# Assignment #1

Submitted by


Alvi Mahadi V00912845

Rizwana Rahman V00866797


## Question #1

Construct the root and the first level of a decision tree for the contact lenses data. Use the ID3 algorithm. Show the details of your construction. Then, check your solution with Weka.

## Answer #1

**Contact Lens Data**

| age | spectacle-prescrip | astigmatism | tear-prod-rate | contact-lenses |
|---|---|---|---|---|
| young | myope | no | reduced | none |
| young | myope | no | normal | soft |
| young | myope | yes | reduced | none |
| young | myope | yes | normal | hard |
| young | hypermetrope | no | reduced | none |
| young | hypermetrope | no | normal | soft |
| young | hypermetrope | yes | reduced | none |
| young | hypermetrope | yes | normal | hard |
| pre-presbyopic | myope | no | reduced | none |
| pre-presbyopic | myope | no | normal | soft |
| pre-presbyopic | myope | yes | reduced | none |
| pre-presbyopic | myope | yes | normal | hard |
| pre-presbyopic | hypermetrope | no | reduced | none |
| pre-presbyopic | hypermetrope | no | normal | soft |
| pre-presbyopic | hypermetrope | yes | reduced | none |
| pre-presbyopic | hypermetrope | yes | normal | none |
| presbyopic | myope | no | reduced | none |
| presbyopic | myope | no | normal | none |
| presbyopic | myope | yes | reduced | none |
| presbyopic | myope | yes | normal | hard |
| presbyopic | hypermetrope | no | reduced | none |
| presbyopic | hypermetrope | no | normal | soft |
| presbyopic | hypermetrope | yes | reduced | none |
| presbyopic | hypermetrope | yes | normal | none |

### Which Attribute to select?


### Attribute "age"

$age\ =\ young$

$info([4,2,2]) = entropy(4/8,\ 2/8,\ 2/8) = -\,4/8\log(4/8)\ -\ 2/8\log(2/8)\ -\ 2/8\log(2/8)\ =\ 1.5$

$age\ =\ pre-presbyopic$

$info([5,2,1]) = entropy(5/8,\ 2/8,\ 1/8) = -\,5/8\log(5/8) - 2/8\log(2/8) - 1/8\log(1/8) = 1.23$

$age = presbyopic$

$info([6,1,1]) = entropy(6/8,\ 1/8,\ 1/8) =-\,6/8\log(6/8) - 1/8\log(1/8) - 1/8\log(1/8) = 1.06$

Expected info:

$info([4,2,2],[5,2,1],[6,1,1]) = 1.5 * (8/24) + 1.23 * (8/24) + 1.06 * (8/24) = 1.29$

**Attribute "spectacle-prescrip"**

$spectacle - prescrip = myope$

$info([7,2,3]) = entropy(7/12,\ 2/12,\ 3/12) =- 7/12 \log(7/12)\ -\ 2/12 \log(2/12)\ -\ 3/12 \log(3/12)\ =\ 1.38$

$spectacle - prescrip = hypermetrope$

$info([8,3,1]) = entropy(8/12,\ 3/12,\ 1/12) =- 8/12 \log(8/12)\ -\ 3/12 \log(3/12)\ -\ 1/12 \log(1/12)\ =\ 1.19$

Expected info:

$info([7,2,3],[8,3,1]) = 1.38 * (12/24) + 1.19 * (12/24) = 1.235$

**Attribute "astigmatism"**

$astigmatism = yes$

$info([8,0,4]) = entropy(8/12,\ 0/12,\ 4/12) =- 8/12 \log(8/12)\ -\ 0/12 \log(0/12)\ -\ 4/12 \log(4/12)\ =\ 0.918$

$astigmatism = no$

$info([7,5,0]) = entropy(7/12,\ 5/12,\ 0/12) =- 7/12 \log(7/12)\ -\ 5/12 \log(5/12)\ -\ 0/12 \log(0/12)\ =\ 0.98$

Expected info:

$info([8,0,4],[7,5,0]) = 0.918 * (12/24) + 0.98 * (12/24) =\ 0.95$

**Attribute "tear-prod-rate"**

$tear - prod - rate = reduced$

$info([12,0,0]) = entropy(12/12,\ 0/12,\ 0/12) =- 12/12 \log(12/12)\ -\ 0/12 \log(0/12)\ -\ 0/12 \log(0/12)\ =\ 0$

