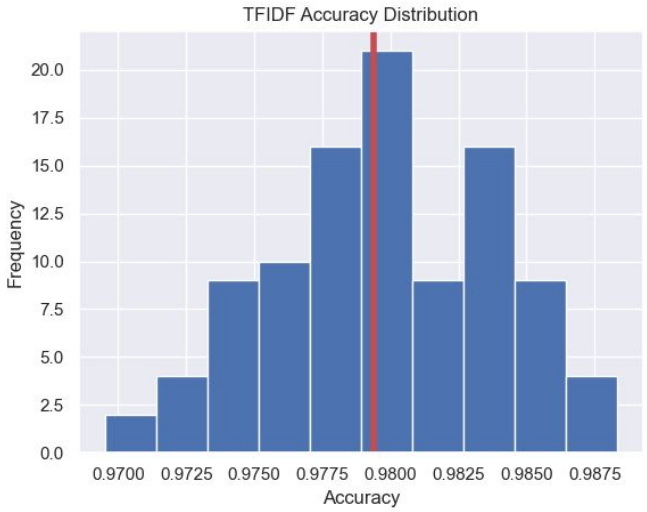


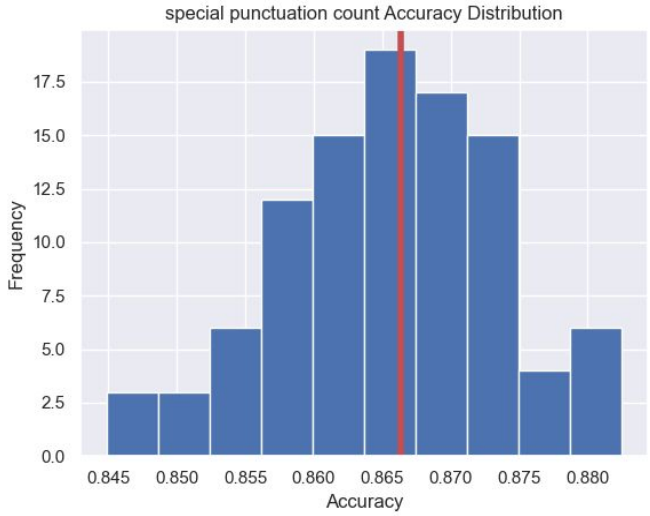
Decision Tree	Naive Bayes	SVM	Random Forest
0.9614349775785	0.8672645739910	0.9820627802691	0.9704035874439

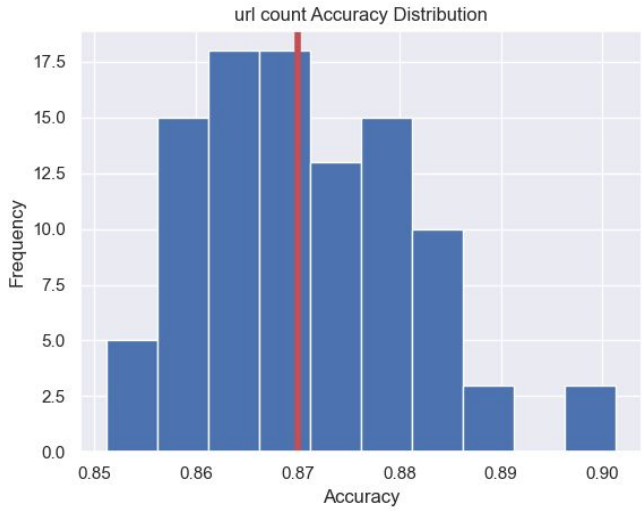
4. **What features are most important in predicting spam messages?**

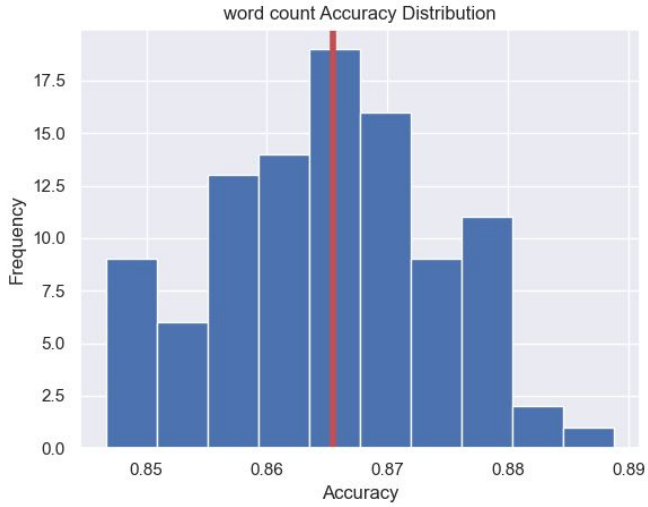
In this part, we are interested in determining the most important feature for our most effective model. We use the SVM model and at each time we only pass in one feature at a time and then we calculate the accuracy of this one feature model for 100 times. Those plots above are the distribution we get from each one-feature model. As we can see in the plot, the red line indicates the median value of all 100 runs. We are surprised that only TF-IDF is achieving an impressive result which is around 0.98 and all the other four features only achieve around 0.86 accuracy which is roughly the same as only guessing ham messages for prediction. We think this result might be due to two reasons. The first one is that the other four features are not the main feature for the prediction model. Another reason which is the one we believe is that our dataset is heavily biased towards ham messages so that the presence of spam messages could be considered as rare cases. Therefore, although we have good features extrapolated from the text, it's still hard for one feature to do an accurate prediction. On the other hand, the TF-IDF feature takes some potential features into account such as term frequency and inverse document frequency so that it gives an impressive result than all the others.


