

1

Veri, Madencilik ve Öğrenme

Bu kitabın devamındaki tüm konuların daha iyi anlaşılabilmesi için öncelikle bazı temel kavramların net bir şekilde bilinmesi önemlidir. Bu sebeple öncelikle bu bölümde alanın temel kavramları ve uygulamaları hakkında bilgi verilmiştir.

1.1. Veri

Olaylar ve olgular ile ilgili işlenmemiş gerçeklere veri denir. Ham halde olması nedeniyle tek başına herhangi bir anlam ifade etmez. Ticari bir işletme açısından veri, gün içerisindeki tüm işlemlerinin kayıt altına alınmasıdır. Örnek olarak, bir lokanta için bir adet çay, iki adet kahve ve acılı ya da acısız adana birer veridir. Buradaki kayıtlar kendi başlarına bir anlam ifade etmeyip yalnızca olup bitenlerin bir kısmını açıklar. Örneğin müşterinin neden onları tükettiği veya aynı müşterinin genelde neleri sevdiği bu verilere bakılarak bilinemez. Dolayısıyla içerisinde yorum ve değerlendirme yoktur yani tek başına güvenilecek bir temel oluşturmaz [1].

Veriler ölçüm işlemi, sayım işlemi, deney yapma, gözlem yapma veya araştırma yolu ile elde edilir. Ölçüm veya sayım yolu ile elde edilen ve sayısal olarak ifade edilen verilere nicel veriler denir. Sayısal olarak ifade edilemeyen bilgilere ise nitel veriler denir. Her sembolik gösterim gibi, veri de belirli bir nesne, birey ya da olguya ilişkin bir soyutlamadır [2].

1.2. Enformasyon

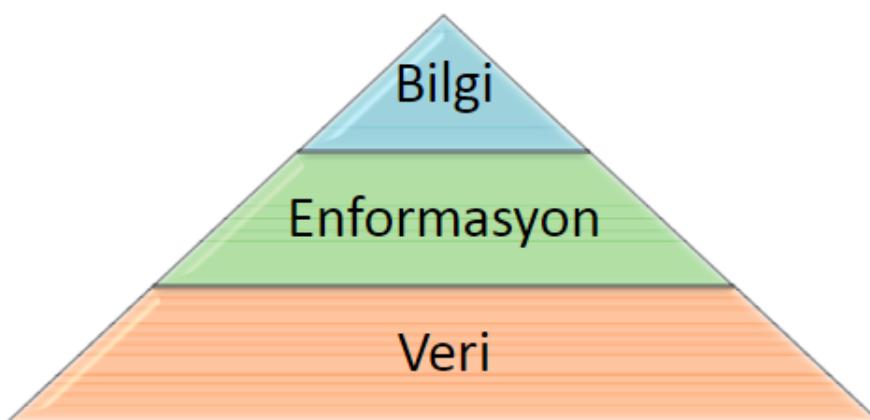
Verinin belirli işlemlerden geçirildikten sonra daha anlamlı biçimine enformasyon denir. Enformasyon veriden üretilip bir içerik ve amaca sahip

olur. Başka bir ifade ile enformasyona organize edilmiş veri denilebilir. Veriler üzerinde toplama, sayma, gruplama, özetleme, eleme, ayıklama, elle veya bilgisayar ile işleme sonucu enformasyona dönüştürülür. Bu şekilde veriler bir anlam kazanır ve böylece problem çözme veya karar verme süreçlerine temel teşkil eder [3].

Örneğin, bir lokantadaki toplam yiyecek satışlarının toplamı, satılan toplam çay adedi, bir ay boyunca kahveden elde edilen kar işletme için enformasyondur. Bir diğer adıyla malumat olarak da adlandırılır. Malumat belirli ve sınırları tanımlı bir konuya ilişkin belirli bir amaç doğrultusunda veri parçalarının işlenmesi sonucu elde edilir.

1.3. Bilgi

Kişinin çevresinde gerçekleşen olayları tam ve doğru bir şekilde kavramasını sağlayan kişiselleştirilmiş enformasyonlara bilgi denir [4]. Diğer bir deyişle enformasyonun anlam kazanmış haline bilgi denilebilir. Bilgi kişiye özel olup, alıcılar aracılığıyla kişinin beynine ulaşan enformasyonun ve burada önceki bilgiler ile işlenmesi sonucu bilgiye dönüşür [5].



Şekil 1.1. Veri, Enformasyon, Bilgi Piramidi

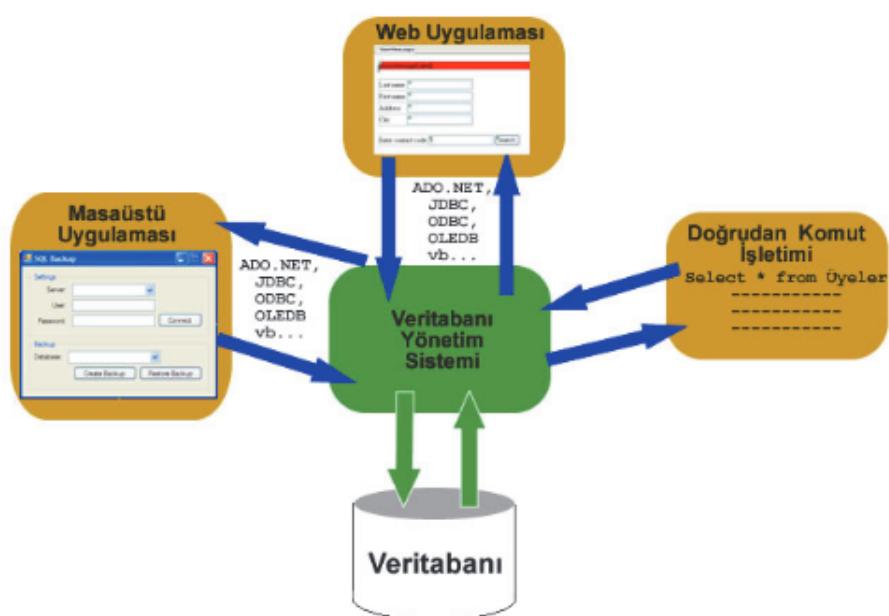
Yukarıdaki şekilde gösterildiği gibi enformasyon veriden elde edilirken, bilgi de enformasyondan elde edilir. Ayrıca çok büyük miktardaki veriden daha az enformasyon çıkarken, enformasyondan daha az miktarda bilgi üretilir.

Bilgi, kişisel olması nedeniyle inançlar, doğrular, bakış açıları, yargılar, beklentiler vb. değerlerden doğrudan etkilenir. Bilgi, enformasyonun incelemesi, değerlendirilmesi, sentezlenmesi ve en son olarak bir karar verme

sürecinden geçirilmesi sonucu elde edilir [3]. Örnek olarak bir lokantanın geçmiş haftalardaki Pazartesi günleri tüketilen yemek sayıları ve her bir yemekten elde edilen kâr dikkate alınarak bir sonraki Pazartesi günü için üretilen yemekleri belirlemek bilgidir. Burada bir önceki haftaların Pazartesi günü yemek sayıları ve elde edilen kârlar enformasyondur. Fakat bir sonraki Pazartesi için her bir lokanta aynı sayısal sonuçlara baksa da bir sonraki Pazartesi için yemek tür ve üretim sayıları farklı olacaktır. Elbette ki burada bir önceki haftalardaki veriler üzerinden elde edilen enformasyon bilgiye dönüsür.

1.4. Veri Tabanları

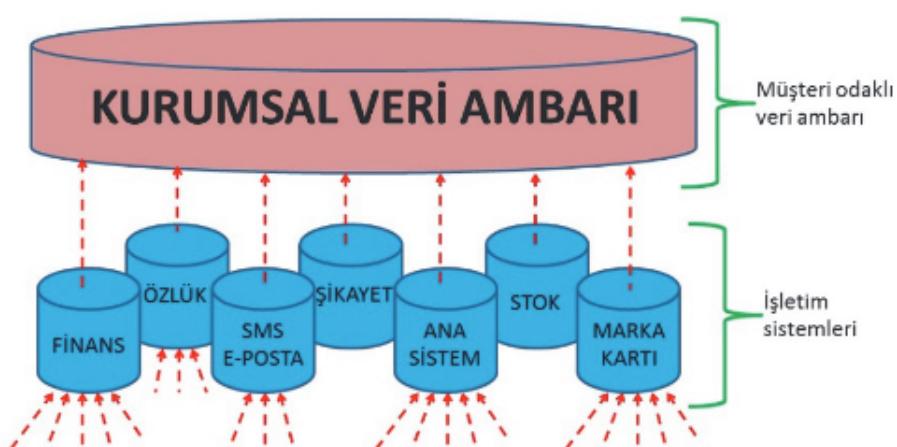
Gereksiz verileri kontrol ederek tekilleştiren ve verileri merkezileştirerek birçok uygulamayı etkin ve verimli bir şekilde sunmak amacıyla düzenlenmiş veri yiğinına veritabanı denir. Veriler her bir uygulama için bağımsız olarak ayrı ayrı saklanmak yerine tek bir konumda tutulur ve çoklu uygulamalara hizmet sunabilir [6]. Örneğin bir fabrika üretilen ürünleri, ürün satışlarını, fabrika giderlerini ve personel bilgilerini ayrı ayrı dosyalarda saklamak yerine ortak bir dosyada tutabilir. Bu tür kurumların verilerini merkezi yapıda tutmasını sağlayan, etkili ve verimli bir şekilde kullanım ve erişim sağlayan programlara ise veritabanı yönetim sistemleri denir. Aşağıdaki şekilde veritabanı ve veritabanı yönetim sistemi görsel olarak anlatılmıştır [7].



Şekil 1.2. Veritabanı ve Veritabanı Yönetim Sistemi

1.5. Veri Ambarları

Farklı veritabanlarında tutulan veriler arasında ilişki kurularak bu ilişkili verilerin tutulmasına veri ambarları denir. Veri ambarı, verilerin bir veya birden fazla yerdeki verilerin bir yere transfer edilip depolanmasından ziyade bu veriler üzerinde sorgulama ve analiz yapılmasına olanak tanıyan ilişkisel veri tabanıdır. Birden farklı yerdeki verileri kendi bünyesinde saklayabileceği gibi içerisinde farklı veri kaynakları da barındırabilir. Veri ambarları büyük miktardaki verilerin birden fazla yerden taşınmasından doğacak sorunları çözebileceği gibi farklı kaynaklardaki verileri kullanılabilir hale dönüştürür [8]. Aşağıdaki şekilde bir kurumun farklı sistemlerde tutulan verilerinden müşteri odaklı veri ambarı oluşturulması görsel olarak gösterilmiştir [9].



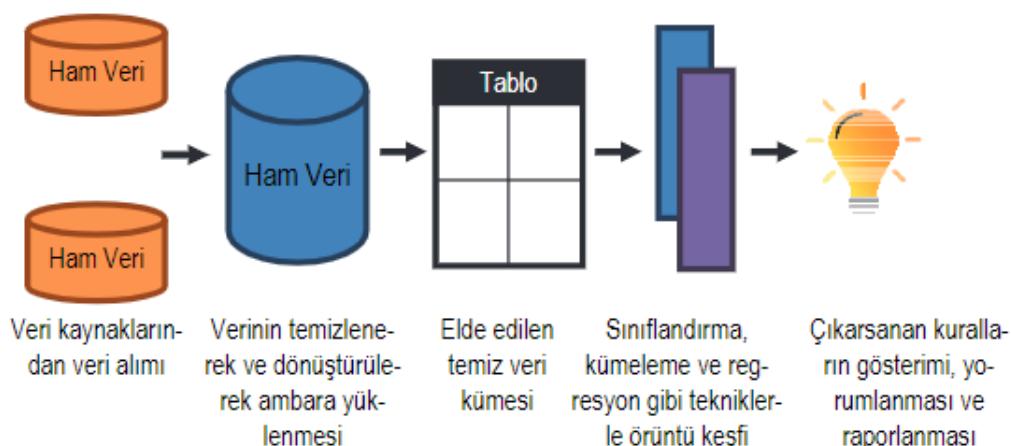
Şekil 1.3. Örnek Veri Ambarı

1.6. Veri Madenciliği

Veri yığınları arasında gizli kalmış, bir kısım analizler sonucu geçerli ve işe yarar bilgi elde edilmesine veri madenciliği denir. Veri madenciliği büyük miktardaki veriler üzerinde işlem yapması nedeniyle veritabanları ile doğrudan ilişkilidir. Belirli bir amaca yönelik oluşturulmasının yanı sıra verilere hızlı ve etkin bir şekilde erişim olanağı tanıyan veri ambarlarından yararlanılır. Veri madenciliği kendi başına çözüm değildir fakat karar verme süreçlerini destekler ve problemin çözümü için gerekli bilgileri sağlar [10].

Veri madenciliği veri toplama, temizleme, model oluşturma, model testi ve uygulama gibi birçok aşamaları içeren bir süreçtir. Bu süreçlerde her ne

kadar bilişim sistemleri kullanılsa dahi insanın yorum ve katkısı büyük bir öneme sahiptir. Bu sebeple bu sürecin tamamen bilgisayar tarafından otomatize edilmesi mümkün değildir [11]. Aşağıdaki şekilde veri madenciliği süreci görsel olarak gösterilmiştir [12].



Şekil 1.4. Veri Madenciliği Süreci

Veri madenciliği uygulamaları sağlık, pazarlama, bankacılık ve sigortacılık ile birçok alanda kendisine yer bulmuştur. Aşağıda bir kısım uygulama alanları kısaca verilmiştir [10].

- Veri tabanı analizi ve karar verme desteği
- Hedef pazar, müşteriler arası benzerliklerin saptanması, sepet analizi, çapraz pazar incelemesi gibi pazar araştırmalarında
- Kalite kontrol, rekabet analizi, öngörü, sahtekârlıkların saptanması gibi risk analizlerinde
- Haber kümeleri ve e-postalar üzerindeki benzerlikler
- Müşteri kredi risk araştırmaları
- Kurum kaynaklarının en uygun biçimde kullanımı
- Geçmiş ve mevcut yapı analiz edilerek geleceğe yönelik tahminlerde bulunma
- Müşterilerin satın alma örüntülerinin belirlenmesi
- Müşterilerin demografik özellikleri arasındaki bağlantıların bulunması

- Posta kampanyalarında cevap verme oranının artırılması
- Mevcut müşterilerin elde tutulması, yeni müşterilerin kazanılması
- Müşteri ilişkileri yönetimi
- Müşteri değerlendirme
- Satış tahmini
- Farklı finansal göstergeler arasında gizli korelasyonların bulunması
- Kredi kartı dolandırıcılıklarının tespiti
- Kredi kartı harcamalarına göre müşteri gruplarının belirlenmesi
- Kredi taleplerinin değerlendirilmesi
- Yeni poliçe talep edecek müşterilerin tahmin edilmesi,
- Sigorta dolandırıcılıklarının tespiti,
- Riskli müşteri örüntülerinin belirlenmesi

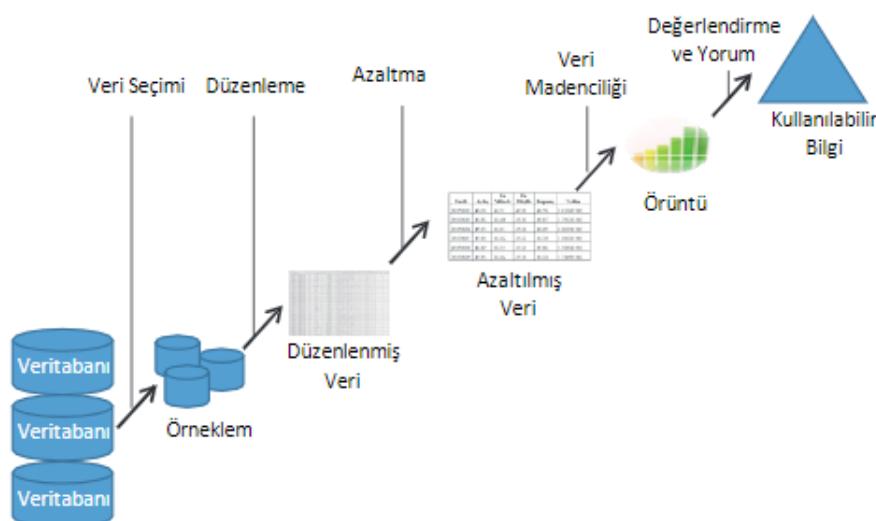
1.7. Veri Madenciliğinde Verileri Hazırlama

Veri madenciliği uygulamalarının yaygınlaşmasının temel nedenleri arasında büyük veritabanlarının erişilebilir olması yer alır. Birçok kuruluş personel ve araç giriş çıkışlarından, hatta açılıp kapatılan kapılardan, tüketilen her tür yiyecek-içecek fiyat ve miktarlardan daha sayılamayacak kadar tür ve sayıda veriyi kendi sunucularında saklamaktadır. Tabi ki burada tutulan bu kadar çok sayıdaki verilerin hepsinin tamamen doğru veriler olduğu garanti edilemez. Ayrıca bu verilerin tamamının doğru olduğu varsayılsa dahi bu verilerin tümünün istenilen amaç için kullanılacağı düşünülemez. Yani bu kadar veri yiğini arasında bir kısım veriler eksik, bir kısım veriler yanlış veya anlamsız, bazı veriler ise aynı anlamı taşıyarak farklı değerlere sahip şekilde gereksiz tekerrür ederek girilmiş olabilir. Dolayısıyla bu veri yiğini arasından temizleme veya düzenleme yapılması zorunludur. Veriler üzerinde yapılacak işlemler aşağıdaki gibi özetlenebilir [13].

- Kayıp verilerin düzenlenmesi
- Gürültünün ortadan kaldırılması
- Bütünleştirme

- Dönüşürme
- Azaltma

Veri madenciliği disiplinler arası bir çalışma olması nedeniyle istatistik, matematik, derin öğrenme, makine öğrenmesi, yapay zekâ gibi disiplinlerin yöntemlerini kullanmaktadır. Veri madenciliğinde belirli adımların yerine getirilmesi sonrası veri yiğinları arasından bilgi keşfi gerçekleşir. Aşağıdaki şekilde bu durum görsel olarak gösterilmiştir.



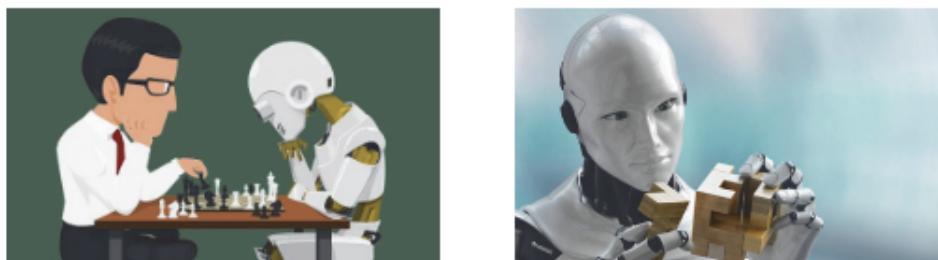
Şekil 1.5. Veri Madenciliğinde Bilgi Keşfi

1.8. Yapay Zekâ

İnsan beyni hem dünyanın en karmaşık makinesi hem de en mükemmel makinesi olarak kabul edilebilir. İnsan beyni üzerine çalışmalar yıllardır devam etmesine rağmen halen daha çözümlenmiş değil. Bilgisayarlar ile karşılaşıldığında belki sayısal bir işlemi bilgisayardan daha yavaş yapabilir fakat idrak etmeye yönelik problemleri inanılmaz kısa sürede çözer. Örneğin bir çocuğun camdan aşağı sarktığını gören bir baba saniyeler içerisinde yaşanmış veya yaşanmamış olduğu fark etmeksizin geçmiş tecrübelerinden yola çıkarak bebeğin düşebileceğini düşünür ve ona engel olur. Burada kendisinin bir başka çocuğunun daha önce düşmesinden oluşan bir tecrübeye sahip olabileceği gibi başkalarından duymuş olduğu bilgiler ile de tecrübeye sahip olur ya da sezgilerini kullanır. Buradaki durumun benzerini sergileyebilecek bilgisayarlar zekâsı üretmek için uzun yıllardır çalışmalar sürdürmektedir. Fakat bilgisayarlar çok karmaşık sayısal problemleri çok kısa sürelerde

çözmeye rağmen, deneyimlerden elde edilen bilgileri kullanma ve idrak gerektiren konularda yetersiz kalmaktadır [14].

Çevreyi algılama, karar verme ve hareketleri kontrol etme yeteneğine zekâ denir. Zekâ iyileştirilebilir, geliştirilebilir ve değiştirilebilir. İnsanlar çok farklı zekâ gruplarına sahip olup, her insanda kendine özgü ve her insanda farklı gelişim süreçlerin sahiptir. İnsan zekâsı ile gerçekleştirilen düşünme, anlama, kavrama, yorumlama ve öğrenme işlemlerini bilgisayar programları aracılığıyla problem çözümüne uygulanmasına ise yapay zekâ denir. Elbette bir insanın sahip olduğu his, sezgi ve ruhsal durum gibi özelilikler ile düşünme eyleminin bilgisayara kazandırılması mümkün olmayaçaktır. Bilgisayarlar, işlemci hızı, hafızası ve çalışma hızı ile değerlendirilirken yapay zekâ ise hesaplama gücü, kullanılan yöntem ve hafıza olarak değerlendirilmektedir. İnsan beynindeki nöronların kimyasal olarak birbirine bağlanma potansiyeli hesaplama gücü iken, yapay zekâ donanımı işlemci hızı ile hesaplama gücüne karşılık gelmektedir. Problem çözümünde insanlar çözümleme metodolojisi ve prensiplerini yöntem olarak kullanırken, yapay zekâ matematiksel yöntemini kullanılır. İnsanların belleği bilgiyi tutan hafıza kapasitesini ifade ederken, yapay zekâda bilgisayar hafızasına karşılık gelir [15]. Aşağıdaki şekilde düşünen robotlar ile yapay zekâ görselleştirilmiştir [16, 17].

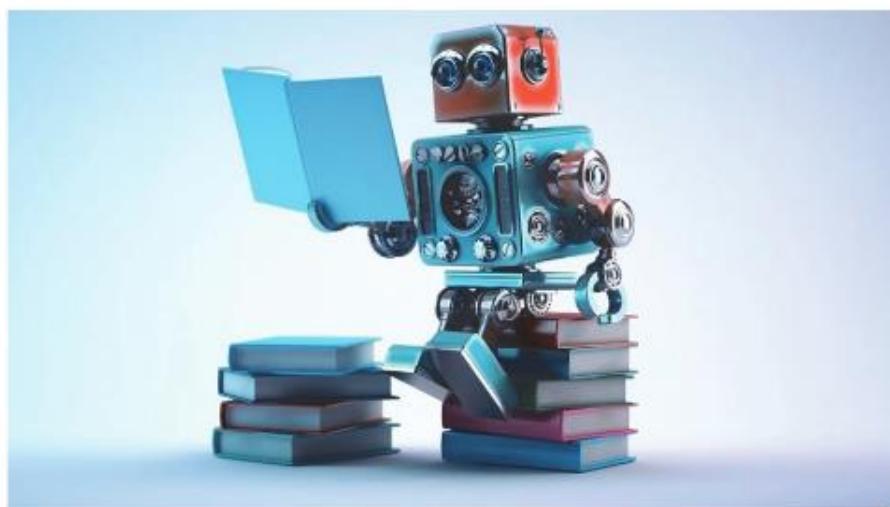


Şekil 1.6. Yapay Zeka ve Düşünme

1.9. Makine Öğrenmesi

Makine öğrenmesi yapay zekânın bir alt dalıdır. Veriler arasındaki karmaşık örüntülerin çıkarılması için bilgisayar tabanlı matematiksel modellerden yararlanır. Her an milyonlarca verinin saklandığı yiğinlar çeşitli analizler yapılması için tutulmaktadır. Bu veri yiğinları üzerinde yapılacak her bir analizde ise farklı sonuçlar çıkacağının düşünülmektedir. Örneğin kesintisiz görüntü kaydı yapan güvenlik kameraları olağan dışı bir durumu olağan durum olan önceki veriler ile karşılaştırarak öğrenmeye çalışır. Kredi kartı

hareketlerinin tutulması sonucu önceki hareketlere bakarak kişinin daha fazla borçlanmayı ödeyip ödeyemeyeğini öğrenmeye çalışmaktadır. Anlatılan bu tür sistemlerin tümünde bu kadar çok geçmiş veriyi saklamadan temel amaçları arasında bunların çeşitli analizler ile değerlendirilip yorumlanması ile geleceğe yönelik öngörüler elde etmektir. İşte bu tür analizler için makine öğrenmesi yöntemleri geliştirilmiştir. Makine öğrenmesi yöntemleri geçmiş verilere bakarak, onlar arasındaki örüntüyü bulacak en iyi modeli tespit eder ve sonraki verileri bu modeli uygulayarak tahmin eder. Dolayısıyla önceki verilere bakarak örüntüyü nasıl bulacağını öğrenmiş olur. Sonraki verilere bu öğrendiğini uygular ve bu tahmin/öngörü olarak karşımıza çıkar. Sonuç olarak; karar verme süreçlerine destek olmak için çok miktardaki geçmiş verinin analiz edilerek gelecek ile ilgili öngörüler elde edilmesini sağlayan yöntemlere makine öğrenmesi denir. Aşağıdaki şekilde bu durum bir görsel ile gösterilmiştir [18].

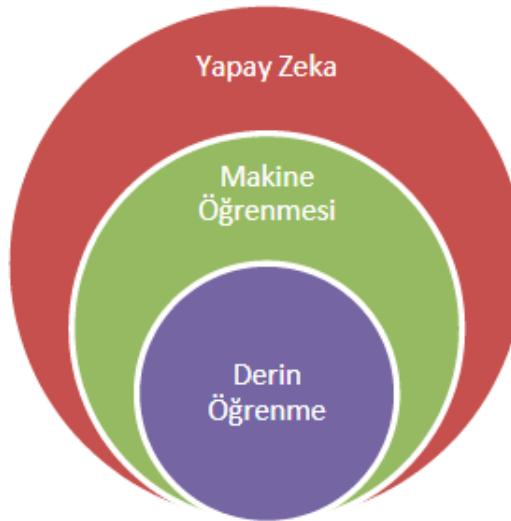


Şekil 1.7. Makine Öğrenmesi

1.10. Derin Öğrenme

Yapay sinir ağlarının ne olduğu bu kitabın bir sonraki bölümünde detaylı olarak ele alınacaktır. Derin öğrenme kavramı yeni bir kavram olmamakla birlikte yapay sinir ağlarının geliştirilmesi ile ortaya çıkmaya başladı. Önceleri yapay sinir ağları uygulamaları bir veya iki katmandan oluşur ve bu nedenle de bazı problemlerin çözümünde basit kalındı. İşte buradaki iç katman sayılarının ve iç katmandaki her bir sinir sayısının en ideal modeli tespit etmek için bulunması, yapay sinir ağlarında kullanılan parametrelerin en iyi değerini kendisi keşfeden sistemlere derin öğrenme denir. Aşağıdaki şekilde

de gösterildiği üzere yapay zekâ kavramı en genel bir kavram olmakla birlikte makine öğrenmesi ise derin öğrenme kavramını kapsar [19].



Şekil 1.8. Yapay zeka, makine öğrenimi ve derin öğrenme

1.11. Yapay Zekâ Yöntemleri

Yapay zekâ uygulamalarında kullanılan yapay zekâ yöntemleri aşağıdaki gibi gruplandırılabilir [20].

- Sınıflandırma: Geçmiş verilerin hangi sınıf içerisinde yer aldığı belirtilmediği durumda yeni verinin hangi sınıf içerisinde yer alacağını bulma işlemidir.
- Kümeleme: Geçmiş verilerin hangi sınıf içerisinde yer aldığı belirtilmediği veya bilinmediği durumda verilerin benzerliklerine kümelere ayrıştırılması işlemidir.
- Regresyon (Eğri Uydurma): Geçmiş verilerin sürekli gösteren sayısal değerlerden oluştugu durumlarda, bu değerlerden bir eğri modeli üretme işlemidir.
- Özellik Belirleme: Geçmiş verilerin çok fazla olması durumunda bu verilerin sınıfını belirleyen özellikler belirlenir. Bu belirleme işlemi sırasında mevcut özelliklerden bir alt kümeye oluşturulabileceği gibi, bunların birleşiminden yeni özellikler de oluşturulabilir.
- İlişki Çıkarımı: Bir veri ile bir başka verinin birlikte yer alma durumunun analiz edilerek en çok birlikte olan verilerin belirlenmesidir. Özellikle sepet analizlerinde kullanılır.

Buradaki yöntemler öğrenme biçimine göre gruplandırılırsa eğer, veriler ile birlikte bu verilerin hangi sınıflara ait olduğunu bilinmesi olan sınıflandırma ve regresyon yöntemlerinin tümü Güdümlü/Gözetimli/ Öğretmenli Öğrenme (*Supervised Learning*) olarak kabul edilir. Fakat kümeleme yöntemlerin tümü ise verilerin ait olduğu sınıfların bilinmemesi nedeniyle Güdümsüz/ Gözetimsiz/Öğretmensiz Öğrenme (*Unsupervised Learning*) olarak kabul edilir. Buradaki her iki yöntemin karıştırılarak kullanılmasına ise Yarı Güdümsüz Öğrenme (*Semi-Supervised Learning*) denir. Aşağıdaki şekil bu durumu görsel olarak anlatmıştır [18].



Şekil 1.9. Yapay Zeka Yöntemleri

1.12. Yapay Zekâ Uygulama Alanları ve Geleceği

Yapay zekâ denilince ilk akla gelen zekâsı olan robotların ileride dünyayı istila edebileceğii fikridir. Fakat yapay zekâ yalnızca robotlar ile sınırlı değildir. Günümüzde yapay zekânın kullanılma potansiyeli olan binlerce yapay zekâ uygulama alanı vardır. Matematik, fizik, biyoloji, tıp, kimya, elektrik, elektronik gibi tüm bilim dalları yakından ilgilenmektedir. Bankacılık, sigortacılık, sağlık, market alışverişleri, askeri, güvenlik, ses tanıma, doğal dil işleme, görüntü işleme gibi daha birçok alanlarda şunda kullanılmaktadır ve bu kullanım her geçen gün yaygınlaşmaktadır.

Yapay zekâ bilhassa otonom kontrol özelliği ile askeri alanda yaygın bir şekilde kullanılırken, bankacılık ve bilgisayar sektörlerinde de çok fazla tercih edilmektedir. Hatta gelecekte yapay zekâ sahibi robotların günlük hayatın vazgeçilmezi olarak insanoğlunun yaşam standartlarını çok ileri seviyelere yükselteceği düşünülüyor. Eşyaların interneti ile evlerin akıllı hale gelmesi bunun en bariz örneğidir. Özellikle askeri alanda belki de bir zaman sonra savaşlarda insanlar yerine tamamen robotlar yer alacak gibi görünüyor. İnsansız hava araçları, hedef tespit sistemleri ve sürücüsüz araç teknolojileri bunların en güzel örneğidir.

1.13. Yapay Zekâ Uygulamalarında Kullanılan Programlar

Yapay zekâ uygulamalarında birçok programdan faydalankmaktadır. Bunlardan bir kısmı aşağıda listelenmiştir. Fakat her birinin detayları ile ilgili bilgi verilmemiştir. Bu konuda daha detaylı bilgi için Doğan [21]'nın çalışmasının incelenmesi yararlı olacaktır.

- | | |
|--------------------------|------------------------|
| 1. ADaM | 20. Mining Mart |
| 2. AdamSoft | 21. ML-Flex |
| 3. Alpha Miner | 22. MDP |
| 4. Apache mahout | 23. NLTK |
| 5. CMSR Data Miner | 24. OpenNN |
| 6. Databionic ESOM Tools | 25. Orange |
| 7. Data Melt | 26. Pandas |
| 8. Dlib | 27. Pybrain |
| 9. ELKI | 28. R |
| 10. Fityk | 29. Rapid Miner (Yale) |
| 11. GGobi | 30. Rattle GUI |
| 12. GNU Octave | 31. Rosetta |
| 13. Jubatus | 32. SIPINA |
| 14. KNIME | 33. Shogun |
| 15. Keel | 34. Scikit Learn |
| 16. LIBSVM | 35. SenticNet API |
| 17. LIBLINEAR | 36. TANAGRA |
| 18. Lattice Miner | 37. Vowpal Wabbit |
| 19. Mallet | 38. Weka |

2

Yapay Sinir Ağları

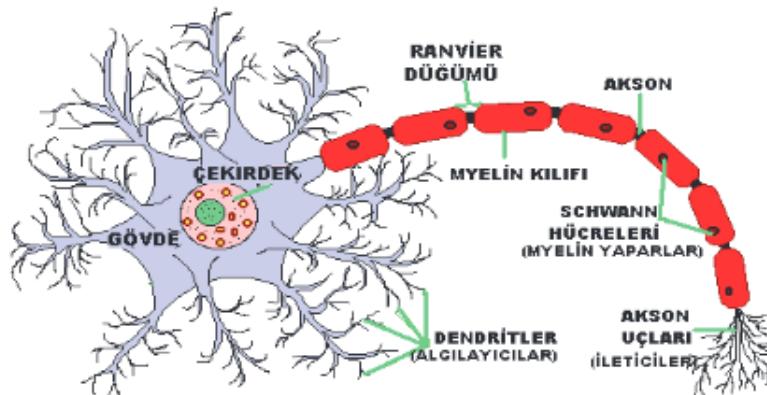
İnsanın en önemli özellikleri arasında yer alan düşünme ve öğrenebilme yetenekleri üzerine yapılan araştırmalarda yapay zekâ kavramı ön plana çıkmıştır. İnsanın düşünme yapısını anlayarak benzer sonuçlar üreticek bilgisayar işlemleri geliştirmeye çalışmaya yapay zekâ denir. Bir başka deyişle programlanmış bilgisayarlara düşünme yeteneği sağlama girişimidir. Yapay zekânın amacı insan zekâsı düzeyinde bilgisayarları geliştirmek ve insanların zeki davranışlarına benzer makineler yapmaktadır. Yapay sinir ağları (YSA) yapay zekâ çalışmalarına destek sağlayan bir başka alandır. Dolayısıyla YSA için yapay zekânın bir alt dalı olduğu ve öğrenebilen sistemlerin temelini oluşturduğu söylenebilir. YSA'lar insan beyninin temel işlem elemanı olan nöronu şekilsel ve işlevsel olarak basit bir şekilde taklit etmeye çalışırlar. Bu yolla biyolojik sinir sisteminin basit bir simülasyonunu gerçekleştiren programlar olduğu söylenebilir [22, 23].

İnsan beyninin çalışma ilkesinden esinlenerek geliştirilmiş, her biri belirli ağırlıklara sahip bağlantılar aracılığıyla birbirine bağlanan ve yine her biri kendi belleğine sahip işlem elemanlarından oluşan paralel ve dağıtılmış bilgi işleme yapılarına yapay sinir ağları denir [24]. Bir başka deyişle, biyolojik sinir ağlarını taklit eden bilgisayar programları olduğu da söylenebilir [25]. Bu programlara olan ilgi günden güne artmakte ve birçok problemin modellemesinde, kontrol edilmesinde veya çözümlenmesinde en iyi çözümler üretmeye devam etmektedir [26].

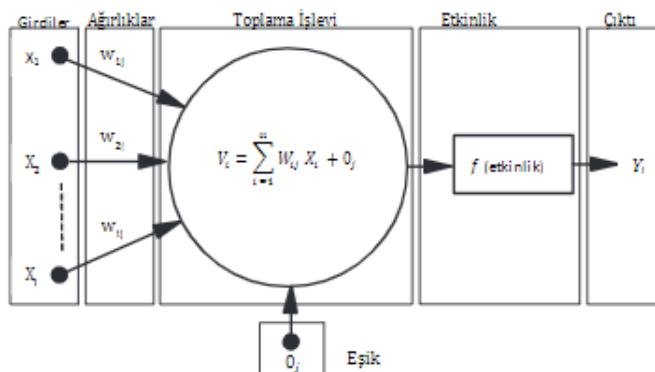
2.1. Yapay Bir Sinirin Öğeleri

Yapay bir sinir hücresi, biyolojik sinirlere göre daha basit olmasına birlikte biyolojik sinirlerin 4 temel işlevini gerçekleştirmeye çalışırlar. Aşağı-

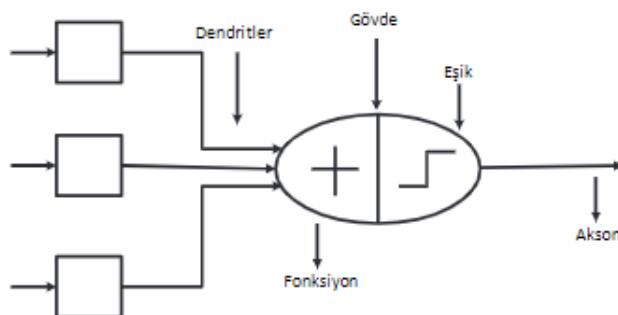
daki Şekil 2.1, Şekil 2.2 ve Şekil 2.3'te biyolojik bir sinir hücresi ve yapay bir sinir ağları şekilsel olarak gösterilmiştir.



Şekil 2.1. Biyolojik bir beyin sinir hücresi [27]



Şekil 2.2. Yapay bir bir sinir elemanı



Şekil 2.3. Biyolojik Sinir Hücresi ile Yapay Bir Sinir Hücresinin Benzetimi

Tüm yapay sinir ağları bu temel yapıdan üretilmiş olup öğrenme yeteneği, seçilmiş olan öğrenme algoritması içerisindeki ağırlıkların en ideal

şekilde ayarlanmasına bağlıdır. Elmas [28] yapay bir sinirin temel öğelerini aşağıdaki gibi açıklamıştır;

- **Girdiler:** Kendinden önceki sinirlerden veya dış dünyadan aldığı bilgiyi sinire getirir. Çoğunlukla bir sinir hücresi birden fazla gelişigüzel girdi alır.
- **Ağırlıklar:** Her bir girdinin sinir üzerindeki etkisini belirleyen ve her birinin kendisine özel uygun katsayılardır. Ağırlık katsayısının değerinin büyüklüğü önemi ile doğru orantılıdır.
- **Toplama İşlevi:** Sinirde her bir ağırlığın ait olduğu girdiler ile çarpılması ve eşik değer ile toplanması sonucu elde edilir ve etkinlik işlevine gönderilir.
- **Etkinlik İşlevi:** Toplama işlevinden elde edilen sonuç etkinlik işlevinden geçirilip çıktı katmanına gönderilir. Burada toplama işlevinin çıktıının değişmesine izin verilir.
- **Ölçekleme ve Sınırlama:** Düğümlerde, etkinlik işlevinin sonuçları ölçekleme ya da sınırlama işlemlerine tabi tutulabilir. Ölçeklendirme bir ölçek etmeni ile etkinlik değerinin çarpılmasından elde edilir. Sınırlandırmada ise, ölçeklenerek elde edilmiş sonuçların en az ve en çok sınırlarını aşmaması sağlanır.
- **Çıktı İşlevi:** Etkinlik işlevinden elde edilen sonucun dış dünyaya ya da bir başka sinire girdi olarak gönderildiği yerdır. Bir sinir yalnızca bir çıktıya sahiptir.

2.2. Yapay Sinir Ağlarının Güçlü ve Zayıf Yönleri

Yapay sinir ağlarının öğrenebilme kabiliyetlerinin olması ve farklı öğrenme algoritmaları ile çalışabilmesi güçlü yanlarından iken sistemin çalışma mantığının analiz edilememesi ve öğrenme sonucunda başarılı olamama riski zayıf yanları arasındadır. Elmas [14] tarafından yapay sinir ağlarının güçlü ve zayıf yönleri aşağıdaki Tablo 2.1'de özetlenmiştir.

Tablo 2.1. Yapay sinir ağlarının güçlü ve zayıf yönleri

Güçlü Yönleri	Zayıf Yönleri
Matematiksel modele ihtiyaç duymamaları	Sistem içerisinde ne olduğunu bilinememesi
Kural tabanı kullanımı gerektirmemeleri	Bazı ağlar dışında kararlılık analizinin yapılamaması
Öğrenme kabiliyetlerinin olmaları ve farklı öğrenme algoritmaları ile öğrenebilmeleri	Farklı sistemlere uyarlanmasıın zorluğu

2.3. Yapay Sinir Ağlarının Genel Özellikleri

Yapay sinir ağlarının genel özellikleri veya dezavantajları kullanılan modele göre değişiklik göstermektedir. Burada Öztemel [29] tarafından YSA'nın genel özellikleri ve dezavantajları aşağıdaki gibi belirtilmiştir.

- Yapay sinir ağları olayları öğrenerek benzer olaylar karşısında benzer kararlar vermeye çalışarak bilgisayarın öğrenmesini sağlamaya çalışırlar.
- Geleneksel programlama yöntemlerinden farklı çalışma stillerine sahiptirler.
- YSA'da bilgi, ajan bağlantılarının değerleri ile ölçülmekte ve bağlantı larda saklanmaktadır. Geleneksel programlarda olduğu gibi veriler veri tabanında veya programın içinde değildir.
- Öğrenme örnekler üzerinden gerçekleştirilir. Bütün yönleri ile gösterilen örnek olaylar hakkında, genelleme yapabilecek yeteneğe kavuşturulurlar.
- YSA'ların güvenle çalıştırılabilmesi için öncelikle eğitilmeleri ve sonrasında performanslarının test edilmesi gerekmektedir. Örnekler iki gruba ayrılarak; bir grup ağı eğitmek için diğer grup ise performansı test etmek için kullanılır.
- Daha önce görülmeyen olaylar hakkında, önceki örneklerden yaptığı genellemelerle bilgi türetebilirler.
- Bilgiye dayalı sistemlerde uzman sistemler kullanılırken algılamaya yönelik olaylarda daha çok YSA'lar kullanılır.
- Ağların çoğunu amacı kendisine örnekler halinde verilen örüntülerin kendisi veya diğerleri ile ilişkilendirilebilmesi olmakla birlikte verilen örneklerin sınıflandırılması da yapılabilmektedir.
- Bazı durumlarda ağa eksik bilgileri içeren örüntü ve şekil verildiğinde ağ bu eksik bilgileri tamamlayabilir.
- Yapay sinir ağlarının örnekler ile kendisine gösterilen yeni durumlara adapte olması ve sürekli yeni olayları öğrenebilmesi mümkündür.
- Eğitildikten sonra eksik bilgiler ile de çalışabilir ve gelen yeni örneklerde eksik bilgi olmasına rağmen sonuç üretебilirler. Ajan performans düşerse eksik bilginin önemli olduğuna karar verilir. Eğer performans düşmez ise eksik bilginin önemsiz olduğuna karar verilir.

- YSA'ların eksik bilgiler ile çalışabilmesi hatalara karşı toleranslı olmalarını sağlamaktadır. Böylelikle ağır bir kısmı hücrelerinin bozulması veya çalışmamaz durumda olması halinde bile ağ çalışmaya devam edebilir. Fakat bozuk hücrelerin önemine göre ağır performansı da etkilenir.
- Belirsiz, tam olmayan bilgileri işleyebilme yetenekleri vardır.
- Hatalara karşı toleranslı olmaları aşamalı olarak bozulmalarına neden olur. Bir ağ zaman içerisinde yavaş yavaş ve zarif bir şekilde bozulur. Böylelikle herhangi bir problem karşısında hemen bozulmazlar.
- Bilgi ağa yayılmış durumdadır. Bu sebeple de dağıtık belleğe sahiptirler. Hücrelerin birbirleri ile bağlantılarının değerleri ağır bilgisini gösterir. Bu sebeple tek bir bağlantı herhangi bir anlam ifade etmez.
- Yalnızca sayısal veriler ile çalışabilirler. Sayısal olmayan ifadelerin sayısal gösterime çevrilmesi zorunludur.

2.4. Yapay Sinir Ağlarının Dezavantajları

Yapay sinir ağlarının dezavantajları da genel özellikleri gibi kullanılan modele göre değişiklik göstermektedir. Öztemel [29] tarafından YSA'nın dezavantajları aşağıdaki gibi belirtilmiştir.

- Donanım bağımlı çalışmaları önemli bir sorun olarak kabul edilmektedir. Paralel işlemciler üzerinde çalışabilmeleri temel varoluş nedenleri arasındadır. Özellikle gerçek zamanlı bilgi işlerken paralel çalışabilen işlemcilerin olması bir zorunluluktur. Günümüzdeki makineler aynı anda tek bir bilgiyi işleyerek seri şekilde çalışmaktadır.
- Probleme uygun ağ yapısının belirlenmesinin deneme yanılma yolu yapılması önemli bir problemdir. Eğer problem için uygun bir ağ oluşturulmaz ise çözümü olan bir problemin çözülememesi veya düşük performanslı çözümler elde edilmesi söz konusu olabilir. Bu sebeple bulunan çözümün en iyi çözüm olduğu garanti edilemez. Bir başka deyişle YSA'lar kabul edilebilir çözümler üretirken en iyi çözüm olduğunu garanti etmez.
- Bir ağın nasıl oluşturulması gerektiğini belirleyecek kuralların olmaması da bir başka dezavantajdır. Her problem farklı sayıda işlemci gerektirebilir. Bazı problemleri çözebilmek için gerekli olan paralel işlemcilerin tamamını bir anda çalışırmak mümkün olmayabilir.
- Problemin sayısal gösterimlere dönüştürülmesi kullanıcının tecrübesine bağlıdır. Uygun bir gösterim mekanizmasının kurulamamış olmasının

problemin çözümünü engelleyebilir veya düşük performansa sahip bir öğrenme oluşabilir. Problemin sayısal gösterimi mümkün olsa dahi bunun ağa gösteriliş şekli problemin başarılı bir şekilde çözülmesini yakından etkiler. Bir olayın hem ayrık hem de sürekli değerler ile gösterilebilmesi durumunda hangisinin daha başarılı öğrenme gerçekleştireceği bilinmemektedir. Hatta bu konuda kullanıcının tecrübeşi dahi yeterli değildir. Bu durum günümüzdeki birçok olayın YSA'lar ile çözülememesindeki en önemli nedenler arasındadır.

