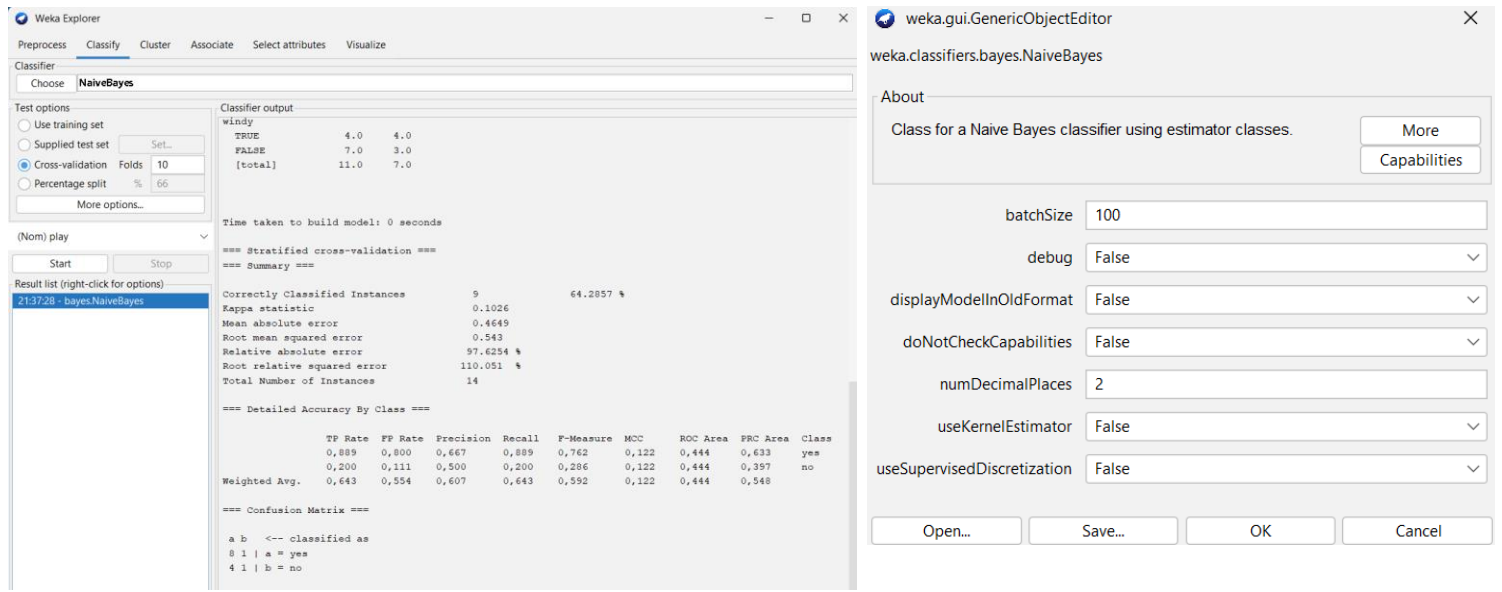


Bu ödevde WEKA yazılımı kullanılarak 4 adet sınıflandırıcı ve 1 adet öbekleyicinin sonuçları incelenmiştir. .zip uzantılı dosyada ilgili model ve txt dosyaları da eklenmiştir. Ninovada yüklenen .arff uzantılı veri kümesini programa yükledim. Bu program belirli hava koşullarında golf oynanıp oynanılmayacağına göre sınıflandırılmıştır.

1. Naive Bayes Sınıflandırıcı

Naive Bayes algoritması örüntü tanıma sorunlarına kısıtlayıcı görülen önerme ile kullanılabilir. Bu önerme, örüntü tanımada kullanılacak her bir tanımlayıcı öznelite ya da parametrenin istatistik açıdan bağımsız olması gerekliliğidir.



Üstteki fotoğrafta görüldüğü üzere WEKA arayüzü üzerinden **classify** sekmesine gidilerek **NaiveBayes** sınıflandırıcı seçilir. Yukarıdaki fotoğrafta sayfanın sol kısmında **Test Option** sekmesi ve bu sekmede test ve training aşamaları için bazı uygulamalar mevcut. Bu uygulamalar arasında **cross-validation** ve **percentage split** seçenekleri var. Bu uygulamaları açıklamak gerekirse:

Cross Validation istatistiksel bir analizde bağımsız bir veri setinde nasıl bir sonuç elde edeceğini kontrol eden bir model doğrulama tekniğidir. Burada da görüldüğü üzere her işlemde belirlenen **folds** değeri kadar veriyi test için kullanır. Geri kalan verileri ise training için kullanır.

