SpamShield

Rachit, Rudradeep, Vineet

Outline

Overview

Spam detection: A challenge

Dataset and References

Preprocessing

Classification Part 1: Tuning the classifiers

Classification Part 2: Classifier Results

Introducing: SpamShield

Overview

Mobile phone spam also known as (unsolicited messages, especially advertising), directed at the text messaging or other communications services of mobile phones or smartphones. Fighting SMS spam is complicated by several factors (compared to Internet email), including the lower rate of SMS spam, which has allowed many users and service providers to ignore the issue, and the limited availability of mobile phone spam-filtering software.

In our project, we analysed different methods to identify spam/ham messages. We used different approaches to establish relation between the text and the category, based on size of message, word count, and term-frequency inverse document-frequency (tf-idf) transform.

Filtering SMS spam is still a challenge...

01 lack of public and real datasets

02 low number of features that can be extracted per message

03 fact that the text is rife with idioms and abbreviations

Dataset And References

- The dataset for this project is taken from the [UCI Machine Learning Repository.]
- This dataset is comma-separated values (CSV) file.
- Contains ~5000 SMSs with message and their class label : {ham, spam}
- Gómez Hidalgo, J.M., Almeida, T.A., Yamakami, A. On the Validity of a New SMS Spam Collection. Proceedings of the 11th IEEE International Conference on Machine Learning and Applications (ICMLA'12), Boca Raton, FL, USA, 2012
- Almeida, T.A., Gómez Hidalgo, J.M., Silva, T.P. Towards SMS Spam Filtering: Results under a New Dataset. International Journal of Information Security Science (IJISS), 2(1), 1-18, 2013.

Out[127]:

	class	message
0	ham	LOL we will party at night
1	spam	Win 1 million dollars jackpot. Call now
2	ham	Yes I am watching netflix
3	ham	I am going to school tomorrow
4	spam	you have won a lottery ticket. call and claim ...
5	spam	Winner! You have been rewarded with a gift cou...

Preprocessing

Preprocessing

- Remove stop words using nltk stopwords
- Remove punctuation
- Stemming: Reduce words to their roots using nltk SnowballStemmer
- Convert words to lowercase

Before

```
Out[128]: 0                    LOL we will party at night
          1            Win 1 million dollars jackpot. Call now
          2                     Yes I am watching netflix
          3                  I am going to school tomorrow
          4    you have won a lottery ticket. call and claim ...
          Name: message, dtype: object
```

After

```
Out[130]: 0                    lol parti night
          1    win 1 million dollar jackpot call
          2                yes watch netflix
          3               go school tomorrow
          4     lotteri ticket call claim prize
          Name: message, dtype: object
```

TF-IDF Vectorization (Bag of words Representation)

- Turn a collection of text documents into numerical feature vectors
- SMS is described by word occurrences while completely ignoring the relative position information of the words

Model:

- Tokenizing strings and giving an integer ID for each possible token
- Counting the occurrences of tokens in each SMS
- Normalizing and weighting with diminishing importance tokens that occur in the majority of SMSs (tf-idf normalized weighting)

TF-IDF vectorization

- Extract tokens from SMS string
- Tokens become feature names

```
['call', 'claim', 'coupon', 'day', 'dollar', 'gift', 'go', 'jackpot', 'lol', 'lotteri', 'million', 'netfli
x', 'night', 'parti', 'prize', 'reward', 'school', 'ticket', 'tomorrow', 'valid', 'watch', 'win', 'winne
r', 'yes']
```

Feature array of 1st SMS.
The 9th word “lol” has TF-IDF weight of 0.577.

