ML Lab Program: 6

- Assuming a set of documents that need to be classified, use the **naïve Bayesian Classifier** model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the **accuracy, precision, and recall** for your data set.

Naïve Bayes Classifier

- Naive Bayes is a statistical classification technique based on Bayes Theorem.
- It is one of the simplest supervised learning algorithms.
- Naive Bayes classifier is the fast, accurate and reliable algorithm.
- Naive Bayes classifier calculates the probability of an event in the following steps:

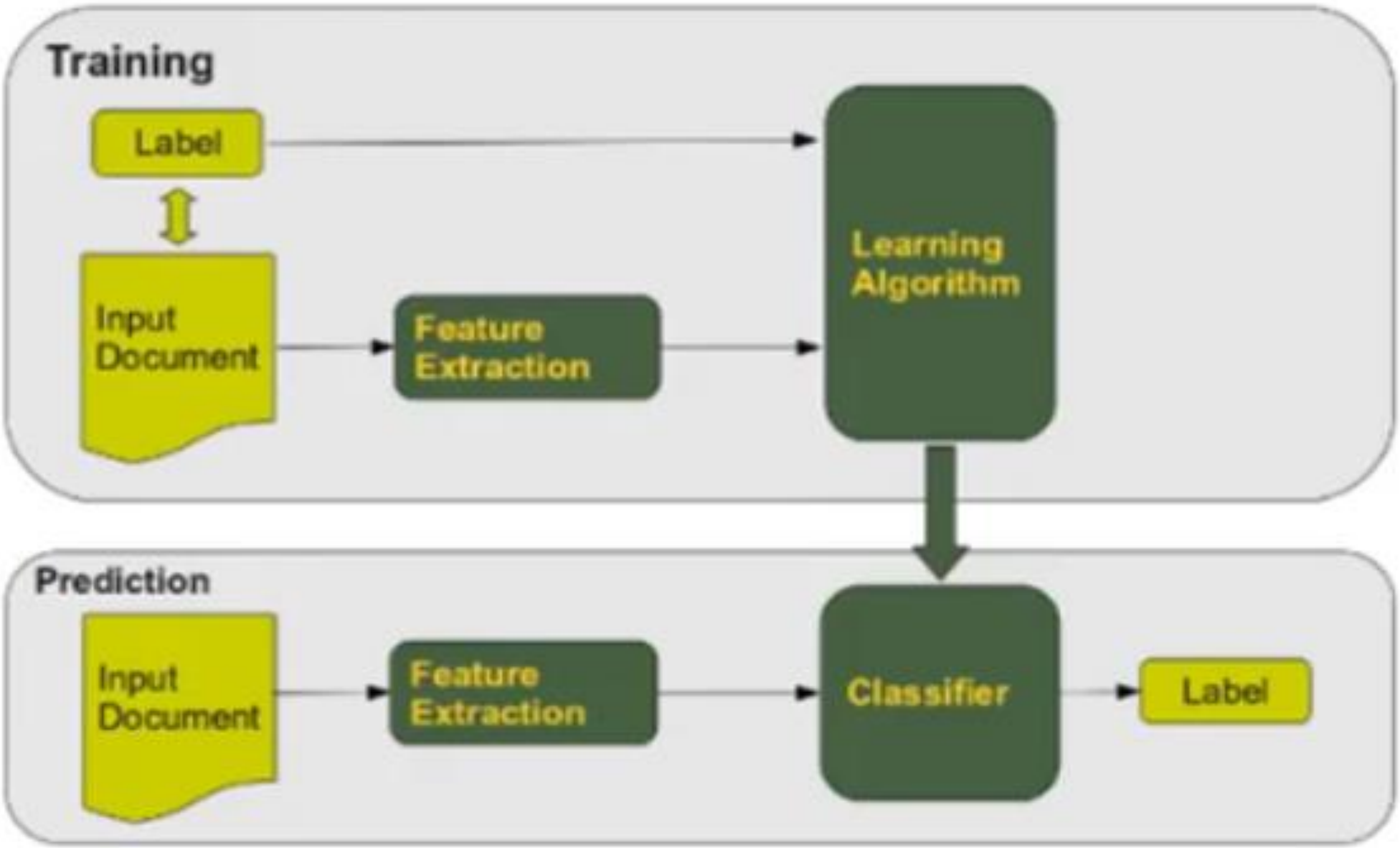
Step 1: Calculate the prior probability for given class labels

Step 2: Find Likelihood probability with each attribute for each class

Step 3: Put these value in Bayes Formula and calculate posterior probability.

Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

What is text classification ?



Source: https://www.python-course.eu/text_classification_introduction.php

Evaluation Metrics

These metrics are used to evaluate the results of classifications:

- Accuracy
- Precision
- Recall

Evaluation Metrics

Accuracy

Accuracy is a statistical measure which is defined as the quotient of correct predictions (both True positives (TP) and True negatives (TN)) made by a classifier divided by the sum of all predictions made by the classifier, including False positives (FP) and False negatives (FN).

