

<report on classifications of two dataset>

컴퓨터학과 #2017320233 김채령

-목차-

1. 실험 요약	… 2
2. Car Evaluation Dataset	
2.1 데이터에 대한 설명	… 2
2.2 실험 설계 및 방법과 진행 내용	… 2
2.3 비교 분석 결과	… 14
3. Air Quality Dataset	
3.1 데이터에 대한 설명	… 15
3.2 실험 설계 및 방법과 진행 내용	… 16
3.3 비교 분석 결과	… 23
4. 결론	… 24
참고 문헌	… 24

1. 실험 요약

UCI Machine Learning Repository에서 Car Evaluation과 Air Quality라는 두 개의 주어진 dataset을 학습하여 weka에서 classification 모델을 만들어본다. 각 dataset을 weka에서 불러올 수 있는 형식으로 저장하고 경우에 따라 dataset에 전처리를 해준다. ID3, C4.5, Naivebayes, Logistic Regression, MLP 알고리즘을 이용하여 학습하고 각 모델은 zeroR과 oneR을 baseline으로 비교한다. Instance가 충분하지 않으므로 본 실험에서는 별도의 test set을 생성하지 않고 training set만을 이용한다.

2. Car Evaluation Dataset

2.1 데이터에 대한 설명

- classification에 적합한 데이터이다.
- attribute의 속성이 nominal하다.
- 6개의 attributes가 있다.

Attribute	Info	Label #1	Label #2	Label #3	Label #4
buying	구매 가격	v-high	high	med	Low
maint	유지 비용	v-high	High	Med	Low
doors	문 개수	2	3	4	5-more
persons	수용가능 인원	2	4	more	(empty)
lug_boot	트렁크 크기	small	med	big	(empty)
safety	측정된 안전도	Low	med	high	(empty)

- 총 1728개의 instance가 있고, 각 attribute의 각 label에 동일한 개수의 instance가 분포한다.
- missing attribute values가 없고, instances가 완전히 attribute space를 커버한다.
- 아래 표는 4개의 classes와 number of instances per class(class distribution)를 보여준다.

Class	N	N[%]
unacc	1210	70.023%
acc	384	22.222%
good	69	3.993%
v-good	65	3.762%

-instance들이 unacc에 집중되어 분포한다.

2.2 실험 설계 및 방법과 진행 내용

<실험 설계>

- Weka에서 여러 종류의 classifier를 이용하여 dataset을 학습해보고, classifier를 비교해 본다.
- ID3, C4.5, NaiveBayes, Logistic Regression, MLP 이렇게 5가지의 classifier를 이용한다.
- zeroR과 oneR의 결과를 baseline으로 두고 비교한다.
- 따로 test set을 설정하지 않고 training set을 활용한다.
- 실행 결과창에서는 classifier model, summary, confusion matrix 이렇게 3가지 항목을 중심으로 결과를 확인한다.

<실험 방법 및 진행 내용>

<전처리>

- weka에서 불러올 수 있는 C4.5 형식으로 불러왔다.

<zeroR>

: 가장 많은 instance를 포함한 class만을 고려하여 class를 추측한다.

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

```
ZeroR predicts class value: unacc
```

```
Time taken to build model: 0 seconds
```

학습 결과 생성된 모델에서는 총 1700여개의 instances 중 약 70%정도가 unacc이라는 class에 속하기 때문에 unacc를 선택한다.

```
==== Summary ===
```

Correctly Classified Instances	1210	70.0231 %
Incorrectly Classified Instances	518	29.9769 %
Kappa statistic	0	
Mean absolute error	0.229	
Root mean squared error	0.3381	
Relative absolute error	100	%
Root relative squared error	100	%
Total Number of Instances	1728	

Summary 항목은 학습 정확도, 학습 오차 등의 통계적인 정보를 보여준다. 총 1728개의 instances를 가장 빈도가 높았던 unacc이라는 class만 고려하여 분류했기 때문에 원래 unacc에 속했던 1210개, 즉 70%정도의 instance들만 정확히 분류되었다. 따라서 나머지 3개의 class에 속했던 518개의 instance들은 부정확하게 분류된다. 즉, 가장 많은 instance를 가진 class 이외의 class에 대해서는 정확한 분류가 이루어지지 않고 있다. Kappa statistics는 단순히 random으로 추측했을 때보다 얼마나 classifier의 성능이 개선되었는 가를 보여주는데, 이 수치가 0인 것은 zeroR로 분류한 이 모델이 이 dataset을 효과적으로 분류하지 못한다는 근거가 된다.

```
==== Confusion Matrix ===
```

a	b	c	d	<- classified as
1210	0	0	0	a = unacc
384	0	0	0	b = acc
69	0	0	0	c = good
65	0	0	0	d = vgood

Confusion matrix는 실제 정답과 model이 예측한 정답을 보여준다. 앞서 언급했듯이 zeroR 모델에서는 가장 빈도수가 높았던 class, unacc만 고려하였기 때문에 나머지 세 class값에 속한 instance들은 모두 unacc로 분류되는 부정확함이 있음을 다시 한번 알 수 있다. 정리하자면, unacc를 선택한 zeroR 모델에서, 원래 unacc에 속했던 1210개의 instances를 제외한 나머지 3개의 class값에 속했던 instance들은 부정확하게 분류되었다.

<oneR>

: error rate가 가장 적은 attribute를 하나의 rule로 두고 이를 고려하여 분류한다.

```

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

buying:
    vhigh  -> unacc
    high   -> unacc
    med    -> unacc
    low    -> unacc
(1210/1728 instances correct)

```

학습 결과 만들어진 oneR 모델에서 채택된 attribute는 buying(구매 가격)임을 확인할 수 있고 해당하는 branch는 unacc class를 가리킨다. 여기서 buying이 채택되었다는 것은 이 attribute가 가장 적은 error rate를 초래한다는 것을 전제한다.

```
==== Summary ====
```

Correctly Classified Instances	1210	70.0231 %
Incorrectly Classified Instances	518	29.9769 %
Kappa statistic	0	
Mean absolute error	0.1499	
Root mean squared error	0.3871	
Relative absolute error	65.4574 %	
Root relative squared error	114.5023 %	
Total Number of Instances	1728	

6가지 attribute 중 buying을 채택한 이 oneR 모델은 1728개의 instance 중 1210개를 정확하게 분류하고, 이는 70%정도의 확률이다. 이 역시 unacc에 약 70%정도의 instance들이 집중해서 분포하므로 이와 같은 확률이 나타난다. Kappa statistic은 zeroR과 같이 0이므로 효과적인 분류 모델은 아니라고 할 수 있다.

```
==== Confusion Matrix ====
```

a	b	c	d	<-- classified as
1210	0	0	0	a = unacc
384	0	0	0	b = acc
69	0	0	0	c = good
65	0	0	0	d = vgood

이 모델에서는 attribute buying을 unacc에 branch로 가리켰기 때문에, oneR과 동일하게 unacc에 속했던 instance들만 원래의 class(unacc)으로 정확하게 분류되고 나머지 3개의 class로는 분류되지 않음을 확인할 수 있다.

<ID3>

: ID3 decision tree는 nominal한 attribute만 사용할 수 있는 알고리즘이다. Car Evaluation dataset은 모든 attribute의 속성이 nominal하기 때문에 따로 전처리 과정을 거칠 필요가 없다. ID3는 각 attribute의 entropy를 계산하고, entropy가 가장 적은 혹은, information gain이 가장 큰 attribute부터 이용하여 분할해 나가면서 decision tree를 만드는 기법이다.

ID3 decision tree는 기본적으로 사용할 수 없어서 패키지 관리자에서 따로 설치를 해주었다.

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

```
Id3
```

```
safety = low: unacc
safety = med
| persons = 2: unacc
| persons = 4
| | buying = vhigh
| | | maint = vhigh: unacc
| | | maint = high: unacc
| | | maint = med
| | | | lug_boot = small: unacc
| | | | lug_boot = med
| | | | | doors = 2: unacc
| | | | | doors = 3: unacc
| | | | | doors = 4: acc
| | | | | doors = 5more: acc
| | | | lug_boot = big: acc
| | | maint = low
| | | | lug_boot = small: unacc
| | | | lug_boot = med
| | | | | doors = 2: unacc
| | | | | doors = 3: unacc
| | | | | doors = 4: acc
| | | | | doors = 5more: acc
| | | | lug_boot = big: acc
| | buying = high
| | | lug_boot = small: unacc
| | | lug_boot = med
| | | | doors = 2: unacc
| | | | doors = 3: unacc
| | | | doors = 4
| | | | | maint = vhigh: unacc
| | | | | maint = high: acc
| | | | | maint = med: acc
| | | | | maint = low: acc
| | | | doors = 5more
| | | | | maint = vhigh: unacc
```

Classifier model 부분을 보면 ID3 알고리즘을 활용해서 엔트로피가 적은 순으로, information gain이 많은 attribute 순으로 branch를 전개해 나가고 있음을 알 수 있다.

```
== Summary ==
```

Correctly Classified Instances	1728	100	%
Incorrectly Classified Instances	0	0	%
Kappa statistic	1		
Mean absolute error	0		
Root mean squared error	0		
Relative absolute error	0	%	
Root relative squared error	0	%	
Total Number of Instances	1728		

우선 1728개의 instance들이 모두 정확하게 분류되었다. 무작위로 class 값을 추측했을 때의 값이 0이었던 kappa statistic 수치를 봐도 1로 크게 향상되었다. 이는 이 classifier가 아주 정확하게 분류를 수행한다는 것을 보여준다. 이는 다르게 말해서 이 dataset에 대해서 overfitting될 확률이 높다는 것을 의미한다. zeroR, oneR과 비교해서 정확도와 Kappa statistic 모두 높음을 확인할 수 있다.