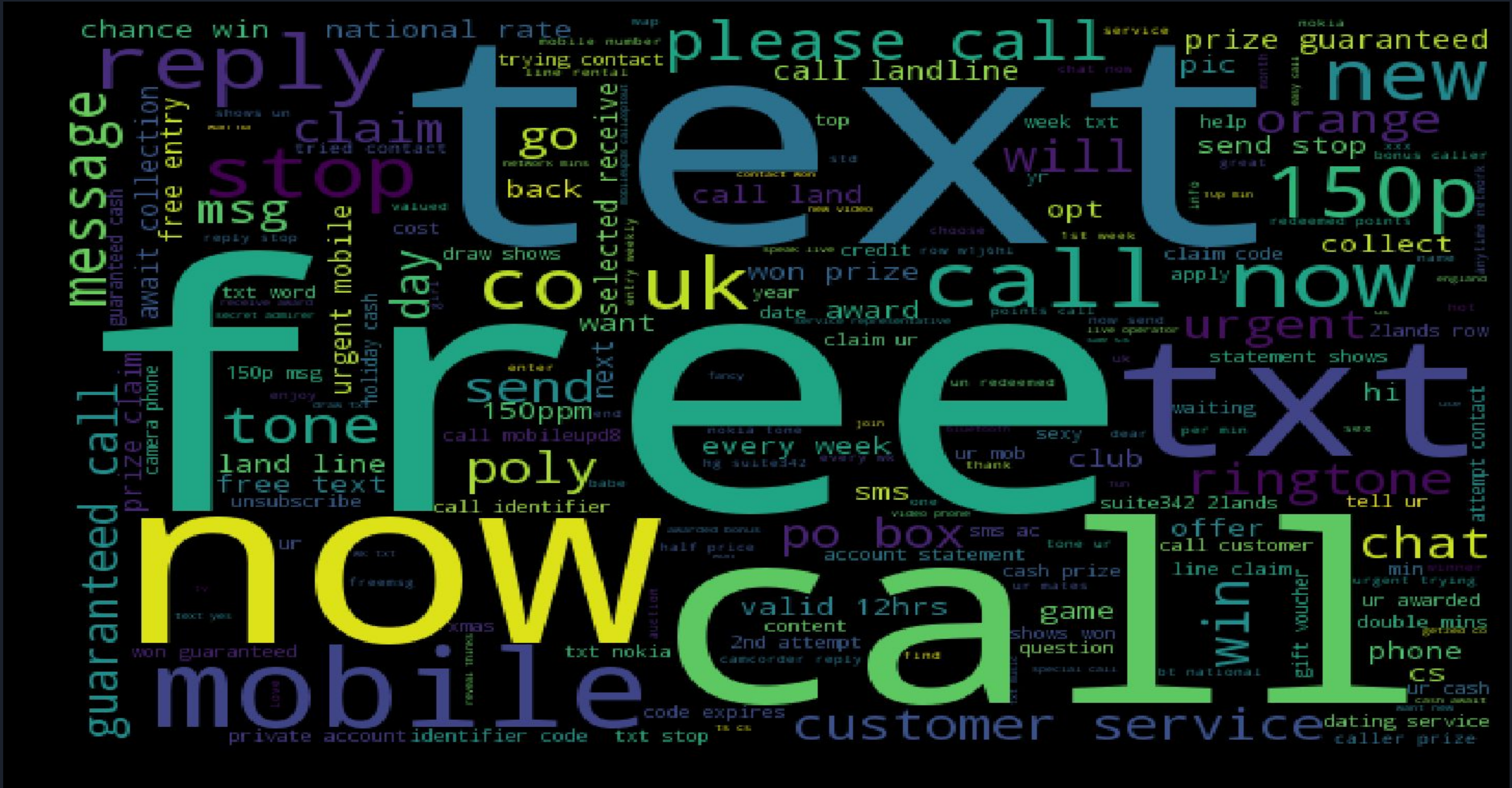
```
[[ 0.          0.          0.          0.          0.          0.          0.
   0.          0.57735027  0.          0.          0.          0.57735027
   0.57735027  0.          0.          0.          0.          0.          0.
   0.          0.          0.          0.        ]
```

TF-IDF vectorization

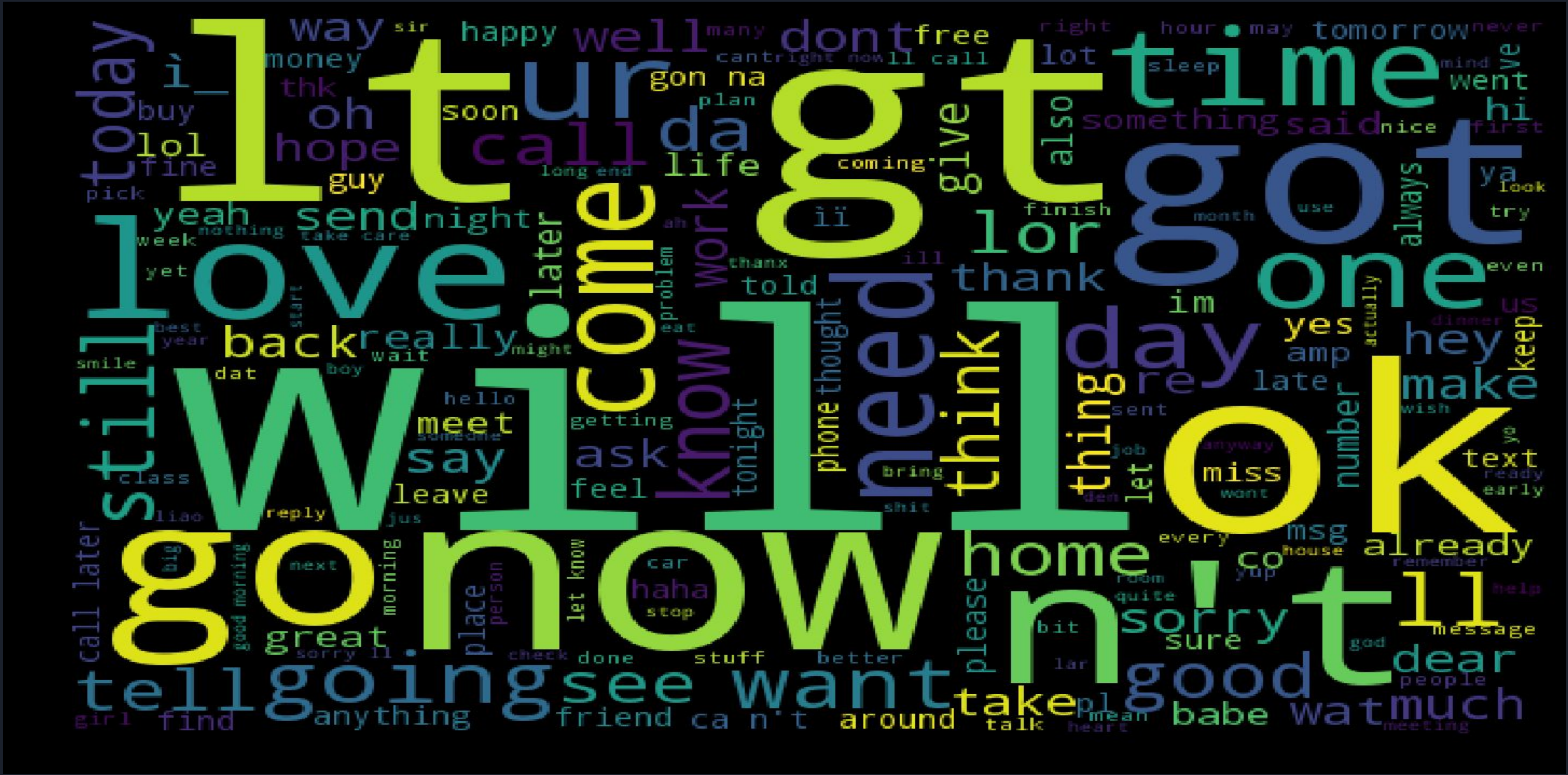
- Convert features array to matrix form
- Matrix with 1 row per sms and 1 column per token
- Each term found by the analyzer during the fit is assigned a unique integer index
- (1,7) 0.46 means that for 2nd SMS, the TF-IDF of 8th feature is 0.46
- Feed the matrix into classifier

```
(0, 8)        0.57735026919
(0, 13)       0.57735026919
(0, 12)       0.57735026919
(1, 21)       0.462624791156
(1, 10)       0.462624791156
(1, 4)        0.462624791156
(1, 7)        0.462624791156
(1, 0)        0.379358946687
(2, 23)       0.57735026919
(2, 20)       0.57735026919
(2, 11)       0.57735026919
(3, 6)        0.57735026919
```

