İçindekiler

VERİ MADENCİLİĞİ.....2

SORU-1: WEKA ile eğitim modelinin kaydedilmesi hakkında bilgi veriniz......3

SORU-2: Sınıflandırma (Classification) hakkında bilgi veriniz......5

SORU-3: Naive Bayes ile Metin Sınıflandırılması hakkında bilgi veriniz......6

SORU-4: Naif Bayes Sınıflandırıcısı (Naive Bayes) hakkında bilgi veriniz......9

SORU-5: Linear Regression (Doğrusal İlkelleme) hakkında bilgi veriniz......12

SORU-6: Imputation (Töhmet) hakkında bilgi veriniz......16

SORU-7: Tip 1 ve Tip 2 hatalar (Type 1 and Type 2 error rates) hakkında bilgi veriniz......20

SORU-8: TF-IDF hakkında bilgi veriniz......22

SORU-9: WEKA hakkında bilgi veriniz......24

VERİ MADENCİLİĞİ

SORU-1: WEKA ile eğitim modelinin kaydedilmesi hakkında bilgi veriniz.

Bu yazının amacı, bir makine öğrenmesi (machine learning) ve veri madenciliği (data mining) aracı olan ve iş zekası (business intelligence) gibi farklı alanlarda kullanımı olan WEKA aracının üzerinde yapılan eğitim modellerinin nasıl kaydedilip, farklı test kümeleri üzerine nasıl uygulandığını anlatmaktır.

WEKA arcını kullanan kişilerin yaşadığı bir durum, WEKA'nın en sık kullanılan ve en kolay ekranı olan Explorer ekranındaki eğitim (train) ve test kümelerinin aynı anda veriliyor olması ve farklı test kümelerinin verilemiyor olmasıdır. Oysaki WEKA'nın en kuvvetli özelliklerinden birisi olan, JAVA ile tam uyum sağlayan yapısı, yazılan kodlarda aynı eğitim modelinin tekrar tekrar eğitilmeden kullanılmasını mümkün kılmaktadır.

Yani modelimizi bir kere eğitip sonra çok farklı testler için kullanabiliriz. Mesela, bir güvenlik firmasının kamera üzerinden yüz tanıma programını yazdığımızı düşünelim. Elimizdeki yüzlerce personelin yüzlerini sisteme tanıtıp modelimizi eğittikten sonra bu eğitilmiş modeli, her yeni gelen kişi için kullanmak gerekir. Aksi halde her yeni kişi için yeniden yüzlerce kişinin yüzünü sistemde eğitmemiz ve bunun için zaman ve donanım kaynaklarını israf etmemiz gerekir ki pek de mantıklı bir yol olduğu söylenemez.

Deneme aşamasında, WEKA'yı konsoldan çağırmak iyi bir çözümdür.

```
java weka.classifiers.trees.J48 -t egitim.arff -d egitilmis.model
```

Yukarıdaki komut ile J48 karar ağacını, verdiğimiz egitim.arff dosyası üzerinden eğitiyoruz ve eğitilmiş modeli egitilmis.model isimli dosyaya kaydediyoruz.

```
java weka.classifiers.trees.J48 -l egitilmis.model -T test.arff
```

Yukarıdaki komutu ise istediğimiz kadar, istediğimiz farklı dosyalar ile çağırabiliriz. Burada ise bir önceki adımda kaydettiğimiz egitilmis.model dosyasını kullanarak test.arff dosyasındaki verilerin sınıflandırılmasını istiyoruz. egitilmis.model dosyasını oluşturduktan sonra ikinci adımı istediğiniz kadar farklı dosyaya uygulayabilirsiniz.

Yukarıdaki, ikinci komutu, istediğimiz kadar farklı test dosyası üzerine uygulamamız ve sonuçlarını almamız mümkündür. Ancak yukarıdaki yaklaşımda bir problem, yapmış olduğumuz eğitim ve test işlemlerinin yazılım ile entegre olamamasıdır.

Bunun için biraz java kodlamaya girip yukarıdaki işlemin aynısını JAVA kodumuza entegre etmeye çalışalım. Bu yaklaşımda, java'nın bir özelliği olan nesne serileme (object serialization) özelliğini kullanacağız.

Ancak öncelikle nesne serileme işlemini kısaca hatıraltalım. Nesne serileme, basitçe bir nesnenin bütün bilgilerinin dizgiye (string) çevrilmesidir. Bu çevirim ne işe yarar derseniz, hafızada (RAM) yaşayan nesne bir kere dizgiye (string) çevrildikten sonra artık dosyaya yazılabilir veya ağ üzerinden farklı bir kaynağa yollanabilir. Biz de bu özelliği, java hafızada bir nesne olarak tuttuğumuz eğitilmiş model üzerine uygulayacağız ve bu sayede nesnenin içindeki eğitilmiş modeli her seferinde tekrar tekrar kullanabileceğiz.

```
// Modelin tanimi
 Classifier cls = new J48();
```

```
// Egitim
Instances inst = new Instances(
                    new BufferedReader(
                      new FileReader("egitimkumesi.arff")));
inst.setClassIndex(inst.numAttributes() - 1);
cls.buildClassifier(inst);

// modelin serilenmesi
ObjectOutputStream oos = new ObjectOutputStream(
                           new FileOutputStream("egitilmis.model"));
oos.writeObject(cls);
oos.flush();
oos.close();
```

Yukarıdaki kodda, öncelikle bir sınıflandırıcı nesnesini tanımlıyoruz. Ardından dosyayı okuyup bir instance nesnesine transfer edip cls.buildClassifier metodu ile verilen inst değişkenindeki nesneyi eğitim için kullanıyoruz. Artık bu satırdan sonra elimizde eğitilmiş bir model bulunuyor. Ardından eğitilmiş ve eğitim verilerini tutan modelimizi serileyerek bir dosyaya kaydediyoruz. Bunun için JAVA'nın sınıflarından birisi olan ObjectOutputStream sınıfının ve bu sınıfın writeObject metodunu kullanıyoruz. Son olarak açtığımız akışları (stream) kapatıyoruz.

Alternatif olarak WEKA'nın 3.3.5 versiyonundan sonraki sürümleri için weka'nın içindeki bir sınıfı da kullanabiliriz:

```
weka.core.SerializationHelper.write("egitilmis.model", cls);
```

Yukarıdaki tek satır ile, ilk kodda bulunan ObjectOutputStream sınıfının tanımı ve writeObject metodunun çağırılması işlemlerini tek bir adımda yapılabilir.

Şimdi gelelim oluşturduğumuz bu modelin test için kullanılmasına. Bunun için tersserileme (deserialization) uygulayarak daha önce dizgiye (string) çevirdiğimiz nesneyi, tekrar dizgiden (string) nesneye geri çevireceğiz. JAVA'nın sınıflarını kullanarak aşağıdaki şekilde bu kodu yazabiliriz:

```
ObjectInputStream ois = new ObjectInputStream(
                          new FileInputStream("egitim.model"));
 Classifier cls = (Classifier) ois.readObject();
 ois.close();
```

Yukarıda, ObjectInputStream sınıfını ve bu sınıfın readObject metodunu kullanarak, daha önce kaydettiğimiz egitim.model dosyasından veriyi okuyarak cls isimli nesneye yükledik. Artık bu nesneyi test kümesini sınıflandırmak için kullanabiliriz. Veya yine alternatif olarak, weka'nın 3.3.5 versiyonu sonrası sürümleri ile gelen aşağıdaki satırı kullanabiliriz:

```
Classifier cls = (Classifier)
weka.core.SerializationHelper.read("egitim.model");
```

SORU-2: Sınıflandırma (Classification) hakkında bilgi veriniz.

Bu yazının amacı, bilgisayar bilimleri ve iş zekası (business intelligence) gibi disiplinlerin ortak çalışma alanlarından olan veri madenciliği (data mining) konusunda kullanılan metotlardan birisi olan sınıflandırma (classification) kavramını açıklamaktır.

Sınıflandırma kavramı, basitçe bir veri kümesi (data set) üzerinde tanımlı olan çeşitli sınıflar arasında veriyi dağıtmaktır. Sınıflandırma algoritmaları, verilen eğitim kümesinden bu dağılım şeklini öğrenirler ve daha sonra sınıfının belirli olmadığı test verileri geldiğinde doğru şekilde sınıflandırmaya çalışırlar.

Veri kümesi üzerinde verilen bu sınıfları belirten değerlere etiket (label) ismi verilir ve gerek eğitim gerekse test sırasında verinin sınıfının belirlenmesi için kullanılırlar.

Konunun daha kolay anlaşılabilmesi için bir örnek üzerinden anlatmaya çalışalım.

Örneğin aşağıdaki şekilde öğrencilerden toplanan bir veri kümemiz bulunsun.

Yaş	Boy	Kilo	Cinsiyet
20	175	70	Erkek
21	179	80	Erkek
19	162	50	Kız
22	169	55	Kız
20	183	90	Erkek
19	181	75	Erkek
21	171	57	Kız

Örneğin problemimizi şu şekilde tanımlayalım. Bir sınıflandırıcı yöntemimiz, yukarıdaki kümeye bakarak verilen yaş boy ve kilo değerlerine göre bir öğrencinin cinsiyetini öğrenecek olsun. Yani yukarıdaki veri kümesini bir eğitim kümesi olarak kullanacağız. Ardından gelen yeni bir kayıt için, yaş, boy ve kilo değerleri verildiğinde, sınıflandırıcımız cinsiyetini otomatik olarak tahmin edecek (prediction).

Çok sayıdaki sınıflandırma algoritmalarından basit birini seçelim. Diyelim ki sınıflandırma algoritmamız, verilen etiketteki değerlerin ortalamasını alacak ve bu ortalama değer, öğrendiği değer olacak. Ardından gelen test değerleri için bulmuş olduğu ortalamaya uzaklığına bakacak ve kime yakınsa o etiketten kabul edecek.

