Web Mining(CSE3024) – Lab 10

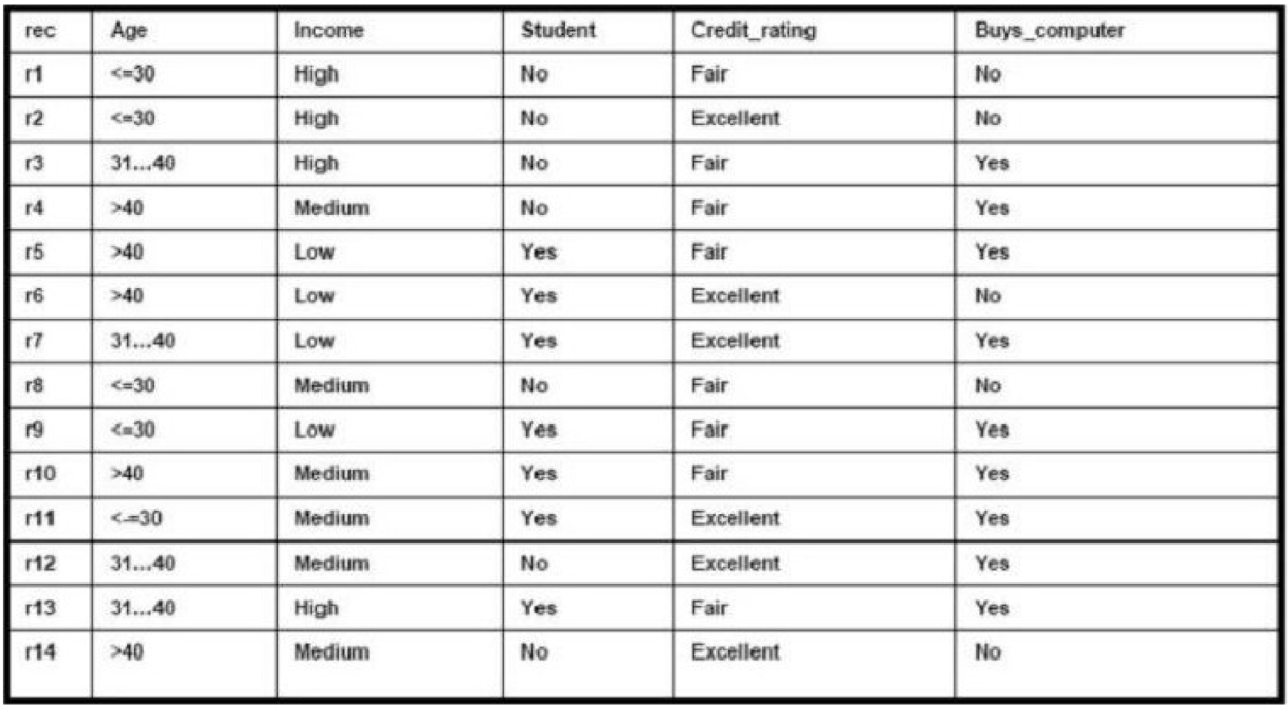
Naive Bayes Clustering

By Abhijeet Ambadekar (16BCE1156)

Problem Statement:

Part A-

Build a Naïve Bayes Classifier using this data.



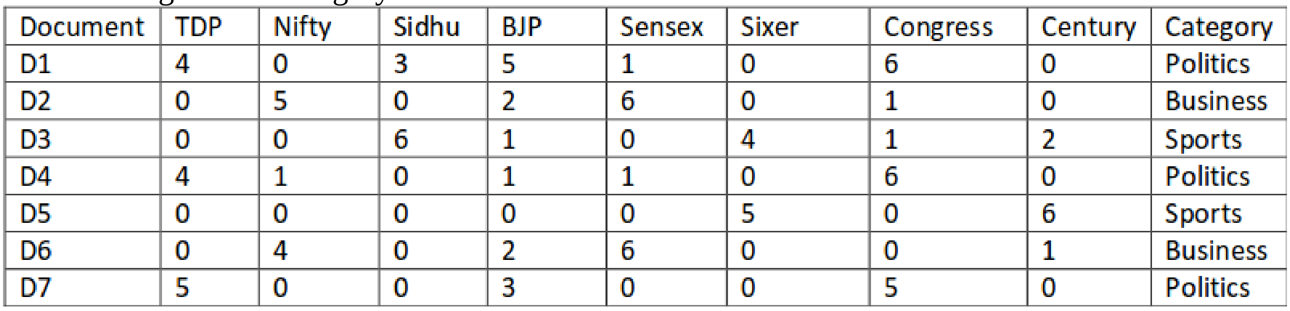
rec	Age	Income	Student	Credit_rating	Buys_computer
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31...40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31...40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31...40	Medium	No	Excellent	Yes
r13	31...40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