Top spam keywords



Top ham keywords



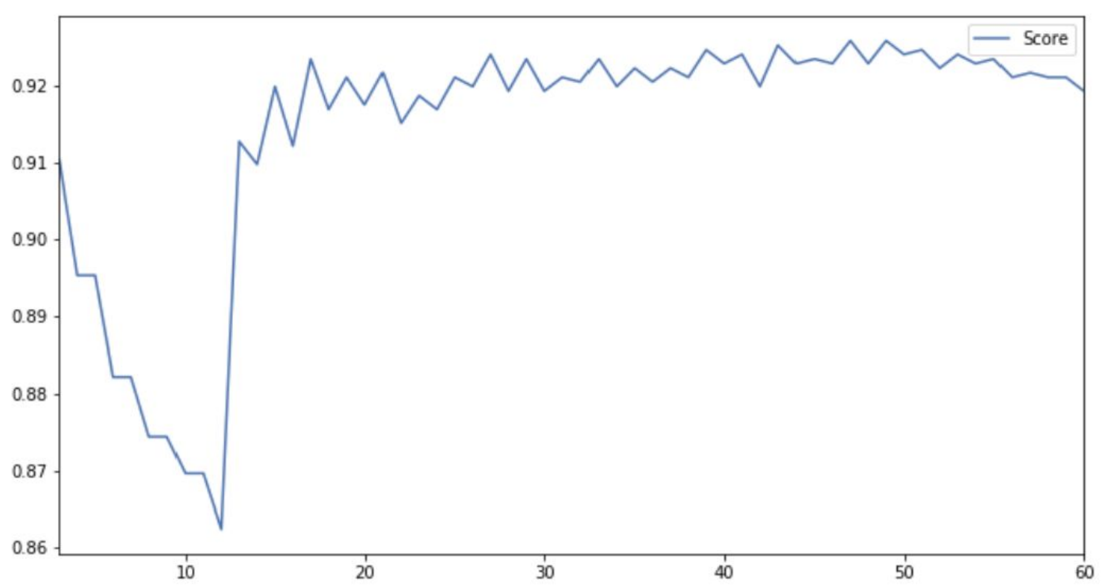
Classification Part 1: Tuning the classifiers

Data Split
70% train. 30% test.

k-neighbours classifier

- Classifier implementing the k-nearest neighbors vote.
- Run classifier for k in {3,61}.

```
Out[138]:  <matplotlib.axes._subplots.AxesSubplot at 0x285a2c66160>
```