- Bir kısım ağlarda parametre değerleri belirlenirken bir kuralın olmaması problemdir. Bu durum iyi çözümler bulmayı zorlaştıran bir etkendir. Bu parametrelerin belirlenmesi kullanıcının tecrübesine bağlıdır. Parametre değerleri için belirli standartların oluşturulamamasından dolayı her problem için ayrı ayrı değerlendirme yapılması uygun olacaktır. Bu önemli bir dezavantajdır.
- Ağın eğitiminin ne zaman sonlandırılacağına yönelik geliştirilmiş bir yöntem bulunmamaktadır. Ağın örnekler üzerindeki hatasının belirli bir değerin altına indirilmesi eğitimin tamamlanması için yeterli kabul edilmemektedir. Bu sebeple sonuçta en iyi öğrenmenin gerçekleştiği söylememekte fakat iyi çözümler üretebilen ağ oluştugu kabul edilmektedir. En iyi sonuçları üreten bir mekanizma henüz daha geliştirilememiştir.
- YSA'ların öğrenme süresi uzundur. Ağın eğitiminde çok fazla deneme-ye ihtiyaç duyulmaktadır. Eğitim zamanının kısaltılması kritik öneme sahiptir.
- Ağın davranışları açıklanamamaktadır. Bir probleme çözüm üretildiği zaman bunun nasıl ve neden üretildiği konusunda bir bilgi bulmak mümkün değildir. Bu durum ise ağın ürettiği sonuçlara olan güvenin azalmasına sebep olmaktadır.

2.5. Yapay Sinir Ağlarının Temel Öğrenme Kuralları

Öğrenme biçimine göre yapay sinir ağları “Danışmanlı (Supervised) Öğrenme” veya “Öğretmenli Öğrenme” ile “Danışmansız (Unsupervised) Öğrenme” veya “Gözetimsiz Öğrenme” olarak ikiye ayrılmaktadır. Danışmanlı öğrenmede ağa veriler hem girdi hem de çıktı değerleri olarak verilmektedir. Fakat danışmansız öğrenmede ise ağa girdi verileri verilerek problemin çözümü ağıdan istenmektedir [30].

Literatür incelediğinde birçok öğrenme algoritmasının kullanıldığı görülebilir. Bu öğrenme algoritmalarından birçoğu ağırlıkların matematiksel olarak güncelleştirilmesi için kullanılmaktadır. Bu algoritmaların birçoğu Hebb kuralından türetilmiştir. Araştırmacılar sürekli olarak insanın öğrenmesine benzeyen yeni öğrenme kuralları geliştirmeye devam etmektedir [31]. Literatürde kullanılan Hebb, Delta, Kohonen ve Hopfield kuralları Sağıroğlu, Erkan ve Erler [15] tarafından aşağıdaki gibi özetlenmiştir.

- **Hebb Kuralı:** Hebb'in Davranış Organizasyonu kitabında açıklanan bu kuralın temelinde, bir nöronun bir başka nörondan girdi alması ve her iki nöronun da aktif olması durumunda (matematiksel olarak aynı işarette sahip olması), nöronlar arasındaki ağırlıklar kuvvetlendirilir.
- **Hopfield Kuralı:** Zayıflatma veya kuvvetlendirme büyülüüğü haricinde Hebb kuralına çok benzemektedir. Eğer istenilen girdi ve çıktılardan her ikisi birlikte aktif veya aktif değilse, öğrenme oranı tarafından bağlantı ağırlığı arttırılır, diğer durumlarda ise azaltılır. Birçok öğrenme algoritmasında, öğrenme katsayısı, oranı veya sabiti vardır. Çoğunlukla 0 ile 1 arasındaki bir değere sahiptirler.
- **Delta Kuralı:** Hebb kuralının bir başka formu olup en sık kullanılan öğrenme algoritmalarındandır. Bu kural, nöronun gerçek çıktısı ile istenilen çıktı değerleri arasındaki farkı azaltan, girdi bağlantılarını güçlendiren ve sürekli olarak değiştiren bir düşünceye dayanmaktadır. Bu kural, ortalama küresel hataya bağlantı ağırlık değerlerinin değiştirilmesiyle (azaltma veya artırma) düşürme prensibine dayanır. Hata, aynı anda bir katmandan bir önceki katmanlara geri yayılarak azaltılır. Ağın hatalarının düşürülmesi işlemi, çıktı katmanından girdi katmanına ulaşınca kadar devam eder. Aynı zamanda geri yayılım, Widrow-Hoff veya en küçük ortalama karesel öğrenme kuralı olarak da anılır.
- **Kohonen Öğrenme Kuralı:** Burada nöronlar öğrenmek için yarışırlar ve kazananın ağırlığı değiştirilir. En büyük çıktı değerine sahip işlemci sinir kazanır ve komşularını uyarma ve yasaklama kapasitesine sahiptir. Bu kuralın hedef çıktı gereksinimi olmadığı için damışmansız öğrenme metodudur.

2.6. Yapay Sinir Ağları Modelleri

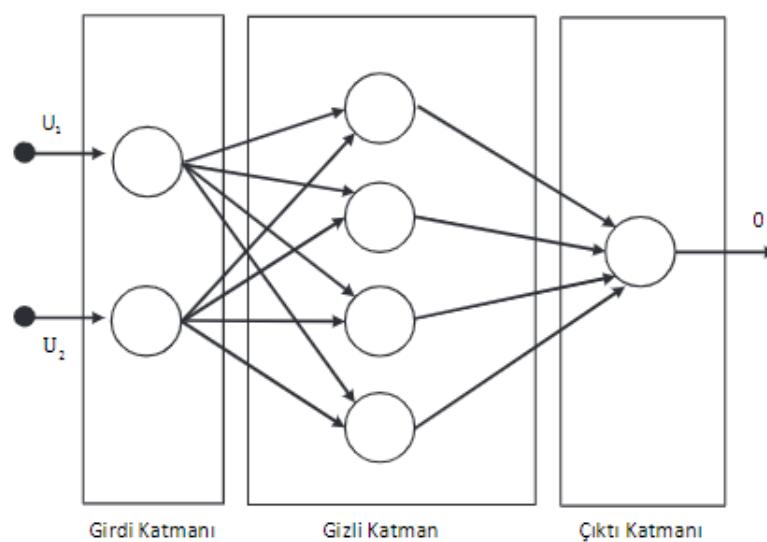
Yapay sinir ağları, hücrelerin birbirleri ile farklı şekillerde bağlanması sonucu oluşur. Hücrelerin çıktıları, ağırlıklar ile ya kendilerine ya da diğer hücrelere girdi olarak bağlanır ve buradaki bağlantınlarda gecikme birimi de

kullanılır. Hücrelerin bağlantı şekilleri, aktivasyon fonksiyonları ve öğrenme kuralları dikkate alındığında farklı YSA modelleri geliştirilmiştir [32].

2.6.1. İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli yapay sinir ağları sınıflandırma, tanıma, tanımlama ve sinyal işleme gibi birçok problemlerin çözümü için elverişlidir [33]. Verilerin yalnızca ileriye doğru, girdi birimlerinden çıktı birimlerine doğru aktığı ağ yapısıdır. Buradaki yapıda sınırlar katmanlar olarak düzenlenmektedir. Bir katmandaki sınırın çıktıları kendisinden sonraki katmana ağırlıklar üzerinden girdi olarak gönderilmektedir [34]. Bilgi, orta ve çıktı katmanında işlendikten sonra ağır çıkıştı oluşturulur. Bu yapısı sebebiyle doğrusal olmayan statik bir işlev gerçekleştirirler. İleri beslemeli YSA'lar girdi tabakası, saklı tabaka ve çıktı tabakasından oluşur [35].

Girdi katmanındaki nöron sayısı, girdi verisi kadar olup her biri bir veri alır. Buradaki veriler ağır temel işlevini gören saklı katmana gönderilir. Saklı tabakaların sayısı probleme göre değişiklik göstermekte olup tasarımcının tecrübeyle belirlenir. Burada girdi tabakasından ağırlıklandırılarak gelen veriler probleme uygun bir fonksiyonla işlenerek bir sonraki katmana gönderilir. İleri besleme aşamasında girdi tabakasındaki nöronlar ağırlıklandırılarak bir taşıma fonksiyonu ile işlenir ve bir sonraki tabakaya ya da doğrudan çıktıya gönderilir. Aşağıdaki Şekil 2.4'te ileri beslemeli YSA modeli gösterilmiştir [36].

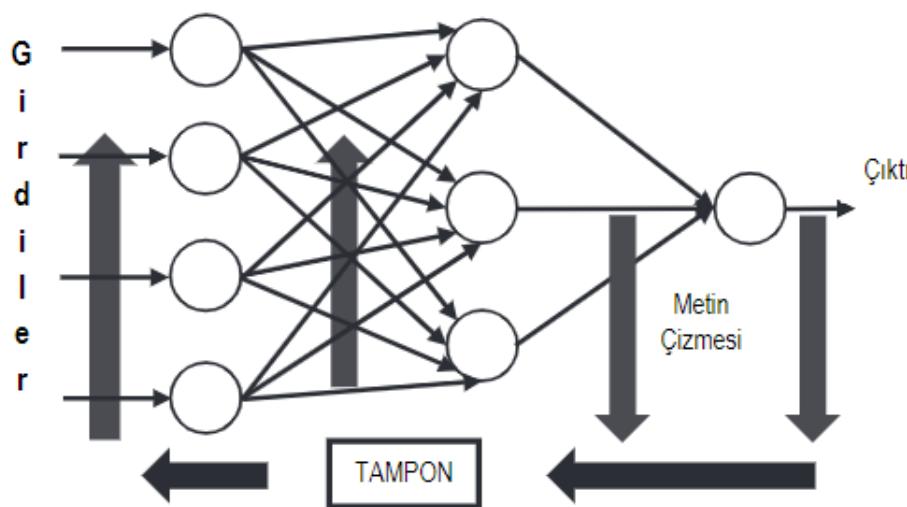


Şekil 2.4. İleri Beslemeli Yapay Sinir Ağlarının Yapısı

2.6.2. Geri Beslemeli Yapay Sinir Ağları

Veri akışının sadece ileriye doğru olmadığı geriye doğru da olabileceği ağ yapılarıdır. Bu yapılarda en az bir tane geri besleme çevrimi olur. Geri besleme, hem aynı katmandaki hücreler arasında hem de farklı katmanlarda ki nöronlar arasında da olabilir [34]. Bu yapısı sebebiyle geri beslemeli yapay sinir ağları, doğrusal olmayan dinamik bir davranış gösterirler [32]. Yapay sinir ağlarının parametrelerinin güncellenmesi için literatürde en sık kullanılan yöntemdir [36].

Bu türdeki ağlar bir tampon aracılığıyla çıktıdan alınan geri besleme sinyalini gizli tabaka ile girdi tabakasına gönderirler. Ağa yeni girdiler alınırken, önceden gönderilen geri besleme sinyali de göz önüne alınmış olur. Geri besleme ağın daha doğru sonuçlar elde etmesine yardımcı olan bir unsurdur. Geri beslemeli YSA'ların yapısı aşağıdaki Şekil 2.5'te gösterilmiştir [37].

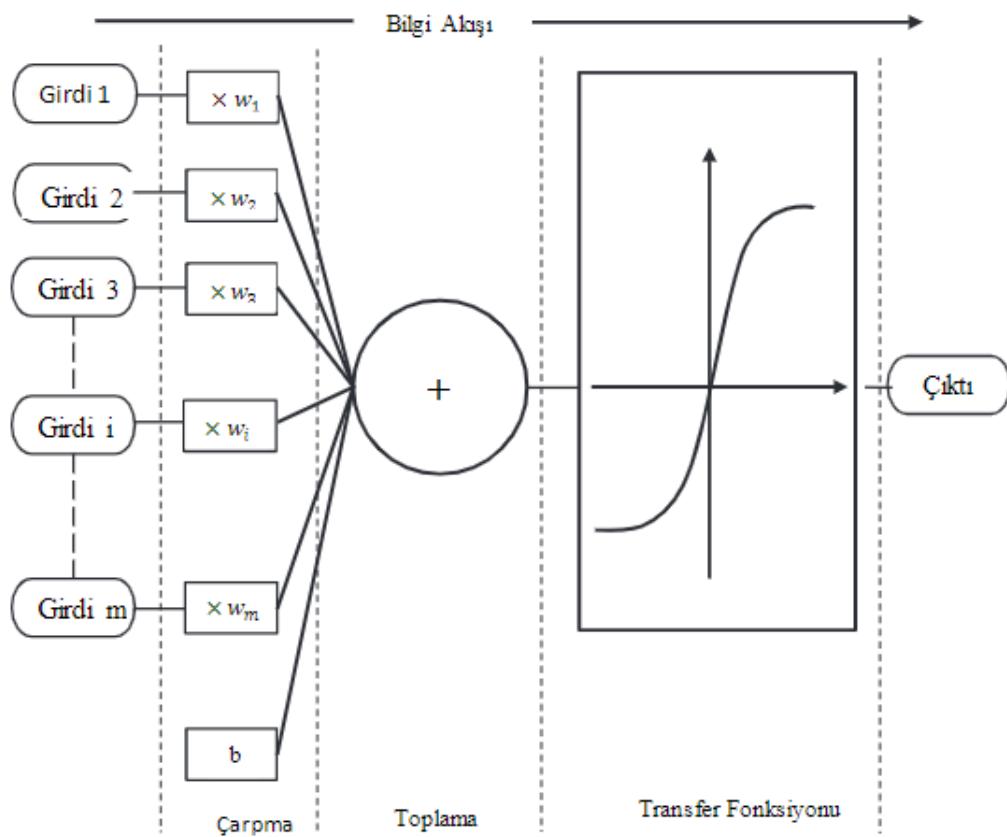


Şekil 2.5. Geri Beslemeli YSA Yapısı

2.7. Yapay Sinir Ağlarının Eğitilmesi

Yapay Sinir Ağları biyolojik sinir ağlarının yapısını ve fonksiyonlarını taklit etmeye çalışan bir matematiksel modeldir. Her yapay sinir ağının temel yapı taşı bir matematiksel fonksiyondan oluşmaktadır. Bunlar çarpma, toplama ve aktifleştirmidir. Yapay sinirin girdi değerleri, her girdinin bireysel ağırlığı ile çarpılması sonucu ağırlıklandırılır. Yapay sinirin orta bölümünde ise tüm ağırlıklandırılmış girdi değerleri ve sapma değerleri toplama işlemine tabi tutulur. Yapay sinirin çıktısında ise daha önceki ağırlıklı girdi-

ler ile sapma değerlerinin toplamı transfer fonksiyonu olarak adlandırılan aktivasyon fonksiyonudur. Buradaki durum aşağıdaki Şekil 2.6'da görsel olarak anlatılmaya çalışılmıştır [38].



Şekil 2.6. Yapay Bir Sınırın Çalışma Prensibi

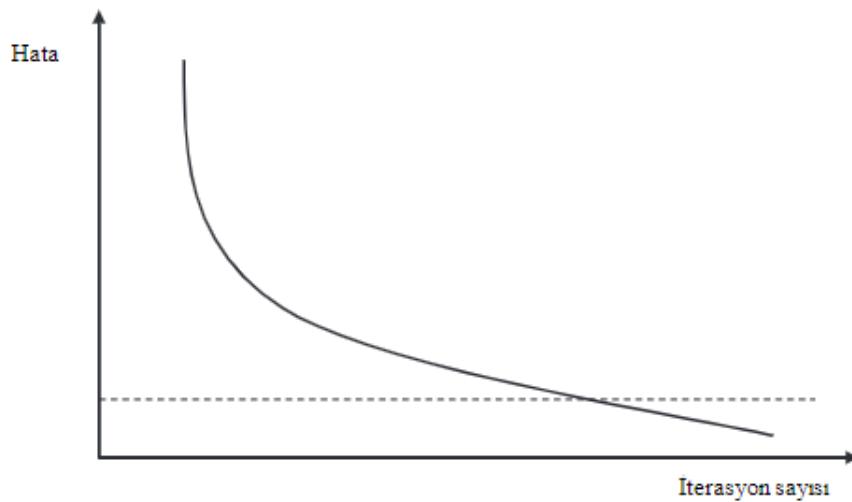
Buradaki yapay sinirin her bir girdisine ait ağırlık değerlerinin belirlenmesi işlemine “ağın eğitilmesi” işlemi denmektedir. İlk başta bu değerlerin tümü rastgele atanırken sonradan ağa verilen örnekler göre ağırlık değerleri değiştirilmiş olur. Burada verilen örnekler dikkate alınarak doğru çıktıları elde edebilecek ağırlık değerlerinin bulunması amaçlanmaktadır. Ağın optimum ağırlık değerlerini elde etmesi, verilen örnekler ile genellemeler yapabilme yeteneği elde ettiğini göstermektedir. Çeşitli öğrenme kuralları olması sebebiyle, ağın çıktı değerleri ve ağırlıklarının değiştirilmesi de buna göre farklılık göstermektedir. Ağın eğitilmesinden sonra son durumun performansını ölçmek için yapılan deneme işlemine “ağın test edilmesi” denir. Test işleminde kullanılan örnekler daha önce kullanılmamış ağın görmediği örneklerden oluşmalıdır [29].

Model kurulumunda hangi verilerin eğitim kümesi, hangi verilerin ise test kümesi olarak kullanılacağı başlangıçta verilmesi gereken bir karardır. Benzer şekillerde verinin miktarına bağımlı olmadan ağıın testi sonucundan anlamsız sonuçlar elde edilirse, kurulan ağıın mimarisi, kullanılan değişkenler, öğrenme algoritması, nöron sayıları, kullanılan gizli katman ve veri tipleri gözden geçirilerek model kontrol edilmelidir [39].

Yapay sinir ağlarının eğitilmesi sürecinde, istenen hata düzeyine ulaşılamaası durumunda öğrenme sürecinde bir kısım değişiklikler yaparak ağıın yeniden eğitilmesi gerekebilir. Bu tür durumlarda yapılabilecekler Karahan [40] tarafından aşağıdaki gibi belirtilmiştir.

- Eğitim başlangıç değerleri değiştirilebilir,
- Sinir ağı topolojisinde, işlemci eleman sayısında ve ara katman sayısında değişikliklere gidilebilir,
- Momentum ve öğrenme katsayıları gibi parametre değişikliklerine gidilebilir,
- Örnekler ve sunum şekli değiştirilerek yeni örnekler ağa sunulabilir,
- Öğrenme setindeki örnek sayısında azalma ve artırma yoluna gidilebilir.

Öğrenme süresinin gereğinden fazla uzun olması sinir ağlarının eğitilmesinde karşılaşılan önemli sorunlardan biridir. Başlangıçta ağırlık değerlerinin büyük olması, ağıın yerel sonuçlara düşmesine ve bir başka yerel sonuca sıçramasına neden olmaktadır. Tam tersi şeklinde de, ağırlıkların küçük olması doğru değerleri bulmanın uzun zaman almasına sebep olacaktır. Bazı problemlerin çözümü birkaç yüz adet iterasyon sürerken bazı problemlerin ise birkaç milyon iterasyon gerektirmektedir. Optimum başlama koşullarının bu durum dikkate alınarak belirlenmesi gerekmektedir. Aşağıdaki Şekil 2.7'de işlemci elemanın ağırlık değerinde yapılacak değişim sayısı olarak adlandırılan iterasyon sayısı ile hata düzeyinin ilişkisi gösterilmektedir [29].



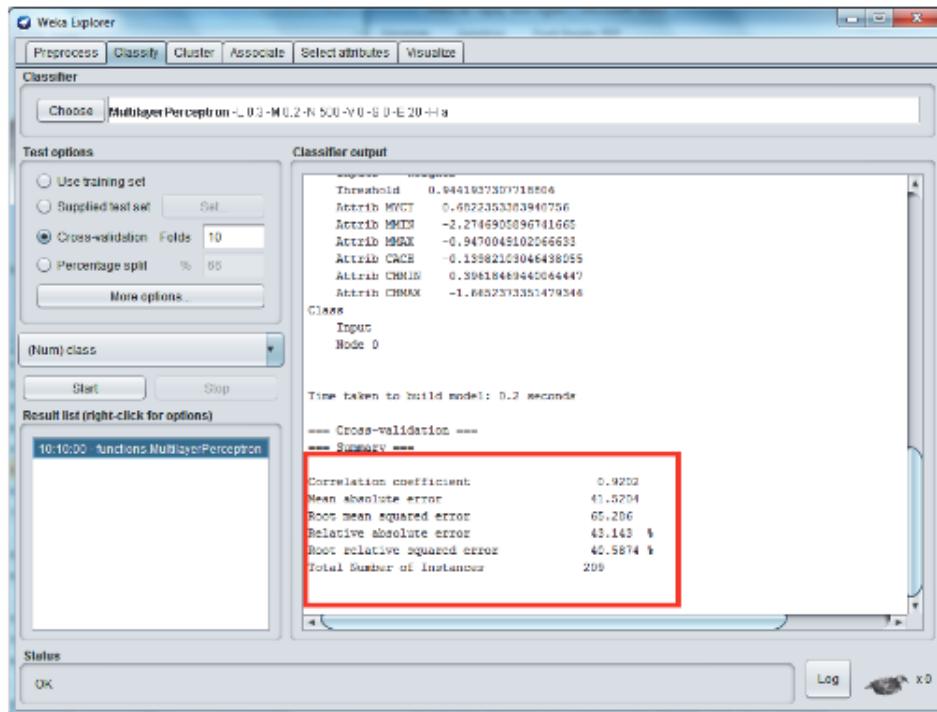
Şekil 2.7. Yapay Sinir Ağlarında İterasyon Sayısı ile Hata Düzeyi İlişkisi

Yukarıdaki Şekil 2.7'de görüleceği üzere iterasyon sayısının artması ağıın öğrenmesini artırmakta ve hata oranını düşürmektedir. Fakat bu durum iterasyon sayısının belirli bir sayıya ulaşmasına kadar geçerlidir. Ağın eğitilme işlemi tamamlandıktan sonra öğrenme durmakta ve daha iyi bir sonuç elde edilememektedir.

3

Tahminin Hata Düzeyinin Belirlenmesi

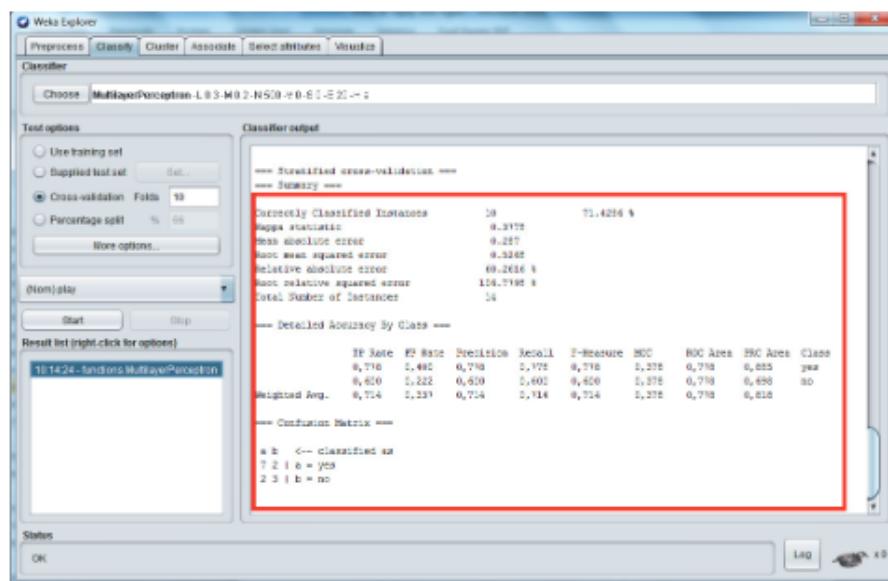
Weka programı iki tipteki tahmin sonuçlarını elde ederken farklı hata formüllerini kullanmaktadır. Aşağıdaki şekilde cpu.arff adlı dosya seçilmiştir. Bu dosyanın tahmin edilen *class* adlı sütunu sayısaldır. Aşağıda sayısal türde sonuçların tahminleri elde edilirken hata ölçüm değerlerinin nasıl olduğu gösterilmektedir.



Şekil 3.1. Sayısal Sonuç Tahmininde Hata Ölçümleri

Nominal veya diğer adıyla kategorik türdeki bir tahmin elde edildiğinde ise hata ölçüm değerleri daha farklı olmaktadır. Weather.arff dosyasının ise

tahmin edilen sütun değeri kategoriktir. Aşağıdaki şekilde ise kategorik bir değerin tahmin edilmesinde hata ölçümlerinin nasıl yapıldığı görülmektedir.



Şekil 3.2. Kategorik Sonuç Tahmininde Hata Ölçümleri

Yukarıdaki iki şekele dikkat edilirse kategorik değer tahminlerinde sayısal değer tahminlerine göre daha fazla hata ölçüm değerleri gösterilmiştir. Bunlar aşağıdaki başlıklarda daha detaylı olarak ele alınmıştır.

3.1. Sayısal Sonuçları Tahmin Etmek

Tahmin edilen talep seviyesi ile gerçekleşen talep seviyesi arasındaki farka bakarak tahmin yönteminin etkinliği ölçülebilir. Bu hata testleri tahmin yapmada kullanılan modelin başarısını izlemede önem arz etmektedir. Bütün istatistik değerleri tahmin edilen talep ile gerçekleşen talep arasındaki farka başka bir şekilde bakarlar. Bunların tümü birbirinden ne kadar uzak olduğunu sayısal değerler ile göstermeye çalışır. Bazen karekökler alınırken bazen de mutlak değerler dikkate alınır. Karekök alılarak aşırı yüksek değerlerin sonuç üzerindeki etkisini azaltmaya çalışılmaktır.

Tesadüfi ve tesadüfi olmayan nedenler olmak üzere hata nedenleri ikiye ayrılır. Doğru değişkenlerin belirlenmemesi, hatalı eğim doğrusu seçilmesi, değişkenler arasında yanlış ilişkiler kullanılması, mevsimsel etkilerin göz ardı edilmesi gibi nedenler tesadüfi olmamakla beraber hatanın sürekli olmasına sebep olurlar. Fakat tahmin modeli tarafından açıklanmayan hatalar ise tesadüfi hata olarak kabul edilir [41].

Aşağıdaki başlıklarda sayısal sınıfları tahmin ederken kullanılan hata değerleri ve hesaplama formülleri ele alınmıştır.

3.1.1. Korelasyon Katsayısı

Korelasyon katsayısı (*Correlation coefficient*) iki değişken arasındaki ilişkiyi veya bir değişkenin daha fazla değişken ile ilişkisini tespit etmek, varsa yönünü ve şiddetini bulmak için kullanılan yöntemdir. Bu değer -1 ile +1 arasında değişmektedir. Değerin -1'e yaklaşması ilişkinin ters yönde olduğu anlamına gelirken, +1'e yaklaşması ise ilişkinin aynı yönde olduğu anlamına gelmektedir. Korelasyon katsayısı +1'e yaklaşıkça kuvvetli pozitif ilişki olduğu söylenirken, -1'e yaklaşıkça kuvvetli negatif ilişki olduğu ve 0'a yaklaşıkça ise ilişkinin azlığı söylenir [42].

Tablo 1. Korelasyon Katsayısı Değer Aralıklarına Göre Yorumları

Korelasyon Katsayısı Değer Aralıkları	Yorum
+0,90 ile +1,00 arası	Pozitif Çok Yüksek Korelasyon
+0,70 ile +0,90 arası	Pozitif Yüksek Korelasyon
+0,40 ile +0,70 arası	Pozitif Normal Korelasyon
+0,20 ile +0,40 arası	Pozitif Düşük Korelasyon
0,00 ile +0,20 arası	Pozitif Çok Düşük Korelasyon
0,00 ile -0,20 arası	Negatif Çok Düşük Korelasyon
-0,20 ile -0,40 arası	Negatif Düşük Korelasyon
-0,40 ile -0,70 arası	Negatif Normal Korelasyon
-0,70 ile -0,90 arası	Negatif Yüksek Korelasyon
-0,90 ile -1,00 arası	Negatif Çok Yüksek Korelasyon

Korelasyon katsayısı (r) hesaplanırken aşağıdaki formül kullanılır.

$$r = \frac{\sum(xy) - \frac{(\sum x)(\sum y)}{n}}{\sqrt{\left(\sum x^2 - \frac{(\sum x)^2}{n}\right)\left(\sum y^2 - \frac{(\sum y)^2}{n}\right)}}$$

3.1.2. Ortalama Mutlak Hata

Tahmin yöntemleri arasında en sık kullanılan hata ölçülerini Üreten [42] dört grupta toplamıştır. Bunlardan biri ortalama mutlak hata (*Mean Absolute Error*)'dır.

$$\text{Ortalama mutlak hata} = \frac{\sum_{i=1}^n |\hat{\theta}_i - \theta_i|}{n}$$

n = örneklem sayısı

θ_i = i sıra numaralı gerçek talep

$\hat{\theta}_i$ = i sıra numaralının tahmin edilen talebi

i = örneklem sırası

Burada tahminde tam başarı elde edilebilmesi için, tahmin hatasının sıfıra eşit olması beklenir. Fakat bu durum gerçekten çok nadir ortaya çıkar.

3.1.3. Hataların Karelerinin Ortalamasının Karekökü

Bir diğer hata ölçüsü hataların karelerinin ortalamasının karekökü (*Root Mean Squared Error*) hesaplanmasıdır. Bu durum aşağıdaki denklem aracılığıyla hesaplanır.

$$\text{Hata kareleri ortalamasının karekökü} = \sqrt{\frac{\sum_{i=1}^n (\hat{\theta}_i - \theta_i)^2}{n}}$$

n = örneklem sayısı

θ_i = i sıra numaralı gerçek talep

$\hat{\theta}_i$ = i sıra numaralının tahmin edilen talebi

i = örneklem sırası

3.1.4. Göreceli Mutlak Hata

Tahmin ortalamasının, talepte oluşan değişikliklere ne kadar uyum sağladığını görebilmek için göreceli mutlak hata (*Relative absolute error*) kullanılabilir. Bu durum aşağıdaki denklem aracılığıyla hesaplanır.

$$\text{Göreceli Mutlak Hata} = \frac{\sum_{i=1}^n |\hat{\theta}_i - \theta_i|}{\sum_{i=1}^n |\bar{\theta} - \theta_i|}$$

n = örneklem sayısı

θ_i = i sıra numaralı gerçek talep

$\hat{\theta}_i$ = i sıra numaralının tahmin edilen talebi

i = örneklem sırası

$\bar{\theta}$ = gerçek taleplerin ortalaması

Buradaki göreceli mutlak hata yapılan tahminin gerçek değerlerle olan yakınlığını izlemenin bir yoludur.

3.1.5. Göreceli Mutlak Hata Karekökü

Tahmin hatasını izlemede kullanılan bir başka yol da Göreceli Mutlak Hata Karekökü (*Root relative squared error*)'dır. Aşağıdaki formül aracılığıyla hesaplanır [43].

$$\text{Göreceli Mutlak Hata Karekökü} = \sqrt{\frac{\sum_{i=1}^n (\hat{\theta}_i - \theta_i)^2}{\sum_{i=1}^n (\bar{\theta} - \theta_i)^2}}$$

n = örneklem sayısı

θ_i = i sıra numaralı gerçek talep

$\hat{\theta}_i$ = i sıra numaralının tahmin edilen talebi

i = örneklem sırası

$\bar{\theta}$ = gerçek taleplerin ortalaması

3.2. Nominal Sonuçları Tahmin Etmek

3.2.1. Hata Matrisi

Kategorik sınıf değerleri tahmin edilirken amaç kaç tane türden kaç tanesinin doğru sınıfa yerleştirildiğini tahmin etmektir. Buradaki sınıflardan hangilerinin hangi sınıfa yerleştirildiğini görmek için hata matrisi (*Confusion Matrix*) tablosu kullanılır. Aşağıdaki şekilde de görüleceği üzere hata matrisi tablosunda satırlar test verilerinin kendi değerlerini gösterirken sütunlar ise sınıflandırıldığı değerleri göstermektedir. Bu durumda “yes” etiketli değerlerin 7 tanesi “yes” olarak doğru sınıflandırılırken 2 tanesi “no” olarak yanlış sınıfta sınıflandırılmıştır. Yine aynı şekilde “no” değerlerinin 3 tanesi “no” olarak doğru sınıfa yerleştirilirken 2 tanesi “yes” olarak yanlış sınıfta yerleştirilmiştir.

Correctly Classified Instances	10	71.4286 %
Kappa statistic	0.3778	
Mean absolute error	0.287	
Root mean squared error	0.5268	
Relative absolute error	60.2616 %	
Root relative squared error	106.7798 %	
Total Number of Instances	14	
--- Detailed Accuracy By Class ---		
TP Rate	FP Rate	Precision
0,778	0,400	0,778
0,600	0,222	0,600
Weighted Avg.	0,714	0,714
Recall	F-Measure	MCC
0,778	0,778	0,378
0,600	0,600	0,378
ROC Area	PRC Area	Class
0,778	0,885	yes
0,778	0,698	no
	0,778	0,818
--- Confusion Matrix ---		
a b	<-- classified as	Sınıflandırılan Değerler
7 2 a = yes		
2 3 b = no		Kendi Değerleri

Şekil 3.3. Hata Matrisi Örneği

Hata matrisi tablosu incelenerek test verilerinden kaç tanesinin doğru ve yanlış yerleştirildiği sonucu elde edilebilir. Yukarıda belirtildiği üzere 7 tane “yes” sınıfı ile 3 tane “no” sınıfı doğru yerleştirilmiştir. Bunların toplamı 10 tane yapar.

3.2.2. Doğru Yerleştirme Başarısı

Toplam test verisi sayısı ise Toplam Örnek Sayısı (*Total Number of Instances*) olarak 14 tane gösterilmiştir. Toplam Doğru Yerleştirilen Örnek Sayısı (*Correctly Classified Instances*) değeri üst kısmında gösterilmektedir. Bu sayının hemen yanına ise Toplam Doğru Yerleştirilen Örnek Sayısının Toplam Örnek Sayısına oranı yani $10/14=0.714286$ değerini göstermektedir. Buradaki değer oransal olduğu için 100 ile çarpılarak yüzdesel değer gösterilmiştir. Aşağıdaki şekilde bu değerlerin yerleri ve açıklamaları işaretlenmiştir.

Correctly Classified Instances	10	71.4286 %	Doğru yerleştirilen örnek sayısı ve toplam örnek sayısına oranı
Kappa statistic	0.3778		
Mean absolute error	0.287		
Root mean squared error	0.5268		
Relative absolute error	60.2616 %		
Root relative squared error	106.7798 %		
Total Number of Instances	14		Toplam Örnek Sayısı
--- Detailed Accuracy By Class ---			
TP Rate	FP Rate	Precision	
0,778	0,400	0,778	
0,600	0,222	0,600	
Weighted Avg.	0,714	0,714	
Recall	F-Measure	MCC	
0,778	0,778	0,378	
0,600	0,600	0,378	
ROC Area	PRC Area	Class	
0,778	0,885	yes	
0,778	0,698	no	
	0,778	0,818	
--- Confusion Matrix ---			
a b	<-- classified as	Sınıflandırılan Değerler	
7 2 a = yes			
2 3 b = no		Kendi Değerleri	

Şekil 3.4. Doğru Yerleştirilen Örnek Sayısı ve Oranı

3.2.3. Kappa İstatistiği

Kappa istatistiği (*Kappa statistic*) satır ve sütun sayısı eşit olan tablolarada iki değişken arasındaki uyumu ölçmek için kullanılır. Kolay hesaplanıp pratik olarak yorumlanabilen şans ile beklenen arasındaki uyumu düzeltmeyi temel alır. Kappa istatistiği, -1 ile +1 arasında değerler alır ve 0'dan küçük olması durumunda uyum olmadığı 1'e yaklaşıkça ise tam bir uyumun olduğunu gösterir. Kappa istatistiği hesaplanırken iki farklı olasılık hesaplanır. Bunlar $Pr(a)$ ve $Pr(e)$ 'dır. $Pr(a)$ iki değerlendirmeli için gözlemlenen uyumlaraın toplam orantısı iken, $Pr(e)$ bu uyumun şansa bağlı ortaya çıkma olasılığıdır. Bu iki olasılık üzerinden Cohen'in kappa istatistiği için kullanılacak formül aşağıdaki gibidir [44].

$$K = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}$$

3.2.4. Tahmin Sınıflarının Doğruluk Detayları

Tahmin edilen sınıf kategorik olması sebebiyle her bir kategorinin ne kadar doğru sınıflandırıldığına detaylı analizi görülebilmektedir. Bunun için öncelikle ikili bir sınıflandırma tahmininde hata matrisindeki değerleri doğru anlamak gereklidir. Öncelikle aşağıdaki tanımlar verilmiştir.

- **Doğu Pozitif (True Positive-TP):** Gerçek değeri pozitif olup pozitif olarak tahmin edilenler. Türkçe *dp* olarak kısaltılacaktır.
- **Yanlış Negatif (False Negative-FN):** Gerçek değeri pozitif olup negatif olarak tahmin edilenler. Türkçe *yn* olarak kısaltılacaktır.
- **Yanlış Pozitif (False Positive-FP):** Pozitif olarak tahmin edilmiş fakat gerçek değeri negatif olanlar. Türkçe *yp* olarak kısaltılacaktır.
- **Doğu Negatif (True Negative-TN):** Negatif olarak tahmin edilmiş ve gerçek değeri negatif olanlar. Türkçe *dn* olarak kısaltılacaktır.

Kontenjans tablosunun pozitif olma durumu dikkate alınırsa Doğru Pozitif (True Positive-TP), Yanlış Negatif (False Negative-FN), Yanlış Pozitif (False Positive-FP), Doğru Negatif (True Negative-TN) değerleri aşağıdaki tabloda görüldüğü gibi olur.

Tablo 2.1. Pozitif Sınıf Kabulü İçin Kontenjans Tablosu

		Tahmin Değerleri		
		a	b	
Gerçek Değerler	a	Doğru Pozitif (TP)	Yanlış Negatif (FN)	Gerçek a toplamı
	b	Yanlış Pozitif (FP)	Doğru Negatif (TN)	Gerçek b toplamı
	Toplam	Toplam Pozitif	Toplam Negatif	

Kontenjans tablosunun negatif olma durumu dikkate alınırsa Doğru Pozitif (True Positive-TP), Yanlış Negatif (False Negative-FN), Yanlış Pozitif (False Positive-FP), Doğru Negatif (True Negative-TN) değerleri aşağıdaki tabloda görüldüğü gibi olur. Dikkat edilirse aşağıdaki tablo bir öncekinin tersinin alınmış halidir.

Tablo 2.1. Negatif Sınıf Kabulü İçin Kontenjans Tablosu

		Tahmin Değerleri		
		a	b	
Gerçek Değerler	a	Doğru Negatif (TN)	Yanlış Pozitif (FP)	Gerçek a toplamı
	b	Yanlış Negatif (FN)	Doğru Pozitif (TP)	Gerçek b toplamı
	Toplam	Toplam Pozitif	Toplam Negatif	

Yukarıda Doğru Pozitif (dp), Yanlış Negatif (yn), Yanlış Pozitif (yp), Doğru Negatif (dn) değerleri tanımlanmıştır. Bunların oranları hesaplanırken aşağıdaki formüller kullanılmaktadır.

$$dp \text{ oranı} = \frac{dp}{(dp + yn)}$$

$$dn \text{ oranı} = \frac{dn}{(dn + yp)}$$

$$yp \text{ oranı} = \frac{yp}{(yp + dn)}$$

$$yn \text{ oranı} = \frac{yn}{(yn + dp)}$$

Weka programı özellikle doğru pozitif oranı ve yanlış pozitif oranı değerlerini hem her bir kategori için oranlarını hem de tüm kategoriler için ağırlıklı ortalama sonucunu hesaplamaktadır. Aşağıdaki şekilde bu durum gösterilmiştir.

Classifier output									
Mean absolute error	0.287								
Root mean squared error	0.5268								
Relative absolute error	60.2616 %								
Root relative squared error	106.7798 %								
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	0,400	0,778	0,778	0,778	0,378	0,778	0,885	yes
	0,600	0,222	0,600	0,600	0,600	0,378	0,778	0,698	no
Weighted Avg.	0,714	0,337	0,714	0,714	0,714	0,378	0,778	0,818	
--- Confusion Matrix ---									
a b	<-- classified as								
7 2		a = yes							
2 3		b = no							

Şekil 3.5. Doğru Pozitif Oranı ve Yanlış Pozitif Oranı

Kategorik değişkenleri tahmin ederken yukarıda bahsedilen hesaplama değerlerinin dışında Kesinlik (*Precision*), Hassasiyet (*Recall*) ve F-Ölçüsü (*F-Measure*) değerleri de hesaplanmaktadır. Özellikle sınıfların çok dengezsiz olduğu durumlarda yararlanılan bir ölçütür. Bu değerler aşağıdaki formüller ile hesaplanmaktadır.

$$\text{Hassasiyet } (p) = \frac{dp}{dp + yn}$$

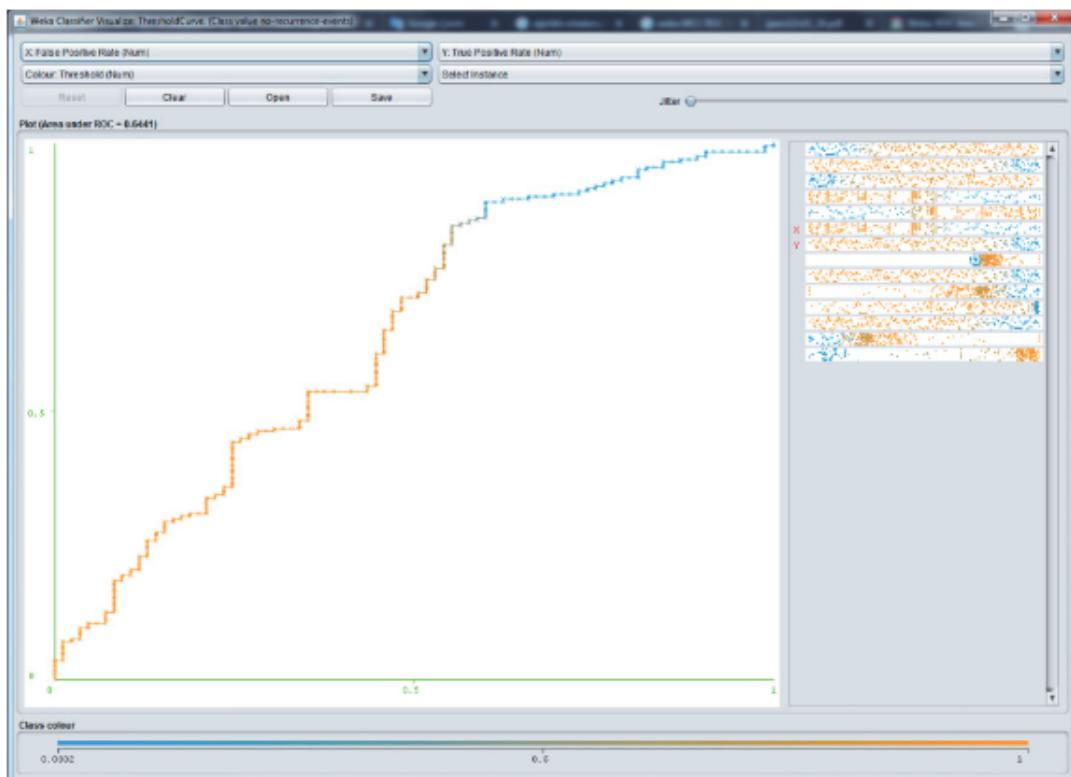
$$\text{Kesinlik } (r) = \frac{dp}{dp + yp}$$

$$F - \text{Ölçüsü} = 2 \times \frac{\text{Kesinlik} \times \text{Hassasiyet}}{\text{Kesinlik} + \text{Hassasiyet}} = 2 \frac{pr}{p + r}$$

Aşağıdaki doğruluk oranı ise doğru tahmin edilen sınıfların toplam örneklem sayısına oranıdır. Başarı ölçütü olarak en çok kullanılan formül olmakla birlikte çok kolay olarak hesaplanır.

$$\text{Doğruluk Oranı} = \frac{dp + dn}{dp + dn + yp + yn}$$

Doğru pozitif oranları ile yanlış pozitif oranlarının bir grafik üzerine yerleştirilmesi ile ROC eğrileri oluşturulur. Alıcı İşlem Karakteristikleri (*Receiver Operating Characteristic*) olarak Türkçeleştirilen bu eğrilerde x ekseninde yanlış pozitif (*yp*) oranı gösterilirken y ekseninde ise doğru pozitif (*dp*) oranı gösterilir. Aşağıdaki şekilde ROC eğrisi gösterilmektedir.