The formula:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Confusion Matrix

A confusion matrix, also called a contingency table or error matrix, is used to visualize the performance of a classifier.

Confusion Matrix		Predicted classes	
		negative	positive
Actual classes	negative	TN	FP
	positive	FN	TP

Evaluation Metrics

- **Precision:** Precision is the ratio of the correctly identified positive cases to all the predicted positive cases, i.e. the correctly and the incorrectly predicted cases as positive. Precision is the fraction of retrieved documents that are relevant to the query.
- The formula:

$$precision = \frac{TP}{TP + FP}$$

Evaluation Metrics

- **Recall**, also known as sensitivity, is the ratio of the correctly identified positive cases to all the actual positive cases, which is the sum of the "False Negatives" and "True Positives".

- The formula:

$$recall = \frac{TP}{TP + FN}$$

Naïve Bayes classifier algorithm for text classification

LEARN_NAIVE_BAYES_TEXT(*Examples*, *V*)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. collect all words, punctuation, and other tokens that occur in Examples

- *Vocabulary* ← the set of all distinct words and other tokens occurring in any text document from *Examples*

2. calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms

- For each target value v_j in V do
  - $docs_j$ ← the subset of documents from *Examples* for which the target value is v_j
  - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
  - $Text_j$ ← a single document created by concatenating all members of $docs_j$
  - n ← total number of distinct word positions in $Text_j$
  - for each word w_k in *Vocabulary*
    - n_k ← number of times word w_k occurs in $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY_NAIVE_BAYES_TEXT(*Doc*)

Return the estimated target value for the document Doc. a_i denotes the word found in the ith position within Doc.

- *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return v_{NB}, where

$$v_{NB} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Example : Movie Review

Training Data

Examples	Text	Class
1	I loved the movie	+
2	I hated the movie	-
3	a great movie. good movie	+
4	poor acting	-
5	great acting. good movie	+

Vocabulary

< I, loved, the, movie, hated, a, great, good, poor, acting >

Test Data

I hated the poor acting

Example : Text Classification

$docs_j \leftarrow$ the subset of documents from *Examples* for which the target value is v_j

Documents with positive (+) class

Docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
1	1	1	1	1							+
3				2		1	1	1			+
5				1			1	1		1	+

$P(v_j) \leftarrow |\, docs_j \,| \,/\, |\text{Examples}|$ $\qquad P(+) = \frac{3}{5} = 0.6$

$Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$

$Text_j \leftarrow$ < I loved the movie a great movie. good movie great acting. good movie >

Docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
1	1	1	1	1							+
3				2		1	1	1			+
5				1			1	1		1	+

$n \leftarrow$ total number of distinct word positions in $Text_j$

$n \leftarrow$

Example 1:Text Classification

- For each word in vocabulary
- Nk<-no.of times word wk occurs in textj

$$P(w_k/v_j) \leftarrow (n_k + 1) / (n + |Vocabulary|)$$

Docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
1	1	1	1	1							+
3				2		1	1	1			+
5				1			1	1		1	+

Calculating all prior probabilities

$P(I|+)=(1+1)/(13+10)=0.08695$
$P(loved|+)=(1+1)/(13+10)=0.08695$
$P(the|+)=(1+1)/(13+10)=0.08695$
$P(movie|+)=(4+1)/(13+10)=0.2174$
$P(hated|+)=(0+1)/(13+10)=0.0435$
$P(a|+)=(1+1)/(13+10)=0.08695$
$P(great|+)=(2+1)/(13+10)=0.1304$
$P(good|+)=(2+1)/(13+10)=0.1304$
$P(poor|+)=(0+1)/(13+10)=0.0435$
$P(acting|+)=(1+1)/(13+10)=0.08695$

Example 1: Text Classification

$docs_j \leftarrow$ the subset of documents from *Examples* for which the target value is v_j

Documents with positive (-) class

docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
2	1		1	1	1						-
4									1	1	-

$P(v_j) \leftarrow |\, docs_j \,| \,/\, |\text{Examples}|$ $\qquad P(-) = \frac{2}{5} = 0.4$

$Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$

$Text_j \leftarrow$ < I hated the movie poor acting >

docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
2	1		1	1	1						-
4									1	1	-