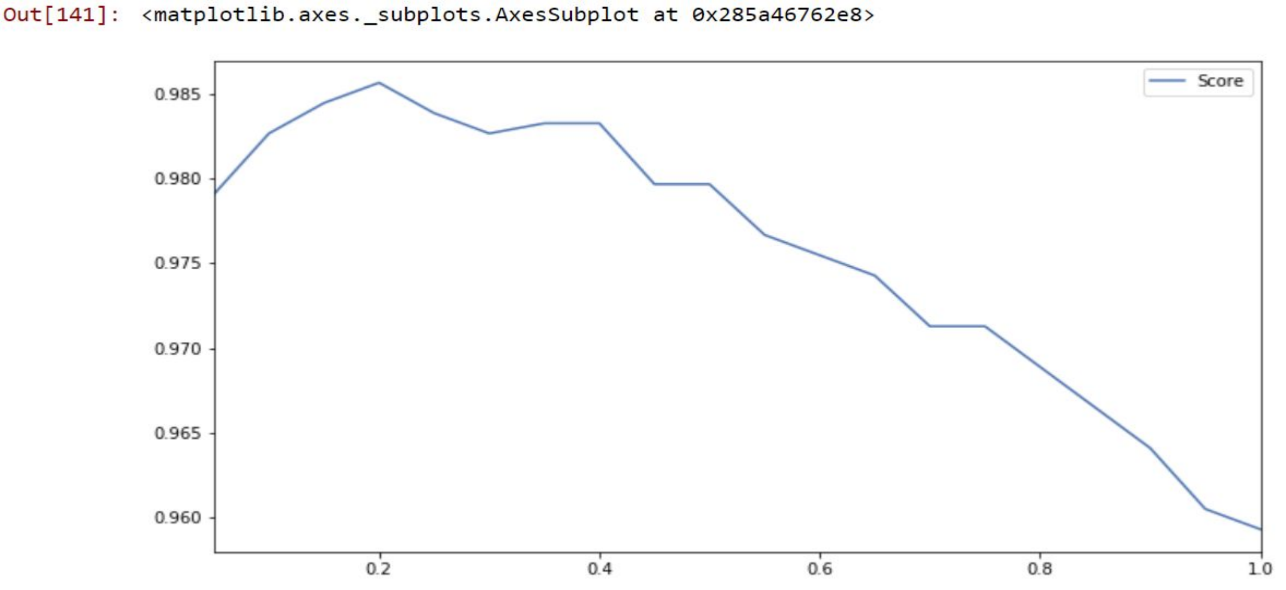


Highest accuracy at k = 49

	Score
47	0.925837
49	0.925837

Multinomial Naive Bayes classifier

- Naive Bayes classifier for multinomial models
- Suitable for classification with discrete features like TF-IDF weights



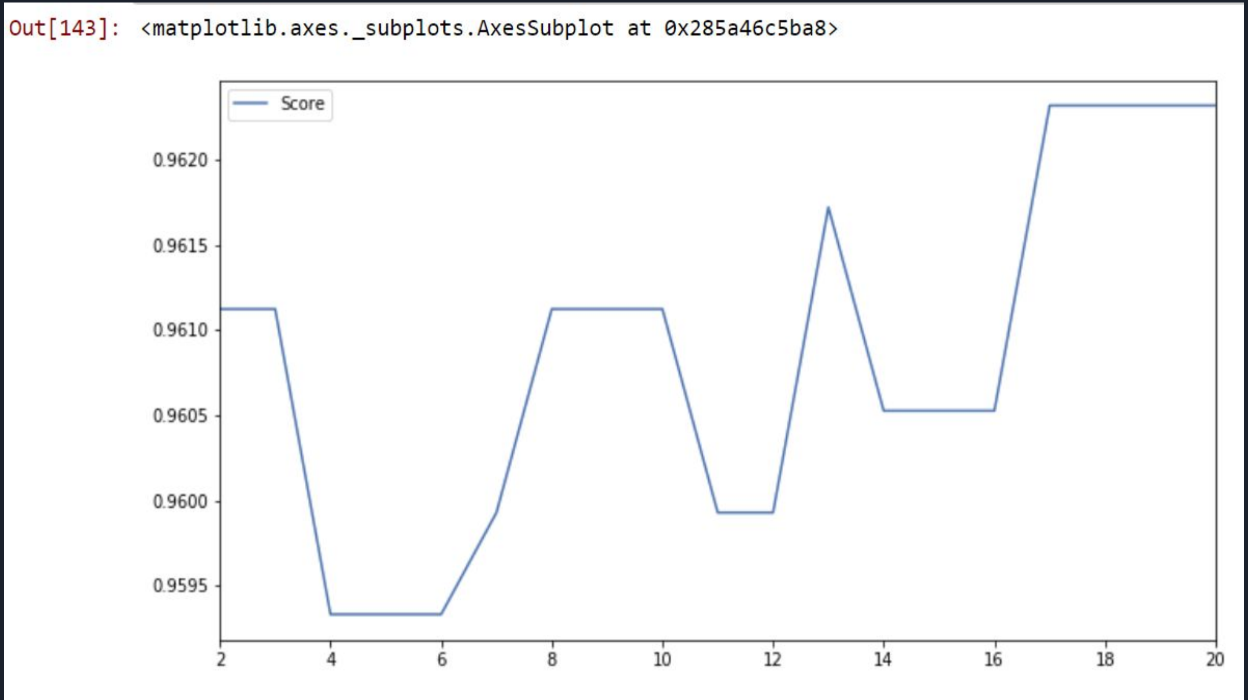
Highest accuracy at alpha = 0.2

	Score
0.2	0.985646

Alpha = Additive (Laplace/Lidstone) smoothing parameter

Decision Tree classifier

- Uses a decision tree for classification



Highest accuracy at min_samples_split = 17

	Score
17	0.962321
18	0.962321
19	0.962321
20	0.962321

The minimum number of samples required to split an internal node

k = 49
alpha = 0.2
min_samples_split = 17

Now let's train our classifiers.

