Naive Bayes Classifier

A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification task.Below you can see a formula for the calculation:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of A happening, given that B has occurred.

In various applications such as spam filtering, text classification, sentiment analysis, and recommendation systems, Naive Bayes classifier is used successfully. It reccomend itself as fastest classification algorithm for a large chunk of data.

We There are 3 types of Naïve Bayes algorithm:
1. Gaussian Naïve Bayes:use when dealing with continuous data.
2. Multinomial Naïve Bayes:mostly used for document classification problem(whether a document belongs to the some category). The features/predictors used by the classifier are the frequency of the words present in the document.
3. Bernoulli Naïve Bayes:similar to the multinomial naive bayes but the predictors are boolean variables.

Advantages:
1. Works quickly.
2. Suitable for solving multi-class prediction problems.
3. If its assumption of the independence of features holds true, it can perform better than other models and requires much less training data.
4. Naive Bayes is better suited for categorical input variables than numerical variables.

Disadvantages:
1. Asumes that all predictors (or features) are independent, rarely happening in real life.
2. Suitable for solving multi-class prediction problems.
3. Faces the ‘zero-frequency problem’
4. Estimations can be wrong in some cases.

sklearn.naive_bayes

In this package we have several types of naive bayes classifier:

1. Gaussian Naive Bayes-implements the Gaussian Naive Bayes algorithm for classification. Syntax for import:from sklearn.naive_bayes import GaussianNB.
2. Multinomial Naive Bayes-implements the naive Bayes algorithm for multinomially distributed data, very suitable for text classification, where data represented as word vector counts/tf-idf vectors.
3. Complement Naive Bayes-implements the complement naive Bayes (CNB) algorithm, you can say that it's an adaptation of standard multinomial naive Bayes algorithm ,suitable for imbalanced data sets.
4. Bernoulli Naive Bayes-implements the naive Bayes training and classification algorithms,suitable for data that is distributed according to Multivariate Bernoulli distributions. Requires data to be represented as binary-valued feature vectors.Also particularly useful when a feature can be present or not.
5. Categorical Naive Bayes-implements the categorical naive Bayes algorithm for categorically distributed data.

NLTK

In this package we have python class that represents naive bayes classifier.Paramaterized by two probability distributions:P(label) gives the probability that an input will receive each label, given no information about the input's features and P(fname=fval|label) gives the probability that a given feature (fname) will receive a given value (fval), given that the label (label).

- How to import:from nltk import NaiveBayesClassifier
- So becouse it's a python class we have here a constractor and several methods:
    - First of all we need to train classifier with data: classifier = nltk.NaiveBayesClassifier.train(training_set)
    - After traiing you can test it:nltk.classify.accuracy(classifier, testing_set))
    - labels()-return list of category labels used by this classifier.
    - classify(featureset)-return the most appropriate label for the given featureset.
    - prob_classify(featureset)-return a probability distribution over labels for the given featureset.
    - If we want we can analyze the top feature words that were used for classification using the show_most_informative_features(n-10).
    - most_informative_features(n=100)-return a list of the 'most informative' features used by this classifier.

NLTK with sklearn wrapper (nltk.classify.scikitlearn module)

To use this wrapper:

```
>>> from sklearn.svm import LinearSVC
>>> from nltk.classify.scikitlearn import SklearnClassifier
>>> classif = SklearnClassifier(LinearSVC())
```

It's also represented as python class that has:

1. Constractor
2. classify_many(featuresets)-Classify a batch of samples. Return predicted class label for each input sample.
3. prob_classify_many(featuresets)-Compute per-class probabilities for a batch of samples.
4. labels()-return list of category labels used by this classifier.
5. train(labeled_featuresets)-Train the scikit-learn estimator.

Textblob.classifiers

Textblob provides build-in classifiers module to create a custom classifier and pass training data into the classifier:

```
from textblob import classifiers
classifier = classifiers.NaiveBayesClassifier(training)
```

It's also represented as python class that has:

1. show_informative_features() - display a listing of the most informative features.
2. prob_classify_many(text)-Compute per-get the label probability distribution.
3. Another way to classify text is to pass a classifier into the constructor of TextBlob and call its classify() method.
4. To compute the accuracy :classufuer.accuracy(data)
5. update(new_data) -update a classifier with new training data.

TextBlob also offers Decision tree classifier

```
## decision tree classifier
dt_classifier = classifiers.DecisionTreeClassifier(training)
```

Accepts two arguments:

- Training data
- Format: If we pass a file inside the train_ set parameter, then format contains the format of the file like "csv" or "json". If None value is passed, it tries to detect the file format itself.

