**CS4445 B10 Homework 3 Part I Solution**

Yutao Wang

**1. Classification Rules**

**1.1.1** Instances that are under consideration when starting the construction of the 3rd rule (the other data instances are already covered by the first two rules):

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| 1 | 60 | 0.5 | tail | no | auto |
| 3 | 40 | 0.9 | head | no | auto |
| 4 | 65 | 0 | head | no | auto |
| 5 | 45 | 0.2 | head | yes | auto |
| 7 | 30 | 0.4 | head | yes | noauto |
| 8 | 90 | 0.6 | head | no | auto |
| 9 | 65 | 0.1 | head | no | auto |
| 11 | 25 | 0.6 | tail | yes | auto |
| 12 | 40 | 0.4 | tail | yes | noauto |
| 13 | 15 | 0.6 | tail | yes | noauto |
| 15 | 30 | 0.2 | head | yes | auto |
| 16 | 35 | 0.4 | head | yes | noauto |
| 17 | 70 | 0.6 | tail | no | auto |
| 18 | 20 | 0.5 | tail | yes | auto |
| 19 | 75 | 0.1 | tail | no | auto |

**1.1.2** To construct the rule "If ? then class = noauto", consider the p/t ratio table for above instances:

| attribute | value | p/t |
|---|---|---|
| STABILITY | <=50 | 4/9 |
| STABILITY | >50 | 0/6 |
| ERROR | <=0.3 | 0/5 |
| ERROR | >0.3 | 4/10 |
| ERROR | <=0.7 | 4/14 |
| ERROR | >0.7 | 0/1 |
| WIND | head | 2/8 |
| WIND | tail | 2/7 |
| **VISIBILITY** | **yes** | **4/8** |
| VISIBILITY | no | 0/7 |

The best antecedent for the rule here is "If VISIBILITY = yes, then class = noauto". This rule is not perfect yet. To construct the rule "If VISIBILITY = yes and ?, then class = noauto", consider the following instances:

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| 5 | 45 | 0.2 | head | yes | auto |
| 7 | 30 | 0.4 | head | yes | noauto |
| 11 | 25 | 0.6 | tail | yes | auto |
| 12 | 40 | 0.4 | tail | yes | noauto |
| 13 | 15 | 0.6 | tail | yes | noauto |
| 15 | 30 | 0.2 | head | yes | auto |
| 16 | 35 | 0.4 | head | yes | noauto |
| 18 | 20 | 0.5 | tail | yes | auto |

The p/t table for these instances:

| attribute | value | p/t |
|---|---|---|
| STABILITY | <=50 | 4/8 |
| ERROR | <=0.3 | 0/2 |
| **ERROR** | **>0.3** | **4/6** |
| ERROR | <=0.7 | 4/8 |
| WIND | head | 2/4 |
| WIND | tail | 2/4 |

The best rule here is "If VISIBILITY = yes and ERROR >0.3, then class = noauto". This rule is not perfect yet. To construct the rule "If VISIBILITY = yes and ERROR >0.3 and ?, then class = noauto", consider the following instances:

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| 7 | 30 | 0.4 | head | yes | noauto |
| 11 | 25 | 0.6 | tail | yes | auto |
| 12 | 40 | 0.4 | tail | yes | noauto |
| 13 | 15 | 0.6 | tail | yes | noauto |
| 16 | 35 | 0.4 | head | yes | noauto |
| 18 | 20 | 0.5 | tail | yes | auto |

The p/t table for these instances:

| attribute | value | p/t |
|---|---|---|
| STABILITY | <=50 | 4/6 |
| ERROR | <=0.7 | 4/6 |
| **WIND** | **head** | **2/2** |
| WIND | tail | 2/4 |

The best rule here is "If VISIBILITY = yes and ERROR >0.3 and WIND = head, then class = noauto".
The rule is now perfect (it has 100% accuracy) so we do not have to add anything else to this rule.

The 3rd rule for class = noauto is:
If VISIBILITY = yes and ERROR >0.3 and WIND = head then class = noauto

**1.2.1** To construct the 1st rule for Class =auto, we need to consider all the instances in the dataset:

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| 1 | 60 | 0.5 | tail | no | auto |
| 2 | 75 | 1 | head | yes | noauto |
| 3 | 40 | 0.9 | head | no | auto |
| 4 | 65 | 0 | head | no | auto |
| 5 | 45 | 0.2 | head | yes | auto |
| 6 | 80 | 0.1 | tail | yes | noauto |
| 7 | 30 | 0.4 | head | yes | noauto |
| 8 | 90 | 0.6 | head | no | auto |
| 9 | 65 | 0.1 | head | no | auto |
| 10 | 85 | 0.5 | head | yes | noauto |
| 11 | 25 | 0.6 | tail | yes | auto |
| 12 | 40 | 0.4 | tail | yes | noauto |
| 13 | 15 | 0.6 | tail | yes | noauto |
| 14 | 25 | 0.8 | head | yes | noauto |
| 15 | 30 | 0.2 | head | yes | auto |
| 16 | 35 | 0.4 | head | yes | noauto |
| 17 | 70 | 0.6 | tail | no | auto |
| 18 | 20 | 0.5 | tail | yes | auto |
| 19 | 75 | 0.1 | tail | no | auto |
| 20 | 80 | 0.2 | head | yes | noauto |
| 21 | 85 | 0.8 | tail | yes | noauto |
| 22 | 60 | 0.9 | tail | yes | noauto |

