Best SMS Spam Model

1 First Model

The first model is just a *Decision Tree Model* with the following settings:

- Each leaf should only contain 1 sample label as 1 or 0
- Features from first 10 Mutual Information words in the **Training Set**

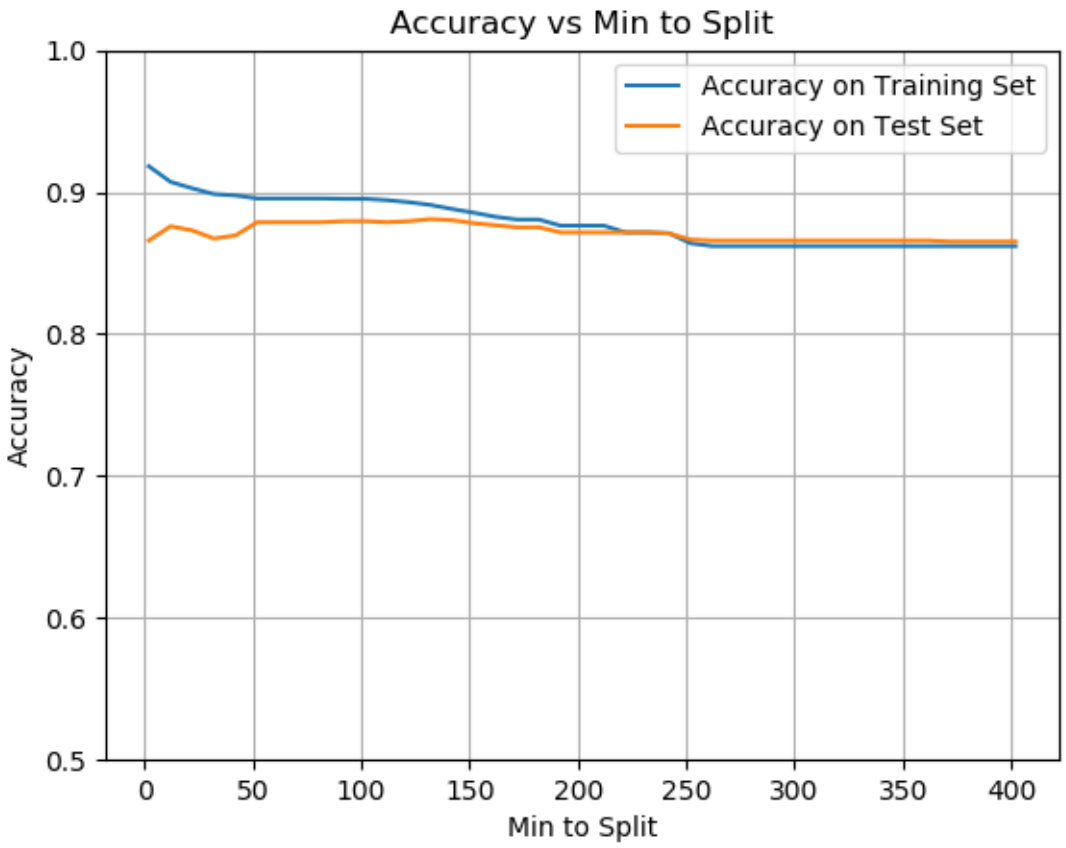
The accuracy of this model on **Test Set** is 86.5854% with error bound [84.7963%, 88.3745%].

2 Check for Overfitting and Underfitting

By instinct, we can know that when every leaf only contains 1 sample in the *Decision Tree Model*, there must have some Overfitting on the **Training Set**.

2.1 Find the best min_to_split

By sweeping the min_to_split parameter (from 2 to 402, step as 10), we have the following accuracy curves on both **Training Set** and **Test Set**.[1]



[1] For raw program output of the whole report, please check the file *Result-BestModel.txt*. This file includes all the outputs that every parameter sweeping

For the curves we can find that:

- When min_to_split parameter is less than 50, the accuracy gap between **Training Set** and **Test Set** is large. And as the min_to_split parameter increase, the gap become smaller, this means in this area, the *Decision Tree Model* is overfitting based on the **Training Set**.
- When min_to_split parameter is greater than 150, accuracy from both **Training Set** and **Test Set** become lower and lower, this means the *Decision Tree Model* become underfitting in this area.