Percentage Split, belli bir yüzdeye göre sınıflandırıcının ne kadar iyi olduğunu test eder. % bölgesine girilen değere göre bu işlemi gerçekleştirir. Yukarıdaki fotoğrafta görüldüğü üzere seçimleri bu şekilde bıraktım.

Naive Bayes'in default ayarlarını yukarıda belirttim. Buradaki **batchsize** parametresi olduğunu görüyorum. Bu parametre çoklu tahmin yapma durumunda kullanılan bir parametredir. İşlem yapılacak olan verinin kaçlı şekilde seçilerek uygulama yapılacağını söyler. Varsayılanda ayarlarında bıraktım. Kısıtlı bir veri setine sahip olduğumuz için üzerinde değişiklik yapma ihtiyacı duymadım.

Sınıflandırıcının çıktıları aşağıdaki gibidir.

```
Correctly Classified Instances      9          64.2857 %
Kappa statistic                    0.1026
Mean absolute error                 0.4649
Root mean squared error             0.543
Relative absolute error             97.6254 %
Root relative squared error        110.051 %
Total Number of Instances          14

=== Detailed Accuracy By Class ===

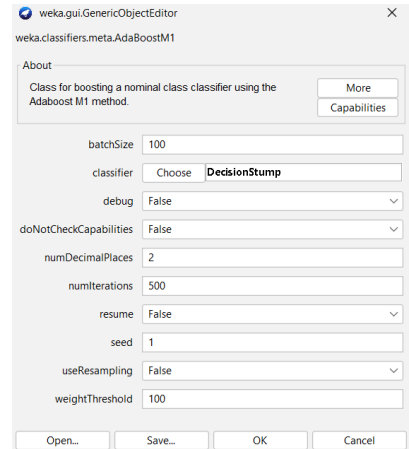
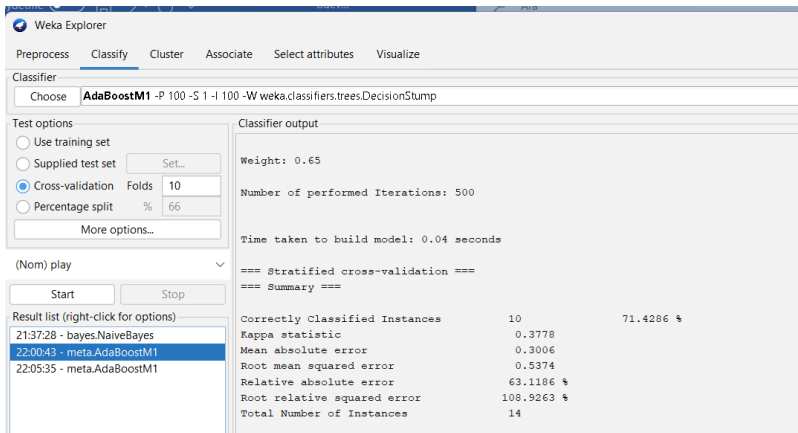
                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0,889    0,800    0,667      0,889    0,762      0,122    0,444    0,633    yes
                0,200    0,111    0,500      0,200    0,286      0,122    0,444    0,397    no
Weighted Avg.    0,643    0,554    0,607      0,643    0,592      0,122    0,444    0,548

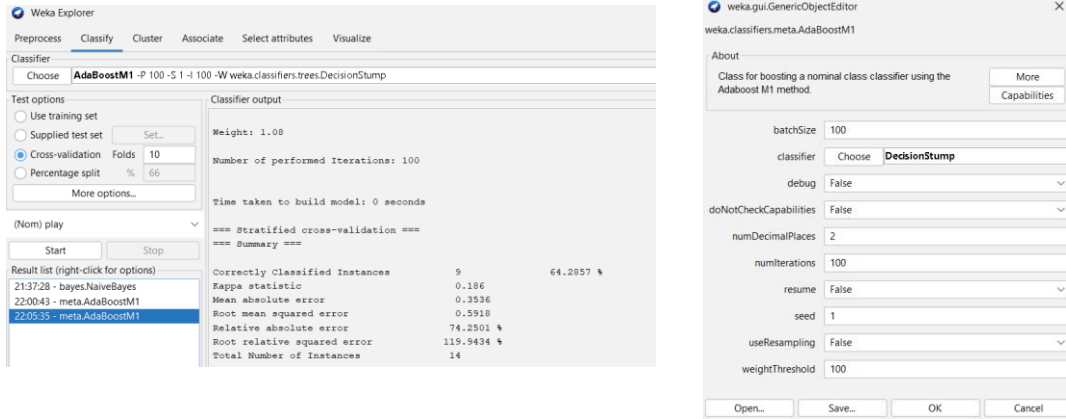
=== Confusion Matrix ===

a b  <-- classified as
8 1 | a = yes
4 1 | b = no
```