==== Confusion Matrix ===

```
a    b    c    d    <-- classified as
1210  0    0    0 |   a = unacc
      0  384  0    0 |   b = acc
      0    0  69  0 |   c = good
      0    0    0  65 |   d = vgood
```

결과적으로, 이 classifier를 사용하면 주어진 dataset이 정확하게 분류되는 것을 다시 한번 확인할 수 있다. 원래 각 class에 속했던 instance들이 하나도 빠짐없이 정확하게 해당하는 class로 분류되었다.

<C4.5(J48) decision tree>

: ID3 알고리즘의 연장선이며 역시 decision tree를 만든다. 이 알고리즘은 이전의 ID3에서는 취급하지 못했던 numerical한 attribute의 속성들까지 다룬다. 불완전한 데이터를 처리하고 pruning(가지치기)으로 overfitting 문제를 해결할 수 있다.

주로 pruning과 관련된 option을 제공하여 준다.

-unpruned tree (minNumObj = 2)

먼저 pruning을 하지 않은 j48 classifier를 보기 위해서 unpruned 옵션을 True로 설정하고 나머지는 default값을 사용하였다.

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

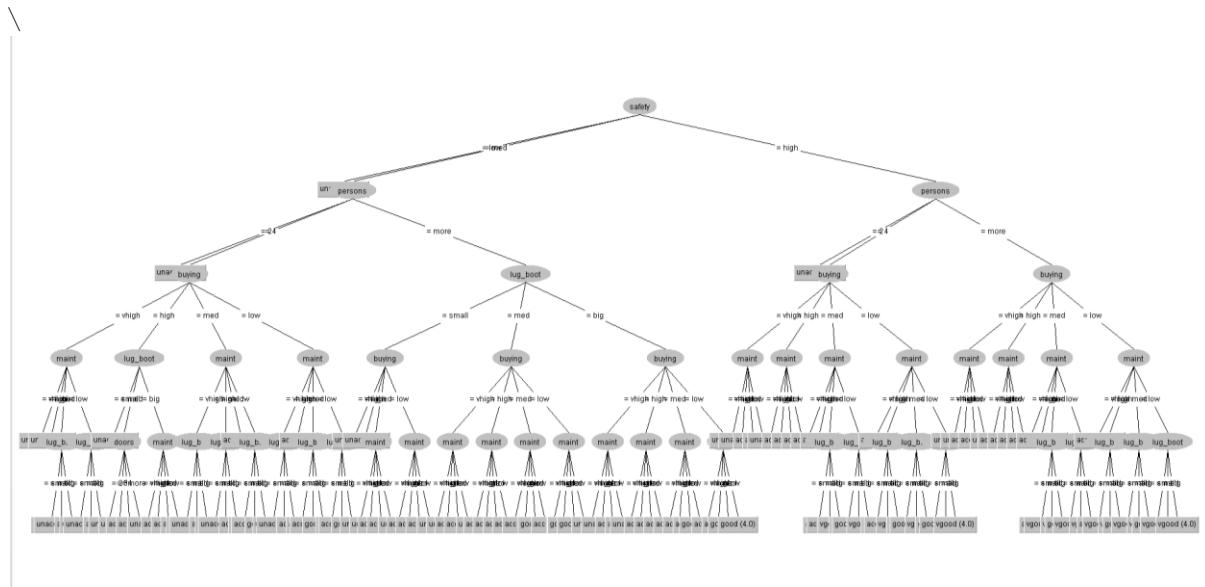
J48 unpruned tree
-----

safety = low: unacc (576.0)
safety = med
| persons = 2: unacc (192.0)
| persons = 4
|   | buying = vhigh
|   |   | maint = vhigh: unacc (12.0)
|   |   | maint = high: unacc (12.0)
|   |   | maint = med
|   |   |   | lug_boot = small: unacc (4.0)
|   |   |   | lug_boot = med: unacc (4.0/2.0)
|   |   |   | lug_boot = big: acc (4.0)
|   |   |   | maint = low
|   |   |   |   | lug_boot = small: unacc (4.0)
|   |   |   |   | lug_boot = med: unacc (4.0/2.0)
|   |   |   |   | lug_boot = big: acc (4.0)
|   |   |   | buying = high
|   |   |   |   | lug_boot = small: unacc (16.0)
|   |   |   |   | lug_boot = med
|   |   |   |   |   | doors = 2: unacc (4.0)
|   |   |   |   |   | doors = 3: unacc (4.0)
|   |   |   |   |   | doors = 4: acc (4.0/1.0)
|   |   |   |   |   | doors = 5more: acc (4.0/1.0)
|   |   |   |   |   | lug_boot = big
|   |   |   |   |   |   | maint = vhigh: unacc (4.0)
|   |   |   |   |   |   | maint = high: acc (4.0)
|   |   |   |   |   |   | maint = med: acc (4.0)
|   |   |   |   |   |   | maint = low: acc (4.0)
|   |   |   |   | buying = med
|   |   |   |   |   | maint = vhigh
|   |   |   |   |   |   | lug_boot = small: unacc (4.0)
|   |   |   |   |   |   | lug_boot = med: unacc (4.0/2.0)
|   |   |   |   |   |   | lug_boot = big: acc (4.0)
```

Number of Leaves : 134

Size of the tree : 186

이렇게 pruning도 하지 않고 최소 노드의 개수도 2개로 상대적으로 적게 두었을 때, leaf의 개수는 134, tree의 크기는 186으로 큰 tree가 만들어짐을 아래에서 확인할 수 있다.



==== Summary ====

Correctly Classified Instances	1666	96.412 %
Incorrectly Classified Instances	62	3.588 %
Kappa statistic	0.9223	
Mean absolute error	0.0239	
Root mean squared error	0.1094	
Relative absolute error	10.4462 %	
Root relative squared error	32.3444 %	
Total Number of Instances	1728	

Pruning을 하지 않고 minNumObj를 2(default value)로 설정한 j48 decision tree는 ID3만큼은 아니지만 높은 성능을 보인다. 총 1728개의 instance 중 1666개, 즉 약 96%의 확률로 정확하게 분류하였으며 Kappa statistic도 0과는 많이 떨어진 1에 가까운 값을 보여준다. zeroR, oneR과 비교해서 정확도와 Kappa statistic 모두 높음을 확인할 수 있다. ID3와 결과가 차이가 나는 이유는 pruning은 하지 않았지만 minNumObj를 2로 설정해주었기 때문에 ID3보다 general하게 되었기 때문임을 생각해볼 수 있다. 이는 후에 따로 test set을 두었을 때, overfitting 문제를 어느정도 해결할 수 있을 것이다. option을 unpruned, minNumObj = 1로 설정을 하면 ID3와 동일한 정확도를 얻을 수 있다.

==== Confusion Matrix ====

a	b	c	d	<- classified as
1182	25	3	0	a = unacc
10	369	3	2	b = acc
0	6	60	3	c = good
0	4	6	55	d = vgood

Confusion matrix를 보아도 각 class별로 높은 확률로 정확하게 분류되고 있음을 알 수 있다. 세로로 이 matrix를 읽으면 class unacc에는 1192개의 instance가, acc에는 404개가, good에는 72

개가, 그리고 vgood에는 60개가 분류되었으며, 그 중에서 unacc에는 10개의 instance가, acc에는 35개가, good에는 12개가, vgood에는 5개의 instance들이 잘못 분류되었음을 알 수 있다. 가로로 보면 unacc에 속하는 1210개의 instance 중 1182개는 정확히 unacc로 분류되고 나머지 instance들은 acc, good, vgood의 class들로 각 25개, 3개, 0개씩 잘못 분류되었음을 나타낸다. 나머지 행에 대해서도 마찬가지로 해석할 수 있다.

지금까지는 pruning을 하지 않고 minNumObj = 2인 옵션으로 j48 알고리즘을 실행한 결과인데, 똑같이 pruning은 하지 않고 minNumObj의 값을 10으로 늘렸을 때의 결과를 살펴보겠다.

-unpruned tree (minNumObj = 10)

```
Number of Leaves : 50
```

```
Size of the tree : 68
```

최소 노드 개수가 2에서 10으로 늘어나자 leaf(말단 노드)의 개수와 tree의 크기가 확연히 줄었다. 그만큼 general한 모델을 얻을 수 있다는 것이면서, minNumObj를 너무 크게 설정해버리면 underfitting의 문제가 있을 수 때문에 주의를 기울여야 한다는 것을 암시한다.