Predict whether the following user will buy a computer?

X= (age <=30, income = medium, student = yes, credit_rating = fair)

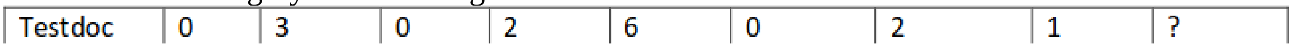
Part B -

The following gives a Term frequency of some of the documents for the given keywords and the last column gives the category of the document



Document	TDP	Nifty	Sidhu	BJP	Sensex	Sixer	Congress	Century	Category
D1	4	0	3	5	1	0	6	0	Politics
D2	0	5	0	2	6	0	1	0	Business
D3	0	0	6	1	0	4	1	2	Sports
D4	4	1	0	1	1	0	6	0	Politics
D5	0	0	0	0	0	5	0	6	Sports
D6	0	4	0	2	6	0	0	1	Business
D7	5	0	0	3	0	0	5	0	Politics

Predict which Category the following document will fall into



Testdoc	0	3	0	2	6	0	2	1	?

Program Code:

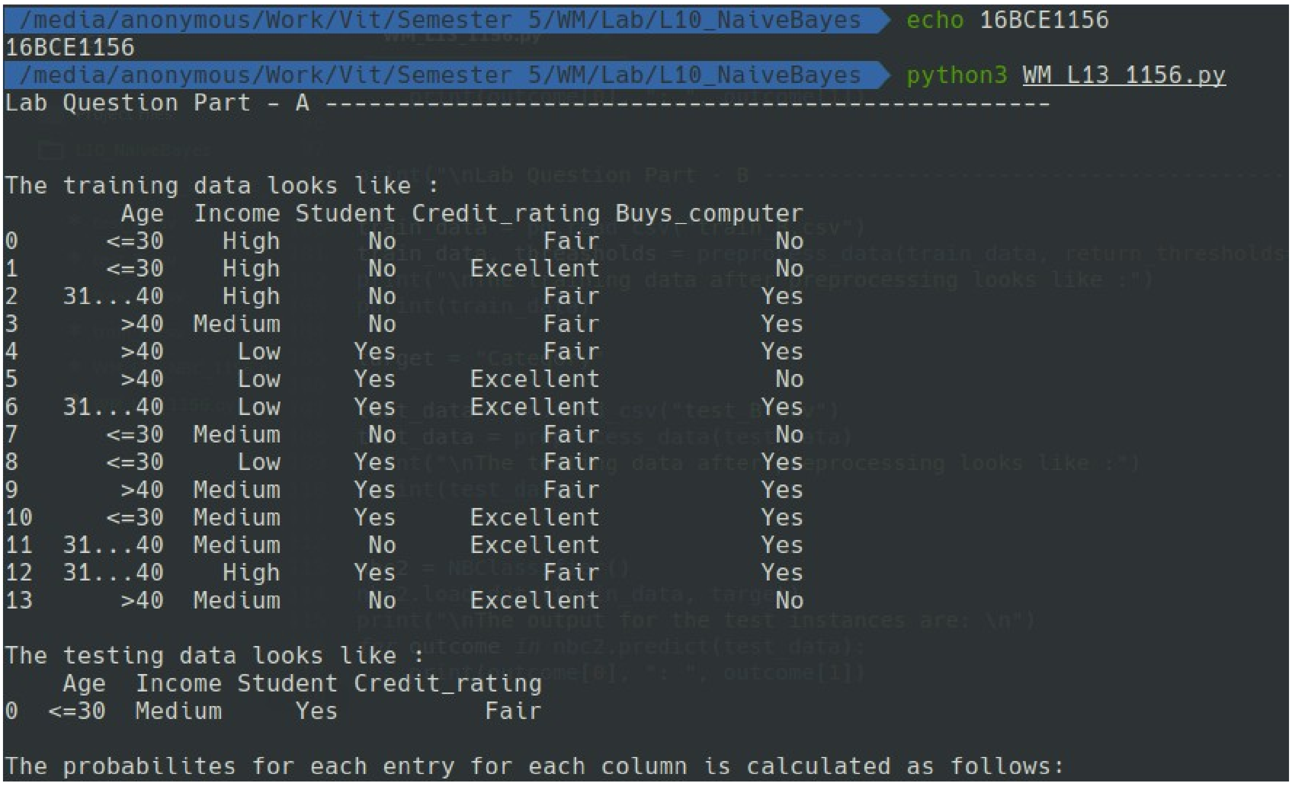
```
import pandas as pd
# import math
from operator import itemgetter
from pprint import pprint

def preprocess_data(data, return_thresholds=False):
      columns = data.columns
      thresholds = []
      for column in columns[:-1]:
            min_val = 0
            max_val = data.loc[:, column].max()
            threshold = (max_val + min_val)/2
            thresholds.append(threshold)
            data.loc[(data[column] <= threshold), column] = 0
            data.loc[(data[column] > threshold), column] = 1
      thresholds = pd.DataFrame({'column': columns[:-1], 'threshold':
thresholds}).set_index(keys = ['column'])
      # print(thresholds.head(len(columns[:-1])))
      # print('\n')
      if return_thresholds:
            return data, thresholds
      else:
            return data


class NBClassifier():
      def __init__(self):
            pass

      def load_data(self, train_data, target=None):
            self.train_data = train_data
            if target:
                  self.target = self.train_data.columns[-1]
            else:
                  self.target = target

            self.outcomes = self.train_data[self.target].unique()


            self.outcomeProb = {}
            total_count = self.train_data.shape[0]
            for outcome in self.outcomes:
                  count = self.train_data.loc[(self.train_data[self.target] ==