With this experiment, we can use min_to_split parameter as 125.

3 Feature Engineering

When find the best features we need to use, we did the following features engineering works to make sure the features we used is the good features for the model.

3.1 Manually Craft Features

3.1.1 Look at the Training Set

By looking at the **Training Set**, we can find these features by instinct:

- Contains words that fully uppercase
- Contain pure number words
- Contains words neither in alphabet nor numbers
- Length of the sentence
- Count of the words in sentence
- Contains URL
  - Start with "http"
- Contains Uncommon Punctuation

3.1.2 Leave-Out-One Wrapper Search on Features

To determine if those manual selected features is good or not, we use cross-validation on the leave-out-one wrapper search on those features, and here is the result after the first iteration:

Without "Has Full Uppercase Word"	Without "Has Number Word"	Without "Has Special Word"
86.3397%	84.0670%	85.7895%

without 'Length'	without 'Word Count'	without 'Start with "http"'	without 'Has Uncommon Punctuation'
85.6459%	86.1244%	85.7895%	86.3397%

As the base line, when we use all 8 features, the cross-validation accuracy is 86.0526%.

In this iteration we can find that for feature: "**Contains Uncommon Punctuation**" is totally useless and without this feature the accuracy is even higher than others. So, will remove this feature in next iteration.

3.1.3 Categorize Mistakes

And, in the first iteration, we can output all the mistake the *Decision Tree Model* has made, and manually exam them. For example:

- we can find that some spam message contains the URL, but not start with the "http", but have ".com", we will add this feature in next iteration for checking if that feature affects a lot.

3.1.4 Finally

After several times of iterations of Leave-Out-One Wrapper Search on Features and manual inspect the mistakes of the *Decision Tree Model*. We choose the following 5 features for future use:

- Contains uppercase words
- Contain pure number words
- Contains words neither in alphabet nor numbers
- Length of the sentence
- Start with “http”

3.2 Just Enough Mutual Information Features

Then we can find enough word features from the **Training Set** by compute the mutual information between the words and the labels.

But before we use cross-validation and parameter sweeping to find out the enough word features count, we normalize the data first to make the data easier to be use.

3.2.1 Normalize

After the manual check, we can normalize the data with the following step:

- Lowercase all data

By doing so, will simplify our Mutual Information words set, such as, we will consider “CALL” and “call” will be in one feature instead of been 2 different features.

3.2.2 Parameter Sweeping

By sweeping the number of Mutual Information Features, from 10 to 200 step 10, we can find that when there have 70 Mutual Information Features, the *Decision Tree Model* has best performance.

4 Use Random Forests

After getting good features and parameters for the *Decision Tree Model*. Now we can expand the model to use the *Random Forests Model*.

4.1 Parameter Sweeping

For built up the *Random Forests Model*, we need another round of parameter sweeping for find the good parameters for the *Random Forests Model*. Here is the sweeping range:

- Number of *Decision Tree Model*
  - Range 10 to 150
- Use Bagging when training
  - True or False
- Restrict Features number for each *Decision Tree Model*
  - 0 to 70, 0 means use all features in every *Decision Tree Model*

And since the *Random Forests Model* can correct for *Decision Tree Model*' habit of overfitting to the training set. We need sweep the min_to_split (range 2 to 152) again.

As the output shows, the *Random Forests Model* has best cross-validation accuracy of 89.0670%, when using the following parameters:

- Number of *Decision Tree Model* 50
- Use Bagging when training True
- Restrict Features number for each *Decision Tree Model* 70
- min_to_split 50

4.2 Evaluation the Best Model

By using the best *Random Forests Model* for predicting the **Test Set**, we can get the following evaluation statistic:

- Accuracy: 90.0287%
- Error Bound: [88.4558%, 91.6016%]
- Precision: 88.5057%
- Recall: 56.4103%
- False Positive Rate: 1.7841%
- False negative Rate: 43.5897%

And the Confusion Matrix as:

	PREDICT TRUE	**PREDICT FALSE**
ACTUALLY TRUE	154	119
ACTUALLY FALSE	20	1090

4.3 Compare with First Model

At the beginning of this report, the first *Decision Tree Model* we built has the 86.5854% accuracy on **Test Set** with error bound [84.7963%, 88.3745%]. Since our best *Random Forests Model* have the lower bound 88.4558%, which is higher than the first model's upper bound. We can now say the best *Random Forests Model* is guaranty better than the first *Decision Tree Model*.