Classification Part 2: Classifier Results

k-neighbours classifier

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=49, p=2,
           weights='uniform')
             precision    recall  f1-score   support

        ham       0.92      1.00      0.96      1440
       spam       0.99      0.47      0.64       232

avg / total       0.93      0.93      0.91      1672
```

Classification Report

```
[[1439    1]
 [ 123  109]]
```

Confusion Matrix

Multinomial Naive Bayes classifier

```
MultinomialNB(alpha=0.2, class_prior=None, fit_prior=True)
             precision    recall  f1-score   support

        ham       0.99      0.99      0.99      1440
       spam       0.96      0.94      0.95       232

avg / total       0.99      0.99      0.99      1672
```

Classification Report

```
[[1430   10]
 [  14  218]]
```

Confusion Matrix

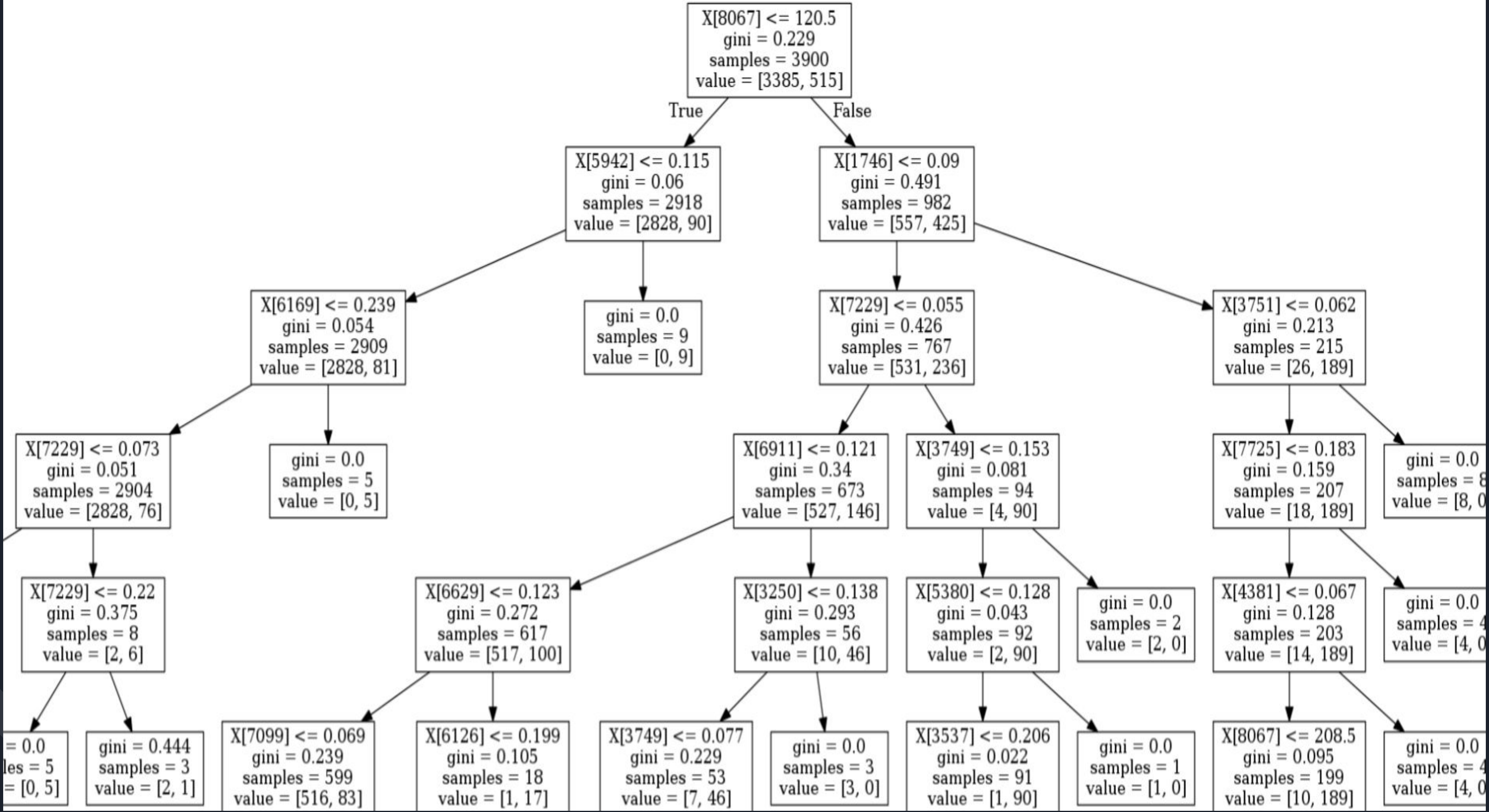
Decision Tree classifier

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=17,
            min_weight_fraction_leaf=0.0, presort=False, random_state=111,
            splitter='best')
             precision    recall  f1-score   support

        ham       0.97      0.99      0.98      1440
       spam       0.90      0.82      0.86       232

avg / total       0.96      0.96      0.96      1672
```