Yukarıdaki veri kümesini iki sınıf için ikiye bölelim ve ortalama değerlerini alalım:

Erkekler için öğrenme işlemimiz aşağıda verilmiştir:

Yaş	Boy	Kilo	Cinsiyet
20	175	70	Erkek
21	179	80	Erkek
20	183	90	Erkek
19	181	75	Erkek
20	179,5	78,75	Ortalama

Aynı işlemin kızlar için olanı aşğaıdaki şekildedir:

Yaş	Boy	Kilo	Cinsiyet

21	171	57	Kız
19	162	50	Kız
22	169	55	Kız
20,66666667	167,3333333	54	Ortalama

Sonuç olarak algoritmamız erkekler için (20,179.5, 78.5) değerlerini öğrenirken, kızlar için (20.66, 167.33, 54) değerlerini öğreniyor. Diyelim ki yeni gelen ve test edilmesini istediğimiz değer de, yaş : 21, boy: 165, kilo 60 değerlerinde bir kişi olsun. Şimdi algoritmamız öğrendiği değerlere göre bu yeni gelen kişinin cinsiyetini tahmin etmeye çalışacak. Basitçe her değere olan mesafeyi hesaplayacak (burada da çok farklı mesafe hesaplama algoritmaları olmasına karşılık biz yine amacımız temel kavramlar olduğu için konuyu basit tutarak gauss mesafesini (gaussian distance) kullanalım).

Algoritmamızın Erkek tanımından öğrendiği değerler ile yeni gelen kişinin mesafesini hesaplayalım:

$$\sqrt{((20-21)^2+(179.5-165)^2+(78.75-60)^2)}=23.72$$

Benzer şekilde kızlar için öğrendiğimiz değere olan mesafesini hesaplamaya çalışalım.

$$\sqrt{((20.66-21)^2+(167.33-165)^2+(54-60)^2)}=6.44$$

Buna göre algoritmamızın verdiği değer, erkeklere olan mesafesinin 23.72 olduğu ve kızlara olan mesafesinin 6.44 olduğudur. Demek ki algoritmamız yeni gelen kişiyi kız sınıfından tanımlamıştır.

İşte sınıflandırma algoritmalarının çalışması, yukarıda da anlatıldığı gibi en basit anlamıyla, 2 aşamadan oluşur.

- Eğitim verisi üzerinden öğrenme
- Öğrenilen değerlerle test verisi üzerinde sınıflandırma

Ancak veri madenciliği ve iş zekası çalışmalarında sınıflandırma sadece çalışmanın bir türü ve ufak bir parçası olmaktadır.

SORU-3: Naive Bayes ile Metin Sınıflandırılması hakkında bilgi veriniz.

Bu yazının amacı, naif bayes sınıflandırıcısının (naive bayes) metinler üzerinde nasıl kullanıldığını açıklamaktır. Oldukça basit ve etkili bir metin madenciliği yöntemi olan naif bayes sınıflandırıcısını anlamak için bir örnek kullanalım.

Örneğin iki metin aşağıdaki şekilde verilmiş olsun:

metin 1 : java bazı bilgisayar mühendisliği bölümlerinde eğitimi verilen bir programlama dilidir.

metin 2 : bazı bilgisayar mühendisliği projelerinde java programlama dili kullanılmaktadır.

Bu metinlerin, kavram metin masfufu (term document matrix) daha önceki bir yazıda gösterilmiştir. Şimdi amacımız yeni gelen aşağıdaki metnin, hangisine daha yakın olduğunu bulmak olsun.

Metin 3 : bilgisayar kavramları eğitimi

Problemi çözmeye başlamadan önce Naive Bayes yönteminin klasik formülünü hatırlayalım:

$$P(A|B)=\frac{p(A\cap B)}{p(B)}$$

Bu klasik formülde bir koşullu olasılığın, nasıl küme olasılıklarına indirgendiği gösterilmektedir.

Metin madenciliğinde (text mining) önemli konulardan birisi de metin içerisinden özelliklerin nasıl çıkarılacağıdır. Bu yazı kapsamında, konuyu dağıtmamak için en basit yöntem olarak kelime seviyesinde çalışacağız. Yani bir kelimenin bir sınıfı belirlemede kullanılmasını hedefliyoruz.

Yine olayı basitleştirmek için yukarıdaki bayes formülünde olduğu üzere 2 sınıf var olduğunu kabul edeceğiz.

Bu durumda herhangi bir kelimenin herhangi bir sınıfta bulunma olsılığı aşağıdaki şekilde gösterilebilir:

$$P(k_i|S)$$

Yani diğer bir deyişle S sınıfı gerçekleşen olaysa, ki kelimesinin bu olayın içinde yer alma olsılığıdır.

Bu durumda naive bayes sınıflandırma yöntemini kullanırsak (lütfen ilgili yazı için tıklayın) kelime bazlı sınıflandırıcımızın aşağıdaki şekilde olduğunu söyleyebiliriz.

$$P(X|S)=\prod_i p(k_i|S)$$

Yani diğer bir deyişle, sistemimizde sınıflandırılmak istenen X metninin S sınıfında olma olasılığı, bu metindeki bütün kelimelerin S sınıfında olma olasılıklarının çarpımıdır.

Neticede elimizdeki bütün sınıflar için metin denenerek en yüksek olasılığa sahip sınıfa aittir denilebilir.

Örneğimize dönecek olursak metinlerdeki kelimeleri listeleyelim:

Terimler \ Metinler	Metin 1	Metin 2
Java	1	1
Bazı	1	1
Bilgisayar	1	1
Mühendisliği	1	1
Bölümlerinde	1	0
Eğitimi	1	0
Verilen	1	0
Bir	1	0
Programlama	1	0
Dilidir	1	0
Projelerinde	0	1
Dili	0	1
Kullanılmaktadır	0	1

Şimdi bu listeye göre metin1′de 9 kelime bulunuyor ve hepsinin frekansı (sıklığı) aynı, dolayısıyla bütün kelimeler için

$$p(k_i|M_1)=\frac{1}{9}$$

oranındadır denilebilir. Aynı durum Metin2 için de geçerlidir ve 7 kelimenin tamamı için:

$$p(k_i|M_2)=\frac{1}{7}$$

değerindedir denilebilir. Neticede yeni gelen ve sınıflandırılmak istenen Metin 3 için bütün kelimelerin değerlerine bakacağız.

M3 = {bilgisayar, kavramları, eğitimi}

$$p(bilgisayar|M_1)=\frac{1}{9}, \quad p(kavramları|M_1)=0, \quad p(eğitimi|M_1)=\frac{1}{9}$$

Burada sık karşılaşılan bir problem olan 0 durumunu düzeltmek için iki yöntemden birisi kullanılır, ya normalleştirme yapılarak bu değer sistemden çıkarılır, ya da çok düşük bir değer konulur. Biz normalleştirme yapalım:

$$p(M_3|M_1)=\sum_{i=1}^{3} p\frac{(k_i|M_1)}{3}$$ şeklinde ortalama alıyoruz.

$$\frac{\frac{1}{9}+0+\frac{1}{9}}{3}=\frac{2}{27}=0.074$$ Olarak bulunur.

Şimdi gelelim ikinci metin için aynı hesabı yapmaya:

$$p(bilgisayar|M_2)=\frac{1}{7}\text{,}\quad p(kavramları|M_2)=0\text{,}\quad p(eğitimi|M_2)=0$$

Bu durumda normalleştirilmiş değer:

$$\frac{\frac{1}{7}+0+0}{3}=\frac{1}{21}=0.0476$$ olarak bulunur. Demek ki yeni gelen metin 3, metin 1′e daha yakındır denilebilir. Sınıflandırmamız neticesinde bir sınıf seçilmesi istenseydi, metin 1′in bulunduğu sınıfta olduğu söylenebilirdi.

SORU-4: Naif Bayes Sınıflandırıcısı (Naive Bayes) hakkında bilgi veriniz.

Bu yazının amacı, literatürde naif Bayes (Naive Bayes) olarak bilinen algoritmanın, sınıflandırma (classification) için kullanılma şeklini açıklamaktır. Herhangi bir sınıflandırma probleminde olduğu gibi, amacımız birden fazla özelliği taşıyan bir yöney (vektör) kullanarak verilen bilgilerden bir eğitim oluşturmak ve bu eğitim neticesinde gelen yeni verileri doğru bir şekilde sınıflandırmaktır.

Sınıflandırma işlemi sırasında, nasıl bir yol izlendiğini bir örnek üzerinden açıklamaya çalışalım.

Örneğimiz, aşağıda verilen tablodaki verilerden bir sınıflandırıcı eğitimi olsun (train).

Kısım	Maaş	Yaş	İş Tecrübesi
Yazılım	3000	26	4
Muhasebe	1500	22	2
Yazılım	5000	30	9
Muhasebe	2000	30	7

Muhasebe	500	18	3
Yazılım	2000	20	2
Yazılım	7000	29	5
Muhasebe	6000	45	15

Yukarıda, sadece muhasebe ve yazılım kısımlarında çalışan kişilerin maaş, yaş ve iş tecrübelerini içeren temsili bir tablo verilmiştir. Buna göre aşağıdaki şekilde bize bir bilgi verilse:

Maaş : 3000, Yaş : 30, Tecrübe : 5yıl

Bu kişinin hangi kısımda çalıştığını acaba bulabilir miyiz?

Öncelikle eğitim ile işe başlayalım sonra da testimizi yaparız. Basitçe veri kümemizideki (data set) Yazılım ve Muhasebe kısımlarının ortalama ve varyans değerlerini hesaplıyoruz. Bu değerler aşağıdaki şekildedir:

Muhasebe	2500	28,75	6,75
Yazılım	4250	26,25	5
Muhasebe	5833333,333	142,25	34,91666667
Yazılım	4916666,667	20,25	8,666666667

Yukarıdaki ilk iki satır ortalama ve ikinci iki satır ise varyans değerleridir. Bu değerleri basitçe Excel ile hesapladık.

Şimdi beklenen değeri hesaplayacağız. Yani gelen test verimizin Muhasebe kısmında birisine ait olması veya Yazılım kısmından birisine ait olması için beklenen durum hesabı yapacağız.