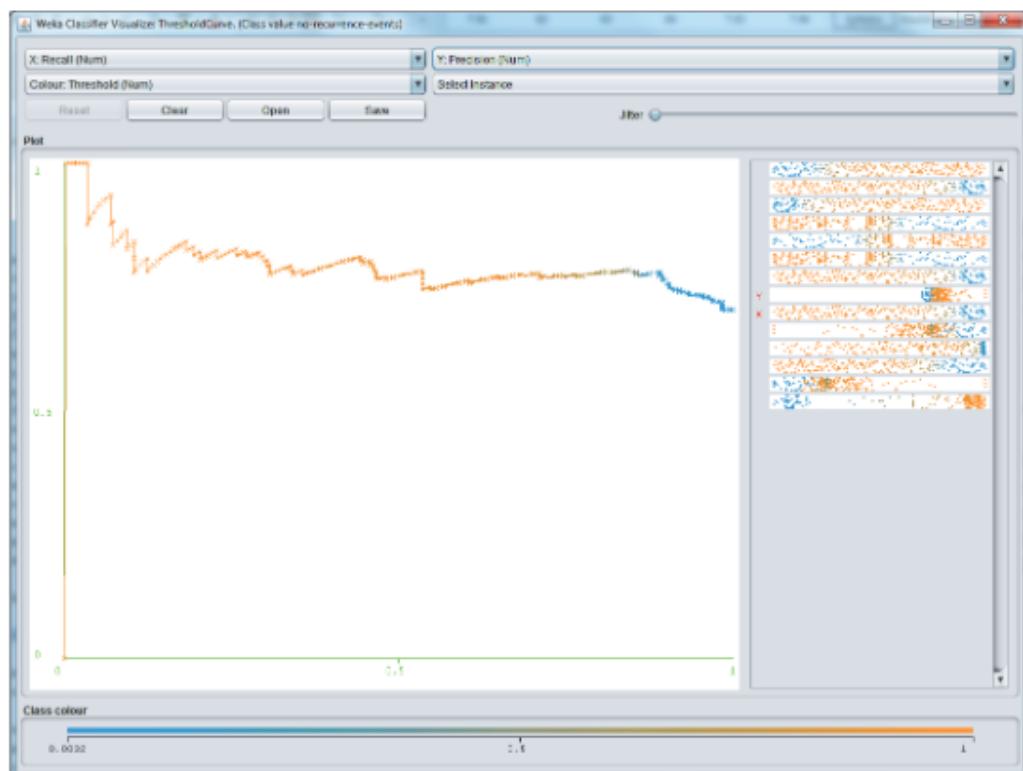


Şekil 3.6. ROC Eğrisi

Doğru pozitif (*dp*) oranının 1'e yaklaşıkça daha iyi tahminler elde edildiği yanlış pozitif (*yp*) oranının ise 0'a yaklaşıkça daha az hatalı tahminler elde edildiği bilinmektedir. Bu sebeple grafik sol üst köşeye doğru ne kadar yaklaşırsa o kadar iyi sınıflandırıldığı kabul edilmektedir. Başka bir deyişle grafiğin altında kalan alan ne kadar büyükse ve 1'e yaklaşıyorsa o kadar başarılı bir sınıflandırma olduğu söylenir. Grafiğin altında kalan alan ROC Alanı (*ROC Area*) olarak ifade edilir ve Weka tarafından otomatik olarak hesaplanarak çıktı ekranında gösterilir.

x ekseninde Hassasiyet (*Recall*) ve y ekseninde Kesinlik (*Precision*) değerlerinin yer aldığı grafik ise PRC grafiği (*Precision-Recall Curve*) olarak geçer. Bu grafiğin altında kalan alan ise PRC Alanı (*PRC Area*) olarak Weka tarafından otomatik hesaplanır ve çıktı ekranında gösterilir. Buradaki

değerin 1'e yaklaşması yüksek hassasiyet ile doğru sonuçlar ürettiğini gösterir. Aşağıdaki şekilde PRC grafiği gösterilmiştir.



Şekil 3.7. PRC Grafiği

4

WEKA

Weka yeni Zelanda'daki Waikato Üniversitesi'nde geliştirilmiş olan ve GNU ile GPL lisansına sahip bir yazılımdır. Yani halka açık, ücretsiz ve açık kaynaklı olarak sunulmakta olan bir yazılımdır. "Waikato Environment for Knowledge Analysis" ifadesinin baş harflerinden ismini almıştır. Java tabanlı olarak yazılan bu programın adı aynı zamanda Yeni Zelanda adalarında bulunan, uçamayan ve soyu tükenmeye olan bir kuş adıdır.



Şekil 4.1. Weka programı açılış ekranı introsu

Veri madenciliği ve makine öğrenmesi uygulamaları yapmak amacıyla kullanılan bu yazılım sık kullanılan hemen hemen tüm algoritmaları kendi içinde barındırmaktadır. Veri kümeleri üzerinde kendi içindeki mevcut yöntemleri kullanarak hızlı bir şekilde deneyebilmek için üretilmiştir. Bu özelliğinin yanı sıra sonuçların da detaylı analizine imkân tanımaktadır. Program

aracılığıyla genel olarak sınıflandırma, kümeleme ve birliktelik gibi temel veri madenciliği işlemleri yapılmaktadır. Linux, Windows ve Macintosh başta olmak üzere tüm sistemlerde çalışabilmektedir.

Weka programı giriş verilerini hazırlamak, girdi verileri ve öğrenmenin sonucunu görselleştirmek ve bunların yanı sıra öğrenme şemalarını istatiksel olarak değerlendirmek gibi tüm deneysel veri madenciliği süreçlerinde kapsamlı destek sağlar. Çok çeşitli öğrenme algoritmalarını sunarken aynı zamanda çok çeşitli ön işleme araçları içermektedir. Program kullanıcıları farklı yöntemleri karşılaştırabilir ve mevcut probleme yönelik en uygun olan seçeneği görebileceği arayüzler sunar.

Verileri tanıtmak işin en önemli parçasıdır. Weka birçok farklı veri görselleştirme olanağı ve veri ön işleme araçları sağlamaktadır. Weka'daki tüm algoritmalar girdilerini bir dosyadan okur veya tek bir tablodan (*ilişkileri yapılmış*) sorgu olarak çeker. Weka kullanmanın yolu, herhangi bir veri kümesine öğrenme yöntemlerinden birini uygulamak ve çıktı hakkında detaylı bilgi elde etmek için çıktıları analiz etmektir. Bir başka yol ise yeni verilere yönelik tahminler üretmek için, daha önce öğrenilmiş ve kaydedilmiş modeli seçmek ve sonuçlar üretmektir. Üçüncü bir yöntem ise birçok farklı öğrenme algoritmasını uygulamak ve bunların performanslarını karşılaştırmaktır. Tüm algoritmalar ayarlanabilir parametrlere sahiptir. Performans ölçümlünde kullanılan değerlendirme modülü ise ortak ve tektir. İstenirse de filtreler olarak adlandırılan verileri önceden işlemek için kullanılan araçlar ile filtre seçilir ve gereksinimlere göre uygulanabilir.



Şekil 4.2. Weka programı ana sayfası

Program aracılığıyla yapılan işlemler dört farklı arayüz kullanılarak yapılabilir. Bunlar [45];

- **Explorer:** Veri ön işleme, kümeleme, sınıflandırma, nitelikleri seçme ve grafik çizimleri yapmak amacıyla kullanılan temel ekranları barındırır.
- **Experimenter:** Explorer arayüzü kullanılarak yapılamayan iş ve işlemleri yapmak için kullanılan daha detaylı bir kullanım sunan ekranları barındırır. Bu ekran ile sınıflandırma ve regresyon teknikleri kullanılırken, hangi problemler için en uygun yöntem ve en uygun parametre değerlerinin kullanılacağına karar verir. Bunu yapmak için sınıflandırma ve filtreleme yöntemleri için farklı parametreler çalıştırmayı kolaylaştırır, performans istatistiklerini toplar ve anlamlılık testleri gerçekleştirerek süreci otomatize eder.
- **KnowledgeFlow:** Bağlantıların akışlar ile gösterildiği ekranları barındırır. Bu ekran ile öğrenme algoritmalarını ve veri kaynaklarını temsil eden kutuları sürüklemeye ve istenilen konfigürasyonda birleştirmeye olanak sağlar. Böylelikle veri kaynaklarını, ön işleme araçlarını, öğrenme algoritmalarını, değerlendirme yöntemlerini ve görselleştirme modüllerini temsil eden kutuların birbirine bağlanmasıyla bir veri akışı belirlenir. Filtreler ve öğrenme algoritmaları artımlı öğrenme yeteneğine sahip ise veriler aşamalı olarak yüklenir ve işlenir.
- **Workbench:** Bu arayüz diğer üç ekranı ve kullanıcının yüklediği ekenglileri tek bir uygulamada birleştirerek kullanım imkânı sağlayan ekranlardır. Kullanıcı tarafından hangi ayarların ve ekenglilerin görüneceğini belirlemeye olanak tanıyacak şekilde yapılandırılabilen bir ekranıdır.
- **Simple CLI:** Console ekranı ile kullanıcının weka'nın tüm özelliklerine erişim sağlayan metin komutları girerek ham formda kullanılmasına imkân tanıyan ekranıdır.

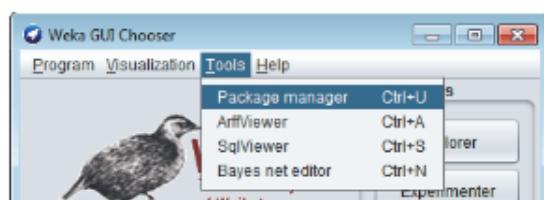
4.1. Weka Programını İndirme

<http://www.cs.waikato.ac.nz/ml/weka> adresi üzerinden indirilerek kurulumu yapılabilir. Weka programının çalıştırılabilmesi için Java programının da kurulu olması gerekmektedir. Eğer sisteminizde kurulu değilse onun da kurulumunu yapmalısınız.

4.2. Paket Yönetim sistemi

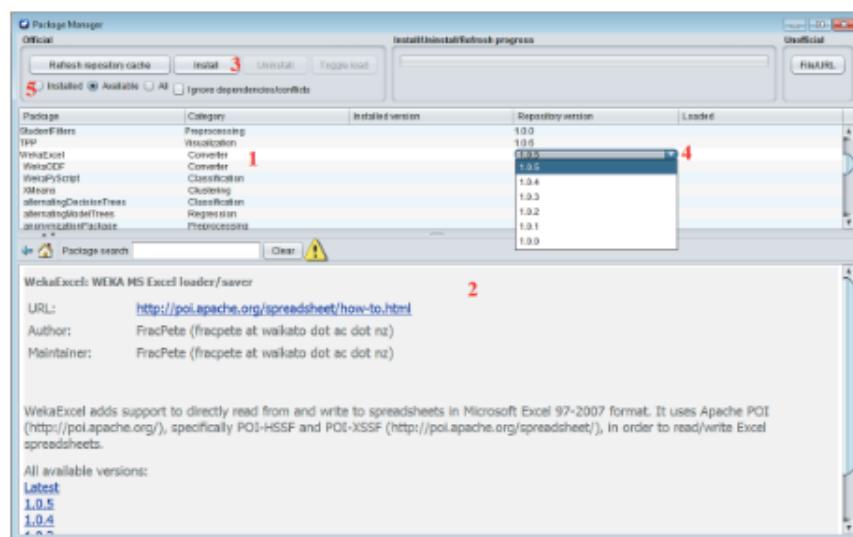
Weka yazılımı muhtemelen bu kitabı okuduğunuz zaman kendisini büyük ölçüde geliştirmiş hatta yeni sürümlerini çıkartmış olacaktır. Weka'nın yeni sürümlerinde yeni algoritmalar veya özellikler görmeyiz son derece muhtemeldir. Bunların bazılarını sürüm güncellemesi ile yaparken bazılarını ise ek paket kurulumları ile kullanılabılır kılacaklardır.

Weka programı ilk kurulduğunda varsayılan olarak sık kullanılan ve genel ihtiyaçları karşılayan algoritmalar ve özellikler ile gelmektedir. Fakat ileri düzey algoritmalar kullanmak isteyenler için ise bu algoritmalar ve özellikler ek paketler halinde kullanıcının isteğine bağlı olarak internet indirilerek kullanılabilir olmaktadır. Eklenti kurulumu yapıldıktan sonra internet olmadan da kullanılabilecektir.



Şekil 4.3. Weka Paket Yöneticisi Açıma Ekranı

Weka programı içerisinde paket yöneticisine erişebilmek için üst menüde yer alan Tools menüsü altında yer alan “Package manager” adlı menüye tıklanır. Bu durum Şekil 4.4’te gösterilmiştir.

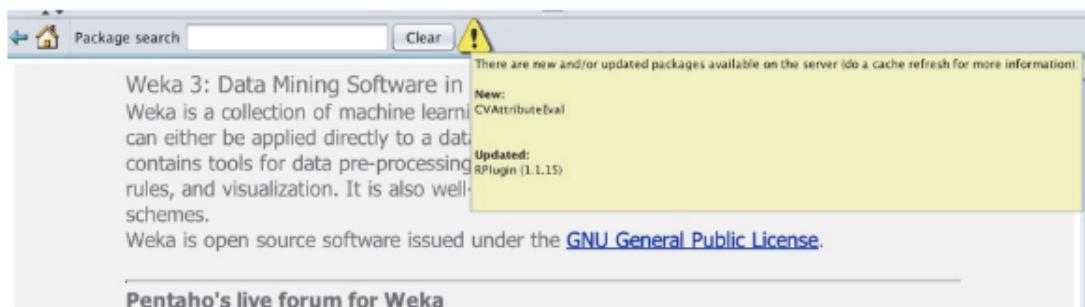


Şekil 4.4. Paket Yönetici Üzerinden Ek Paket Kurma

Paket yöneticisi açıldığında eklentiler listelenmektedir. Eklentilerin veya diğer adıyla paketlerin kategorisi, yüklenip yüklenmediği, yüklü olan versiyonu ve var olan güncel sürüm bilgileri gösterilmektedir. Paket adı sıralamalı olarak sıralanan liste içinden istenirse “Package search” kısmından arama yapılabilir. Şekil 4.4’te görüldüğü gibi 1 numara ile belirtilen yerden paket seçimi yapılınca ilgili pakete yönelik detaylı bilgiler alt kısımdaki 2 numara ile belirtilen yerde gösterilmektedir. Paketin ne işe yaradığı konusunda kısa bir açıklama ve eski sürüm gibi bilgiler gösterilmektedir. Örneğin burada seçili olan WekaExcel paketi Weka içine verileri excel dosyalarından da ekleyebilme imkânı sağlayan eklentidir. İlgili paket seçildikten sonra 3 numara ile belirtilen yerden *Install* butonuna basılmasıyla paketin yüklemesi başlatılacaktır. İnternet ve bilgisayar hızınıza bağlı olarak kısa bir süre içerisinde paket yüklenecektir ve pakete ait özellikler programın kapatılıp açılmasıyla kullanılabilir hale gelecektir.

İstenirse 4 numara ile gösterilen yerden farklı bir sürümün de yüklenmesi sağlanabilir. İhtiyaç duyulması halinde yüklü olan paketler de seçilip *Unistall* butonu ile kaldırılabilir. 5 numara ile gösterilen yerden ise liste içinden sadece yüklü olanlar, yüklenmemiş olanlar ve tümü şeklinde filtrelemeler yapılabilir. Bazı durumlarda herhangi bir paketin çalışabilmesi için bir başka paketin varlığına ihtiyaç duyulabilir. Bu durumda kullanıcıya bilgi vererek ihtiyaç duyulan paketin de otomatik olarak yüklenmesi sağlanır.

Yeni bir paket veya mevcut bir paketin yeni sürümü çıktıığında paket yöneticisi kullanıcıya sarı bir uyarı simgesi ile bilgi verir. Buradan istenirse paketin yeni sürümü indirilebilir. Şekil 4.5’tе bu durum gösterilmiştir.

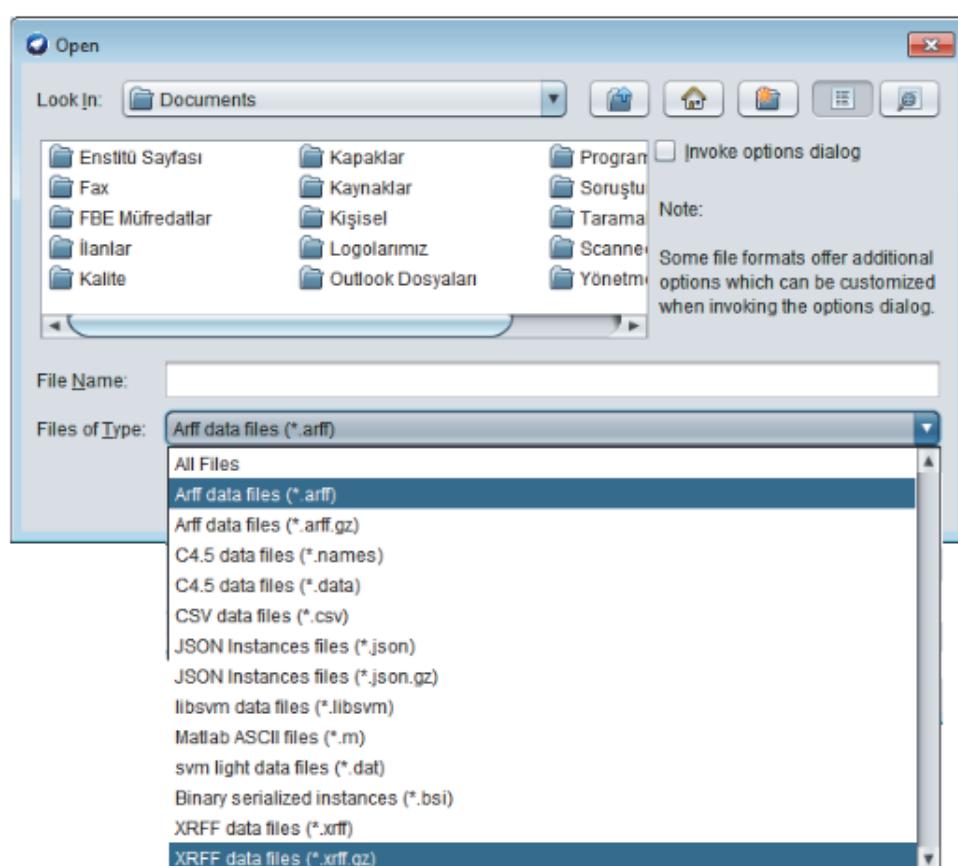


Şekil 4.5. Paketin Yeni Sürümü Olduğu Ekranı

4.3. Başlangıç

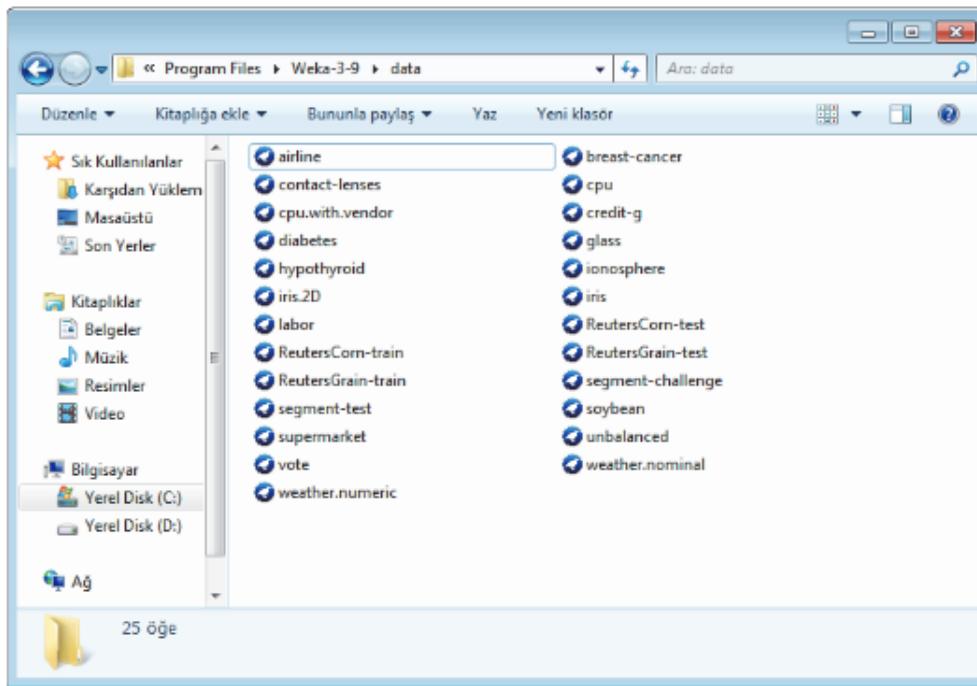
4.3.1. Veri Dosyası ve Formatları

İlk sekme olan önişleme sekmesinde verilerin kayıtlı olduğu dosya seçilir ve seçili verilere ait küçük bir önişleme analizi görüntülenir. Verilerin kayıtlı olduğu dosya için birden fazla türde format kullanılabilir. Yaygın olarak kullanılan ARFF (Attribute-Relation File Format) uzantılı dosya formatları olmasının yanı sıra CSV, DAT, JSON gibi uzantılara sahip format-taki dosyalar da kullanılabilir.



Şekil 4.6. Explorer Penceresinden Dosya Seçme

Her bir dosya formatının özelliklerini incelemeden en çok kullanılan ARFF ve CSV dosya formatının özelliklerinden bahsedilecektir. Weka programı kurulum sırasında kullanıcıların basit birkaç testi yaparak programı kullanabilmesini sağlamak için birtakım verileri ARFF uzantılı dosyalar olarak bilgisayara kaydeder. Bunlar Şekil 4.7'de görülmektedir.



Şekil 4.7. Weka Programı Kurulurken Gelen Bazı ARFF Dosyaları

Örnek olarak Şekil 4.8'deki diabetes.arff dosyasını incelersek bu dosya Dünya Sağlık Örgütü verilerine göre bir hastanın diyabet belirtileri göstermediğinin verileridir. ARFF dosyalarında kullanılan yüzde (%) simgesi açıklama satırlarını ifade eder. En başta @relation ile başlayan ve bir boşluk karakteri sonrası dosyanın adı yazılır. Sonra alt satırlara geçerek @attribute ile başlayıp bir boşluk bırakılır ve verilere ait sütun başlığı ve nitelik adı yazılır. Burada boşluk karakter kullanılmaz fakat kullanılmak istenirse tek tırnaklar içeresine alarak yazılabilir. Nitelik adı yazıldıktan sonra bir boşluk bırakıp bu niteliğin veri türü belirtilir. Veri türleri aşağıdakilerden biri olabilir.

- **Numeric:** Sayısal veri türüdür. Nokta ile ayrılmış tam sayı olmayan değerler de kullanılabilir. Örneğin; “@attribute yas numeric” tanımlaması kişilerin yaşını sayısal olarak girmek için kullanılır.
- **Nominal:** Tek tırnaklar arasına yazılarak ifade edilen metin türündeki kategorik değerlerdir. Fakat buradaki metinlerin türleri sabittir ve nitelik adı yazıldıktan sonra bir boşluk bırakılarak metin türleri süslü parantezler içeresine yazılır. Diğer bir deyişle gruplanmış metinler için kullanılır.
- **String:** Tek tırnaklar arasına yazılan çok uzun metinler halindeki veriler için kullanılır. Metin analizi yapıılırken kullanılan veri türüdür.

- **Real:** Nokta ile ayrılmış ondalıklı sayılarından oluşan veri tipidir.
 - **Integer:** Tam sayı veri tipidir.
 - **Date:** Tarih tipindeki veriler için kullanılır. Tarih tipi verilirken formatı da 'yyyy-MM-dd' veya istenilen farklı bir şeklinde belirtilmelidir.

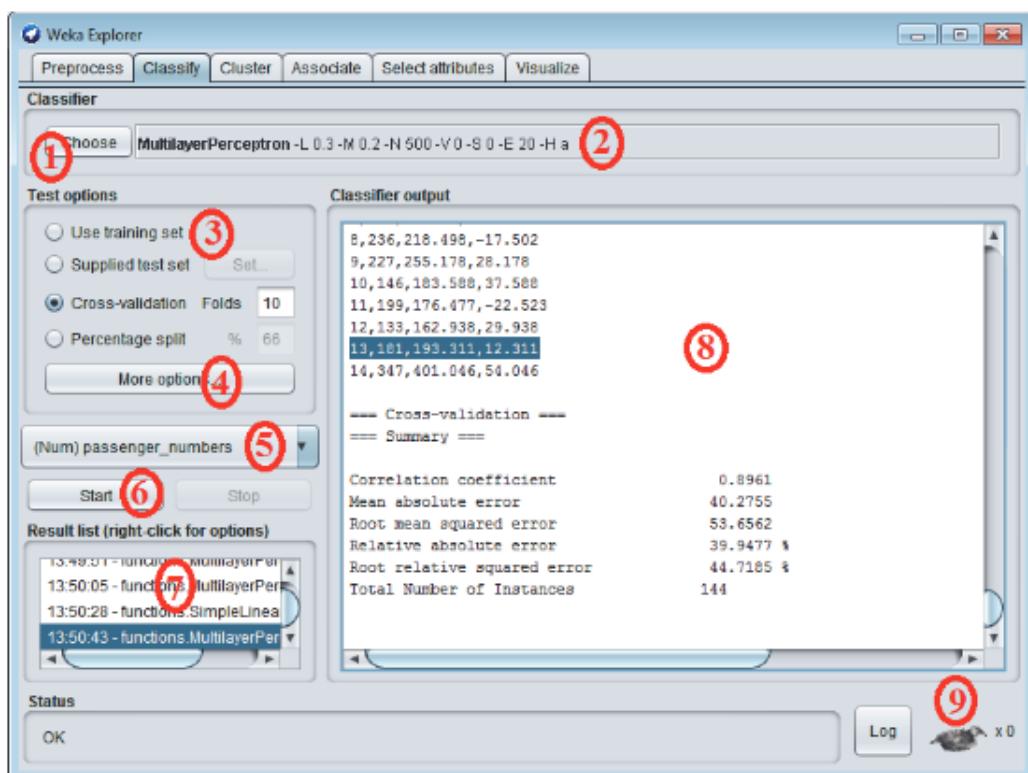
Şekil 4.8. Diabetes.arff Dosyası

@Data ifadesinin bir altından başlayarak veriler yazılır. Veriler arasında soru işaretçi (?) varsa bu bilinmeyen veya eksik bir değeri göstermektedir. Boş veriler için boşluk veya 0 (sıfır) yazmak aynı anlam taşımamaktadır. Boşluk karakteri bırakılması durumunda hata alınacaktır. Ayrıca veriler birbirinden ayrılırken virgül kullanılmak zorunda değildir. Veriler arasında bir sekme bırakılarak da kullanılabilir. Dosyanın hiçbir yerinde Türkçe karakter kullanılmamaya özen gösterilmelidir. Farklı sorunların ortaya çıkmasına sebep olabilmektedir.

4.3.2. Örnek Bir Analiz Yapma

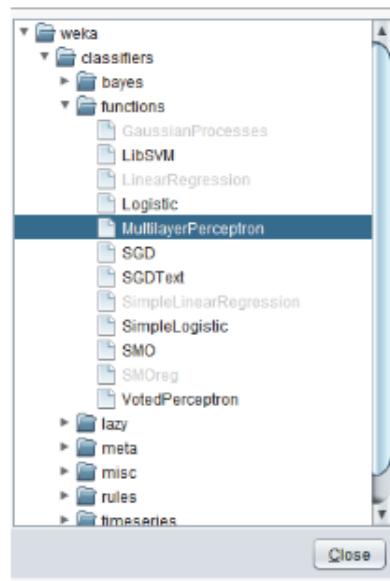
Örnek bir analizin nasıl yapıldığını görmek için aşağıdaki adımların sırası ile yapılması gerekecektir.

1. Öncelikle airline.arff adlı dosyanın *Open file* ile seçilmesi gerekir.
2. *Classify* sekmesine geçiş yapılır.
3. 1 numara ile (*Choose*) belirtilen buton ile açılan hiyerarşik yapıda listeden istenilen sınıflandırıcı seçilir.
4. 6 numaralı (*Start*) buton ile seçili sınıflandırıcı analizi başlatılır.



Şekil 4.9. Örnek Bir Analiz Sonuç Ekranı

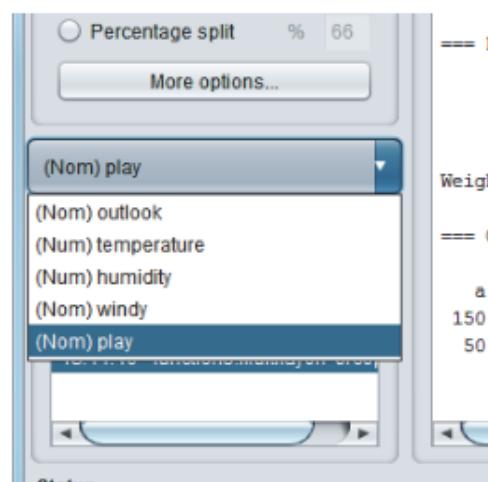
Bu sırada işlem veri miktarı ve parametrelere göre belirli bir süre alabilir. Bu süre içerisinde 9 numarada belirtilen kuş aşağı ve yukarı hareket ederek işlem yapıldığını ifade eder. Eğer kuş oturmuş durumda ise işlemin bittiği anlaşılmaktadır.



Şekil 4.10. Analiz Seçim ekranı

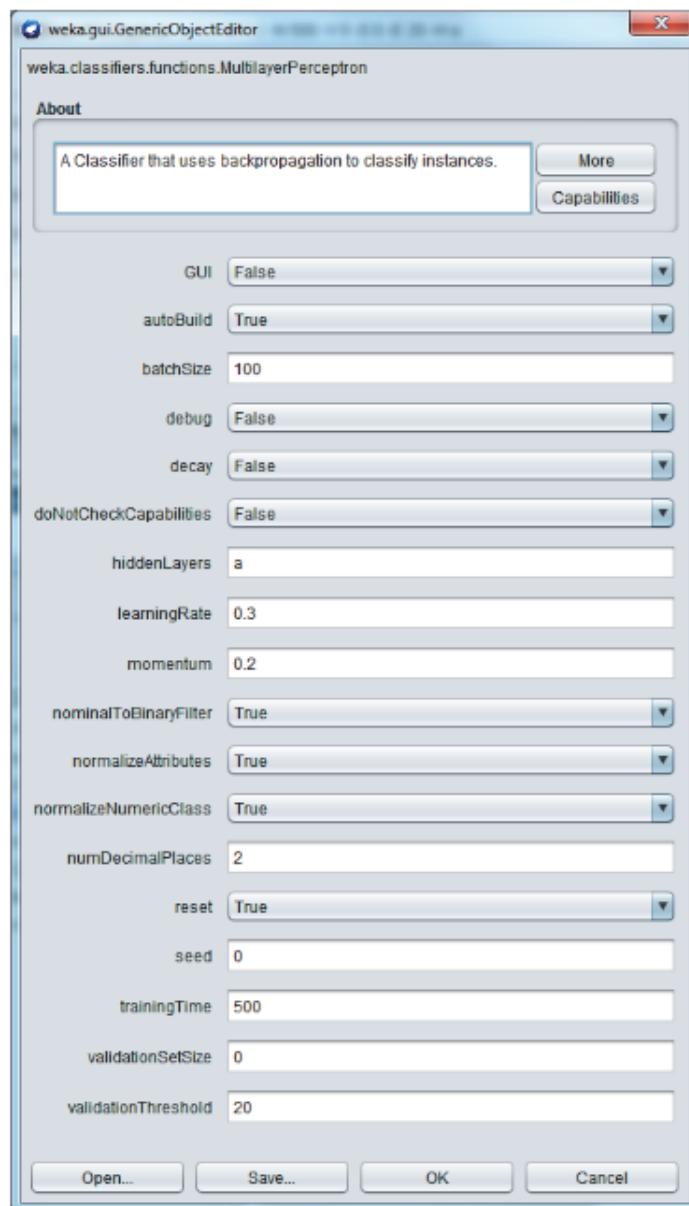
8 numarada belirtilen bölümde analizin sonuçları gösterilmektedir. Korelasyon katsayısı (*Correlation coefficient*), ortalama mutlak hata (*Mean absolute error*), hata karelerinin ortalaması (*Mean square error*) vb. hata analiz verileri sunulmaktadır.

Birden fazla kez, farklı parametreler ile aynı sınıflama algoritması uygulanabileceği gibi farklı bir algoritma da seçili verilere uygulanır. Bunların tümü 7 numara ile belirtilen bölümde listelenir. 6 numaralı açılır listede tahmin edilecek sütun başlığı seçimi yapılır. İstenirse kullanıcı tarafından tahmin edilecek veri başlığı değiştirilebilir.



Şekil 4.11. Tahmin Edilecek Veri Başlığı Seçimi

3 numaralı bölme ise verilerin eğitim ve test olarak nasıl ayrılacağının seçildiği kısımdır. 4 numaralı buton (*More options*) ile analiz sonuçlarının nasıl görüntüleneceğine yönelik seçenekler listelenir. Örneğin aşağıdaki şekil için analiz sonuçlarına tahmin değerlerinin de eklenmesi istenmiş ve gösterilmiştir.

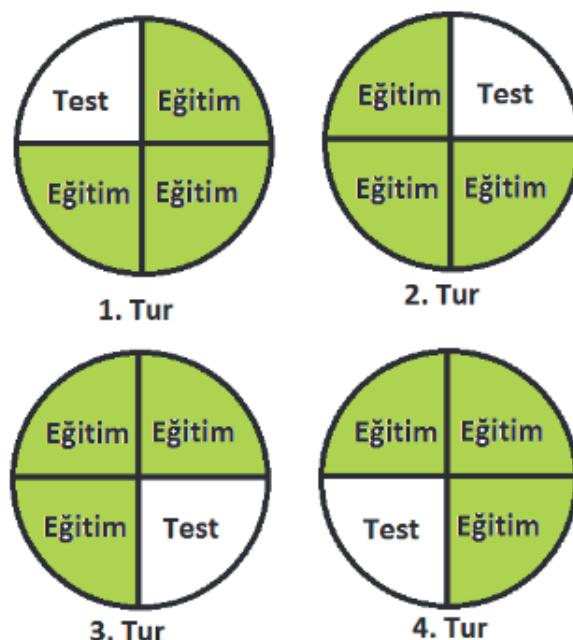


Şekil 4.12. Yapay Sinir Ağları Analizi Parametre Giriş Ekranı

Seçili analizin parametreleri 2 numaralı bölüme tıklanınca listelenecektir. Weka seçili analize en uygun parametreleri kendisi otomatik olarak seçili getirir. Kullanıcı istege bağlı olarak her bir parametreyi değiştirebilir.

4.3.3. Eğitim ve Test Verilerini Parçalama

İsteğe bağlı olmak üzere eğitim verilerini kullanma seçeneği ile (*Use training set*) bir dosyadan seçerek eğitim verileri verilebilir. Test verilerini kullanma seçeneği ile (*Supplied test set*) test verileri bir dosyadan seçilerek bağımsız olarak verilebilir. Eğitim ve test verileri istenirse yüzdesel ayırtırma seçeneği ile (*Percentage split*) seçilen yüzdesel oranda eğitim verisi geri kalan da test verisi olarak kullanılabilir. Çapraz doğrulama yöntemi (*Cross-validation folds*) seçilip yüklenen veriler istenen parça sayısına bölündüp bunlardan bir parçası test geri kalanlar eğitim verisi olarak kullanılır. Ve bu işlem tüm parçalar için yer değiştirerek uygulanır. Çapraz doğrulama ile analiz yapılması durumunda sol alt köşedeki durum çubuğu her bir parça (*fold*) için işlem yapıldığı gösterilmektedir.



Şekil 4.13. Çapraz Doğrulama Mantığı

Yukarıdaki şekele dikkatle bakılırsa çapraz doğrulama işlemi dört parça için yapıldığı durumda birinci turda bu dört eşit parçadan birinci parça test için diğer üç parça eğitim verisi için kullanılacakken ikinci turdan bu kez ikinci parça test için diğer kalan üç parça eğitim için kullanılacaktır. Bu durum dört parça için böylece devam edecektir. Böylelikle her parça yani tüm veriler hem eğitim için hem de test için kullanılmış olacaktır.

4.3.4. Çıktının İncelenmesi

Weka analiz sonucunu gösterirken ilk kısımda çalışma bilgilerini listelemektedir. Hangi metodun kullanıldığı ve hangi parametre değerlerini aldığı *Scheme* başlığında gösterilmektedir. *Relation* başlığı ile hangi dosyanın kullanıldığı bilgisi verilmektedir. *Instances* başlığı ise verilerin toplam satır sayısını ifade eder. *Attributes* ile verilerin sütun başlıklarları gösterilir. *Test mode* başlığı ise verilerin eğitim ve test kümeleri olarak nasıl ayrıldığını belirtmektedir.

```

1  === Run information ===
2  Scheme: weka.classifiers.functions.MultilayerPerceptron -L 0.5 -M 0.2 -R 500 -V 0 -E 20 -H 8
3  Relation: airline_passenger
4  Instances: 144
5  Attributes: 2
6  Attribute(s):
7  	passenger_number
8  	Date
9  Test mode: 10-fold cross-validation
10 
11 === Classifier model (full training set) ===
12 Linear Node 0
13  Input  Weights
14   Threshold  0.808603490303289
15   Node 1  -1.894653303200949
16  Kappa(s) Node 1
17  Input  Weights
18   Threshold  0.4213387408712272
19   Attr 0  -2.0545903642662147
20  Class
21  Input
22  Node 0
23 
24 Time taken to build model: 0.32 seconds
25 
26 === Predictions on test data ===
27  actual,predicted,error
28  1,289,283,618,-8,866
29  2,235,233,619,-8,869
30  3,235,235,614,-8,816[]
31  4,362,423,231,47,213
32  5,364,253,421,-46,377[]
33  6,370,137,655,-32,145
34 
35 === Cross-validation ===
36 === Summary ===
37 Correlation coefficient: 0.5981
38 Mean absolute error: 40.2798
39 Root mean squared error: 53.6592
40 Relative absolute error: 39.8477 %
41 Root relative squared error: 44.7105 %
42 Total Number of Instances: 144
43
44

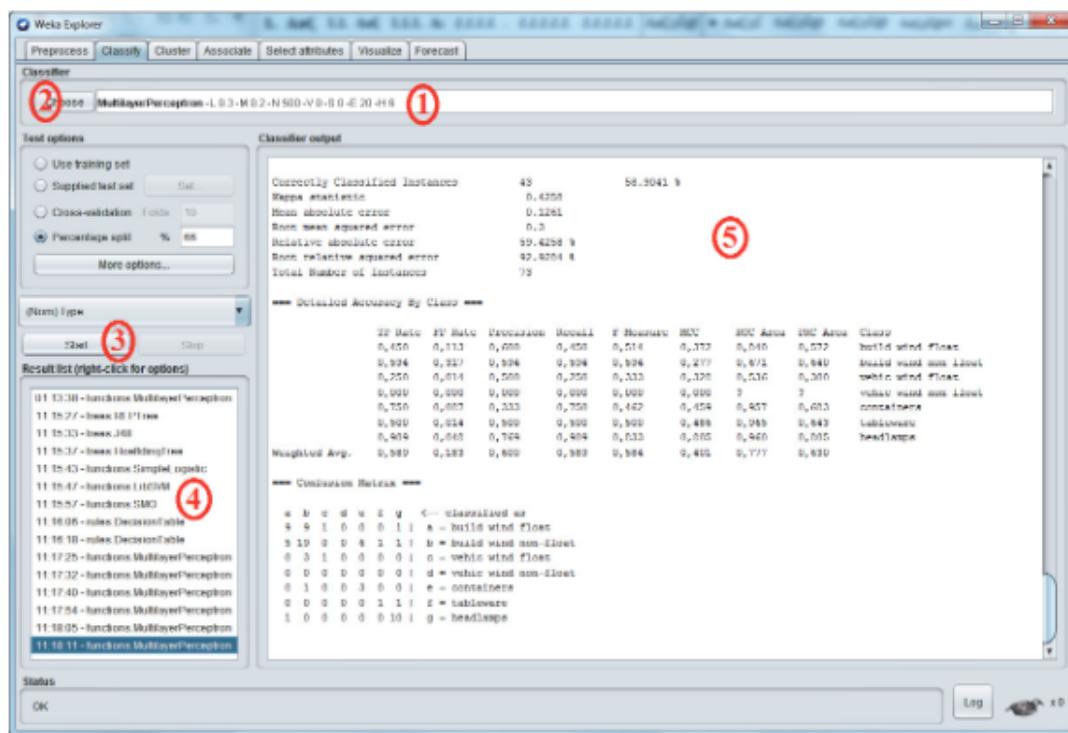
```

Şekil 4.14. Weka Sayısal Veri Analiz Sonuç Verileri Örneği

İkinci kısımda (*Classifier model*) ise algoritmanın model bilgisi verilmektedir. Örneğin burada Yapay Sinir Ağları (*MultilayerPerceptron*) algoritması kullanılmış. Bu sebeple seçili algoritmanın sınırlarındaki ağırlık değerleri verilmiştir. Ayrıca bu öğrenme işleminin ne kadar zaman aldığı (*Time taken to build model*) saniye cinsinden verilmektedir. Aşağıdaki şekilde gösterilen sonuç ekranı için tahmin değerlerinin de gösterilmesi seçeneği aktif edilmiştir. Bu sebeple test verileri için üretilen tahmin sonuçları (*Predictions on test data*) da listelenmiştir. Burada test verisinin sıra numarası, gerçek değeri, tahmin edilen değeri ve gerçek değer ile tahmin değeri arasındaki hata payı pozitif veya negatif sayısal olarak verilmektedir.

```

204
205
206
207
208
209
210
211
212
213
214
215
216
217 Time taken to build model: 0.34 seconds
218
219 === Stratified cross-validation ===
220
221
222 Correctly Classified Instances      145           67.757 %
223
224 Error statistics
225 Mean absolute error               0.1114
226 Root mean squared error           0.1627
227 Relative absolute error          0.5515 %
228 Root relative squared error     0.5588 %
229 Total Number of Instances        214
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
```

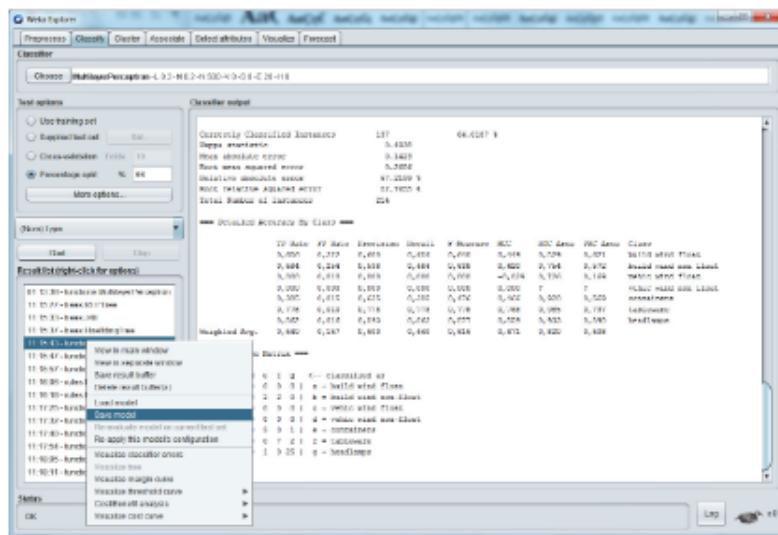


Şekil 4.16. Weka ile Tekrar Tekrar Farklı Analizler Yapma

Aynı analizin farklı parametreler ile çalıştırılması mümkün olduğu gibi seçili olan aktif verilere farklı bir analizin de uygulanması mümkündür. Bu-nun için tek yapılması gereken 2 numara ile belirtilen *Seç* adlı (*Choose*) butondan ilgili analizin seçilmesi 3 numaralı başlat (*Start*) butonuna bas-maktır. Yapılan tüm analizler 4 numara ile belirtilen bölümde listeleneciktir. Her bir analize tıklayarak seçili analizin sonuçları 5 numara ile belirtilen bölümden görüntülenecektir. Her bir analizin değerlendirmesi yapılp karşı-laştırmalar yapılabilir.

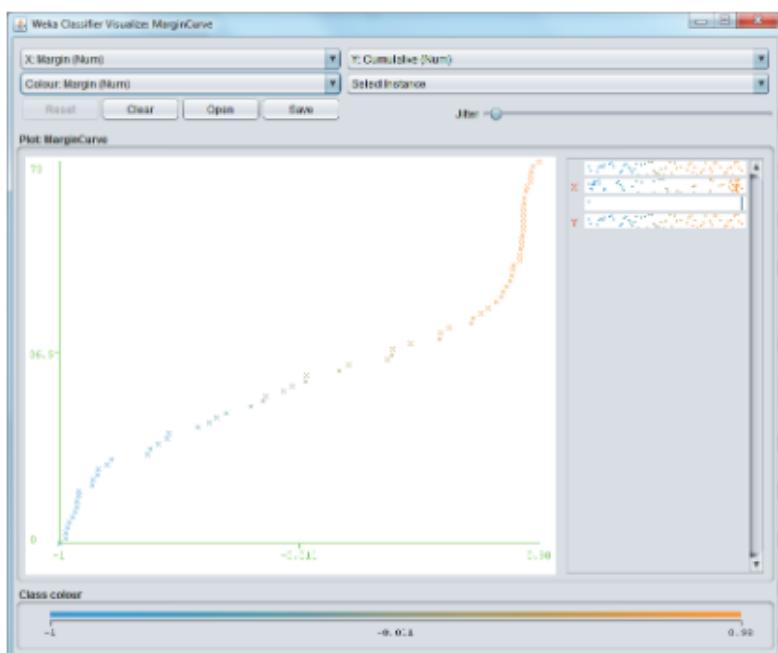
4.3.6. Modeller ile Çalışma

Weka ile yapılan analizlerin modeli Java nesne dosya biçiminde istege bağlı olarak kaydedilebilir. Daha önce kaydedilmiş model sonradan seçile-rek yeni verilere uygulanıp sonuçlar o model temel alınarak üretilib-ilir. Özellikle eski model üzerinden yeni test verilerine uygulayıp tahminler elde etmek için kullanılan bir seçenektedir.