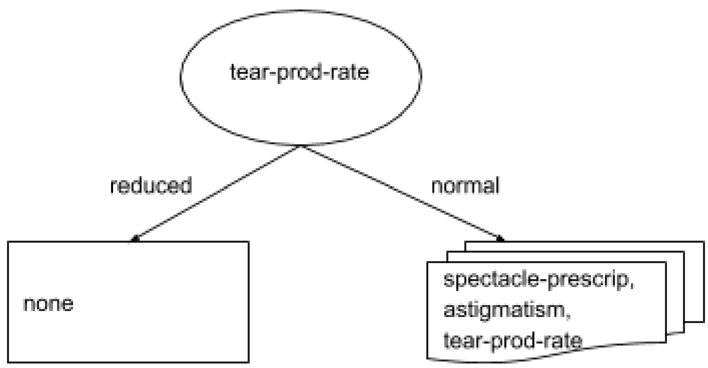
$tear - prod - rate = normal$

$info([3,5,4]) = entropy(3/12,\ 5/12,\ 7/12) =- 3/12 \log(3/12)\ -\ 5/12 \log(5/12)\ -\ 4/12 \log(4/12)\ =\ 1.55$

Expected info:

$info([12,0,0],[3,5,4]) = 0 * (12/24) + 1.55 * (12/24) =\ 0.775$

We can observe from the above entropy calculation that **Attribute "tear-prod-rate"** has the lowest average entropy. So **Attribute "tear-prod-rate"** will be the root.

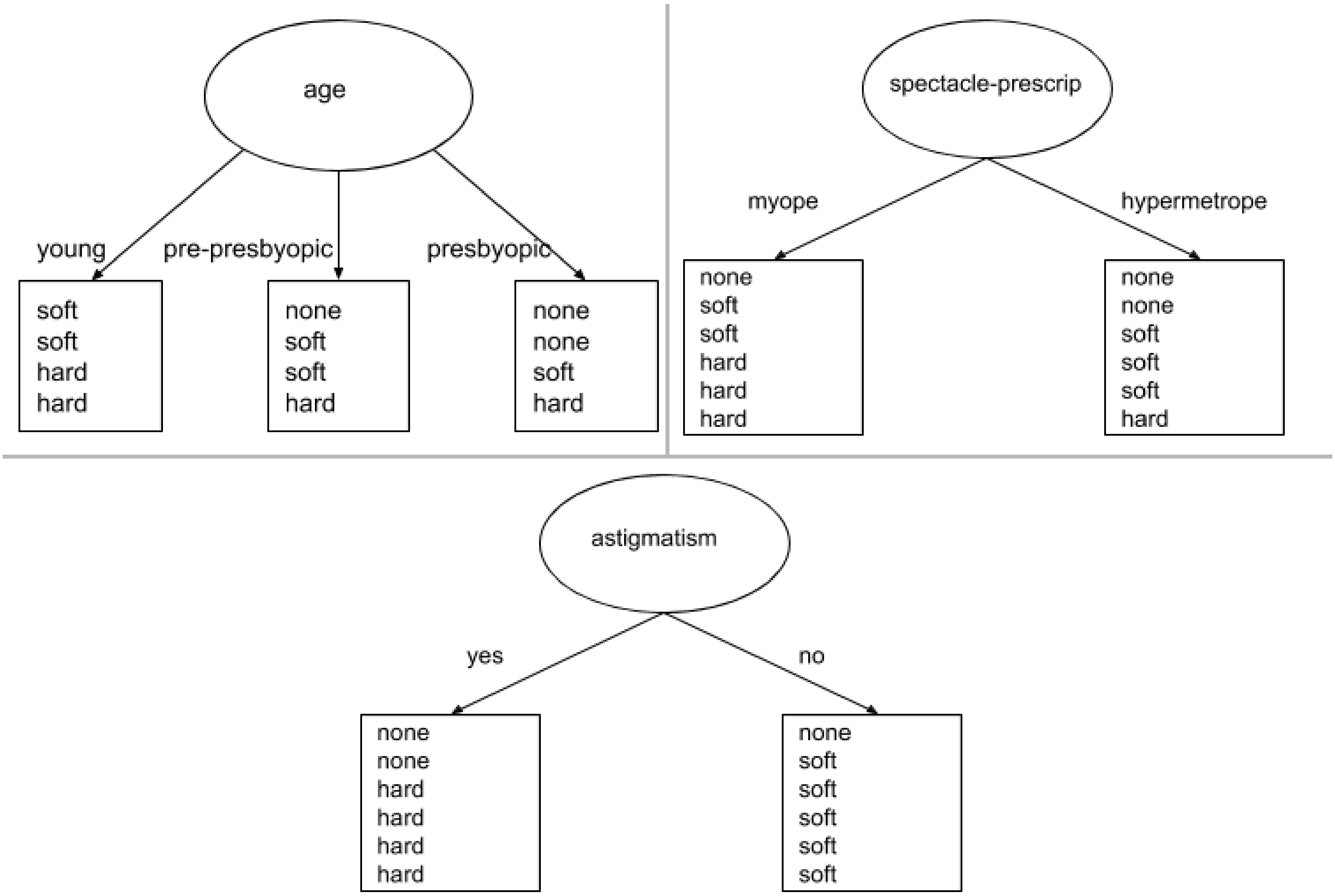



## Continue to split

| tear-prod-rate | age | spectacle-prescrip | astigmatism | contact-lenses |
|---|---|---|---|---|
| normal | young | myope | no | soft |
| normal | young | myope | yes | hard |
| normal | young | hypermetrope | no | soft |
| normal | young | hypermetrope | yes | hard |
| normal | pre-presbyopic | myope | no | soft |
| normal | pre-presbyopic | myope | yes | hard |

