

Drexel University
College of Computing and Informatics
INFO 371 – Data Mining Applications
Assignment 4

Name: Yuming Chen

Student Number: 320180939611

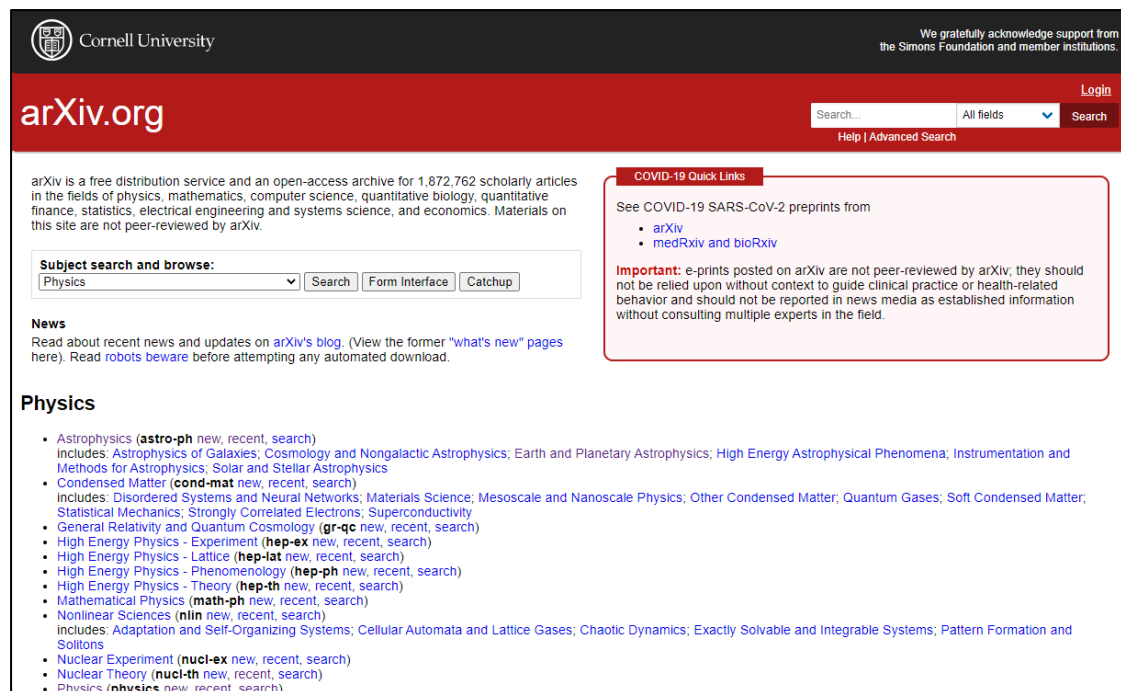
Due Date: 11:59pm, Sunday, May 23, 2021

Catalog

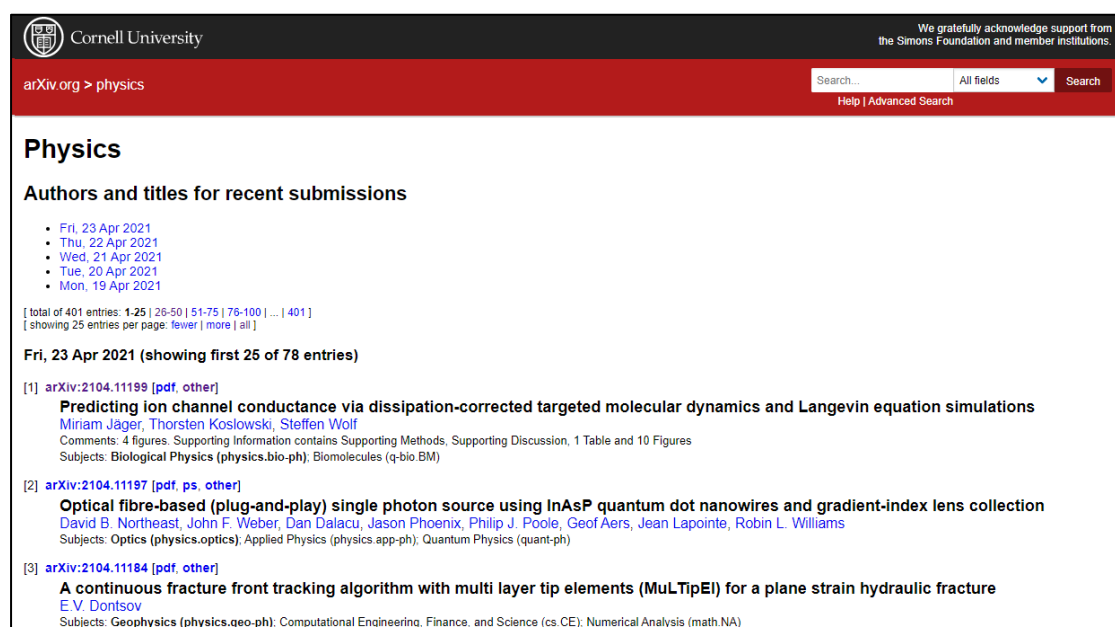
1. Data collection	3
1.1 Python Code.....	4
1.2 Overview of Dataset	6
1.3 Overview of ARFF file	6
2. Task 1: Split the Data set into Training and Test sets.....	7
2.1 Split.....	7
2.2 Overview the Test Set.....	7
2.3 Overview the Training Set	8
3. Task 2: Naïve Bayes Classification using Weka	8
3.1 Training model.....	8
3.2 Questions and Explanation.	11
4. Task 3: Naïve Bayes Classification by Manually Computing Conditional Probabilities	17
4.1 Data Preprocess.....	17
4.2 Compute and Analyze.....	19

1. Data collection

The source of data is a famous curated research-sharing platform named **arXiv** (<https://arxiv.org>) supported by Cornell University Library. **arXiv** is the world's premier e-print repository in physics, math, computer science and related disciplines enabling scientists worldwide to share and access research before it is formally published. In this assignment, I chose three different domains which were “Physics”, “Mathematic (math)” and “Computer Science (cs)” to complete the next steps.



The screenshot shows the arXiv.org homepage. At the top, there is a Cornell University logo and a search bar. Below the search bar, there is a section for "Subject search and browse:" with a dropdown menu set to "Physics". To the right, there is a "COVID-19 Quick Links" section with links to "arXiv" and "medRxiv and bioRxiv". Below this, there is a "News" section with a link to "arXiv's blog". The main content area is titled "Physics" and lists various subfields with links to "new", "recent", and "search" for each. The subfields listed are: Astrophysics (astro-ph), Condensed Matter (cond-mat), High Energy Physics - Experiment (hep-ex), High Energy Physics - Lattice (hep-lat), High Energy Physics - Phenomenology (hep-ph), High Energy Physics - Theory (hep-th), Mathematical Physics (math-ph), Nonlinear Sciences (nlin), Nuclear Experiment (nucl-ex), Nuclear Theory (nucl-th), and Physics (physics).



The screenshot shows the arXiv.org physics page. At the top, there is a Cornell University logo and a search bar. Below the search bar, there is a section for "Authors and titles for recent submissions". The page lists recent submissions with their titles, authors, and subjects. The first submission is titled "Predicting ion channel conductance via dissipation-corrected targeted molecular dynamics and Langevin equation simulations" by Miriam Jäger, Thorsten Koslowski, and Steffen Wolf. The second submission is titled "Optical fibre-based (plug-and-play) single photon source using InAsP quantum dot nanowires and gradient-index lens collection" by David B. Northeast, John F. Weber, Dan Dalacu, Jason Phoenix, Philip J. Poole, Geof Aers, Jean Lapointe, and Robin L. Williams. The third submission is titled "A continuous fracture front tracking algorithm with multi layer tip elements (MuLTiPEI) for a plane strain hydraulic fracture" by E.V. Dontsov.