**1.2.2** To construct the rule "If ? then class = auto", consider the p/t ratio table for above instances:

| attribute | value | p/t |
|---|---|---|
| STABILITY | <=50 | 5/10 |
| STABILITY | >50 | 6/12 |
| ERROR | <=0.3 | 5/7 |
| ERROR | >0.3 | 6/15 |
| ERROR | <=0.7 | 10/17 |
| ERROR | >0.7 | 1/5 |
| WIND | head | 6/12 |
| WIND | tail | 5/10 |
| VISIBILITY | yes | 4/15 |
| **VISIBILITY** | **no** | **7/7** |

The best rule here is "If VISIBILITY = no, then class = auto". This rule is perfect. Hence, the 1st rule for class = auto is: If VISIBILITY = no, then class = auto

**1.3 Rule Pruning**

Given the rule: "**If VISIBILITY = yes and ERROR <=0.7 and ERROR >0.3 and STABILITY <=50 and WIND = tail then Class = noauto**", the instances that are covered by this rule are:

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| v4 | 10 | 0.6 | tail | yes | noauto |
| v5 | 40 | 0.5 | tail | yes | auto |
| v7 | 25 | 0.4 | tail | yes | auto |
| v9 | 20 | 0.6 | tail | yes | noauto |

# of positive instances: 2
# of negative instances: 2
(p-n)/(p+n) = 0/4 = 0.

The rule that would result from pruning the last predicate in the antecedent of the rule (i.e., WIND = tail) is: "**If VISIBILITY = yes and ERROR <=0.7 and ERROR >0.3 and STABILITY <=50 then Class = noauto**", the instances covered by this resulting rule are:

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| v11 | 40 | 0.5 | head | yes | noauto |
| v12 | 15 | 0.4 | head | yes | noauto |
| v4 | 10 | 0.6 | tail | yes | noauto |
| v5 | 40 | 0.5 | tail | yes | auto |
| v7 | 25 | 0.4 | tail | yes | auto |
| v9 | 20 | 0.6 | tail | yes | noauto |

# of positive instances: 4
# of negative instances: 2
(p-n)/(p+n) = 2/6 = 0.3333.
Since 0.3333>0, we prune WIND = tail.

Now consider the rule that would result from pruning the last predicate in the antecedent of the rule (i.e., STABILITY <= 50): "**If VISIBILITY = yes and ERROR <=0.7 and ERROR >0.3 then Class = noauto**", the instances covered by this rule are:

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| v11 | 40 | 0.5 | head | yes | noauto |
| v12 | 15 | 0.4 | head | yes | noauto |
| v2 | 80 | 0.6 | tail | yes | noauto |
| v4 | 10 | 0.6 | tail | yes | noauto |
| v5 | 40 | 0.5 | tail | yes | auto |
| v6 | 80 | 0.6 | tail | yes | noauto |
| v7 | 25 | 0.4 | tail | yes | auto |
| v8 | 80 | 0.6 | tail | yes | auto |
| v9 | 20 | 0.6 | tail | yes | noauto |

# of positive instances: 6
# of negative instances: 3
(p-n)/(p+n) = 3/9 = 0.3333.
Since 0.3333 = 0.3333, we do not prune the last predicate of this rule.

Now, consider the rule that would result from pruning the last two predicates in the antecedent of the rule (i.e., ERROR >0.3 and STABILITY <= 50): "**If VISIBILITY = yes and ERROR <=0.7 then Class = noauto**", the instances covered by this rule are:

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| v11 | 40 | 0.5 | head | yes | noauto |
| v12 | 15 | 0.4 | head | yes | noauto |
| v2 | 80 | 0.6 | tail | yes | noauto |
| v4 | 10 | 0.6 | tail | yes | noauto |
| v5 | 40 | 0.5 | tail | yes | auto |
| v6 | 80 | 0.6 | tail | yes | noauto |
| v7 | 25 | 0.4 | tail | yes | auto |
| v8 | 80 | 0.6 | tail | yes | auto |
| v9 | 20 | 0.6 | tail | yes | noauto |

# of positive instances: 6
# of negative instances: 3
(p-n)/(p+n) = 3/9 = 0.3333.
Since 0.3333 = 0.3333, we do not prune the (joint) predicates ERROR >0.3 and STABILITY <= 50 .