Hesaplamaya geçmeden önce yazının konusu olan naive bayes kavramını hızlıca açıklayalım. Aslında naive bayes sınıflandırıcısı basiçte bütün koşullu olasılıkların çarpımıdır.

Aşağıdaki şekilde gösterilebilir:

$$sınıflandırma(s_1, s_2, \ldots, s_n) = azami_c\, p(K=k) \prod_{i=1}^{n} p(S_i = s_i | K = k)$$

Bu formülde görüleceği üzere s1′den sn'e kadar olan sınıflar arasından bir seçim yapılırken aslında bu sınıfın olasılık değeri ve bu sınıfları yerine getiren k koşulları için çarpımından bir farkı yoktur.

Yani diğer bir deyişle her sınıfın bir koşullu olasılık değeri vardır ve biz sınıflardan hangisine ait olduğunu bulmak için bu koşullu olasılık değerlerini çarparız.

Bu durumda bizim formülümüz aşağıdaki şekildedir denilebilir:

$$beklenti(Yazılım) = \frac{P(Yazılım)\, p(maaş|yazılım)\, p(Yaş|yazılım)\, p(iş\ tecrübesi|yaz}{normalleştirme}$$

Yani bir kişinin yazılım kısmında olduğunu anlamak için öncelikle yazılım kısmında olan kişilerin oranını buluyoruz, P(Yazılım), ardından yazılım kısmındaki kişiler için verilen maaş, yaş ve iş tecrübesine göre koşullu olasılıklarını bulup bunu normalleştirme değerine bölüyoruz.

Benzer şekilde muhasebe kısmındaki kişilere ait beklenti de aşağıdaki gibi hesaplanabilir

$$beklenti\,(Muhasebe) = \frac{P(Muhasebe)\,p(maaş|Muhasebe)\,p(Yaş|Muhasebe)\,p(iş\ tecrübesi|Muhasebe)}{normalleştirme}$$

Buradaki fark, verilen bilgilerin muhasebe kısmındaki kişilere göre koşullu olasılığının alınmasıdır.

Normalleştirme değeri ise sistemimizde bulunan bütün ihtimalleri içeren ve beklenti değerlerini normalleştiren değerdir. Aşağıdaki şekilde hesaplanabilir:

normalleştirme = P(Muhasebe) p(maaş | Muhasebe) p (Yaş | Muhasebe) p (iş tecrübesi | Muhasebe) + {P(Yazılım) p(maaş | yazılım) p (Yaş | yazılım) p (iş tecrübesi | yazılım)

Şimdi iki sınıfımız olduğu ve aslında ikisi arasında seçim yapacağımız için, iki sınıfı da bölen ve pozitif çıkmasını beklediğimiz normalleştirme değerini göz ardı edebiliriz.

Sırasıyla olasılıkları hesaplayalım:

P(Yazılım) = 8 kişiden 4′ü yazılım kısmında

P(Yazıılm) = 0.5

Benzer şekilde,

P(Muhasebe) = 0.5

olarak buluruz. Ardından koşullu olasılık değerlerini hesaplayacağız. Bu aşamada gauss dağılımını kullanmak isteyelim (farklı dağılımlar kullanılabilir ve başarı bu dağılıma göre değişebilir) ve dağılım fonksiyonunu hatırlayalım:

$$p(maaş|yazılım) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

varyans, maaş ve ortalama değerlerini yazacak olursak:

$$\frac{1}{\sqrt{2\pi\,4.91E6^2}}\exp\left(\frac{-(3000-4250)^2}{24.91E6^2}\right) = \frac{1}{1.46E7}\exp\frac{(1250^2)}{4.82E13} = 6.84E\text{-}8$$

Yani sonuç olarak 0.0000000687′e yakın bir değer bulunuyor. Bunun anlamı, verilen 3000 lira maaş ile çalışan kişinin Gauss dağılımında (dağılımın toplam alanının 1 olduğunu hatırlayınız), yazılım kısmında çalışan birisi olması ihtimalinin 6.87E-7 olduğudur. Daha doğru bir ifadeyle, bir kişinin yazılımcı olduğunu kabul edersek (verilen koşul) bu kişinin

3000 lira maaş alma durumunu hesaplamış olduk. Şimdi aynı maaş değerine sahip kişinin muhsabe kısmında çalışma olasılığını hesaplayalım.

$$\frac{1}{\sqrt{2\pi 5.83E6^2}}\exp\left(\frac{-(3000-2500)^2}{25.83E6^2}\right)=\frac{1}{4,91E7}\exp\frac{(-500^2)}{6,79E13}=8,11E\text{-}8$$

Yukarıdaki ikinci hesaplamanın sonucuna bakarak aslında bu maaşın, yazılım kısmındaki kişilere daha uygun olduğunu söyleyebiliriz.

Benzer hesaplamaları diğer koşullu olasılık durumları için de yaparsak aşağıdaki gibi bir tablo elde edebiliriz:

	Maaş	Yaş	Tecrübe
Muhasebe	6,84074E-08	0,002805118	0,011414151
Yazılım	8,11614E-08	0,019705849	0,046043474

Sonuçta naive bayes yöntemine göre bu verilen olasılıkların çarpımlarını alacağız:

$$beklenti\,(Yazılım)=\frac{P(Yazılım)\,p(maaş|yazılım)\,p(Yaş|yazılım)\,p(iş\ tecrübesi|yazılım)}{normalleştirme}$$

olduğunu hatırlayalım. Bu durumda beklenti(yazılım) aşağıdaki şekilde yazılabilir:

$$beklenti\,(Yazılım)=\frac{0.5\,x\,6,84E\text{-}8\,x\,0.0028\,x\,0,0114}{normalleştirme}=1,9E\text{-}12$$

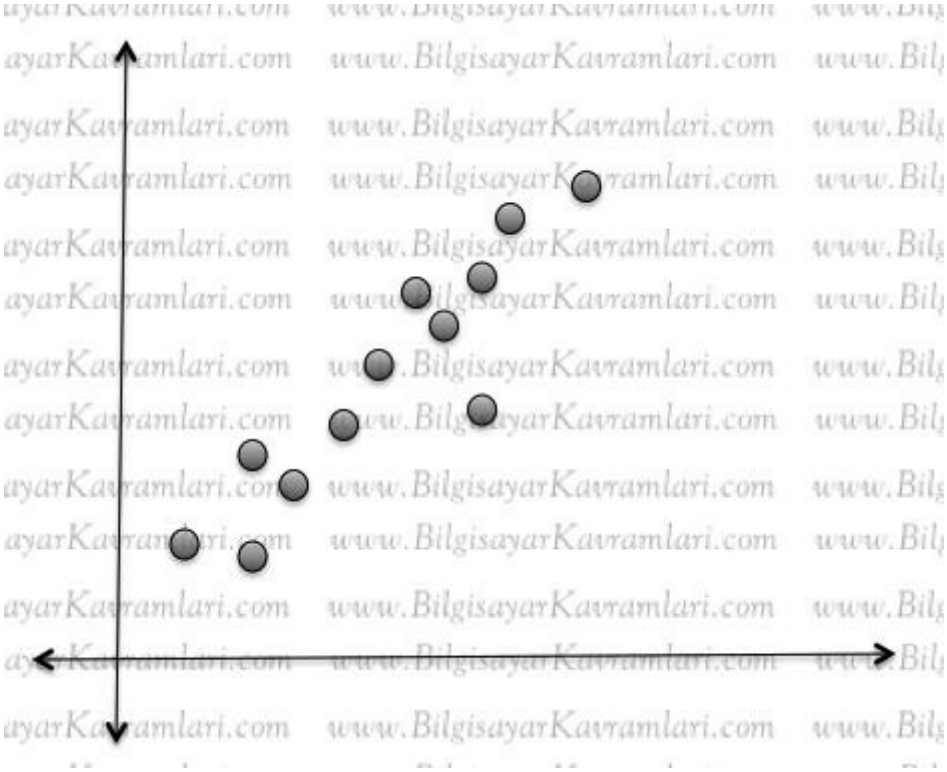
Burada normalleştirme değeri daha önce de belirtildiği üzere hesaba katılmamıştır. Benzer şekilde muhasebe kısmına ait beklenti de aşağıdaki şekilde hesaplanabilir.

$$beklenti\,(Muhasebe)=\frac{0.5\,x\,8,11E\text{-}8\,x\,0.0019\,x\,0,046}{normalleştirme}=3,68E\text{-}11$$

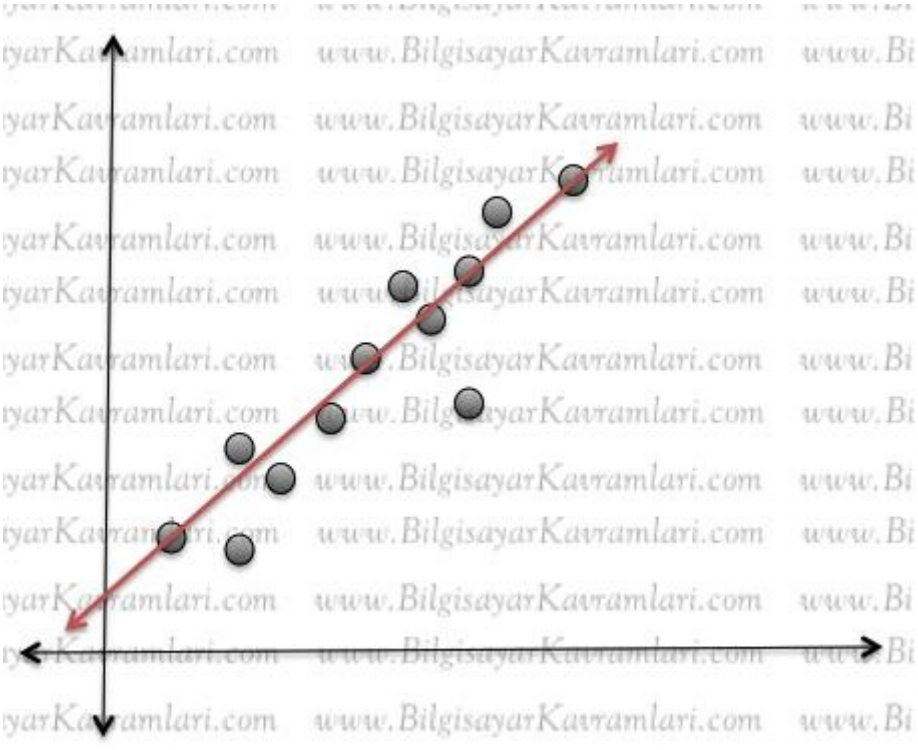
Görüldüğü üzere muhasebe beklentisi, yazılım beklentisinin yaklaşık 20 misli daha yüksek çıkmıştır. Demek ki naive bayes sınıflandırmasına göre bu kişinin muhasebe kısmında çalıştığını söyleyebiliriz, en azından beklentimiz bu yönde olur.