outcome)].shape[0]
                  self.outcomeProb[outcome] = count/total_count


            self.probTable = pd.DataFrame(columns =
['key'].append(self.outcomes))
```

```
            for column in train_data.columns[:-1]:
                  for entry in train_data[column].unique():
                        temp = {}
                        temp['key'] = column + '_' + str(entry)
                        for outcome in self.outcomes:
                              temp[outcome] = self.calcProb(column, entry,
outcome)
                        self.probTable =
self.probTable.append(pd.DataFrame(temp.copy(), index = [0]), ignore_index =
True)
            self.probTable = self.probTable.set_index('key')

            print("\nThe probabilites for each entry for each column is
calculated as follows: ")
            pprint(self.probTable)

      def calcProb(self, column, entry, outcome):
            tot_count = self.train_data.loc[self.train_data[self.target] ==
outcome].shape[0]
            count = self.train_data.loc[(self.train_data[column] == entry) &
(self.train_data[self.target] == outcome)].shape[0]
            return count/tot_count


      def predict(self, test_data):
            labels = []
            for _, test_instance in test_data.iterrows():
                  results = {}
                  for outcome in self.outcomes:
                        results[outcome] = 1
                        for key in test_instance.keys():
                              results[outcome] *= self.probTable.loc[key + '_' +
str(test_instance.loc[key]), outcome]
                        results[outcome] *= self.outcomeProb[outcome]
                  labels.append(max(results.items(), key=itemgetter(1))[0])
            return list(enumerate(labels))



print("Lab Question Part - A --------------------------------------------------\
n")

train_data = pd.read_csv("train_A.csv")
print("\nThe training data looks like :")
pprint(train_data)

target = "Buys_computer"

test_data = pd.read_csv("test_A.csv")
print("\nThe testing data looks like :")
pprint(test_data)
```