$n \leftarrow$ total number of distinct word positions in $Text_j$

$n \leftarrow 6$

Example 1: Text Classification

- for each word w_k in *Vocabulary*

$n_k \leftarrow$ number of times word $\mathbf{w_k}$ occurs in $Text_j$ $\quad$ $P(w_k|v_j) \leftarrow (n_k + 1) / (n + |Vocabulary|)$

docs	I	loved	the	movie	hated	a	great	good	poor	acting	Class
2	1		1	1	1						-
4									1	1	-

$$P(I\,|-) = \frac{1+1}{6+10} = 0.125 \qquad P(a\,|-) = \frac{0+1}{6+10} = 0.0625$$

$$P(loved\,|-) = \frac{0+1}{6+10} = 0.0625 \qquad P(great\,|-) = \frac{0+1}{6+10} = 0.0625$$

$$P(the\,|-) = \frac{1+1}{6+10} = 0.125 \qquad P(good\,|-) = \frac{0+1}{6+10} = 0.0625$$

$$P(movie|-) = \frac{1+1}{6+10} = 0.125 \qquad P(poor|-) = \frac{1+1}{6+10} = 0.125$$

$$P(hated\,|-) = \frac{1+1}{6+10} = 0.125 \qquad P(acting|-) = \frac{1+1}{6+10} = 0.125$$

Contd..

Let's classify the new document

I hated the poor acting

If $V_j = -$

then,

$= P(-)\,P(I \mid -)\,P(hated \mid -)\,P(the \mid -)\,P(poor \mid -)\,P(acting \mid -)$

$= 0.4 * 0.125 * 0.125 * 0.125 * 0.125 * 0.125$

$= \boxed{1.22 \times 10^{-5}}$

•

Contd..

If $V_j = +$

then,

$= P(I|+)\, P(hated|+)\, P(the|+)\, P(poor|+)\, P(acting|+)$

$= 0.08695 \times 0.0435 \times 0.08695 \times 0.0435 \times 0.08695$

$= 1.2439 \times e^{-6}$

- So,it belongs to negative class.

Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

msg=pd.read_csv('/Users/Chachu/Documents/Python Scripts/naivetext.csv',names=['message','label'])

print('The dimensions of the dataset',msg.shape)

msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
y=msg.labelnum

#splitting the dataset into train and test data
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print ('\n the total number of Training Data :',ytrain.shape)
print ('\n the total number of Test Data :',ytest.shape)


#output the words or Tokens in the text documents
cv = CountVectorizer()
xtrain_dtm = cv.fit_transform(xtrain)
xtest_dtm=cv.transform(xtest)
print('\n The words or Tokens in the text documents \n')
print(cv.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=cv.get_feature_names())
```

		Message	label	
	1	I love this sandwich	pos	
	2	This is an amazing place	pos	
	3	I feel very good about these beers	pos	
	4	This is my best work	pos	
	5	What an awesome view	pos	
	6	I do not like this restaurant	neg	
	7	I am tired of this stuff	neg	ytrain
xtrain	8	I can't deal with this	neg	
	9	He is my sworn enemy	neg	
	10	My boss is horrible	neg	
	11	This is an awesome place	pos	
	12	I do not like the taste of this juice	neg	
	13	I love to dance	pos	
	14	I am sick and tired of this place	neg	
	15	What a great holiday	pos	
	16	That is a bad locality to stay	neg	
xtest	17	We will have good fun tomorrow	pos	ytest
	18	I went to my enemy's house today	neg	

Contd..

```
# Training Naive Bayes (NB) classifier on training data.
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
```

```
#printing accuracy, Confusion matrix, Precision and Recall
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))
print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision', metrics.precision_score(ytest,predicted))
print('\n The value of Recall', metrics.recall_score(ytest,predicted))
```

Output:

```
The dimensions of the dataset (18, 2)

 the total number of Training Data : (13,)

 the total number of Test Data : (5,)

 The words or Tokens in the text documents

['am', 'amazing', 'an', 'and', 'awesome', 'bad', 'best', 'boss', 'do', 'enemy', 'fun', 'good', 'great', 'have', 'he', 'holida
y', 'horrible', 'is', 'juice', 'like', 'locality', 'love', 'my', 'not', 'of', 'place', 'restaurant', 'sandwich', 'sick', 'sta
y', 'stuff', 'sworn', 'taste', 'that', 'the', 'this', 'tired', 'to', 'tomorrow', 'view', 'we', 'what', 'will', 'work']

 Accuracy of the classifier is 1.0

 Confusion matrix
[[2 0]
 [0 3]]

 The value of Precision 1.0

 The value of Recall 1.0
```

CSV: naivetext.csv

1	I love this sandwich	pos
2	This is an amazing place	pos
3	I feel very good about these beers	pos
4	This is my best work	pos
5	What an awesome view	pos
6	I do not like this restaurant	neg
7	I am tired of this stuff	neg
8	I can't deal with this	neg
9	He is my sworn enemy	neg
10	My boss is horrible	neg
11	This is an awesome place	pos
12	I do not like the taste of this juice	neg
13	I love to dance	pos
14	I am sick and tired of this place	neg
15	What a great holiday	pos
16	That is a bad locality to stay	neg
17	We will have good fun tomorrow	pos
18	I went to my enemy's house today	neg