SORU-5: Linear Regression (Doğrusal İlkelleme) hakkında bilgi veriniz.

Bu yazının amacı, doğrusal ilkelleme yöntemini (linear regression) açıklamaktır. Basitçe bir veri kümesinin iki boyutlu bir uzayda dağılıdığını düşünelim.



Veri kümemizdeki değerlerin iki boyutlu uzayda, yukarıdaki şekilde gösterildiğini kabul edelim. Şimdi doğrusal ilkelleme ile amaçlanan bu noktaların tamamına en yakın geçen doğruyu elde etmektir. Örneğin aşağıdaki şekilde olabilir:



Her doğrunun bir formülü olduğu gibi bu doğrunun da karakteristik bir şekilde

y= ax + b denklemine uygun bir formülü, daha doğrusu bir (a,b) ikilisi bulunacaktır.

İşte doğrusal ilkelleme, çok sayıdaki karmaşık hesaplama ve ölçümlere dayalı verinin basit bir doğruya indirgenmesi (ilkellenmesi) olarak düşünülebilir. Buradaki ama, veri kümesindeki verilerin değerlerinden böyle bir doğru denklemini elde edebilmektir.

Ardından bu doğru çeşitli amaçlar için kullanılabilir. Örneğin herhangi bir x değeri için y değerinin bulunması artık doğru denklemi ile mümkün olacağından, eksik verilerin töhmeti (imputation) veya bütün veri kümesinin tutulması yerine basitçe sadece doğrunun tutulması ve işlenmesi veya kesitirim (estimation) ve öngörü (forecasting) problemlerinin çözümünde (örneğin gelecek seneki satış oranları veya deprem tahminleri gibi problemler) ve daha pek çok veri madenciliği probleminde kullanılabilir.

Buraya kadar doğrusal ilkelleme (linear regression) konusuna hızlı bir giriş yaptıktan sonra veri kümesinin nasıl ilkellendiğini açıklayabiliriz.

Aslında doğrusal ilkelleme verilen n adet nokta için ki bu noktaları {yi,xi} çifti olarak yazabiliriz, i = 0,1,2 … n olmak şartıyla.

y =ax+b şeklindeki karakteristik denklemde en az hata miktarına sahip a,b ikilisini bulmak istiyoruz.

Hata miktarını ise noktaların bu doğruya olan mesafesi olarak düşünürsek formülümüz aşağıdaki şekilde olacaktır.

$$asg\,H(a,b)\,için\,H(a,b)=\sum_{i=1}^{n} h_i^2=\sum_{i=1}^{n}(y_i-a-bx_i)^2$$ Olarak yazılabilir.

Burada H ile gösterilen fonksiyon, a,b ikilisi için bütün noktaların uzaklığının ölçülüdüğü hata miktarıdır. Elbette bu bir çift fonksiyon olmalıdır yani eksi hata olmaz. Dolayısıyla kare alınmıştır. Her nokta için ayrı ayrı hatanın hesabı ise, y-a-bx değeri olarak hesaplanabilir.

İşte amacımız bu hata değerini en asgari seviyeye indirmektir.

Örnek

Konuyu bir örnek üzerinden açıklamaya çalışalım.

Örneğin 4 nokta koordinatı aşağıdaki şekilde verilmiş olsun.

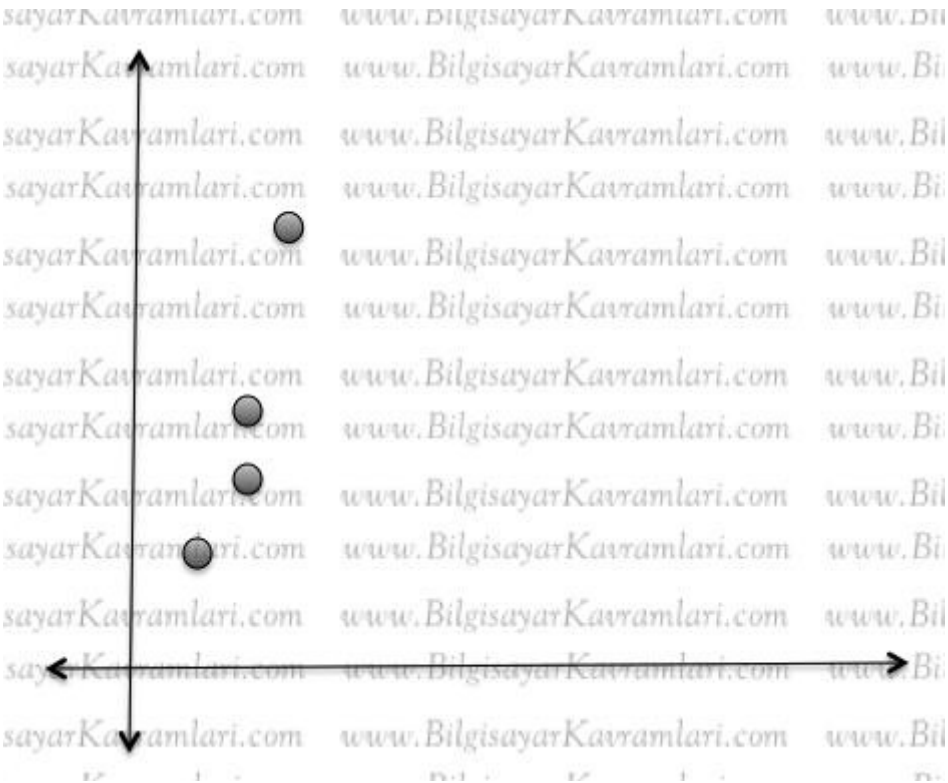
(1,2)

(2,3)

(3,7)

(2,4)

Konuyu daha iyi anlayabilmek için noktaları koordinat sisteminde gösterelim:



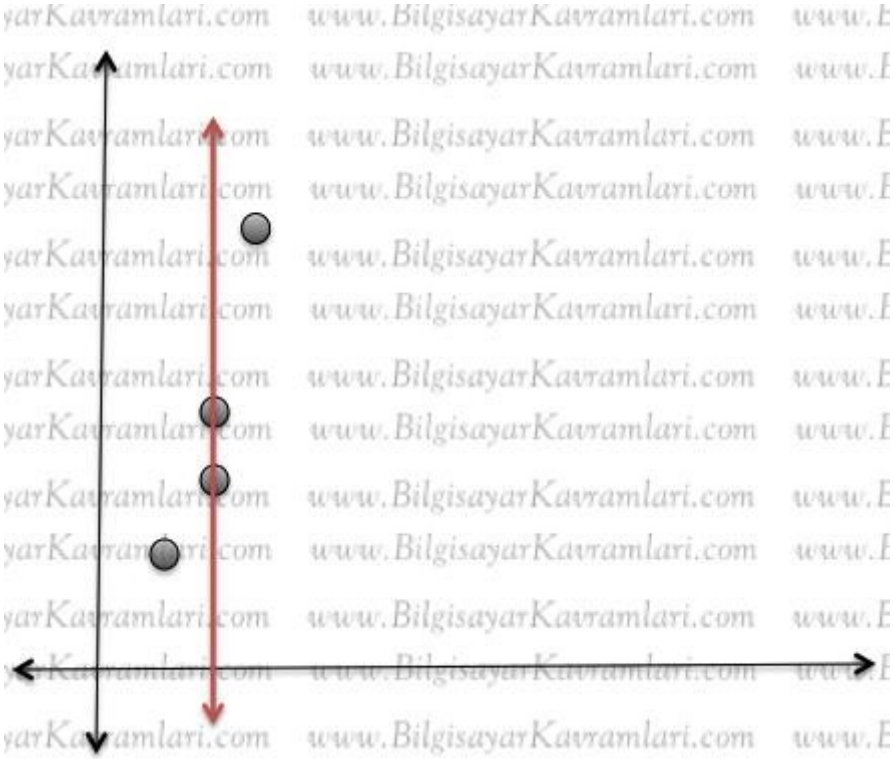
Şimdi bütün bu noktalara en yakın geçen doğru denklemini bulmaya çalışalım:

Sadece örnek olması için hızlıca aşağıdaki şekilde nümerik bir kod yazdım ve çalıştırdım:

Neticede çalışma sonucu a=0, b=2 olarak bulundu. Bunun anlamı aşağıdaki doğrudur:

View Code C

```
#include <stdio.h>
int main(){
int n[4][2] = {{1,2},{2,3},{3,7},{2,4}};
int h =1000;
int atemp,btemp;
for(int a= 0;a <10;a++){
   for(int b= 0;b <10;b++){
      for(int i = 0;i<4;i++){
        int htemp=(n[i][1]-a-b*n[i][0]) *(n[i][1]-a-b*n[i][0]) ;
        if(htemp < h){
                h = htemp;
                atemp = a;
                btemp = b;
        }
     }
   }
}
printf("a : %d b : %d",atemp,btemp);
}
```



Kodun çok kabaca ve sadece fikir vermek için yazıldığını hatırlatır ve bulunan değerlerin sadece tam sayılar ile sınırlı olmasından dolayı yukarıdaki gibi bir sonuç çıktığını söylemek isterim.

SORU-6: Imputation (Töhmet) hakkında bilgi veriniz.