Classification Report

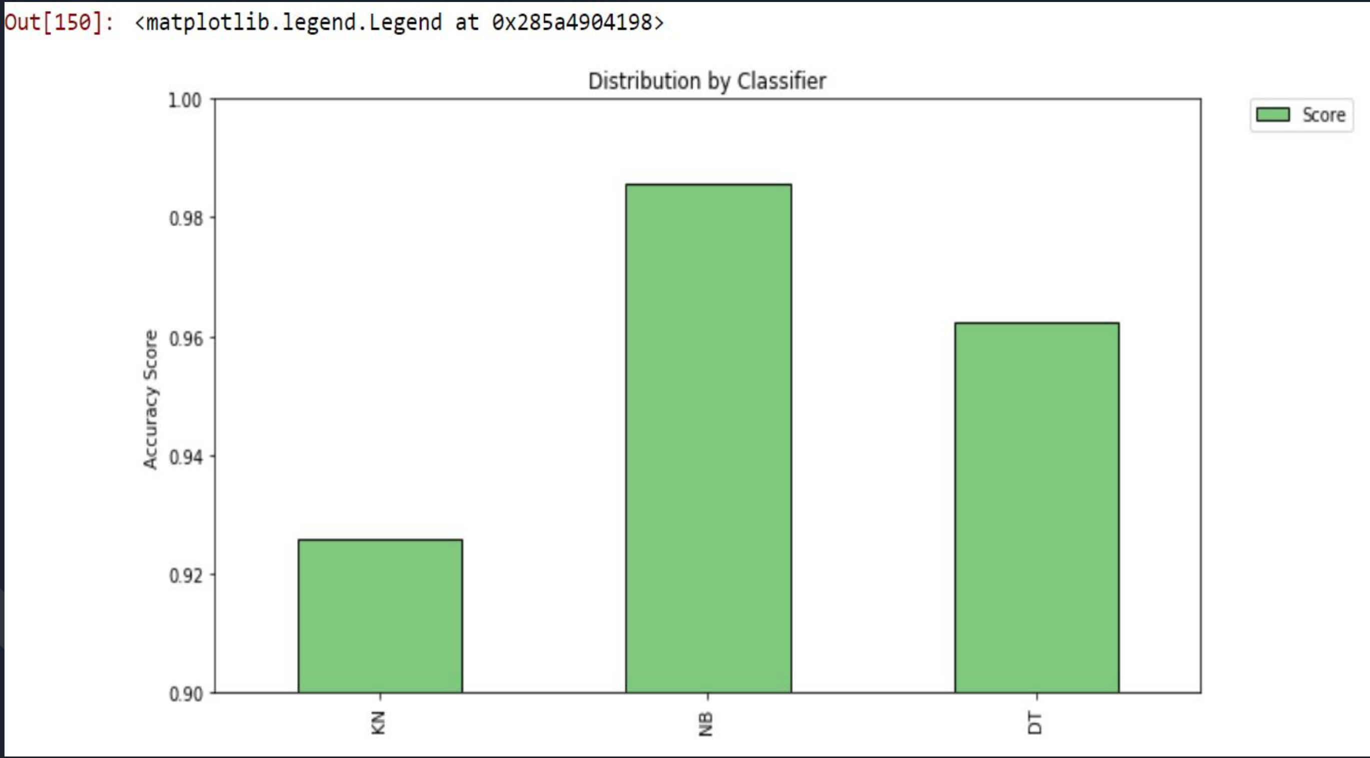
```
[[1419   21]
 [  42  190]]
```

Confusion Matrix



X[8067] <= 120.5
gini = 0.229
samples = 3900
value = [3385, 515]
True
False
X[5942] <= 0.115
gini = 0.06
samples = 2918
value = [2828, 90]
X[1746] <= 0.09
gini = 0.491
samples = 982
value = [557, 425]
X[6169] <= 0.239
gini = 0.054
samples = 2909
value = [2828, 81]
gini = 0.0
samples = 9
value = [0, 9]
X[7229] <= 0.055
gini = 0.426
samples = 767
value = [531, 236]
X[3751] <= 0.062
gini = 0.213
samples = 215
value = [26, 189]
X[7229] <= 0.073
gini = 0.051
samples = 2904
value = [2828, 76]
gini = 0.0
samples = 5
value = [0, 5]
X[6911] <= 0.121
gini = 0.34
samples = 673
value = [527, 146]
X[3749] <= 0.153
gini = 0.081
samples = 94
value = [4, 90]
X[7725] <= 0.183
gini = 0.159
samples = 207
value = [18, 189]
gini = 0.0
X[7229] <= 0.22
gini = 0.375
samples = 8
value = [2, 6]
X[6629] <= 0.123
gini = 0.272
samples = 617
value = [517, 100]
X[3250] <= 0.138
gini = 0.293
samples = 56
value = [10, 46]
X[5380] <= 0.128
gini = 0.043
samples = 92
value = [2, 90]
gini = 0.0
samples = 2
value = [2, 0]
X[4381] <= 0.067
gini = 0.128
samples = 203
value = [14, 189]
gini = 0.0
= 0.0
gini = 0.444
samples = 3
value = [2, 1]
X[7099] <= 0.069
gini = 0.239
samples = 599
value = [516, 83]
X[6126] <= 0.199
gini = 0.105
samples = 18
value = [1, 17]
X[3749] <= 0.077
gini = 0.229
samples = 53
value = [7, 46]
gini = 0.0
samples = 3
value = [3, 0]
X[3537] <= 0.206
gini = 0.022
samples = 91
value = [1, 90]
gini = 0.0
samples = 1
value = [1, 0]
X[8067] <= 208.5
gini = 0.095
samples = 199
value = [10, 189]
gini = 0.0