1.1 Python Code

1.1.1 Crawler

Firstly, I captured and analyzed the packets of **arXiv** website. Then, I wrote a python script with “request” package which is a simple HTTP library to get the contents of abstracts from **arXiv** website for each article in the 3 given domains. According to the requirement, I filtered abstracts with *words less than 150*. Moreover, in order to reduce the negative impact on the model’s effect caused by the abstracts which belong to several domains at same time, I filtered the *abstracts which belong to serval domains*.

```
# -*- coding: utf-8 -*-
"""
INFO-371 Data-Mining Assignment 2
Simple python script to get abstract for article in given domains from arXiv.
"""
__license__ = "GPL V3"
__version__ = "0.1"
__status__ = "Experimental"
from bs4 import BeautifulSoup
from tqdm import tqdm
import pandas as pd
import requests
import json
import time
import re

class ArxivCrawler:
    def __init__(self, headers_path, domains):
        with open(headers_path, 'r') as f:
            self.headers = json.load(f)
        self.root = "https://arxiv.org"
        self.domains = domains
        self.process = tqdm(self.domains)

    def get(self, url):
        while True:
            try:
                return BeautifulSoup(requests.get(url, headers=self.headers).text, "lxml")
            except requests.exceptions.ConnectionError:
                self.process.set_description("Fixing")
                time.sleep(10)
                continue

    def run(self):
        results = pd.DataFrame(columns=["abstract", "domain"])
        for domain in self.process:
            self.process.set_description("Article in Domain {} collecting".format(domain))
            domain_url = self.get(self.root + "/list/{}/recent".format(domain)).find(text="all")
                .parent.get("href")
            soup = self.get(self.root + domain_url)
            articles = zip(soup.find_all("dt"), soup.find_all("dd"))
            count = 0
            for article_url, article_info in articles:
                article_url = article_url.find(attrs={"title": "Abstract"}).get("href")
                if self.__filter(domain, article_info.find(attrs={"class": "list-subjects"}).text):
                    self.process.set_description("Article No.{} in Domain {} downloading"
                                                .format(count, domain))
                    soup = self.get(self.root + article_url)
                    abstract = self.__text_process(soup.find(attrs={"class": "abstract mathjax"}).text)

                    if abstract:
                        results = results.append({"abstract": abstract, "domain": domain},
                                                ignore_index=True)
                        count += 1
                    if count >= 20:
                        break
            return results

    @staticmethod
    def __filter(domain, info):
        return len(re.findall("{}\{{}[^;]*\}".format(domain), info)) >= len(info.split(";"))-1

    @staticmethod
    def __text_process(text):
        text = text.replace("\n", " ").strip()
        return text[len("Abstract:"):] if len(text[len("Abstract:"):].split(" ")) >= 150 else False
```

1.1.2 Prepare the data in the proper ARFF format

1.1.2.1 Remove the non-English characters

I wrote python script with “re” package which provides regular expression matching operations to remove the non-English characters including digits in the data set gotten by crawler.

```
# -*- coding: utf-8 -*-
"""
INFO-371 Data-Mining Assignment 2
Python script to remove the non-English characters.
"""
__license__ = "GPL V3"
__version__ = "0.1"
__status__ = "Experimental"

import pandas as pd
import re

fil = re.compile(u'^a-zA-Z0-9'+
', re.UNICODE)
data = pd.read_csv("abstracts.csv")
data["abstract"] = data.abstract.apply(lambda x: fil.sub(' ', x))
data.domain = data.domain.apply(lambda x: "{}\{}".format(x))
data.to_csv("abstracts_new.csv", index=False)
```

1.1.2.2 Restructure

I wrote python script to restructure the DataFrame into standard format of Weka’s ARFF file.

```
# -*- coding: utf-8 -*-
"""
INFO-371 Data-Mining Assignment 2
Python script to process and reformat "csv" file into "arff" file.
"""
__license__ = "GPL V3"
__version__ = "0.1"
__status__ = "Experimental"

import pandas as pd

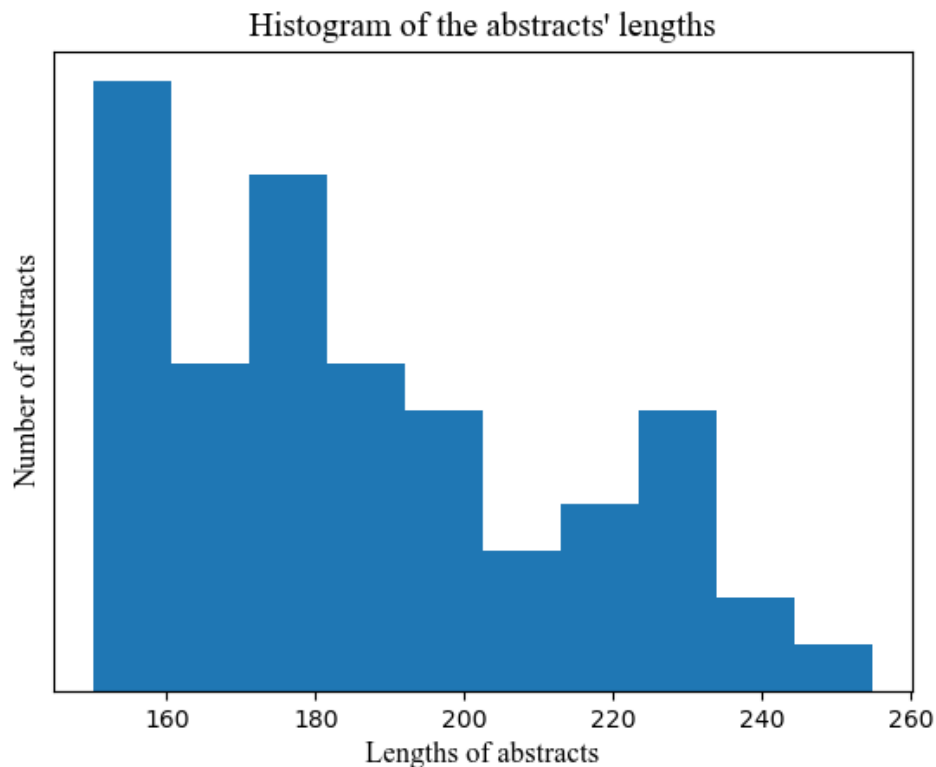
class ArffWriter:
    def __init__(self, data: pd.DataFrame, attributes: dict, filename: str):
        self.data = data
        self.attributes = attributes
        self.filename = filename

    def write(self):
        with open(self.filename+".arff", "w", encoding="utf-8", errors="ignore") as f:
            f.write("@relation {}\n\n".format(self.filename))
            for name, attribute in self.attributes.items():
                f.write("{}\n".format(attribute["des"]))
                f.write("@attribute {name} {type}\n"
                    .format(name=name, type=attribute["type"]))
            f.write("\n@data\n")
            for values in self.data.values:
                for value_id in range(len(values)):
                    mark = "\"\" if attributes[list(attributes.keys())[value_id]]["type"] ==
                    "string" else ""
                    f.write("{}{}{}\n".format(mark, values[value_id], mark))
                if value_id != len(values)-1:
                    f.write(",")
            f.write("\n")

if __name__ == "__main__":
    filename = "abstracts"
    data = pd.read_csv("abstracts_new.csv")
    attributes = {"abstract":
        {"type": "string",
         "des": "The abstract of the article"},
        "domain":
        {"type": "%s" % " ".join(data["domain"].unique()),
         "des": "The domain of the article"}}
    arffwriter = ArffWriter(data[["abstract", "domain"]], attributes, filename)
    arffwriter.write()
```

1.2 Overview of Dataset

Domain	Number of Articles	Length of articles' abstracts	
		Max(words)	Min(words)
Computer Science	20	257	155
Mathematics	20	324	157
Physics	20	280	158