| | | | | |
|---|---|---|---|---|
| normal | pre-presbyopic | hypermetrope | no | soft |
| normal | pre-presbyopic | hypermetrope | yes | none |
| normal | presbyopic | myope | no | none |
| normal | presbyopic | myope | yes | hard |
| normal | presbyopic | hypermetrope | no | soft |
| normal | presbyopic | hypermetrope | yes | none |

### Which Attribute to select?




### Attribute “age”

$age \ = \ young$

$info([0,2,2]) = entropy(0/4,\ 2/4,\ 2/4) = -\ 0/4\log(0/4)\ -\ 2/4\log(2/4)\ -\ 2/4\log(2/4)\ =\ 1.0$

$age \ = \ pre-presbyopic$

$info([1,2,1]) = entropy(1/4,\ 2/4,\ 1/4) = -\ 1/4\log(1/4) - 2/4\log(2/4) - 1/4\log(1/4) = 1.5$

$age = presbyopic$

$info([2,1,1]) = entropy(2/4,\ 1/4,\ 1/4) =-\ 2/4\log(6/8) - 1/4\log(1/8) - 1/4\log(1/8) = 1.70$

Expected info:

$info([0,2,2],[1,2,1],[2,1,1]) = 1.0 * (4/12) + 1.5 * (4/12) + 1.70 * (4/12) = 1.40$

### Attribute “spectacle-prescrip”

$spectacle - prescrip = myope$

$info([1,2,3]) = entropy(1/6,\ 2/6,\ 3/6) =-\ 1/6\log(1/6)\ -\ 2/6\log(2/6)\ -\ 3/6\log(3/6)\ =\ 1.46$

$spectacle - prescrip = hypermetrope$
$info([2,3,1]) = entropy(2/6,\ 3/6,\ 1/6) = -2/6\log(2/6)\ -\ 3/6\log(3/6)\ -\ 1/6\log(1/6)\ =\ 1.19$

Expected info:

$info([1,2,3],[2,3,1]) = 1.46 * (6/12) + 1.19 * (6/12) = 1.325$

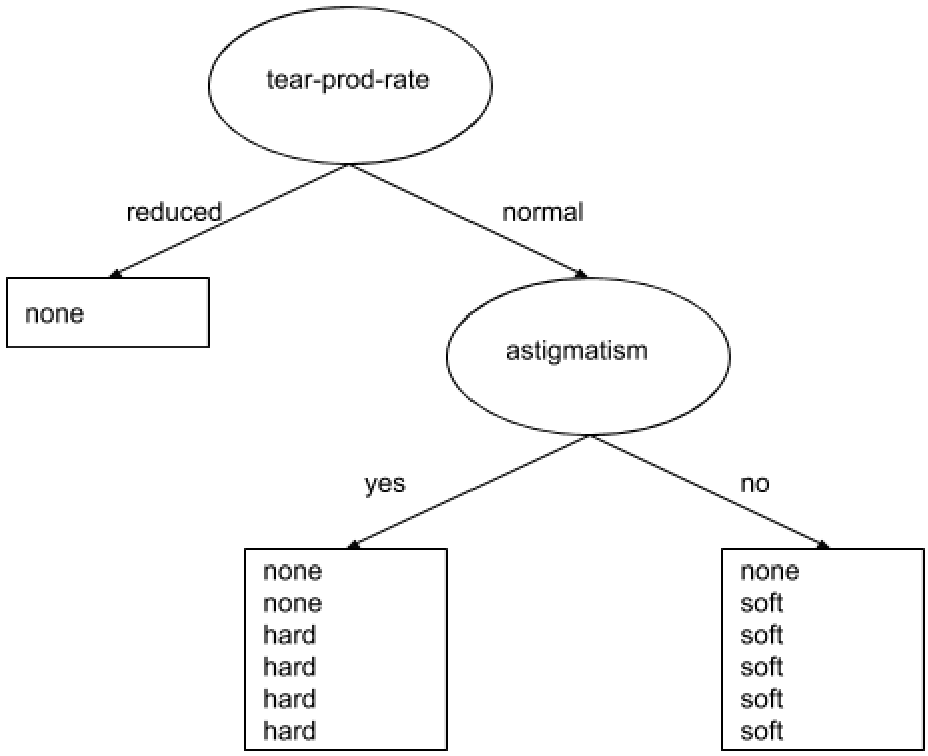
**Attribute "astigmatism"**

$astigmatism = yes$
$info([2,0,4]) = entropy(2/6,\ 0/6,\ 4/6) = -2/6\log(2/6)\ -\ 0/6\log(0/6)\ -\ 4/6\log(4/6)\ =\ 0.918$
$astigmatism = no$
$info([1,5,0]) = entropy(1/6,\ 5/6,\ 0/6) = -1/6\log(1/6)\ -\ 5/6\log(5/6)\ -\ 0/6\log(0/6)\ =\ 0.65$