Bu yazının amacı, bilgisayar bilimlerinde özellikle veri madenciliği (data mining) konularında eksik verilerle karşılaşılması halinde bir çözüm olarak bu eksik verilerin töhmet edilmesi (yerine uygun verilerin üretilmesi, imputation) yöntemini açıklamaktır.

Töhmet, sözlükte olmayan birşeyin yüklenmesi anlamındadır. Örneğin olmayan bir suçun birisine yüklenmesine töhmet altında bırakmak denilebilir. Bu anlamda vir veri kümesi (data set) üzerinde çalışılırken bazı sebeplerden dolayı verilerin eksik olması halinde bu verilerin uygun başka sayılarla tamamlanması sağlanabilir.

Genelde verinin hatalı okunması, veri kaynağında yaşanan bozulma gibi sorunlar veya bazı verilere erişim zorluğu eksik verilere sebep olabilmektedir.

Bu verilerin eksik olması durumu genelde sorunlara sebep olur. Örneğin çoğu hazır istatistik paketleri (SAS, SPSS, Weka veya r-project gibi) bu tip durumlarda sorunlar yaşamaktadır. Gerçi bu paketlerin ücretli ve gelişmiş olanlarının çoğunda (SAS veya SPSS gibi) töhmet modülleri (imputation) bulunmakta ve bu işi otomatik olarak yapabilmektedirler ancak bu yazı kapsamında gerek bu modüllerin nasıl çalıştığını anlamaya çalışacağız gerekse bu işlemi elle yapmak istediğimizde nasıl müdahale etmemiz gerektiğini açıklamaya çalışacağız.

Töhmet yöntemleri duruma ve beklentilerimize göre çeşitlilik arz eder ve halen üzerinde çalışılmaktadır. Bunlardan çok bilinen bazılarını saymamız gerekirse:

- Sıcak deste (hot deck)
- Soğuk deste (cold deck)
- Liste boyunca silme (listwise deletion)
- Eşlerin silinmesi (pairwise deletion)
- Ortalama töhmet (mean imputation)
- İlkelleme töhmedi (regression imputation)
- Son gözlemin taşınması (last observation carried forward)
- Olasılıksal töhmet (stochastic imputation)
- Çoklu Töhmet (Multiple imputation)

Yukarıdaki bu kavramları kısaca açıklamaya çalışalım. Öncelikle sıcak deste ve soğuk deste (hot deck , cold deck) algoritmaları için ne yazık ki tam bir mutabakat olmadığını ve çeşitli versiyonları bulunduğunu belirtmek isterim (bkz. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3130338/) Bu yüzden bu iki algoritmayı atlayacağım ancak hot deck (sıcak deste) için kısaca rast gele bir verinin seçilerek eksik olan veri yerine töhmet edildiğini söylememiz yeterli olacaktır. Bu rast gele seçim ise oldukça tartışmalı bir konu. ve diğer metotları kısaca açıklayacağım.

Liste boyunca silme (listwise deletion)

Örneğin bir veri tabanından okunan tabloda bazı değerlerin eksik olduğunu düşünelim:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—-	–	Muhasebe
Veli Demir	43	Yönetim

Yukarıdaki tabloda sondan ikinci ve üçüncü satırlarda kayıp bilgi bulunmaktadır. Bu durumda, örneğin çalışanların yaş ortalamalarının alınması istendiğinde hata ile karşılaşılacaktır.

Liste boyunca silme yaklaşımında (listwise deletion) bu kayıp veri içeren satırların tamamı tablodan temizlenir ve tablo aşağıdaki hale getirilerek işlem sonuçlandırılır.

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Veli Demir	43	Yönetim

Artık bu tablo üzerinde istenilen işlemler yapılabilir.

Eşlerin silinmesi (Pairwise deletion)

Bu yöntemde bütün tablonun temizlenmesi yerine gerekli olan işlem sırasındaki eksik veriler temizlenir. Örneğin bir önceki tabloya geri dönecek olursak:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—-	–	Muhasebe
Veli Demir	43	Yönetim

Ve bu tablo üzerinde çalışanların yaşının ortalaması istenmişti. Bu durumda, bu soruya özel olarak sadece sondan ikinci satırın silinmesi yeterlidir:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
Veli Demir	43	Yönetim

Artık istenen işlem yani yaşların ortalaması çalıştırılabilir.

Bu yöntemde farklı bir işlem yapılmak istendiğinde, örneğin çalışanların kısımlara göre dağılım grafiği istendiğinde ise işlem yapacağımız tablo aşağıdaki şekilde olacaktır:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
—-	–	Muhasebe
Veli Demir	43	Yönetim

Görüldüğü üzere sadece o işlem için problem çıkaran satır silinmiş diğer satırlar farklı kolonlarında eksik veri bulunmasına rağmen saklanmıştır.

Bu yöntemin menfi yanı, her işlem için farklı veri kümesi üretiliyor olması ve hatta her işlem sonucunda farklı sayıda veri ele alınıyor olmasıdır. İşlemler sonucunda verilerin karşılaştırılma güçlüğü ortaya çıkabilir.

Ortalama Töhmet (Mean imputation)

Bu yöntemde, bir eksik verinin töhmeti sırasında ortalama değer hesaplanır. Örneğin yine aynı tablo üzerinden töhmet yaklaşımını izah edelim:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar

Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—-	–	Muhasebe
Veli Demir	43	Yönetim

Şayet bu tablodaki ortalama yaş için eksik olan satırların töhmeti söz konusuysa bu durumda satırların silinmesi yerine veri üretilir. Örneğimizdeki yaşların ortalaması:

$$ortalama\ yaş = \frac{33+23+26+33+43}{5} = 31.6$$

olarak bulunur. Yaşın ondalıklı sayı olması mümkün olmadığı için 32 olarak yuvarlanarak kabul edilebilir ve veri kümemiz aşağıdaki şeklini alır:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—-	32	Muhasebe
Veli Demir	43	Yönetim

Bu işlemin ardından artık istenen veri madenciliği yöntemleri uygulanabilir.

Bu yöntemin dez avantajı ise, çok büyük veriler de uygulanma zorluğudur (örneğin güncel problemlerin artık haritalama-indirgeme (map reduce) ortamlarında işlendiği düşünülürse, böyle bir ortamda kullanılamaz). Ayrıca işlenmesi için bütün verinin hafızaya yüklenip hesaplama yapılması zorluğu da bulunmaktadır.

Son olarak veri eksikliğinin çok fazla olduğu durumlarda üretilen verilerin sağlığı problem çıkarmaktadır.

İlkelleme Töhmeti (Regression Imputation):

Bu yöntemde, mevcut veriler üzerinden fonksiyonel bir ilkelleme yapılır (örneğin doğrusal ilkelleme (linear regression)) ve ardından ilkellenen fonksiyondan üretim yapılarak eksik veri doldurulur.

Son Gözlemin Taşınması (Last Observation Carried Forward):

Bu yöntemde, veri kümesindeki eksik veriler, kendilerinden bir önceki veri kopyalanmak marifetiyle töhmet edilirler. Örneğimize geri dönecek olursak:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe

Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	——
—-	–	Muhasebe
Veli Demir	43	Yönetim

Veri kümesindeki verilerin son gözlemin taşınması yöntemiyle töhmet edilmiş hali aşağıdaki şekildedir:

İsim	Yaş	Kısım
Şadi Evren ŞEKER	33	Bilgisayar
Ali Demir	23	Muhasebe
Cem Yıldız	26	Bilgisayar
Ahmet Yılmaz	33	Bilgisayar
Ahmet Yılmaz	33	Muhasebe
Veli Demir	43	Yönetim

Artık bu veri kümesi üzerinde veri madenciliği yöntemleri uygulanabilir.

Bu yöntemin dez avantajı ise bazı marjinal noktaların çoğaltılması olarak görülebilir. Örneğin Yaş sütununda marjinal olarak 70 yaşında bir çalışan varsa bu kişinin kopyalanması, veri madenciliği sonuçlarını menfi etkileyebilir.

Son iki yöntemimiz olan istatistiksel töhmet (stochastic imputation) ve çoklu töhmet (multiple imputation) için şunları söylememiz yeterli olacaktır.

Öncelikle mevcut veri kümesi üzerinden bir istatistiksel dağılım elde edilir (örneğin ilkelleme (regression) burada kullanılabilir) ardından bu dağılım sayesinde eksik verileri dolduran bir bağlantı kullanılır. Çoklu töhmette ise bu işlem birden fazla kere yapılarak her bir veri kümesi daha sonra kullanılmak üzere saklanır. Ayrıca her veri kümesinin hata miktarı hesaplanır. Ardından bu veri kümelerinin ortalama değerleri alınarak nihai veri kümesi elde edilir. Buradaki tek ayrıntı standart hatanın (standard error) hesaplanması sırasında iki veri kümesinin birleştirilmesi için iki standart hata miktarı toplanıp karekökleri alınır.

Kısacası çoklu töhmet, daha iyi sonuç elde etmek için istatistiksel töhmetin birden fazla kere çalıştırılması olarak düşünülebilir.

SORU-7: Tip 1 ve Tip 2 hatalar (Type 1 and Type 2 error rates) hakkında bilgi veriniz.

Bu yazının amacı, özellikle istatistik konusunda geçen tip 1 ve tip 2 hataları (type 1 and type 2 errors) açıklamaktır. Konu aslında, bir tahmin ve gerçekleşen durum arasında yaşanmaktadır ve istatistikte bulunan null hypothesis (yokluk hipotezi) üzerine kuruludur.

Yazının istatistik ile ilgili kısmına geçmeden önce biraz felsefe bilgimizi tazeleyerek yokluk hipotezinden bahsetmek istiyorum.

Basitçe olmayana ergi (proof by contradiction, burhan-ı mütenakis) yapısında bir ispat yöntemidir. İspatlanmak istenen istatistiksel olgu bir hipotez olarak ortaya atılır ve buna

alternatif hipotez (alterantive hypothesis) ismi verilir. Ardından bu alternatif hipotezin tam tersini gösteren ve yokluğu ispat edilmesi amaçlanan yokluk hipotezi (null hypothesis) konulur.