```
==== Summary ====
```

Correctly Classified Instances	1545	89.4097 %
Incorrectly Classified Instances	183	10.5903 %
Kappa statistic	0.7711	
Mean absolute error	0.0698	
Root mean squared error	0.1868	
Relative absolute error	30.4752 %	
Root relative squared error	55.245 %	
Total Number of Instances	1728	

Overfitting을 조금 해결하고 더 general해진만큼 정확도도 약 96%에서 89%로 감소했음을 알 수 있다. 또 그에 따라서 Kappa statistic 수치도 내려갔다. zeroR, oneR과 비교해서 정확도와 Kappa statistic 모두 높음을 확인할 수 있다.

이번에는 option을 모두 default로 맞춰서 결과를 보겠다.

-default option (pruned, minNumObj = 2)

```
Number of Leaves : 131
```

```
Size of the tree : 182
```

다른 option은 모두 같은 unpruned tree보다 tree의 사이즈나 leaf의 개수가 조금 줄어들었음을 확인할 수 있다. 또 모델을 만드는데 드는 시간도 pruned tree가 0.01초 더 걸렸다.

```
==== Summary ====
```

Correctly Classified Instances	1596	92.3611 %
Incorrectly Classified Instances	132	7.6389 %
Kappa statistic	0.8343	
Mean absolute error	0.0421	
Root mean squared error	0.1718	
Relative absolute error	18.3833 %	
Root relative squared error	50.8176 %	
Total Number of Instances	1728	

정확도나 Kappa statistic도 pruned tree가 unpruned보다 조금 더 내려갔음을 알 수 있다. zeroR, oneR과 비교해서 정확도와 Kappa statistic 모두 높음을 확인할 수 있다. 지금까지 3가지의 J48

decision tree를 비교해보면서 이 dataset에서는 unpruned보다는 pruned tree가, minNumObj를 크게 설정하기보다는 크게 설정하는 것이 overfitting을 좀 더 피해서 더 general한 tree를 만들 수 있지만 정확도는 떨어짐을 알 수 있었다.

<NaiveBayes>

: Bayes 정리를 적용한 확률 분류기이며, 모든 attribute 값들이 서로 독립임을 가정하는 조건부 확률 모델이다.

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

```
Naive Bayes Classifier
```

Attribute	Class			
	unacc	acc	good	vgood
<hr/>				
buying				
vhigh	361.0	73.0	1.0	1.0
high	325.0	109.0	1.0	1.0
med	269.0	116.0	24.0	27.0
low	259.0	90.0	47.0	40.0
[total]	1214.0	388.0	73.0	69.0
<hr/>				
maint				
vhigh	361.0	73.0	1.0	1.0
high	315.0	106.0	1.0	14.0
med	269.0	116.0	24.0	27.0
low	269.0	93.0	47.0	27.0
[total]	1214.0	388.0	73.0	69.0
<hr/>				
doors				
2	327.0	82.0	16.0	11.0
3	301.0	100.0	19.0	16.0
4	293.0	103.0	19.0	21.0
5more	293.0	103.0	19.0	21.0
[total]	1214.0	388.0	73.0	69.0
<hr/>				
persons				
2	577.0	1.0	1.0	1.0
4	313.0	199.0	37.0	31.0
more	323.0	187.0	34.0	36.0
[total]	1213.0	387.0	72.0	68.0
<hr/>				
lug_boot				
small	451.0	106.0	22.0	1.0
med	393.0	136.0	25.0	26.0
big	369.0	145.0	25.0	41.0
[total]	1213.0	387.0	72.0	68.0
<hr/>				
safety				
low	577.0	1.0	1.0	1.0
med	358.0	181.0	40.0	1.0
high	278.0	205.0	31.0	66.0
[total]	1213.0	387.0	72.0	68.0

Naïve Bayes Classifier는 각 attribute 내의 속성 값에 따라 해당하는 class에 분류되는 조건부 확률을 계산함으로 학습이 진행된다. 위의 classification model 부분의 각 attribute 속성 값 별

수치는 조건부에 해당하는 instance의 개수를 나타내며 이를 확률로 변환할 수 있다.

==== Summary ===

Correctly Classified Instances	1505	87.0949 %
Incorrectly Classified Instances	223	12.9051 %
Kappa statistic	0.7065	
Mean absolute error	0.1112	
Root mean squared error	0.2218	
Relative absolute error	48.5842 %	
Root relative squared error	65.5935 %	
Total Number of Instances	1728	

총 1728개의 instance 중에 1505개, 약 87%가 정확하게 분류되었다. 이는 앞선 ID3나 C4.5의 모델보다 정확도와 Kappa statistic 수치가 떨어짐을 확인할 수 있다. zeroR, oneR과 비교해서 정확도와 Kappa statistic 모두 높음을 확인할 수 있다.

==== Confusion Matrix ===

a	b	c	d	<-- classified as
1162	46	2	0	a = unacc
87	287	10	0	b = acc
0	46	21	2	c = good
0	30	0	35	d = vgood

Confusion Matrix는 분류된 결과를 좀 더 자세히 보여준다. good 그리고 vgood, 이 두가지 class에 해당하는 분류는 unacc, acc보다 잘못 분류되는 정도가 심하다. 이 Naïve Bayes 모델은 성능은 떨어지지만 아주 간단히 확률적인 모델을 만들 수 있다는 것이 장점이다.

〈Logistic Regression〉

: linear regression을 classification으로 사용하기 위해서 logistic함수를 적용하여 classification 결과 값을 0과 1 사이의 확률 값으로 표현할 수 있는 기법이다. 이 모델은 각 attribute가 class 결정에 미치는 영향력을 확인할 수 있게 해준다.

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

Variable	Class		
	unacc	acc	good
<hr/>			
buying=vhigh	23.9697	20.1921	4.8332
buying=high	21.4886	19.7706	-7.4951
buying=med	-18.4572	-16.2681	2.265
buying=low	-27.0011	-23.6947	0.3969
maint=vhigh	31.5583	27.7507	11.3713
maint=high	0.8769	-0.1154	-13.0698
maint=med	-13.8595	-11.4566	0.8103
maint=low	-18.5757	-16.1787	0.8881
doors=2	6.5652	4.865	2.461
doors=3	0.439	0.5897	0.6621
doors=4	-3.5021	-2.7273	-1.5615
doors=5more	-3.5021	-2.7273	-1.5615
persons=2	39.0971	16.4507	8.5029
persons=4	-19.0524	-7.539	-3.5921
persons=more	-20.0448	-8.9117	-4.9108

위의 상수들은 0과 1사이의 값으로 제한할 수 있다. 여기서 각 attribute 속성들이 classification에 미치는 영향을 알 수 있는데 buying이 vhigh일 경우 class는 good이나 vgood 될 가능성보

다 unacc나 acc가 될 확률이 훨씬 높다. 반대로 buying이 low일 경우, class는 good과 vgood에 좀 더 많은 확률로 분류된다. 캡쳐한 화면을 바탕으로 우리는 buying(구매가격)이 높으면 unacc로 낮으면 vgood쪽으로 분류될 확률이 높고, doors(차 문 개수)가 많으면 vgood쪽으로 분류될 확률이 높다는 등의 각 attribute의 영향력을 쉽게 분석할 수 있다. 이렇게 logistic regression은 각 attribute value가 class 분류에 미치는 영향을 자세히 알 수 있게 해준다.

==== Summary ====

Correctly Classified Instances	1634	94.5602 %
Incorrectly Classified Instances	94	5.4398 %
Kappa statistic	0.8819	
Mean absolute error	0.0393	
Root mean squared error	0.1387	
Relative absolute error	17.1827 %	
Root relative squared error	41.0335 %	
Total Number of Instances	1728	

Logistic Regression을 사용한 이 모델은 총 1728개의 instance 중에서 1634개를 정확하게 분류하는, 약 94%의 정확도를 가지고 있다. Kappa statistic도 0.8819로 상당히 큰 수치로 성능이 꽤 괜찮음을 알 수 있다. zeroR, oneR과 비교해서 정확도와 Kappa statistic 모두 높음을 확인할 수 있다.

Time taken to build model: 0.44 seconds

하지만 이 모델은 같은 dataset을 사용해서 학습하는 다른 모델들과 비교하여, 만들기까지 시간이 많이 필요하다.

==== Confusion Matrix ====

a	b	c	d	<-- classified as
1166	43	1	0	a = unacc
32	346	4	2	b = acc
0	6	59	4	c = good
0	2	0	63	d = vgood

Confusion matrix에서도 대부분의 instance들이 해당하는 class로 분류됨을 보여준다.

〈MLP(Multilayer Perceptron)〉

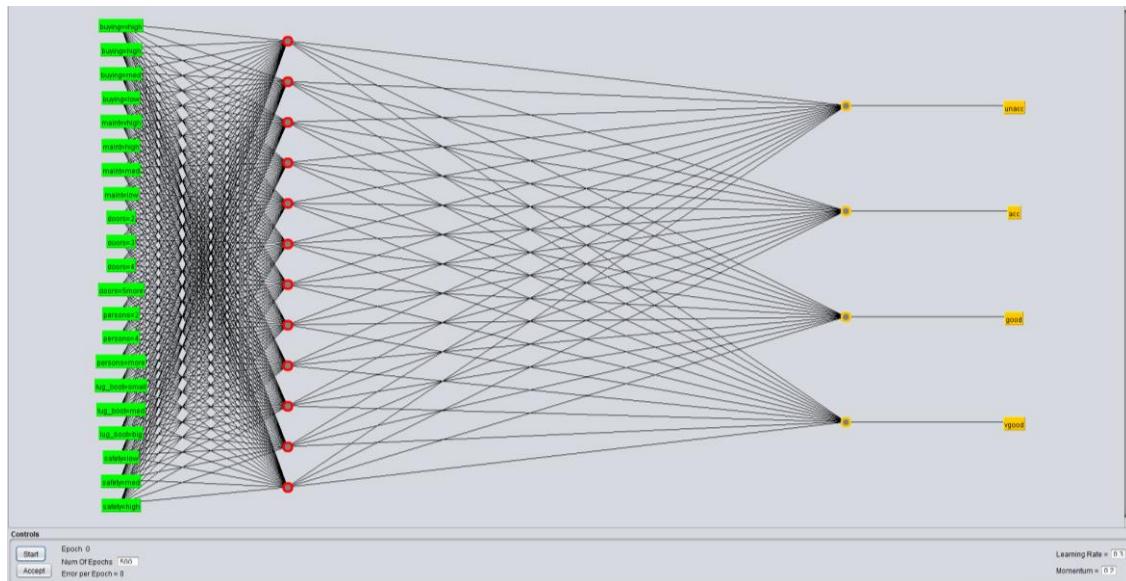
: 여러 단계를 거쳐 결정을 만들어내는 선형 모델의 일반화된 것으로, 많은 계수를 학습하여 가중치를 계산한다. 기본적으로 입력층과 출력층 사이에 1개의 hidden layer를 가진 3층 구조이다.

C4.5처럼 option을 설정할 수 있다.

-default option

먼저 MLP 모델을 가시적으로 보기 위해서 GUI를 True로 설정하고 hiddenlayers는 디폴트 값은 a로 두어서 입력층 node와 출력층 node 개수를 통해 자동으로 hidden node 개수가 정해지도록 하였다. 이는 입력과 출력의 중간 값 정도의 수치로 정해진다.

다음에서 가시적으로 모델을 확인할 수 있다.



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