Now, consider the rule that would result from pruning the last three predicates in the antecedent of the rule (i.e., ERROR <= 0.7 and ERROR >0.3 and STABILITY <= 50): "**If VISIBILITY = yes then Class = noauto**", the instances covered by this rule are:

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| v11 | 40 | 0.5 | head | yes | noauto |
| v12 | 15 | 0.4 | head | yes | noauto |
| v2 | 80 | 0.6 | tail | yes | noauto |
| v4 | 10 | 0.6 | tail | yes | noauto |
| v5 | 40 | 0.5 | tail | yes | auto |
| v6 | 80 | 0.6 | tail | yes | noauto |
| v7 | 25 | 0.4 | tail | yes | auto |
| v8 | 80 | 0.6 | tail | yes | auto |
| v9 | 20 | 0.6 | tail | yes | noauto |

# of positive instances: 6
# of negative instances: 3
(p-n)/(p+n) = 3/9 = 0.3333.
Since 0.3333 = 0.3333, we do not prune the (joint) predicates ERROR <= 0.7 and ERROR >0.3 and STABILITY <= 50.

Now, consider the rule that would result from pruning the last four predicates in the antecedent of the rule (i.e., VISIBILITY = yes and ERROR <= 0.7 and ERROR >0.3 and STABILITY <= 50):
"**If <nothing> then Class = noauto**", the instances covered by this rule are (all the instances in the dataset):

| INSTANCE | STABILITY | ERROR | WIND | VISIBILITY | CLASS |
|---|---|---|---|---|---|
| v1 | 35 | 0.1 | head | no | auto |
| v2 | 80 | 0.6 | tail | yes | noauto |
| v3 | 35 | 0.1 | head | no | auto |
| v4 | 10 | 0.6 | tail | yes | noauto |
| v5 | 40 | 0.5 | tail | yes | auto |
| v6 | 80 | 0.6 | tail | yes | noauto |
| v7 | 25 | 0.4 | tail | yes | auto |
| v8 | 80 | 0.6 | tail | yes | auto |
| v9 | 20 | 0.6 | tail | yes | noauto |
| v10 | 35 | 0.1 | head | no | auto |
| v11 | 40 | 0.5 | head | yes | noauto |
| v12 | 15 | 0.4 | head | yes | noauto |

# of positive instances: 6
# of negative instances: 6
(p-n)/(p+n) = 0/12 = 0.
Since 0<0.3333, so do not prune the (joint) attributes VISIBILITY = yes and ERROR <= 0.7 and ERROR >0.3 and STABILITY <= 50.

So the resulting rule is:
If VISIBILITY = yes
and ERROR <=0.7
and ERROR >0.3
and STABILITY <=50 then Class = noauto

**2 Instance-Based Learning**
**2.1 Variation 1**
**Matlab code:** see Appendix 1;
**Output:**
*Data Attributes: Original;*
*Distance metric: (Plain) Euclidean distance;*
*Voting method: Majority vote with no distance weighting;*
---
*Distances from instance 23 to instances 1~22 are:*
*1: 25.023189, 2: 40.022619, 3: 5.063596, 4: 30.000167, 5: 10.050373, 6: 45.022217, 7: 5.107837, 8: 55.002273, 9: 30.000000, 10: 50.011599, 11: 10.111874, 12: 5.204805, 13: 20.056171, 14: 10.074225, 15: 5.100000, 16: 1.044031, 17: 35.017853, 18: 15.071828, 19: 40.012498, 20: 45.011221, 21: 50.024894, 22: 25.052744,*
*4 nearest neighbors of instance 23 are instances: 16, 3, 15, 7,*
*Majority vote is 2 auto vs 2 noauto*
*The classification result is auto*
---
*Distances from instance 24 to instances 1~22 are:*
*1: 20.025234, 2: 5.114685, 3: 40.026116, 4: 15.078461, 5: 35.016568, 6: 0.500000, 7: 50.010399, 8: 10.099505, 9: 15.074813, 10: 5.100000, 11: 55.000000, 12: 40.000500, 13: 65.000000, 14: 55.009454, 15:*

*50.011599, 16: 45.011554, 17: 10.049876, 18: 60.000083, 19: 5.123475, 20: 1.077033, 21: 5.003998, 22: 20.002250,*
*4 nearest neighbors of instance 24 are instances: 6, 20, 21, 10,*
*Majority vote is 0 auto vs 4 noauto*
*The classification result is noauto*