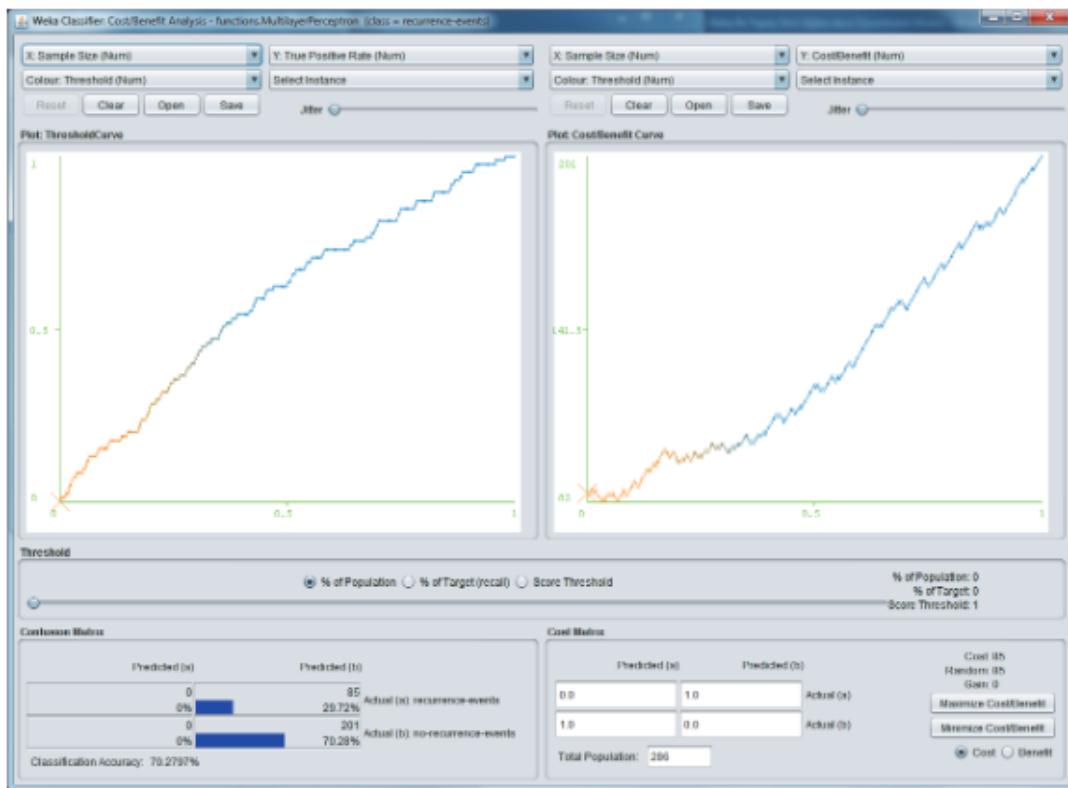


Şekil 4.17. Weka Üretilen Modeli Kaydetme

Yukarıdaki şekilde gösterildiği gibi yapılan analizlerden herhangi birinin üzerine sağ tıklandığında açılan menü sonuçları çeşitli şekillerde görselleştirme imkânı tanır. Sınıflandırma hatalarını görselleştirme (*Visualize classifier errors*), modelin ağaç yapısı olması durumunda ağaç yapısını oluşturma (*Visualize tree*), tolerans eğrisini çizdirme (*Visuallize margin curve*), eşik eğrisini çizdirme (*Visualize threshold curve*), Fayda/maliyet analizi (*Cost/Benefit analysis*), maliyet analizi (*Visualize cost curve*) olanakları sunar.

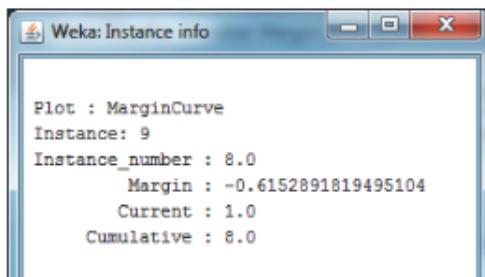


Şekil 4.18. Weka Sonuç Görselleştirme Ekranı

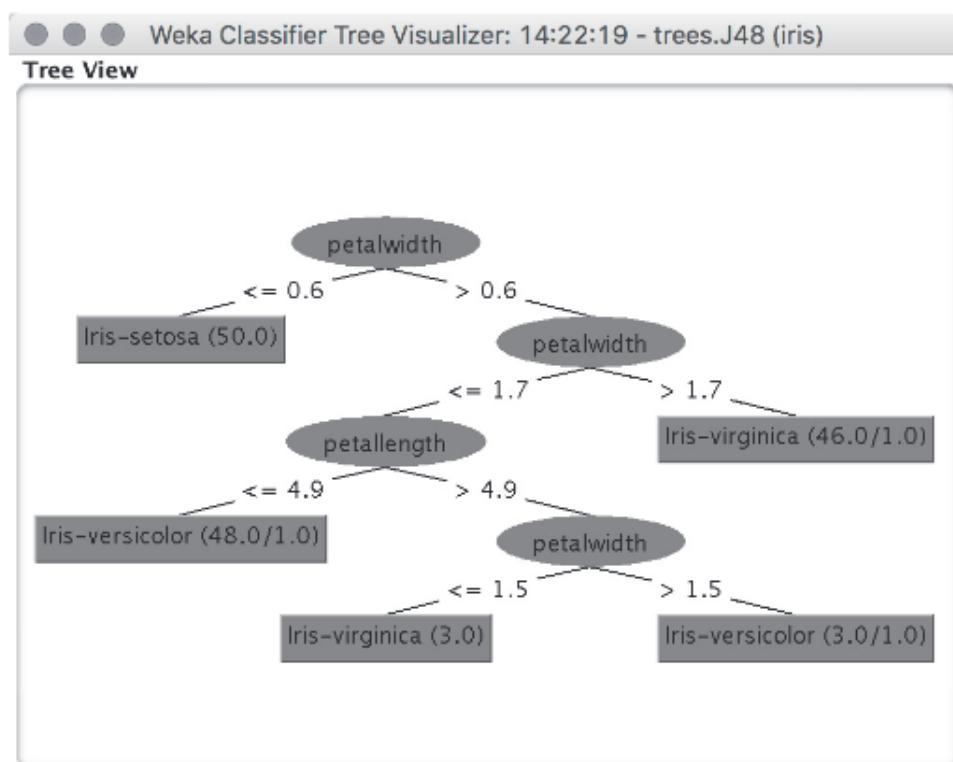


Şekil 4.19. Weka Fayda/Maliyet Analizi Görselleştirme Ekranı

Buradaki veri noktaları sınıflarına göre renk alır. Mavi, kırmızı, yeşil renkleri sırasıyla kullanılır ve ekranın altında bunu gösteren bir anahtar yer alır. Yanlış sınıflandırmalar kutu olarak simgelenir iken doğru sınıflandırılmış örnekler artı işaretü ile simgelenir. Herhangi bir noktanın üzerinde sağ tıklayarak, ilgili örneğin numarası, niteliklerin değerleri ve kendi sınıfı ile tahmin edilen sınıf bilgisi görüntülenecektir.



Şekil 4.20. Veri Görselleştirme Ekranında Sağ Tıklama



Şekil 4.21. Weka Ağaç Yapısı Görselleştirme Ekranı

Analiz sonuçları normal olarak sağ taraftaki ekranda gösterilir. Bunun için ana ekranın göster (View in main window) seçeneği kullanılır. Farklı analiz sonuçlarını daha rahat görüp karşılaştırmak için ayrı yeni bir pencere de göster (View in separate window) seçeneği de kullanılabilir. Aktif analizin görünen sonuçları istenirse ekrandaki sonuçları kaydet (Save result buffer) seçeneği ile bir dosyaya kaydedilebilir veya aktif analizi ekrandan silmek için ekrandaki sonuçları sil seçeneği (Delete result buffer) kullanılabilir.

Aktif analiz daha sonra tekrar kullanılmak üzere kaydedilmek istenirse modeli kaydet (Save model) seçeneği ile yol verilir ve Java nesnesi olarak kaydedilir. Daha önce kaydedilmiş dosya model yükle (Load model) seçeneği ile sisteme yüklenir ve yeni verilere uygulanabilir. Seçili olan aktif veri dosyası değiştirildikten sonra bu modelin özelliklerini tekrar uygula (Re-apply this model's configuration) seçeneği yeni verilere aktif analizin ayarlanmış tüm parametreleri aynı şekilde uygulanır. Aktif seçili model istenirse yalnızca belirlenen test verilerine de uygulanabilir. Bunun için aktif test verilerine bu modeli uygula (Re-evaluate model on current test set) seçeneği kullanılır.

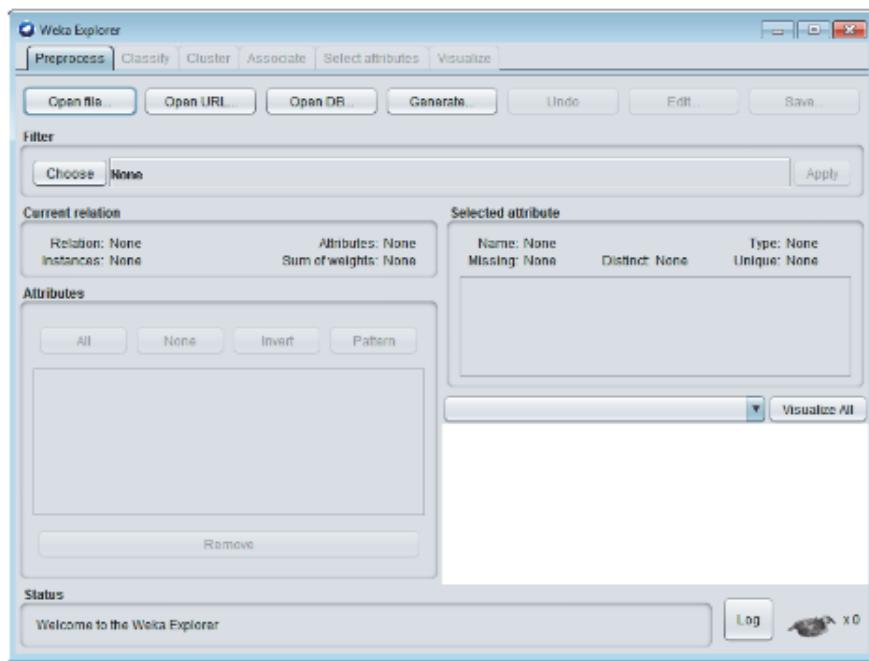
4.3.7. Hata Oluştuğunda

Sonuç geçmişi listesinin altında, basitçe OK yazan bir durum satırı vardır. Bazen bu durum, hata günlüğüne bakılması gerektiğini ve bir şeyin yanlış gittiğini ifade eder. Örneğin, bir analiz seçim panelinde yapılabilecek analizler listelenir ve uygun olmayan analizler pasif olarak görüntüleyerek onların seçilmesi izin vermez. Ancak bazen verilerin karmaşık yapısı nedeniyle uyumsuz bir analiz seçilerek işlem başlatılabilir. Bu durumda, WEKA uyumsuzluğu (genellikle Başlat'a basıldığında) saptadığında durum satırı değişir. Hatayı görmek için, arabirimin sağ alt köşesindeki kuşun solundaki Günlük düğmesini tıklanır. WEKA ayrıca, kullanıcının ev dizininin wekafiles dizininin altında weka.log adlı bir dosyaya ayrıntılı bir günlük yazar. Bu genellikle, Explorer'in Günlük penceresinden sorunların nedenleri hakkında daha fazla bilgi içerir çünkü standart çıkış ve hata kanallarına yönelik hata ayıklama çıktısını yakalar.

4.4. Explorer Penceresi

Weka'da kullanılan ana ekrandır. Weka'nın tüm olanaklarına menü seçenekleri ve ekrandaki ilgili yerleri doldurarak erişim sağlanır. Varsayılan olarak en çok kullanılan altı sekme ekranda gelecektir. Eğer paketlerden herhangi bir tanesi yüklenmiş ise onun ile ilgili bir sekmenin eklendiği görülecektir. Veriler yüklenmeden veriler üzerinde işlem yapacak butonlar ve komutlar pasif durumda görüntülenmektedir.

Weka'yı kullanmanın en kolay yolu, Explorer adı verilen grafiksel kullanıcı arayüzüdür. Bu arayüz üzerinde fare üzerinden geçtikçe çıkan ipuçları ile her bir komut veya işlemin ne yaptığı konusunda fikir sahibi olunur. Explorer ekranının temel dezavantajı her şeyi ana bellekte tutmasıdır. Bir veri kümesi açıldığında onu tamamen belleğe yükler. Bu durum çoğunlukla küçük ve orta ölçekli problemlere uygulanabileceği anlamına gelmektedir. Fakat elbetteki, Weka çok büyük veri kümelerini işlemek için kullanabilecek bazı algoritmalar içermektedir.



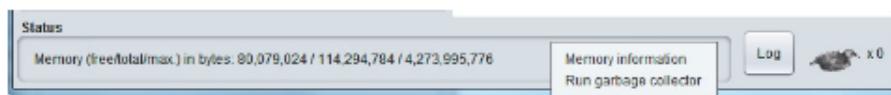
Şekil 4.22. Weka Explorer Penceresi

Bu penceredeki sekmelerin ne iş yaptıklarını aşağıdaki gibi kısaca özetleyebiliriz.

- **Preprocess:** Veri kümesini seçilmesi ve çeşitli şekillerde üzerinde değişiklikler yaparak verinin ön işlemesine imkân tanıyan sekmedir.
- **Classify:** Sınıflandırma veya regresyon analizleri ile eğitim ve test işlemlerinin yapıldığı sekmedir.
- **Cluster:** Veri kümesi üzerinden küme analizi yapıldığı sekmedir.
- **Associate:** Veriler arasındaki ilişki kuralları oluşturmak için eğitim ve test işlemlerinin yapıldığı sekmedir.
- **Select Attributes:** Veri kümesinin en çok ilişkili yönlerini seçildiği sekmedir.
- **Visualize:** Verilerin farklı iki boyutlu grafiklerini görüntülemek ve onları etkileşimli olarak incelendiği sekmedir.

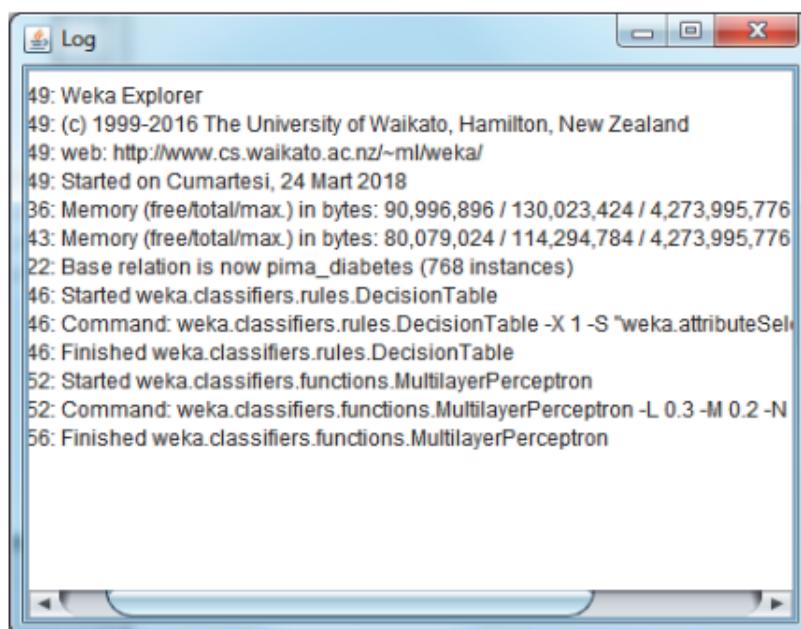
Her sekmenin altında bir Durum kutusu ve bir Günlük düğmesi bulunur. Durum kutusunda, neler olduğu konusunda bilgi veren mesajlar görüntülenir. Örneğin, Explorer'a bir dosya yükliyse, durum kutusu dosya yüklü olduğunu söyleyecektir. Bu kutunun içinde herhangi bir yere sağ tıklamak, iki seçenekli küçük bir menü açar: WEKA'nın kullanabileceği bellek miktarını

görüntüler ve Java çöp toplayıcısını çalıştırır. Çöp toplayıcının sürekli bir arka plan görevi olarak çalıştığını unutulmamalıdır.



Şekil 4.23. Weka Durum Satırı

Günlük düğmesine (*Log*) tıklamak, WEKA'nın bu oturumda gerçekleştiği eylemlerin metin zamanlarını, zaman damgalarıyla açar.



Şekil 4.24. Weka Günlük Dosyası

Kuşun yanındaki sayı ise, kaç eşzamanlı işlemin çalıştığını gösterir. Kuş ayakta duruyor, ancak hareket etmeyi bırakıyorsa, bu onun hasta olduğunu ve bir şeylerin ters gittiğini ifade etmektedir. Bu durumda Explorer penceresinin yeniden başlatılması gerekebilir.

4.4.1. ARFF Uzantılı Dosyalar Oluşturma

Weka programına verileri yüklerken ARFF dosyaları dışında CSV uzantılı dosyalar da kullanılabilir. CSV uzantılı dosyalar verilerin virgül ile ayrıldığı satır ve sütunlardan oluşan dosya türleridir. Microsoft'un yaygın bir şekilde kullanılan Excel programı CSV uzantılı dosya kaydetmek için gayet ideal bir yöntemdir. Böylece veritabanı benzeri sistemlerden elde edilen

veriler Excel ortamına aktarıldıkten sonra kolaylıkla CSV uzantılı dosya elde edilebilecektir. Fakat burada dikkat edilmelidir ki CSV uzantılı dosyalar Weka programına yüklenliğinde tüm veriler nominal kabul edilmekte olup sayısal değer olarak görülmemektedir. Bu sebeple ARFF dosya türleri ile çalışmak her zaman daha sağlıklıdır.

CSV -> ARFF	
1	prep,pies,pco,skin,insu,masu,pedi,age,class
2	6,115,72,35,0,33,6,0,627,50,tested_positive
3	1,109,60,29,0,26,6,0,30,1,0,tested_negative
4	6,115,64,0,0,33,3,0,672,50,tested_positive
5	1,89,66,0,0,94,28,1,0,167,21,tested_negative
6	0,197,40,35,148,0,0,288,53,tested_positive
7	5,116,74,0,0,35,6,0,201,50,tested_negative
8	9,78,50,32,88,31,0,249,26,tested_positive
9	10,115,0,0,0,35,5,0,134,29,tested_negative
10	2,197,70,45,545,30,5,0,158,53,tested_positive
11	8,125,96,0,0,0,0,232,54,tested_positive
12	4,110,92,0,0,37,6,0,151,30,tested_negative
13	10,168,75,0,0,38,0,857,94,tested_positive
14	10,138,80,0,0,27,1,1,441,57,tested_negative
15	1,189,60,28,646,30,1,0,390,59,tested_positive
16	5,166,72,19,178,23,8,0,587,51,tested_positive
17	7,100,0,0,0,30,0,484,32,tested_positive
18	0,118,84,47,230,45,8,0,551,91,tested_positive
19	7,107,74,0,0,29,6,0,254,91,tested_positive
20	1,105,30,38,68,49,5,0,189,38,tested_negative
21	1,115,70,30,56,94,6,0,529,32,tested_positive
22	3,128,88,41,235,39,5,0,707,27,tested_negative

1	disealition,pina_diabetes
2	attribute 'prep' numeric
3	attribute 'pco' numeric
4	attribute 'skin' numeric
5	attribute 'insu' numeric
6	attribute 'masu' numeric
7	attribute 'pedi' numeric
8	attribute 'age' numeric
9	attribute 'class' { tested_negative, tested_positive }
10	data
11	6,198,72,35,0,33,6,0,627,50,tested_positive
12	1,88,66,23,0,26,6,0,311,51,tested_negative
13	6,183,64,0,0,28,3,0,672,52,tested_positive
14	1,89,66,23,94,28,1,0,167,21,tested_negative
15	0,197,40,35,148,0,0,288,53,tested_positive
16	9,116,74,0,0,35,5,0,134,29,tested_negative
17	10,115,0,0,0,35,5,0,134,29,tested_negative
18	3,78,50,32,88,31,0,249,26,tested_positive
19	10,168,75,0,0,38,0,857,94,tested_positive
20	2,197,70,45,545,30,5,0,158,53,tested_positive
21	8,125,96,0,0,0,0,232,54,tested_positive
22	4,110,92,0,0,35,5,0,134,29,tested_negative

Şekil 4.25. Örnek Bir CSV ve Dönüştürülmüş ARFF Dosyası

CSV dosyalarını ARFF dosyasına dönüştürmek hem kolay hem de Weka içinde daha sorunsuz analiz yapılmasına imkân tanıyacaktır. Bunun için tek yapılması gereken CSV dosyasını herhangi bir metin editörü ile açıp öncelikle @relation ile başlayıp dosya adı belirtilmelidir. Sonrasında alt satırda geçip @attribute ile başlayıp sütun başlığı ve veri türü belirtilmelidir. En son olarak @data ifadesi verilerin başına yazılmalıdır.

Burada belirtilenlerin dışında aşağıdaki dosya formatları da Weka tarafından kabul edilmektedir. Fakat unutulmamalıdır ki ARFF uzantılı dosyalar dışındaki dosyalarda veri türlerinin belirtilmemiş olması sorunlarınmasına sebep olabilir. Tabi ki burada belirtilen dosyaların kabul edilmesi için Weka sonuç olarak bu dosya türlerini yine kendisinin kabul edeceği ARFF dosya tiplerine dönüştürmektedir.

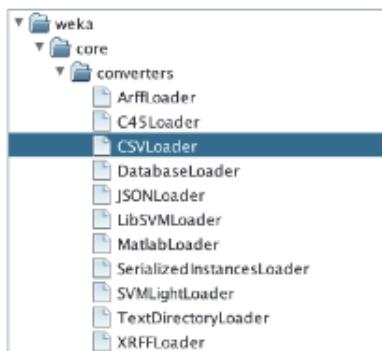
- .names ve .data uzantılı C4.5'in doğal dosya formatı,
- .bsi uzantılı olan dizi haline dönüştürülmüş veriler,
- .libsvm uzantılı olan LIBSVM dosya formatı
- .dat uzantılı olan SVM-Light dosya formatı
- .xrff uzantılı olan XML tabanlı ARFF dosya formatı
- .json uzantılı olan JSON tabanlı ARFF dosya formatı
- .m uzantılı olan ASCII Matlab dosya formatı

Weka'da dosya yüklenirken ARFF uzantılı dosyalar yüklenmelidir. Fakat eğer yükleme sırasında hatalı bir dosya seçimi yapılrsa otomatik olarak dosya dönüştürme ekranı olan aşağıdaki şekil açılır. Burada varsayılan olarak CSV dosya yükleme seçeneği gelir.



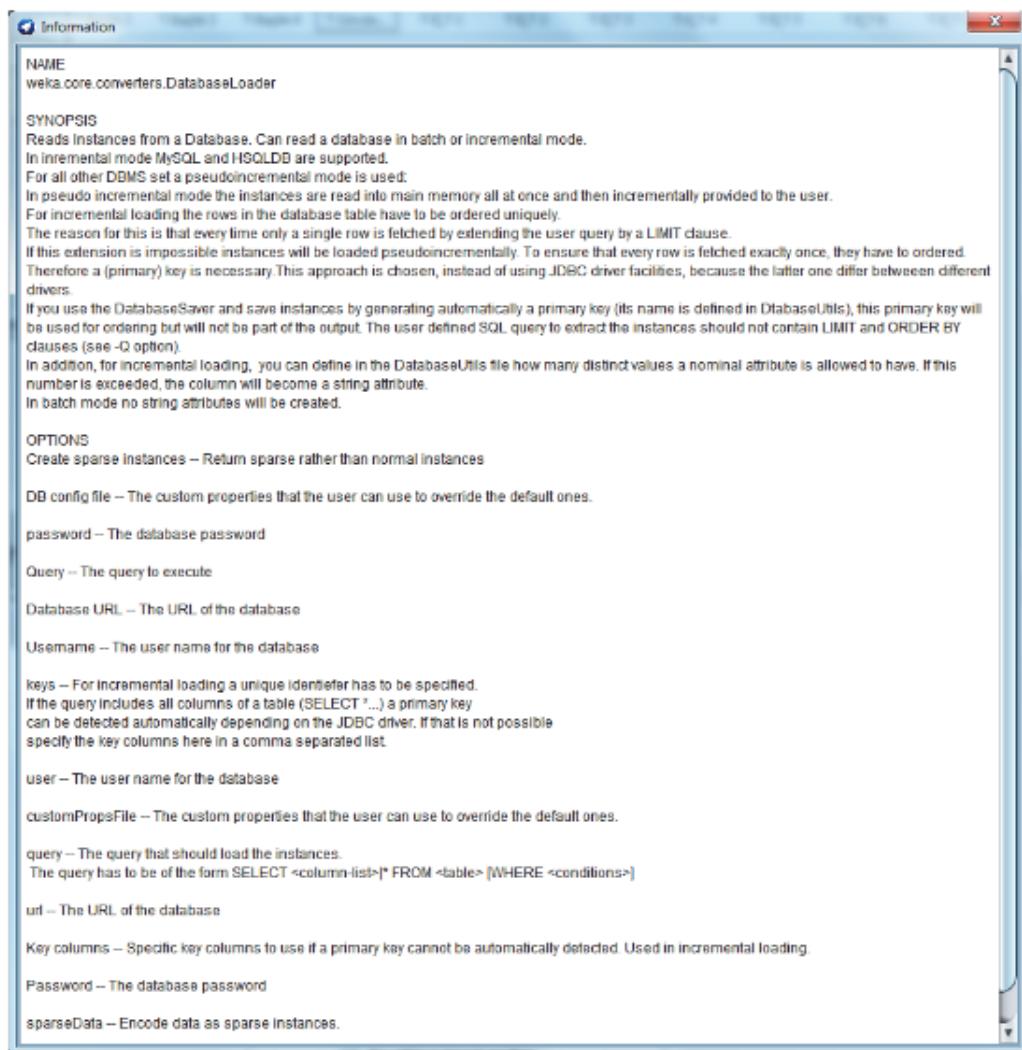
Şekil 4.26. CSV Dosya Dönüşümü Ekranı

Elbetteki yukarıda saydığımız dosya tiplerinden de veri yüklenmesi desteklenmektedir. Bunun için sol üst köşede yer alan Seç (*Choose*) butonu ile aşağıdaki şekilde de görüldüğü gibi diğer dosya türleri seçimi görüntülenir.



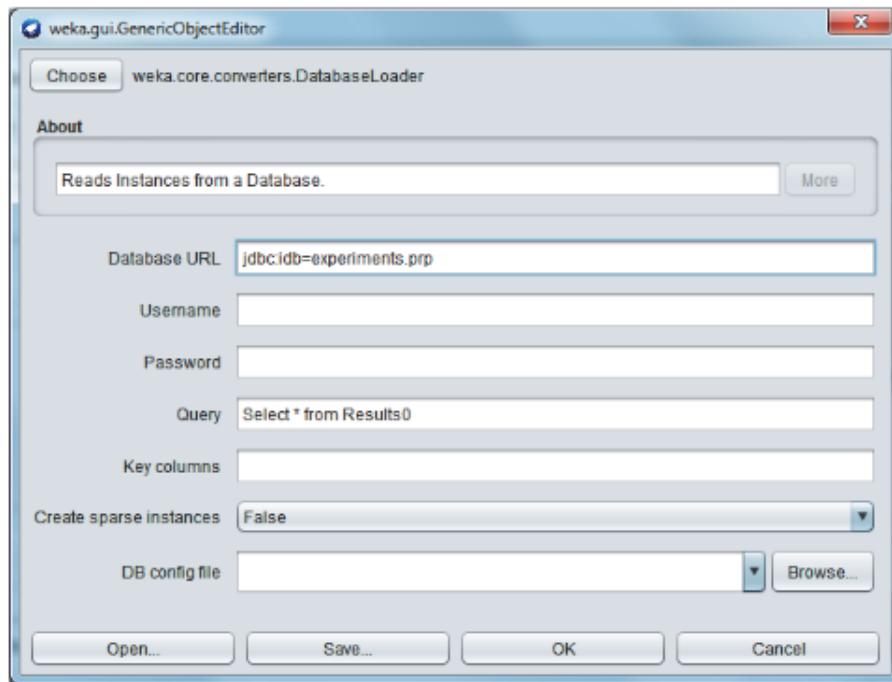
Şekil 4.27. Farklı Dosya Türlerinden Veri Yükleme Seçenekleri

Seçili olan dosya tipinden verileri yüklerken nelerin dikkate alınacağı ve başka soru işaretlerine cevap bulabilmek için Daha Fazlası (*More*) butonu tıklanır. Aşağıdaki gibi açılan pencerede örneğin veritabanından veriler yüklenirken nelere dikkat edileceğin hususunda bilgiler verilir.



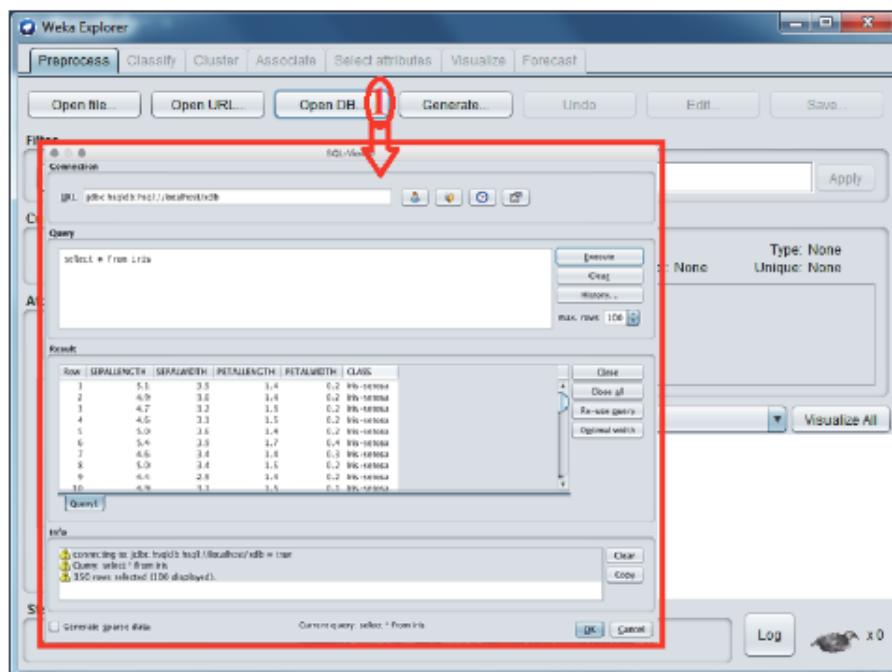
Şekil 4.28. Farklı Dosya Tipinde Veri Yüklerken Açılan Bilgilendirme Ekranı

Yukarıda da belirtildiği üzere varsayılan olarak CSV dosya tipinde veri yükleme görüntülenmektedir. Farklı veri tipi seçildiğinde parametreleri de ona göre değişecektir. Örneğin aşağıdaki şekilde veritabanından veri yüklerken hangi bilgilere ihtiyaç duyulacağı görüntülenmektedir.



Şekil 4.29. Veritabanında Veri Yüklerken Görüntülenen Bilgi Giriş Ekranı

Fakat veritabanından veriler yüklenirken Önişleme sekmesinde yer alan Dosya Açı (*Open File*) seçeneği yerine Veritabanı Açı (*Open DB*) seçeneği ile açılan SQLViewer ekranı daha kullanıcı dostu bir ekrandır.



Şekil 4.30. Veritabanından Veri Yüklemek İçin SQLViewer Ekranı

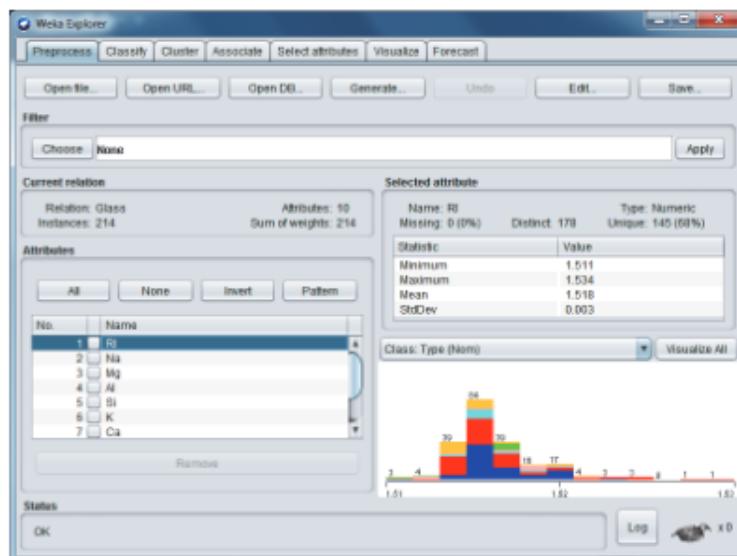
Örneğin bir başka dosya türü seçimi olarak C4.5Loader seçilebilir. Bu tür seçildiğinde bir veri kümesi için iki farklı dosya seçilir. Bu dosyalardan biri veriler için diğerini alan adları içindir. Fakat tüm bu dosya tiplerine nazaran bilhassa büyük bir veri kümesinden tekrar tekrar yükleme yapmak için ARFF dosyası yüklemek daha hızlıdır.

Metin klasörü yükleme (*TextDirectoryLoader*) seçeneği ile metin makdenciliğinde kullanılmak için TXT uzantılı düz metin dosyalarını içeren bir klasörü yüklemek için kullanılır. Her metin dosyası veri kümesinde ayrı bir örnek olur. Dosyalar sınıf niteliğine göre alt dizinlere ayrılır. Sonrasında StringToWordVector analiz seçeneği ile sözcük frekanslarına dönüştürülür.

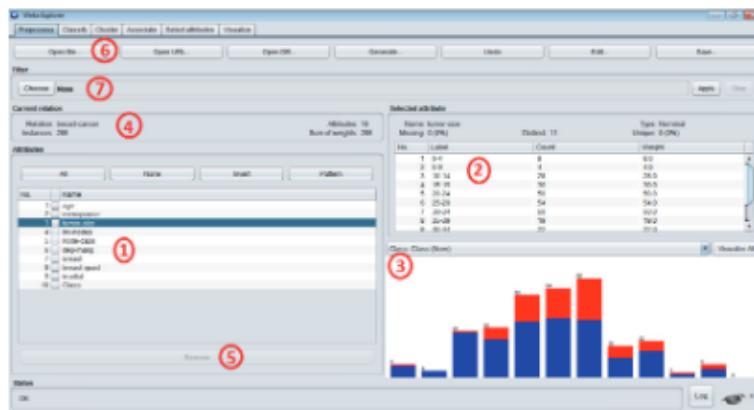
Şekil 4.13 ve Şekil 4.16 gösterilen dosya açma ekranında her bir seçim türüne göre parametrik değerlerin girilmesi istenmektedir. Eğer dosya tekrar tekrar açılmak istenirse bu parametre değerlerini sürekli girmek makul bir seçenek olmayacağındır. Bu sebeple en alta yer alan Kaydet (*Save*) butonu ile bu ayarlar Weka dosyası olarak kaydedilir. Daha sonradan kaydedilmiş ayarlar tekrar yüklemek için Açı (*Open*) butonu ile kayıtlı dosya seçilir.

4.4.2. Veri Ön İşleme

Weka veri kümelerine kolayca uygulanabilecek öğrenme algoritmaları sağlar. Bir veri kümесini önceden işleyebilir, bir öğrenme şemasına/algoritmasına gönderebilir ve sonuçta ortaya çıkan performansı analiz edebilir. Tüm bunlar için herhangi bir kod yazımıza ihtiyaç duymaz.



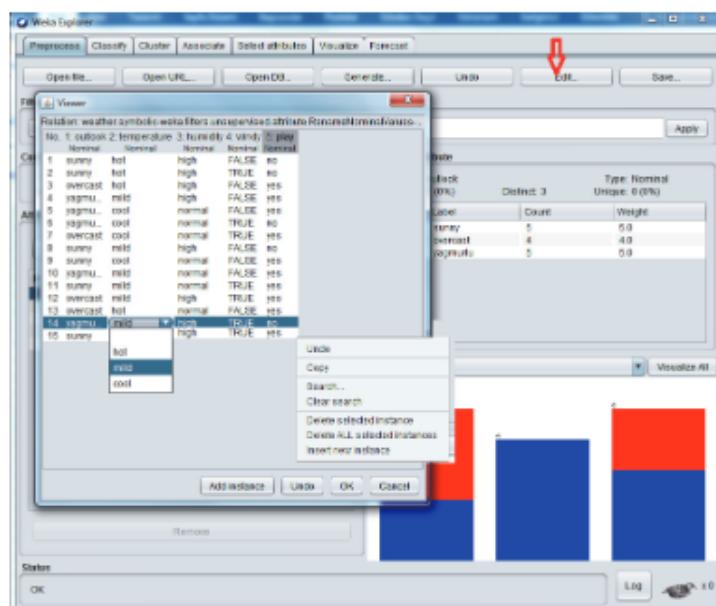
Şekil 4.31. Sayısal Bir Veri İçin Önisleme Ekranı



Şekil 4.32. Nominal Bir Veri İçin Önisleme Ekranı

Yukarıdaki şekilde görünen ve seçili olan veri nominaldir. Sınıflamalı veriler histogram grafiği olarak 3 numaralı yerde ve bu histogram grafiğinin sayısal analizi ise 2 numaralı yerde gösterilir. Sayısal veriler için ise minimum ve maksimum değerlerini, ortalama ve standart sapmasını 2 numaralı ekranda 3 numaralı ekran ise bu değerlerin grafik hali gösterilir.

Veriye ait herhangi bir nitelik 1 numaralı kısımdan seçiliip silinebilir. Seçim için tümünü seçme, seçili olan ve olmayanları ters çevirme veya Pattern (Desen) butonu ile kullanıcının belirdiği ifade ile eşleşen nitelikler seçilebilir. 6 numaralı kısımdan ise Edit (Düzenle) butonu ile veriler incelenebilir, örnekler veya nitelikler silinerek düzenlenebilir. Bunun için değerle-re veya sütun başlıklarına sağ tıklamak yeterlidir.

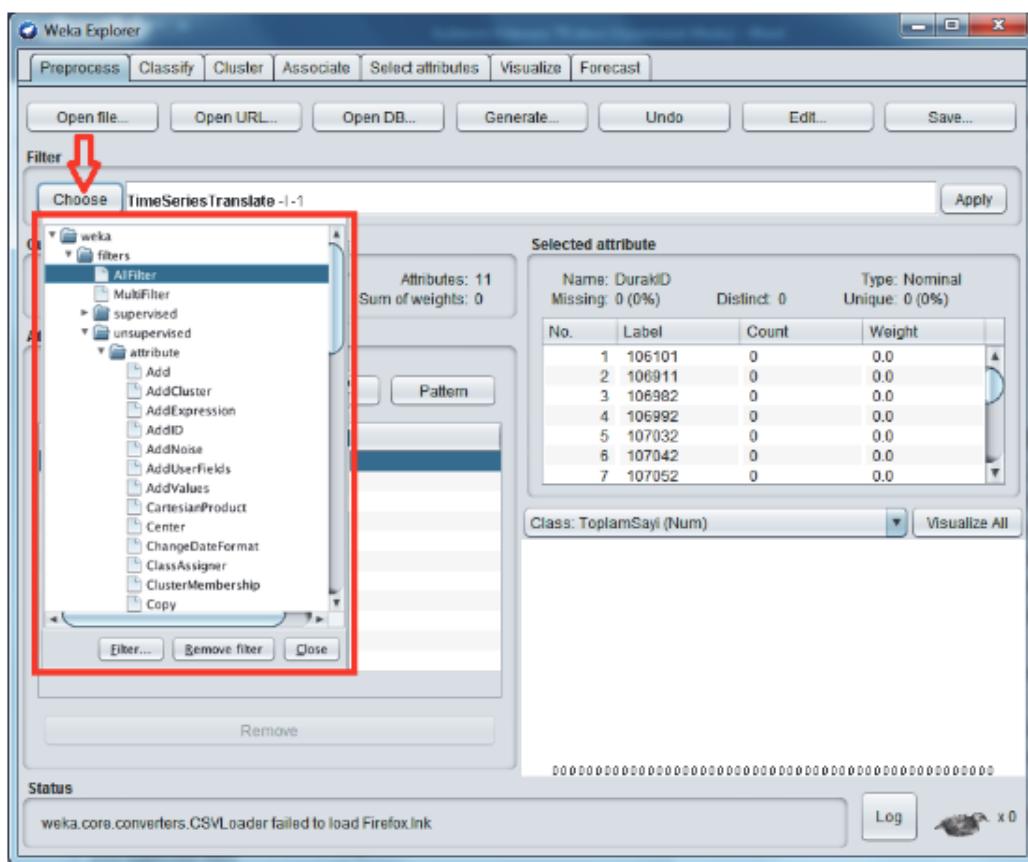


Şekil 4.33. Verileri Düzenleme Ekranı

Bu verileri düzenleme ekranı ile istenirse Yeni Örnek Ekle (*Add instance*) seçeneği ile en son satırda bir adet daha kopyalanır ve o satırda istenilen değişiklikler yapılabilir. Seçili olan bir veya birden fazla satırdaki veri silinebilir.

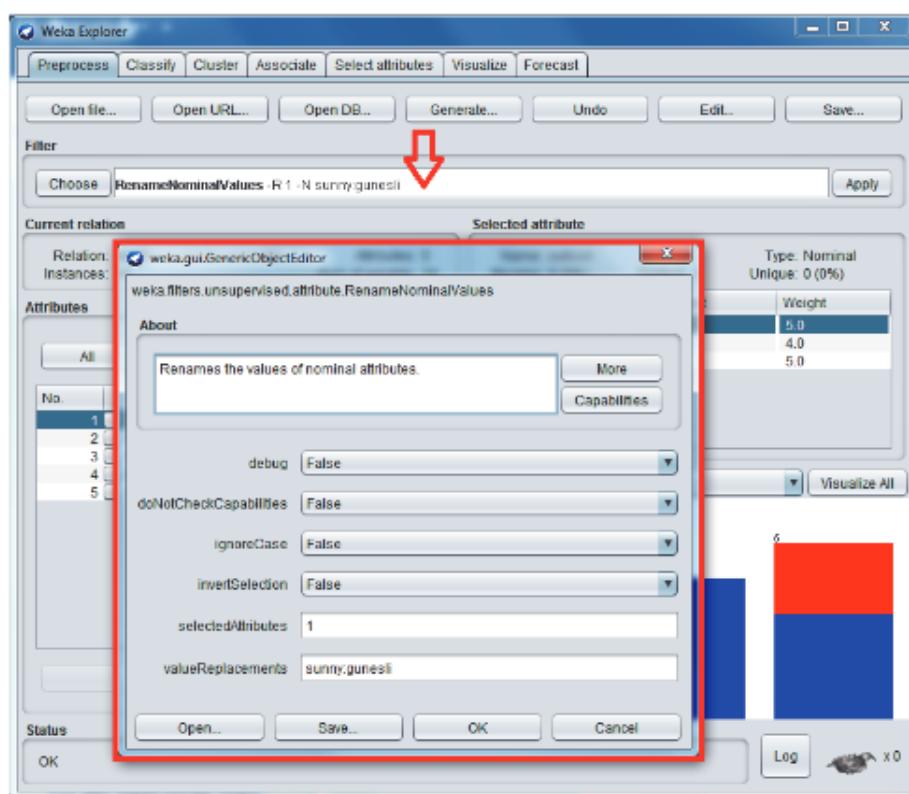
4.4.3. Filtreleri Kullanma

Explorer penceresindeki önişleme (*Preprocess*) sekmesinde dosya seçildikten sonra Filtre panelindeki Seç (*Choose*) butonu ile filtre listesi görüntülenir. Filtreler gruplanmış şekilde verildiği için yandaki ok işaretine tıklayarak ya da çift tıklama ile alt listelere ulaşılabilir.



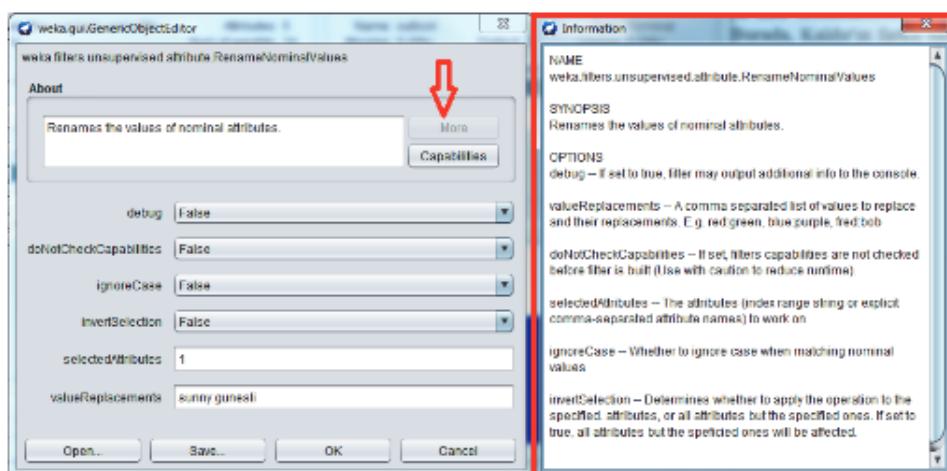
Şekil 4.34. Filtre Penceresi

Burada örnek olarak unsupervised -> attribute -> RenameNominalValues adlı filtre weather.numeric.arff adlı dosyaya uygulanabilir. Bunun için öncelikle dosya yüklemesi yapılır ve ardından yukarıdaki şekilde gösterildiği gibi filtre adı seçilir.