Has the sane nethods as classifier discussed earlier.

Feature Selection

- Definition:It's the process of isolating the most consistent, non-redundant, and relevant features to use in model construction.
- In text classificationit can improve the scalability, efficiency and accuracy of a text classifier.
- For feature selection we can use- sklearn.feature_selection module.
- Filter Methods-rely on the features' characteristics without using any machine learning algorithm,always used at the beginning of the selection process to get rid of the irrelevant, duplicated, correlated, and constant features:
    1. Advantages:selected features can be used in any machine learning algorithm and you can process thousands of features in a matter of seconds.
    2. The mehtods:Basic Filter Methods,Correlation Filter Methods,Statistical and Ranking Filter Methods.
- Wrapper methods-usually result in better predictive accuracy than filter methods.:
    1. Advantages:detect the interaction between variables and find the optimal feature subset for the desired machine learning algorithm
    2. The process:Search for a subset of features->Build a machine learning model->Evaluate model performance->Repeat
    3. Stopping Criteria:need to be defined by the developer.
- Embedded methods:
    1. Advantages:take into consideration the interaction of features like wrapper methods do,faster,more accurate than filter methods.
    2. The process:train a model->derive feature importance from this model->remove non-important features using the derived feature importance.
- We need use feature selection to text classification because features consume a lot of time and computing power to get the result. Particular in case of text where features are words.Using it helps a lot to Naive Bayes classifier because he is exspensive to train.
- Feature selection is necessary for multivariate Bernoulli NB for avoiding noise and multi-counting.

Evaluating a text-classified model

Cross-validation is a common method to evaluate the performance of a text classifier. It works by splitting the training dataset into random, equal-length example sets. For each set, a text classifier is trained with the remaining samples. After that classifiers make predictions on their respective sets and the results are compared against the human-annotated tags. This will determine when a prediction was right and when not.

With results that we get we can build performance metrics for assessment on how well our classifier works:accuracy,precision, recall and harmonic mean of precision and recall.

In [135]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_20newsgroups
from sklearn import metrics
import time
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
import pandas as pd
#-----------------------------------------------------------------------------------------
chosen_categories = ['sci.med','sci.space']
#-----------------------------------------------------------------------------------------
train = fetch_20newsgroups(subset='train',categories=chosen_categories)
vectorizer = TfidfVectorizer()
count=CountVectorizer()
#-----------------------------------------------------------------------------------------
df = pd.DataFrame(train.data,columns=['Train data'])
```

In [136]:

```
df.head(10)
```

Out[136]:

	Train data
0	From: flb@flb.optiplan.fi ("F.Baube[tm]")\nSub...
1	From: geb@cs.pitt.edu (Gordon Banks)\nSubject:...
2	From: ab961@Freenet.carleton.ca (Robert Alliso...
3	From: rind@enterprise.bih.harvard.edu (David R...
4	From: nsmca@aurora.alaska.edu\nSubject: Space ...
5	From: uabdpo.dpo.uab.edu!gila005 (Stephen Holl...
6	From: c23st@kocrsv01.delcoelect.com (Spiros Tr...
7	From: pgf@srl03.cacs.usl.edu (Phil G. Fraering...
8	From: nsmca@aurora.alaska.edu\nSubject: Re: Wh...
9	From: ralph.buttigieg@f635.n713.z3.fido.zeta.o...

Building Dictionary

In [137]:

```python
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
# Import clean tweets
stem = PorterStemmer()
temp = df_train['Train data']
nan_value = float("NaN")
temp.replace("", nan_value, inplace=True)
original = [] # Used by boolean find functions
pattern = r'[)(@<>+=#$%^:]'

tokens_without_sw = [word for word in temp if not word in stopwords.words()]
# Tokenize and calculate freq
text_dict = {}
for t in tokens_without_sw:
    new_temp = re.sub(pattern, '', t)
    original.append(new_temp)
    token = new_temp.split()
    tPart2.append([stem.stem(t) for t in token])
    for word in token:
        if word in text_dict:
            text_dict[word] += 1;
        else:
            text_dict[word] = 1;
print(text_dict)
```