**2.2 Variation 2**
**Matlab code:** see Appendix 2;
**Output:**
*Data Attributes: Original;*
*Distance metric: (Plain) Euclidean distance;*
*Voting method: Majority vote using distance weighting: wi = 1/di;*
*---*
*Distances from instance 23 to instances 1~22 are:*
*1: 25.023189, 2: 40.022619, 3: 5.063596, 4: 30.000167, 5: 10.050373, 6: 45.022217, 7: 5.107837, 8: 55.002273, 9: 30.000000, 10: 50.011599, 11: 10.111874, 12: 5.204805, 13: 20.056171, 14: 10.074225, 15: 5.100000, 16: 1.044031, 17: 35.017853, 18: 15.071828, 19: 40.012498, 20: 45.011221, 21: 50.024894, 22: 25.052744,*
*4 nearest neighbors of instance 23 are instances: 16, 3, 15, 7,*
*Majority vote is 3.935666e-001 auto vs 1.153604e+000 noauto*
*The classification result is noauto*
*---*
*Distances from instance 24 to instances 1~22 are:*
*1: 20.025234, 2: 5.114685, 3: 40.026116, 4: 15.078461, 5: 35.016568, 6: 0.500000, 7: 50.010399, 8: 10.099505, 9: 15.074813, 10: 5.100000, 11: 55.000000, 12: 40.000500, 13: 65.000000, 14: 55.009454, 15: 50.011599, 16: 45.011554, 17: 10.049876, 18: 60.000083, 19: 5.123475, 20: 1.077033, 21: 5.003998, 22: 20.002250,*
*4 nearest neighbors of instance 24 are instances: 6, 20, 21, 10,*
*Majority vote is 0 auto vs 3.324395e+000 noauto*
*The classification result is noauto*

**2.3 Variation 3**
**Matlab code:** see Appendix 3;
**Output:**
*Data Attributes: Preprocess STABILITY so that it ranges from 0 to 1;*
*Distance metric: (Plain) Euclidean distance;*
*Voting method: Majority vote with no distance weighting;*
*---*
*Distances from instance 23 to instances 1~22 are:*
*1: 1.127436, 2: 1.447220, 3: 0.802773, 4: 0.412311, 5: 1.013794, 6: 1.536229, 7: 1.046157, 8: 0.887568, 9: 0.400000, 10: 1.266667, 11: 1.505914, 12: 1.447220, 13: 1.523519, 14: 1.227916, 15: 1.007196, 16: 1.044031, 17: 1.211519, 18: 1.483240, 19: 1.133333, 20: 1.170470, 21: 1.713022, 22: 1.658647,*
*4 nearest neighbors of instance 23 are instances: 9, 4, 3, 8,*
*Majority vote is 4 auto vs 0 noauto*
*The classification result is auto*
*---*

*Distances from instance 24 to instances 1~22 are:*
*1: 1.039765, 2: 1.079094, 3: 1.540923, 4: 1.549193, 5: 1.173788, 6: 0.500000, 7: 1.218378, 8: 1.420485, 9: 1.513275, 10: 1.007196, 11: 0.733333, 12: 0.569600, 13: 0.866667, 14: 1.256096, 15: 1.266667, 16: 1.183216, 17: 1.008850, 18: 0.806226, 19: 1.120020, 20: 1.077033, 21: 0.210819, 22: 0.401386,*
*4 nearest neighbors of instance 24 are instances: 21, 22, 6, 12,*
*Majority vote is 0 auto vs 4 noauto*
*The classification result is noauto*

2.4 Variation 4
**Matlab code:** see Appendix 4;
**Output:**
*Data Attributes: Preprocess STABILITY so that it ranges from 0 to 1;*
*Distance metric: (Plain) Euclidean distance;*
*Voting method: Majority vote using distance weighting: wi = 1/di;*
*---*
*Distances from instance 23 to instances 1~22 are:*
*1: 1.127436, 2: 1.447220, 3: 0.802773, 4: 0.412311, 5: 1.013794, 6: 1.536229, 7: 1.046157, 8: 0.887568, 9: 0.400000, 10: 1.266667, 11: 1.505914, 12: 1.447220, 13: 1.523519, 14: 1.227916, 15: 1.007196, 16: 1.044031, 17: 1.211519, 18: 1.483240, 19: 1.133333, 20: 1.170470, 21: 1.713022, 22: 1.658647,*
*4 nearest neighbors of instance 23 are instances: 9, 4, 3, 8,*
*Majority vote is 7.297712e+000 auto vs 0 noauto*
*The classification result is auto*
*---*
*Distances from instance 24 to instances 1~22 are:*
*1: 1.039765, 2: 1.079094, 3: 1.540923, 4: 1.549193, 5: 1.173788, 6: 0.500000, 7: 1.218378, 8: 1.420485, 9: 1.513275, 10: 1.007196, 11: 0.733333, 12: 0.569600, 13: 0.866667, 14: 1.256096, 15: 1.266667, 16: 1.183216, 17: 1.008850, 18: 0.806226, 19: 1.120020, 20: 1.077033, 21: 0.210819, 22: 0.401386,*
*4 nearest neighbors of instance 24 are instances: 21, 22, 6, 12,*
*Majority vote is 0 auto vs 1.099040e+001 noauto*
*The classification result is noauto*