Expected info:

$info([2,0,4],[1,5,0]) = 0.918 * (6/12) + 0.65 * (6/12) =\ 0.784$

We can observe from the above entropy calculation that **Attribute "astigmatism"** has the lowest average entropy. So **Attribute "astigmatism"** will be the root.
Tree so far:




## Observation from WEKA

After classify the Contact Lens Data with Weka, the following result is found.

```
Id3


tear-prod-rate = reduced: none
tear-prod-rate = normal
|  astigmatism = no
|  |  age = young: soft
|  |  age = pre-presbyopic: soft
|  |  age = presbyopic
|  |  |  spectacle-prescrip = myope: none
|  |  |  spectacle-prescrip = hypermetrope: soft
|  astigmatism = yes
|  |  spectacle-prescrip = myope: hard
|  |  spectacle-prescrip = hypermetrope
|  |  |  age = young: hard
|  |  |  age = pre-presbyopic: none
|  |  |  age = presbyopic: none

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          17               70.8333 %
Incorrectly Classified Instances         7               29.1667 %
Kappa statistic                          0.4381
Mean absolute error                      0.1944
Root mean squared error                  0.441
Relative absolute error                 51.4706 %
Root relative squared error            100.965  %
Total Number of Instances               24
```

From the classification result of Weba, it is observed that, attribute “tear-prod-rate” is considered as root and the “reduced” label yields decision “none”. The root of the second level is “astigmatism” which is also the second level root of the tree. So it is determined that, the structure of the tree made from manual entropy calculation matches the classification of Weka.

## Question #2

Construct two rules using PRISM for the weather data. Show the details of your construction. Then, check your solution with Weka.

## Answer #2

**Weather Data**

| outlook | temperature | humidity | windy | play |
|---|---|---|---|---|
| sunny | hot | high | FALSE | no |
| sunny | hot | high | TRUE | no |
| overcast | hot | high | FALSE | yes |
| rainy | mild | high | FALSE | yes |
| rainy | cool | normal | FALSE | yes |
| rainy | cool | normal | TRUE | no |
| overcast | cool | normal | TRUE | yes |
| sunny | mild | high | FALSE | no |
| sunny | cool | normal | FALSE | yes |
| rainy | mild | normal | FALSE | yes |
| sunny | mild | normal | TRUE | yes |
| overcast | mild | high | TRUE | yes |
| overcast | hot | normal | FALSE | yes |
| rainy | mild | high | TRUE | no |

## Rules of play = yes

### Rule we seek

```
if ?
      then play = yes
```

### Possible tests

| | |
|---|---|
| outlook = sunny | 2/5 |
| outlook = overcast | 4/4 |
| outlook = rainy | 3/5 |
| temperature = hot | 3/4 |
| temperature = mild | 4/6 |
| temperature = cool | 3/4 |
| humidity = high | 3/7 |
| humidity = normal | 6/7 |
| windy = TRUE | 3/6 |
| windy = FALSE | 6/8 |

### Rule with best test

```
if outlook = overcast
      then play = yes
```

## Other Rules of play = yes

### Rule we seek

```
if ?
      then play = yes
```

### Possible tests

| | |
|---|---|
| outlook = sunny | 2/5 |
| outlook = rainy | 3/5 |
| temperature = hot | 3/4 |
| temperature = mild | 4/6 |
| temperature = cool | 3/4 |
| humidity = high | 3/7 |
| humidity = normal | 6/7 |
| windy = TRUE | 3/6 |
| windy = FALSE | 6/8 |

**Rule with best test added**

```
if humidity = normal
      then play = yes
```

**Instances covered by modified rule**

| outlook | temperature | humidity | windy | play |
|---|---|---|---|---|
| rainy | cool | normal | FALSE | yes |
| overcast | cool | normal | TRUE | yes |
| sunny | cool | normal | FALSE | yes |
| rainy | mild | normal | FALSE | yes |
| sunny | mild | normal | TRUE | yes |
| overcast | hot | normal | FALSE | yes |
| rainy | cool | normal | TRUE | no |

This rule is not accurate. It covers 6 out of 7. So we need further refinement.

