SMS Spam Detection and Analysis using Text Mining

1. **Description of the particular problem within the selected data mining topic to be addressed in this project:**
   People receive hundreds of SMS's each day, but not all of them are relevant or from genuine senders. This project aims to go through these messages, analyze and perform text mining on the contents of the message, and classify them as relevant or otherwise. This could help the user filter out spam messages, saving him/her a lot of time and bother to only look at the useful messages.

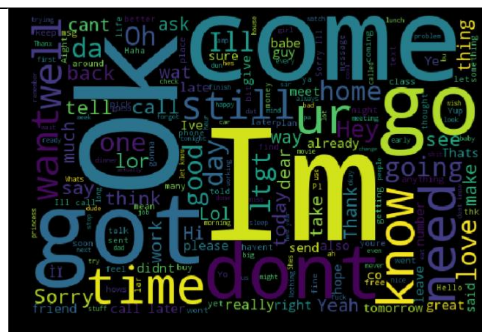
2. **Description of the approach used in this project to tackle the above problem:**
   Classification techniques like K-nearest neighbors, Decision trees, Logistic regression and Random Forest were used to classify the text messages as ham or spam.
   The complete dataset was divided into 75% training and 25% test data sets in order to test various classification models.
   For clustering, methods like K-means and DB Scan were used with different parameter values.

3. **Dataset Name: SMS Spam Collection**

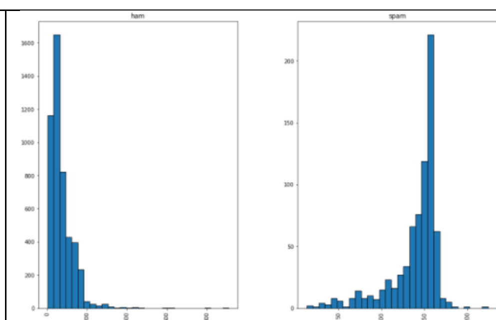
4. **Where found: UCI Machine Learning Repository**

5. **Dataset Description:**
   The SMS Spam Collection is a public set of SMS labeled messages that have been collected for mobile phone spam research. It is a collection composed of 5,572 English, real and non-encoded messages, tagged accordingly being legitimate (ham) or irrelevant and inappropriate(spam).
   The files contain one message per line. Each line is composed of two columns: v1 contains the label (ham or spam) and v2 contains the raw message text.
   The dataset contains 4825 ham messages and 747 spam messages.

Spam Text Word Cloud



Ham Text Word Cloud



Histogram of length of ham and spam messages respectively

From the histogram above, we can see that ham messages tend to be shorter than spam, and therefore might be an important factor in ham/spam classification.

6. **Initial data preprocessing, if any:**
   Dropped irrelevant/empty columns, renamed remaining columns to be more comprehensible.
   Removed punctuations and stop words from text messages.

7. **Three Guiding Questions about the dataset domain:**
   1. How accurately can an SMS message be classified as spam or otherwise by looking at the words in the message?
   2. How accurately can spam messages be further classified into subcategories like Lottery, Free, Adult, Attention and Carrier spam?
   3. Is it possible to cluster the text messages into meaningful and well-defined groups?