2.5 Variation 5
**Matlab code:** see Appendix 5;
**Output:**
*Data Attributes: Preprocess STABILITY so that it ranges from 0 to 1;*
*Distance metric: Weighted Euclidean distance;*
*Voting method: Majority vote using distance weighting: wi = 1/di;*
*---*
*Distances from instance 23 to instances 1~22 are:*
*1: 4.542271, 2: 3.221628, 3: 0.805536, 4: 0.574456, 5: 3.007583, 6: 5.474486, 7: 3.016436, 8: 1.151328, 9: 0.565685, 10: 3.169998, 11: 5.434662, 12: 5.417461, 13: 5.444467, 14: 3.086350, 15: 3.003146, 16: 3.014963, 17: 4.575539, 18: 5.430470, 19: 4.562772, 20: 3.119295, 21: 5.534337, 22: 5.487460,*
*4 nearest neighbors of instance 23 are instances: 9, 4, 3, 8,*
*Majority vote is 5.618515e+000 auto vs 0 noauto*
*The classification result is auto*
*---*

*Distances from instance 24 to instances 1~22 are:*
*1: 3.025264, 2: 4.518726, 3: 5.468902, 4: 5.448853, 5: 4.565693, 6: 0.500000, 7: 4.602053, 8: 5.411613, 9: 5.438750, 10: 4.502098, 11: 1.037090, 12: 0.780313, 13: 1.225652, 14: 4.622289, 15: 4.615072, 16: 4.583667, 17: 3.005920, 18: 1.135782, 19: 3.042842, 20: 4.517743, 21: 0.221108, 22: 0.481894,*
*4 nearest neighbors of instance 24 are instances: 21, 22, 6, 12,*
*Majority vote is 0 auto vs 9.879350e+000 noauto*
*The classification result is noauto*

Appendix 1: instance_based_learning_variation1.m

```
%output variation condition
fprintf(1,'%s\n','Data Attributes: Original;');
fprintf(1,'%s\n','Distance metric: (Plain) Euclidean distance;');
fprintf(1,'%s\n','Voting method: Majority vote with no distance weighting;');
fprintf(1,'%s\n','------------');
%read in data
inputfile = fopen('shuttle-landing-control.txt','r');
data = textscan(inputfile,'%d%d%f%s%s%s','delimiter',sprintf('\t'));
index_col = 1;
stability_col = 2;
error_col = 3;
wind_col = 4;
visibility_col = 5;
class_col = 6;
data{index_col}(23)=23;
data{stability_col}(23) = 35;
data{error_col}(23) = 0.1;
data{wind_col}{23} = 'head';
data{visibility_col}{23} = 'no';
data{index_col}(24) = 24;
data{stability_col}(24) = 80;
data{error_col}(24) = 0.6;
data{wind_col}{24} = 'tail';
data{visibility_col}{24} = 'yes';
%compute distance for instance 23, 24
distances23 = zeros(1,22);
distances24 = zeros(1,22);
for i = 1:22
    distances23(i) = sqrt(double(double((data{stability_col}(i)-
data{stability_col}(23))^2)+ ...
        (data{error_col}(i)-data{error_col}(23))^2+...
        (~isequal(data{wind_col}{i},data{wind_col}{23}))^2+...
        (~isequal(data{visibility_col}{i},data{visibility_col}{23}))^2));
    distances24(i) = sqrt(double(double((data{stability_col}(i)-
data{stability_col}(24)))^2+ ...
        (data{error_col}(i)-data{error_col}(24))^2+...
        (~isequal(data{wind_col}{i},data{wind_col}{24}))^2+...
        (~isequal(data{visibility_col}{i},data{visibility_col}{24}))^2));
end
index_distances = [double(data{index_col}(1:22)) distances23' distances24'];
fprintf(1,'%s\n','Distances from instance 23 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances23(i));
end
index_distances = sortrows(index_distances,2); %sort on distances23
fprintf(1,'\n%s','4 nearest neighbors of instance 23 are instances:');
```

```
auto_count = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    if(isequal(temp{1},'auto'))
        auto_count=auto_count+1; %count the vote for 'auto'
    end
end
class = 'noauto';
if(auto_count >= 4-auto_count)
    class = 'auto'; %if # of 'auto'>= # of 'noauto', class = 'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_count,' auto vs ',4-
auto_count,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
fprintf(1,'%s\n','------------');
fprintf(1,'%s\n','Distances from instance 24 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances24(i));
end
index_distances = sortrows(index_distances,3); %sort on distances24
fprintf(1,'\n%s','4 nearest neighbors of instance 24 are instances:');
auto_count = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    if(isequal(temp{1},'auto'));
        auto_count=auto_count+1; %count the vote for 'auto'
    end
end
class = 'noauto';
if(auto_count >= 4-auto_count)
    class = 'auto'; %if # of 'auto'>= # of 'noauto', class = 'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_count,' auto vs ',4-
auto_count,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
```