Bu durumu bir örnek üzerinden açıklayalım. Örneğin veri tabanı teorisinde sıkça verilen ve üretilebilir veri konusunda geçen iki alanı ele alalım. Diyelim ki bir kişinin yaşı ve doğum günü arasında ilişki olduğunu, bu iki değerin birbirine bağlı olduğunu iddia ediyoruz. Bu bizim alternatif hipotezimiz olmuş oluyor.

Bu durumda ilişki olmadığını göstermek de yokluk hipotezimiz olur.

Bu iki hipotezi aşağıdaki şekilde de gösterebiliriz:

Alternatif Hipotez: $Doğum\ Günü\ Sayısı = a\ Yaş + b$

Burada ifade edilen değer, doğum günleri ile yaş arasında doğrusal bir bağlantı olduğudur (linear, affine). Şimdi yokluk hipotezimizi yazalım (null hypothesis)

$$Doğum\ günü\ sayısı\ artarken\ yaşın\ değişmemesi\ (veya\ tam\ tersi)$$

Yani iki artış arasında bir bağlantı olmaması.

Ardından yapılan çok sayıdaki deneme ile görüyorsunuz ki yaş arttıkça doğum günü kutlamalarının sayısı artıyor veya doğum günü kutlamalarının sayısı arttıkça yaş ilerliyor.

Ardından diyebiliriz ki yokluk hipotezi yoktur (nullify)

Şayet yokluk hipotezinin yok olduğunu gösterebilirsek (istatistiksel olarak) o zaman alternatif hipotezimizin doğru olduğunu söyleyebiliriz. Yani doğum günü kutlaması ve yaş arasında pozitif bağlantı vardır denilebilir.

Bu konu aslında felsefede pek çok konuyu beraberinde getirmektedir. Örneğin Karl Popper'ın iddialarından birisi gerçekte alternatif ve yokulk hipotezlerinin birbirinin tersi olmasının gösterilemesi zor olduğudur.

Ancak biz konumuzla ilgili olan bu kısmı ile iktifa edeceğiz. Şimdi gelelim tip 1 ve tip 2 hata miktarlarına.

Öncelikle rastgele bir süreçten bahsediyoruz demektir (random process) ve bu süreçte beklenen ve gerçekleşen olaylar bulunuyor demektir. Aslında tam olarak yokluk hipotezimizin yok olduğunu ispatlamaya çalıştığımızı düşünelim:

	Beklenen (Expected)		
Gerçekleşen (Measured)		Müsbet	Menfi
	Doğru	TP (DMü)	TN (DMe)
	Yanlış	FP (YMü)	FN (YMe)

Beklentimiz müspet (pozitif) veya menfi (negatif) yönde olabilir. Beklentimize bağlı olarak gerçekleşen olay (veya ölçümlerimiz) de doğru veya yanlış olabilir.

Bu durumu bir örnek üzerinden açıklayalım. Örneğin bir e-posta koruma sistemi yazıyoruz ve sistemdeki izinsiz gönderileri (spam mail) tespit etmek istiyoruz. Yazılımımız bu gönderileri tespit edip engelleyecek. Yazılımı hazırlayıp bir süre test ettikten sonra aşağıdaki gibi bir tablo hazırlanabilir.

	İstenmeyen Tahmini		
Gerçekten İstenmeyen		İstenmeyen	İstenen
	Doğru	TP (DMü)	TN (DMe)
	Yanlış	FP (YMü)	FN (YMe)

Buna göre örneğin bizim istenmeyen posta olarak sınıflandırdığımız ve gerçekte istenen postalar TP (True Positive), bizim istenen olarak sınıflandırdığımız ancak gerçekte istenmeyen olanlar TN (True Negative), bizim istenen diye sınıflandırdığımız ancak gerçekte istenmeyen olanlar FP (False Positive) ve son olarak bizim istenen diye sınıflandırdığımız ancak gerçekte istenmeyen olanlar da FN (False Negative) olarka isimlendirilebilir.

Bunlardan FN (False Negative) Tip 2 hata oraın (Type 2 error rate) ve FP (False Positive) ise Tip 1 hata (Type 1 error rate) olarak isimlendirilir.

SORU-8: TF-IDF hakkında bilgi veriniz.

Bu yazının amacı, metin madenciliği (text mining) olarak da geçen ve doğal dil işleme (natural language processing) ve veri madenciliği (data mining) konularının ortak çalışma alanı olan metinler üzerinde istatistiksel incelemeler konusunda kullanılan TFIDF kavramını açılamaktır. TF-IDF kavramı IR (information retrieval, bilgi getirimi) gibi konuların altında bir sıralama (ranking) algoritması olarak sıkça geçmektedir.

İngilizcedeki Term Frequency – Inverse Document Frequency (Terim frekansı – ters metin frekansı) olarak geçen kelimelerin baş harflerinden oluşan terim basitçe bir metinde geçen terimlerin çıkarılması ve bu terimlerin geçtiği miktara göre çeşitli hesapların yapılması üzerine kuruludur.

Klasik olarak TF yani terimlerin kaç kere geçtiğinden daha iyi sonuç verir. Kısaca TF-IDF hesabı sırasında iki kritik sayı bulunmaktadır. Bunlardan birincisi o anda ele alınan dokümandaki terimin sayısı diğeri ise bu terimi külliyatta içeren toplam doküman sayısıdır.

Örnek:

Konuyu bir örnek üzerinden açıklayalım:

Örneğin 100 dokümandan oluşan bir külliyatımız olsun ve TF-IDF hesaplamak istediğimiz kelime de “şadi” olsun. Bu durumda birinci dokümana bakıp “şadi” kelimesinin kaç kere geçtiğini sayarız. Diyeli ki 4 kere geçiyor olsun. Ardından külliyatımızdaki 100 dokümandan kaçında “şadi” kelimesi geçiyor diye bakarız. Diyelim ki 10 dokümanda bu kelime geçiyor

olsun (dikkat edilecek husu kelimenin geçip geçmediğidir diğer dokümanlarda kaç kere geçtiğinin bir önemi yoktur).

Şimdi TF ve IDF değerlerini ayrı ayrı hesaplayacağız ve sonra bu iki değeri çarpacağız, önce TF hesabına bakalım:

TF hesabı için ihtiyacımız olan bir diğer değer ise o andaki dokümanda en fazla geçen terim sayısıdır. Örneğin o anda baktığımız dokümanda en fazla geçen terimimizin sayısı da 80 olsun.

İlk hesaplama dokümanda bizim ilgilendiğimiz kelimenin en fazla geçen kelimeye oranıdır. Yani kelimemiz 4 kere geçtiğine ve en fazla geçen kelimemiz de 80 kere geçtiğine göre ilk oranımız (ki bu oran aynı zamanda TF (term frequency, terim frekansı) olarak tek başına da anlamlıdır)

TF = 4 / 80 = 0,05 olarak bulunur.

Ardından IDF değerini hesaplayalım. Bunun için basit bir bölme işlemi yapılacak ve logaritması alınacaktır.

IDF = log (Toplam Doküman sayısı / Terimi içeren doküman sayısı)

Buna göre IDF için toplam 100 dokümandan 10 dokümanda aradığımız kelime "şadi" geçtiğine göre

IDF = log (100 / 10) = log (10) = 1 olarak bulunacaktır.

IDF hesabı sırasında bir iki noktaya dikkat etmek gerekir. Öncelikle logaritmanın tabanının bir önemi yoktur. Amaç üssel fonksiyonun tersi yönde bir hesap yapmaktır. Doğal logaritma kökü e, 2 veya 10 gibi sayılar en çok kullanılan değerlerdir. Genelde TF-IDF değerinin kıyas için kullanıldığını ve diğer terimlerin TFIDF değerleri ile kıyaslandığını düşünecek olursak hepsinde aynı tabanın kullanılıyor olması sonucu değiştirmeyecektir.

Diğer dikkat edilecek bir husus ise IDF hesabı sırasında geçen "terimi içeren doküman sayısı" değeridir. Bu değer hesaplama sırasında paydada yer almaktadır ve bu değerin 0 (sıfır) olma ihtimali vardır. Bu durumda sonuç sıfıra bölüm belirsizliğine götürebileceğinden genelde bu değere 1 eklemek sıkça yapılan bir programlama yaklaşımıdır.

Neticede elde ettiğimiz TF = 0,05 ve IDF = 1 değerlerini çarpıyoruz ve terimimizin TF-IDF değeri aşağıdaki şekilde bulunuyor:

TF-IDF = TF x IDF = 0,05 x 1 = 0,05

Yukarıda kullandığımız formülleri aşağıdaki şekilde de açıkça yazmak mümkündür:

$$w_{i,d} = tf_{i,d} \times \log(n / df_i)$$

Yukarıdaki gösterimde, i terimi için ve d dokümanı için hesaplama yapılmaktadır. Öncelikle TF hesaplanır ki bu basitçe terimin o dokümanda kaç kere geçtiğinin en fazla geçen terime oranı şeklinde hesaplanabilir:

$$tf_{i,d} = aza(fr_{i,d} / df_i)$$

Yani i terimi için d dokümanındaki terim frekansı (term frequency), i teriminin d dokümanındaki tekrar sayısının o dokümandaki en yüksek tekrar sayısına sahip terimin tekrar sayısına oranıdır. Veya bu oranların en yükseğidir.

Yukarıda verilen TF-IDF formülünde ayrıca n toplam doküman sayısını df ise doküman frekansını vermektedir ve df aslında i teriminin kaç farklı dokümanda geçtiğinin sayısıdır.

Son olarak TF-IDF yönteminin diğer yöntemlere göre farkını açıklamaya çalışalım. TF-IDF ile bir terimin kaç kere geçtiği kadar kaç farklı dokümanda da geçtiği önem kazanır. Örneğin sadece bir dokümanda 100 kere geçen bir terimle 10 farklı dokümanda onar kere geçen terimin ikisi de aslında toplamda 100 kere geçmiştir ancak TF-IDF ikincisine yani daha fazla dokümanda geçene önem verir.

SORU-9: WEKA hakkında bilgi veriniz.