**Further Refinement**

```
if humidity = normal
      and ?
            then play = yes
```

**Possible tests**

| | |
|---|---|
| outlook = sunny | 2/2 |
| outlook = rainy | 2/3 |
| temperature = hot | 1/1 |
| temperature = mild | 2/2 |
| temperature = cool | 3/4 |
| windy = TRUE | 2/3 |
| windy = FALSE | 4/4 |

**Rule with best test added**

```
if humidity = normal
      and windy = FALSE
            then play = yes
```

**So the two rules are**

```
[1] if humidity = normal
          then play = yes
[2] if humidity = normal
          and windy = FALSE
               then play = yes
```

## Observation from WEKA

After classify the Contact Lens Data with Weka, the following result is found.

```
Prism rules
----------
If outlook = overcast then yes
If humidity = normal
   and windy = FALSE then yes
If temperature = mild
   and humidity = normal then yes
If outlook = rainy
   and windy = FALSE then yes
If outlook = sunny
   and humidity = high then no
If outlook = rainy
   and windy = TRUE then no


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances           9               64.2857 %
Incorrectly Classified Instances         3               21.4286 %
Kappa statistic                          0.4375
Mean absolute error                      0.25
Root mean squared error                  0.5
Relative absolute error                 59.2264 %
Root relative squared error            105.9121 %
UnClassified Instances                   2               14.2857 %
Total Number of Instances               14
```

From the classification of weka using the PRISM algorithm, the outlined rules are obtained. The rules that is the output of weka is similar to the rules from the result. This verifies that the results are correct.

## Question #3

Classify using Naïve Bayes method (on contact lenses data) the data item: pre-presbyopic, hypermetrope, yes, reduced, ? Then, check your solution with Weka.

## Answer #3

P(Contact-lenses=soft | E) = P(Age = pre-presbyopic | Contact-lenses=soft) *

P(Spectacle-prescrip = hypermetrope | Contact-lenses=soft) *

P(Astigmatism = yes | Contact-lenses=soft) *

P(Tear-prod-rate = reduced | Contact-lenses=soft) *

P(Contact-lenses=soft) / P(E)

= (2+1/5+3) * (3+1/5+2) * (0+1/5+2) * (0+1/5+2) * (5+1/24+3) / P(E) = (3/8) * (4/7) * (1/7) * (1/7) * (6/27) / P(E) = 0.00097 / P(E)

P(Contact-lenses=hard | E) = P(Age = pre-presbyopic | Contact-lenses= hard) *

P(Spectacle-prescrip = hypermetrope| Contact-lenses= hard) *

P(Astigmatism = yes | Contact-lenses= hard) *

P(Tear-prod-rate = reduced | Contact-lenses= hard) *

P(Contact-lenses= hard) / P(E)

= (1+1/4+3) * (1+1/4+2) * (4+1/4+2) * (0+1/4+2) * (4+1/24+3) / P(E) = (2/7) * (2/6) * (5/6) * (1/6) * (5/27) / P(E) = 0.00245 / P(E)

P(Contact-lenses=none | E) = P(Age = pre-presbyopic | Contact-lenses= none) *

P(Spectacle-prescrip = hypermetrope| Contact-lenses= none) *

P(Astigmatism = yes | Contact-lenses= none) *

P(Tear-prod-rate = reduced | Contact-lenses= none) * P(Contact-lenses= none) / P(E)

= (5+1/15+3) * (8+1/15+2) * (8+1/15+2) * (12+1/15+2) * (15+1/24+3) / P(E) = (6/18) * (9/17) * (9/17) * (13/17) * (16/27) / P(E) = 0.04234 / P(E)

P(Contact-lenses=soft | E) + P(Contact-lenses=hard | E) + P(Contact-lenses=none | E) = 1

0.00097 / P(E) + 0.00245 / P(E) + 0.04234 / P(E) = 1 P(E) = 0.00097 + 0.00245 + 0.04234

So,

P(Contact-lenses=soft | E) = 0.00097 / P(E) = 0.00097 / (0.00097 + 0.00245 + 0.04234) = 2.120 %

P(Contact-lenses=hard | E) = 0.00245 / P(E) = 0.00245 / (0.00097 + 0.00245 + 0.04234) = 5.354%

P(Contact-lenses=none | E) = 0.04234 / P(E) = 0.04234 / (0.00097 + 0.00245 + 0.04234) = 92.526%