Summary of Experiments. *At most 2 pages.*

Tool	Guiding Ques	Mining Technique	Pre-process	Parameters	Results (Accuracy: A, Precision: P, Recall: R)	Time taken	Evaluation	Observations about experiment, visualization, Interpretation of results
Python	1	Text mining, KNN classification Including message length	Converted document to TF-IDF matrix Calculated message length	Number of neighbors = 40	A: 0.921 P: 0.93 R: 0.92 A: 0.899 P: 0.89 R: 0.90	0.126s 8.046s	KNN gives pretty good accuracy score of 92%, and including length in the model doesn't help, and even takes longer time. It is worth noting that KNN classifies spam messages containing the word 'free' as being a ham mssg.	[[1200 1] [109 83]] Looking at the above confusion matrix, it can be said that KNN classifies almost all the ham messages correctly, with a recall of close to 1. But when it comes to spam messages, the KNN does a poor job of classification, misclassifying 109 instances
Python	1	Text mining, Decision tree classification Including message length	Converted document to TF-IDF matrix Calculated message length	min_samples_split = 7	A: 0.955 P: 0.96 R: 0.95 A: 0.958 P: 0.96 R: 0.96	0.28s 17.031s	Decision trees gives an accuracy of 96%, and including message length has no effect.	[[1167 34] [29 163]] The above confusion matrix shows that Decision trees does a better job of classifying spam messages than KNN, and also does a fare job with ham mssgs.
Python	1	Text mining, Logistic regression classification Including message length	Converted document to TF-IDF matrix Calculated message length	Solver = 'liblinear', penalty = 'l1'	A: 0.944 P: 0.94 R: 0.94 A: 0.949 P: 0.95 R: 0.95	0.017s 0.133s	Logistic Regression gives 95% accuracy taking very less time, and including message length minutely improves the scores.	[[1190 11] [67 125]] Logistic Regression performs well in classifying ham messages with just 11 misclassifications, but lacks in spam message classification.
Python	1	Text mining, Random Forest classification Including message length	Converted document to TF-IDF matrix Calculated message length	Number of trees = 33	A: 0.97 P: 0.97 R: 0.97 A: 0.973 P: 0.97 R: 0.97	1.622s 10.927s	Random Forest has the best accuracy score of 97%, including message length takes longer time but doesn't affect the scores much. Increasing or decreasing the number of trees from 33 negatively affects the score.	[[1201 0] [[1201 0] [41 151]] [37 155]] Random Forest has the perfect score when it comes to ham classification, with 100% recall. It also fares well when dealing with spam with 41 misclassifications. When considering message length in the model, this number comes down to 37.
Python	2	1. Text mining, KNN classification 2. Including message length	Converted document to TF-IDF matrix 2. Calculated message length	Number of neighbors = 40	1. A: 0.594 P: 0.61 R: 0.59 2. A: 0.455 P: 0.48 R: 0.45	0.031s 0.179s	KNN performs poorly in classifying spam messages into subgroups, with or without including message length.	Looking at the heat map & confusion matrix, it can be said that KNN overall perfor ms poorly with spam sub cl assification, the worst bein g with adult spam.

Python	2	Text mining, Decision tree classification Including message length	Converted document to TF-IDF matrix Calculated message length	min_samples_split = 7	A: 0.84 P: 0.84 R: 0.84 A: 0.818 P: 0.82 R: 0.82	0.033s 0.111s	Decision trees does a good job in spam sub-classification, considering the number of distinct classes (6), taking very little time.	The confusion matrix for Decision trees shows that it correctly classifies all the lottery spam messages, with few misclassification errors for rest of the spam messages.
Python	2	Text mining, Logistic regression classification Including message length	Converted document to TF-IDF matrix Calculated message length	Solver = 'newton-cg', penalty = 'l2'	A: 0.775 P: 0.84 R: 0.78 A: 0.765 P: 0.84 R: 0.76	0.001s 0.031s	Logistic regression gives an accuracy of 77.5% in very less time, but the scores drop a bit when message length is also considered.	The confusion matrix for Logistic Regression shows that it correctly classifies all the Free spam messages, thus giving a recall of 1. But when it comes to 'other' spam messages, it performs poorly with recall of only 0.24.
Python	2	Text mining, Random Forest classification Including message length	Converted document to TF-IDF matrix Calculated message length	Number of trees = 33	A: 0.882 P: 0.89 R: 0.88 A: 0.888 P: 0.90 R: 0.89	0.156s 0.168s	Random Forest, once again, gives the best classification score of 88-89%, and in this case, considering message length does help. [[11 0 0 0 0 6] [0 13 0 0 0 0] [0 0 14 1 0 5] [0 0 0 28 1 1] [0 0 0 0 39 5] [2 0 0 0 0 61]]	The confusion matrix shows how Random Forest achieves the best accuracy score. It correctly classifies all the lottery spam messages, with very few errors for the other categories. It classifies few 'carrier' spam as 'free' spam due to certain messages which contain words like 'free sms' and thus is difficult to classify.
Python	3	Text mining, K means clustering	Converted document to TF-IDF matrix	k = 4, max_iter = 100		1.508s	K means does not give good results, but manages to cluster ham messages containing the word 'sorry' into one grp.	K-means does not give any significant clusters to draw inferences from, and hence is not useful as a clustering method on this data set. One can say it clusters 'sorry' messages into a grp.
Python	3	Text mining, DBScan clustering	Converted document to TF-IDF matrix	Eps = 0.8, min_samples = 20 Eps = 0.3, min_samples = 10	Silhouette Coeff: -0.0003 -0.283	1st run: 183.895 s 2nd run: 45.54s	DB Scan performs better than K means, giving well defined clusters for few kinds of messages. But it takes a lot of time for the cluster results to be generated.	DB Scan is a good technique to be applied here, clustering ham messages with 'sorry I'll call later', 'ok', 'call', 'i cant pick the phone right now' into separate clusters. Spam messages are clubbed with few ham ones as outliers, on which DB Scan can be recursively applied to form more meaningful clusters.