2. AdaBoost Sınıflandırıcı

Bu modelde eğitim kümesi önce bir zayıf öğrenici ile eğitilir. Eğitimden sonraki aşamada ise yanlış olarak tahminlenen örnekler bakılır. Yanlışları incelemek bu algoritma için önemlidir çünkü ilk tahminlemede yanlış öğrenilenlere daha fazla öncelik verilir. Bundan sonra ise ağırlıkları artırarak tekrar eğitilir.





Yukarıda AdaBoostM1 sınıflandırıcı modeline ait çıktı ve arayüzü ve default ayarları belirtilmiştir. Bu sekmede ilk olarak classifier sekmesinden tekrar bir seçim yapıldı. Meta sekmesinden AdaBoostM1 seçeneği seçildi ve ardından program arayüzünden parametreleri ayarlamaya geçtim. Parametreler ile ilgili düşüncelerimi aşağıda belirttim:

Parametrelerin default halini değiştirdim. Değiştirdiğim **numIterations** parametresi sınıflandırmada kaç iterasyon yapılacağını söyleyen parametredir. İterasyon sayısı artırıldığında sınıflandırıcı daha başarılı hale geliyor. Bunu iterasyon sayısı ilk olarak 500 yaptım, ardından bu değeri 100 olarak değiştirdim. Bu uygulama sonucunda başarı değerim %71'den %64'e düştü.

Değiştirdiğim başka bir parametre **useResampling** parametresi oldu. Bu parametre yeniden ağırlıklandırma yerine yeniden örnekleme kullanılmasını sağlar. Daha iyi sonuç verdiği için True olarak seçtim. True ve False değerleri için oluşan çıktılar aşağıdaki gibidir.

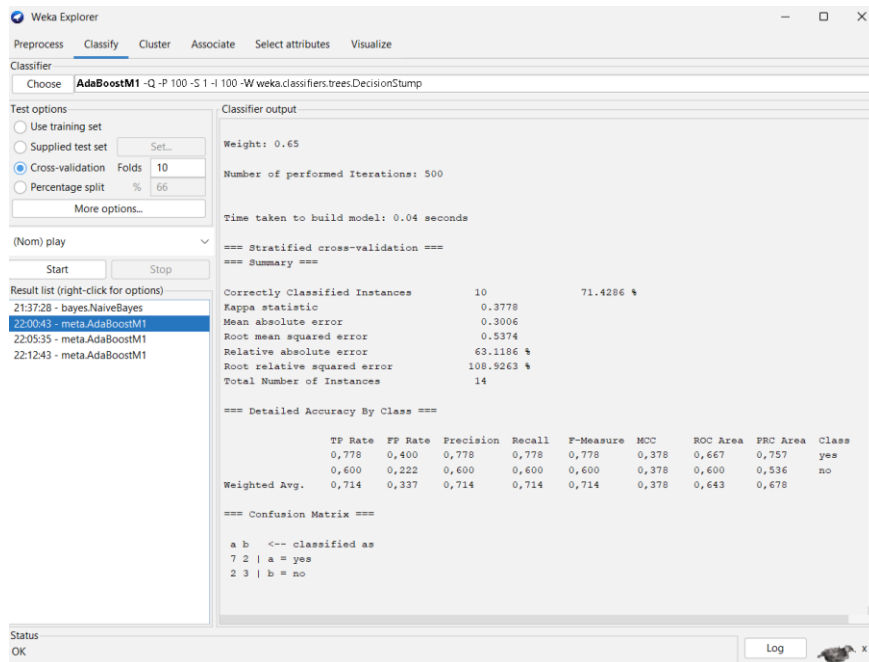
FALSE

```
a b <-- classified as
7 2 | a = yes
2 3 | b = no
```