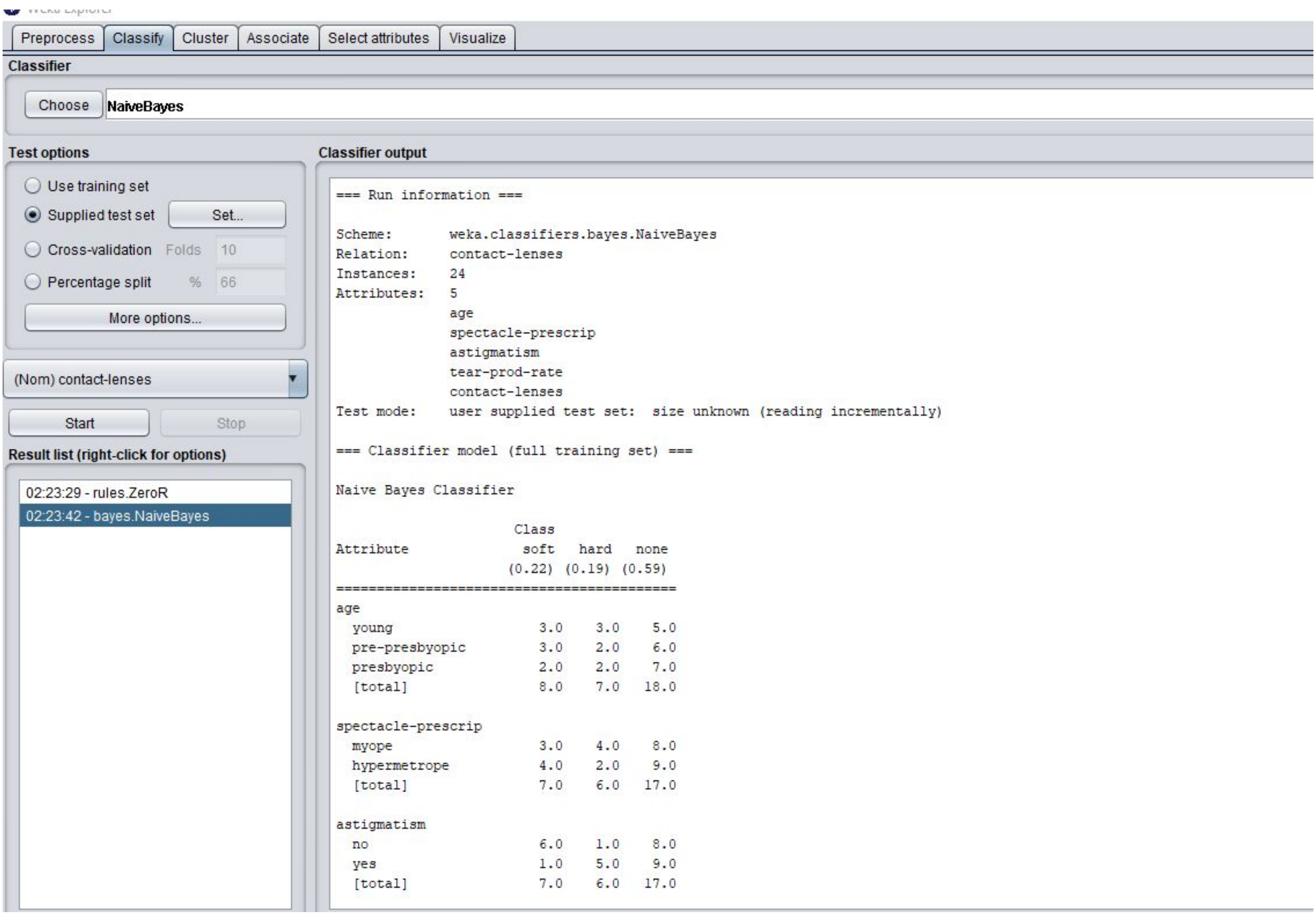
**Observation from WEKA**

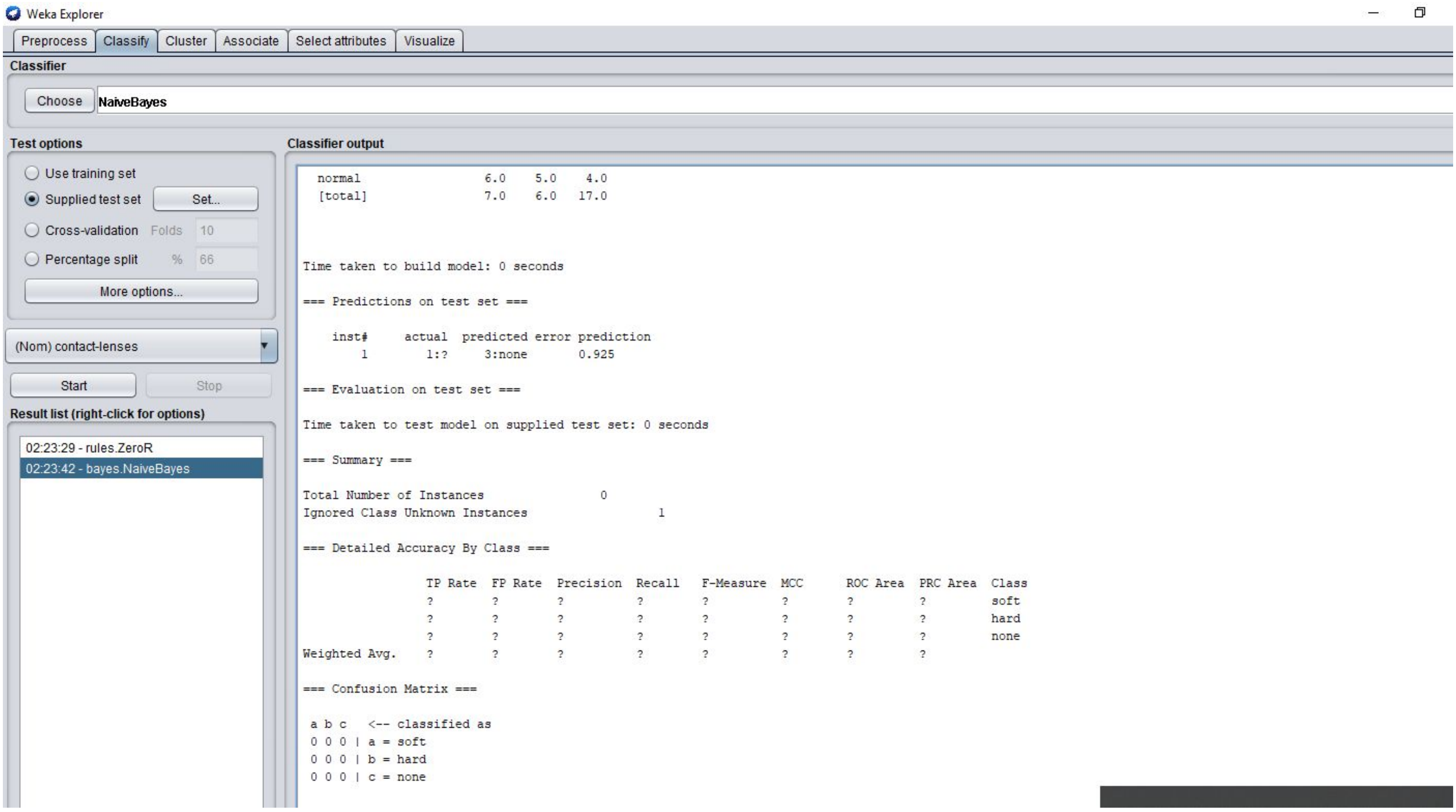
```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     contact-lenses
Instances:    24
Attributes:   5
              age
              spectacle-prescrip
              astigmatism
              tear-prod-rate
              contact-lenses
Test mode:    user supplied test set:  size unknown (reading incrementally)

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

Naive Bayes Classifier

                      Class
Attribute              soft   hard   none
                     (0.22) (0.19) (0.59)
==========================================
age
  young                 3.0    3.0    5.0
  pre-presbyopic        3.0    2.0    6.0
  presbyopic            2.0    2.0    7.0
  [total]               8.0    7.0   18.0

spectacle-prescrip
  myope                 3.0    4.0    8.0
  hypermetrope          4.0    2.0    9.0
  [total]               7.0    6.0   17.0

astigmatism
  no                    6.0    1.0    8.0
  yes                   1.0    5.0    9.0
  [total]               7.0    6.0   17.0

tear-prod-rate
  reduced               1.0    1.0   13.0
  normal                6.0    5.0    4.0
```

```
  [total]                7.0    6.0   17.0

Time taken to build model: 0 seconds

=== Predictions on test set ===

    inst#     actual  predicted error prediction
        1        1:?     3:none       0.925

=== Evaluation on test set ===

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

=== Summary ===

Total Number of Instances                0
Ignored Class Unknown Instances                  1

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
Class
                 ?        ?        ?          ?        ?          ?        ?         ?
soft
                 ?        ?        ?          ?        ?          ?        ?         ?
hard
                 ?        ?        ?          ?        ?          ?        ?         ?
none
Weighted Avg.    ?        ?        ?          ?        ?          ?        ?         ?

=== Confusion Matrix ===

 a b c   <-- classified as
 0 0 0 | a = soft
 0 0 0 | b = hard
 0 0 0 | c = none
```

According to the output of WEKA : test instance has been classified as none and probability is 92.5% which is same as our result.