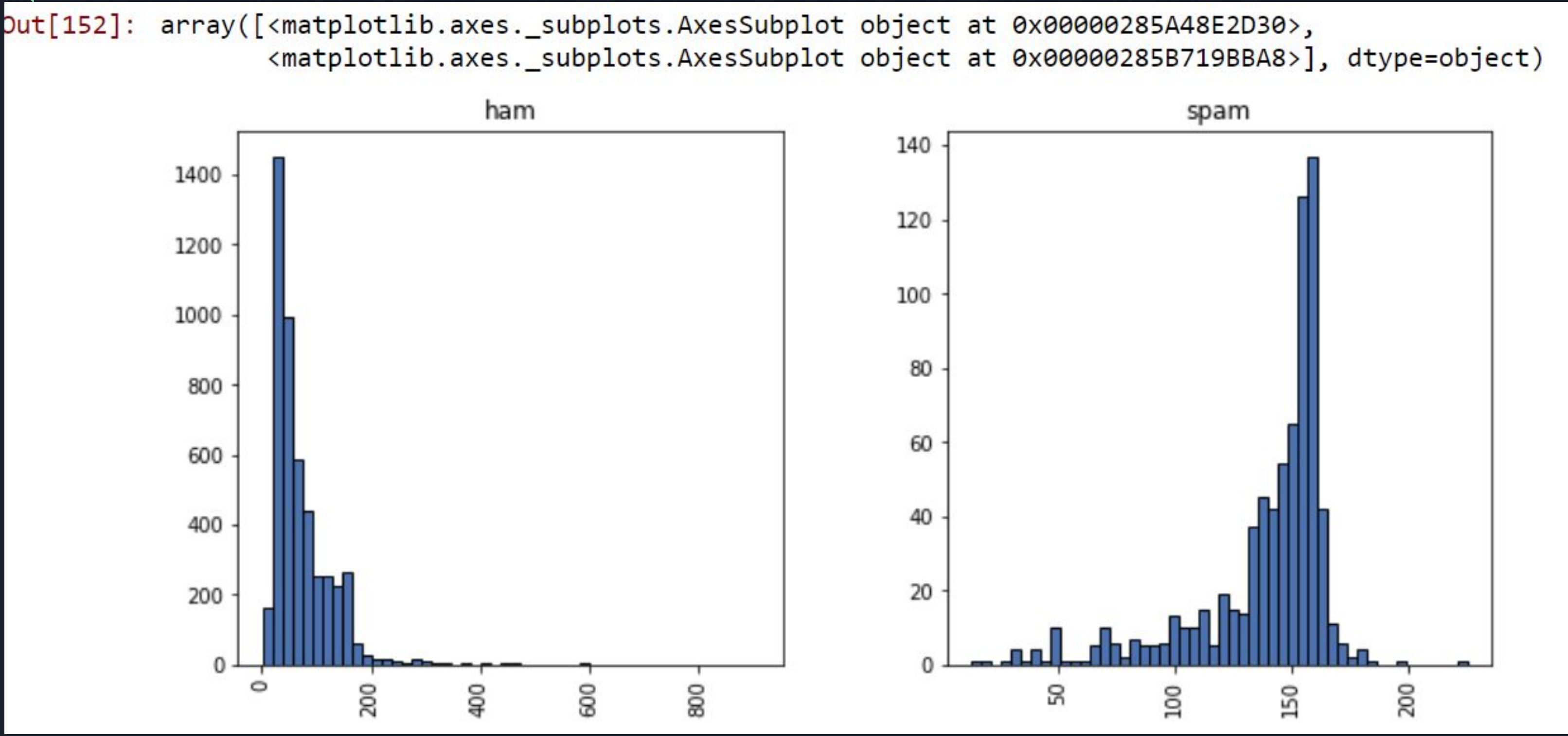
The Winner: Multinomial Naive Bayes



	Score
KN	0.925837
NB	0.985646
DT	0.962321

But… What about message length?