```
Sigmoid Node 0
Inputs      Weights
Threshold   -4.143802085292442
Node 4      -3.243630877912873
Node 5      -2.919064041477984
Node 6      3.8138207545536273
Node 7      -2.209753886296347
Node 8      0.552141387442623
Node 9      4.843246360758957
Node 10     7.1166496080578
Node 11     4.003634178653144
Node 12     7.748383885067937
Node 13     9.5759700583043
Node 14     3.7783640586154736
Node 15     3.6193840323076225
```

```
Sigmoid Node 1
Inputs      Weights
Threshold   -10.539447423115217
Node 4       7.431816195238865
Node 5       6.432118569544556
Node 6       -6.890105215751936
Node 7       10.892813688792662
Node 8       7.036906358890431
Node 9       -6.4008177940649675
Node 10      -7.762335847683183
Node 11      -3.1954978604760114
Node 12      -7.5487321149039355
Node 13      -8.36864356802092
Node 14      -8.472940595926017
Node 15      -2.921078728405908
```

노드 별로 계수가 계산됨을 알 수 있고 16개의 hidden node가 생성되었음을 알 수 있다.

```
==== Summary ====
```

Correctly Classified Instances	1728	100	%
Incorrectly Classified Instances	0	0	%
Kappa statistic	1		
Mean absolute error	0.0023		
Root mean squared error	0.0073		
Relative absolute error	1.0132 %		
Root relative squared error	2.1496 %		
Total Number of Instances	1728		

총 1728개의 instance들이 정확히 모두 분류되었으며 Kappa statistic도 따라서 가장 높은 값인 1을 보여준다. 다음의 confusion matrix에서도 확인할 수 있다. 하지만 이는 overfitting의 문제를 생각해 볼 수 있다. zeroR, oneR과 비교해서는 정확도나 Kappa statistic 모두 높게 나타난다.