Analysis of Results: (at most 1 page)

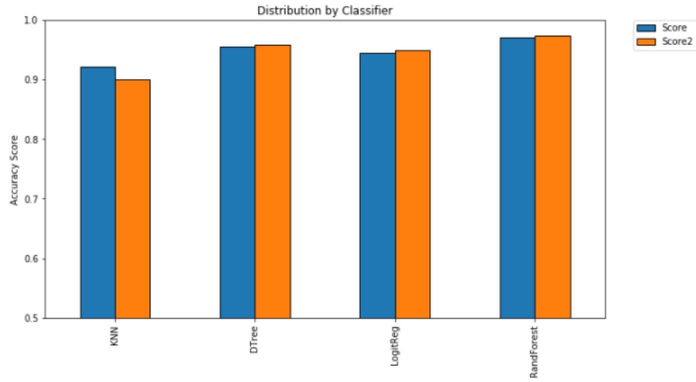


Fig. 1(above) and Fig. 2(below)

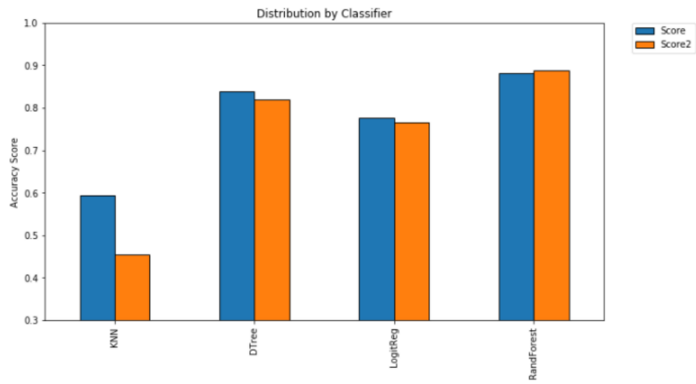
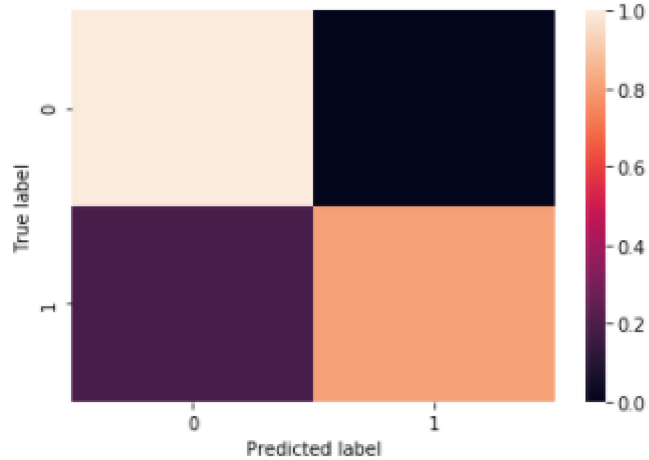
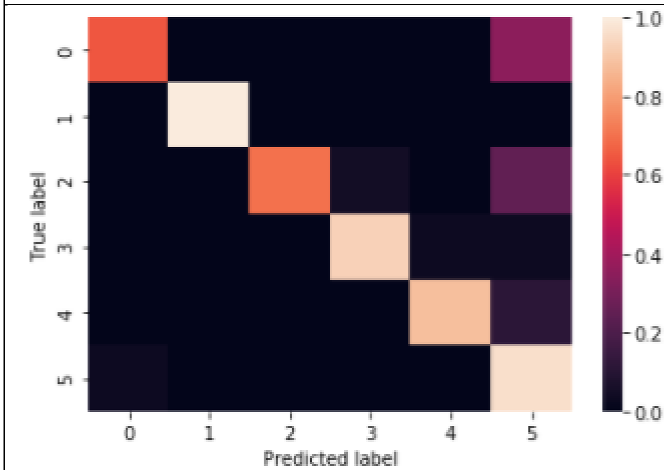
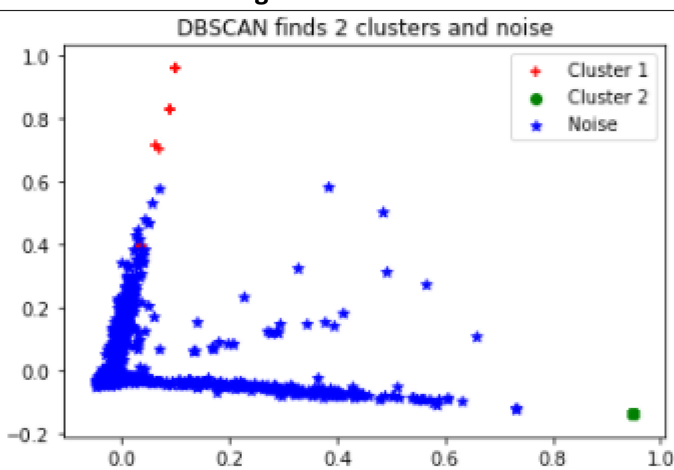


Fig.1 shows the plot of **accuracy scores** for various classifiers used for **ham/spam classification** not considering(**blue**) and considering(**orange**) message length. **Fig.2** shows the same for **spam sub-classification**. Thus, we can see that **Random Forest** performs the **best** in answering both the guiding questions, and the corresponding **heat maps** for the best results are shown **alongside** →

```
Using  RandForest  Classifier:-
[[1201    0]
 [  37  155]]
```







Based on the experiments carried out, we can conclude the following for the 3 guiding questions: **(1)** Yes, it is possible to classify an SMS message as ham/spam, by using Random Forest technique, to an accuracy of 97%, which would be very useful in filtering out unwanted messages. **(2)** Spam sub-classification is also possible to the extent of 88% accuracy, again by using Random Forest. This result is also appreciable, considering that there were 6 classes to classify. The accuracy will reduce as we define more classes, but still this model will help those who wish to study and analyze spam messages, and are looking for ways to block them. **(3)** Clustering of text messages is possible to some extent using DB Scan, which groups messages containing similar combination of words. An example of one such clustering can be seen on the left side.

Project Learnings: -

Through this project, I mainly learnt about text mining and the various steps involved in extracting words, cleaning them and then perform operations like classification and clustering. Working independently on this project allowed me to think from the domain perspective, come up with questions/problems, plan out the solution and tasks, make inferences from the results, and also face the challenges, on my own, that came up during the course of the project.