1.3 Overview of ARFF file

```

1  @relation abstracts
2
3  %The abstract of the article
4  @attribute abstract string
5  %The domain of the article
6  @attribute domain {physics,math,cs}
7
8  @data
9  "Motivated from the quadratic dependence of peak structural displacements to the pulse period T p of pulse like ground mo
10 "In the part I of the foundation of the hyperunified field theory we have shown the presence of entangled hyperqubit spin
11 "For a discrete function f left x right on a discrete set the finitedifference can be either forward and backward However
12 "Nonspecific molecular adsorption like airborne contamination occurs on mostsurfaces including those of 2D materials and
13
14 .....
15
16 "The Python language has extension frameworks such as NumPy and SciPy for providing functionality involving numerical arr
17 "In practical optimisation the dominant characteristics of the problem are often not known prior Therefore there is a nee
18 "Neural Language Models NLM when trained and evaluated with contextspanning multiple utterances have been shown to consi
19 "In the real world medical datasets often exhibit a long tailed datadistribution i e a few classes occupy most of the da
20 "OCaml is particularly well fitted for formal verification On one hand it is a multi paradigm language with a well define

```

2. Task 1: Split the Data set into Training and Test sets

2.1 Split

I used python script to split the entire dataset into a training set with 90% of the instances and a test set with 10% of the instances with *DataFrame* method “sample”. Then, I restructure the *DataFrame* into *Arff* file format.

```
import pandas as pd
from arffwriter import ArffWriter

# Split into train data and test data by sample
total = pd.read_csv("abstracts_new.csv")
total.index.name = "origin_index"
test = total.groupby("domain").sample(frac=0.1)
train = total.drop(index=test.index)

# Restructure the DataFrame
datas = {"train":train,"test":test}
for name in datas:
    attributes = {"abstract":
                  {"type": "string",
                   "des": "The abstract of the article"},
                  "domain":
                  {"type": "{%s}" % ", ".join(datas[name]["domain"].unique()),
                   "des": "The domain of the article"}}
    writer = ArffWriter(datas[name][["abstract", "domain"]], attributes, "abstract"+name)
    writer.write()
```

2.2 Overview the Test Set

As there are 20 instances for each domain: Computer Science, Mathematics or physics , the test set contains $20 \times 10\% = 2$ instances for each domain. In the table, Paper# is the origin index of the paper in its domain. The randomly chosen instances are as follows:

Paper#	Domain	Abstract
37	Mathematics	"A spherical conical metric g on a surface Sig...
33	Mathematics	"In this paper we establish the monotone conve...
9	Physics	"Optical wave based computing has enabled the ...
17	Physics	"Valley degree of freedom an excellent informa...
47	Computer Science	"Aerial scene recognition is a fundamental vis...
58	Computer Science	"In the real world medical datasets often exhi...

```
1 @relation abstract-test
2
3 %The abstract of the article
4 @attribute abstract string
5 %The domain of the article
6 @attribute domain {math,physics,cs}
7
8 @data
9 "A spherical conical metric g on a surface Sigma is a metric of constantcurvature 1 with finitely many isolated conical
10 "In this paper we establish the monotone convex order between two mathbb R valued McKean Vlasov processes X X t t in 0 T
11 "Optical wave based computing has enabled the realization of real timeinformation processing in both space and time doma
12 "Valley degree of freedom an excellent information carrier in valleytronics has been further introduced into advanced mi
13 "Aerial scene recognition is a fundamental visual task and has attracted anincreasing research interest in the last few
14 "In the real world medical datasets often exhibit a long tailed datadistribution i e a few classes occupy most of the da
```

2.3 Overview the Training Set

Paper#	Domain	Abstract
0	Physics	Motivated from the quadratic dependence of pe...
1	Physics	In the part I of the foundation of the hyperu...
.....		
58	Computer Science	In the real-world medical datasets often exhi...
59	Computer Science	OCaml is particularly well fitted for formal ...

```

1  @relation abstract-train
2
3  %The abstract of the article
4  @attribute abstract string
5  %The domain of the article
6  @attribute domain {physics,math,cs}
7
8  @data
9  "Motivated from the quadratic dependence of peak structural displacements tothe pulse period T p of pulse like ground mo
10 "In the part I of the foundation of the hyperunified field theory we haveshown the presence of entangled hyperqubit spin
.....
60 "Neural Language Models NLM when trained and evaluated with contextspanning multiple utterances have been shown to consi
61 "In the real world medical datasets often exhibit a long tailed datadistribution i e a few classes occupy most of the da
62 "OCaml is particularly well fitted for formal verification On one hand it isa multi paradigm language with a well define

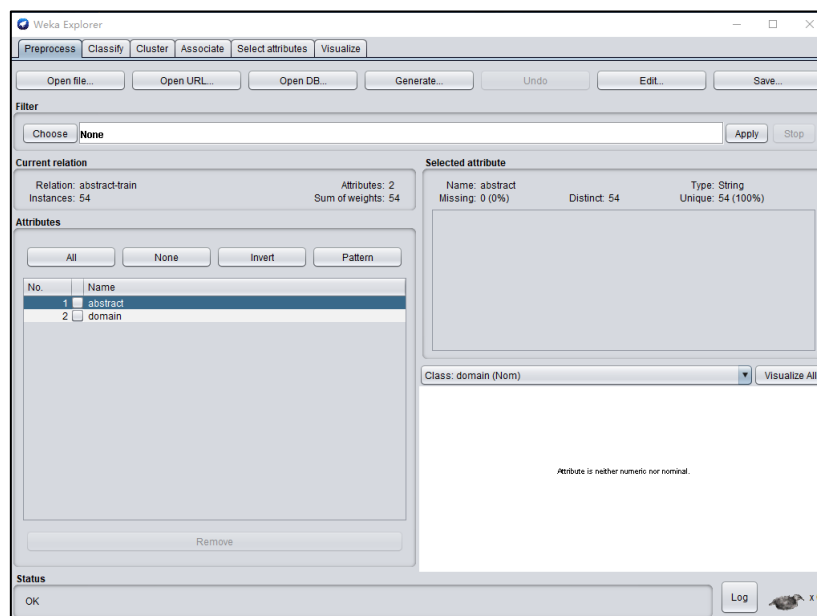
```