```
==== Confusion Matrix ====
```

a	b	c	d	<-- classified as
1210	0	0	0	a = unacc
0	384	0	0	b = acc
0	0	69	0	c = good
0	0	0	65	d = vgood

지금부터는 option을 조금씩 다르게 설정해서 결과가 어떻게 달라지는지 확인해보겠다.

-seed: 5

먼저, seed를 5으로 설정해보면, 각 노드 별로 계수가 다르게 나타나고 아래의 summary와 confusion matrix에서 볼 수 있듯이 정확도가 확연히 달라짐을 보였다. MLP모델은 랜덤하게 생성되는 초기값에 따라 결과가 크게 차이남을 알 수 있다.

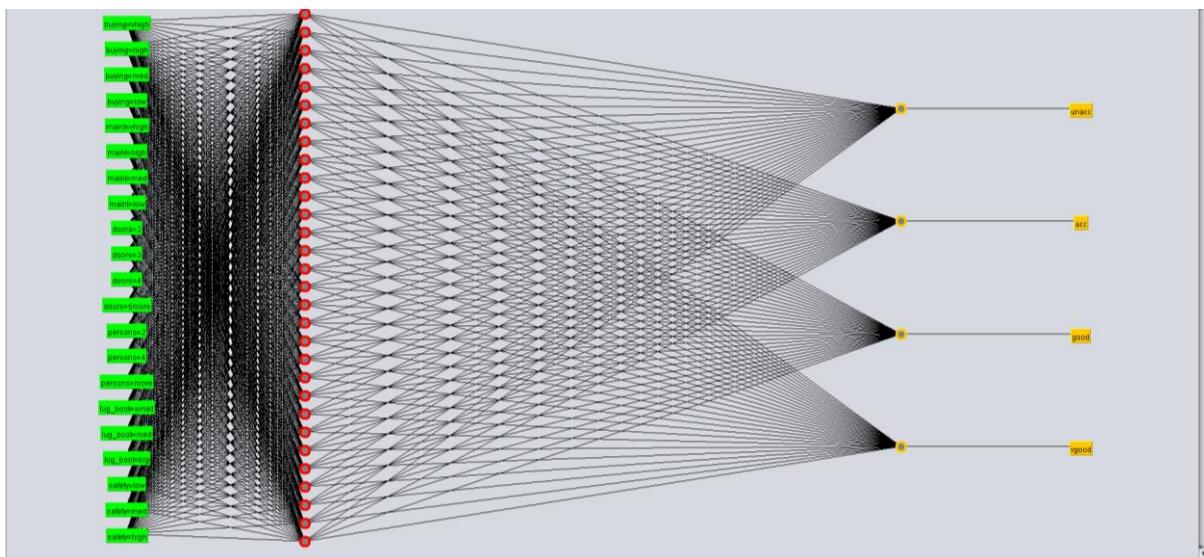
```
==== Summary ====
```

Correctly Classified Instances	363	21.0069 %
Incorrectly Classified Instances	1365	78.9931 %
Kappa statistic	0.0221	
Mean absolute error	0.379	
Root mean squared error	0.4377	
Relative absolute error	165.5159 %	
Root relative squared error	129.4568 %	
Total Number of Instances	1728	

-seed =0, hiddennode = 30

이제는 seed 값을 다시 0으로 돌리고 hiddennode를 30으로 다르게 설정해보았다.

그랬더니 classifier model 부분에 계산되는 노드가 많아졌고 가시화된 모델도 더 복잡하게 보인다.



==== Summary ====

Correctly Classified Instances	65	3.7616 %
Incorrectly Classified Instances	1663	96.2384 %
Kappa statistic	0	
Mean absolute error	0.3781	
Root mean squared error	0.4367	
Relative absolute error	165.1169 %	
Root relative squared error	129.1605 %	
Total Number of Instances	1728	

==== Confusion Matrix ====

a	b	c	d	<-- classified as
0	0	0	1210	a = unacc
0	0	0	384	b = acc
0	0	0	69	c = good
0	0	0	65	d = vgood

Hiddennode를 크게 배정했더니 정확도나 Kappa statistic가 크게 내려감을 보인다. 강의 내용에 의하면 데이터의 복잡도에 따라서 hiddennode의 개수가 많아질수록 classifier의 성능이 내려갈 수도 있다는 것을 참고할 수 있다. 또 hiddennode 개수를 너무 크게 배정해버리면, test set이 주어졌을 때 overfitting의 문제가 발생할 수 있다.

-training Time = 300

마지막으로 trainingTime을 300으로 default보다 낮게 설정해보겠다.

==== Summary ====

Correctly Classified Instances	65	3.7616 %
Incorrectly Classified Instances	1663	96.2384 %
Kappa statistic	0	
Mean absolute error	0.3768	
Root mean squared error	0.4352	
Relative absolute error	164.5715 %	
Root relative squared error	128.7028 %	
Total Number of Instances	1728	

trainingTime이 크게 낮아지니까 역시 정확도나 Kappa statistic가 크게 낮아졌다.

2.3 비교 분석 결과

0) dataset을 학습시킨 모델들의 시간, 정확도, Kappa statistic을 비교하면 다음과 같다.

Algorithm	Time taken to build model	Rate of correctly classified instances	Kappa statistics
oneR	0sec	70.0231%	0
zeroR	0sec	70.0231%	0
ID3	0.2sec	100%	1
C4.5(default option)	0.01sec	92.3611%	0.8343
Naivebayes	0sec	87.0949%	0.7065
Logistic Regression	0.44sec	94.5602%	0.8819
MLP(default)	5.51sec	100%	1

우리는 주어진 dataset에서 training data만을 이용하여 학습했다. 따라서 새로운 data가 주어졌을 때, 이 모델들의 성능을 정확하게 평가할 수 없다. 한 클래스에 약 70%의 instance들이 집중해서 분포하고 있기 때문에 oneR 모델이 정확히 분류할 확률이 상대적으로 높게 나타난다. 이 dataset은 decision tree와 MLP 모델의 정확도가 크게 차이 나지 않는 것을 볼 수 있다. 이는 10월 17일 강의 내용으로부터 이 dataset이 linearly-separable한 성격을 가짐을 추측해볼 수 있다.

3. Air Quality Dataset

3.1 데이터에 대한 설명

- regression에 적합한 dataset이다.
- attribute의 속성이 Real(numeric)이다.
- 15의 attribute가 있고 아래의 표에서 각각 의미하는 바를 확인할 수 있다.

Attribute	info
Date	DD/MM/YYYY
Time	HH.MM.SS
CO(GT)	True hourly averaged concentration CO in mg/m^3 (reference analyzer)
PT08.S1(CO)	(tin oxide) hourly averaged sensor response (nominally CO targeted)
NMHC(GT)	True hourly averaged overall Non Metanic HydroCarbons concentration in microg/m^3 (reference analyzer)
C6H6(GT)	True hourly averaged Benzene concentration in microg/m^3 (reference analyzer)
PT08.S2(NMHC)	(titania) hourly averaged sensor response (nominally NMHC targeted)
NOx(GT)	True hourly averaged NOx concentration in ppb (reference analyzer)
PT08.S3(NOx)	(tungsten oxide) hourly averaged sensor response (nominally NOx targeted)
NO2(GT)	True hourly averaged NO2 concentration in microg/m^3 (reference analyzer)
PT08.S4(NO2)	(tungsten oxide) hourly averaged sensor

	response (nominally O3 targeted)
PT08.S5(O3)	(indium oxide) hourly averaged sensor response (nominally O3 targeted)
T	Temperature in $^{\circ}$ C
RH	Relative Humidity (%)
AH	Absolute Humidity

- 총 9358개의 instance가 있다.
- 200에 tag된 Missing value있고, 전처리하니까 없어졌다.

3.2 실험 설계 및 방법과 진행 내용

<실험 설계>

- Weka에서 여러 종류의 classifier를 이용하여 dataset을 학습해보고, classifier를 비교해 본다.
- Dataset에 대하여 필요한 전처리를 거친다.
- ID3, C45, NaiveBayes, Logistic Regression, MLP 이렇게 5가지의 classifier를 이용한다.
- zeroR과 oneR의 결과를 baseline으로 두고 비교한다.
- 따로 test set을 설정하지 않고 training set을 활용한다.
- 실행 결과창에서는 classifier model, summary, confusion matrix 이렇게 3가지 항목을 중심으로 결과를 확인한다.

<실험 방법 및 진행 내용>

<전처리>

- Dataset을 엑셀을 이용해서 빈 attribute를 지우고 csv형식에 맞게 저장을 해줬다.
- Date와 Time은 학습에 도움이 되지 않는 무의미한 attribute라고 판단하여 지웠다.
(air quality와 상관관계가 없고, ID code 문제를 일으킨다고 판단함)
- instance의 분포가 이상한 attribute는 removeWithValues(filter)를 통해 처리해주었다.
Attribute T에 대해서 splitPoint를 -199.9로 설정해주고 그 이하의 값을 제거했다.
- class 지정: 15개의 attribute 중 NOx(GT)를 채택해 discretize했고 10개의 class가 생성되었다..
근거:
 - 조사 결과 NOx는 고온에서 질소가 산화되어 만들어진 질소산화물로 NO2등의 종류가 있는데, T와 NO2등 NOx와 관련이 있는 attribute들이 존재하므로 NOx를 class로 선정했다.
 - 다른 attribute로 실험적으로 class를 지정해본 결과, instance들의 분류가 너무 복잡하거나 단순하다고 생각되었는데 NOx를 class로 지정했을 때는 그 중간정도의 분류라고 판단했다.
- 전처리 후 원래 dataset의 9358개 instance 중에서 8991개로 instance를 사용하게 되었다.

<zeroR>

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

```
ZeroR predicts class value: '(-32.1-135.8]'
```

```
Time taken to build model: 0.02 seconds
```

본 실험에서 numeric한 attribute를 class로 지정해서 discretize해주었기 때문에, class value에 구간 값이 나타난 것을 볼 수 있다.

```
==== Summary ====
```

	Correctly Classified Instances	2821	31.3758 %
Incorrectly Classified Instances	6170	68.6242 %	
Kappa statistic	0		
Mean absolute error	0.1543		
Root mean squared error	0.2777		
Relative absolute error	100 %		
Root relative squared error	100 %		
Total Number of Instances	8991		

단순히 instance가 가장 많은 class만 고려해서 분류했기 때문에 총 8991개의 instance들 중 제대로 분류한 것은 원래 해당 class에 속해 있던 2821개로 약 30%에 그친다. 또 Kappa statistic이 0이므로 무작위로 추측하는 것과 유사하다고 판단할 수 있다.

```
==== Confusion Matrix ====
```

	a	b	c	d	e	f	g	h	i	j	<-- classified as
0	1595	0	0	0	0	0	0	0	0	0	a = '(-inf--32.1]'
0	2821	0	0	0	0	0	0	0	0	0	b = '(-32.1-135.8]'
0	2559	0	0	0	0	0	0	0	0	0	c = '(135.8-303.7]'
0	1042	0	0	0	0	0	0	0	0	0	d = '(303.7-471.6]'
0	530	0	0	0	0	0	0	0	0	0	e = '(471.6-639.5]'
0	263	0	0	0	0	0	0	0	0	0	f = '(639.5-807.4]'
0	110	0	0	0	0	0	0	0	0	0	g = '(807.4-975.3]'
0	47	0	0	0	0	0	0	0	0	0	h = '(975.3-1143.2]'
0	18	0	0	0	0	0	0	0	0	0	i = '(1143.2-1311.1]'
0	6	0	0	0	0	0	0	0	0	0	j = '(1311.1-inf)'

Confusion Matrix를 보면, zeroR은 가장 많은 instance들이 속한 클래스(구간 값)로 모두 분류함을 볼 수 있다. 따라서 원래 두번째 구간 값에 속했던 2821개의 instance만이 정확히 분류된다.

〈oneR〉

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

```
NO2(GT):
  '(-inf--146.7]' -> '(-inf--32.1]'
  '(-146.7--93.4]' -> '(-inf--32.1]'
  '(-93.4--40.1]' -> '(-inf--32.1]'
  '(-40.1-13.2]' -> '(-32.1-135.8]'
  '(13.2-66.5]' -> '(-32.1-135.8]'
  '(66.5-119.8]' -> '(-32.1-135.8]'
  '(119.8-173.1]' -> '(135.8-303.7]'
  '(173.1-226.4]' -> '(303.7-471.6]'
  '(226.4-279.7]' -> '(639.5-807.4]'
  '(279.7-inf)' -> '(639.5-807.4]'

(5726/8991 instances correct)
```

```
Time taken to build model: 0.03 seconds
```

Classifier model을 보면 이 oneR 모델에서 채택한 attribute는 NO2(GT)임을 알 수 있다. 이 attribute들의 값들의 구간에 따라서 class가 결정된다. 이 dataset에서는 zeroR보다 모델을 만들기 위한 시간이 조금 더 늘어났음을 확인할 수 있다.

==== Summary ===

Correctly Classified Instances	5726	63.6859 %
Incorrectly Classified Instances	3265	36.3141 %
Kappa statistic	0.507	
Mean absolute error	0.0726	
Root mean squared error	0.2695	
Relative absolute error	47.083 %	
Root relative squared error	97.0482 %	
Total Number of Instances	8991	

이 dataset에서 oneR을 이용한 모델은 zeroR에 비해서 정확도가 높은 것을 확인할 수 있다. 또 Kappa statistic도 여전히 1과는 먼 수치지만 zeroR에 비해서는 증가된 수치를 보인다. 이 모델은 총 8991개의 instance 중에서 5726개를 정확하게 분류하였다.

==== Confusion Matrix ===