TRUE

```
=== Confusion Matrix ===
a b <-- classified as
6 3 | a = yes
3 2 | b = no
```

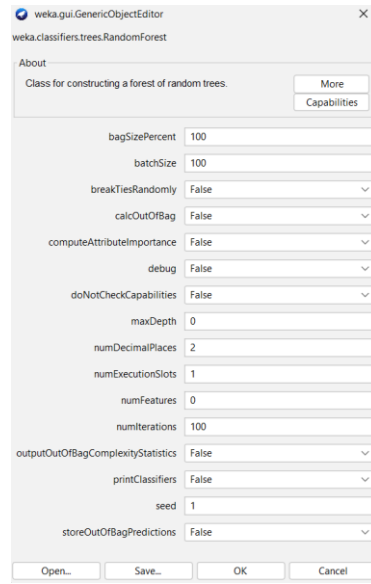
Sınıflandırma sonucu elde edilen sonuç aşağıdaki gibidir.



3. Random Forest Sınıflandırıcı

Bu sınıflandırıcı, birden çok karar ağacı üzerinden her bir karar ağacını farklı bir gözlem örneği üzerinde eğitilir. Bu eğitim sonucunda çeşitli modeller üretir. Bu modeller sonucunda sınıflandırma oluşturmayı sağlar. Kullanım alanlarına örnek vermek gerekirse sınıflandırma ve regresyon problemleri söylenebilir. Bu programlarda kullanılmasının sebeplerinden birisi de kullanımın kolay ve esnek olmasıdır.

Önceki örneklerde yaptığımız gibi **classifier** sekmesine girip trees sekmesinden **RandomForest** sınıflandırıcısını seçiyoruz. Random Forest sınıflandırıcısının parametreleriyle alakalı düşünceleri belirtilen fotoğrafın altında belirttim. Parametreler aşağıdaki gibidir.



The screenshot shows the 'weka.gui.GenericObjectEditor' window for the 'weka.classifiers.trees.RandomForest' classifier. The 'About' tab is selected, displaying the description: 'Class for constructing a forest of random trees.' Below this, there are two buttons: 'More' and 'Capabilities'. The main area lists various parameters with their current values:

Parameter	Value
bagSizePercent	100
batchSize	100
breakTiesRandomly	False
calcOutOfBag	False
computeAttributeImportance	False
debug	False
doNotCheckCapabilities	False
maxDepth	0
numDecimalPlaces	2
numExecutionSlots	1
numFeatures	0
numIterations	100
outputOutOfBagComplexityStatistics	False
printClassifiers	False
seed	1
storeOutOfBagPredictions	False

At the bottom, there are four buttons: 'Open...', 'Save...', 'OK', and 'Cancel'.

Seed parametresi, K-means yöntemine benzer bir şekilde işleme aynı random değerlerden başlanmasını sağlar. **bagSizePercent** parametresi eğitim verisinin bir yüzdesi olarak her çantanın büyüklüğü temsil eder. **calcOutOfBag** çanta dışında kalma hatası oluştuğunda yapılacak hesaplamalardır. Error olmadığı için bu değeri False seçtim. **printClassifiers** sınıflandırma işleminde kullanılan her bir ağacı yazdırmaktır. Çıktı uzadığı için bu parametreyi false olarak belirledim. **numIterations** sınıflandırmadaki ağaç sayısını temsil eden bu parametreyi 100 seçtim. **outputOutOfBagComplexityStatistics** çanta dışı olduğunda gösterilen istatistikleri belirtir. **breakTiesRandomly** parametresi birkaç öznitelik aynı durumda ve iyi ise rastgele seçim yapar. **maxDepth** ağaçların maksimum değerini söyler, 0 olduğunda herhangi bir üst sınır belirtmez. Sonsuz olması için 0 değeri verildi. **computeAttributeImportance** özniteliklerin önem sırasını söyler. **storeOutOfBagPredictions** çanta dışında kalan sınıflandırmaların tutulup tutulmamasını kararlaştırılır.

