Report - Text Classifier.

Problem overview.

Text classification is the process of assigning tags or categories to text according to its content. It's one of the fundamental tasks in Natural Language Processing (NLP) with broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection.

Algorithm explanation.

For text classification i will use Naive Bayes Classifier.
First step is **pre processing data**. I convert data into pandas DataFrame, where first column 'message' contains documents and column 'labels' - labels of documents.

I **split data** into train and test with proportion 80/20 with class balance. After splitting all classes have equal representation in test dataset.

Model Naive Bayes Classifier have two methods:

- NB_fit, fit method
- NB_predict, predict method

NB_fit method, use train dataset and calculates:

- shape of classes
- prior probability of classes, ***shape_of_class / shape_of_dataset***
- *l*en of vocabulary(unic words from all classes), ***V***
- word frequencies in each class

NB_predict method, calculates probability and defines classes for documents from test dataset. I use Bayes rule for calculation class probability. ***P(class|text) = P(text|class) * P(class)***

Naive Bayes makes two assumptions :

- bag of words (assume position doesn't matter)
- conditional independence (assume the feature probabilities P(word_i | class) are independent given the class), ***P(text | class) = P(word_1 | class) * P(word_2 | class) * ...* P(word_n | class)***

Problem #1 with calculation P(word_i | class). If we don't see word in training documents, we will have zero conditional probability ***P(text | class)***.
We can solve this problem and use Laplace smoothing (add-1):
P(word_i | class) = (count(word_j, class) +1) / (all_words_of_class + |Vocabulary|)

Problem #2 with calculating *P(class|text)*. If we have a lot of words in document, we will have zero value for *P(class|text),* because Python uses double-precision floats, which can hold values from about 10 to the -308 to 10 to the 308 power.
We can use natural logarithm trick and change formula for *P(class|text)* into:
log(P(text|class))=log(P(word_1|class))+log(P(word_2|class))+...+log(P(word_n | class))+ log(P(class))

Final step, we compare probabilities for each class and chose class with biggest probability (or logarithm of probability) .

Results of NB text classifier

Simple example from presentation.

Input data:
train=['Chinese Beijing Chinese', 'Chinese Chinese Shanghai', 'Chinese Macao', 'Tokyo Japan Chinese']
labels=[0,0,0,1]
test=['Chinese Chinese Chinese Tokyo Japan',]

Output data:

```
The number of words = 11
The number unique words = 6
class probabilities =[0.75,0.25], N_classes=[8,3], Vocabulary=6
```

```
Chinese: p_wc_0=0.4286, p_wc_1=0.22
Tokyo:  p_wc_0=0.0714, p_wc_1=0.22
Japan:  p_wc_0=0.0714, p_wc_1=0.22
```

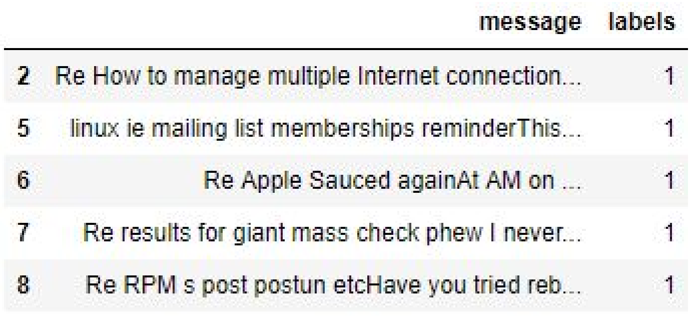
```
P_0 = p(Chinese|0) * p(Chinese|0) * p(Chinese|0) * p(Tokyo|0) * p(Japan|0) = 0.0003
P_1 = p(Chinese|1) * p(Chinese|1) * p(Chinese|1) * p(Tokyo|1) * p(Japan|1) = 0.0001
```

```
predict class: [0], because P_0 > P_1
```

Email spam filter dataset.

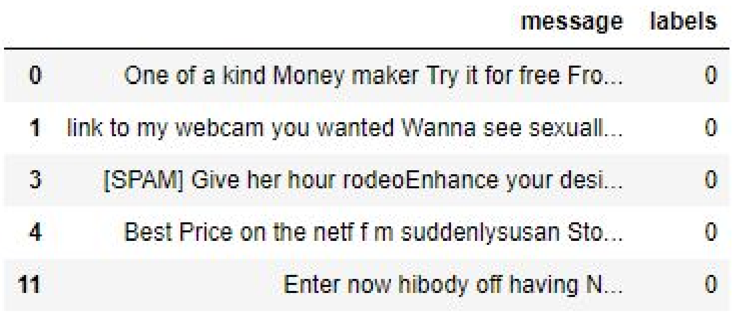
Dataset have 1118 emails. 380 emails are spam and 738 emails are ham class

Ham samples:



	message	labels
2	Re How to manage multiple Internet connection...	1
5	linux ie mailing list memberships reminderThis...	1
6	Re Apple Sauced againAt AM on ...	1
7	Re results for giant mass check phew I never...	1
8	Re RPM s post postun etcHave you tried reb...	1

Spam samples:



	message	labels
0	One of a kind Money maker Try it for free Fro...	0
1	link to my webcam you wanted Wanna see sexuall...	0
3	[SPAM] Give her hour rodeoEnhance your desi...	0
4	Best Price on the netf f m suddenlysusan Sto...	0
11	Enter now hibody off having N...	0

```
Shape dataset after splitting: train shape (896, 2), test shape (222, 2)
```

```
Result of text class predictions (Naive Bayes text classifier):
The number of words = 226343
The number unique words = 28958
class probabilities=[spam - 0.3,ham - 0.7]
N_classes=[80126,146217], Vocabulary=28958
```

Accuracy 88.3. I think it's good enough result for NB.

TF/IDF algorithm.
Term frequency–inverse document frequency is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word.

Algorithm explanation.

1) For each word in document calculate word frequency in this document, *freq_w = count(w|doc) / len(doc)*
2) For each word calculate inverse document frequency, *IDF = log(all_docs_in_class / count (doc|w))*
3) Probability of doc = (f*req_w1*IDF_w1) * (*freq_w2*IDF_w2) * … * (*freq_wn*IDF_wn) * prior(class)*
4) If we have a lot of words in document we can use,
log(Probability of doc) =sum(log((f*req_wn*IDF_wn))) + prior(class), for n=1,..., len(Class vocabulary)*
5) Compare probabilities for each class and chose class with biggest probability (or logarithm of probability)

Results of NB + TF-IDF text classifier

Email spam filter dataset
```
Accuracy for text classification with TF-IDF = 90.54
```

Total results

Table of accuracy for different text classifier

NB (bag of words)	NB (TF-IDF)
88.3	90.54

Vasyl Shandyba
23.04.2019

link to jupyter notebook on google colab:
https://drive.google.com/file/d/1hsok5L0oFFXLxptaTF_fYAmWdicjA32c/view?usp=sharing