```

a   b   c   d   e   f   g   h   i   j   <- classified as
1595  0   0   0   0   0   0   0   0   0 |   a = '(-inf--32.1]'
1 2737  83  0   0   0   0   0   0   0   0 |   b = '(-32.1-135.8]'
2 1326 1147  84  0   0   0   0   0   0   0 |   c = '(135.8-303.7]'
0  277  540 211  0   14  0   0   0   0   0 |   d = '(303.7-471.6]'
0   80  237 182  0   31  0   0   0   0   0 |   e = '(471.6-639.5]'
0   16  96 115  0   36  0   0   0   0   0 |   f = '(639.5-807.4]'
0   1   35  50  0   24  0   0   0   0   0 |   g = '(807.4-975.3]'
0   0   7  23  0   17  0   0   0   0   0 |   h = '(975.3-1143.2]'
0   0   1   7  0   10  0   0   0   0   0 |   i = '(1143.2-1311.1]'
0   0   0   2  0   4   0   0   0   0   0 |   j = '(1311.1-inf)'

```

<ID3>

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

ID3

```

NO2(GT) = '(-inf--146.7]'
| RH = '(-inf-17.15]': '(-inf--32.1]'
| RH = '(17.15-25.1]'
| | PT08.S4(NO2) = '(-inf-773.4]': '(-inf--32.1]'
| | PT08.S4(NO2) = '(773.4-995.8]': null
| | PT08.S4(NO2) = '(995.8-1218.2]': '(-inf--32.1]'
| | PT08.S4(NO2) = '(1218.2-1440.6]': '(-inf--32.1]'
| | PT08.S4(NO2) = '(1440.6-1663]': '(-inf--32.1]'
| | PT08.S4(NO2) = '(1663-1885.4]'
| | | T = '(-inf-2.75]': null
| | | T = '(2.75-7.4]': null
| | | T = '(7.4-12.05]': null
| | | T = '(12.05-16.7]': null
| | | T = '(16.7-21.35]': null
| | | T = '(21.35-26]': '(-inf--32.1]'
| | | T = '(26-30.65]': '(-inf--32.1]'
| | | T = '(30.65-35.3]': '(135.8-303.7]'
| | | T = '(35.3-39.95]': '(-inf--32.1]'
| | | T = '(39.95-inf)': null
| | PT08.S4(NO2) = '(1885.4-2107.8]': '(-inf--32.1]'
| | PT08.S4(NO2) = '(2107.8-2330.2]': null
| | PT08.S4(NO2) = '(2330.2-2552.6]': null
| | PT08.S4(NO2) = '(2552.6-inf)': null
| RH = '(25.1-33.05]': '(-inf--32.11'

```

```
==== Summary ===
```

Correctly Classified Instances	8667	96.3964 %
Incorrectly Classified Instances	324	3.6036 %
Kappa statistic	0.9532	
Mean absolute error	0.0083	
Root mean squared error	0.0645	
Relative absolute error	5.3898 %	
Root relative squared error	23.2182 %	
Total Number of Instances	8991	

oneR, zeroR에 비해서 정확도와 Kappa statistic 수치가 많이 증가했을 뿐만 아니라 1에 가까운 정도를 보이고 있다. 8991개의 instance 중 8667개를 정확히 분류해서 약 96%정도의 정확도를 보인다. 이는 ID3가 주어진 dataset에 대해서 분류를 정확히 수행한다는 것이지만, test set에 대해 overfitting의 문제를 지닐 수 있다는 것을 동시에 내포한다.

```
==== Confusion Matrix ===
```

a	b	c	d	e	f	g	h	i	j	<-- classified as
1595	0	0	0	0	0	0	0	0	0	a = '(-inf--32.1]'
0	2777	43	1	0	0	0	0	0	0	b = '(-32.1-135.8]'
0	124	2430	5	0	0	0	0	0	0	c = '(135.8-303.7]'
0	1	67	965	9	0	0	0	0	0	d = '(303.7-471.6]'
0	0	2	40	485	3	0	0	0	0	e = '(471.6-639.5]'
0	0	0	5	11	247	0	0	0	0	f = '(639.5-807.4]'
0	0	0	0	0	9	101	0	0	0	g = '(807.4-975.3]'
0	0	0	0	0	1	2	44	0	0	h = '(975.3-1143.2]'
0	0	0	0	0	0	0	1	17	0	i = '(1143.2-1311.1]'
0	0	0	0	0	0	0	0	0	6	j = '(1311.1-inf)'

<C4.5>

```
-option: minNumObj = 5 pruned tree
```

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

```
J48 pruned tree
```

```
-----  
NO2(GT) = '(-inf--146.7]': '(-inf--32.1]' (1598.0/3.0)  
NO2(GT) = '(-146.7--93.4]': '(-32.1-135.8]' (0.0)  
NO2(GT) = '(-93.4--40.1]': '(-32.1-135.8]' (0.0)  
NO2(GT) = '(-40.1-13.2]': '(-32.1-135.8]' (14.0)  
NO2(GT) = '(13.2-66.5]'  
| C6H6(GT) = '(-inf-6.46]': '(-32.1-135.8]' (1124.0/30.0)  
| C6H6(GT) = '(6.46-12.82]'  
| | PT08.S4(NO2) = '(-inf-773.4]': '(-32.1-135.8]' (0.0)  
| | PT08.S4(NO2) = '(773.4-995.8]': '(-32.1-135.8]' (0.0)  
| | PT08.S4(NO2) = '(995.8-1218.2]': '(-32.1-135.8]' (1.0)  
| | PT08.S4(NO2) = '(1218.2-1440.6]': '(135.8-303.7]' (17.0/5.0)  
| | PT08.S4(NO2) = '(1440.6-1663]'  
| | | PT08.S5(O3) = '(-inf-451.2]': '(135.8-303.7]' (0.0)  
| | | PT08.S5(O3) = '(451.2-681.4]': '(-32.1-135.8]' (4.0)  
| | | PT08.S5(O3) = '(681.4-911.6]': '(-32.1-135.8]' (23.0/3.0)  
| | | PT08.S5(O3) = '(911.6-1141.8]': '(135.8-303.7]' (39.0/9.0)  
| | | PT08.S5(O3) = '(1141.8-1372]': '(135.8-303.7]' (6.0/1.0)  
| | | PT08.S5(O3) = '(1372-1602.2]': '(135.8-303.7]' (2.0)  
| | | PT08.S5(O3) = '(1602.2-1832.41]': '(135.8-303.71]' (0.0)
```

```
==== Summary ====
```

Correctly Classified Instances	7434	82.6827 %
Incorrectly Classified Instances	1557	17.3173 %
Kappa statistic	0.7745	
Mean absolute error	0.0496	
Root mean squared error	0.1575	
Relative absolute error	32.1807 %	
Root relative squared error	56.7333 %	
Total Number of Instances	8991	

이 모델은 이 dataset에 대해서 ID3를 이용한 모델보다는 정확도나 Kappa statistic이 떨어짐을 확인할 수 있다. 그 이유는 ID3의 overfitting 가능성을 pruning을 통해 좀 더 general한 모델을 만들었기 때문이다. 총 8991개의 instance 중 7434개를 정확하게 분류하는 약 82%정도의 정확도를 보인다. 또 이 dataset에 대해서 zeroR, oneR에 비해 이 모델이 정확도와 Kappa statistic 수치 모두 높은 것을 볼 수 있다.