WEKA, bilgisayar bilimlerinin önemli konularından birisi olan makine öğrenmesi (machine language) konusunda kullanılan paketlerden birisinin ismidir. Waikato üniversitesinde açık kaynak kodlu olarak JAVA dili üzerinde geliştirilmiştir ve GPL lisansı ile dağıtılmaktadır. İsmi de buradan gelir ve Waikato Environment for Knowledge Analysis kelimelerinin baş harflerinden oluşur.

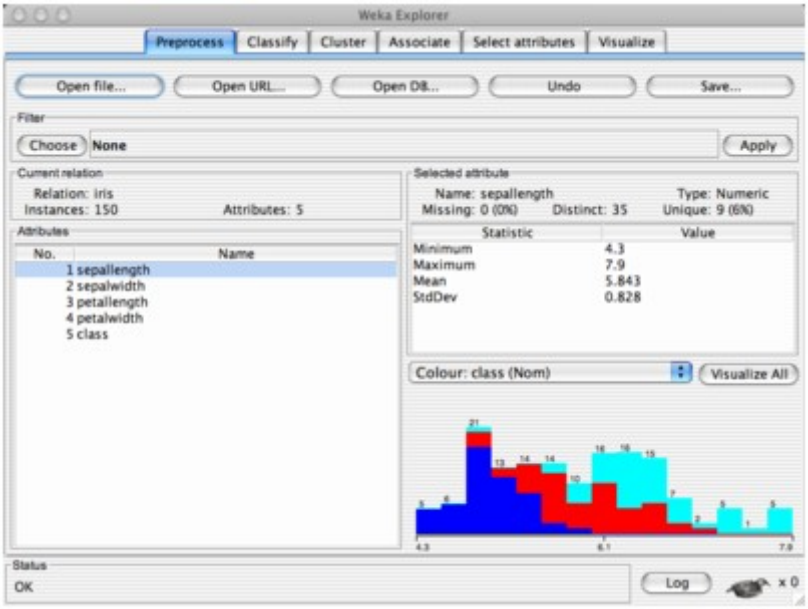
WEKA verileri basit bir dosyadan okur ve veriler üzerindeki stokastik değişkenlerin sayısal veya nominal değerler olduğunu kabul eder. Aynı zamanda veritabanı (database) üzerinden de veri çekebilir ancak bu durumda verilerin bir dosya verisi şeklinde olması beklenir.

WEKA üzerinde makine öğrenmesi ve istatistik ile ilgili pekçok kütüphane hazır olarak gelmektedir. Örneğin veri ön işlemesi (data preprocessing), ilkelleme (regression), sınıflandırma (classification), gruplandırma (clustering), özellik seçimi veya özellik çıkarımı (feature extraction) bunlardan bazılarıdır. Ayrıca bu işlemler sonucunda çıkan neticelerinde görsel olarak gösterilmesini sağlayan görüntüleme (visualization) araçları bulunmaktadır.

Aşağıda bu amaçla yazılan ekranlardan bazıları gösterilecektir

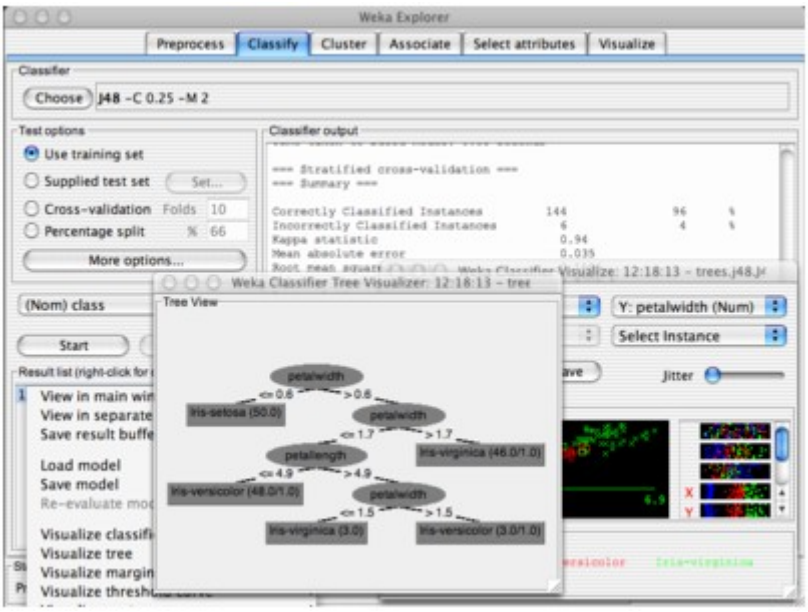
Ön işleme paneli (Preprocess Panel)

Bilgi dolaşıcısının (knowledge explorer) başlangıç noktası önişleme panelidir. Bu panelde veri kümleri (dataset) yüklenebilir veya WEKA içerisinde bulunan filitreler ile veriler üzerinde işlemler yapılarak veri işlenebilir.



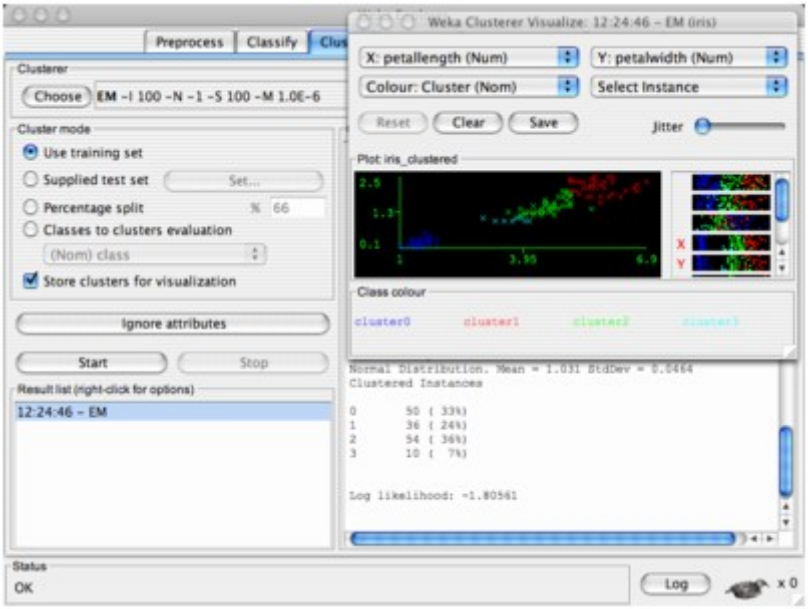
Sınıflandırıcı Paneli (Classifier Panel)

WEKA içerisinde yüklü olan sınıflandırma algoritmalarından herhangi birisini kullanarak mevcut veri kümesi üzerinde bu ekran marifetiyle sınıflandırma yapılabilir. Ayrıca bu ekranda test ve sağlama (validation) için ayrı kümeler kullanmak da mümkündür. Sınıflandırma hataları ayrı bir ekranda açılır ve şayet sınıflandırma algoritması bir karar ağacı (decision tree) oluşturursa bu da ayrıca bir ekranda görüntülenir.



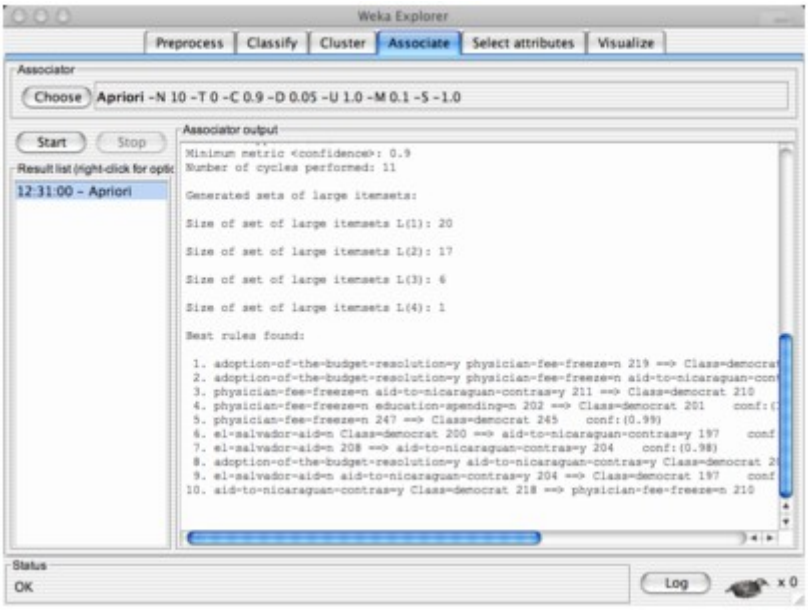
Klasör Paneli (Cluster Panel)

Sınıflamaya benzer şekilde klasösrleme (gruplama) için kullanılan ekrandır ve benzer şekilde görselleştirme arayüzü bulunmaktadır.



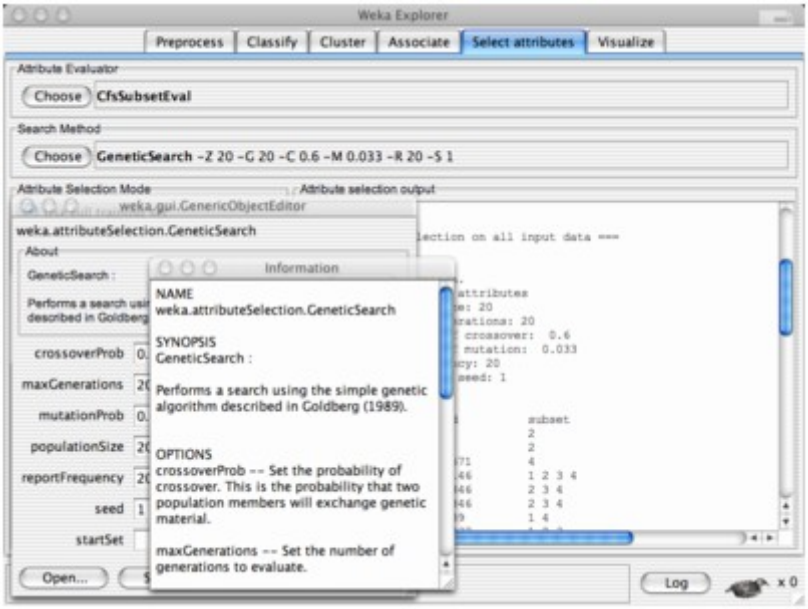
Birleştirme Paneli (Associate Panel)

weka içerisinde tanımlı birleştirme algoritmaları kullanılarak, mevcut veri kümesi üzerinde veri madenciliği (data mining) işlemi yapılmasını sağlar.