Appendix 2: instance_based_learning_variation2.m

```
%output variation condition
fprintf(1,'%s\n','Data Attributes: Original;');
fprintf(1,'%s\n','Distance metric: (Plain) Euclidean distance;');
fprintf(1,'%s\n','Voting method: Majority vote using distance weighting: wi =
1/di;');
fprintf(1,'%s\n','------------');
%read in data
inputfile = fopen('shuttle-landing-control.txt','r');
data = textscan(inputfile,'%d%d%f%s%s%s','delimiter',sprintf('\t'));
index_col = 1;
stability_col = 2;
error_col = 3;
wind_col = 4;
visibility_col = 5;
class_col = 6;
data{index_col}(23)=23;
data{stability_col}(23) = 35;
```

```
data{error_col}(23) = 0.1;
data{wind_col}{23} = 'head';
data{visibility_col}{23} = 'no';
data{index_col}(24) = 24;
data{stability_col}(24) = 80;
data{error_col}(24) = 0.6;
data{wind_col}{24} = 'tail';
data{visibility_col}{24} = 'yes';
%compute distance for instance 23, 24
distances23 = zeros(1,22);
distances24 = zeros(1,22);
for i = 1:22
    distances23(i) = sqrt(double(double((data{stability_col}(i)-
data{stability_col}(23))^2)+ ...
        (data{error_col}(i)-data{error_col}(23))^2+...
        (~isequal(data{wind_col}{i},data{wind_col}{23}))^2+...
        (~isequal(data{visibility_col}{i},data{visibility_col}{23}))^2));
    distances24(i) = sqrt(double(double((data{stability_col}(i)-
data{stability_col}(24)))^2+ ...
        (data{error_col}(i)-data{error_col}(24))^2+...
        (~isequal(data{wind_col}{i},data{wind_col}{24}))^2+...
        (~isequal(data{visibility_col}{i},data{visibility_col}{24}))^2));
end
index_distances = [double(data{index_col}(1:22)) distances23' distances24'];
fprintf(1,'%s\n','Distances from instance 23 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances23(i));
end
index_distances = sortrows(index_distances,2); %sort on distances23
fprintf(1,'\n%s','4 nearest neighbors of instance 23 are instances:');
auto_vote = 0; %initialize vote
noauto_vote = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    weight = 1/index_distances(i,2);
    if(isequal(temp{1},'auto'))
        auto_vote = auto_vote + weight; %adding weighted vote into total vote
    else
        noauto_vote = noauto_vote + weight;
    end
end
class = 'noauto';
if(auto_vote >= noauto_vote)
    class = 'auto'; %if weighted auto_vote >= weighted noauto_vote, class =
'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_vote,' auto vs
',noauto_vote,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
fprintf(1,'%s\n','------------');
fprintf(1,'%s\n','Distances from instance 24 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances24(i));
end
index_distances = sortrows(index_distances,3); %sort on distances24
fprintf(1,'\n%s','4 nearest neighbors of instance 24 are instances:');
```

```
auto_vote = 0; %initialize vote
noauto_vote = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    weight = 1/index_distances(i,3);
    if(isequal(temp{1},'auto'))
        auto_vote = auto_vote + weight; %adding weighted vote into total vote
    else
        noauto_vote = noauto_vote + weight;
    end
end
class = 'noauto';
if(auto_vote >= noauto_vote)
    class = 'auto'; %if weighted auto_vote >= weighted noauto_vote, class =
'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_vote,' auto vs
',noauto_vote,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
```

Appendix 3: instance_based_learning_variation3.m

```
%output variation condition
fprintf(1,'%s\n','Data Attributes: Preprocess STABILITY so that it ranges
from 0 to 1;');
fprintf(1,'%s\n','Distance metric: (Plain) Euclidean distance;');
fprintf(1,'%s\n','Voting method: Majority vote with no distance weighting;');
fprintf(1,'%s\n','------------');
%read in data
inputfile = fopen('shuttle-landing-control.txt','r');
data = textscan(inputfile,'%d%f%f%s%s%s','delimiter',sprintf('\t'));
index_col = 1;
stability_col = 2;
error_col = 3;
wind_col = 4;
visibility_col = 5;
class_col = 6;
data{index_col}(23)=23;
data{stability_col}(23) = 35;
data{error_col}(23) = 0.1;
data{wind_col}{23} = 'head';
data{visibility_col}{23} = 'no';
data{index_col}(24) = 24;
data{stability_col}(24) = 80;
data{error_col}(24) = 0.6;
data{wind_col}{24} = 'tail';
data{visibility_col}{24} = 'yes';
%preprocess STABILITY
max_stability = max(data{stability_col}(1:22));
min_stability = min(data{stability_col}(1:22));
data{stability_col}(1:24)=(data{stability_col}(1:24)-
min_stability)/(max_stability-min_stability);%process for both train and test
data
%compute distance for instance 23, 24
distances23 = zeros(1,22);
distances24 = zeros(1,22);
```

```
for i = 1:22
    distances23(i) = sqrt(double(double((data{stability_col}(i)-
data{stability_col}(23))^2)+ ...
        (data{error_col}(i)-data{error_col}(23))^2+...
        (~isequal(data{wind_col}{i},data{wind_col}{23}))^2+...
        (~isequal(data{visibility_col}{i},data{visibility_col}{23}))^2));
    distances24(i) = sqrt(double(double((data{stability_col}(i)-
data{stability_col}(24)))^2+ ...
        (data{error_col}(i)-data{error_col}(24))^2+...
        (~isequal(data{wind_col}{i},data{wind_col}{24}))^2+...
        (~isequal(data{visibility_col}{i},data{visibility_col}{24}))^2));
end
index_distances = [double(data{index_col}(1:22)) distances23' distances24'];
fprintf(1,'%s\n','Distances from instance 23 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances23(i));
end
index_distances = sortrows(index_distances,2); %sort on distances23
fprintf(1,'\n%s','4 nearest neighbors of instance 23 are instances:');
auto_vote = 0; %initialize vote
noauto_vote = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    if(isequal(temp{1},'auto'))
        auto_vote = auto_vote + 1; %adding unweighted vote into total vote
    else
        noauto_vote = noauto_vote + 1;
    end
end
class = 'noauto';
if(auto_vote >= noauto_vote)
    class = 'auto'; %if auto_vote >= noauto_vote, class = 'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_vote,' auto vs
',noauto_vote,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
fprintf(1,'%s\n','------------');
fprintf(1,'%s\n','Distances from instance 24 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances24(i));
end
index_distances = sortrows(index_distances,3); %sort on distances24
fprintf(1,'\n%s','4 nearest neighbors of instance 24 are instances:');
auto_vote = 0; %initialize vote
noauto_vote = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    if(isequal(temp{1},'auto'))
        auto_vote = auto_vote + 1; %adding unweighted vote into total vote
    else
        noauto_vote = noauto_vote + 1;
    end
end
class = 'noauto';
if(auto_vote >= noauto_vote)
```

```
    class = 'auto'; %if auto_vote >= noauto_vote, class = 'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_vote,' auto vs
',noauto_vote,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
```