average word length Accuracy Distribution
Frequency
Accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.85
0.86
0.87
0.88
0.89


TFIDF Accuracy Distribution
Frequency
Accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.9700
0.9725
0.9750
0.9775
0.9800
0.9825
0.9850
0.9875


special punctuation count Accuracy Distribution
Frequency
Accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.845
0.850
0.855
0.860
0.865
0.870
0.875
0.880





Challenge Goals

1. **Machine Learning**

   Our project meets the goal of Machine Learning since we look through various model types, including Decision Tree Classifier, Naive Bayes Classifier, SVM Classifier, and Random Forest Classifier, and the different settings of hyperparameters, max depth for Decision Tree and Random Forest, to decide which model possesses higher accuracy in predicting spam messages.

   This is a challenge for us since we use some models that we do not know before, so we need to read the documentation and to understand the meaning of the model and those hyperparameters for them. This asks us to use our capability to read

documentation and some other resources and examples we could find online, which is a very important skill for us in the future.

2. **New Library**

In addition, we use a new library, wordcloud package, of advanced visualizations to perform the analysis of the differences between spam and ham messages (machine learning models performance) and use a new library of natural language processing to process the large volume of the text in our dataset. For calculating the TF-IDF matrix, we import CountVectorizer and TfidfTransformer from the sklearn.feature_extraction.text package and also we implement stopword from the nltk package. While doing the machine learning part, we use the hstack from sklearn to combine the TF-IDF matrix with the other features.

During the time we try to analyze our dataset of spam and ham messages, it is a big challenge for us to find some useful packages to help us to represent the information or to calculate the TF-IDF, which is a very important feature in our following steps. So we searched for some examples and read some documentation about how to represent words in image and how to create a TF-IDF matrix. This is also an important skill that we may use a lot in the future.

Work Plan Evaluation

Task 1: Process Data

In this task our three group members will work together for about 3 days to process our dataset. We will discuss and analyze the data together, and then we can figure out how to build our dataset that will be used in the following tasks. We will clean the data and build the five features we want for our dataset. Specifically, we will use WeChat to have a group chat to discuss together and use visual studio live share to write code together.

Evaluation:

This work plan is a good estimation for this task. Cleaning up the dataset is fairly easy. Also we did expect extracting features from text could be time consuming so we gave ourselves 3 days to do the task. Although in the end we did not use visual studio live share, we developed our project on Ed instead and it went well.

Task 2: Finding Effective Model

For finding the most effective model, we will separately write code in visual studio to test which model is the best one with the highest accuracy for predicting spam messages. Each of us will test at least 2 models and after that we will share our results and choose the best model to use. We will spend about 4 days completing this task.

Evaluation:

The estimation work plan for this part of the project is fair. As soon as we decided that we were going to use only four models as our competing models, we started to build them. Checking the documentation could kill some time and since we had the TF-IDF as a sparse matrix, it took sometimes for us to figure out how we should combine a matrix features to our features in the dataframe. The most difficult part is finding the hyperparameters. Since we were not familiar with some models, we need to check lots of documentation to decide if the max_depth value is suitable for all of the models.

Task 3: Finding Important Features

Finally, for finding important features, we will separately write code in visual studio to test the importance of each of the 5 features (word count / average word length / URLs count / special punctuation count / TF-IDF). Then we will come together and choose some important features with higher accuracy in predicting spam messages to build our final predicting model. This whole process will take about 4 days to finish.

Evaluation:

The estimation for this part is a little bit overestimated. As soon as we decided the most effective model from the previous part, finding out the important feature(s) is a fairly easy job. Although we encountered some surprising results in the end, the whole process of this part went smoothly and took only 2 days.

Testing

We have tests for the preprocess.py which contains feature extrapolation and the exploratory analysis. We borrowed the cse163_utils.py from a homework assignment so that we can use the assert_equals() function to compare the expected value with the value we get from the preprocess.py. Also, for testing our feature extraction function, we create our own test file with all the cases included so that we can make sure we did not miss any rare cases in the actual dataset.

Collaboration

Online source: https://www.kaggle.com/dejavu23/sms-spam-or-ham-beginner

Citation

ABC News, ABC News Network,
abcnews.go.com/blogs/technology/2012/08/69-of-mobile-phone-users-get-text-spam.