3. Task 2: Naïve Bayes Classification using Weka

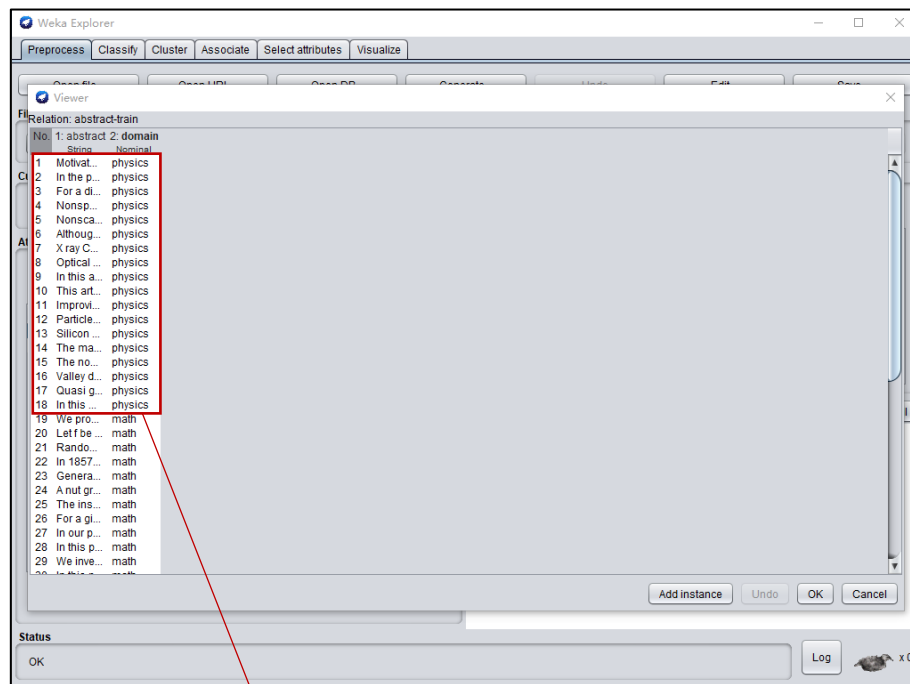
3.1 Training model

3.1.1 Load train dataset

3.1.1.1 Open abstracts-train.arff.

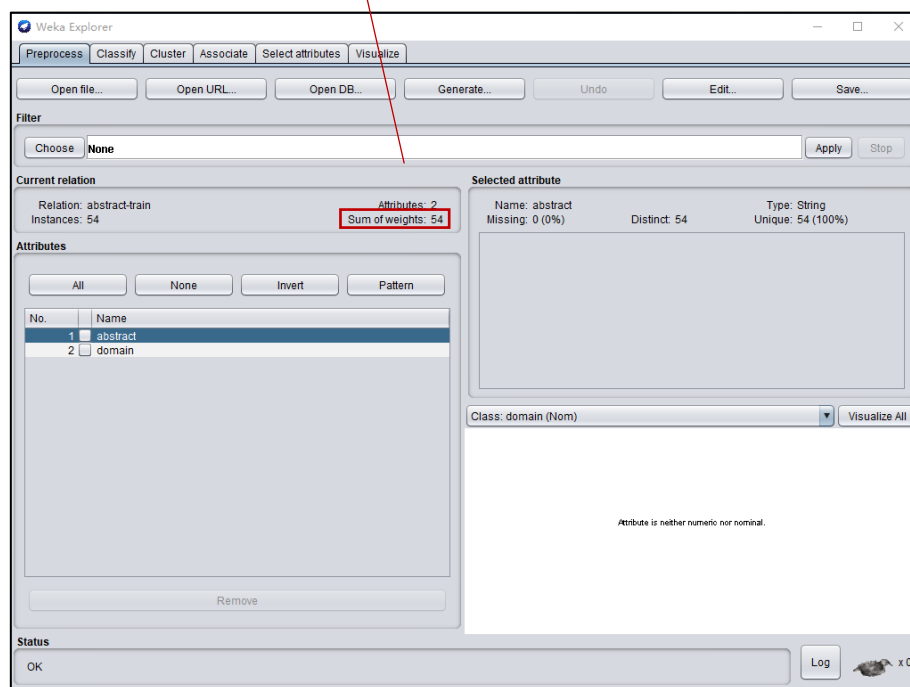


3.1.1.2 Click the “Edit” button to view the raw data.

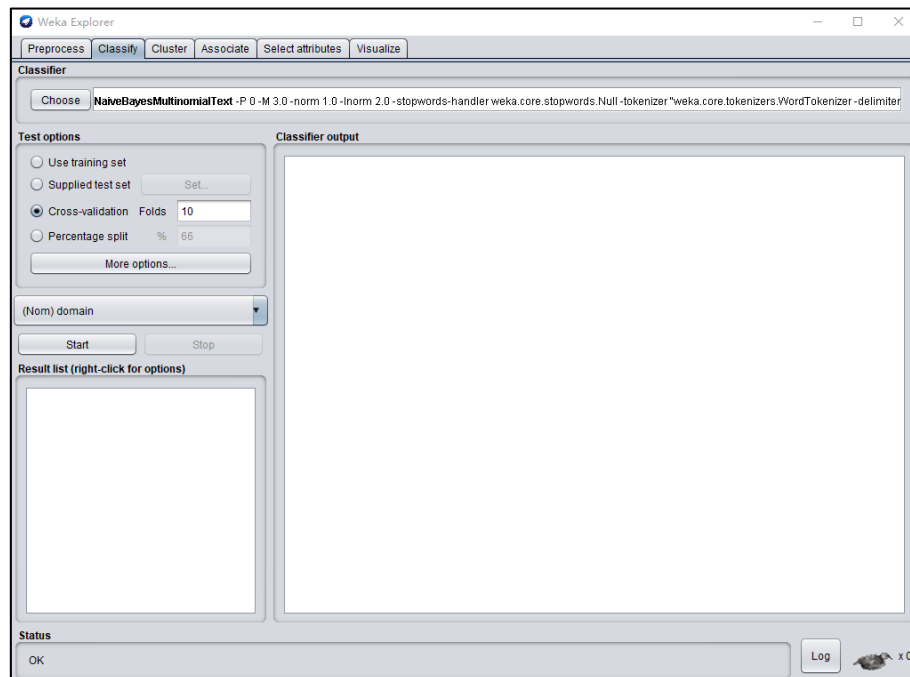


3.1.1.3 Make sure there are 90% of the instances in each domain.

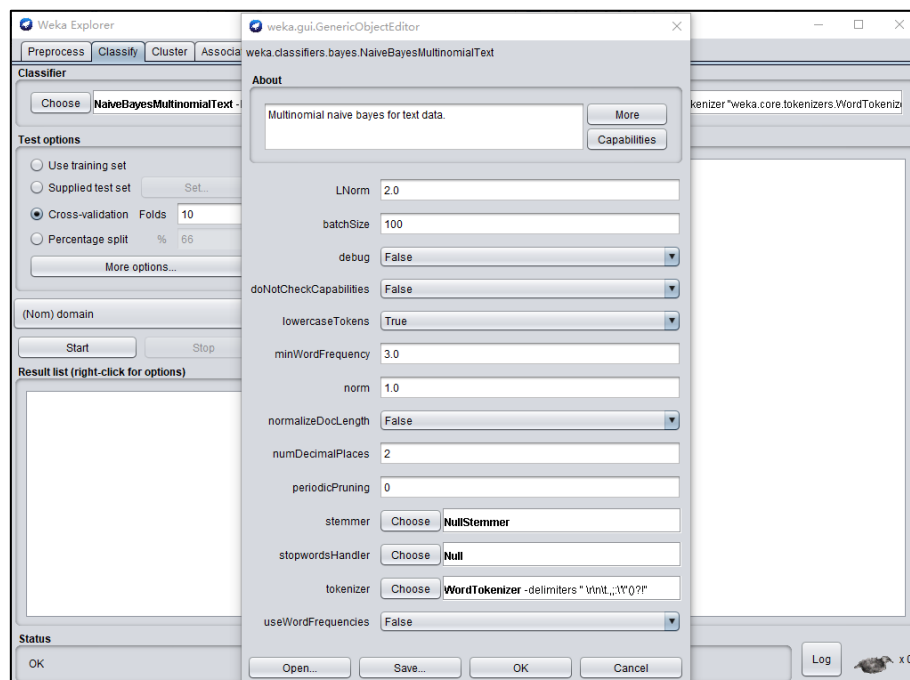
From the main explorer, the train dataset has 90% of the instances. From the viewer, there are 90% of the instances in each domain.



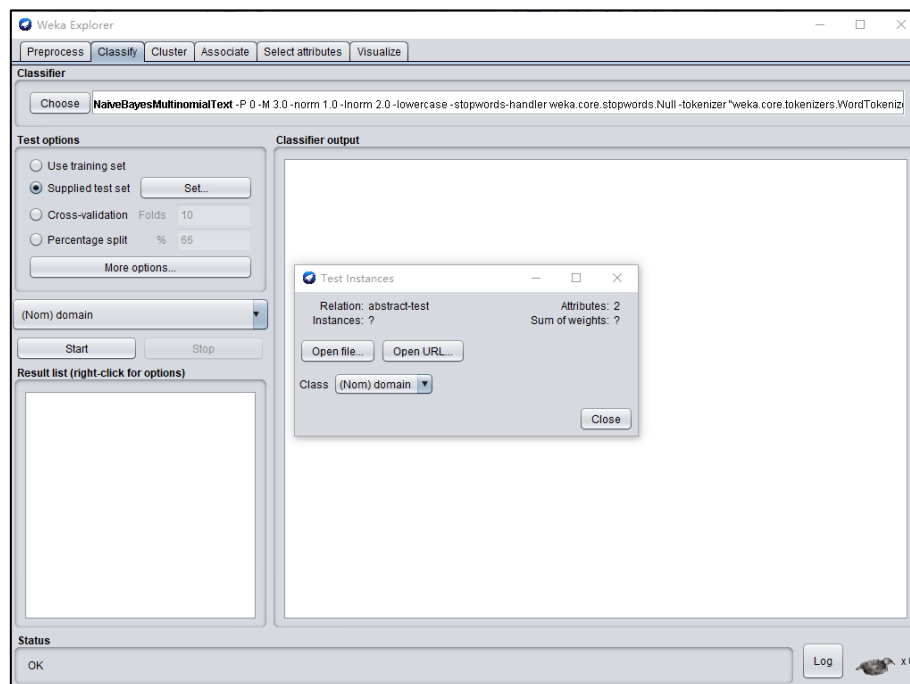
3.1.2 Click the classify tab and choose **weka->classifiers->bayes->NaiveBayesMultinomialText**.