```
==== Confusion Matrix ====
```

a	b	c	d	e	f	g	h	i	j	<-- classified as
1595	0	0	0	0	0	0	0	0	0	a = '(-inf--32.1]'
1 2560	257	3	0	0	0	0	0	0	0	b = '(-32.1-135.8]'
2 335	2063	150	8	1	0	0	0	0	0	c = '(135.8-303.7]'
0 5	234	687	106	9	1	0	0	0	0	d = '(303.7-471.6]'
0 0	28	136	343	16	6	1	0	0	0	e = '(471.6-639.5]'
0 0	4	12	124	97	23	2	1	0	0	f = '(639.5-807.4]'
0 0	1	1	17	21	65	2	3	0	0	g = '(807.4-975.3]'
0 0	1	0	3	13	14	12	4	0	0	h = '(975.3-1143.2]'
0 0	0	0	1	4	4	0	9	0	0	i = '(1143.2-1311.1]'
0 0	0	0	0	0	2	0	1	3	1	j = '(1311.1-inf)'

<Naivebayes>

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

Naive Bayes Classifier

Attribute	Class				
	'(-inf--32.1]' (0.18)	'(-32.1-135.8]' (0.31)	'(135.8-303.7]' (0.28)	'(303.7-471.6]' (0.12)	'(471.6-639.5]' (0.06)
<hr/>					
CO(GT)					
'(-inf--178.81]'	1196.0	345.0	89.0	15.0	5.0
'(-178.81--157.62]'	1.0	1.0	1.0	1.0	1.0
'(-157.62--136.43]'	1.0	1.0	1.0	1.0	1.0
'(-136.43--115.24]'	1.0	1.0	1.0	1.0	1.0
'(-115.24--94.05]'	1.0	1.0	1.0	1.0	1.0
'(-94.05--72.86]'	1.0	1.0	1.0	1.0	1.0
'(-72.86--51.67]'	1.0	1.0	1.0	1.0	1.0
'(-51.67--30.48]'	1.0	1.0	1.0	1.0	1.0
'(-30.48--9.29]'	1.0	1.0	1.0	1.0	1.0
'(-9.29--inf)'	401.0	2478.0	2472.0	1029.0	527.0
[total]	1605.0	2831.0	2569.0	1052.0	540.0
<hr/>					
PT08.S1(CO)					
'(-inf--786.3]'	76.0	177.0	19.0	1.0	1.0
'(786.3-925.6]'	446.0	1041.0	272.0	28.0	1.0
'(925.6-1064.9]'	486.0	1031.0	788.0	147.0	14.0
'(1064.9-1204.2]'	276.0	464.0	774.0	355.0	76.0
'(1204.2-1343.5]'	160.0	104.0	396.0	317.0	185.0
'(1343.5-1482.8]'	90.0	10.0	212.0	119.0	184.0
'(1482.8-1622.1]'	49.0	1.0	85.0	45.0	64.0
'(1622.1-1761.4]'	18.0	1.0	18.0	21.0	12.0
'(1761.4-1900.7]'	3.0	1.0	4.0	13.0	1.0
'(1900.7--inf)'	1.0	1.0	1.0	6.0	2.0
[total]	1605.0	2831.0	2569.0	1052.0	540.0
<hr/>					
NMHC(GT)					

NMHC(GT)

각 attribute 속성 값에 대해서 확률 값을 계산하기 위한 조건부를 만족하는 instance 개수들을 확인할 수 있다.

==== Summary ===

Correctly Classified Instances	6342	70.5372 %
Incorrectly Classified Instances	2649	29.4628 %
Kappa statistic	0.6224	
Mean absolute error	0.0662	
Root mean squared error	0.2045	
Relative absolute error	42.8985 %	
Root relative squared error	73.6345 %	
Total Number of Instances	8991	

앞에서 제시한 ID3나 C4.5 모델들에 비해서는 정확도나 Kappa statistic이 떨어진다. 하지만 여전히 이 dataset에 대해서 zeroR, oneR에 비해 이 모델이 정확도와 Kappa statistic 수치 모두 높은 것을 볼 수 있다. 총 8991개의 instance 중 6342개를 정확히 분류하는 약 70%의 정확도를 보인다.

==== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	<-- classified as
1563	1	0	1	6	1	3	10	8	2	a = '(-inf--32.1]'
1 2381	416	20	3	0	0	0	0	0	0	b = '(-32.1-135.8]'
2 600	1408	286	205	48	6	2	2	2	0	c = '(135.8-303.7]'
0 15	225	492	227	52	17	5	7	2	2	d = '(303.7-471.6]'
0 0	13	135	265	104	8	4	1	0	1	e = '(471.6-639.5]'
0 0	1	11	68	155	22	5	0	1	1	f = '(639.5-807.4]'
0 0	0	1	11	37	48	7	5	1	1	g = '(807.4-975.3]'
0 0	0	0	0	8	18	16	3	2	2	h = '(975.3-1143.2]'
0 0	0	0	0	2	5	2	9	0	1	i = '(1143.2-1311.1]'
0 0	0	0	0	0	0	0	1	5	1	j = '(1311.1-inf)'

<Logistic regression>

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

Logistic Regression with ridge parameter of 1.0E-8
Coefficients...

Variable	Class	'(-inf--178.81]'	'(-32.1-135.8]'	'(135.8-303.7]'	'(303.7-471.6]'	'(471.6-639.5]'	'(639.5-807.4]'	'(807.4-975.3]'	'(975.3-1143.2]'	'(1143.2-1311.1]'	'(1311.1-inf)'
CO(GT)='(-inf--178.81]'	10.7235	-0.8676	-1.74	-2.7073	-2.8491						
CO(GT)='(-178.81--157.62]'	0	0	0	0	0	0	0	0	0	0	
CO(GT)='(-157.62--136.43]'	0	0	0	0	0	0	0	0	0	0	
CO(GT)='(-136.43--115.24]'	0	0	0	0	0	0	0	0	0	0	
CO(GT)='(-115.24--94.05]'	0	0	0	0	0	0	0	0	0	0	
CO(GT)='(-94.05--72.86]'	0	0	0	0	0	0	0	0	0	0	
CO(GT)='(-72.86--51.67]'	0	0	0	0	0	0	0	0	0	0	
CO(GT)='(-51.67--30.48]'	0	0	0	0	0	0	0	0	0	0	
CO(GT)='(-30.48--9.29]'	0	0	0	0	0	0	0	0	0	0	
CO(GT)='(-9.29--inf)' PT08.SI(CO)='(-inf-786.3]'	-10.7235 -2.5985	0.8676 -3.0066	1.74 -4.6013	2.7074 -20.598	2.8491 -4.1229						
PT08.SI(CO)='(786.3-925.6]'	0.0736	0.5061	-1.7215	0.2408	-9.4664						
PT08.SI(CO)='(925.6-1064.9]' PT08.SI(CO)='(1064.9-1204.2]' PT08.SI(CO)='(1204.2-1343.5]' PT08.SI(CO)='(1343.5-1482.8]' PT08.SI(CO)='(1482.8-1622.1]' PT08.SI(CO)='(1622.1-1761.4]' PT08.SI(CO)='(1761.4-1900.7]' PT08.SI(CO)='(1900.7-inf)' NMHC(GT)='(-inf--61.1]' NMHC(GT)='(-61.1--77.8]' NMHC(GT)='(-77.8--216.7]' NMHC(GT)='(-216.7--355.6]' NMHC(GT)='(-355.6--494.5]' NMHC(GT)='(-494.5--633.4]' NMHC(GT)='(-633.4--772.3]' NMHC(GT)='(-772.3--911.2]' NMHC(GT)='(-911.2--1050.1]'	-3.3413 -0.0379 -0.6542 2.5567 7.4747 16.2199 -0.5687 -36.0545 -4.4154 -2.623 -2.6337 6.416 9.5051 13.3267 25.4628 77.1118 65.0989	2.13 1.8787 1.5546 0.51 -19.0034 -4.2664 -23.4148 -16.3562 -4.2912 3.2506 3.9284 7.9275 -20.5326 -0.8645 2.5764 65.3216 57.9886	-0.0704 -0.1678 -0.1627 -0.225 6.2109 26.2409 -9.8169 -44.7978 -5.6445 0.2428 1.203 5.9038 1.557 23.1647 17.3273 63.544 61.6605	-0.2401 0.1405 0.2827 0.2407 6.7709 27.795 -6.1837 1.7721 -0.6063 -21.9161 3.2918 8.7681 4.5566 27.3636 17.3573 67.1757 66.9375	2.0774 2.4923 2.4407 9.463 30.3373 -127.6422 23.2217 7.657 2.1629 -8.0284 -5.7836 -18.6601 17.2573 -46.975 -65.6854 -41.2568						

각 attribute가 각 class에 미치는 영향이 계산되었음을 볼 수 있다.

```
Time taken to build model: 17035.39 seconds
```

이 dataset에 대해서 logistic regression을 사용해서 모델을 만드는 데에는 지금까지 만든 모델들에 비해서 몇 백배 이상의 시간이 걸렸다. 이전에 주어진 Car Evaluation dataset에 비해 instance 수가 많았던 것도 일부 영향을 주었을 것이다.