Lengthier message = likely a spam



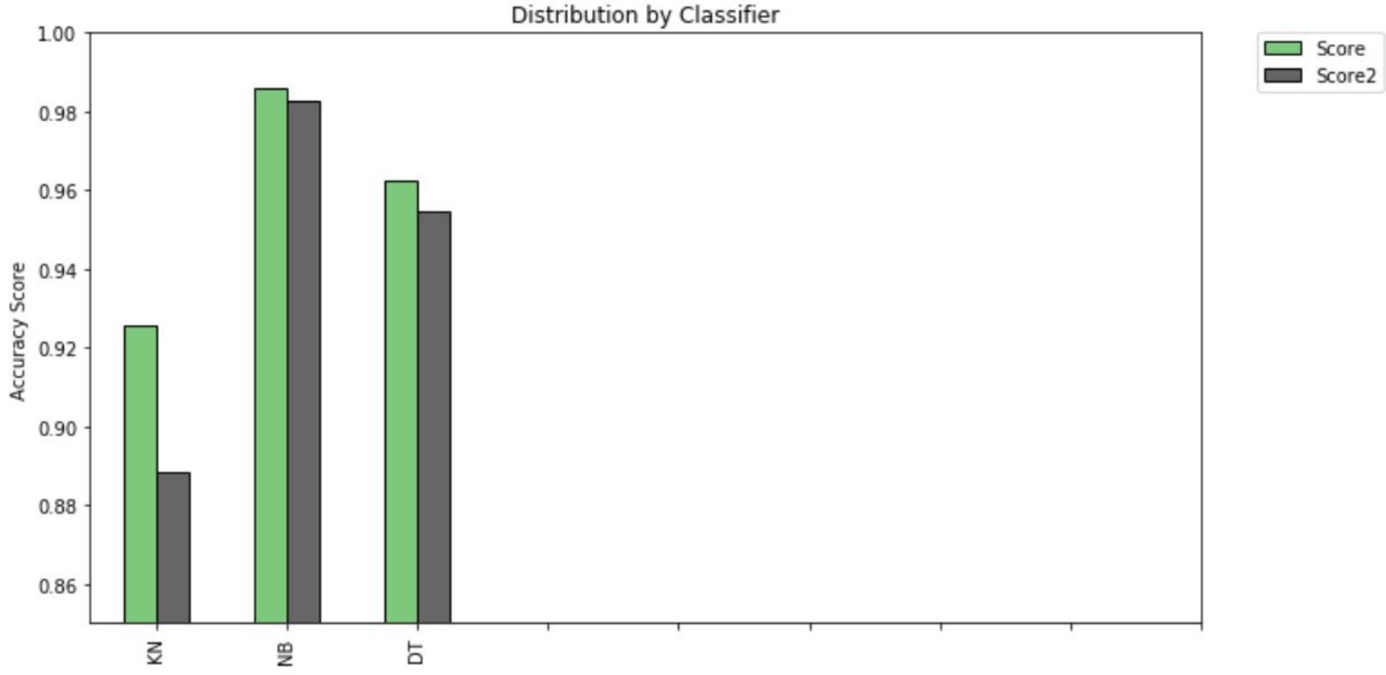
Add message length

Out[151]:

	class	message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

However, things get a bit worse.

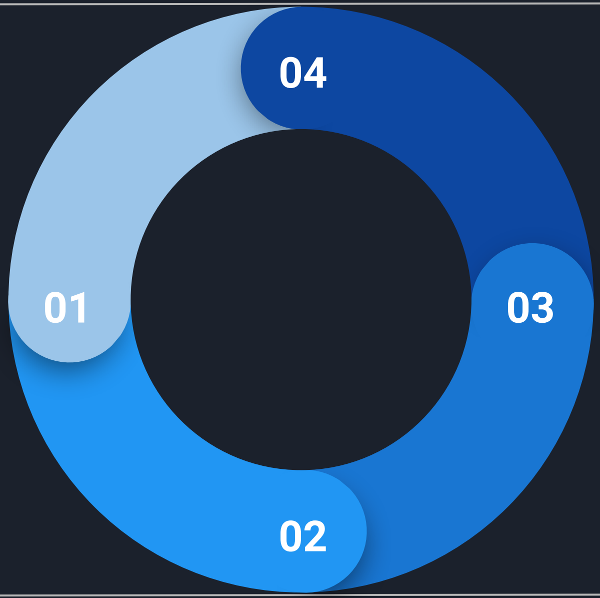
Out[157]: <matplotlib.legend.Legend at 0x28595738cc0>



Cycle diagram

Preprocess

Classify

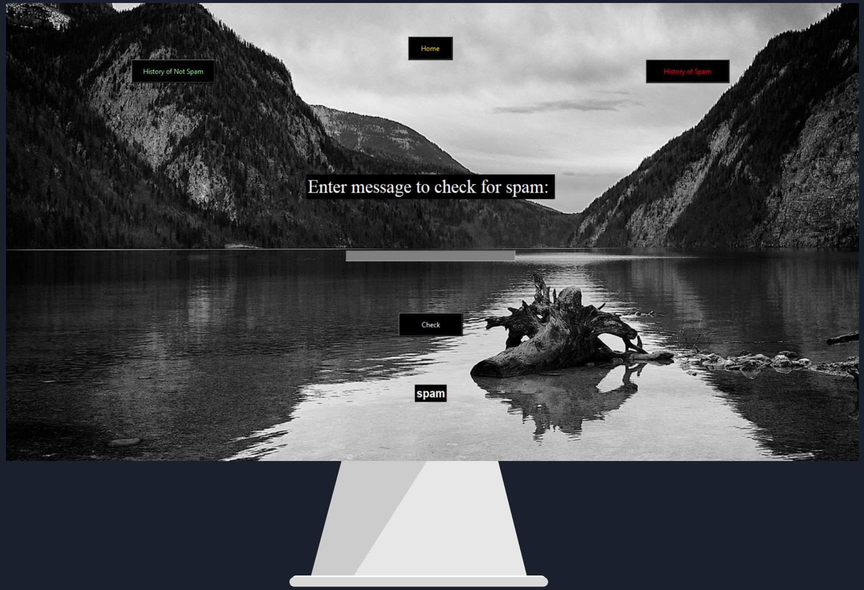


Tune

Train

Introducing: SpamShield

A GUI based application for classifying SMS



Thank you!