Yukarıda söylenilen parametrelere göre sınıflandırıcı çalıştırdım. Aşağıdaki görseller sırasıyla Percentage split ve Cross-validation seçenekleri için çıktıları göstermektedir. Bunlara göre ayrı ayrı programı çalıştırdım. Amacım arasındaki başarı oranının farkını gözlemektir.

CROSS-VALIDATION

Correctly Classified Instances	9	64.2857 %
Kappa statistic	0.186	
Mean absolute error	0.4733	
Root mean squared error	0.5221	
Relative absolute error	99.3961 %	
Root relative squared error	105.8227 %	
Total Number of Instances	14	

=== Summary ===

Correctly Classified Instances	9	64.2857 %
Kappa statistic	0.186	
Mean absolute error	0.4733	
Root mean squared error	0.5221	
Relative absolute error	99.3961 %	
Root relative squared error	105.8227 %	
Total Number of Instances	14	

==== Detailed Accuracy By Class ====

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,778	0,600	0,700	0,778	0,737	0,189	0,444	0,669	yes
	0,400	0,222	0,500	0,400	0,444	0,189	0,444	0,385	no
Weighted Avg.	0,643	0,465	0,629	0,643	0,632	0,189	0,444	0,567	

=== Confusion Matrix ===

```
a b  <-- classified as
7 2 | a = yes
3 2 | b = no
```

PERCENTAGE SPLIT

Correctly Classified Instances	4	80 %
Kappa statistic	0.5455	
Mean absolute error	0.3741	
Root mean squared error	0.4	
Relative absolute error	79.133 %	
Root relative squared error	81.4215 %	
Total Number of Instances	5	

=== Summary ===

Correctly Classified Instances	4	80 %
Kappa statistic	0.5455	
Mean absolute error	0.3741	
Root mean squared error	0.4	
Relative absolute error	79.133 %	
Root relative squared error	81.4215 %	
Total Number of Instances	5	

==== Detailed Accuracy By Class ====

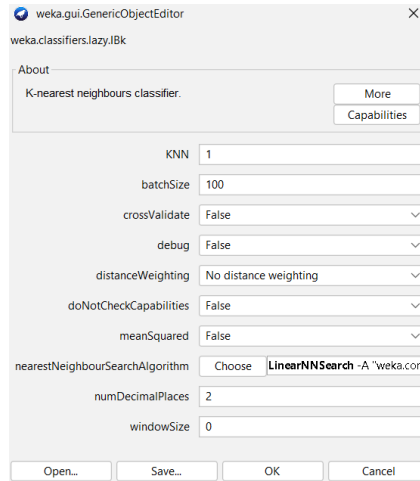
	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1,000	0,500	0,750	1,000	0,857	0,612	1,000	1,000	yes
	0,500	0,000	1,000	0,500	0,667	0,612	1,000	1,000	no
Weighted Avg.	0,800	0,300	0,850	0,800	0,781	0,612	1,000	1,000	

=== Confusion Matrix ===

```
a b  <-- classified as
3 0 | a = yes
1 1 | b = no
```

4. k-NN Sınıflandırıcı

Bu sınıflandırıcı, içerisinde tahmin edilecek değerin bağımsız değişkenlerinin oluşturduğu vektörün en yakın komşularının hangi sınıfta yoğun olduğu bilgisine dayanarak sınıf tahmini yapan bir sınıflandırıcı metodudur.



Yukarıdaki fotoğrafta bu sınıflandırıcıya ait parametreler verilmiştir. Bunları açıklamaya çalıştım.

KNN parametresi en yakın kaç komşu üzerinden hesaplama yapılacağını belirler. Bu değeri 1 olarak belirledim çünkü bu durumda overfit etme oranı artacaktır. Diğer taraftan bakacak olursa bu değerin artmasıyla başarı oranının düştüğünü gördüm, bu sebepten ötürü bu değeri 1 olarak bıraktım. Bu parametre sınıflandırıcıya doğrudan etki ettiği için ve başarı oranını çok fazla değiştirdiği için çok dikkatli seçilmesi gerekir. Sınıflandırıcı için en önemli noktadır. **nearestNeighbourSearchAlgorithm** parametresi içerisinde 5 farklı arama algoritması kapsar.