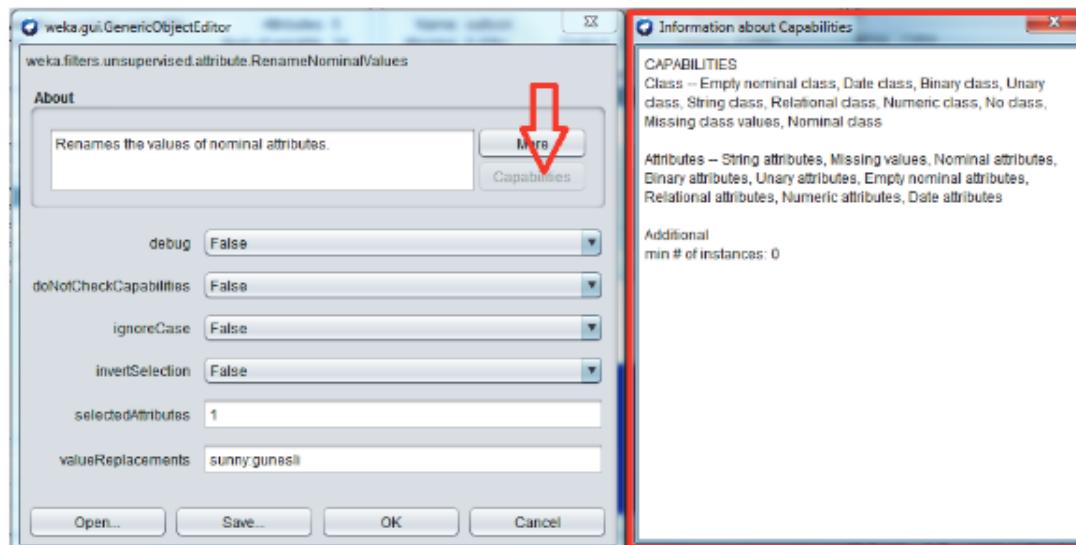
Şekil 4.35. Filtre Parametrelerinin Giriş Ekranı

Seçili filtrenin kendisine özel parametrelerinin girişi yapılmak istenirse yukarıdaki şekilde de ok işaretü ile gösterildiği gibi滤re adının yazılı olduğu satırı tıklanır. Açılan bu pencere her filtrenin kendisine özel parametreler içerir. Bu parametrelerin ne anlama geldiği ve nasıl kullanılacağı ile ilgili bilgiler Dahası (*More*) adlı butona tıklanınca görüntülenir.



Şekil 4.36. Seçili Filtrenin Kullanımı Hakkında Bilgi

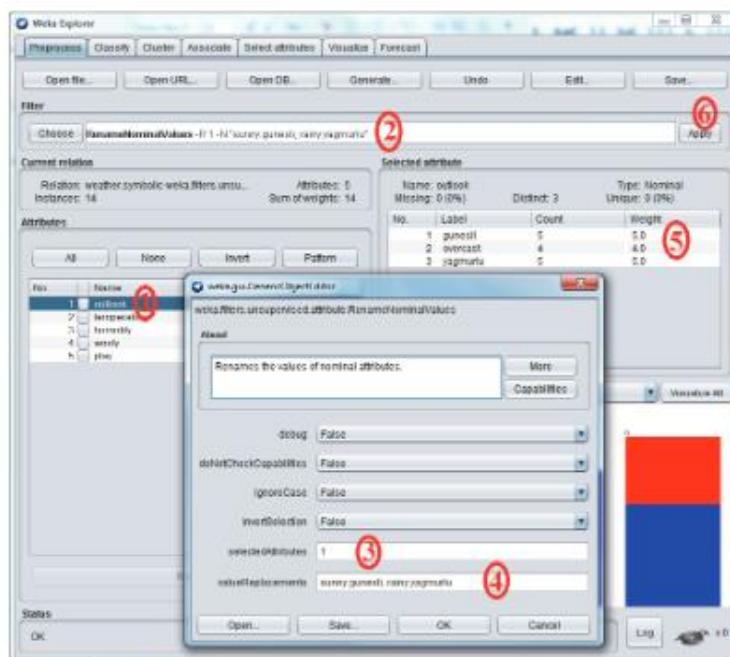
Seçili fitrenin parametre giriş ekranında bu filtrenin yetenekleri hakkında bilgi veren ekrana ulaşmak için ise Yetenekler (*Capabilities*) butonu tıklanınca aşağıdaki gibi bir ekran açılır. Bu filtrenin farklı türler ve eksik değerler gibi birçok niteliği ele alabileceği konusunda bilgi verir.



Şekil 4.37. Seçili Filrenin Yetenekleri Hakkında Bilgi Ekranı

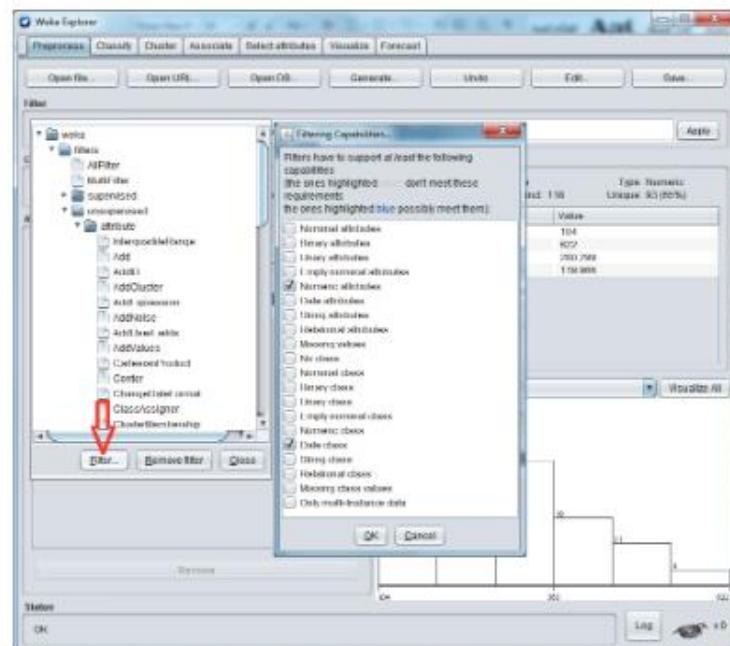
Buradaki parametre girdi değerlerinin bir dosyaya kaydedilip daha sonra bu ekranın hızlıca kullanılması istenirse Kaydet (*Save*) butonu ile Weka dosyası olarak kaydedilir. Sonradan aynı parametre girdileri ile kullanılmak istendiğinde Açı (*Open*) butonu ile ilgili dosya seçiliip yüklenir ve önceki değerlerin geldiği görünecektir.

Seçili dosyanın 1 indisli *outlook* sütununda yer alan *sunny* metnini *gunesli* olarak *rainy* metnini ise *yagmurlu* olarak değiştirilmek istenirse; *RenameNominalValues* adlı filtre seçildikten sonra 2 numara ile belirtilen satırı tıklayarak parametre giriş ekranı açılır. Parametrelerin nasıl kullanılacağı konusunda bilgi edindikten sonra seçili nitelik (*selectedAttributes*) girişine 1 indis yazılırak *outlook* sütunu ifade edilir. Değer değişikliklerine (*valueReplacements*) ise önce eski değer ve sonrasında iki nokta işaretinden sonra yeni değer yazılır. Aralarına virgül işaretleri eklenerek bu durum tekrarlanabilir. Bu ekranda Tamam (*Ok*) butonuna basılıp bu ekran kapatılır. 6 numara ile belirtilen Uygula (*Apply*) butonuna basılmasıyla ilgili filtre uygulanır. 5 numara ile gösterilen bölümde ilgili değişikliğin yapıldığı görüntülenecektir.



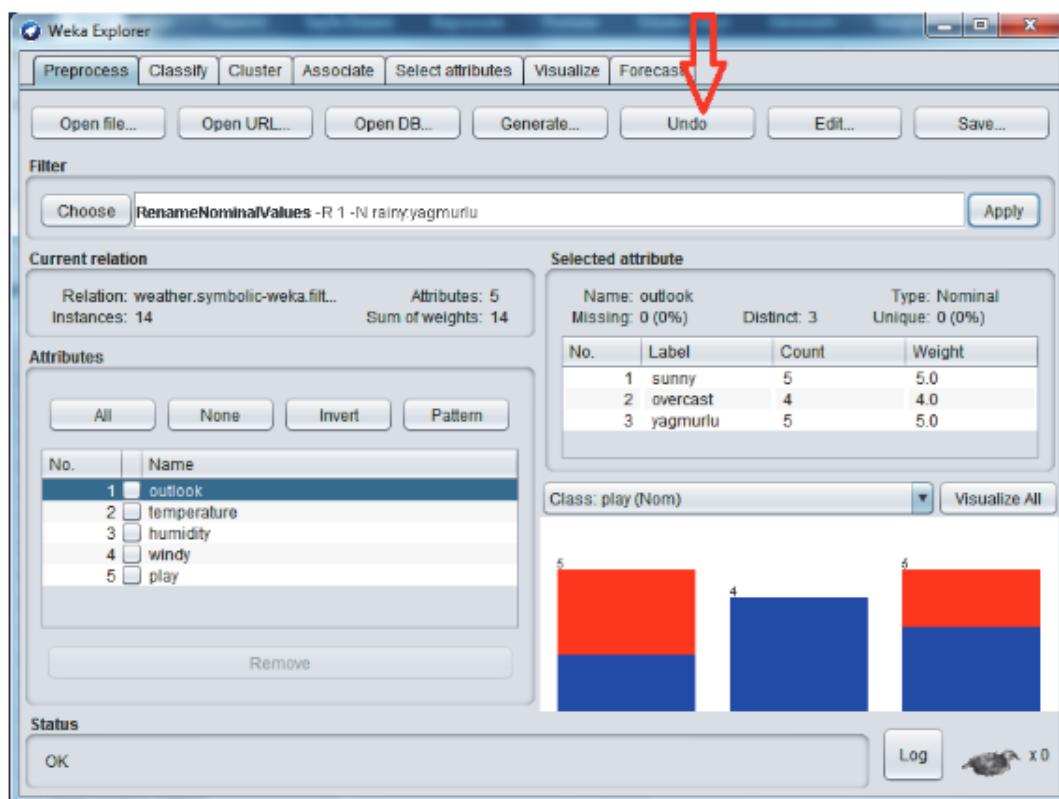
Şekil 4.38. Sınıflandırılmış Metinlerde İsim Değişikliği Filtresi

Filtre listesi açık iken sol en alta Filtre (Filter) butonu uygulanabilecek filtrelerin yeteneklerini listeler. Aşağıdaki şekilde seçilen veri yalnızca tarih ve metin sınıfı tiplerinden oluştugu için bu türlere yönelik filtreler kullanılabılır. Yanlış bir filtrle seçilmesi durumunda Uygula (Apply) butonu pasif görünecektir.



Şekil 4.39. Filtreleme Yetenekleri Ekranı

Filtre işlemi uygulandıktan sonra ekranın üstündeki Geri (*Undo*) butonu pasif iken aktif hale gelecektir. Bu butona basılması durumunda en son yapılan değişiklik olan filtre geri alınacaktır. Kaydet (*Save*) butonu ise değişiklik yapılmış halin yeni bir ARFF uzantılı dosyaya kaydedecektr.



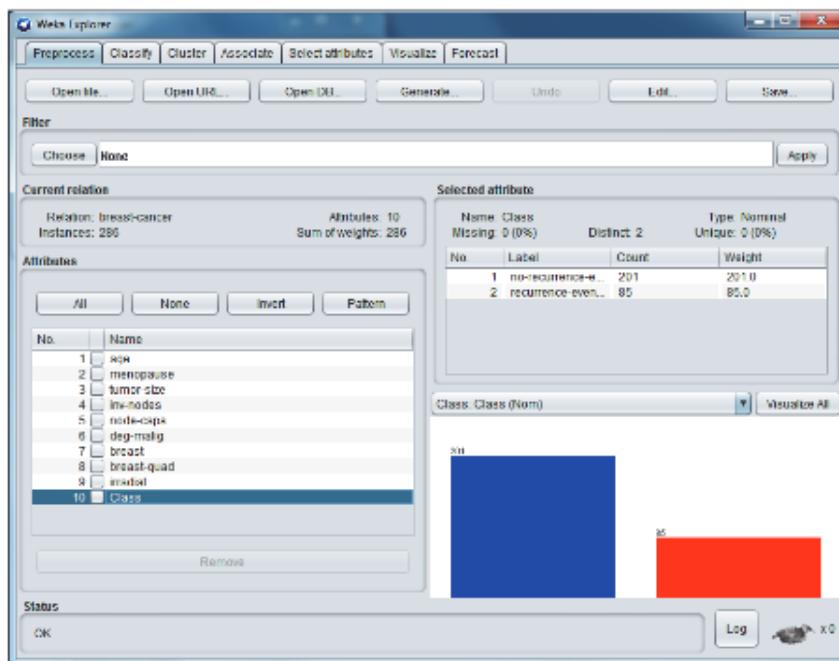
Şekil 4.40. Filtre Uygulanması Sonrası Geri Butonu

Daha önceki konularda anlatıldığı üzere Düzenle (*Edit*) butonu ile verilerin son hali listelenir ve istenirse üzerinde değişiklik de yapılabilir.

4.4.4. Öğrenme Şemaları ile Eğitim ve Test

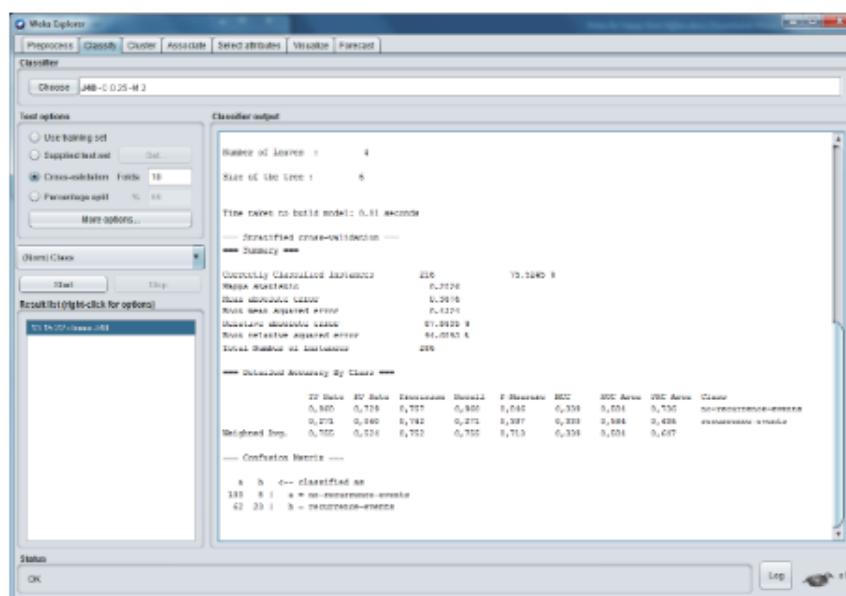
Sınıflandırma sekmesi, sınıflandırma veya regresyon yöntemleri ile verilerin eğitilmesine ve test edilmesine imkân tanır. Tahmin edilecek niteliğin metin sınıfı veya sayısal olması durumuna göre performans ölçümleri değişecektir.

Aşağıdaki şekilde breast-cancer.arff dosyası seçilmiştir. Dikkat edilirse son nitelik olan Sınıf (*Class*) sütununun veri tipi metin grubu (*nominal*)’dur. Tahmin için bu değer kullanılacaktır.



Şekil 4.41. breast-cancer.arff Dosyasının Yüklenmesi

Sınıflandırma (*Classify*) sekmesine geçerek sınıflandırma algoritmalarından birisi seçilebilir. Dikkat edilirse sayısal tahmin üreten algoritmalar pasif görünmektedir. Örnek olarak M5P algoritması gösterilebilir. Burada örnek olarak J48 algoritması uygulanacaktır. Diğer seçenekler (*More options*) butonu ile tahmin değerlerinin gözükmesi de sağlandıktan sonra Başlat (*Start*) butonu ile algoritma başlatılabilir.



Şekil 4.42. Breast-cancer.arff Dosyasına J48 Algoritmasının Uygulanması

Buradaki sonuçlarda 286 örnekten 216 tanesinin doğru bir şekilde sınıflandırıldığı ve %75.52 oranla doğru sınıflandırma yaptığı görülmektedir.

```
==== Run information ====
Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: breast-cancer
Instances: 286
Attributes: 10
    age
    menopause
    tumor-size
    inv-nodes
    node-caps
    deg-malig
    breast
    breast-quad
    irradiat
    Class
Test mode: 10-fold cross-validation

==== Classifier model (full training set) ====
J48 pruned tree
-----
node-caps = yes
| deg-malig = 1: recurrence-events (1.01/0.4)
| deg-malig = 2: no-recurrence-events (26.2/8.0)
| deg-malig = 3: recurrence-events (30.4/7.4)
node-caps = no: no-recurrence-events (228.39/53.4)

Number of Leaves :      4
Size of the tree :      6

Time taken to build model: 0.04 seconds

==== Predictions on test data ====
inst#,actual,predicted,error,prediction
1,2:recurrence-events,1:no-recurrence-events,+,0.799
2,2:recurrence-events,2:recurrence-events,,0.667
3,2:recurrence-events,1:no-recurrence-events,+,0.799
...
26,1:no-recurrence-events,1,no-recurrence-events,,0.761
27,1:no-recurrence-events,1,no-recurrence-events,,0.761
28,1:no-recurrence-events,1,no-recurrence-events,,0.761

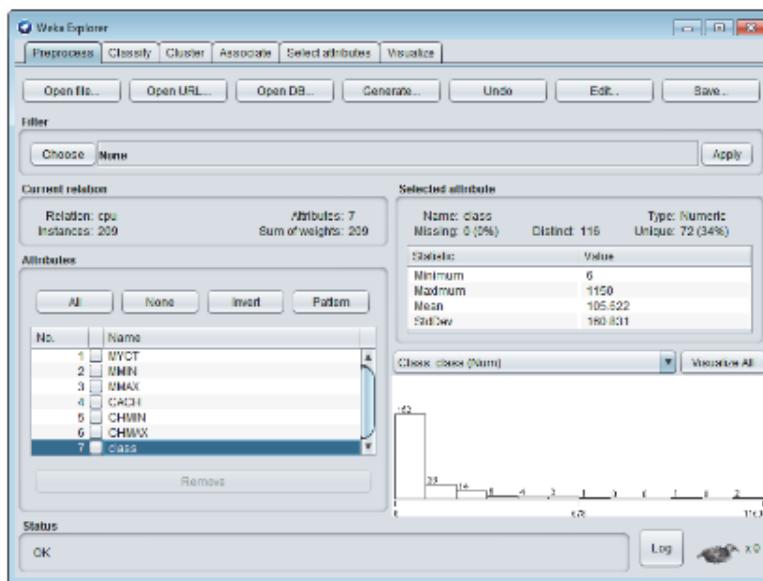
==== Stratified cross-validation ===
==== Summary ===
Correctly Classified Instances   216      75.5245 %
Kappa statistic                 0.2826
Mean absolute error              0.3676
Root mean squared error          0.4324
Relative absolute error           87.8635 %
Root relative squared error      94.6093 %
Total Number of Instances        286

==== Detailed Accuracy By Class ====
      TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
      0,960  0,729  0,757  0,960  0,846  0,339  0,584  0,736  no-recurrence-events
      0,271  0,040  0,742  0,271  0,397  0,339  0,584  0,436  recurrence-events
Weighted Avg.  0,755  0,524  0,752  0,755  0,713  0,339  0,584  0,647

==== Confusion Matrix ====
a  b  <-- classified as
193  8 |  a = no-recurrence-events
 62  23 |  b = recurrence-events
```

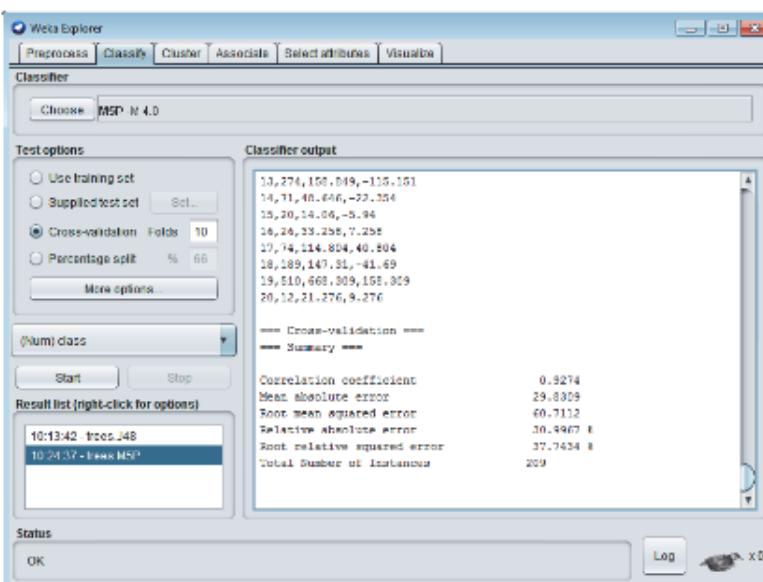
Şekil 4.43. Gruplanmış Metin Türünde Veriye J48 Algoritmasının Uygulanması Sonucu

Sayısal bir sınıflandırma yapmak için ise *cpu.arff* adlı dosya seçilir. Aşağıdaki şekilde de görüldüğü üzere en son sütun olan *Sınıf* (*Class*) adlı değerler sayısal (*numeric*) değerlerdir.



Şekil 4.44. Cpu.arff Adlı Dosyanın Seçilmiş Görüntüsü

Buradaki seçili olan verilere M5P algoritması seçilerek uygulanabilir. Dikkat edilirse bu kez J48 algoritmasının pasif olduğu görülecektir. Başlat (*Start*) butonu ile algoritma uygulanabilir. Aşağıdaki şekilde algoritmanın bittiği hali görülmektedir.



Şekil 4.45. Cpu.arff Adlı Dosyaya M5P Algoritmasının Uygulanmış Hali

Bu algoritmanın daha detaylı sonuçları görülmek istenirse aşağıdaki şekil incelenebilir.

```
==== Run information ====
Scheme: weka.classifiers.trees.M5P -M 4.0
Relation: cpu
Instances: 209
Attributes: 7
    MYCT
    MMIN
    MMAX
    CACH
    CHMIN
    CHMAX
    class
Test mode: 10-fold cross-validation

==== Classifier model (full training set) ====
M5 pruned model tree:
(using smoothed linear models)

CHMIN <= 7.5 : LM1 (165/12.903%)
CHMIN > 7.5 :
| MMAX <= 28000 :
| | MMAX <= 13240 :
| | | CACH <= 81.5 : LM2 (6/18.551%)
| | | CACH > 81.5 : LM3 (4/30.824%)
| | MMAX > 13240 : LM4 (11/24.185%)
| | MMAX > 28000 : LM5 (23/48.302%)

LM num: 1
class = -0.0055 * MYCT + 0.0013 * MMIN + 0.0029 * MMAX + 0.8007 * CACH + 0.4015 * CHMAX + 11.0971
LM num: 2
class = -1.0307 * MYCT + 0.0086 * MMIN + 0.0031 * MMAX + 0.7866 * CACH - 2.4503 * CHMIN + 1.1597 *
CHMAX + 70.8672
LM num: 3
class = -1.1057 * MYCT + 0.0086 * MMIN + 0.0031 * MMAX + 0.7995 * CACH - 2.4503 * CHMIN + 1.1597 *
CHMAX + 83.0016
LM num: 4
class = -0.8813 * MYCT + 0.0086 * MMIN + 0.0031 * MMAX + 0.6547 * CACH - 2.3561 * CHMIN + 1.1597 *
CHMAX + 82.5725
LM num: 5
class = -0.4882 * MYCT + 0.0218 * MMIN + 0.003 * MMAX + 0.3865 * CACH - 1.3252 * CHMIN + 3.3671 *
CHMAX - 51.8474

Number of Rules : 5
Time taken to build model: 0.36 seconds

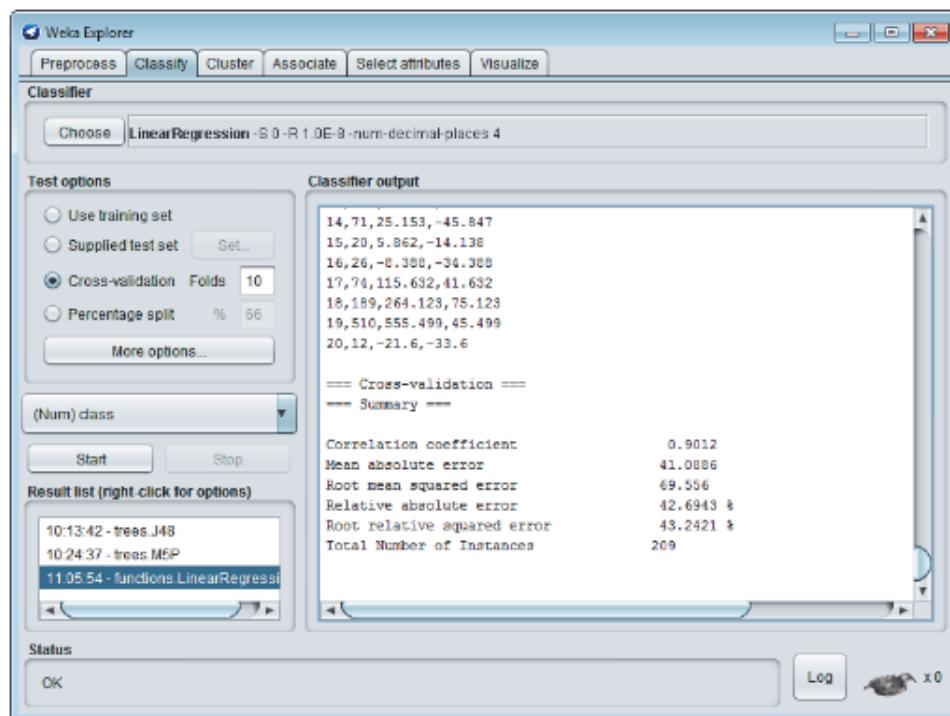
==== Predictions on test data ====
inst#,actual,predicted,error
1,70,72.759,2.759
2,116,145.62,29.62
...
20,12,21.276,9.276

==== Cross-validation ====
==== Summary ====
Correlation coefficient      0.9274
Mean absolute error          29.8309
Root mean squared error      60.7112
Relative absolute error       30.9967 %
Root relative squared error   37.7434 %
Total Number of Instances     209
```

Şekil 4.46. Cpu.arff Adlı Dosyanın M5P Algoritma İçin Detaylı Sonuçları

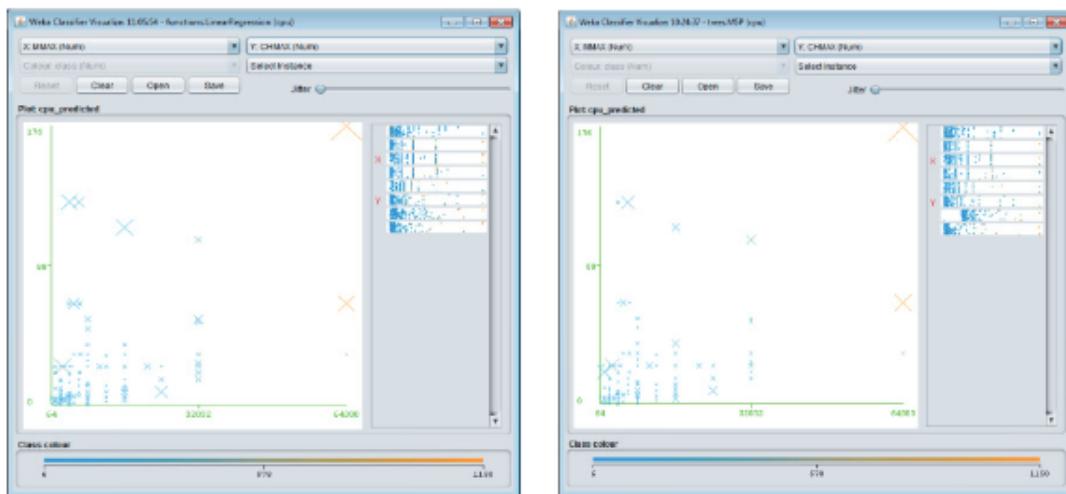
Burada dikkat edilirse sonuç analiz ölçümleri birbirinden farklılıklar göstermektedir. Sınıflandırma modeli başlığı (*Classifier model (full training set)*) altında budanmış model ağaçları görüntülenmektedir. Kök, CHMIN niteliğine göre bölünmektedir. Sağ ve sol dallara doğrusal bir modele göre ayırmaktadır. En son yaprakların yanında parantez içinde bazı sayılar verilmektedir. Bu sayılarından ilki kendisine ulaşan örnek sayısını göstermektedir. İkinci sayı ise tüm veriler üzerinden hesaplanan hata karelerinin ortalamasının karekökü ile standart sapmanın yüzdesidir.

Sayısal tahminlerde kullanılan doğrusal regresyon (*LinearRegression*) algoritmalar listesi altındaki fonksiyonlar altında yer alır. Bu algoritma tek bir doğrusal regresyon modeli oluşturur ve bu nedenle performansı daha düşüktür. Aşağıdaki şekilde aynı cpu.arff verilerine doğrusal regresyon algoritmasının uygulanmış hali görülmektedir.



Şekil 4.47. Cpu.arff Dosyasına Doğrusal Regresyon Uygulanmış Sonuçları

M5P algoritması ile Doğrusal Regresyon (*LinearRegression*) algoritması sonuçları karşılaştırıldığında M5P'nin daha iyi sonuçlar ürettiği rahatlıkla görülmektedir. Bu durumun bir grafik olarak gösterilmesi istenirse analiz adı üzerinde sağ tıklayarak Sınıflandırma Hata Grafiği (*Visualize classifier errors*) seçeneği ile aşağıdaki şekiller çıkarılabilir.

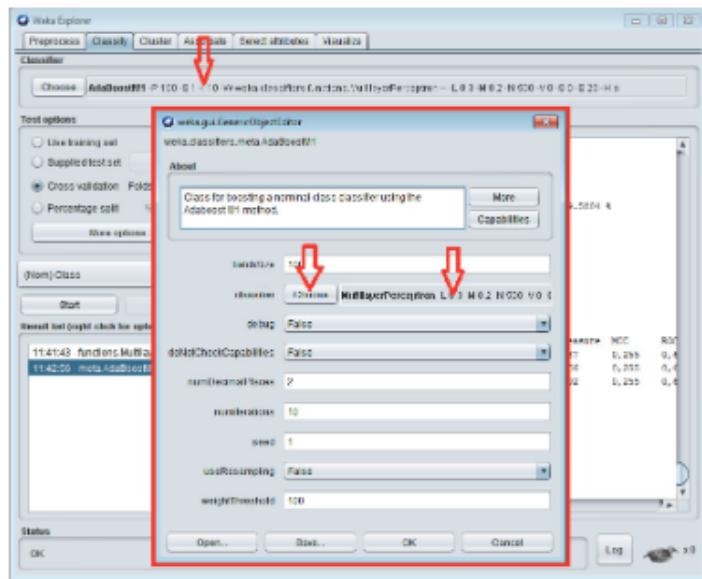


Şekil 4.48. Cpu.arff Dosyası İçin Doğrusal Regresyon ve M5P Algoritmasının Sınıflandırma Hata Grafiği

Makul bir dağılım sağlamak için X ekseninde MMAX değeri, Y ekseninde CHMAX değeri gösterilmiştir. Grafikler incelendiğinde M5P için daha küçük çarpı işaretlerinin olması onun daha üstün olduğunu göstermektedir.

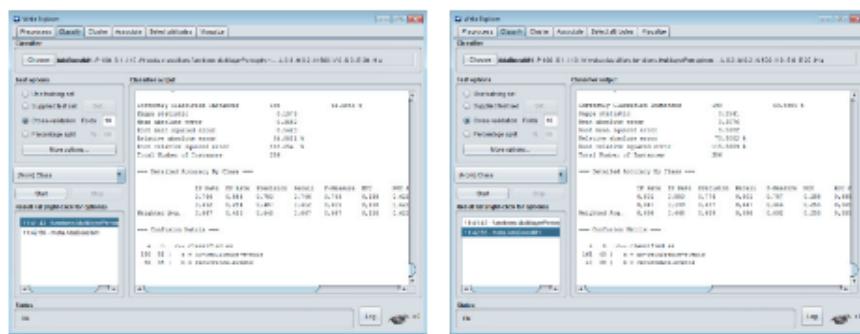
4.4.5. Meta Öğrenme Algoritması Kullanma

Meta öğrenmeler basit sınıflandırma algoritmalarını alır ve onları daha güçlü öğrenme algoritmalarına dönüştürür. Önceki modelin yanlış sınıflandırıldığı örnekleri vurgulamak için her yeni modeli eğiterek işlem yapar. *breast-cancer.arff* dosyasına Yapay Sinir Ağları (*MultilayerPerception*) algoritması uygulanırsa ve aynı verilere AdaBoostM1 algoritması ile Yapay Sinir Ağları (*MultilayerPerception*) alt algoritması ile 10 katına kadar uygulanırsa sonuçlar aşağıdaki gibi olur. AdaBoostM1 algoritması sınıflandırma algoritmaları altında Meta klasörü altında yer almaktadır. Algoritmayı uygulamak için seçim yapıldıktan sonra ilgili parametrelerine ulaşmak aşağıdaki şekilde gösterilen satırı tıklanarak ilgili ekrana ulaşılır. Açılan ekranda sınıflandırma (*classifier*) satırında **Seç (Choose)** adlı buton ile istenilen alt algoritma seçilebilir. Buradaki uygulamada yine Yapay Sinir Ağları (*MultilayerPerception*) algoritması seçilmiştir. İstenirse bundan farklı bir algoritma da seçilebilir. Üst algoritma ile alt algoritma aynı olmak zorunda değildir.



Şekil 4.49. AdaBoostM1 Algoritması Parametre Giriş Ekranı

Seçili olan alt algoritma için de parametre girişi yapılmak istenirse yan satırına tıklanması yeterli olacaktır. Tamam (*OK*) butonu ile ekran kapatılır ve Başlat (*Start*) butonu ile algoritma uygulanmaya başlanacaktır. Doğal olarak önceki algoritmanın süresine nazaran daha fazla zaman alacaktır.



Şekil 4.50. AdaBoostM1 Algoritmasının Yapay Sinir Ağları ile Karşılaştırılmış Sonuçları

Buradaki sonuçlar karşılaştırıldığında AdaBoostM1 algoritmasının iç içে Yapay Sinir Ağları (*MultilayerPerception*) uygulanması ile 286 örnekten 185 yerine 199 tanesini doğru sınıflandırdığı görülmektedir.

4.4.6. Kümeleme Analizleri

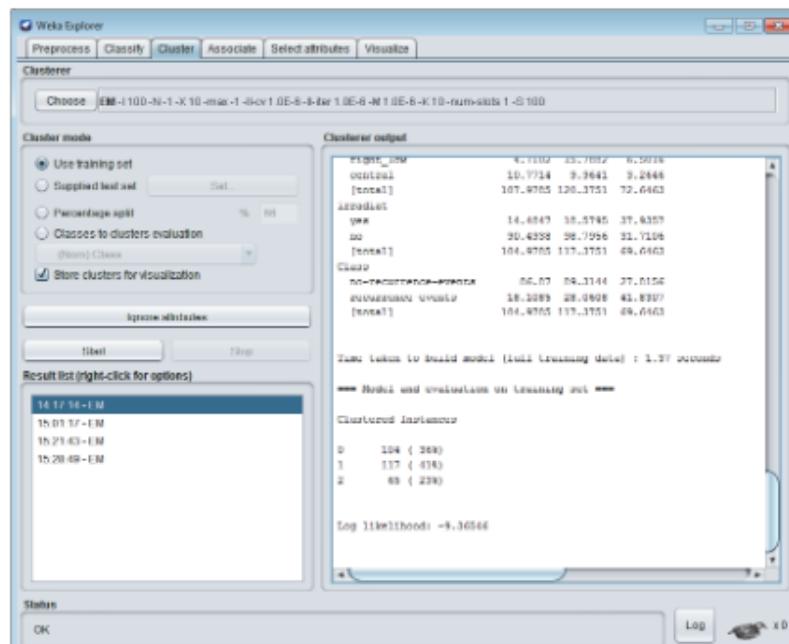
Kümeleme analizi yapılırken, Weka küme sayısını ve her kümenin kaç örneği içerdigini göstermektedir. Bazı algoritmalarla ise, parametre olarak ekranın maksimum ve minimum küme sayısı girilebilmektedir. Olasılıksal kümeleme analizlerinde Weka, eğitim verisi üzerindeki kümelenmelerin

olabilirlik günlük değerini (*log-likelihood*) ölçmektedir. Bu değerin büyük olması modelin verilere daha iyi uyduğunu göstermektedir. Küme sayısının parametre girerken yüksek verilmesi olasılığı artırlabilir fakat aşırılık (*overfit*) oluşabilir.

Kümeleme (*Cluster*) sekmesindeki menüler ve işlemler çoğunlukla sınıflandırma (*Classify*) sekmesine benzer. Hatta aynı algoritmaların bazıları dahi kullanılabilir. Eğitim ve test verilerinin ayrıştırılmasında dört farklı seçenek kullanılabilir. Bunlar;

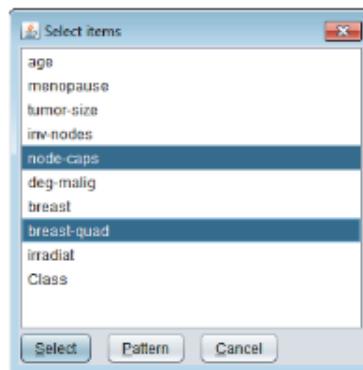
- **Eğitim verisi kullanma (Use training set):** Seçili olan verilerinin tamamının eğitim verisi olarak kullanılması
- **Ayrı bir test verisi (Supplied test set):** Eğitim işlemi yapıldıktan sonra ayrı bir dosya halinde test verilerinin kullanılması
- **Yüzdesel ayırma (Percentage split):** Eğitim ve test verilerinin yüzdesel bir oranla ayrıştırılması, girilen oran eğitim verisinin yüzde kaç olacağı
- **Kümelemede sınıf kullanma (Classes to clusters evaluation):** Sınıf bilgisinin kümelerin değerlendirilmesinde test amaçlı kullanılmasına olanak sağlayan seçenek

Aşağıdaki şekilde Kümeleme (*Cluster*) sekmesinin bir görüntüsü verilmiştir.



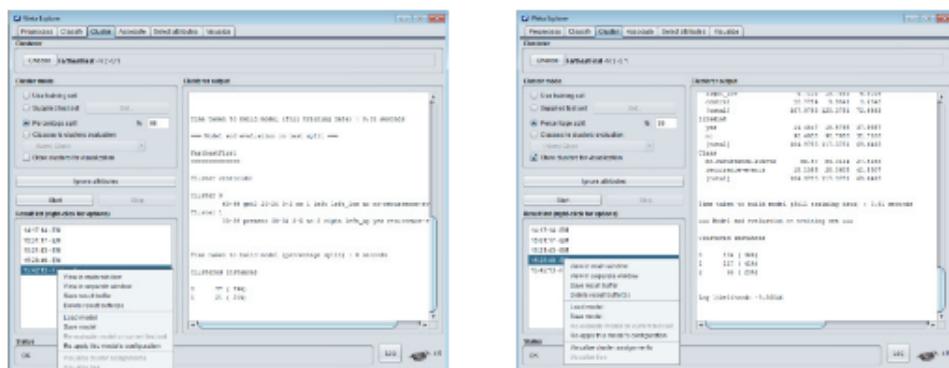
Şekil 4.51. Kümeleme Sekmesinden Bir Görüntü

Kümeleme işlemi yapıılırken veriler içindeki bazı sütunların yok sayılması sağlanabilir. Bunun için Nitelikleri Yoksay (*Ignore attributes*) butonu tıklanır ve aşağıdaki gibi bir ekran açılır. Bu ekranda seçilen bir veya birden fazla sütun kümeleme analizinde göz ardı edilerek dikkate alınmaz. Birden fazla seçim yapmak için CTRL ve Shift tuş kombinasyonları kullanılabilir. Eğer sütun başlıklarını çok fazla ve belirli bir arama kuralına göre filtre yaparak seçim yapmak istenirse Desen (*Pattern*) butonu ile ilgili şart verilebilir.



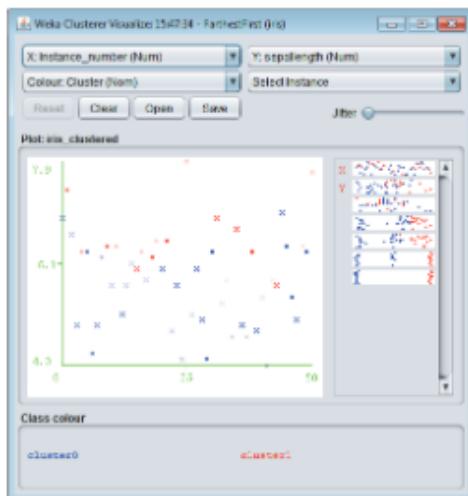
Şekil 4.52. Kümeleme Analizinde Bazı Sütunları Yoksayma Ekranı

Seçim işlemi tamamlandıktan sonra Seç (Select) butonu ile seçim işlemi bitirilir. Bazı sütundaki verilerin çıkartılması ve oluşan sonuçların karşılaştırılması kalan sütunların başarımı ne kadar etkilediğini incelemeye fayda sağlayabilir. Eğer oluşturulan kümeler için görselleştirme özellikleri kullanılaraksa Görselleştirme için Kümeleri Sakla (*Store clusters for visualization*) seçeneği işaretlenmelidir. Aksi durumda görselleştirme özellikleri devre dışı kalacak ve kullanılamayacaktır. Bu durum alan kazanımı için kullanılmaktadır. Aşağıdaki şekilde bu seçeneğin seçili olduğu ve seçili olmadığı her iki durum gösterilmektedir.



Şekil 4.53. Kümeleme Analizinin Saklanıp Saklanmaması Durumları

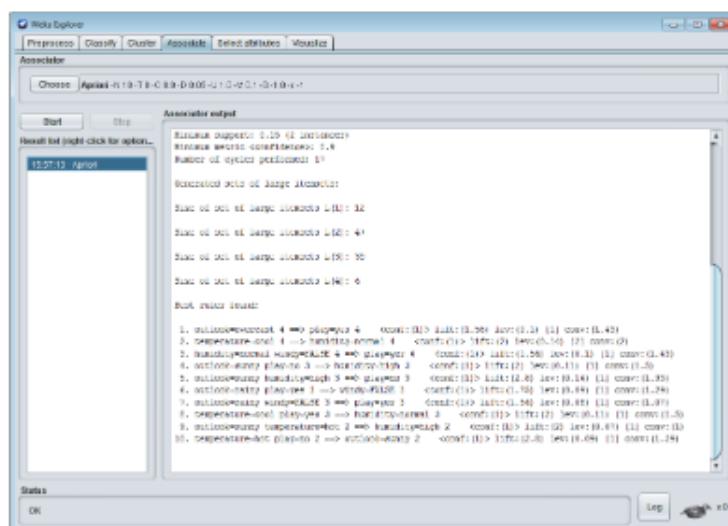
Görselleştirme paneli yukarıdaki şekilde gösterildiği gibi Küme analizi menüsü (Visualize cluster assignments) seçilerek açılır. Aşağıdaki şekilde iris.arff dosyası için iki kümeli bir analiz yapılmış ve renklere göre iki gruba ayırtılmıştır.



Şekil 4.54. Kümeleme Analizi Görselleştirme Ekranı

4.4.7. İlişkilendirme Kuralları

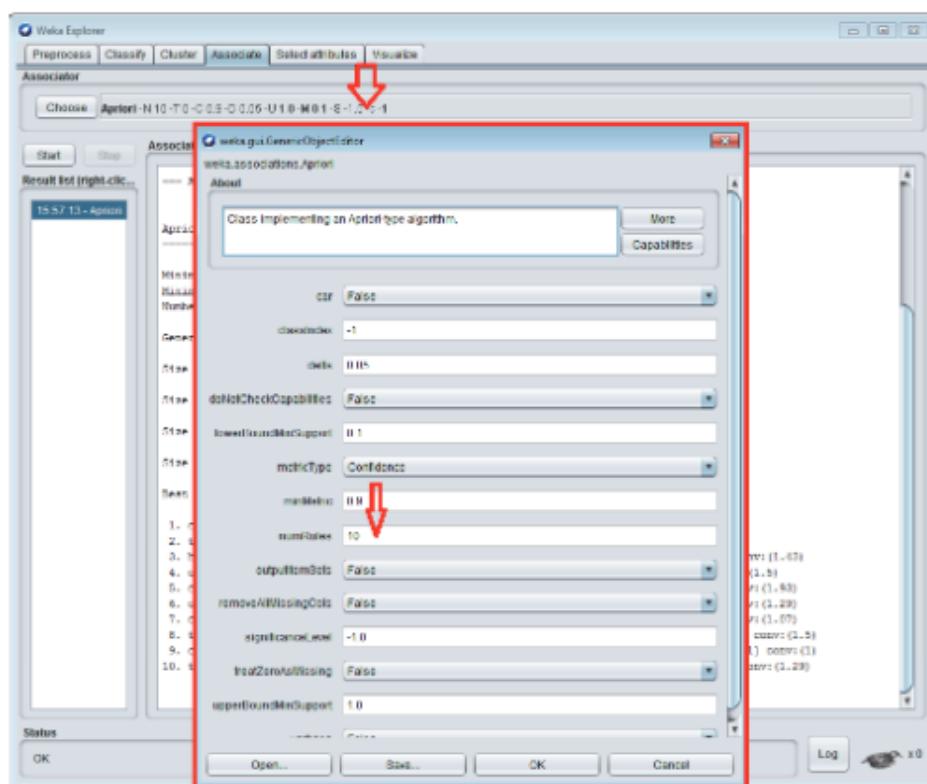
İlişkilendirme sekmesi diğer sekmelere nazaran daha basit özelliklere sahiptir. Weka çeşitli algoritmalar ile birlikte analizi yapar fakat bunların değerlendirmek gibi bir şey söz konusu değildir. Aşağıdaki şekilde weather.nominal.arff dosyası için varsayılan olarak seçili gelen Apriori algoritması uygulanmıştır.