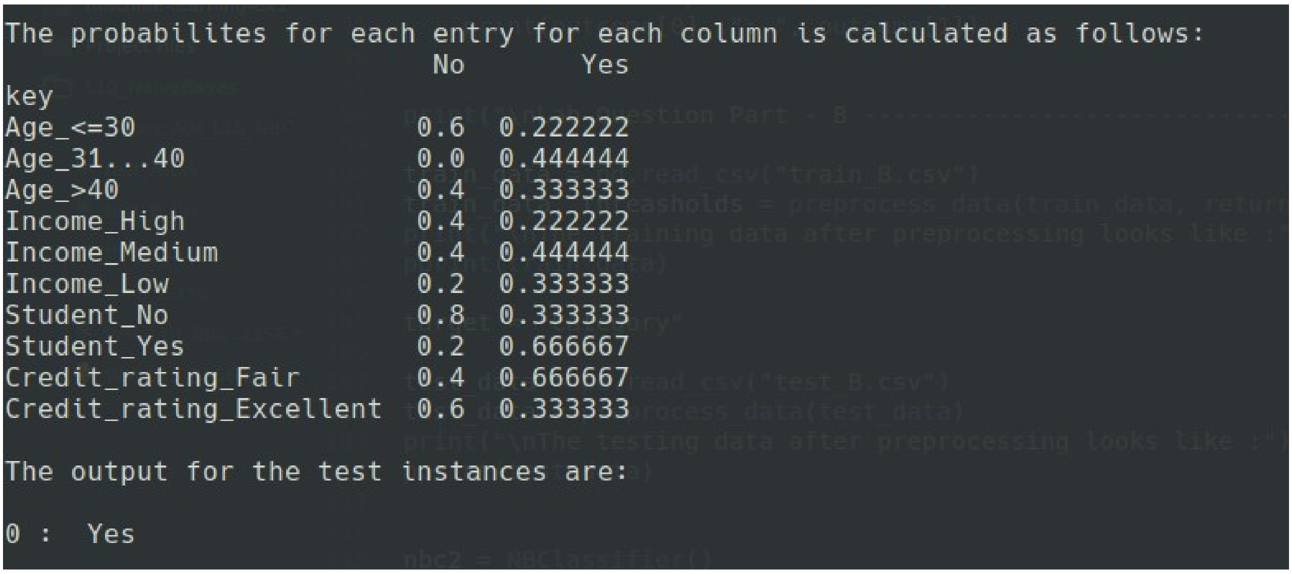
```
nbc = NBClassifier()
nbc.load_data(train_data, target)
print("\nThe output for the test instances are: \n")
for outcome in nbc.predict(test_data):
      print(outcome[0], ": ", outcome[1])


print("\nLab Question Part - B
--------------------------------------------------\n")

train_data = pd.read_csv("train_B.csv")
train_data, threasholds = preprocess_data(train_data, return_thresholds=True)
print("\nThe training data after preprocessing looks like :")
pprint(train_data)

target = "Category"

test_data = pd.read_csv("test_B.csv")
test_data = preprocess_data(test_data)
print("\nThe testing data after preprocessing looks like :")
pprint(test_data)


nbc2 = NBClassifier()
nbc2.load_data(train_data, target)
print("\nThe output for the test instances are: \n")
for outcome in nbc2.predict(test_data):
      print(outcome[0], ": ", outcome[1])
```

Output:

Part A -

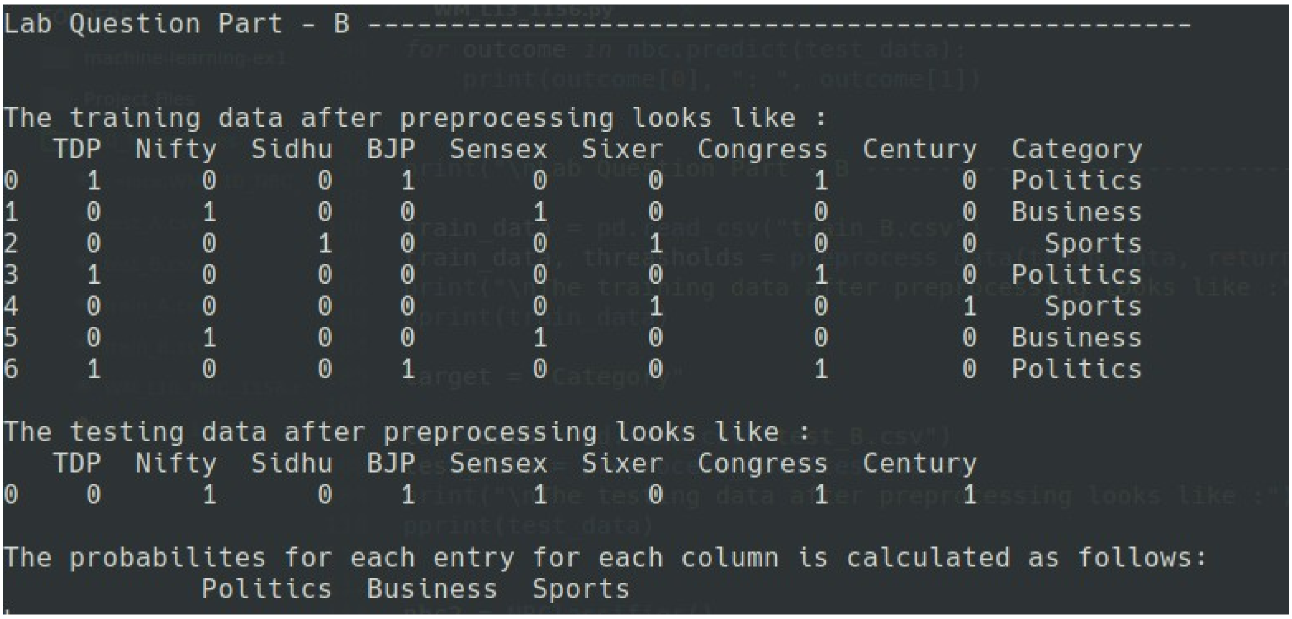


```
/media/anonymous/Work/Vit/Semester 5/WM/Lab/L10_NaiveBayes  echo 16BCE1156
16BCE1156
/media/anonymous/Work/Vit/Semester 5/WM/Lab/L10_NaiveBayes  python3 WM_L13_1156.py
Lab Question Part - A -------------------------------------------------


The training data looks like :
        Age  Income Student Credit_rating Buys_computer
0      <=30    High      No          Fair            No
1      <=30    High      No     Excellent            No
2   31...40    High      No          Fair           Yes
3       >40  Medium      No          Fair           Yes
4       >40     Low     Yes          Fair           Yes
5       >40     Low     Yes     Excellent            No
6   31...40     Low     Yes     Excellent           Yes
7      <=30  Medium      No          Fair            No
8      <=30     Low     Yes          Fair           Yes
9       >40  Medium     Yes          Fair           Yes
10     <=30  Medium     Yes     Excellent           Yes
11  31...40  Medium      No     Excellent           Yes
12  31...40    High     Yes          Fair           Yes
13      >40  Medium      No     Excellent            No

The testing data looks like :
    Age  Income Student Credit_rating
0  <=30  Medium     Yes          Fair

The probabilites for each entry for each column is calculated as follows:
```