Appendix 4: instance_based_learning_variation4.m

```
%output variation condition
fprintf(1,'%s\n','Data Attributes: Preprocess STABILITY so that it ranges
from 0 to 1;');
fprintf(1,'%s\n','Distance metric: (Plain) Euclidean distance;');
fprintf(1,'%s\n','Voting method: Majority vote using distance weighting: wi =
1/di;');
fprintf(1,'%s\n','------------');
%read in data
inputfile = fopen('shuttle-landing-control.txt','r');
data = textscan(inputfile,'%d%f%f%s%s%s','delimiter',sprintf('\t'));
index_col = 1;
stability_col = 2;
error_col = 3;
wind_col = 4;
visibility_col = 5;
class_col = 6;
data{index_col}(23)=23;
data{stability_col}(23) = 35;
data{error_col}(23) = 0.1;
data{wind_col}{23} = 'head';
data{visibility_col}{23} = 'no';
data{index_col}(24) = 24;
data{stability_col}(24) = 80;
data{error_col}(24) = 0.6;
data{wind_col}{24} = 'tail';
data{visibility_col}{24} = 'yes';
%preprocess STABILITY
max_stability = max(data{stability_col}(1:22));
min_stability = min(data{stability_col}(1:22));
data{stability_col}(1:24)=(data{stability_col}(1:24)-
min_stability)/(max_stability-min_stability);%process for both train and test
data
%compute distance for instance 23, 24
distances23 = zeros(1,22);
distances24 = zeros(1,22);
for i = 1:22
    distances23(i) = sqrt(double(double((data{stability_col}(i)-
data{stability_col}(23))^2)+ ...
        (data{error_col}(i)-data{error_col}(23))^2+...
        (~isequal(data{wind_col}{i},data{wind_col}{23}))^2+...
        (~isequal(data{visibility_col}{i},data{visibility_col}{23}))^2));
    distances24(i) = sqrt(double(double((data{stability_col}(i)-
data{stability_col}(24)))^2+ ...
        (data{error_col}(i)-data{error_col}(24))^2+...
        (~isequal(data{wind_col}{i},data{wind_col}{24}))^2+...
        (~isequal(data{visibility_col}{i},data{visibility_col}{24}))^2));
end
index_distances = [double(data{index_col}(1:22)) distances23' distances24'];
fprintf(1,'%s\n','Distances from instance 23 to instances 1~22 are:');
```

```
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances23(i));
end
index_distances = sortrows(index_distances,2); %sort on distances23
fprintf(1,'\n%s','4 nearest neighbors of instance 23 are instances:');
auto_vote = 0; %initialize vote
noauto_vote = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    weight = 1/index_distances(i,2);
    if(isequal(temp{1},'auto'))
        auto_vote = auto_vote + weight; %adding weighted vote into total vote
    else
        noauto_vote = noauto_vote + weight;
    end
end
class = 'noauto';
if(auto_vote >= noauto_vote)
    class = 'auto'; %if weighted auto_vote >= weighted noauto_vote, class =
'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_vote,' auto vs
',noauto_vote,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
fprintf(1,'%s\n','------------');
fprintf(1,'%s\n','Distances from instance 24 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances24(i));
end
index_distances = sortrows(index_distances,3); %sort on distances24
fprintf(1,'\n%s','4 nearest neighbors of instance 24 are instances:');
auto_vote = 0; %initialize vote
noauto_vote = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    weight = 1/index_distances(i,3);
    if(isequal(temp{1},'auto'))
        auto_vote = auto_vote + weight; %adding weighted vote into total vote
    else
        noauto_vote = noauto_vote + weight;
    end
end
class = 'noauto';
if(auto_vote >= noauto_vote)
    class = 'auto'; %if weighted auto_vote >= weighted noauto_vote, class =
'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_vote,' auto vs
',noauto_vote,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
```