Şekil 4.55. weather.nominal.arff Dosyası İçin Birlikteşlik Analizi Sonucu

Weka varsayılan olarak en çok kullanılan üç algoritmayı listelemektedir. Fakat paket yöneticisi (*Tools->Package Manager*) ekranından istenirse farklı algoritma paketleri kurularak kullanılabilir. Burada yapılan analiz en çok birlikte geçen verileri sıraya dizme mantığıdır. En baştaki birbiri ile ilişkisi en yüksek olan verileri ifade eder. Ondan sonrakiler ise bir sonraki en yüksek ilişki içeren verileri gösterir. Bu durum Güven (*Conf.*) katsayısının azalması ile de görülebilmektedir.

Yukarıdaki şekilde en çok birlikte geçen veya birbiri ile ilişkisi olan ilk 10 satır verilmiştir. İstenirse bu satır sayısı çoğaltılabılır. Aşağıdaki şekilde görüldüğü üzere algoritma adının olduğu satıra tıklanınca parametre giriş ekranı açılmaktadır. Buradaki kural sayısı (*numRules*) değeri en çok kaç satırın gösterileceğini ifade etmektedir.

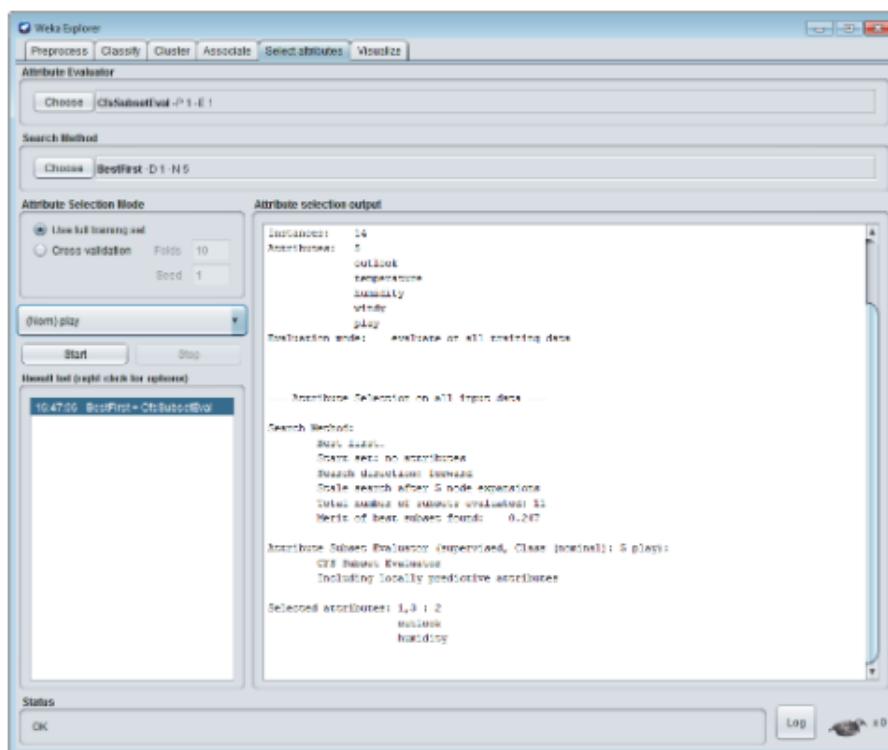


Şekil 4.56. Birliktelik Analizi Parametre Giriş Ekranı

Bu tür analizlerin en çok kullanıldığı alan olarak marketlerden yapılan alışveriş listesi verileri dikkate alındığında en çok birlikte alınan malzemelerin listesinin çıkartılması örnek olarak verilebilir. Weka ilişki sıralaması yaparken en çok kullanılan Güven (*Conf.*) katsayısını dikkate almaktadır. Fakat farklı uygulamalarda diğer katsayıların dikkate alınması gerekebilir.

4.4.8. Nitelik Seçimi

Nitelik seçimi sekmesi çeşitli yöntemler ile özellik seçim işlemi yapmayı sağlar. Bu analiz bir Özellik Değerlendiricisi (*Attribute Evaluator*) ve Arama Yöntemi (*Search Method*) bölümlerinden oluşur. Her ikisi de daha önce anlatıldığı gibi uygun bir algoritma seçilir ve ekranlar aracılığıyla yapılandırılır. Aşağıdaki şekilde bu sekme görüntülenmektedir.

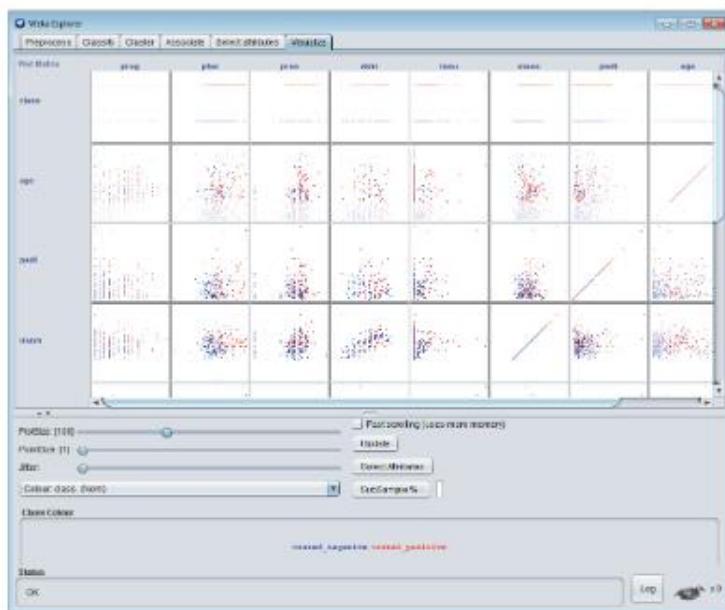


Şekil 4.57. Nitelik Seçim Sekmesi

Verilerin ayrıştırılmasında yalnızca iki yöntem kullanılabilir. Birinci seçenek olarak tüm verilerin eğitim için kullanılması, ikinci seçenek ise çapraz doğrulama yöntemi ile ayrılmaktır. Analizin Başlat (*Start*) butonu başlatılması sonrası seçili olan nitelikler çıktı ekranında listelenir.

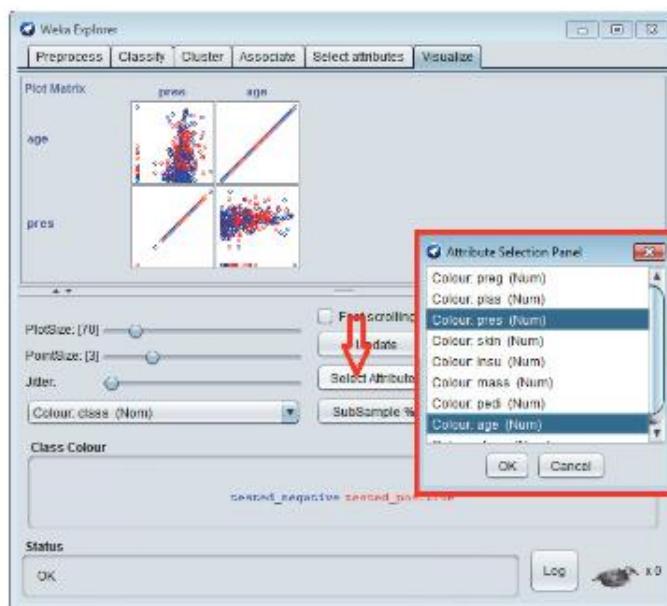
4.4.9. Görselleştirme

Bu sekme bir veri kümesini görselleştirmek için kullanılır. Bir kümeleme ve sınıflandırma modeli sonucunda elde edilen verileri değil de veri kümesinin kendisini görselleştirmek için kullanılır. Aşağıda şekil diabetes.arff dosyası seçili iken buradaki her bir sütun başlığının diğer sütun başlıklarını ile olan X,Y grafiklerini göstermektedir.



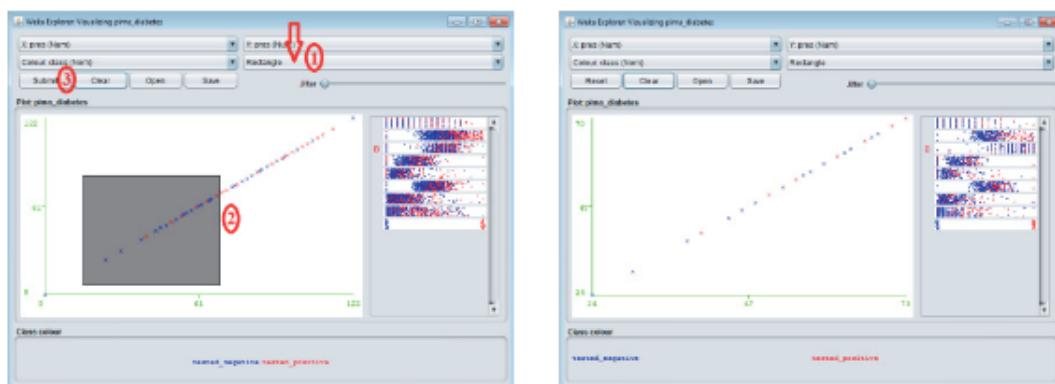
Şekil 4.58. Diabetes.arff Dosyası Verilerinin Görselleştirilmiş Hali

Burada her bir grafik üzerine fare ile sağ tıklayarak yeni pencerede açılabilir ve daha büyük olarak incelenebilir. Benzer şekilde her bir grafik alanının çizim alanı büyütülmek istenirse Çizim Boyutu (*PlotSize*) kaydırma çubuğu, grafik içindeki noktaların boyutu büyütülmek istenirse Nokta Boyutu (*PointSize*) kaydırma çubuğu kullanılır. Titreşim (*Jitter*) kaydırma çubuğu ile noktaların birbirinden ayrışma durumunu belirlenebilir. Fakat tüm bunların hepsi Güncelle (*Update*) butonu ile ekrana yansıyacaktır.



Şekil 4.59. Görselleştirme Ekranında Belirli Niteliklerin Seçimi

Belirli niteliklerin yalnızca grafik olarak gösterilmesi istenirse Nitelik Seç (*Select Attributes*) butonu ile bir veya birden fazla nitelik seçimi CTRL ve Shift tuş kombinasyonları ile yapılabilir. Bu durum yukarıdaki şekilde gösterilmektedir.



Şekil 4.60. Görüleştirmeye Ekranı Yakınlaştırma

Herhangi bir grafik tıklandığında açılan yeni pencerenin sağ üst köşesindeki açılır pencereden 1 numara ile gösterilen yerden dikdörtgen (*Rectangle*) seçilir ise ve ardından grafik alanında yakınlaştırılmak istenen alan dikdörtgen olarak taranır. Gönder (*Submit*) butonu ile grafiğin seçili alanının yakınılaştırıldığı görülecektir. İşlem geri alınmak istendiğinde ise yine 3 numara ile belirtilen yerdeki Gönder (*Submit*) butonunun Sıfırla (*Reset*) butonuna dönüştüğü görülecektir. Sıfırla (*Reset*) butonu işlemi eski haline getirecektir.

4.5. Filtre Algoritmaları

Bu başlıkta daha önce anlatılan filtre uygulamalarının kullanım konusu dışında algoritmalarдан bahsedilecektir. Filtrelere Explorer, Experimenter ve Knowledge Flow ekranlarından ulaşılabilir. Tüm filtrelerin amacı bir şekilde veri kümesini dönüşüme tabi tutmaktır. Seç (*Choose*) butonu ile bir filtre seçildiğinde filtre adı hemen yanındaki satırda gözükecektir. Parametre girdi değerlerini vermek için bu satırı tıklamak ve açılan ekranın girişleri yapmak yeterlidir. Tamam (*OK*) butonuna basıldıktan sonra filtre adının yanında eksi (-) işaretleri ile parametre değerleri belirtilir. Bu satır aynı zamanda Weka komutlarının doğrudan kullanımı ile aynıdır.

İki tür filtre vardır. Bunlar gözetimli filtreler ve gözetimsiz filtrelerdir. Filtreler genellikle önce eğitim veri kümesine uygulanır ve sonrasında test

veri kümesine uygulanır. Filtre gözetimli bir filtre ise bunu test verilerine uygulamak sonuçların sapmasına neden olur. Test verilerine uygulanması gereken eğitim verisinden türetilen ayırtlaştırma aralıklarıdır. Gözetimli filtreler kullanıldığında sonuçların hatasız bir şekilde değerlendirildiğinden ve gözetimli filtrelerden kaynaklanan herhangi bir sorun oluşmadığından emin olunmalıdır.

Weka'nın gözetimsiz filtreleri ve gözetimli filtreleri ayrı başlıklar halinde ele alınacaktır. Her türün içinde, veri kümesindeki nitelikler üzerinde çalışan birbirinden üstün özellikleri vardır. İstenilen bir filtre hakkında daha fazla bilgi sahibi olunmak istenirse, Weka Explorer ekranında filtreyi seçip yanında açılan bilgi penceresinden yararlanmak mümkündür. Buradaki filtre açıklamaları Witten ve diğerlerinin [45] çalışması ile Weka web sitesinden [46] çeviri yapıp yorumlanarak açıklanmıştır. Ayrıca Breiman [47]'nın çalışması da bu alt başlıkların açıklamasında kaynak teşkil etmiştir.

4.5.1. Gözetimsiz Nitelik Filtreleri

Burada öncelikle belirli bir nitelik belirtmeden kullanılan gözetimsiz filtreler ele alınacaktır.

4.5.1.1. Nitelikleri Silme ve Ekleme

Add

Değeri eksik olarak verilen pozisyon'a değer ekler. Filtre metodu seçil dikten sonra parametre giriş ekranında niteliğin adı nitelik listesinde görüneceği gibi belirtilebilir. Metinsel sınıf değerleri için de aynı şekildedir. Tarih türündeki değerler için istenirse tarih formatı da belirtilebilir.

Copy

Nitelikler üzerinde işlem yaparken asıl veriyi kaybetmemek amacıyla var olan nitelik kopyalanabilir. Birden fazla nitelik kopyalanmak istenirse nitelik ID'leri alanına (*attributeIndices*) "first-3,5,9-last" şeklinde yazılabilir. Bu durumda 1, 2, 3, 5, 9, 10, 11, 12 nitelikleri kopyalanacaktır. Yine istenirse burada belirtilecek ID'lerin tam tersi işleme tabi tutulabilir. Bunun için seçimi ters çevir (*InvertSelection*) bilgi giriş kutusundan Evet (*True*) seçilmesi yeterlidir. Bu durum birçok filtre için aynıdır.

AddUserFields

Veri kümesine bir seferde birden fazla yeni nitelik eklemek için kullanılır. Her bir nitelik için adı, türü ve değeri belirtilir.

AddID

Nitelikler listesinde istenilen bir dizine sayısal bir tanımlayıcı eklemek için kullanılır. Bu tanımlayıcı, bir veri kümesinin diğer filtreler tarafından işlendikten sonra özel olarak bazı örneklerin izlenmesi amaçlı kullanılır.

Remove

Belirtilen bir niteliği ya da diğer adıyla sütunu silmek için kullanılır.

RemoveType

Belirtilen herhangi bir veri tipine (*sayısal, metin, tarih vb.*) sahip tüm değerleri silmek için kullanılır.

RemoveUseless

Kullanılmayan değerleri siler. Yani sayısal değerler için tüm örneklerde sabit değer taşıyan veya metin sınıfı değerleri için tüm örneklerin her birinde birbirinden farklı olan değerleri siler. İstenirse belirli bir oran verilerek de silinebilir. Yani sayısal örnekler için %90 oranında sabit kalan değerleri silmek gibi. Unutulmamalıdır ki bazı gözetimsiz filtreler Önişleme (*Preprocess*) panelinde metin sınıfı veri türü olarak ayarlanması durumunda farklılıklar gösterebilir. Varsayılan olarak son nitelik metin sınıfı türündedir. Örneğin *RemoveType* ve *RemoveUseless* metodlarından her ikisi de metin sınıfı türündeki niteliği atlar.

InterquartileRange

Örneklerin değerlerinin aykırı veya aşırı değer olarak kabul edilip edilmeyeceğini belirler. Aykırı ve aşırı değer, bir niteliğin 25. ve 75. çeyrekte bulunan değerler arasındaki farka dayanmaktadır. Aşırı değer ve aykırı değerler kullanıcı tarafından parametre giriş ekranına verilmektedir. Sütunda yer alan değerler, kullanıcının girmiş olduğu aşırı değeri (*extremeValuesFactor*) aşarsa veya 25. çeyrekten küçük, 75. çeyrekten büyük olursa aşırı değer olarak işaretlenir. Bu filtre örnekleri aşırı veya aykırı olarak işaretlenecek şekilde yapılandırılabilir. Tüm aşırı değerleri aykırı değer olarak işaretlemek mümkündür. Bunu yapmak bir niteliğin değerlerinin medyandan saplığı kaç tane çeyrek skala olduğunu görmek için de kullanılabilir.

AddCluster

Filtreleme yapmadan önce verilere küme algoritması uygulamak için kullanılır. Kümeleme algoritması bu filtrenin parametre giriş ekranından seçilir. Kümeleyici algoritması da tipki filtre gibi parametre girdileri yaparak yapılandırılabilir. İşlem sonrası bu metot her bir örneğe yeni bir nitelik olarak küme numarası atayacaktır. Böylelikle istenirse daha önce anlatıldığı gibi belirli bir küme numarası seçilerek göz ardı edilmesi sağlanabilir.

ClusterMembership

Küme üyelik değerleri oluşturmak için kullanılır. Sütun değerleri buna eşit olan her bir örneğin yeni bir sürümünü oluşturur. Kümeleme sırasında istenirse ayarlanacak bir sınıf yok sayılabilir.

AddExpression

Sayısal veri türlerinden matematiksel bir formül baz alınarak yeni bir sütun (nitelik) oluşturmak için kullanılır. Matematiksel fonksiyonlardan *log*, *exp*, *abs*, *sqrt*, *floor*, *ceil*, *rint1*, *sin*, *cos*, *tan* ve parantez işaretleri aritmetik operatörler (+,-,*./,^) kullanılabilir. Nitelikler *a1*, *a2*, *a9* şeklinde belirtilir. Yeni niteliğin değerini, sağlanan ifadenin bir postfix ayrıştırma-sıyla değiştiren bir hata ayıklama seçeneği de vardır.

MathExpression

Bu filtre metodu da *AddExpression* metoduna benzer. Fakat bu metot birden çok niteliğe uygulanabilir. Yeni bir nitelik (*sütun*) oluşturmak yerine var olan değeri değiştirir. Bu nedenle yazılacak matematiksel fonksiyon diğer niteliklerin değerini kullanamaz. *AddExpression* metodunda kullanılan özelliklerin yanı sıra niteliğin (sütunun) minimum, maksimum, ortalama, toplam, toplam karesi ve standart sapması gibi özellikleri de kullanılabilir. Ayrıca basit bir kullanımla *if-then-else* ifadeleri de yazılabilir.

NumericTransform

Belirli bir Java fonksiyonunu seçilen sayısal özelliklere uygulayarak bir dönüşüm gerçekleştirir. Fonksiyonlar içerisinde double türünden değer alabilir ve double türünden değer döndürebilir. Örnek olarak `java.lang.Math` altındaki `sqrt()` metodu kullanılabilir. Birinci kısım işlevi uygulayan Java sınıfının adını ikinci kısım ise fonksiyonun adı olmalıdır.

Normalize

Veri kümelerindeki tüm sayısal değerleri 0 ile 1 arasında ölçeklendirir. Normalleştirilmiş değerler, kullanıcı tarafından sağlanan sabitlerle daha fazla ölçülebilir ve çevrilebilir.

Center

Veri kümelerindeki tüm sayısal değerleri sıfır ortalamaya dönüştürür.

Standardize

Veri kümelerindeki tüm sayısal değerleri sıfır ortalamaya dönüştürür. Aynı zamanda birim varyansı da verir.

RandomSubset

Çıktıya dahil etmek için nitelik değerlerinden rastgele seçerek bir alt kümeye oluşturur. Kullanıcı yüzdesel olarak belirteceği oranda alt kümeye oluşturulur. Girilen değer dahil kabul edilir.

CartesianProduct

İki veya daha fazla nominal özellik arasında Dekart çarpımı sonucu yeni bir nitelik üretir. Yeni niteliğin adı orijinal nitelik adlarının birleşmesinden oluşur.

PartitionedMultiFilter

Bir dizi filtreyi, giriş veri kümelerindeki karşılık gelen nitelik aralığı kümese uygulayan özel bir filtredir. Kullanıcı, her bir filtreyi verir ve yapılandırarak çalışacakları nitelik aralığını tanımlar. Aralığın herhangi biriyle kapsanmayan nitelikleri silmek için bir seçenek vardır. Sadece niteliklerde çalışan filtrelerle izin verilir. Tek tek filtrelerin çıktısı yeni bir veri kümelerine uygulanır.

Reorder

Verilerdeki özelliklerin sırasını değiştirir. Yeni sıralama, nitelik indislerinin bir listesini sağlayarak belirtilir. İndisleri atlayıp çoğaltarak, nitelikleri silmek veya bunların birkaç kopyasını çıkarmak mümkündür.

4.5.1.2. Değerleri Değiştirme

SwapValues

Bir nominal niteliğin iki sütunun yerlerini değiştirir. Değerleri sırası tamamen yüzeyseldir ve öğrenmeyi etkilemez. Ancak sınıf türündeki verilerde hata matrisinin düzenini değiştirir.

MergeTwoValues

İki nominal niteliğin değerlerini tek bir kategoride birleştirir. Yeni niteliğin adı iki orijinal nitelik adının birleşiminden oluşur.

MergeManyValues

Birden fazla nominal niteliğin değerlerini tek bir kategoride birleştirir. Yeni niteliğin adı kullanıcı tarafından belirtilebilir.

MergeInfrequentValues

Birkaç nominal değeri bir kategoride birleştirir. Ancak bu durumda işlem kullanıcı tarafından sağlanan bir minimum frekans tarafından kontrol edilir. Bu minimumdan daha az sayıda olan değerler, yeni kategori ile değiştirilir. Yeni sütun başlığı orijinal olanların birleşimidir. Birleştirilen adların çok uzun olması durumunda kullanıcı karma kodu baz alan bir kısaltmayı da seçebilir.

ReplaceMissingValues

Eksik değerler baş etmenin bir yolu da öğrenme algoritmasını başlatmadan önce onları global olarak değiştirmektir. Bu metot ile her eksik sayısal değer ortalaması ile değiştirilir, nominal olanlar ise en çok geçen değer ile değiştirilir. Eğer bir sınıf seçilmişse eksik değerler varsayılan bir değer ile değiştirilmmez.

ReplaceMissingWithUserConstant

Eğer bir sınıf seçilmişse eksik değerler varsayılan bir değer ile değiştirilebilir. Bu durumda bu filtre kullanıcının sabit bir değer belirtmesine izin verir. Bu sabit değer, sayısal, nominal ve tarih türündeki nitelikler için ayrı ayrı belirtilebilir.

ReplaceWithMissingValue

Kullanıcı tarafından belirtilen bir dizi nitelik için rastgele seçilen eksik değerler ile eksik olmayan değerlerin yerini değiştirir. Belirli bir değerin eksik bir değerle değiştirilme olasılığı bir seçenek olarak belirlenebilir.

NumericCleaner

Çok büyük, çok küçük veya belirtilen bir değere çok yakın değerleri varsayılan bir değer ile değiştirir. Her bir durum için çok büyük, çok küçük veya çok yakın tanımına uyan tolerans değeri belirtilir.

AddValues

Kullanıcı tarafından sağlanan bir listeden nominal bir nitelikte bulunmayan herhangi bir değer ekler.

ClassAssigner

Bir veri kümesine sınıf niteliği ayarlamak veya sınıftan çıkarmak için kullanılır. Kullanıcı yeni sınıf niteliğinin dizinini sağlar. Sıfır değeri mevcut bir sınıf niteliğini sınıftan çıkarır.

SortLabels

Seçilen nominal niteliklerin değerlerini sıralamak için kullanılır.

4.5.1.3. Dönüşürmeler

Discretize

Normal şekilde belirtilen bir sayısal niteliği eşit aralıklı veya eşit frekanslı olarak gruplandırmak için kullanılır. Grupların toplam sayısı, çapraz doğrulamalarдан (*folds*) birisinin kullanılma olasılığını en üst düzeye çıkararak otomatik belirlenebilir veya seçilebilir. Birden fazla değere sahip biri yerine birkaç ikili nitelik oluşturmak da mümkündür. Eşit frekanslı ayıklaşdırımda ise, aralık başına istenen sayıda örnek değiştirilebilir.

PKIDiscretize

Eşit frekanslı gruptama yaparak sayısal nitelikleri ayırtırır. Grupların toplam sayısı, eksik değerler hariç geri kalan değerlerin kareköküdür. PKIDiscretize ve Discretize sınıf nitelikleri varsayılan olarak gözardı eder.

MakeIndicator

Nominal bir niteliği Boolean türündeki niteliğe dönüştürür. Yeni veri kümesinde, belirli bir nitelik istenen değerler arasında ise 1, değilse 0 atanır. Varsayılan olarak sayısal bir değerdir.

NominalToBinary

Tüm nominal nitelikleri ikili sayısal niteliklere dönüştürür. k değeri olan bir nitelik için sınıf nominal ise, k ikili niteliklere dönüştürülür. Zaten ikili olan niteliklerde herhangi bir değişiklik yapılmaz.

NumericToBinary

Tüm sayısal nitelikleri nominal değerlere dönüştürür. Sınıf türü hariç tutulur. Sayısal değer 0 ise yeni niteliğin değeri de sıfır olur, eksik bir değer ise eksik, bunların dışında yeni değer 1 olur.

NumericToNominal

Sayısal değerlerin her birini nominal değerler listesine ekleyerek nominal veri tipine dönüştürür. Bu filtre CSV uzantılı dosyalar içe aktarıldıkten sonra kullanılabilir. Weka'nın CSV içe aktarma özelliği yapılandırılmadıkça değeri sayı olan değişkenleri sayısal veri tipi olarak tanımlar.

FirstOrder

Bir dizi sayısal nitelikten bir sonraki değerden bir önceki değeri çıkartarak (*yani farklarını alarak*) yeni bir sayısal nitelik oluşturur. Örneğin orijinal değerleri 0.1, 0.2, 0.3, 0.1, 0.3 olan bir değişkenden 0.1, 0.1, -0.2, 0.2 yeni değerlerini üretir.

KernelFilter

Verilen veri kümesini bir çekirdek matrisine dönüştürür. Parametre ayarlarından değiştirilmemiği sürece sınıf niteliği değişmeden kalır. Varsayılan olarak *Center*filtresi seçildir fakat istenirse başka filtre de seçilebilir.

PrincipalComponents

Temel bileşen analizi ve verilerin dönüştürülmesini gerçekleştirir. İlk olarak çoklu nominal nitelikler ikili veri türüne dönüştürülür ve eksik olan değerler ise ortalama ile değiştirilir. Veriler varsayılan olarak normalize edilir. Bileşenlerin sayısı kullanıcı tarafından belirlenen varyans oranına göre belirlenir fakat bileşenleri sayısı değer olarak da verilebilir.

Transpose

Örnekler niteliğe, nitelikler örneğe dönüşür. Orijinal verilerin ilk niteliği nominal ve string türünde ise bu veriler yeni niteliğin adı olur.

4.5.1.4. String Dönüşümü

StringToNominal

Bir string niteliğin belirsiz sayıda değeri vardır. Bu filtre string türündeki nitelikleri belirlenen bir sayıda nominal veri türüne dönüştürür. Test verilerinde görünecek verilerin aynı şekilde eğitim verilerinde yer aldığından emin olunmalıdır.

NominalToString

Nominal veri tiplerini string veri türüne dönüştürür.

StringToWordVector

String veri tipinde geçen kelimelerin frekanslarını gösteren sayısal bir nitelik üretir. Tüm kelimeleri kapsar. Farklı kelimeler farklı değerler ile belirtilmek istenirse kullanıcı tarafından bir önek eklenmelidir. Sıngleleştirme işlemini etkileyen birçok seçenek vardır. Kelimeler bitişik olabilir veya belirli bir sınırlayıcı karakter kümesi ile ayrılabilir. İkinci durumda bunlar kullanıcı tarafından sağlanan minimum ve maksimum uzaklık olan n-gramlara ayrılabilir veya *Stemming* algoritması ile işlenebilir. Frekans dikkate alınarak belirli aralıklarda olmayanlar dikkate alınmayabilir.

FixedDictionaryStringToWordVector

StringToWordVector filtresine benzer fakat ondaki gibi kelime sözlüğünü kendisi oluşturmaz. Kullanıcı tarafından sağlanan bir kelime sözlüğünü baz alır. Sözlük dosyasının formatı her satırda bir sözcüktür ve virgül ile ayrılmış sayısıdır.

ChangeDateFormat

Tarih niteliklerindeki formatı değiştirmek için kullanılır. Java'nın *SimpleDateFormat* sınıfındaki herhangi bir format belirtilebilir.

4.5.1.5. Zaman Serisi Verileri

TimeSeriesTranslate

Aktif bir örneğin değerini başka bir eşdeğer nitelikteki örnek değeri ile değiştirir.

TimeSeriesDelta

Aktif örnekteki nitelik değerlerini, aktif değerle diğer başka örnekteki değer arasındaki farkla değiştirir. Her iki durumda da zaman kayması değerinin bilinmediği örnekler kaldırılabilir veya eksik değerler kullanılabilir.

4.5.1.6. Niteliklerin Rastgele Yapılması

AddNoise

Nominal bir nitelik alır ve değerlerinin belirli bir yüzdesini değiştirir. Eksik değerler geri kalanlar ile birlikte muhafaza edilebilir veya değiştirilebilir.

Obfuscate

İlişkiyi, nitelik adlarını, nominal ve string veri türlerini tekrardan adlandırarak verileri anonimleştirir.

RandomProjection

Veri kümesini, birim uzunluktaki sütunlarla rastgele bir matris kullanarak daha düşük boyutlu bir alt alana yansıtır. Sınıf niteliği hariç tutulur.

4.5.2. Verilere Yönelik Gözetimsiz Filtreler

Weka'nın verilere yönelik filtreleri, belirli bir niteliğin veya nitelik kümesinin tüm değerlerinin yerine bir veri kümesindeki tüm örnekleri etkiler.

Randomize

Veri kümesindeki örneklerin sırasını rastgele hale getirir. Kullanıcı bu işlemde kullanılması için bir rastgele adım sayısı belirtebilir.

Resample

Verilerin alt kümesini oluşturmanın yollarından biridir. Değiştirerek veya değiştirmeyerek bir rastgele örnek kümesi oluşturur.

RemoveFolds

Verileri belirli bir çapraz doğrulama katına bölüp ve bunlardan yalnızca birine indirmek için kullanılır. Eğer rastgele adımı belirtilirse, alt küme oluşturulmadan önce veri kümesi karıştırılır.

ReservoirSample

Bir veri kümesinden değerleri değiştirmeden rastgele bir örnek küme oluşturmak için rezervuar örneklemme algoritmasını kullanır.

RemovePercentage

Belirli bir yüzdesel orandaki örnek yüzdesini siler.

RemoveRange

Belirli bir örnek sayı aralığını siler.

RemoveWithValues

Belirli bir değere sahip tüm nominal nitelikleri veya belirli bir eşliğin altındaki/üstündeki sayısal değerlerin tümünü silmek için kullanılır. Varsayılan olarak belirli bir nominal nitelik değerlerinden birini veya belirli bir eşliğin altındaki sayısal değerlerin tümünü siler. Ayrıca, eşleşen kriterler tersine çevrilebilir. Nitelik bilgisi normalde değişmeden kalır fakat bu değiştirilebilir. Bir nominal niteliğin istenen bir değeri tanımlarından silinebilir.

RemoveFrequentValues

Belirli bir nominal niteliğin en az ve en sık geçen değerlerini kaldırmak için kullanılabilir. Kullanıcı bu en az ve en sık değerleri belirtebilir.

SubsetByExpression

Kullanıcı tarafından sağlanan mantıksal ifade ile eşleşen tüm örnekleri seçer. İfade *AddExpression* ve *MathExpression* algoritmalarında kullanılan matematiksel fonksiyon ve işlevleri içerebilir.

RemoveMisclassified

Veri kümesine uygulanan sınıflandırma metodundaki aykırı değerleri silmek için kullanılır. Yanlış sınıflandırılmış örnekler siler. İşlem veriler tamamen silinene kadar tekrarlanır.

NonSparseToSparse

Verileri düzenli gösterimden aralıklı gösterime dönüştürür.

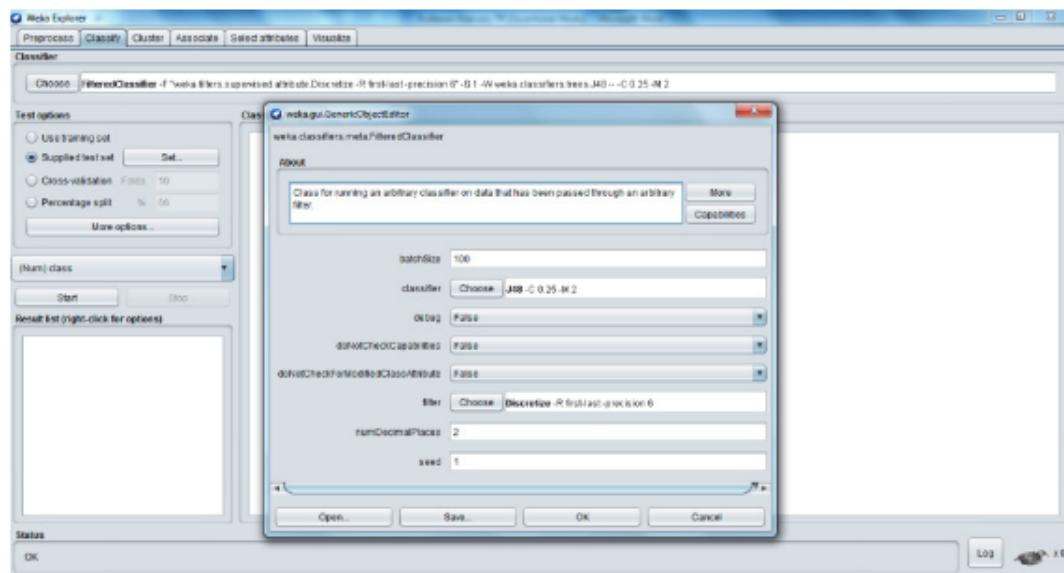
SparseToNonSparse

Verileri aralıklı gösterimden düzenli gösterime dönüştürür.

4.5.3. Gözetimli Filtreler

Gözetimli filtreler tipki gözetimsiz filtreler gibi Explorer ekranındaki önişlemeler sekmesinden erişilir. Fakat bu filtreler tam olarak bir önişleme olmadıklarından kullanırken dikkatli olunmalıdır. Test verilerinin tahmin edilen sınıf değerleri bilinmediği için bu gözetimli filtreler probleme neden olabilir.

Weka talep nedeniyle gözetimli filtreleri gözetimsiz filtreler gibi çalıştırır. Ancak bunlar sınıflandırma için kullanılacak ise farklı bir metodoloji benimsenmelidir. Bir meta öğrenme gibi davranışları. Böylelikle filtreleme modeli oluşturmak öğrenme sürecinin parçası haline dönüşür. Bu filtreler test verilerini eğitim verilerinden oluşturarak çalışır. Bu durum bazı gözetimsiz filtrelerde de kullanılabilir. Örneğin *StringToWordVector* filtresinde sözlük yalnızca eğitim verilerinden oluşur. Test verilerinde yeni olan kelimeler atılır. Gözetimli bir filtre ise bu şekilde kullanılmak istendiğinde Sınıflandırma (*Classify*) sekmesindeki meta öğrenmeler altında bulunan *FilteredClassifier* öğrenme metodu seçilirse burada hem sınıflandırma metodu hem de filtre algoritması seçilebilir. Aşağıdaki şekilde ilgili ekran gösterilmiştir.



Şekil 4.61. Meta Öğrenmeler Altındaki *FilteredClassifier* Öğrenme Metodu

Gözetimli nitelik filtreleri ve gözetimli veri filtreleri ayrı başlıklar altında aşağıda ele alınmaktadır.

4.5.3.1. Gözetimli Nitelik Filtreleri

Discretize

Gözetimli ayrıklaştırmanın MDL yöntemini kullanır. Bir dizi özellik belirtilebilir veya ayrık nitelik ikili olmaya zorlanabilir. Tahmin edilecek sınıf nominal olmalıdır.

NominalToBinary

Bu filtrenin çoklu nominal nitelikleri ikili boolean türüne dönüştüren bir sürümü vardır. Bu versiyonda dönüşüm sınıfın nominal mı yoksa sayısal mı olduğuna bağlıdır. Nominal ise kullanıcı tarafından belirtilen bir değere sahip ise 1 değilse 0 üretir. Sınıf sayısal ise ağaç algoritması altında yer alan M5P tarafından kullanılan yöntem uygulanır. Her iki durumda da sınıfın kendisi dönüştürülmez.

ClassOrder

Sınıf değerinin sırasını değiştirir. Kullanıcı yeni sıralamanın rastgele mi, A'dan Z'ye mi yoksa Z'den A'ya şeklinde olacağını belirler. Bu filtre *FilteredClassifier* meta öğrenmesi ile birlikte kullanılmamalıdır.

AddClassification

Verilere veri kümesinde eğitilebilen veya seri hale getirilmiş nesne olarak dosyadan yüklenebilen belirli bir sınıflandırıcının tahminlerini ekler. Tahmin edilen sınıf değerini, tahmin edilen olasılık dağılımını (*eğer sınıf nominal ise*) veya yanlış sınıflanmış örnekleri (*sayısal değerler için gerçek değer ile tahmin edilen değer arasındaki farkı*) gösteren yeni nitelikler (*süntunlar*) ekleyebilir.

AttributeSelection

Verilerdeki nitelik sayısını azaltmak için belirli bir nitelik seçim teknğini uygular. Kullanıcı nitelik seçim yöntemini uygulamak için hangi niteliğin veya alt değerlendircisinin arama stratejisiyle birlikte kullanılacağını ve birleştirileceğini seçebilir.

ClassConditionalProbabilities

Nominal ve sayısal niteliklerin değerlerini sınıf koşullu olasılık tahminlerine dönüştürür. Bayes sınıflandırıcının hesaplaması ile aynıdır. Nominal değerleri birlestiren gözetimsiz filtreler gibi, bu filtre de çok sayıda farklı değere sahip nominal niteliklerle uğraşırken yardımcı olabilir.

4.5.3.2. Verilere Yönerek Gözetimli Filtreler

Resample

Alt örnekte sınıf dağılımını sürdürmesi dışında, gözetimsiz örnek filtresi gibidir. Alternatif olarak, sınıf dağılımını tekdüze bir şekilde yönlendirmek için yapılandırılabilir. Örneklemme değiştirilmeden veya değiştirilerek yapılabılır.

SpreadSubsample

Rastgele bir alt örnek üretir. Fakat en nadir sınıf ile en yaygın sınıf arasındaki frekans farkı kontrol edilebilir. Ayrıca belirlenen bir sayı kadar herhangi bir sınıfındaki örnek sayısını sınırlanabilir.

Smote

Verileri örnekleyen ve sınıf dağılımını değiştiren bir filtredir. *SpreadSubsample* gibi verideki azınlık ve çoğunluk sınıfları arasındaki göreceli sıklığı ayarlamak için de kullanılabilir. Fakat çoğunluk sınıflarının örneklemesinin yapılmamasını sağlar. Azınlık sınıfını k-en yakın komşu yaklaşımını kullanarak yapay örnekler ile oluşturur.

StratifiedRemoveFolds

Gözetimsiz örnek filtresi *RemoveFolds* gibi, veri kümesi için belirtilen katmanlanmış dışındaki bir çapraz doğrulamayı çıkarır.

4.6. Öğrenme Algoritmaları

Sınıflandırma (*Classify*) sekmesinde Seç (*Choose*) butonu kullanılarak öğrenme algoritması seçilir. Seçilen algoritmanın adı ve aldığı parametre değerleri – (tire) işaretleri ile ayrılarak yan tarafında görüntülenir. Parametre değerlerinde değişiklik yapılmak istendiğinde bu satırın tıklanması ve açılan ekranda istenilen değişikliklerin yapılması yeterlidir. Weka'da sınıflandırıcılar;

- Bayes
- Ağaç (*Tree*)
- Kurallar (*Rules*)
- Meta Öğrenmeler (*Meta*)

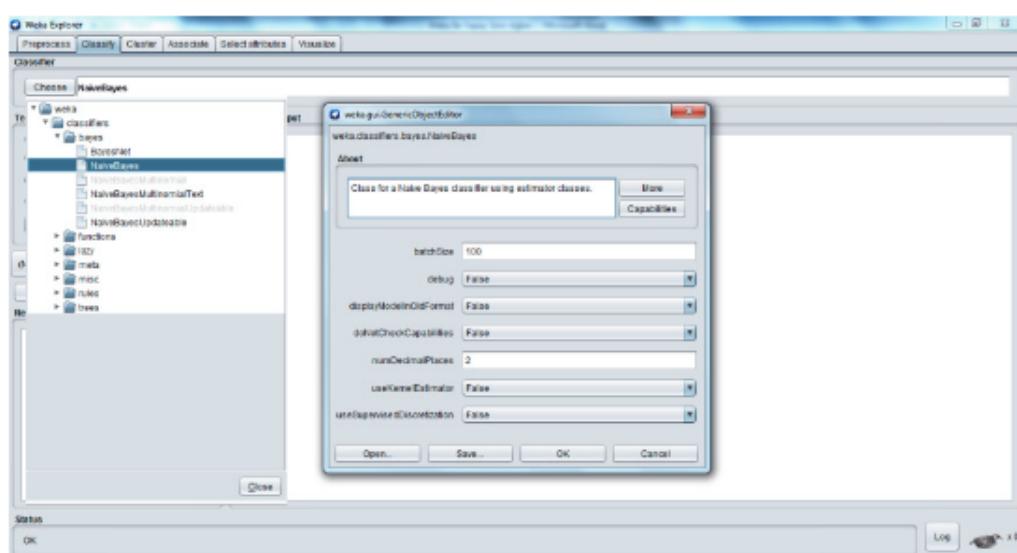
- İşlevler (Functions)
- Tembel sınıflandırıcılar (*Lazy*)
- Çeşitli kategoriler (*Misc*) olarak gruplandırılmıştır.

Burada kısaca parametre değerleriyle birlikte anlatılacaktır. Buradakinden daha fazla bilgi edinmek istenirse Explorer penceresinden herhangi birini seçip fare üzerinde iken çıkan panelden detay elde edilebilir. Ayrıca her bir parametre değeri için ise parametre ekranı açık iken Dahası (*More*) butonu ile bilgi edinilebilir.

4.6.1. Bayes Sınıflandırıcıları

NaiveBayes

Olasılıksal temel bayes sınıflandırıcıyı uygular. *NaiveBayesSimple* sayısal nitelikleri modellemek için normal dağılımı kullanır. Normallik varsayımlı çok yanlışsa performansı arttıran çekirdek yoğunluğu tahmin ediciler kullanılabilir. Gözetimli ayrıklama kullanılarak sayısal nitelikleri de işleyebilir.



Şekil 4.62. NaiveBayes Algoritması ve Parametreleri

Aşağıdaki şekil weather.arff dosyasına *NaiveBayes* algoritması uygulanması sonrası elde edilen sonuçları göstermektedir. Toplam 14 sınıfından 8 tanesini başarılı bir şekilde yerleştirerek %57.14 oranında doğruluk elde etmiştir.