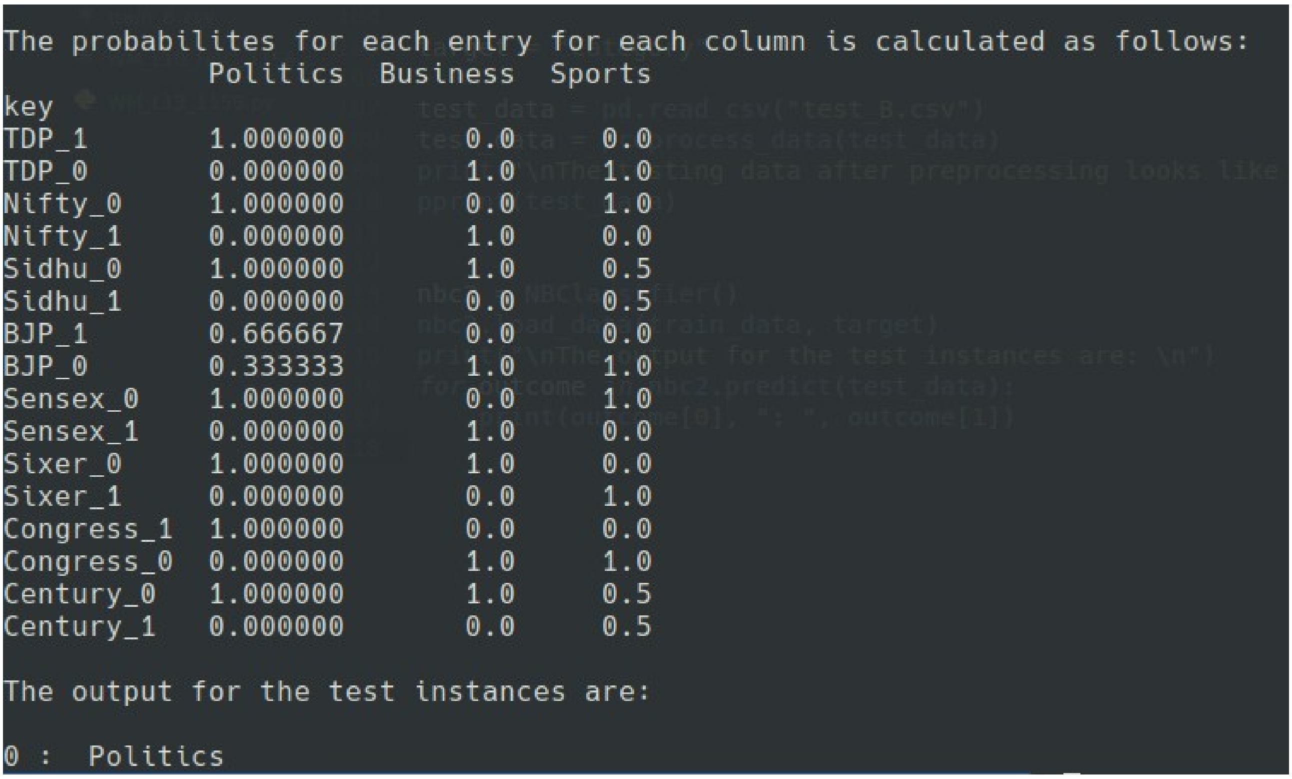


```
The probabilites for each entry for each column is calculated as follows:
                          No       Yes
key
Age_<=30                 0.6  0.222222
Age_31...40              0.0  0.444444
Age_>40                  0.4  0.333333
Income_High              0.4  0.222222
Income_Medium            0.4  0.444444
Income_Low               0.2  0.333333
Student_No               0.8  0.333333
Student_Yes              0.2  0.666667
Credit_rating_Fair       0.4  0.666667
Credit_rating_Excellent  0.6  0.333333

The output for the test instances are:

0 :  Yes
```

Part B -



```
Lab Question Part - B ---------------------------------------------

The training data after preprocessing looks like :
   TDP  Nifty  Sidhu  BJP  Sensex  Sixer  Congress  Century  Category
0    1      0      0    1       0      0         1        0  Politics
1    0      1      0    0       1      0         0        0  Business
2    0      0      1    0       0      1         0        0    Sports
3    1      0      0    0       0      0         1        0  Politics
4    0      0      0    0       0      1         0        1    Sports
5    0      1      0    0       1      0         0        0  Business
6    1      0      0    1       0      0         1        0  Politics

The testing data after preprocessing looks like :
   TDP  Nifty  Sidhu  BJP  Sensex  Sixer  Congress  Century
0    0      1      0    1       1      0         1        1

The probabilites for each entry for each column is calculated as follows:
            Politics  Business  Sports
```



```
The probabilites for each entry for each column is calculated as follows:
            Politics  Business  Sports
key
TDP_1       1.000000       0.0     0.0
TDP_0       0.000000       1.0     1.0
Nifty_0     1.000000       0.0     1.0
Nifty_1     0.000000       1.0     0.0
Sidhu_0     1.000000       1.0     0.5
Sidhu_1     0.000000       0.0     0.5
BJP_1       0.666667       0.0     0.0
BJP_0       0.333333       1.0     1.0
Sensex_0    1.000000       0.0     1.0
Sensex_1    0.000000       1.0     0.0
Sixer_0     1.000000       1.0     0.0
Sixer_1     0.000000       0.0     1.0
Congress_1  1.000000       0.0     0.0
Congress_0  0.000000       1.0     1.0
Century_0   1.000000       1.0     0.5
Century_1   0.000000       0.0     0.5

The output for the test instances are:

0 :  Politics
```