Appendix 5: instance_based_learning_variation5.m

```
%output variation condition
```

```
fprintf(1,'%s\n','Data Attributes: Preprocess STABILITY so that it ranges
from 0 to 1;');
fprintf(1,'%s\n','Distance metric: Weighted Euclidean distance;');
fprintf(1,'%s\n','Voting method: Majority vote using distance weighting: wi =
1/di;');
fprintf(1,'%s\n','------------');
%read in data
inputfile = fopen('shuttle-landing-control.txt','r');
data = textscan(inputfile,'%d%f%f%s%s%s','delimiter',sprintf('\t'));
index_col = 1;
stability_col = 2;
error_col = 3;
wind_col = 4;
visibility_col = 5;
class_col = 6;
data{index_col}(23)=23;
data{stability_col}(23) = 35;
data{error_col}(23) = 0.1;
data{wind_col}{23} = 'head';
data{visibility_col}{23} = 'no';
data{index_col}(24) = 24;
data{stability_col}(24) = 80;
data{error_col}(24) = 0.6;
data{wind_col}{24} = 'tail';
data{visibility_col}{24} = 'yes';
%preprocess STABILITY
max_stability = max(data{stability_col}(1:22));
min_stability = min(data{stability_col}(1:22));
data{stability_col}(1:24)=(data{stability_col}(1:24)-
min_stability)/(max_stability-min_stability);%process for both train and test
data
%compute distance for instance 23, 24
distances23 = zeros(1,22);
distances24 = zeros(1,22);
for i = 1:22
    distances23(i) = sqrt(double((2*(data{stability_col}(i)-
data{stability_col}(23))^2)+ ...
        (data{error_col}(i)-data{error_col}(23))^2+...
        (4.5*(~isequal(data{wind_col}{i},data{wind_col}{23})))^2+...
        (3*(~isequal(data{visibility_col}{i},data{visibility_col}{23})))^2));
    distances24(i) = sqrt(double((2*(data{stability_col}(i)-
data{stability_col}(24))^2)+ ...
        (data{error_col}(i)-data{error_col}(24))^2+...
        (4.5*(~isequal(data{wind_col}{i},data{wind_col}{24})))^2+...
        (3*(~isequal(data{visibility_col}{i},data{visibility_col}{24})))^2));
end
index_distances = [double(data{index_col}(1:22)) distances23' distances24'];
fprintf(1,'%s\n','Distances from instance 23 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances23(i));
end
index_distances = sortrows(index_distances,2); %sort on distances23
fprintf(1,'\n%s','4 nearest neighbors of instance 23 are instances:');
auto_vote = 0; %initialize vote
noauto_vote = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
```

```
    temp = data{class_col}(index_distances(i,1));
    weight = 1/index_distances(i,2);
    if(isequal(temp{1},'auto'))
        auto_vote = auto_vote + weight; %adding weighted vote into total vote
    else
        noauto_vote = noauto_vote + weight;
    end
end
class = 'noauto';
if(auto_vote >= noauto_vote)
    class = 'auto'; %if weighted auto_vote >= weighted noauto_vote, class =
'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_vote,' auto vs
',noauto_vote,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
fprintf(1,'%s\n','------------');
fprintf(1,'%s\n','Distances from instance 24 to instances 1~22 are:');
for i = 1:22
    fprintf(1,'%i: %f, ',i,distances24(i));
end
index_distances = sortrows(index_distances,3); %sort on distances24
fprintf(1,'\n%s','4 nearest neighbors of instance 24 are instances:');
auto_vote = 0; %initialize vote
noauto_vote = 0;
for i = 1:4
    fprintf(1,'%s%d,',' ',uint8(index_distances(i,1)));
    temp = data{class_col}(index_distances(i,1));
    weight = 1/index_distances(i,3);
    if(isequal(temp{1},'auto'))
        auto_vote = auto_vote + weight; %adding weighted vote into total vote
    else
        noauto_vote = noauto_vote + weight;
    end
end
class = 'noauto';
if(auto_vote >= noauto_vote)
    class = 'auto'; %if weighted auto_vote >= weighted noauto_vote, class =
'auto'
end
fprintf(1,'\n%s%d%s%d%s\n','Majority vote is ',auto_vote,' auto vs
',noauto_vote,' noauto');
fprintf(1,'%s%s\n','The classification result is ',class);
```