Adını Matematikçi Thomas Bayes'den alan bir sınıflandırma algoritmasıdır. NaiveBayes sınıflandırması olasılık ilkelerine göre tanımlanmış bir dizi hesaplama ile, sisteme sunulan verilerin sınıfını tespit etmeyi amaçlar.

```

==== Run information ====
Scheme: weka.classifiers.bayes.NaiveBayes
Relation: weather.symbolic
Instances: 14
Attributes: 5 {outlook, temperature, humidity, windy, play}
Test mode: 10-fold cross-validation

==== Classifier model (full training set) ====
Naive Bayes Classifier
    Class
Attribute   yes   no
              (0.63) (0.38)
-----
outlook
sunny      3.0  4.0
overcast   5.0  1.0
rainy      4.0  3.0
[total]    12.0 8.0

temperature
hot        3.0  3.0
mild       5.0  3.0
cool       4.0  2.0
[total]    12.0 8.0

humidity
high       4.0  5.0
normal     7.0  2.0
[total]    11.0 7.0

windy
TRUE      4.0  4.0
FALSE     7.0  3.0
[total]   11.0 7.0

Time taken to build model: 0 seconds

==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances      8      57.1429 %
Kappa statistic                   -0.0244
Mean absolute error               0.4374
Root mean squared error          0.4916
Relative absolute error           91.8631 %
Root relative squared error     99.6492 %
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  0,800  0,636  0,778  0,700  -0,026  0,578  0,697  yes
0,200  0,222  0,333  0,200  0,250  -0,026  0,578  0,557  no
Weighted Avg. 0,571  0,594  0,528  0,571  0,539  -0,026  0,578  0,647

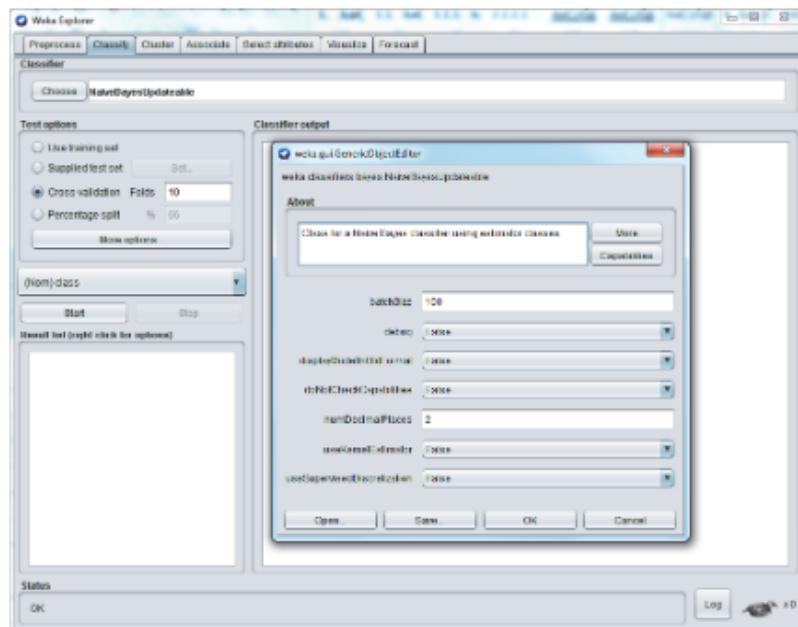
==== Confusion Matrix ====
a b <-- classified as
7 2 | a = yes
4 1 | b = no

```

Şekil 4.63. Weather.arff NaiveBayes Algoritması Uygulanması

NaiveBayesUpdateable

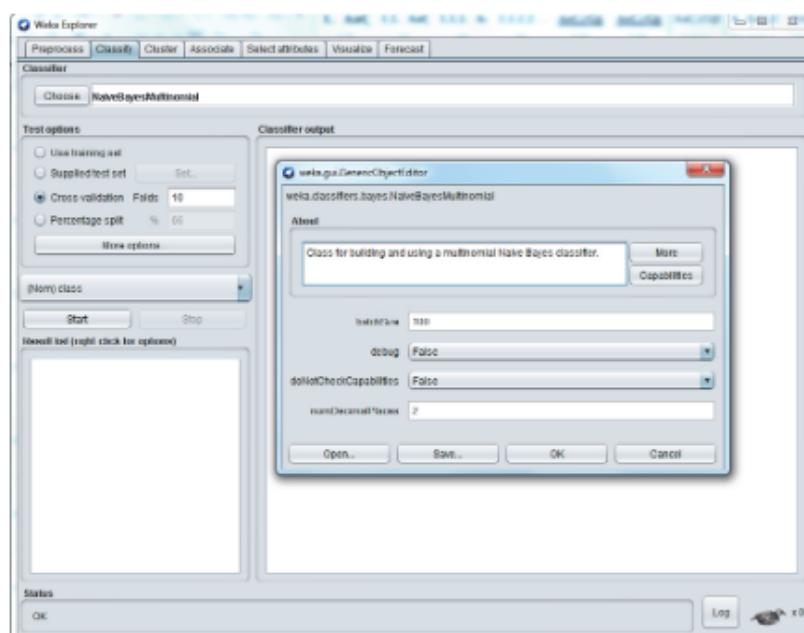
NaiveBayes'e göre her seferinde bir örneği işleyen artımlı bir sürümdür. Bir çekirdek tahmincisi kullanabilir, ancak ayriklaştırma yapamaz.



Şekil 4.64. NaiveBayesUpdateable Algoritması ve Parametreleri

NaiveBayesMultinomial

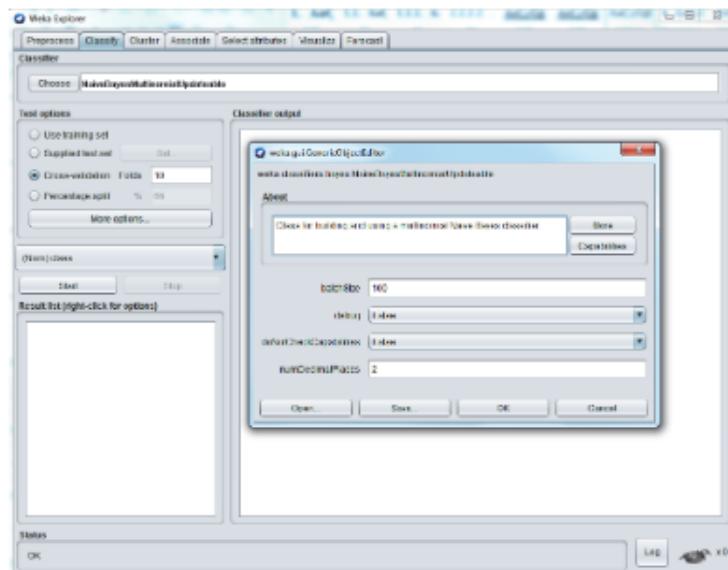
Çok kanallı Bayes sınıflandırıcısını uygular.



Şekil 4.65. NaiveBayesMultiNominal Algoritması ve Parametreleri

NaiveBayesMultinomialUpdateable

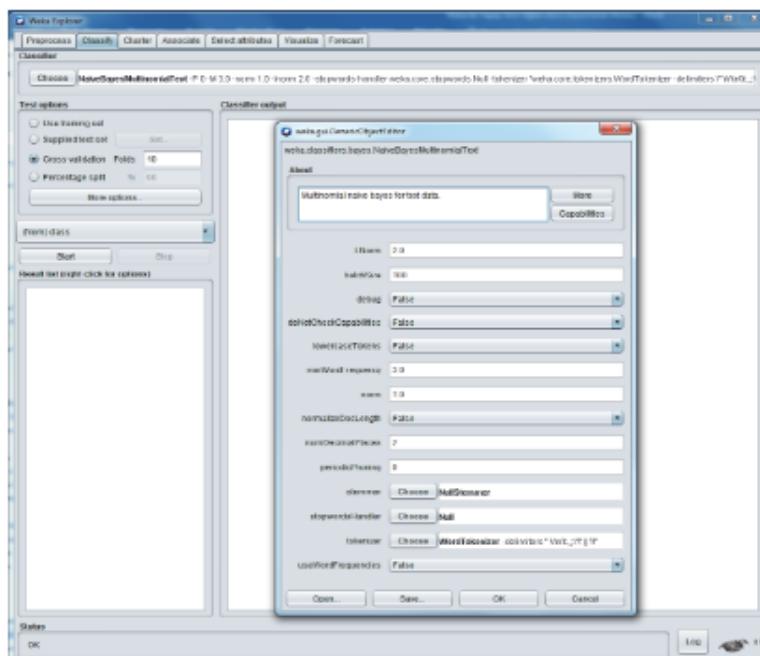
Aşamalı bir sürümürdür.



Şekil 4.66. NaiveBayesMultinomialUpdateable Algoritması ve Parametreleri

NaiveBayesMultinomialText

Dize öznitelikleri üzerinde doğrudan çalışabilen, artımlı çok terimli NaiveBayes'in bir sürümürdür. Net olarak, TF / IDF dönüşümü hariç, StringToWordVector filtresinin işlevini etkin bir şekilde gerçekleştirir.

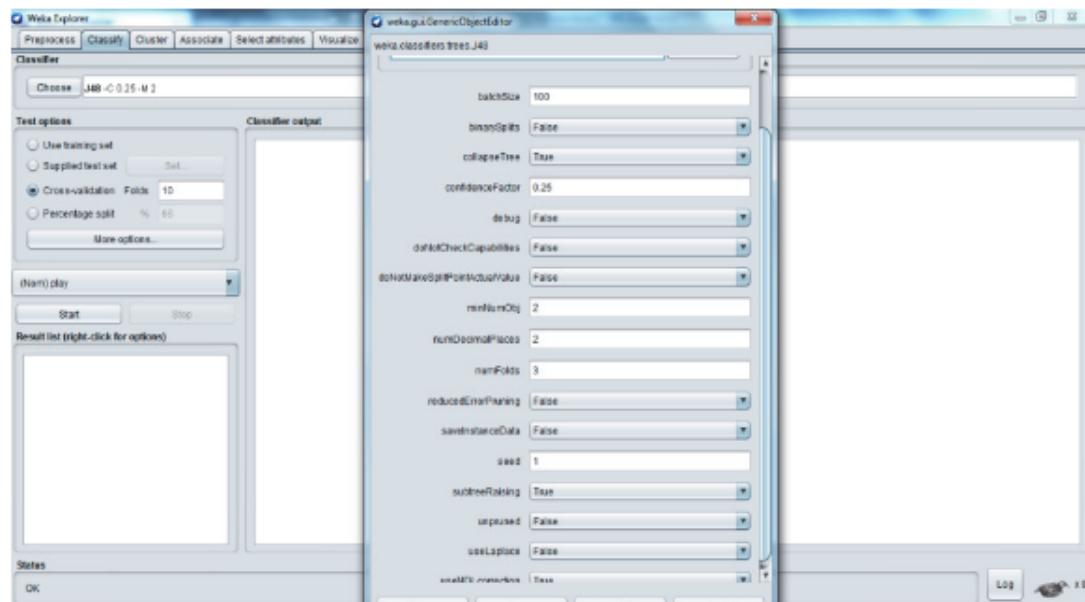


Şekil 4.67. NaiveBayesMultinomialText Algoritması ve Parametreleri

4.6.2. Ağaç Algoritmaları (Tree)

J48

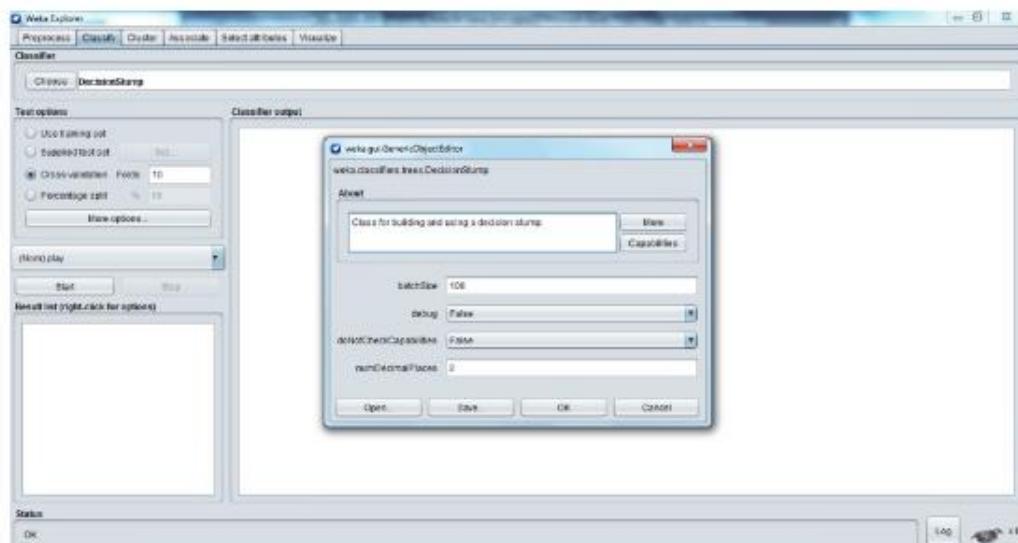
Bu algoritma ile çoklu dalları olan bir ağaç yerine ikili bir ağaç yapısı oluşturulabilir. Budama için kullanıcı tarafından değiştirilebilen güven eşiği (*confidenceFactor*) varsayılan olarak 0.25'tir. Varsayılanı 2 olan bir yaprakta izin verilen minimum örnek sayısı (*minNumObj*) ayarlanabilir. J48 algoritması C4.5 algoritmasının geliştirilmiş halidir. Eski olan C4.5 budaması yerine azaltılmış hata budaması (*reducedErrorPruning*) seçilebilir. Varsayılan olarak 3 değerindeki *NumFolds* parametresi budama kümesinin boyutunu belirler. Veri bu parça sayısına eşit olarak bölünür ve sonuncu parça budama için kullanılır. Ağaç görselleştirmesi yapılacak olması durumunda *saveInstanceData* parametresi aktif edilmesi durumunda orijinal veriler ağaç dalları üzerine tıklanması ile görünür olacaktır. Önbellek gereksinimlerini azaltmak için varsayılan olarak bu özellik kapalıdır. *subtreeRaising* parametresini kapatarak daha verimli bir algoritma elde edilebilir. Bu şekilde algoritma budanmış olanların yerine budanmamış ağaç yapısını kullanmaya zorlanır. Tahmin edilen olasılıkları yumusatmak için *useLaplace* parametresi aktif edilebilir.



Şekil 4.68. J48 Algoritması ve Parametreleri

DecisionStump

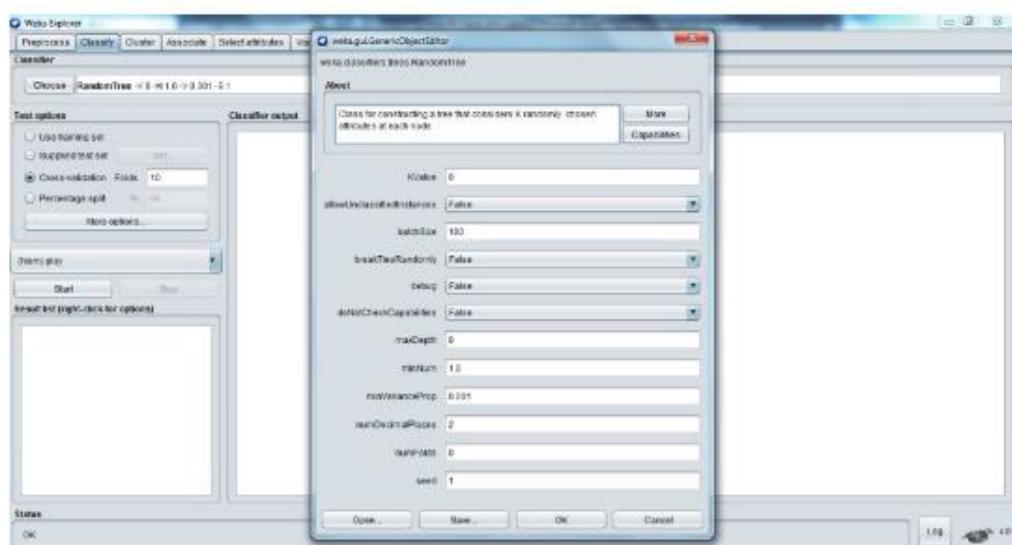
Kategorik (*nominal*) veya sayısal bir sınıfı sahip veri kümeleri için tek seviyeli bir ikili karar ağacı oluşturur. Eksik değerleri kökten üçüncü bir dal uzatarak genişletir ve ayırrır.



Şekil 4.69. DecisionStump Algoritması ve Parametreleri

RandomTree

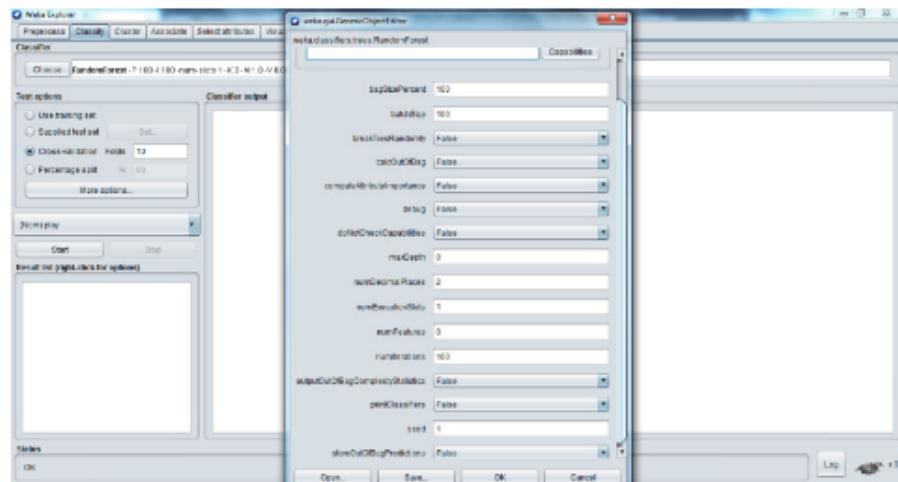
Her düğümde belirli bir sayıdaki rastgele özelliklerini dikkate alır ve buda yapılmaz.



Şekil 4.70. RandomTree Algoritması ve Parametreleri

RandomForest

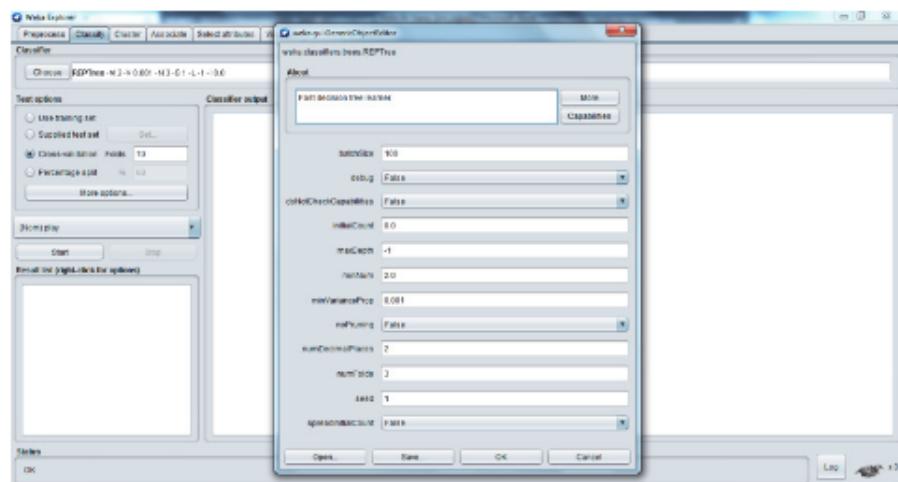
Rastgele ağaçların topluluklarını paketleyerek rastgele ormanlar kurar.



Şekil 4.71. RandomForest Algoritması ve Parametreleri

REPTree

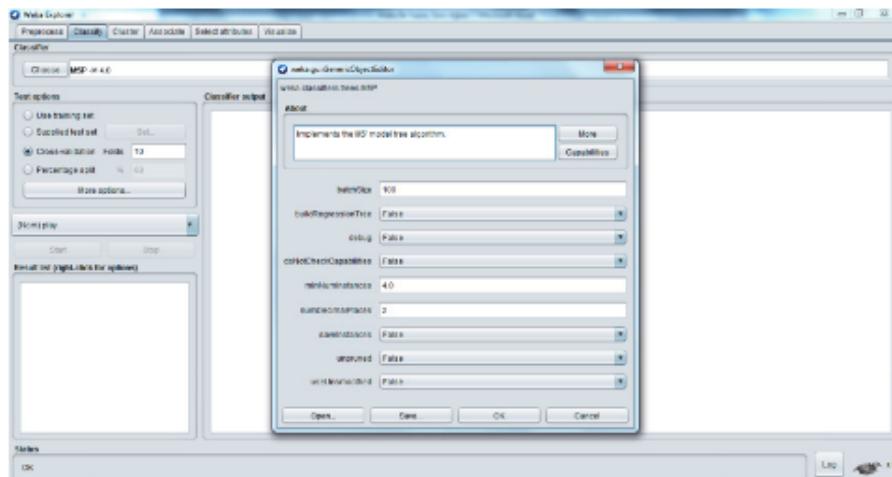
Bilgi kazanımı/varyans azaltma kullanarak bir karar veya regresyon ağaçları oluşturur ve azaltılmış hata budamasını kullanarak bunu eritir. Hız için optimize edilmiştir. Yalnızca bir kez sayısal değerleri sıralar. C4.5'in yaptığı gibi, örnekleri parçalara ayırarak eksik değerleri ele alır. Yaprak başına minimum örnek sayısı (*minNum*), ağaçları hızlandırırken faydalı olan maksimum ağaç derinliği (*maxDept*), yalnızca sayısal sınıflara uygun olan bölünme için eğitim seti varyansının minimum oranı (*minVarianceProp*) ve budama kat sayısı kullanıcı tarafından ayarlanabilir.



Şekil 4.72. RepTree Algoritması ve Parametreleri

M5P

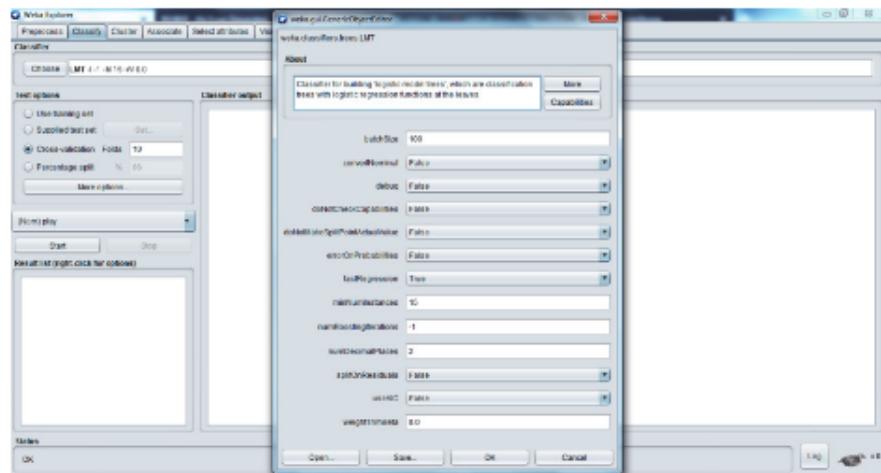
M5 öğrenme modelini kullanan ağaç algoritmasıdır.



Şekil 4.73. M5P Algoritması ve Parametreleri

LMT

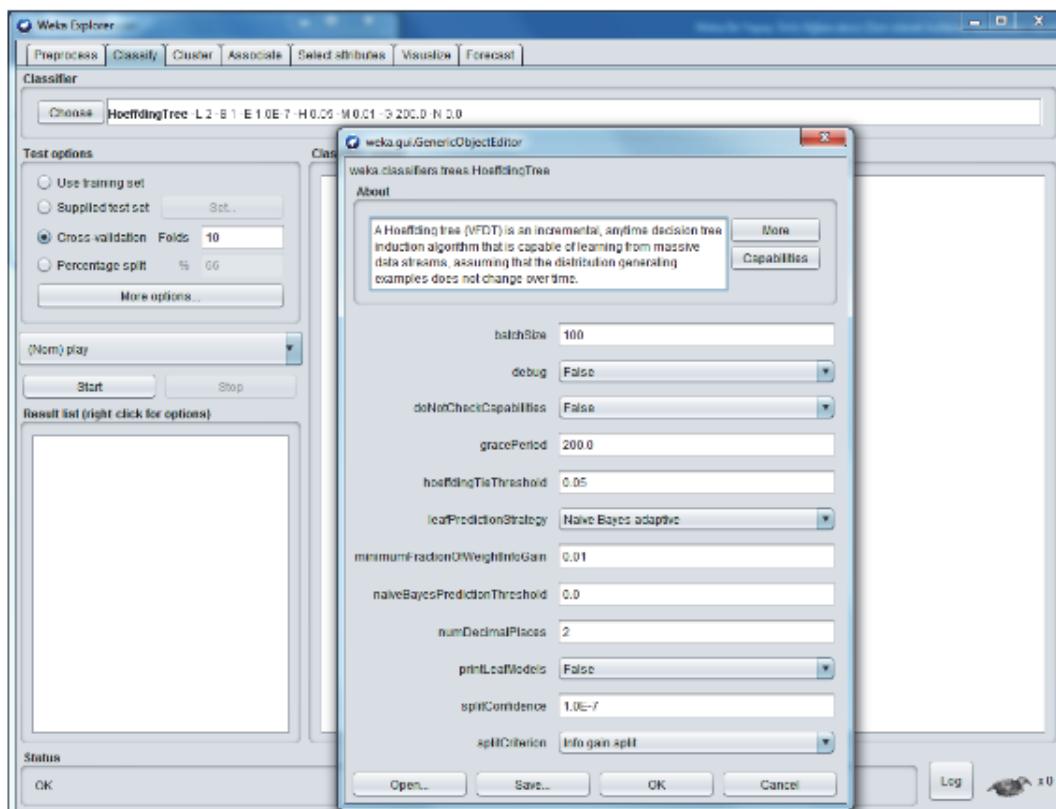
Lojistik model ağaçları oluşturur. İkili ve çoklu sınıfa sahip değerler, sayısal, kategorik ve eksik değerlere sahip nitelikler ile çalışabilir. *LogitBoost* algoritmasını kullanarak bir düğümdeki lojistik regresyon fonksiyonlarını ayarlarken, kaç tane yinelemenin (*iterasyon*) bir kez çalışacağını belirlemek için çapraz doğrulama kullanır ve her düğümde çapraz doğrulama yerine ağaç boyunca aynı sayıyı kullanır. Kullanıcı tarafından ayarlanan bu sezgisellik, doğruluğu çok az etkileyerek çalışma süresini önemli ölçüde iyileştirir. LMT, kompakt bir ağaç yapısı üretmek için minimum maliyetli karmaşıklık budama mekanizmasını kullanır.



Şekil 4.74. LMT Algoritması ve Parametreleri

HoeffdingTree

Artan karar ağacı algoritmasının bir uygulamasıdır. Bilgi kazancı veya Gini endeksine dayalı bölmeler oluşturma seçenekleri sunar. Ağacın yapraklarındaki tahminler, ya çoğunluk sınıfı ya da NaiveBayes modelleri tarafından yapılabılır.

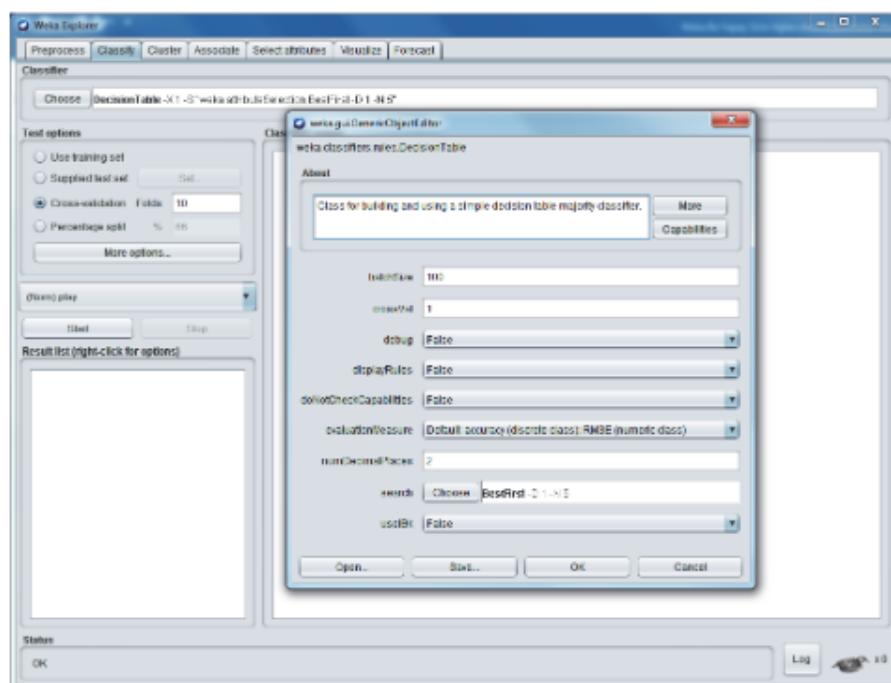


Şekil 4.75. HoeffdingTree Algoritması ve Parametreleri

4.6.3. Kurallar (Rules)

DecisionTable

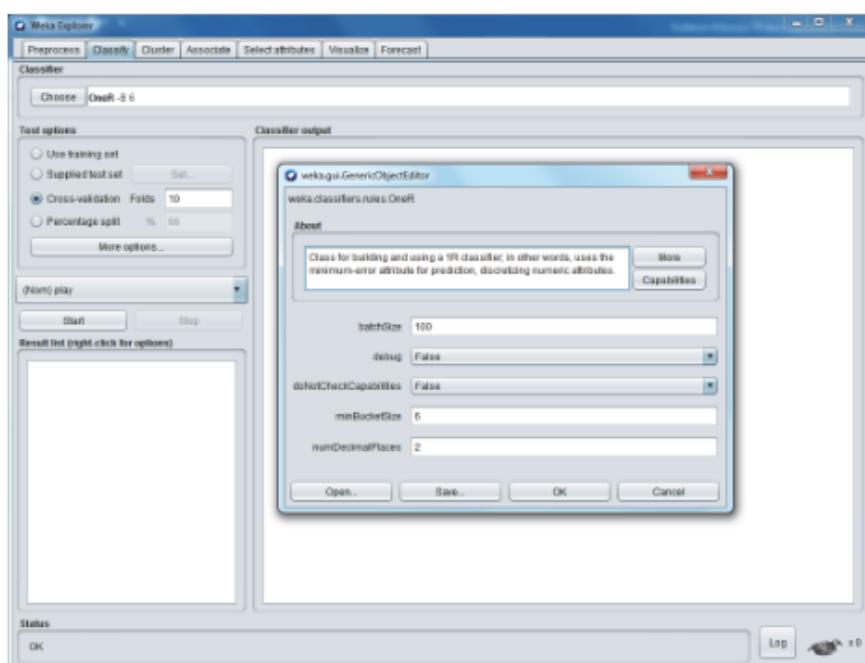
Bir karar tablosu sınıflandırıcısı oluşturur. En iyi ilk arama özelliğini kullanarak özellik alt kümelerini değerlendirir ve değerlendirme için çapraz doğrulamayı kullanabilir. Bir seçenek, aynı özellik kümese dayanarak, tablonun global çoğunluğu yerine, bir karar tablosu girdisiyle kapsamayan her bir örneğin için sınıfı belirlemek için en yakın komşu yöntemini kullanır.



Şekil 4.76. DecisionTable Algoritması ve Parametreleri

OneR

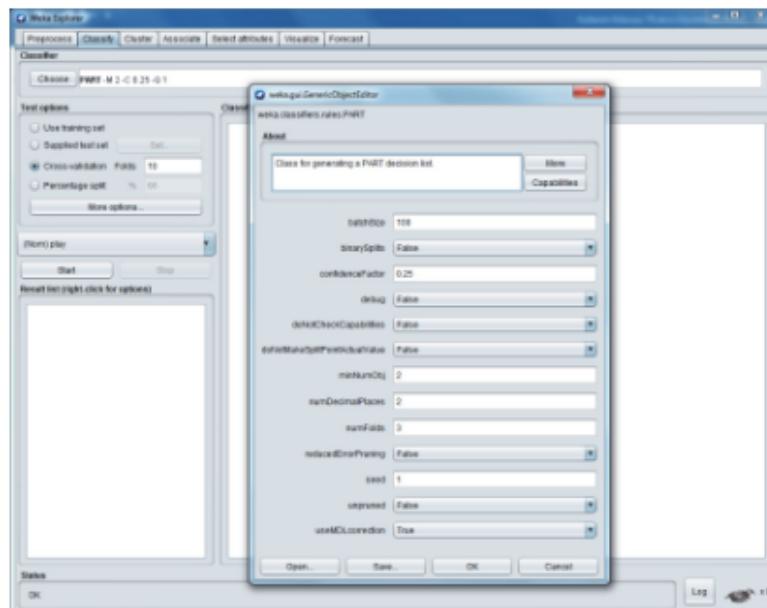
Ayrıklaştırma için minimum kova boyutu ile bir parametrelî 1R sınıflandırıcıdır.



Şekil 4.77. OneR Algoritması ve Parametreleri

PART

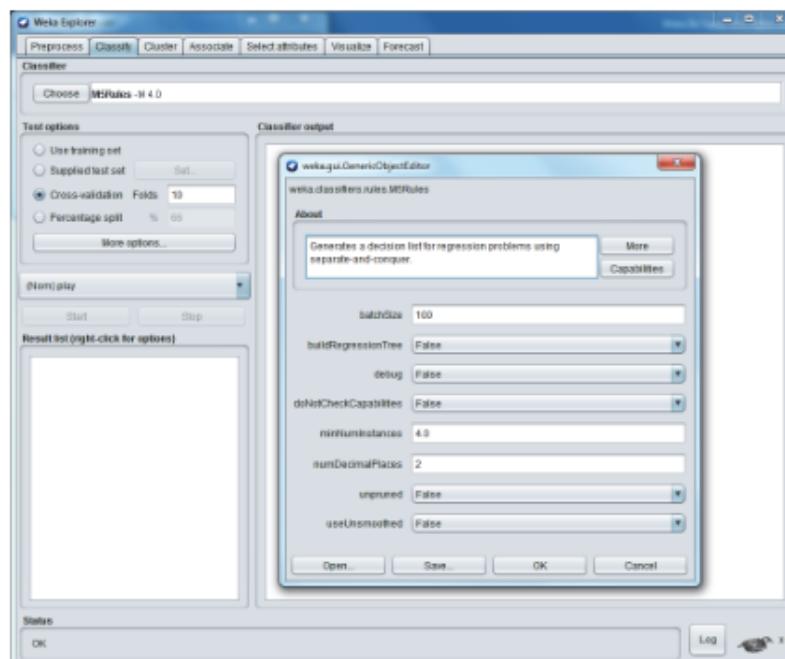
Kısmi karar ağaçlarından kurallar alır. C4.5'in sezgisel özelliklerini kullanarak ağaç J48 ile aynı kullanıcı tanımlı parametrelerle oluşturur.



Şekil 4.78. PART Algoritması ve Parametreleri

M5Rules

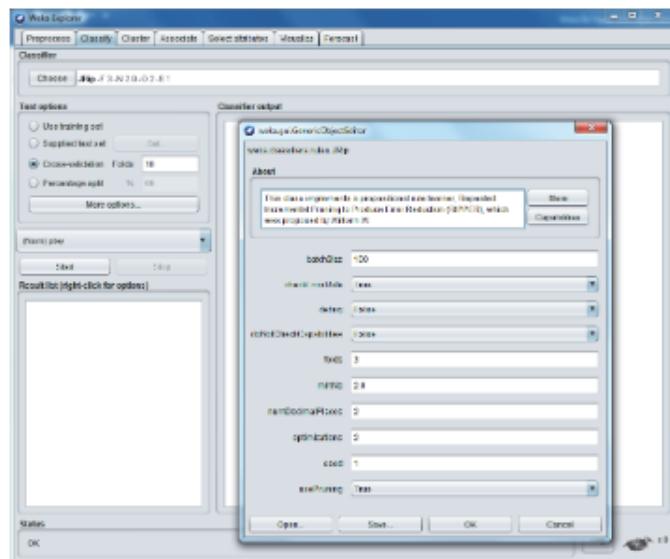
M5'i kullanılarak oluşturulan model ağaçlardan regresyon kurallarını alır.



Şekil 4.79. M5Rules Algoritması ve Parametreleri

JRip

JRip kural kümesinin sezgisel global optimizasyonu dahil olmak üzere Ripper algoritmasını uygular.

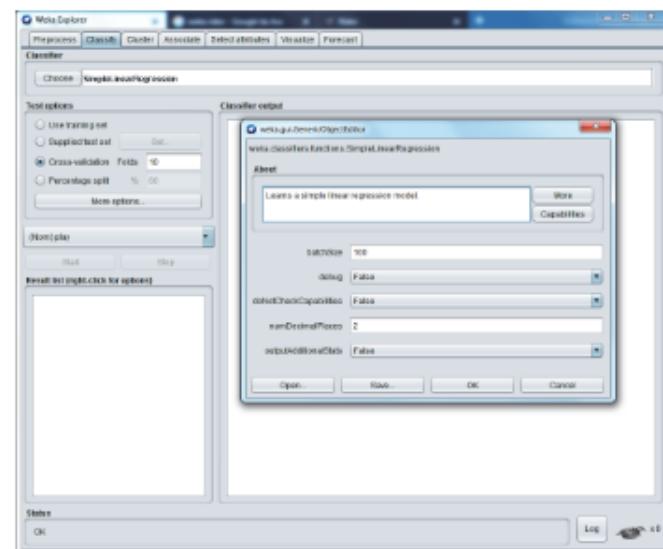


Şekil 4.80. JRip Algoritması ve Parametreleri

4.6.4. Fonksiyonlar

SimpleLinearRegression

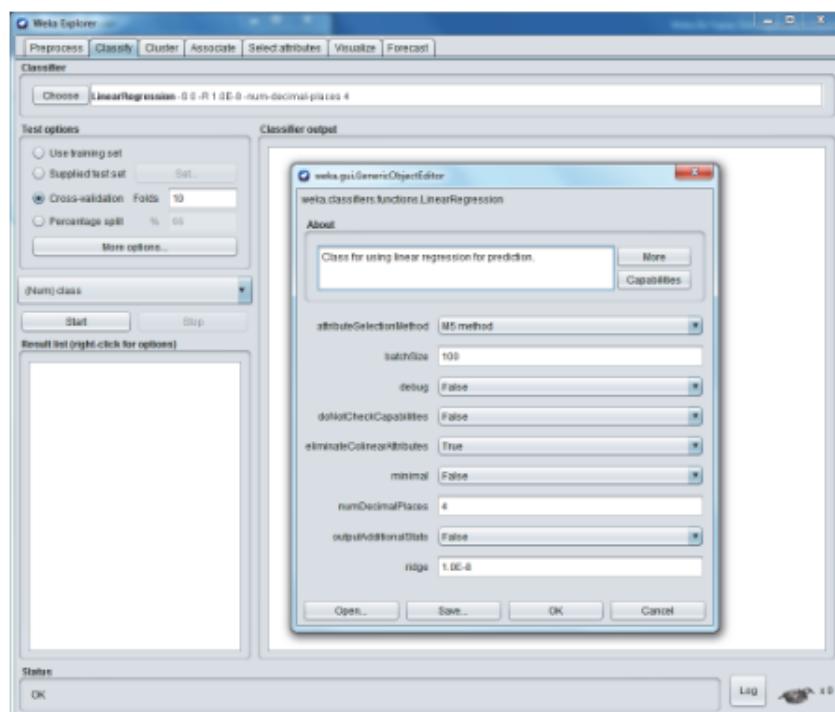
Tek bir niteliğe dayanan bir doğrusal regresyon modeli ile öğrenme gerçekleşir ve en küçük karesel hatayı veren öğeyi seçer. Sayısal olmayan nitelikler ile çalışmamaz.



Şekil 4.81. SimpleLinearRegression Algoritması ve Parametreleri

LinearRegression

En az kareler, nitelik seçimi de dahil olmak üzere çoklu doğrusal regresyon gerçekleştirir. Ya arkadan eleme kullanarak ya da tüm niteliklerden tam bir model oluşturarak ve bir durdurma ölçüyü elde edilene kadar standartlaştırılmış katsayılarının sırasını düşürerek terimleri birer birer düşürür. Her iki yöntem de AIC sonlandırma ölçütünün bir sürümünü kullanır. Nitelik seçimi kapatılabilir. Teknik olarak, *LinearRegression* tepe regresyonu uygular.

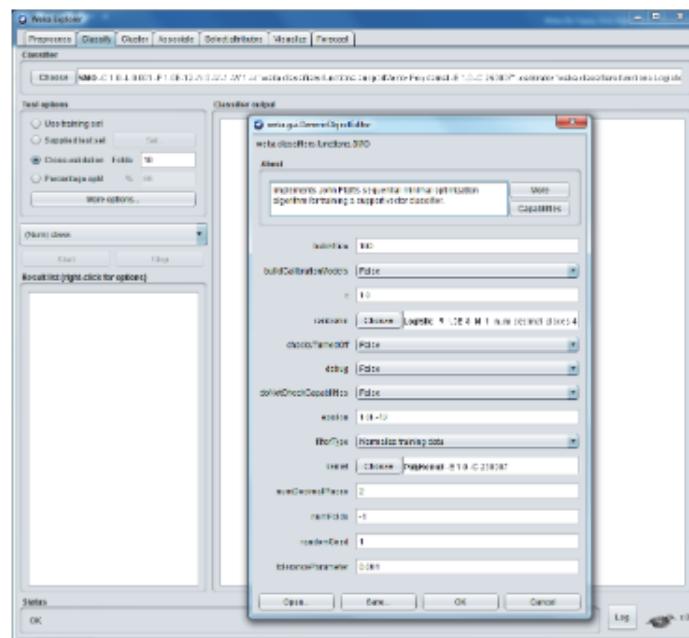


Şekil 4.82. LinearRegression Algoritması ve Parametreleri

SMO

Bir polinom veya Gauss çekirdekleri gibi çekirdek fonksiyonlarını kullanarak, bir destek vektör sınıflandırıcısı eğitimi için sıralı minimal optimizasyon algoritmasını uygular. Eksik değerler, global olarak değiştirilir, nominal nitelikler ikili dosyalara dönüştürülür ve nitelikler varsayılan olarak normalleştirilir. Çıktıdaki katsayıların normalleştirilmiş verilere dayandığı dikkate alınmalıdır. Normalleştirme kapatılabilir veya giriş sıfır ve ortalama varyansa standartlaştırılabilir. Çoklu sınıf problemleri için çift yönlü sınıflandırma kullanılır. Olasılık tahminlerini elde etmek için destek vektör makinesi çıkışına lojistik regresyon modelleri yerleştirilebilir. Çoklu sınıf durumunda, tahmin edilen olasılıklar, ikili eşleştirme kullanılarak birleştirile-

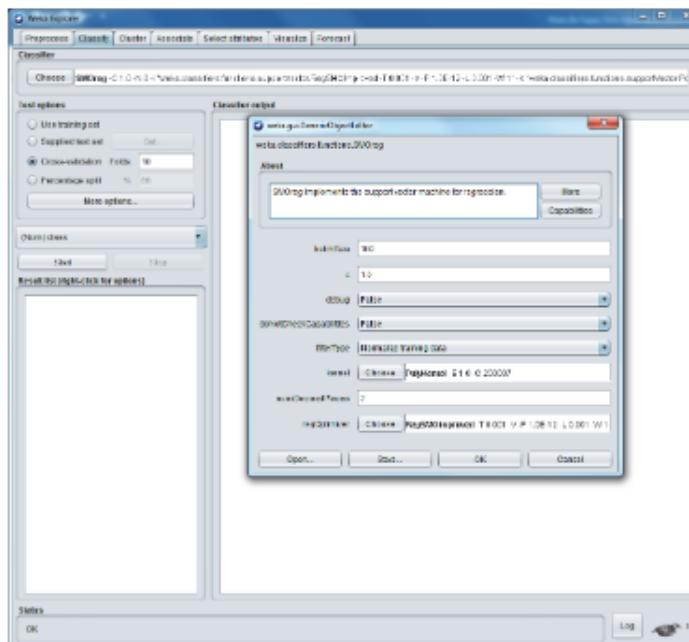
cektir. Az sayıda örneklerle çalışırken, daha hızlı işlem için normalleştirme kapatılabilir.



Şekil 4.83. SMO Algoritması ve Parametreleri

SMOreg

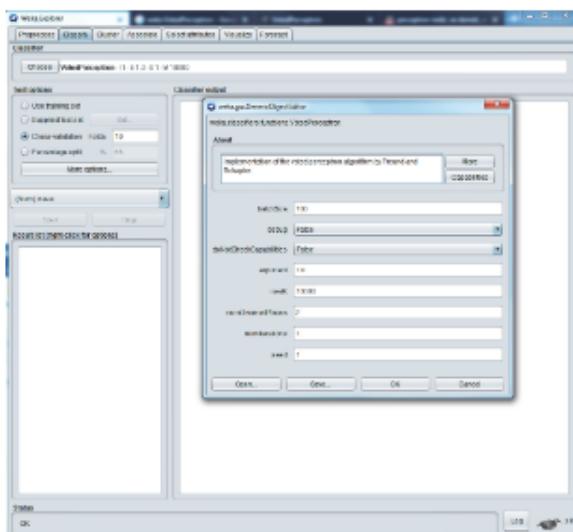
Bir destek vektör regresyon modelini öğrenmek için sıralı minimal optimizasyon algoritmasını uygular.



Şekil 4.84. SMOreg Algoritması ve Parametreleri

VotedPerceptron

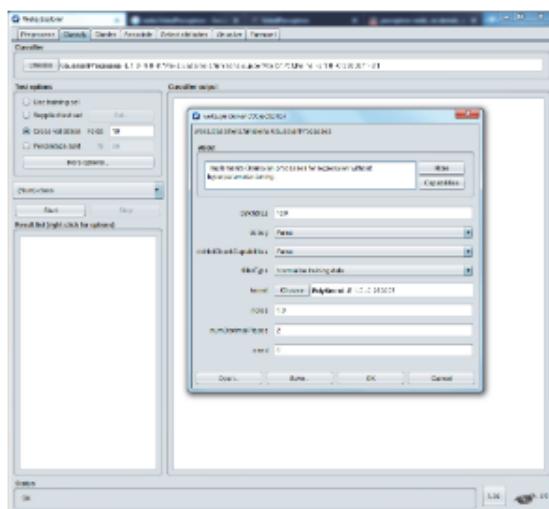
Oy verilmiş geri yayılımlı algoritmasını uygular. Global tüm eksik değerleri değiştirir ve nominal nitelikleri ikili olanlara dönüştürür.