3.1.3 Click the classifier name to open the property editor. On the property editor, check **lowercaseTokens** to be True. Click OK.

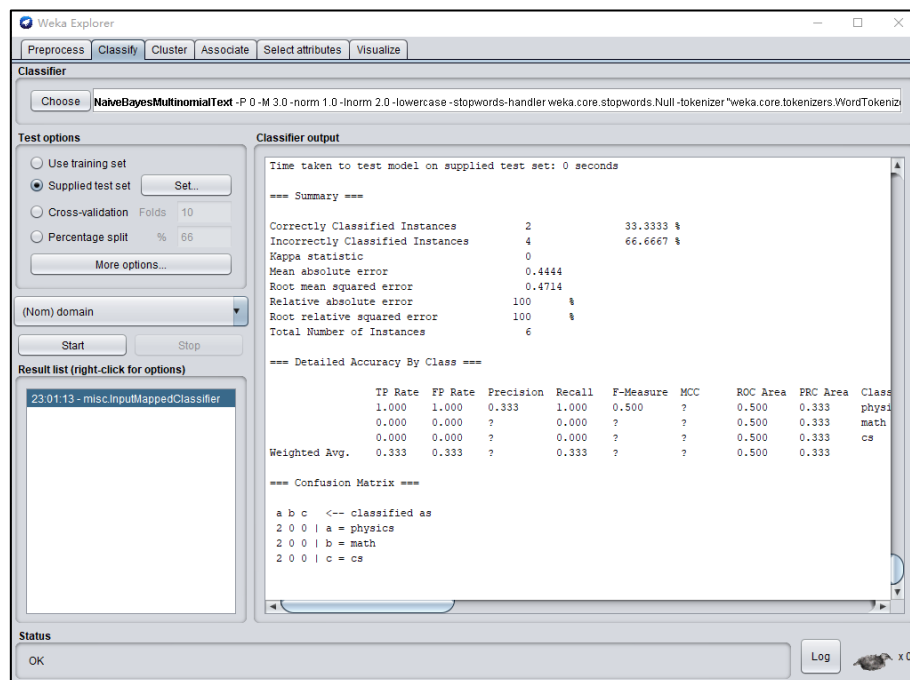


3.1.4 On Test Option, check **Supplied test set**. Click the **Set...** button. Open the **abstracts-test.arff** file. Make sure the Class is **(Nom) domain**.



3.2 Questions and Explanation.

Part of the output is as follows:



```

=== Run information ===
Scheme:      weka.classifiers.misc.InputMappedClassifier -I -trim -W weka.classifiers.bayes.NaiveBayesMultinomialText --
-P 0 -M 3.0 -norm 1.0 -lnorm 2.0 -lowercase -stopwords-handler weka.core.stopwords.Null -tokenizer
"weka.core.tokenizers.WordTokenizer -delimiters \"\\r\\n\\t.,;:\\\\\"'\"()?!\" -stemmer weka.core.stemmers.NullStemmer
Relation:      abstract-train-weka.filters.unsupervised.attribute.StringToWordVector-R1-Pabstract--W100-prune-
rate-1.0-C-N0-L-stemmerweka.core.stemmers.LovinsStemmer-stopwords-handlerweka.core.stopwords.Rainbow-
M1-tokenizerweka.core.tokenizers.WordTokenizer -delimiters \"\\r\\n\\t.,;:\\\"()?!\"
Instances:      54
Attributes:      319
                [list of attributes omitted]
Test mode:      user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===
InputMappedClassifier:

Dictionary size: 0

The independent frequency of a class
-----
physics    19.0
math 19.0
cs    19.0

The frequency of a word given the class
-----
           physics           math           cs

Attribute mappings:

Model attributes                               Incoming attributes
-----
(nominal) domain                             --> 2 (nominal) domain
(numeric) abstract-0                         --> - missing (no match)
(numeric) abstract-1                         --> - missing (no match)
(numeric) abstract-2                         --> - missing (no match)
.....
(numeric) abstract-trust                     --> - missing (no match)
(numeric) abstract-typ                       --> - missing (no match)
(numeric) abstract-user                     --> - missing (no match)
(numeric) abstract-vide                     --> - missing (no match)
(numeric) abstract-world                     --> - missing (no match)

Time taken to build model: 0 seconds

=== Evaluation on test set ===
Time taken to test model on supplied test set: 0 seconds

=== Summary ===
Correctly Classified Instances          2          33.3333 %
Incorrectly Classified Instances        4          66.6667 %
Kappa statistic                        0
Mean absolute error                    0.4444
Root mean squared error                0.4714
Relative absolute error                 100      %
Root relative squared error            100      %
Total Number of Instances              6

=== Detailed Accuracy By Class ===

           TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
           1.000    1.000    0.333    1.000    0.500    ?        0.500    0.333    physics
           0.000    0.000    ?        0.000    ?        ?        0.500    0.333    math
           0.000    0.000    ?        0.000    ?        ?        0.500    0.333    cs
Weighted Avg.   0.333    0.333    ?        0.333    ?        ?        0.500    0.333

=== Confusion Matrix ===
a b c  <-- classified as
2 0 0 | a = physics
2 0 0 | b = math
2 0 0 | c = cs
=== Run information ===

```

Use the results in the Classifier output area to answer the following questions

3.2.1 What is the dictionary size of the classifier model?

```

=== Classifier model (full training set) ===

InputMappedClassifier:
Dictionary size: 651

```

According to the output, the dictionary size of the classifier model is **653**.

3.2.2 What is the total number of instances in the training set?

```

Instances: 54
Attributes: 2
            abstract
            domain

```

According to the output above, the total number of instances in the training set is **54**.

3.2.3 How many instances in each domain in the training set?

```

The independent frequency of a class
-----
physics 19.0
math    19.0
cs      19.0

```

According to the output, the number of instances in each domain in the training set is **18**. (19 - 1 = 18, Weka start from 1)

3.2.4 How many instances in the test set?

```

Correctly Classified Instances      2      33.3333 %
Incorrectly Classified Instances    4      66.6667 %
Kappa statistic                     0
Mean absolute error                 0.4444
Root mean squared error             0.4714
Relative absolute error              100      %
Root relative squared error         100      %
Total Number of Instances          6

```

According to the output, the number of instances in the test set is **6**.

3.2.5 How many test instances were incorrectly classified?

```

Correctly Classified Instances      2      33.3333 %
Incorrectly Classified Instances    4      66.6667 %
Kappa statistic                     0
Mean absolute error                 0.4444
Root mean squared error             0.4714
Relative absolute error              100      %
Root relative squared error         100      %
Total Number of Instances          6

```

According to the output, the number of test instances were incorrectly classified is **4**.

3.2.6 What is the confusion matrix of the test result? What do the values mean in the confusion matrix?

```

=== Confusion Matrix ===
  a b c  <-- classified as
  2 0 0 | a = physics
  2 0 0 | b = math
  2 0 0 | c = cs

```

The confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table). In Weka reporting, the confusion matrix is used to measure the performance for given model.

According to the right part of output, **a** represents the “physics” (Physics), **b** represents the “math” (Mathematic) and **c** represents the “cs” (Computer Science). The columns of confusion matrix show the result of classification predicted by given model on test set.