## Screenshots



## Question #4

Implement a Naive Bayes classifier for text classification. This classifier will be used to classify fortune cookie messages into two classes: messages that predict what will happen in the future and messages that just contain a wise saying. We will label messages that predict what will happen in the future as class 1 and messages that contain a wise saying as class 0. For example,
"Never go in against a Sicilian when death is on the line" would be a message in class 0.
"You will get an A in SENG 474" would be a message in class 1.
You can use any language you wish. There are two sets of data files provided:
1. The training data:
   - traindata.txt: This is the training data consisting of fortune cookie messages.
   - trainlabels.txt: This file contains the class labels for the training data.
2. The testing data:
   - testdata.txt: This is the testing data consisting of fortune cookie messages.
   - testlabels.txt: This file contains the class labels for the testing data. These are

only used to determine the accuracy of the classifier.
Your results must be stored in a file called results.txt.
1. Run your classifier by training on traindata.txt and trainlabels.txt then testing on traindata.txt and trainlabels.txt. Report the accuracy in results.txt (along with a comment saying what files you used for the training and testing data). In this situation, you are training and testing on the same data. This is a sanity check: your accuracy should be very high i.e. > 90%
2. Run your classifier by training on traindata.txt and trainlabels.txt then testing on testdata.txt and testlabels.txt. Report the accuracy in results.txt (along with a comment saying what files you used for the training and testing data). We will not be letting you know beforehand what your performance on the test set should be.
Submit your source code and the results.txt file.

## Answer #4

```
import pandas as pd

traintexts = pd.read_csv("datasets/traindata.txt", sep="\n", header=None, names=['text'])
trainlabels = pd.read_csv("datasets/trainlabels.txt", sep="\n", header=None,
names=['value'])
traindata = pd.concat([traintexts, trainlabels], axis=1)

testtexts = pd.read_csv("datasets/testdata.txt", sep="\n", header=None, names=['text'])
testlabels = pd.read_csv("datasets/testlabels.txt", sep="\n", header=None, names=['value'])
testdata = pd.concat([testtexts, testlabels], axis=1)

total_document_length = len(traindata.axes[0])
result = set()
traindata.text.str.lower().str.split().apply(result.update)
total_distinct_words = len(result)

p_c = len(traindata.query("value == '1'").axes[0])/total_document_length
p_not_c = len(traindata.query("value == '0'").axes[0])/total_document_length

total_words_in_c = traindata.query("value == '1'").text.str.split().apply(len).sum()
total_words_in_not_c = traindata.query("value == '0'").text.str.split().apply(len).sum()

# Evaluate with traindata
f = open("datasets/resultlabels_traindata.txt","w+")
for text in traintexts.text:
```

```
        input_array = text.split()

        p_fp = p_c
        p_ws = p_not_c

        # For 1
        for w in input_array:
              p_fp *= ( (traindata.query("value == '1'").text.str.count(w).sum() + 1) /
                    (total_words_in_c + total_distinct_words) )

        # for 0
        for w in input_array:
              p_ws *= ( (traindata.query("value == '0'").text.str.count(w).sum() + 1) /
                    (total_words_in_not_c + total_distinct_words) )

        if( p_fp >= p_ws ):
              f.write("1\n")
        elif( p_ws > p_fp ):
              f.write("0\n")

f.close()

resultlabels = pd.read_csv("datasets/resultlabels_traindata.txt",
        sep="\n", header=None, names=['value'])

wrong_counter = 0
for i in range(len(trainlabels.axes[0])):
        if( trainlabels.value[i] != resultlabels.value[i] ):
              wrong_counter += 1

correctness = round( 1 - ( wrong_counter / len( trainlabels.axes[0] ) ), 4 ) * 100
r = open("result.txt","a")
r.write("Achieved "+str(correctness)+"%"+" correctness evaluating with traindata.txt\n")
r.close()

# Evaluate with testdata
f = open("datasets/resultlabels_testdata.txt","w+")
for text in testtexts.text:
        input_array = text.split()

        p_fp = p_c
        p_ws = p_not_c

        # For 1
        for w in input_array:
              p_fp *= ( (traindata.query("value == '1'").text.str.count(w).sum() + 1) /
                    (total_words_in_c + total_distinct_words) )

        # for 0
        for w in input_array:
              p_ws *= ( (traindata.query("value == '0'").text.str.count(w).sum() + 1) /
                    (total_words_in_not_c + total_distinct_words) )
```

```
	if( p_fp >= p_ws ):
		f.write("1\n")
	elif( p_ws > p_fp ):
		f.write("0\n")

f.close()

resultlabels = pd.read_csv("datasets/resultlabels_testdata.txt",
	sep="\n", header=None, names=['value'])

wrong_counter = 0
for i in range(len(testlabels.axes[0])):
	if( testlabels.value[i] != resultlabels.value[i] ):
		wrong_counter += 1

correctness = round( 1 - ( wrong_counter / len( testlabels.axes[0] ) ), 4 ) * 100
r = open("result.txt","a")
r.write("Achieved "+str(correctness)+"%"+" correctness evaluating with testdata.txt\n\n")
r.close()
```