```
{'From': 643, 'nyedacnsvax.uwec.edu': 18, 'David': 111, 'Nye': 28, 'S
ubject': 599, 'Re': 455, 'Post': 6, 'Polio': 4, 'Syndrome': 25, 'Info
rmation': 26, 'Needed': 6, 'Please': 34, '!!!': 5, 'Organization': 58
9, 'University': 264, 'of': 4456, 'Wisconsin': 23, 'Eau': 22, 'Clair
e': 22, 'Lines': 589, '21': 29, '[reply': 12, 'to': 3820, 'keithactri
x.gen.nz': 3, 'Keith': 13, 'Stewart]': 1, 'My': 174, 'wife': 35, 'ha
s': 466, 'become': 38, 'interested': 39, 'through': 82, 'an': 537, 'a
cquaintance': 4, 'in': 2333, 'Post-Polio': 3, 'This': 259, 'apparentl
y': 32, 'is': 2571, 'not': 949, 'recognised': 3, 'New': 94, 'Zealan
d': 8, 'and': 3298, 'different': 73, 'symptons': 4, 'eg': 4, 'chest':
10, 'complaints': 10, 'are': 1074, 'treated': 22, 'separately.': 3,
'Does': 52, 'anone': 3, 'have': 909, 'any': 387, 'information': 118,
'on': 794, 'it': 1327, 'It': 243, 'would': 406, 'help': 98, 'if': 38
0, 'you': 860, 'anyone': 135, 'else': 43, 'asking': 8, 'for': 1332,
'medical': 141, 'some': 379, 'subject': 27, 'could': 140, 'ask': 30,
'specific': 25, 'questions,': 9, 'as': 773, 'no': 275, 'one': 397, 'l
ikely': 42, 'type': 49, 'a': 3188, 'textbook': 1, 'chapter': 1, 'cove
ring': 5, 'all': 301, 'aspects': 7, 'the': 6020, 'subject.': 5, 'If':
271, 'looking': 29, 'comprehensive': 8, 'review,': 2, 'your': 407, 'l
egal': 25, 'hospital': 21, 'librarian,': 1, 'Most': 21, 'happy': 6,
```

In [138]:

```
#Build table from dictinary and create frequncy column
dict_table = pd.DataFrame(text_dict.items(),columns=['Word','Frequency'])
dict_table.head(10)
```

Out[138]:

	Word	Frequency
0	From	643
1	nyedacnsvax.uwec.edu	18
2	David	111
3	Nye	28
4	Subject	599
5	Re	455
6	Post	6
7	Polio	4
8	Syndrome	25
9	Information	26

In [139]:

```
#Vectors
from pandas import Series
import nltk
start = time.time()
Frequency=dict_table['Word']
X = []

for s in original:
    vector = []
    help=nltk.word_tokenize(s)
    for word in Frequency:
        if word in help:
            vector.append(1)
        else:
            vector.append(0)
    X.append(vector)
end = time.time()

print(f"Matrix Build time: {end-start}")
```

```
Matrix Build time: 53.796709299087524
```

BOW - count vectors ייצוג של

In [141]:

```
sentence_vectors = np.asarray(X)
print(sentence_vectors)
```

```
[[1 1 1 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 1 0 1]]
```

ייצוג TFIDF

In [143]:

```
start1 = time.time()
data = vectorizer.fit_transform(original)
end1 = time.time()
print(f"TF-IDF Matrix Build time: {end1-start1}")
print()
print(data)
```

```
TF-IDF Matrix Build time: 0.31043195724487305

  (0, 13008)	0.06769733078662438
  (0, 2819)	0.08529284315244871
  (0, 1570)	0.08529284315244871
  (0, 3269)	0.03347822554907115
  (0, 6655)	0.08529284315244871
  (0, 2702)	0.025192906165226958
  (0, 8924)	0.07252638204395752
  (0, 10145)	0.05437073697140048
  (0, 11360)	0.08390060196681462
  (0, 16111)	0.08141154463721859
  (0, 16053)	0.08390060196681462
  (0, 3255)	0.026723011184219807
  (0, 1569)	0.08141154463721859
  (0, 11144)	0.08141154463721859
  (0, 16061)	0.08390060196681462
  (0, 3885)	0.07252638204395752
  (0, 9801)	0.08529284315244871
```

Implementation of Multinimial and Barnauli Naive Bayes Classifier

In [153]:

```
train1 = fetch_20newsgroups(subset='train',categories=['sci.med'])
train2 = fetch_20newsgroups(subset='train',categories=['sci.space'])
```

In [178]:

```
pattern = r'[)(@<>+=#$%^:?!-*]'
clean_med = []
clean_space=[]

for i in train1.data:
     new_temp = re.sub(pattern, '', i)
     token = new_temp.split()
     clean_med.append(token)

for i in train2.data:
     new_temp = re.sub(pattern, '', i)
     token = new_temp.split()
     clean_space.append(token)
```

Multinimial Naive Bayes Classifier

In [234]:

```
from nltk import NaiveBayesClassifier
import random
all_words=[]

documents1=[(w, 'med') for w in clean_med]
documents2=[(w, 'space') for w in clean_med]

documents=documents1+ documents2
random.shuffle(documents)

for w in dict_table['Word']:
    all_words.append(w.lower())

word_features=list(nltk.FreqDist(all_words).keys())[:1200]
features = {}
```

In [235]:

```
def find_features(document):
    words = set(document)
    for w in word_features:
        features[w] = (w in words)
    return features

featureset=[(find_features(rev),category) for(rev,category) in documents]

# set that we'll train our classifier with
training_set = featureset[:300]
# set that we'll test against.
testing_set = featureset[301:1181]

start = time.time()
classifier = nltk.NaiveBayesClassifier.train(training_set)
answer=nltk.classify.accuracy(classifier, testing_set)*100
end = time.time()

print(f"\nSize of test data:{len(training_set)}")
print(f"\nSize of train data:{len(testing_set)}")
print("\nTime: ",end-start)
print(f"\nMultinomialNB accuracy percent is {answer} %")
```

```

Size of test data:300

Size of train data:880

Time:  1.7991461753845215

MultinomialNB accuracy percent is 49.31818181818181 %
```

In [236]:

```
classifier.show_most_informative_features(15)
```

```
Most Informative Features
                       ! = False             med : space  =      1.0 :
1.0
                     !!! = False             med : space  =      1.0 :
1.0
                      "a = False             med : space  =      1.0 :
1.0
                 "acute = False             med : space  =      1.0 :
1.0
                   "corn = False             med : space  =      1.0 :
1.0
           "experimental = False             med : space  =      1.0 :
1.0
              "hybrids", = False             med : space  =      1.0 :
1.0
               "mistake" = False             med : space  =      1.0 :
1.0
              "mistake"? = False             med : space  =      1.0 :
1.0
             "skepticism = False             med : space  =      1.0 :
1.0
                   'heat = False             med : space  =      1.0 :
1.0
                       * = False             med : space  =      1.0 :
1.0
                      ** = False             med : space  =      1.0 :
1.0
                     *** = False             med : space  =      1.0 :
1.0
                   ***** = False             med : space  =      1.0 :
1.0
```

Barnauli Naive Bayes Classifier

In [237]:

```
from nltk.classify.scikitlearn import SklearnClassifier
from sklearn.naive_bayes import BernoulliNB

start = time.time()
BNB_classifier = SklearnClassifier(BernoulliNB())
BNB_classifier.train(training_set)
answer=nltk.classify.accuracy(classifier, testing_set)*100
end = time.time()

print(f"\nSize of test data:{len(training_set)}")
print(f"\nSize of train data:{len(testing_set)}")
print("\nTime: ",end-start)
print(f"\nMBernoulliNB accuracy percent is {answer} %")
```

```
Size of test data:300

Size of train data:880

Time:  1.8999419212341309

MBernoulliNB accuracy percent is 49.31818181818181 %
```

Analysis of Results

We saw that the MNB works faster(1.7991461753845215 sec) than BNB(1.8999419212341309 sec). Maybe we have a difference in tun time because of nature of the activity is different.Bernoulli models the presence/absence of a feature. Multinomial models the number of counts of a feature.This means that, Multinomial NB will classify a document based on the counts it finds of multiple keywords.Bernoulli NB can only focus on a single keyword, but will also count how many times that keyword does not occur in the document.

	Multinimial Naive Bayes Classifier	**Barnauli Naive Bayes Classifier**
Run Time	1.7991461753845215 seconds	1.8999419212341309 seconds
Train Size	300	880
Test Size	300	880
Accuracy percent	49.31818181818181 %	49.31818181818181

Summary

- So we saw that Naive Bayes is very good with text classification and we have several options to chose for what is best for us.
- So why used him? Answer:Naive Bayes predict the tag of a text. They calculate the probability of each tag for a given text and then output the tag with the highest one.
- Why is Naive Bayes better than logistic regression for text classification? Answer: In short Naive Bayes has a higher bias but lower variance compared to logistic regression.
- From this lab we earned that as we said earlier we can choose different technics for calculation, it pretty fast. This Lab come to help us to improve our learning in pracice and it's what always is best.