In confusion matrix, the values of row represent the real value. In the output, the 1st row contains all the instances in the “physics” (Physics) domain and there are 2 instances in this domain. The 2nd row contains all the instances in the “math” (Mathematic) domain and there are 2 instances in this domain. The 3rd row contains all the instances in the “cs” (Computer Science) domain and there are 2 instances in this domain.

On the contrary, the values of column represent the result classified by given model. In the output, the 1st column all contains the instances which are classified as the papers in the “physics” (Physics) domain by given model and there are 4 instance classified as in this domain. The 2nd column contains all the instances which are classified as the papers in the “math” (Mathematic) domain by given model and there is no instances classified as in this domain. The 3rd column contains all the instances which are classified as the papers in the “cs” (Computer Science) domain by given model and there is no instances classified as in this domain.

As for the detailed value, the row value represents its real domain and the column value represents the result classified by give model of it. For instance, the value **2** in the top right corner represents there are two instances in the “physics” (Physics) domain and they are classified as the papers in “cs” (Computer Science) domain.

3.2.7 For each domain, list the recall, precision, and F1-Measure. Describe the underlying methods of computing the metrics.

=== Detailed Accuracy By Class ===									
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1.000	1.000	0.333	1.000	0.500	?	0.500	0.333	physics
	0.000	0.000	?	0.000	?	?	0.500	0.333	math
	0.000	0.000	?	0.000	?	?	0.500	0.333	cs
Weighted Avg.	0.333	0.333	?	0.333	?	?	0.500	0.333	

According to the output, the metrics are as follows:

Domain	Recall	Precision	F1-Measure
Physics	0.333	1	0.5000
Mathematic	?	0	?
Computer Science	?	0	?

The “?” represented the situation when the denominator is 0. This case is caused by there is no instance classified as the paper in “Computer Science” domain and “Mathematic” domain which makes the denominator of Precision and F1-measure become 0.

The description of each metric are as follows:

Metrics	Description
Condition Positive (P)	The number of real positive cases in the data
Condition Negative (N)	The number of real negative cases in the data
True Positive (TP)	The number of instances which are correctly classified as belonging to the positive class.
False Positive (FP)	The number of instances which are incorrectly classified as belonging to the positive class.
True Negative (TN)	The number of instances which are correctly classified as belonging to the negative class.
False Negative (FN)	The number of instances which are incorrectly classified as belonging to the negative class.
Precision	<p>The fraction of relevant instances among the retrieved instances</p> $Precision = \frac{TP}{(FP + TP)}$
Recall	<p>The fraction of relevant instances that were retrieved.</p> $Recall = \frac{TP}{(FN + TP)}$
F1-measure	<p>The F1-measure is the harmonic mean of the precision and recall.</p> $Recall = \frac{2}{\left(\frac{1}{Recall} + \frac{1}{Precision}\right)} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$

3.2.8 What is the average recall, precision, and F1-Measure of the Naïve Bayes Classifier on the datasets? Describe the underlying methods of computing the averages.

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.000	0.000	?	0.000	?	?	1.000	1.000	physics
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	math
	1.000	0.500	0.500	1.000	0.667	0.500	1.000	1.000	cs
Weighted Avg.	0.667	0.167	?	0.667	?	?	1.000	1.000	

According to the output, the metrics are as follows:

Average Recall	Average Precision	Average F1-Measure
?	0.333	?

The “?” represented the situation when there are other “?” values participate in computing this value. The case is caused by the Recall and F1-measure of “Physics” class.

From the output, the average recall is “?” which is caused by the recall of “Physics” class, the average precision is 0.667, and the average F1-Measure is “?” which is caused by the precision of “Physics” class. These values are the weighted averages of metrics for each class and the weight is the proportion of each class.

The description of each metric are as follows:

Metrics	Description
Average Recall	$Average\ Recall = \frac{\sum_{i=1}^n N_i \times Recall_i}{N}$
Average Precision	$Average\ Precision = \frac{\sum_{i=1}^n N_i \times Precision_i}{N}$
Average F1-Measure	$Average\ F1 - Measure = \frac{\sum_{i=1}^n N_i \times F1 - Measure_i}{N}$

i represents the i^{th} class, n represents the number of classes, N represents the number of instances and N_i represents the number of instances for i^{th} class.

Suppose a represents the “physics” (Physics), b represents the “math” (Mathematic) and c represents the “cs” (Computer Science), the formula is:

$$Average\ Recall = \frac{2 \times Recall_a + 2 \times Recall_b + 2 \times Recall_c}{6}$$

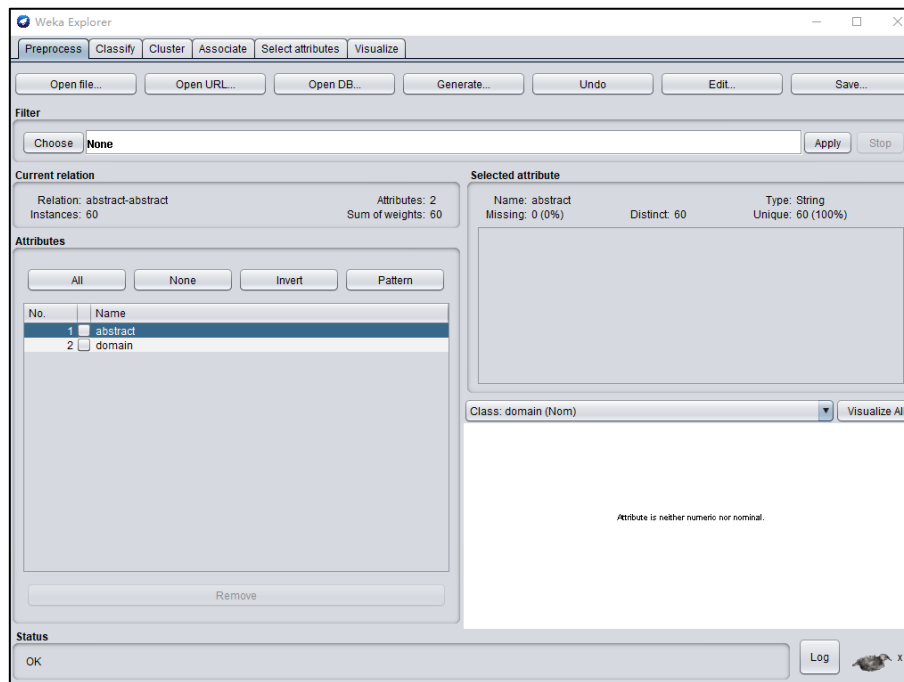
$$Average\ Precision = \frac{2 \times Precision_a + 2 \times Precision_b + 2 \times Precision_c}{6}$$

$$Average\ F1 - Measure = \frac{2 \times F1_a + 2 \times F1_b + 2 \times F1_c}{6}$$

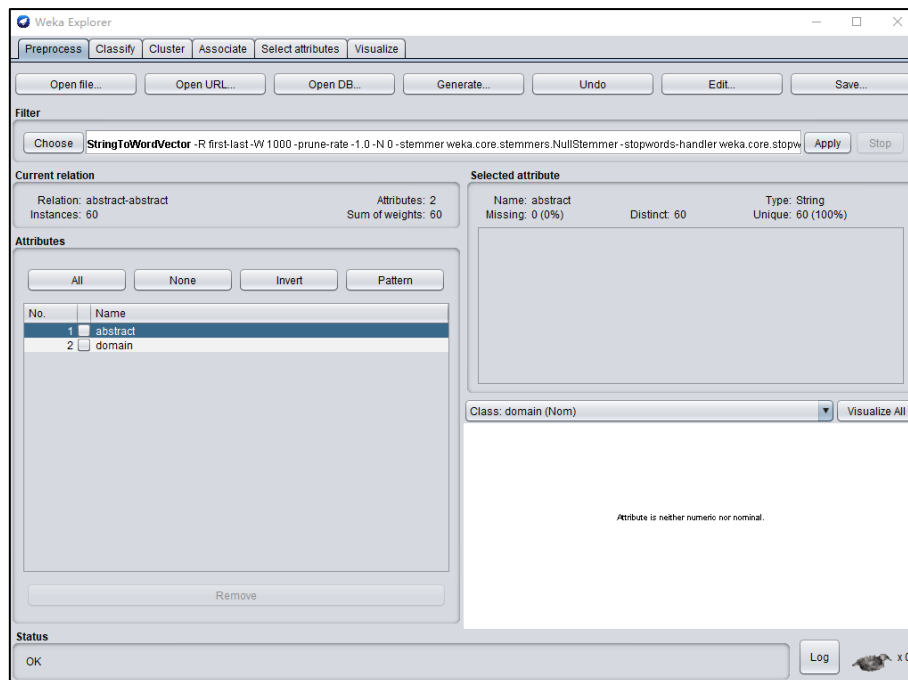
4. Task 3: Naïve Bayes Classification by Manually Computing Conditional Probabilities