The High Recall here might because of the noise. When we do the manual mistake categorization, we can find that many of the SMS message can be marked as spam, but the data is not.

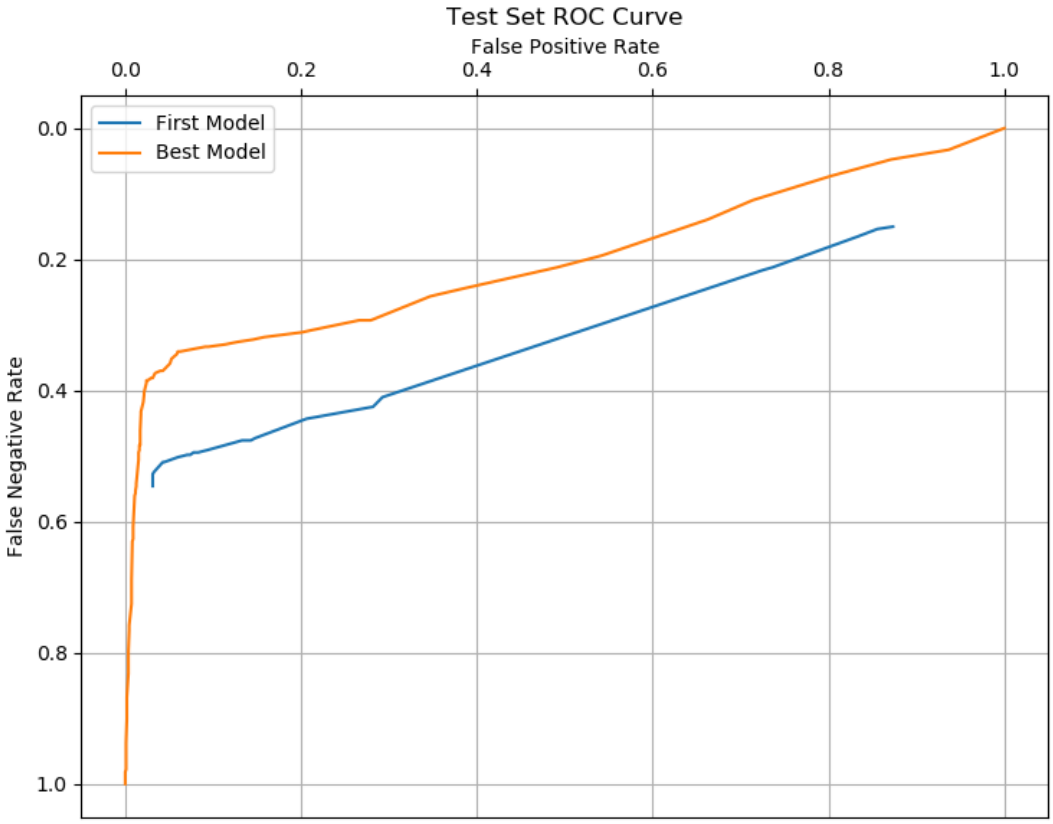
4.3.1 ROC Curve

Here is the ROC Curve between the first *Decision Tree Model* and the best *Random Forests Model*.

From the picture we can find that:

- Since the first *Decision Tree Model* is overfitting on the Training Set, some certain case of the FNR and FPR cannot be reached.
- When the 2 models have same FPR, the FNR from best *Random Forests Model* will be lesser than first *Decision Tree Model*.
- When the 2 models have same FNR, the FPR from best *Random Forests Model* will be lesser than first *Decision Tree Model*.

- Overall, best ***Random Forests Model*** will guaranty performance better than first model.



4.3.2 Determine the Threshold

Depended on the end user requirement, we can update our threshold to make the model behavior differently. Such as:

- When require almost all the spam messages should be mark as the spam and even there have many False Positives are still OK. In this case, we can move our threshold to very small (for certain value, need check the raw output from *Result-BestModel.txt*) to meet the requirement.
- Another case is that, when user can accept to see some spam message in the Inbox, but want message marked as spam are the real spam message. We can move our threshold to very large (for certain value, need check the raw output from *Result-BestModel.txt*) to make it performance like this way.