Bahsedilen algoritmalar örnekler arasındaki uzaklığın ne şekilde hesaplanacağını ifade eder. Hepsini teker teker denediğim ve en iyi sonuç alınan sonucu aldığım LinearNNSearch parametresiyle devam edeceğim.

Tüm bu parametrelerle birlikte alınan çıktı aşağıdaki gibidir.

```

=== Summary ===

Correctly Classified Instances      6           85.7143 %
Kappa statistic                    0.5882
Mean absolute error                 0.2222
Root mean squared error             0.3514
Relative absolute error             46.6667 %
Root relative squared error         73.3799 %
Total Number of Instances          7

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
1,000    0,500    0,833    1,000    0,909    0,645    0,750    0,833    yes
0,500    0,000    1,000    0,500    0,667    0,645    0,750    0,643    no
Weighted Avg.   0,857    0,357    0,881    0,857    0,840    0,645    0,750    0,779

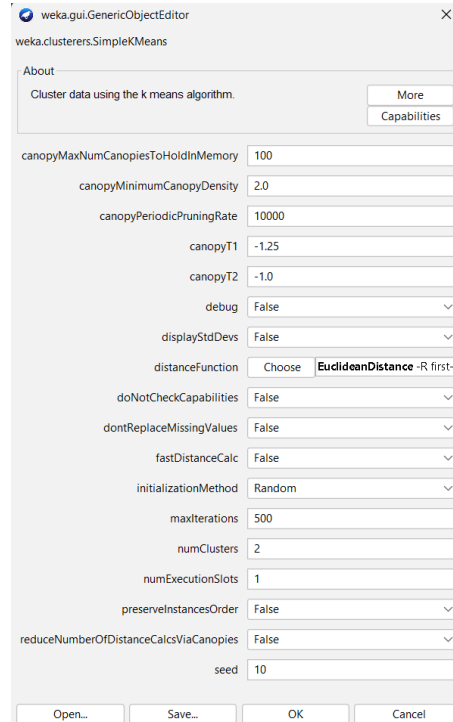
=== Confusion Matrix ===

 a b   <-- classified as
5 0 | a = yes
1 1 | b = no

```

5. k-Means Öbekleyici

k-Means Öbekleyici verileri benzerliklerine göre kümeler. Bu programa WEKA arayüzünden cluster seçeneğine basarak ardından ise SimpleKMeans seçeneğine tıklayarak eriştim.



Yukarıda k-Means ait parametre seçenekleri bulunmaktadır. Bu parametrelerin açıklaması aşağıda yapılmıştır.

Bir ön öbekleme algoritması olan **Canopy**, büyük veri setlerinde kullanılmaya uygundur. Bu işlemleri kısa sürede yapabilir, fakat ödevdeki veri kümesi küçük olduğundan dolayı bu algoritmayı default olarak bıraktım. **Seed** parametresi, seçilecek öbek grupları için rasgele gelme durumunu vermektedir. Parametrenin değişimiyle birlikte öbekler değişmektedir. Sırasıyla değerleri kontrol ettim (10,11,12,13,14,15..). Bu kontrol sonucunda değeri 11 yapmada karar kıldım. **numClusters** işlemdeki öbek sayısını belirten parametredir. Ödevdeki parametreler EVET/HAYIR gruplaması olduğundan dolayı bu değere 2 atadım. **distanceFunciton** uzaklık parametresidir. Öklid uzaklığı olarak seçtim. **maxIteration** iterasyon sayısını belirtir. İterasyon sayısını değiştirdiğimde sonuçta bir fark gözlemlemedim, bu parametre default şekilde bırakıldı.

Yukarıda verilen parametre bilgisi ve değerleri sonucunda çıktım aşağıdaki gibidir.

```
kMeans
=====

Number of iterations: 4
Within cluster sum of squared errors: 13.637860300785825

Initial starting points (random):

Cluster 0: sunny,69,70,FALSE,yes
Cluster 1: overcast,81,75,FALSE,yes

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute      Full Data      Cluster#
              (14.0)      (6.0)      (8.0)
=====
outlook        sunny        sunny    overcast
temperature    73.5714      74.6667    72.75
humidity       81.6429      83.5      80.25
windy          FALSE        TRUE      FALSE
play           yes          no        yes

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      6 ( 43%)
1      8 ( 57%)
```