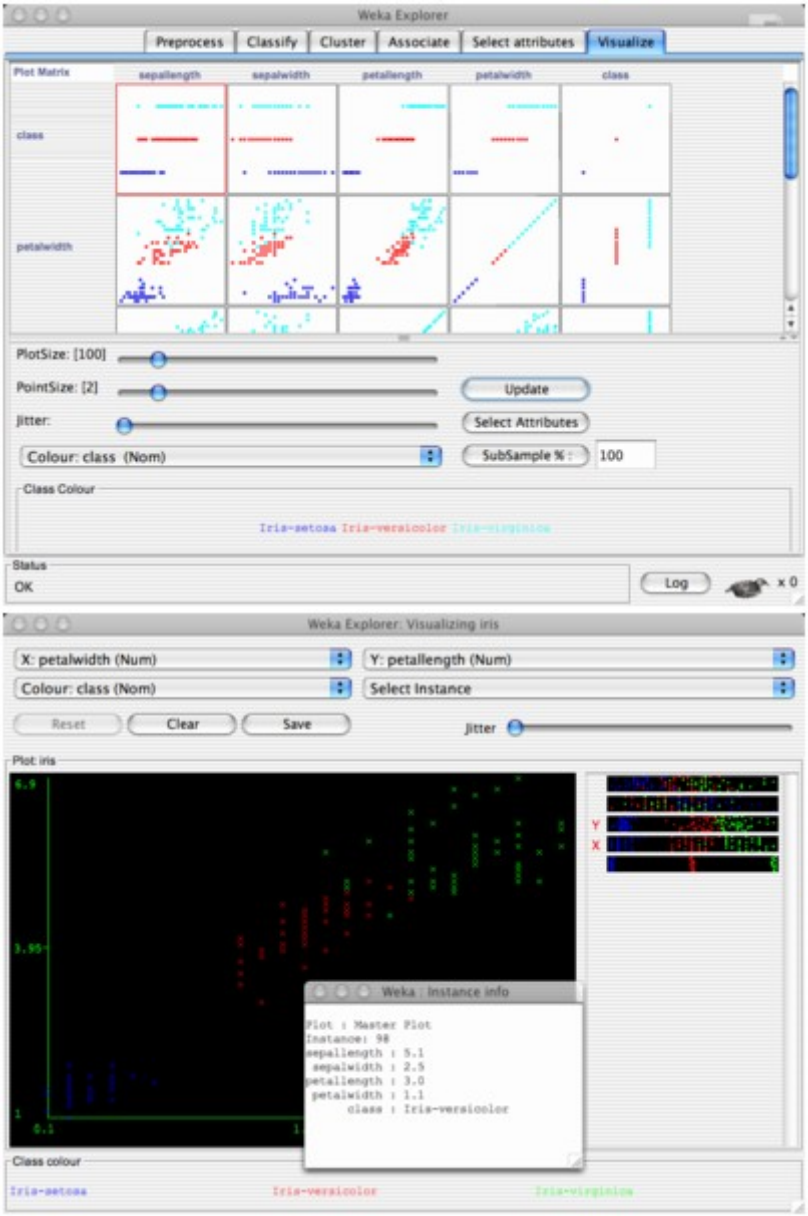
Seçim Özellikleri Paneli (Select Attributes Panel)

Veri kümesi üzerinde yapılan seçme ve işleme özelliklerini ayarlamaya yarar. Şayet seçme şemalarından birisi veriyi dönüştürüyorsa, dönüşmüş veri görselleştirme ekranında görülebilir.



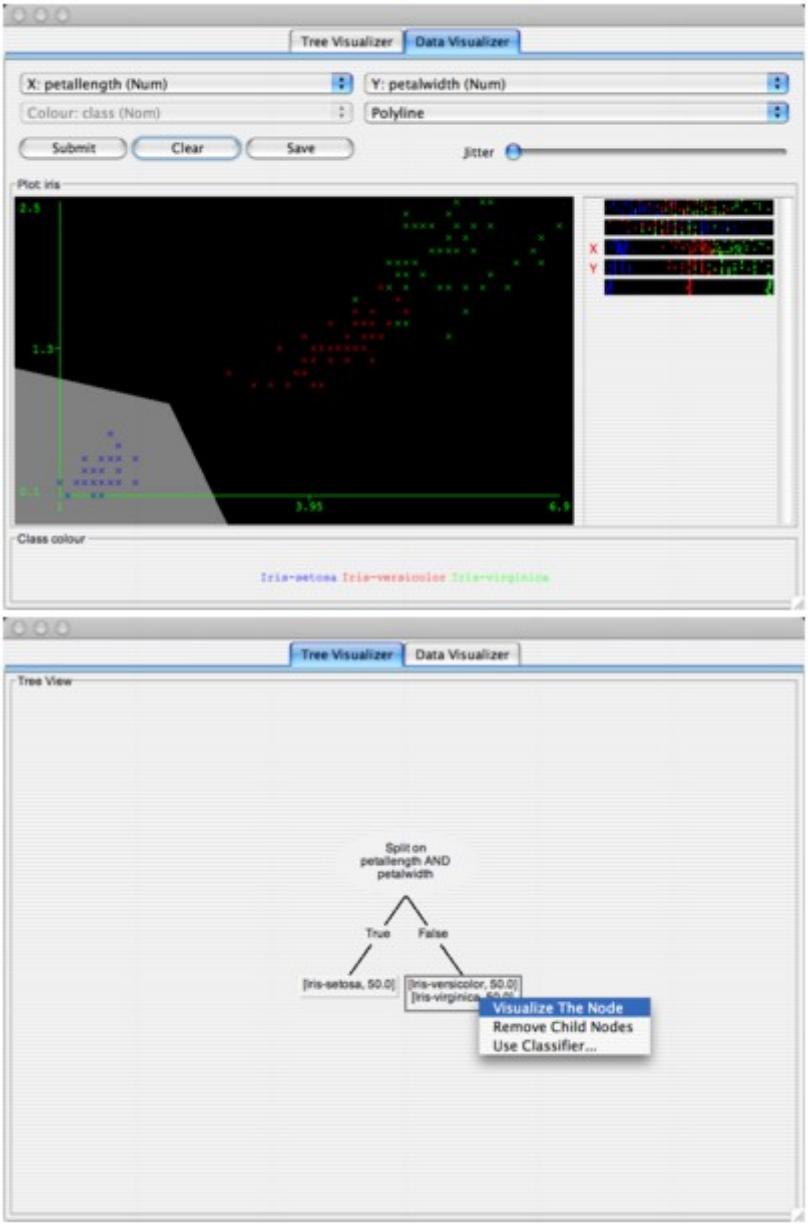
Visualize Panel

Bu panelde veri kümesi üzerinden bir çizim gösterilebilmektedir. Hücrelerin ve noktaların boyutları, ekranın alt tarafındaki panelden ayarlanabilir. Seçim özellikleri erkranından, matris üzerindeki hücre sayısı değiştirilebilir. Ayrıca çok büyük veri kümeleri ile çalışılırken, işlem kolaylığı olması açısından sadece alt örneklem uzayının kullanılması da mümkündür.



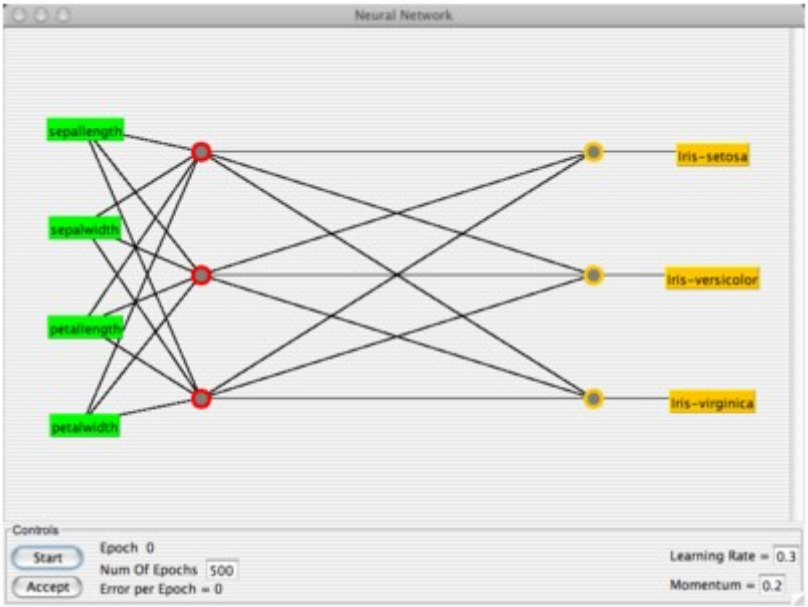
Etkileşimli Karar Ağacı İnşası (Interactive decision tree construction)

WEKA içerisindeki bu araç ile çift alternatifli (bi-variate) bölünmeleri ve bu bölünmeler üzerinde bir ağaç yapsını etkileşimli olarak inşa etmek mümkündür. Ayrıca inşa edilen bu ağacın yeniden değerlendirilmesi veya değiştirilmesi de mümkündür



Yapay Sinir Ağı (Neural Network GUI)

WEKA içerisinde bulunan yapay sinir ağı (neural network) arayüzüdür. Bu arayüz marifetiyle çok seviyeli perseptron (multi layer perceptron)ve eğitimi kontrol eden parametrelerin girilmesi mümkündür.


Neural Network
sepallength
sepalwidth
petallength
petalwidth
Iris-setosa
Iris-versicolor
Iris-virginica
Controls
Start
Accept
Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2