4.1 Data Preprocess

4.1.1 Open **abstracts.arff** in Weka Explorer.



4.1.2 Choose Filter: weka->filters->unsupervised-> attribute-> StringToWordVector.



4.1.3 Click textbox “StringToWordVector –R ...” to bring up the property editor to change options below:

4.1.3.1 Select **true** for *lowerCaseTokens*;

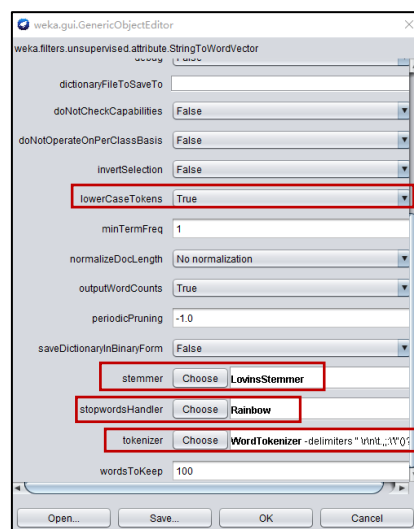
4.1.3.2 Select **true** for *outputWordCounts* (this will produce Term Frequency/counts rather than binary 0/1 values);

4.1.3.3 For *stemmer*, choose **LovinsStemmer**;

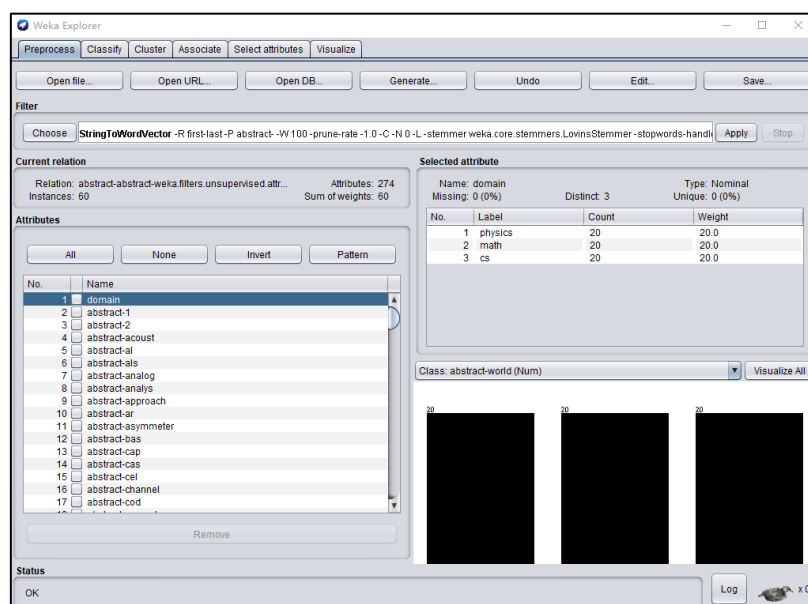
4.1.3.4 For *stopwordsHandler*, choose **Rainbow**;

4.1.3.5 For *wordsToKeep*, specify **100**;

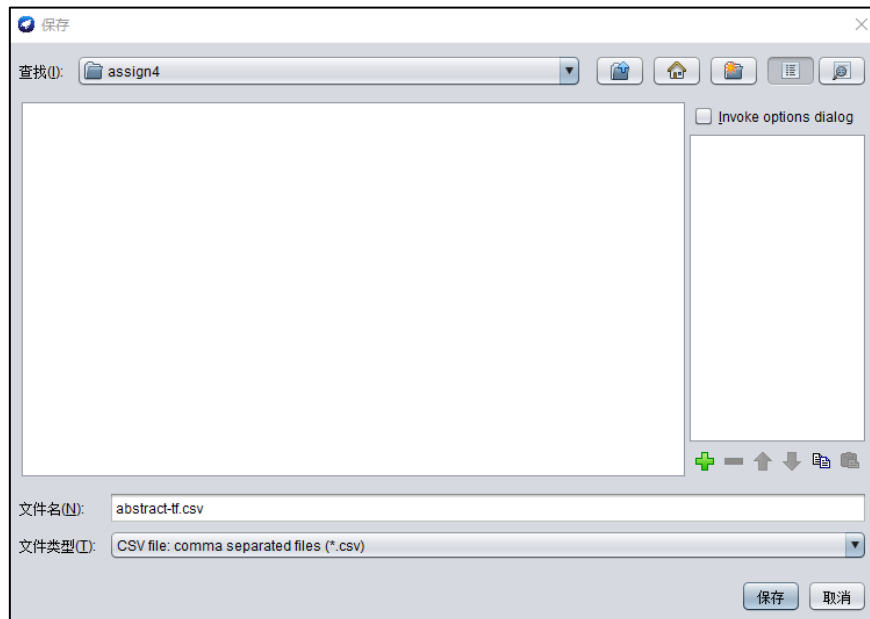
4.1.3.6 click **OK**.



4.1.4 Click **Apply** to run the vectorization filter and you will see new word attributes.



4.1.5 Click **Save...** to save the results (tf vector representation) as a **CSV** file called **abstracts-tf.csv**.



4.2 Compute and Analyze

4.2.1 Click **Edit...** to view the processed data and manually select 6 important words to **create** a data table as follows:

No	1: domain	2: abstract-1	3: abstract-2	4: abstract-acoust	5: abstract-al	6: abstract-als	7: abstract-analog	8: abstract-analys	9: abstract-approach	10: abstract-ar	11: abstract
1	physics	0.0	1.0	0.0	1.0	0.0	0.0	0.0	3.0	0.0	2.0
2	physics	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
3	physics	2.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	physics	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	physics	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
6	physics	5.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
7	physics	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0
8	physics	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
9	physics	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	physics	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	3.0
11	physics	0.0	0.0	0.0	3.0	1.0	0.0	0.0	0.0	2.0	3.0
12	physics	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	physics	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
14	physics	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
15	physics	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	2.0
16	physics	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
17	physics	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
18	physics	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	1.0
19	physics	0.0	0.0	2.0	0.0	1.0	0.0	0.0	0.0	0.0	4.0
20	physics	0.0	0.0	0.0	4.0	1.0	0.0	1.0	1.0	1.0	7.0
21	math	2.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
22	math	0.0	2.0	0.0	4.0	1.0	0.0	0.0	0.0	0.0	1.0
23	math	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0
24	math	1.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	2.0
25	math	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
26	math	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
27	math	6.0	0.0	0.0	5.0	1.0	0.0	0.0	0.0	0.0	0.0
28	math	0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0

4.2.1.1 Pick 6 words. Consider to reuse the 3 rare words (unique in each area/domain) and three other words/attributes in Assignment 2. I will use these 6 words as a small dictionary for Naïve Bayes Classification on the test instances.

Rare words: sigm, channel and datas

Other words: bas, propos and paper

4.2.1.2 Create a data table Containing the frequency of each of the above 6 words in each domain.

Frequency of the Selected 6 Dictionary Word							
Domain	datases	sigm	channel	bas	propos	paper	Grand Total
physics	0	0	10	10	12	5	37
math	0	12	0	7	4	13	36
cs	10	0	0	16	17	8	51

4.2.1.3 In the above table, an inner cell contains the frequency of a dictionary word in a specific domain. open the saved CSV file **abstracts-tf.csv** in MS Excel. Create appropriate pivot table to sum up the frequencies of the chosen 6 dictionary words.