```
==== Summary ===
```

Correctly Classified Instances	7408	82.3935 %
Incorrectly Classified Instances	1583	17.6065 %
Kappa statistic	0.771	
Mean absolute error	0.0475	
Root mean squared error	0.1543	
Relative absolute error	30.7699 %	
Root relative squared error	55.5573 %	
Total Number of Instances	8991	

이 모델은 이 dataset에 대해서 Naivebayes를 사용한 모델보다는 더 높은 정확도와 Kappa statistic을, C4.5와는 유사한 수치를, 그리고 ID3에 비해서는 낮은 수치를 보이고 있다. 그렇지만 zeroR, oneR에 비해서는 두 수치 모두 높은 것은 확실하다.

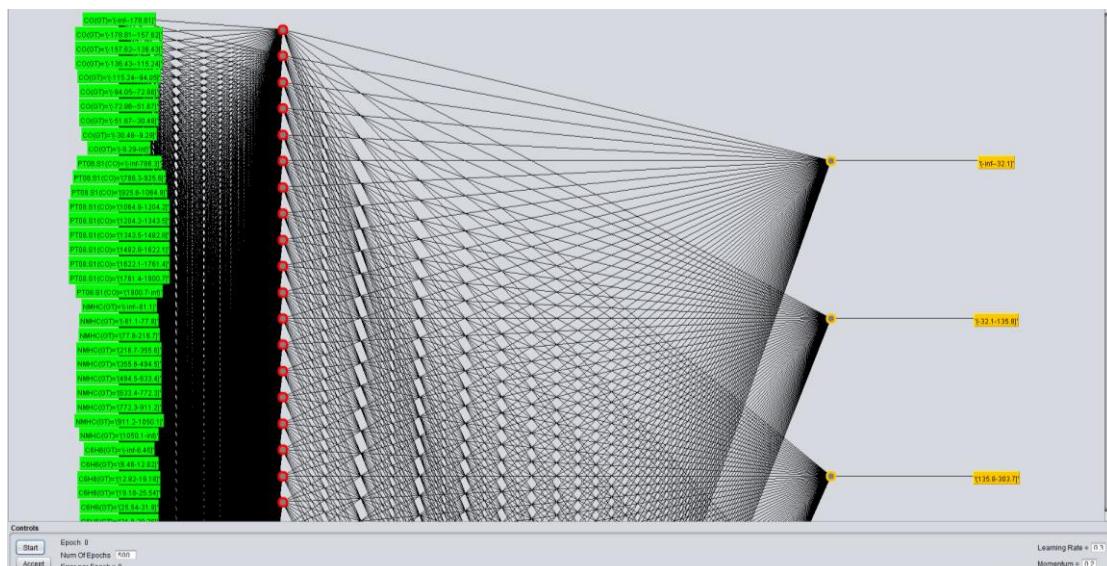
```
==== Confusion Matrix ===
```

a	b	c	d	e	f	g	h	i	j	<-- classified as
1595	0	0	0	0	0	0	0	0	0	a = '(-inf--32.1]'
0	2510	311	0	0	0	0	0	0	0	b = '(-32.1-135.8]'
0	303	2074	172	7	3	0	0	0	0	c = '(135.8-303.7]'
0	2	260	666	111	3	0	0	0	0	d = '(303.7-471.6]'
0	0	13	148	308	57	4	0	0	0	e = '(471.6-639.5]'
0	0	0	6	88	149	19	1	0	0	f = '(639.5-807.4]'
0	0	0	1	9	39	50	11	0	0	g = '(807.4-975.3]'
0	0	0	0	0	4	10	32	1	0	h = '(975.3-1143.2]'
0	0	0	0	0	0	0	0	18	0	i = '(1143.2-1311.1]'
0	0	0	0	0	0	0	0	0	6	j = '(1311.1-inf)'

<MLP>

-option: GUI = True, 나머지는 default랑 같게 설정함

아래 사진은 만들어진 모델을 가시화하여 보여준다. Hiddenlayer 개수는 h=a로 지정했다.



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

```
Sigmoid Node 0
  Inputs      Weights
Threshold    0.0489199301501676
Node 10     -0.03819265879751237
Node 11     -0.01683404965912949
Node 12     -0.010674051907431793
Node 13     0.022170878805949062
Node 14     0.027377843688759243
Node 15     -0.04890391584193803
Node 16     -0.04617069987305111
Node 17     -0.025807382872149377
Node 18     -0.015403389835847395
Node 19     0.00536181766598165
Node 20     -0.02208326747868038
Node 21     0.03152178018159123
Node 22     0.021644897026905405
Node 23     0.027257920000207866
Node 24     -0.0051945783881978125
Node 25     0.029610825300549995
Node 26     0.002532028938712648
Node 27     0.047577174150016305
Node 28     0.03206922203898738
Node 29     0.03724720752220198
Node 30     -0.03333364784646695
Node 31     0.020474836437688604
Node 32     0.02398121734729876
Node 33     -0.03742578927603525
```

입력층과 출력층 사이에 hidden node들이 생성되었음을 알 수 있다.

```
Time taken to build model: 8.76 seconds
```

Logistic regression 모델보다는 훨씬 적은 시간이지만, 다른 모델들보다 시간이 좀 걸렸다.

```
== Summary ==
```

Correctly Classified Instances	2822	31.3869 %
Incorrectly Classified Instances	6169	68.6131 %
Kappa statistic	0.0009	
Mean absolute error	0.1799	
Root mean squared error	0.2999	
Relative absolute error	116.6538 %	
Root relative squared error	108.0115 %	
Total Number of Instances	8991	

이 dataset에 대해서 zeroR, oneR보다는 정확도나 Kappa statistic이 높은 수치를 보이지만 다른 알고리즘을 활용한 모델들과 비교했을 때 상당히 낮은 정확도와 Kappa statistic 수치를 보인다.

```
== Confusion Matrix ==
```

a	b	c	d	e	f	g	h	i	j	<- classified as
0	1595	0	0	0	0	0	0	0	1	a = '(-inf--32.1]'
0	2821	0	0	0	0	0	0	0	1	b = '(-32.1-135.8]'
0	2558	0	0	0	0	1	0	0	0	c = '(135.8-303.7]'
0	1039	0	0	0	0	3	0	0	0	d = '(303.7-471.6]'
0	521	0	0	0	0	9	0	0	0	e = '(471.6-639.5]'
0	261	0	0	0	0	2	0	0	0	f = '(639.5-807.4]'
0	109	0	0	0	0	1	0	0	0	g = '(807.4-975.3]'
0	47	0	0	0	0	0	0	0	0	h = '(975.3-1143.2]'
0	18	0	0	0	0	0	0	0	0	i = '(1143.2-1311.1]'
0	6	0	0	0	0	0	0	0	0	j = '(1311.1-inf)'

3.3 비교 분석 결과

oneR, zeroR과 다섯가지 알고리즘을 활용한 모델들을 시간과 정확도, Kappa statistic 측면에서 비교한 표는 다음과 같다.

Algorithm	Time taken to build model	Rate of correctly classified instances	Kappa statistics
zeroR	0.02sec	31.3758 %	0
oneR	0.03sec	63.6859%	0.507
ID3	0.11sec	96.3964%	0.9532
C4.5(default option)	0.42sec	82.6827%	0.7745
Naivebayes	0.02sec	70.5372%	0.6224
Logistic Regression	17035.39sec	82.3935%	0.771
MLP(default)	8.76sec	31.3869%	0.0009

이 Air Quality dataset은 앞선 Car Evaluation dataset보다 모델들 간의 정확도 차이가 상대적으로 큰 편이다. Decision tree와 MLP모델의 정확도 차이가 매우 크다는 것은, 물론 모델 자체 알고리즘의 차이도 있지만, 이 dataset이 linearly-separable한 성격이 적다는 것을 추측해볼 수 있다. 즉, linear classification model과 nonlinear classification model의 정확도 차이가 많이 나고, nonlinear classifier의 정확도가 우선 이 training set에 대해서 더 높다는 것은, 이 dataset이 linear한 성질을 띵기보다는 nonlinear한 성질을 가지고 있다고 판단하는 것이 더 합리적이다.

4. 결론

지금까지 Car Evaluation, Air Quality 두가지 dataset을 ID3, C4.5, Naivebayes, Logistic Regression, MLP 이 다섯가지의 알고리즘을 활용하여 classification 모델을 만들어 보았다. 따로 test set을 두지 않고 training set만을 본 실험에 활용하여 학습했기 때문에 아쉬운 점은 모델이 새로운 데이터를 받았을 때 어느정도 제대로 classify할 수 있는 능력이 되는지, overfit하는지, underfit하는지 알아 낼 수 없다는 것이 본 실험의 한계다. 두가지 dataset 모두 공통적으로 ID3를 활용한 모델이 new data가 주어졌을 때 가장 overfitting될 가능성이 높아보였고, C4.5의 pruning을 통해 조금 해결할 수 있음을 보였다. Car Evaluation dataset보다 Air Quality dataset이 instance 수가 많고, instance의 분포도 덜 고른 경향이 있어서 모델의 정확도가 조금 떨어진 것이라고 생각해보았다. Car Evaluation dataset의 경우, instance들의 분포가 정확하지는 않지만 약간 변형된 normal distribution의 분포 모양을 닮아 있었고, 각 attribute label별 instance 개수의 분포는 균일했지만 특정 class에 instance가 너무 집중되어 있어 zeroR, oneR에서 Air Quality dataset보다 높은 정확도를 보여주었다. 또한 statistical modeling의 경우 모든 attribute가 동일한 중요도를 가지기보다는 특정 attribute가 더 중요할 수도 있을 것 같다.

참고문헌

Carlos Guestrin, 2007, Decision Trees

Michael Paul, 2017, Nonlinear Classification

Randall Balestrieri, 2017, Neural Decision Trees