在以下区域间拖动字段:

筛选

列

Σ 数值

行

domain

Σ 值

sigm

channel

datases

bas

propos

paper

4.2.1.4 The rightmost column of the above table contains the grand total of word frequency in each domain. These total numbers will be used to compute the conditional probabilities for each word for given a domain.

Domain	sigm	channel	datases	bas	propos	paper	Grand Total
cs	0	0	10	16	17	8	51
math	12	0	0	7	4	13	36
physics	0	10	0	10	12	5	37
Sum	12	10	10	33	33	26	124

4.2.2 Create a table containing the conditional probabilities of the 6 dictionary words. To estimate the conditional probabilities by taking zero-frequency into consideration, you need to use the following Laplace-estimator formula:

$$P(w|domain) = \frac{\text{frequency of } w \text{ in the domain} + 1}{\text{grand total frequency of all words in the domain} + \text{size of the dictionary}}$$

According to the formular, I compute the conditional probabilities for the above 6 dictionary words which selected by me using the frequencies in the above table:

Conditional Probability of the Selected 6 Dictionary Word							Sum of the probabilities
Domain	dases	sigm	channel	bas	propos	paper	
physics	$\frac{0+1}{37+6}$	$\frac{0+1}{37+6}$	$\frac{10+1}{37+6}$	$\frac{10+1}{37+6}$	$\frac{12+1}{37+6}$	$\frac{5+1}{37+6}$	1
	= 0.023	= 0.023	= 0.256	= 0.256	= 0.302	= 0.140	
math	$\frac{0+1}{36+6}$	$\frac{12+1}{36+6}$	$\frac{0+1}{36+6}$	$\frac{7+1}{36+6}$	$\frac{4+1}{36+6}$	$\frac{13+1}{36+6}$	1
	= 0.024	= 0.310	= 0.024	= 0.190	= 0.119	= 0.333	
cs	$\frac{10+1}{51+6}$	$\frac{0+1}{51+6}$	$\frac{0+1}{51+6}$	$\frac{16+1}{51+6}$	$\frac{17+1}{51+6}$	$\frac{8+1}{51+6}$	1
	= 0.193	= 0.018	= 0.018	= 0.298	= 0.316	= 0.158	

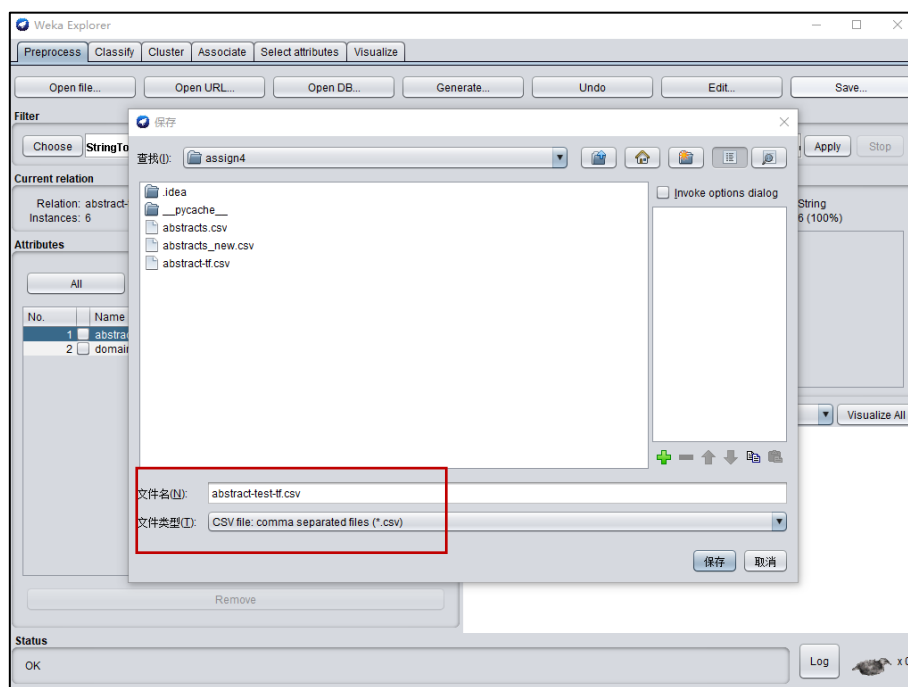
4.2.3 For each of the test instance created in Task 1, obtain the frequencies of the selected 6 dictionary words from the raw vectorization data in Weka: clicking the Edit button or from the CSV file opened in Excel.

Given the word frequencies in the test instances and the conditional probabilities of each dictionary word in the above table, I use the following formula to classify each instance:

$$d = \operatorname{argmax}_{d \in \text{domains}} P(d) \times (P(w_1|d))^{n_1} \times (P(w_2|d))^{n_2} \times \dots \times (P(w_k|d))^{n_k}$$

Where $P(d)$ is the probability of the domain d , $P(w_i|d)$ is the conditional probability of the word w_i given the domain d , and n_i is the frequency of word w_i in the instance. Since we have equal number of instances in the dataset, we can ignore the probability $P(d)$ in the calculation.

For example, the following table contains the frequencies of each dictionary word in each test instance. The columns under the title “Un-normalized Conditional Probability” contains the results of applying the above formula for each domain. Finally, the last column is the classification result.



domain	al	algebr	algorithm		world	yield
math	1	1	1		0	0
math	0	0	0		0	0
physics	0	0	0		0	0
physics	0	0	0	0	0
cs	0	0	0		1	1
cs	0	0	0		1	0

4.2.4 Create the same table as above and filled up the table using your own data.

Instance# (domain)	Frequency of the selected 6 Dictionary Word in Test Instance						Un-normalized Conditional Probability			Classification
	daseses	sigm	channel	bas	propos	paper	Given physics	Given math	Given cs	Result
37 (math)	0	2	0	0	0	1	7.5E-05	0.0319	4.9E-05	math
33 (math)	0	4	0	1	0	1	1E-08	0.0006	4.5E-09	math
9 (physics)	0	0	2	4	4	2	4.6E-08	1.7E-11	6E-10	physics
17 (physics)	0	0	1	1	1	0	0.0198	0.0005	0.0017	physics
47 (cs)	4	0	0	2	3	1	7.4E-11	6.6E-12	6.1E-07	cs
58 (cs)	1	0	0	1	1	0	0.0018	0.0005	0.01818	cs

4.2.5 Create the confusion matrix for the test instances and evaluate the performance in terms of precision, recall, and F-Measure.

4.2.5.1 Confusion Matrix

math	physics	cs	
2	0	0	math
0	2	0	physics
0	0	2	cs

4.2.5.2 The performance

Domain	Recall	Precision	F1-Measure
Physics	1	1	1
Mathematic	1	1	1
Computer Science	1	1	1
Weighted Average	1	1	1

4.2.6 Compare your results with that of Weka. Discuss any factors that may affect the results. Discuss your observations of using the classification model.

According to the above output, the result of manually computing is much better than that of Weka. In the result of Weka, there are 2 instances were incorrectly classified by Weka and 4 instances were correctly classified. However, in the result of manually computing conditional, there are 6 instances were correctly classified by Weka and there is any instance was incorrectly classified by Weka.

I think the reason is that the rare words (**sign**, **channel** and **datases**) I chose which only appear in the paper of one domain can typically represent the characteristic for each class. According to the Naïve Bayes formula, when the frequency of word is 0, the multiplier which represents the word in formula will be set as 1 which has not contribution to finally result. Therefore, the rare words will largely increase the accuracy of classification. Moreover, the other words (**bas**, **propos** and **paper**) I chose basically appear in the articles of the test set, and the frequency is basically the same. This means that they have little impact on the results and do not cause serious noise.