Şekil 4.85. VotedPerceptron Algoritması ve Parametreleri

GaussianProcesses

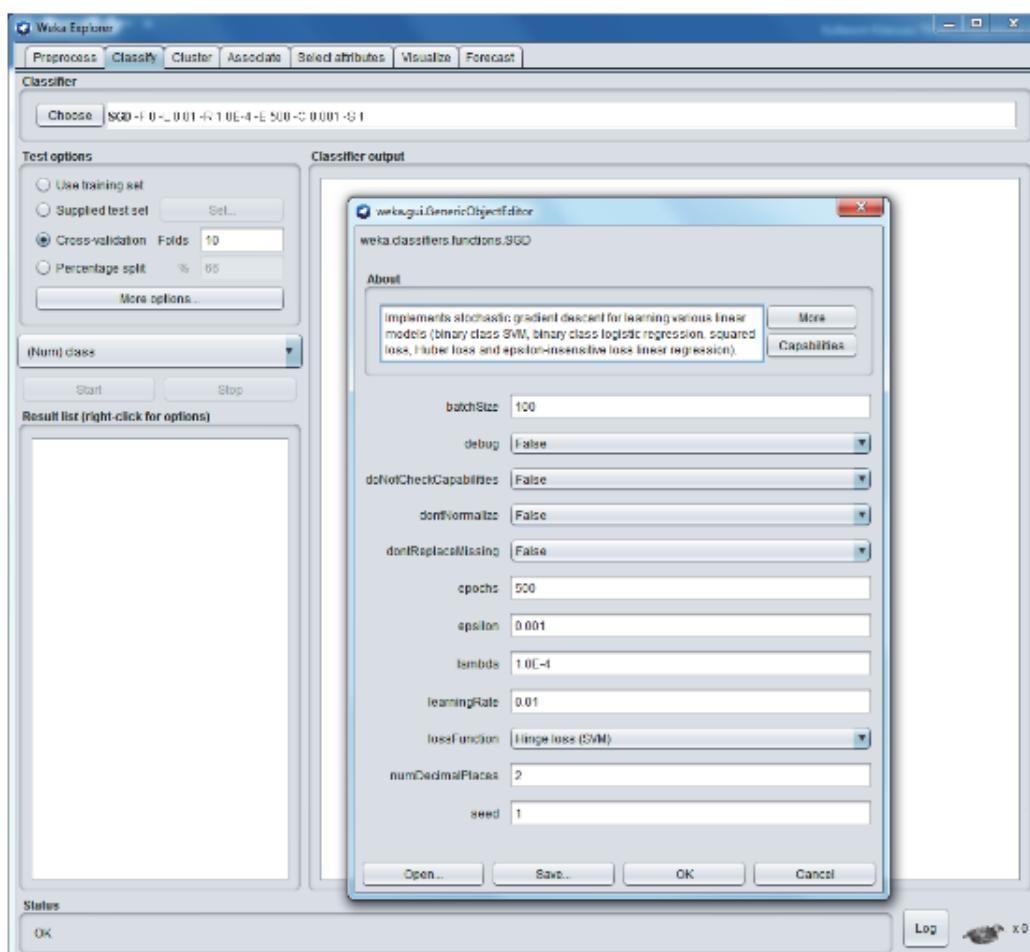
Lineer olmayan regresyon için Bayes Gauss işlem tekniğini uygular. Kullanıcılar, çekirdeğin fonksiyonunu, uyumun yakınlığını kontrol etmek için bir “gürültü” regülasyon parametresi ile birlikte belirtebilirler. Regresyonun öğrenilmesinden önce eğitim verilerinin normalize edilmesini veya standardize edilmesini seçebilirler. Nokta tahminleri için, bu yöntem, çekirdek tepe regresyonuna eşdeğerdir.



Şekil 4.86. GaussianProcesses Algoritması ve Parametreleri

SGD

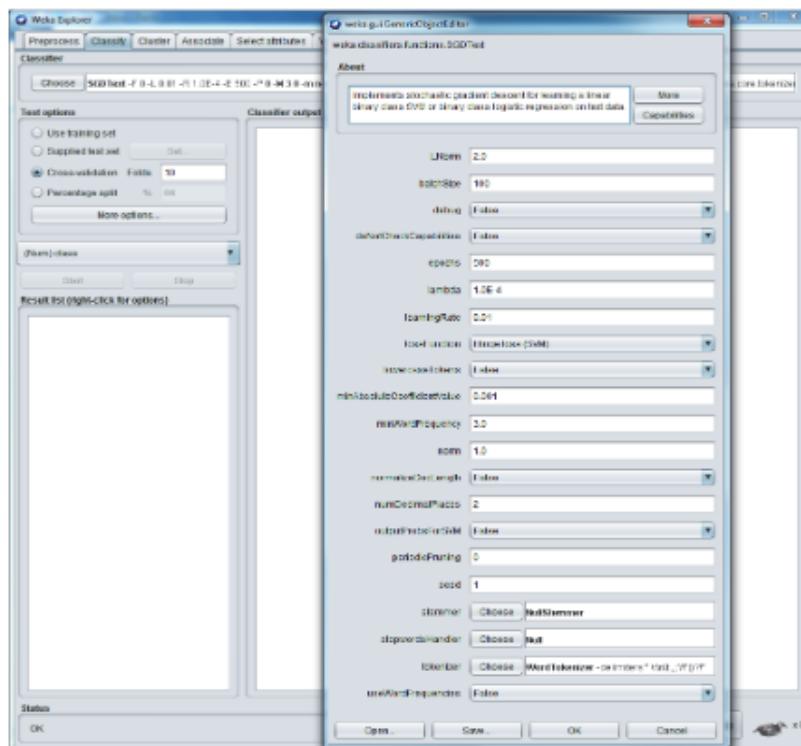
Doğrusal modellerin stokastik gradyan iniş öğrenimini uygular. Modeller, toplu veya artımlı şekilde öğrenilebilir. Seçilen kayıp fonksiyonu, öğrenilen modelin türünü belirler. Mevcut kayıp fonksiyonları arasında menteşe kaybı (doğrusal destek vektör sınıflandırıcı), log kaybı (lojistik regresyon), karesel kayıp (en küçük kareler doğrusal regresyon), epsilon duyarsız kayıp (destek vektör regresyonu) ve Huber kaybı (sağlam regresyon) sayılabilir.



Şekil 4.87. SGD Algoritması ve Parametreleri

SGDText

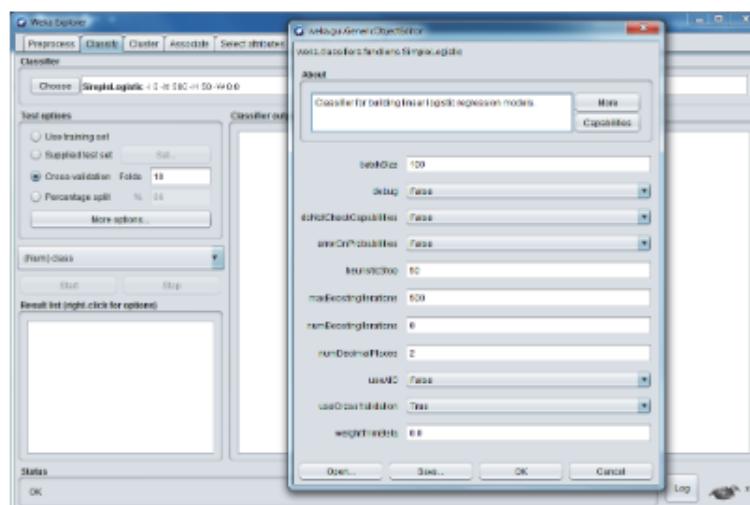
NaiveBayesMultinomialText gibi SGDText, metin sınıflandırması için tasarlanmış ve dize öznitelikleri üzerinde doğrudan çalışabilen bir sürümüdür. SGDText sadece menteşe ve log kaybı ile sınırlıdır.



Şekil 4.88. SGDText Algoritması ve Parametreleri

SimpleLogistic

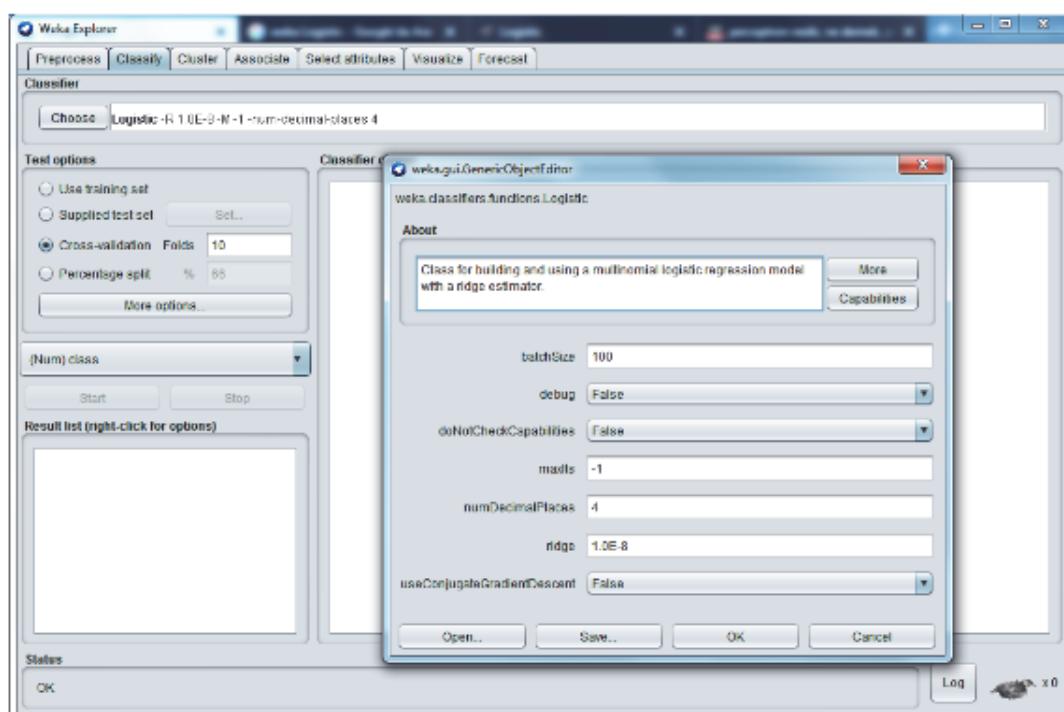
LogitBoost'u temel öğrenenler olarak basit regresyon fonksiyonları ile birleştirerek ve çapraz doğrulama kullanarak kaç tane yineleme gerçekleştirileceğini belirleyerek, otomatik özellik seçimini destekleyen lojistik regresyon modellerini oluşturur. SimpleLogistic, tek bir düğüm içeren bir dejenere lojistik model ağaçları oluşturur ve LMT için daha önce verilen seçenekleri destekler.



Şekil 4.89. SimpleLogistic Algoritması ve Parametreleri

Logistic

Bir tepe değer tahmin edicisi ile kategorik lojistik regresyon modelini kullanarak sınıfları oluşturur. Regresyon katsayıları tablosunun altında, her bir girdi niteliği ve sınıf değeri için olasılık oranının bir tahminini veren ikinci bir tablo bulunmaktadır. Belirli bir nitelik için, bu değer, diğer niteliklerin değerleri sabit tutulduğunda sınıf üzerindeki etkisinin bir belirtisini verir.

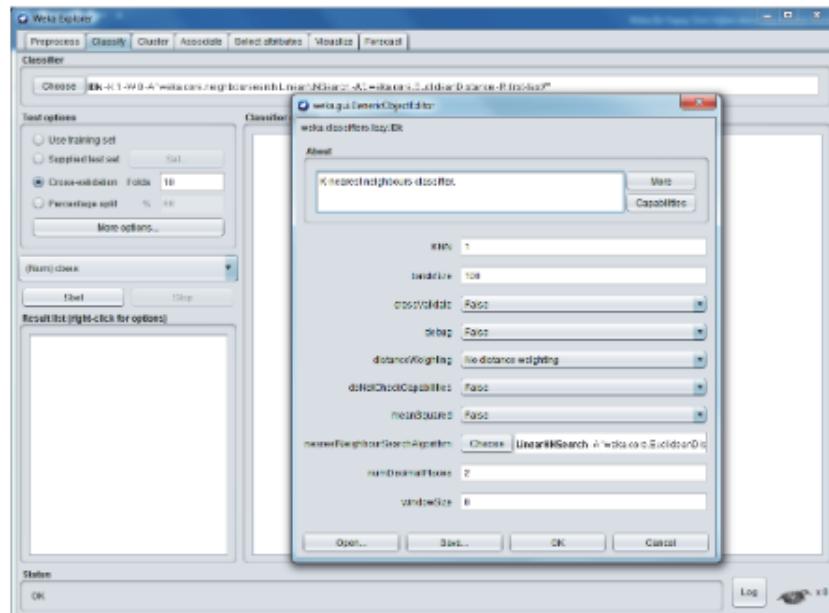


Şekil 4.90. Logistic Algoritması ve Parametreleri

4.6.5. Tembel Algoritmalar (Lazy)

IBk

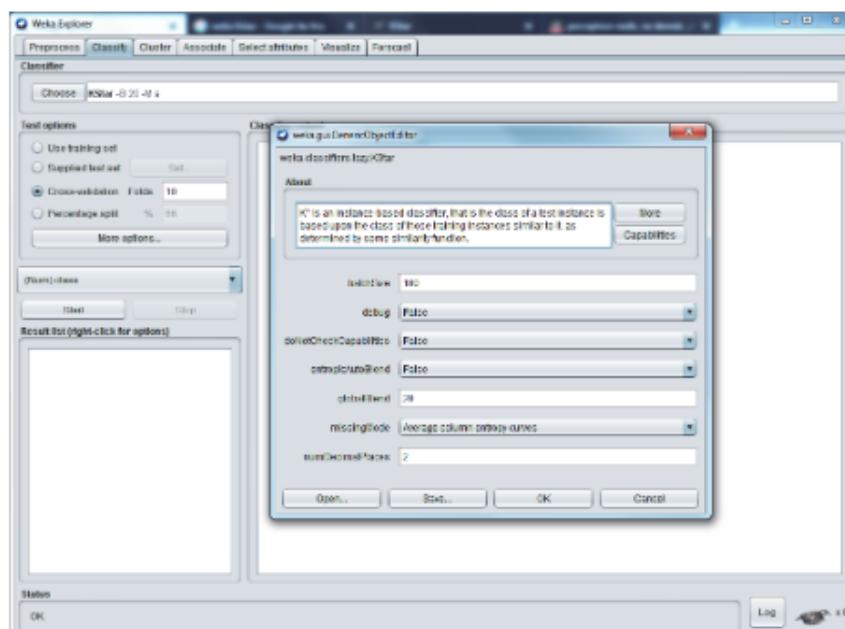
Tembel öğrenme algoritmaları eğitim örneklerini saklarlar ve sınıflandırma zamanına kadar gerçek bir çalışma yapmazlar. En basit tembel öğrenme algoritması IBk olarak adlandırılan k-en yakın komşu sınıflandırıcısıdır. En yakın komşu bulma görevini hızlandırmak için çeşitli farklı arama algoritmaları kullanabilir. Varsayılan olarak doğrusal arama kullanılır fakat KDTree, BallTree, CoverTree gibi diğer seçenekler de kullanılabilir.



Şekil 4.91. IBk Algoritması ve Parametreleri

KStar

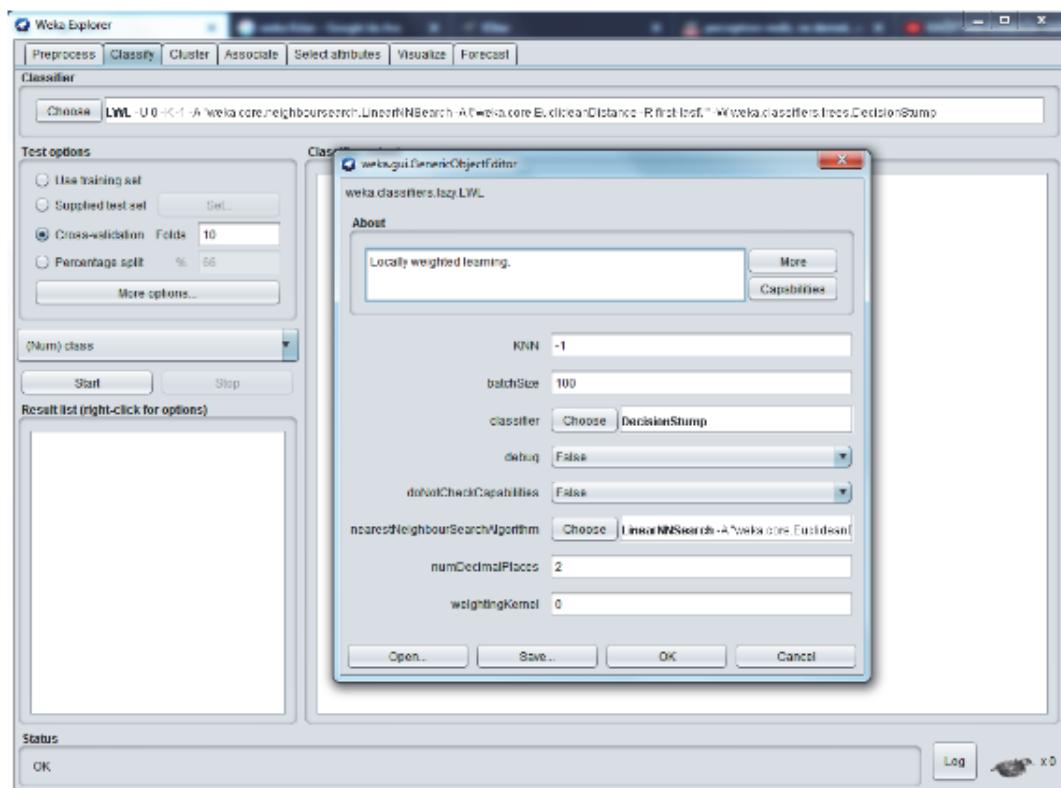
Verileri temel alan bir sınıflandırıcıdır. Yani test verilerinin sınıfı, benzerlik işlevinin belirlediği gibi, ona benzer eğitim verilerinin sınıfını temel alır. Entropi tabanlı bir mesafe fonksiyonu kullandığı için diğer veri tabanlı öğrenme algoritmalarından farklıdır.



Şekil 4.92. KStar Algoritması ve Parametreleri

LWL

Yerel ağırlıklı öğrenme için genel bir algoritmadır. Verilere dayalı yöntem kullanarak ağırlıklar atar ve ağırlıklı verilerden bir sınıflandırıcı oluşturur. Sınıflandırıcı, *LWL*'nin nesne editöründe seçilir: iyi bir seçim NaiveBayes veya sınıflandırma problemleri için lojistik regresyon, regresyon problemleri için doğrusal regresyondur. Nitelikleri normalleştirme varsayılan olarak açıklıktır.

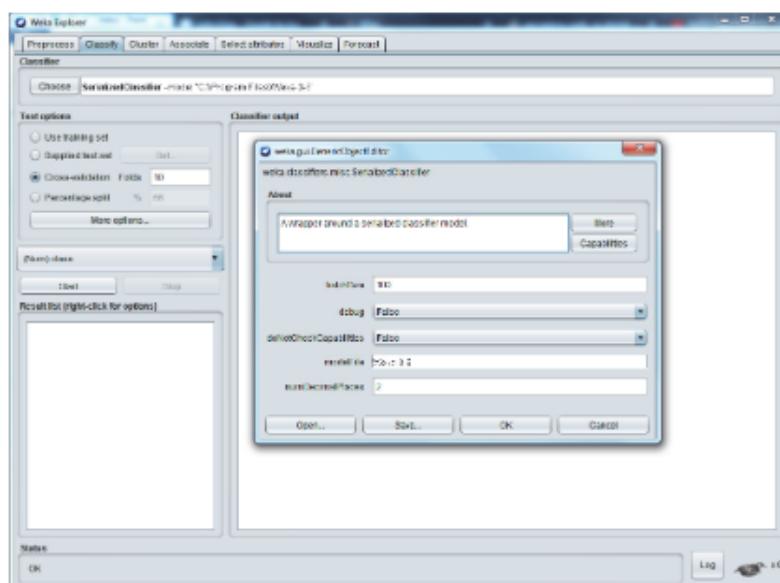


Şekil 4.93. LWL Algoritması ve Parametreleri

4.6.6. Çeşitli Sınıflandırıcılar (Misc)

SerializedClassifier

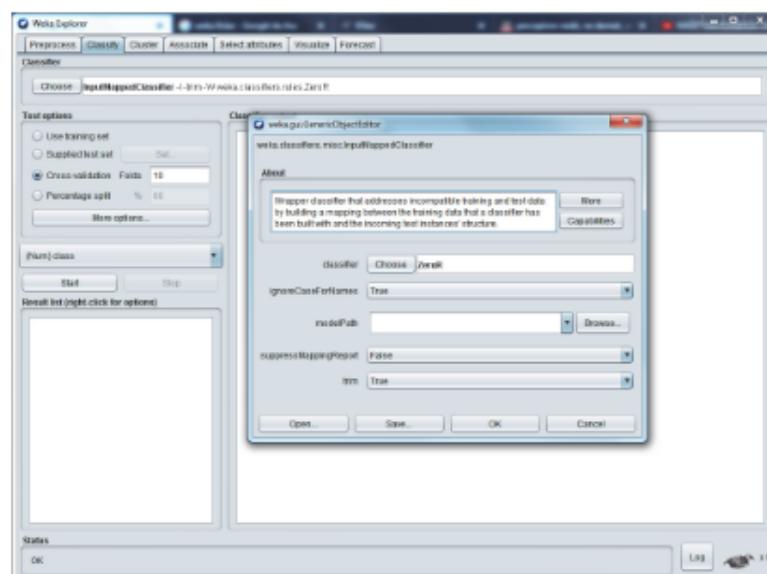
Bir dosyaya serileştirilmiş bir model yükler ve bunu tahmin için kullanır. Yeni bir eğitim veri seti sağlamak hiçbir etkiye sahip değildir çünkü bir statik modeli kapsülemeğtedir. Benzer şekilde, SerializedClassifier kullanılarak çapraz doğrulama gerçekleştirilmesi çok az anlam ifade eder.



Şekil 4.94. SerializedClassifier Algoritması ve Parametreleri

InputMappedClassifier

Bir temel sınıflandırıcıyı (ya da bir dosyaya serileştirilmiş olan modeli) sarar ve gelen test verilerinde bulunan özellikler ile model eğitildiğinde görülen özellikler arasında bir eşleme oluşturur. Test verilerinde bulunan, ancak eğitim verilerinde bulunmayan nitelikler için değerler basitçe göz ardı edilir. Eğitim verilerinde olan, ancak test verilerinde bulunmayan nitelikler, eksik değerleri alır. Benzer şekilde, eksik değerler eğitim verilerinde olmayan yeni nominal değerler için kullanılır.



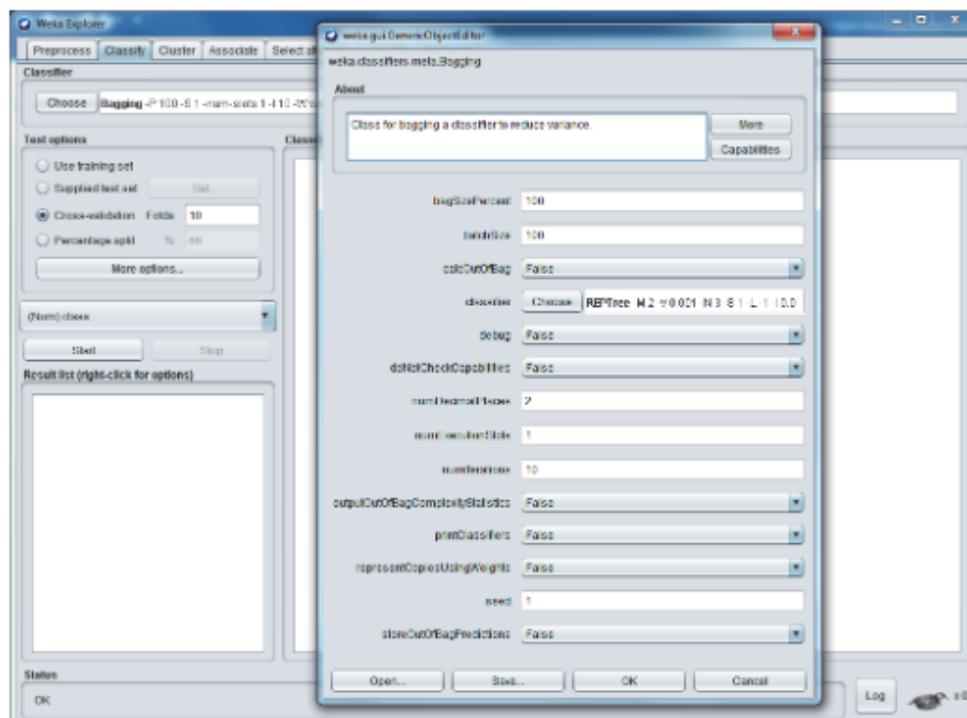
Şekil 4.95. InputMappedClassifier Algoritması ve Parametreleri

4.6.7. Meta Öğrenme Algoritmaları

Meta öğrenme algoritmaları sınıflandırıcılar alır ve onları daha güçlü öğrenenlere dönüştürür. Bir parametre, temel sınıflandırıcıyı/sınıflandırıcıları belirtir; diğerleri, torbalama, hızlandırma ve rastgele sayı üreteci için bir başlangıç tohumu gibi yinelemeli şemalar için yineleme sayısını belirtir. Filtrenin kendi parametreleri, yalnızca verileri test etmek için gözetimli bir filtrenin uygulanması için uygun yol olan eğitim verilerine dayanmaktadır.

Bagging

Eğitim veri setlerinin küçük boyutları ile çalışır. Özgün eğitim seti N adet alt kümelere bölünmüştür. Bu alt kümelerin her biri eğitim seti olarak kullanılır. Her bir alt küme aynı zamanda bir sınıflandırıcı oluşturur. Bu sınıflandırıcıları bir birleşik sınıflandırıcı bileşir [48]. Bu nedenle *Bagging* (torbalama) ismi verilmiştir.

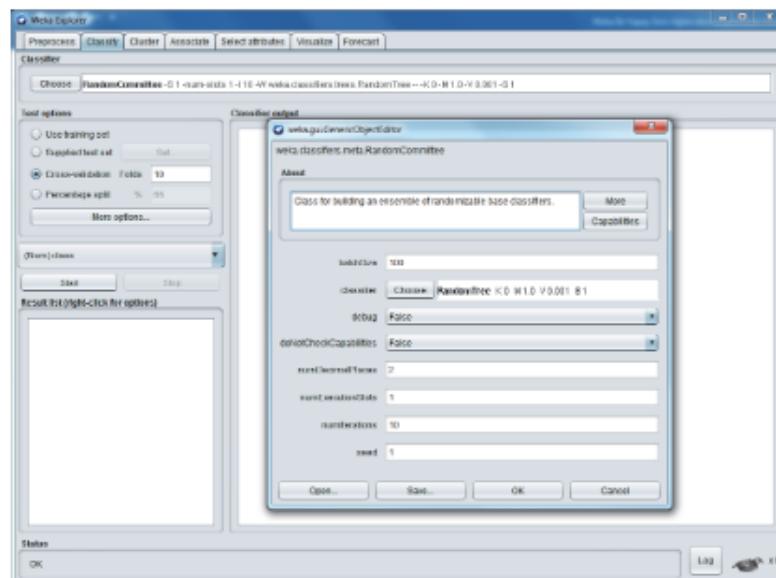


Şekil 4.96. Baggin Algoritması ve Parametreleri

RandomCommittee

Daha basit bir algoritmadır. Bir temel sınıflandırıcılar topluluğu oluşturur ve tahminlerini değerlendirir. Her biri aynı verilere dayanır ancak farklı

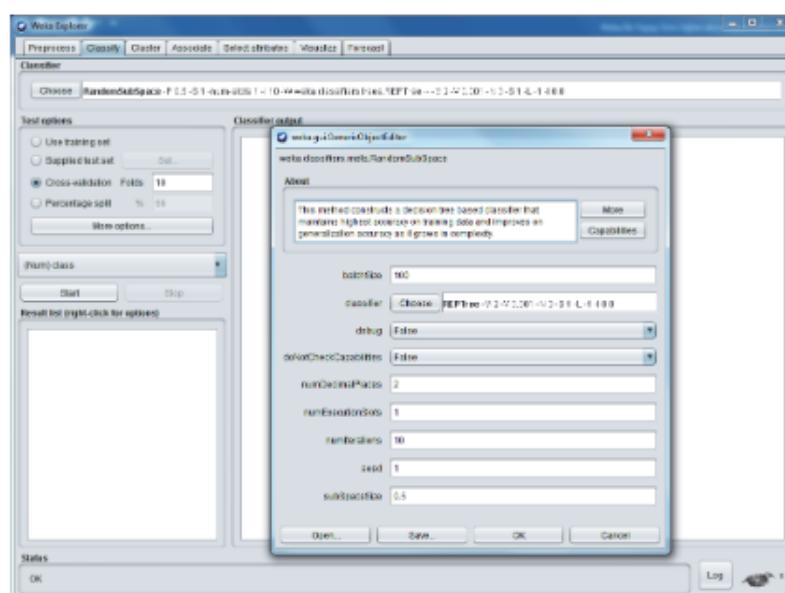
bir rastgele sayı çekirdeği kullanır. Bu sadece temel sınıflandırıcı randomize ise anlamlıdır; aksi halde, tüm sınıflandırıcılar aynı olur.



Şekil 4.97. RandomCommittee Algoritması ve Parametreleri

RandomSubSpace

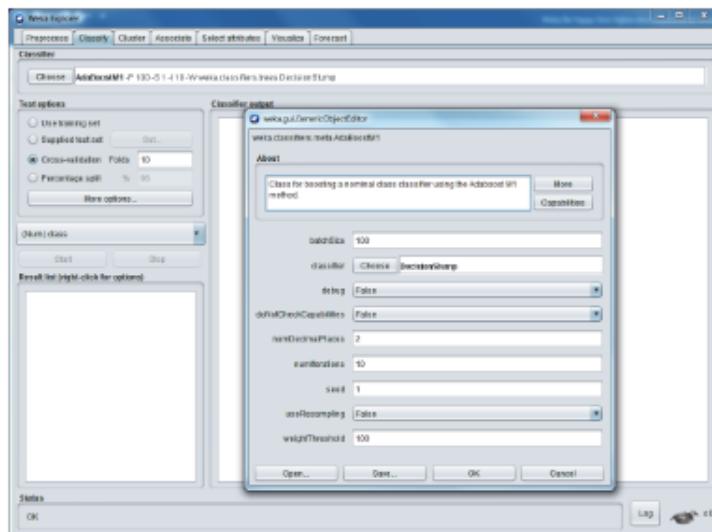
Her bir giriş niteliklerinin rastgele seçilmiş bir alt kümesi kullanılarak eğitilen bir sınıflayıcılar topluluğu oluşturur. Yineleme sayısı ve kullanılacak rastgele adım sayısının yanı sıra, nitelik alt kümelerinin boyutunu denetlemek için bir parametre sağlar.



Şekil 4.98. RandomSubSpace Algoritması ve Parametreleri

AdaBoostM1

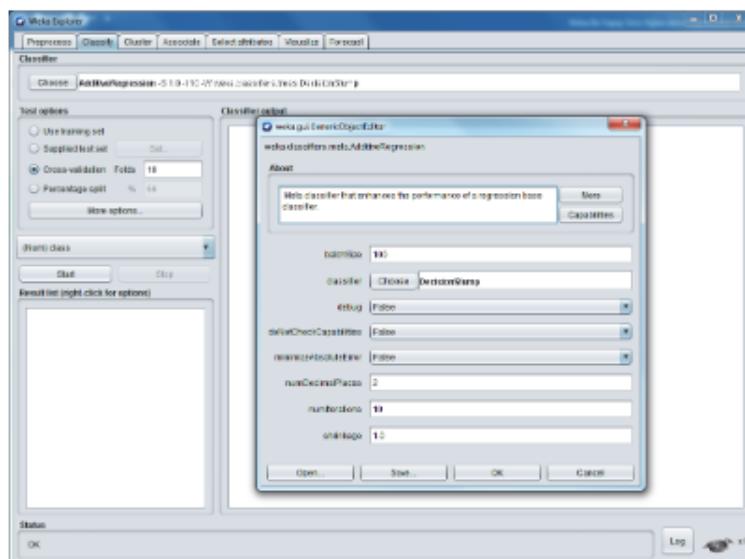
Klasik hızlandırma algoritmasını uygular. Ağırlık budaması için bir eşik belirterek hızlandırılabilir. Temel sınıflandırıcı, ağırlıklı örneklerle başa çıkmazsa, *AdaBoostM1* örnekleri yeniden örneklenir. Sadece nominal sınıflar için geçerlidir.



Şekil 4.99. AdaBoostM1 Algoritması ve Parametreleri

AdditiveRegression

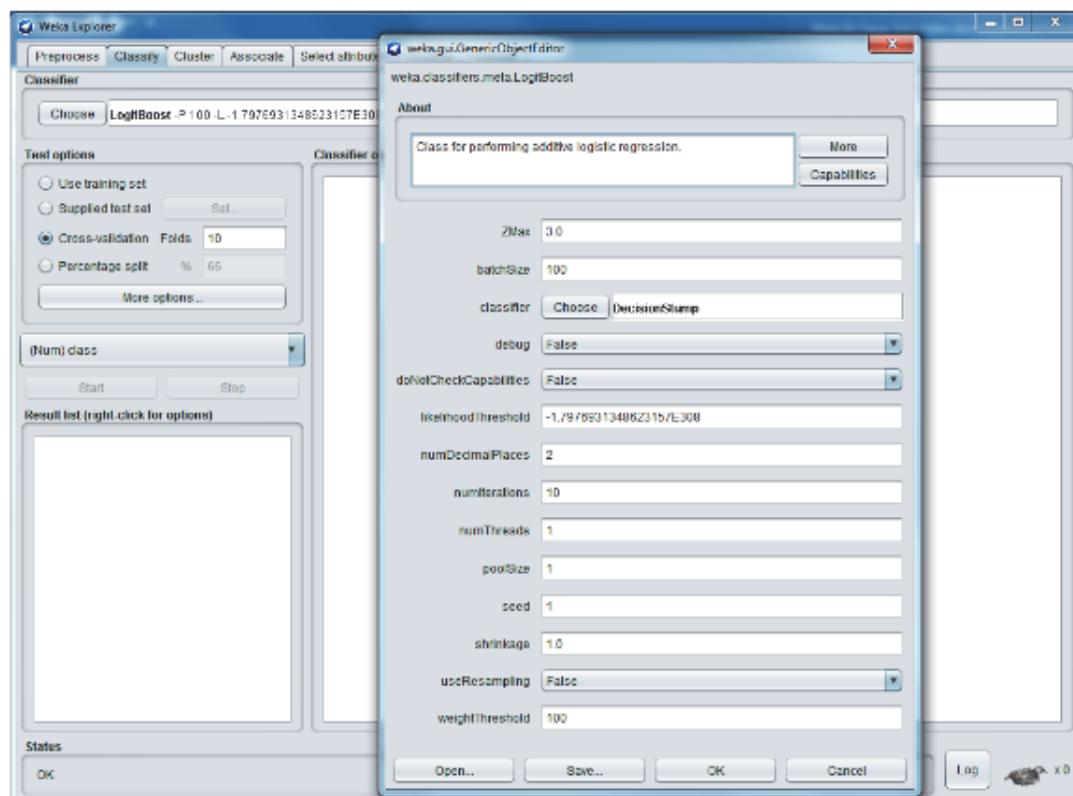
Bir regresyon öğrenmesinin performansını artırır. İki parametre vardır: öğrenme oranını yöneten büzülme ve üretilecek maksimum model sayısı. Eğer ikincisi sonsuz ise, hata duruncaya kadar çalışma devam eder.



Şekil 4.100. AdditiveRegression Algoritması ve Parametreleri

LogitBoost

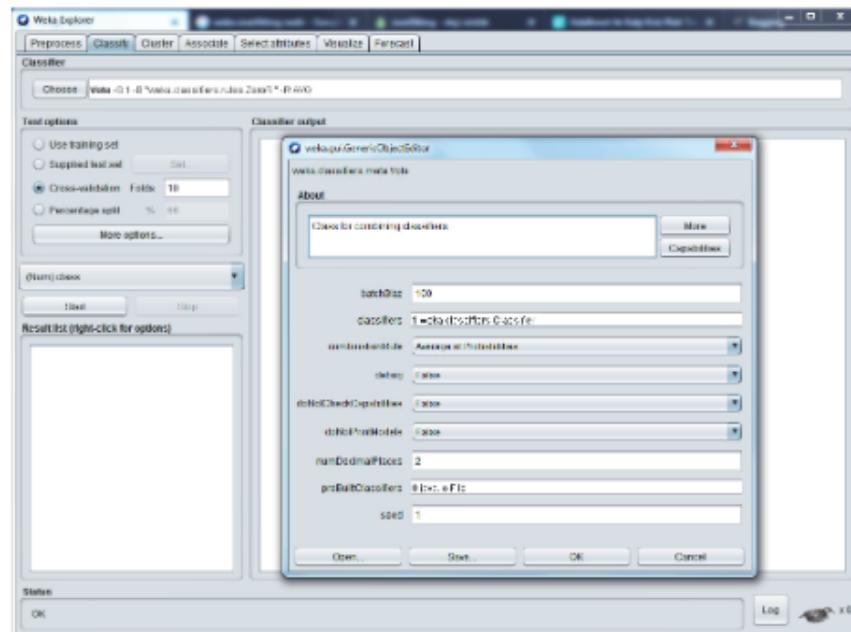
Ek bir lojistik regresyon gerçekleştirir. *AdaBoostM1* gibi, ağırlık budama için bir eşik belirterek hızlandırılabilir. Uygun iterasyon sayısı iç çapraz doğrulama kullanılarak belirlenebilir; ezberlemeyi önlemek için ayarlanabilen bir büzülme parametresi vardır ve yeniden ağırlıklandırma yerine yeniden örnekleme seçilebilir.



Şekil 4.101. LogitBoost Algoritması ve Parametreleri

Vote

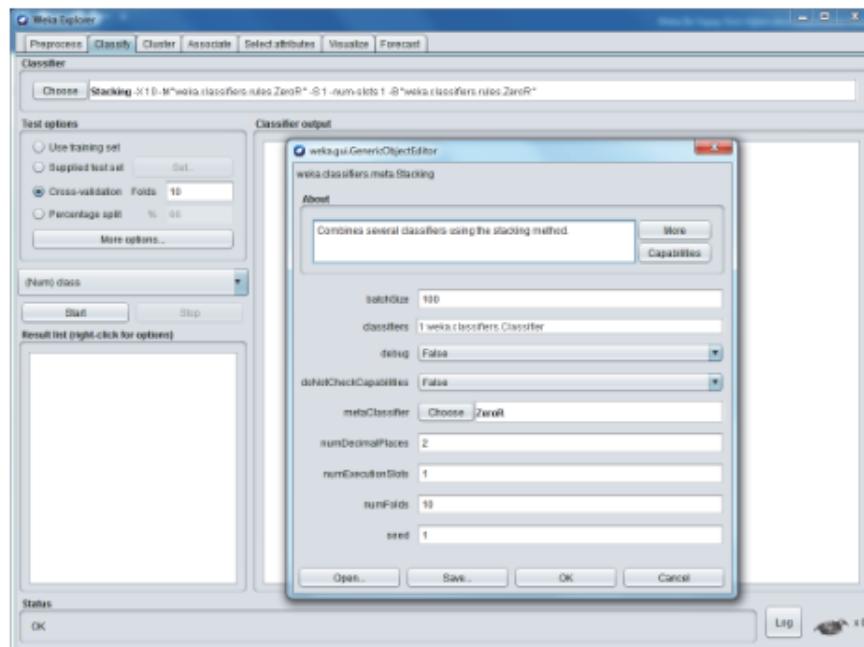
Sınıflandırıcıları birleştirmek için temel bir yöntem sağlar. Varsayılan şema, sırasıyla sınıflandırma ve regresyon için olasılık tahminlerini veya sayısal tahminleri ortalamasıdır. Örneğin, sınıflandırma için çoğunluk oyu kullanarak diğer kombinasyon şemaları mevcuttur.



Şekil 4.102. Vote Algoritması ve Parametreleleri

Stacking

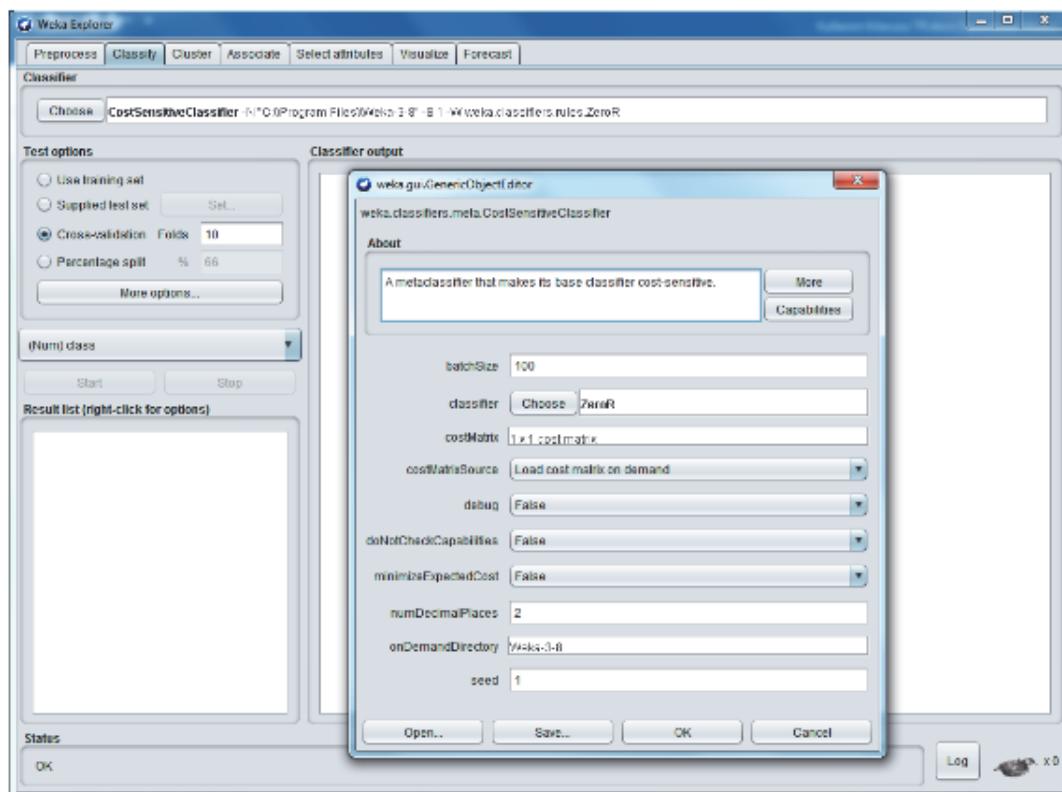
Sınıflandırıcıları hem sınıflandırma hem de regresyon problemleri için istiflemeyi birleştirir. Temel sınıflandırıcılar, meta öğrenme ve çapraz doğrulama katlamalarının sayısını belirtilebilir.



Şekil 4.103. Stacking Algoritması ve Parametreleri

CostSensitiveClassifier

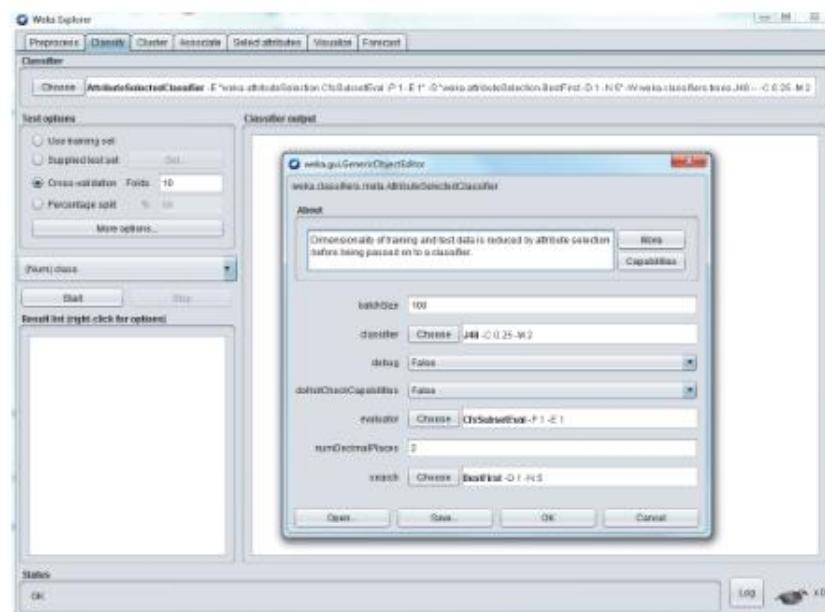
Maliyet matrisi bir parametre olarak veya birleşik isimle ve uzanti maliyetiyle isimlendirilen *onDemandDirectory* özelliği tarafından ayarlanan dizindeki bir dosyadan yüklenebilir. *CostSensitiveClassifier*, ya eğitim örneklerini her bir sınıfı atanan toplam maliyete göre yeniden değerlendirir ya da sınıfı en olası olandan ziyade, minimum beklenen yanlış sınıflandırma maliyetiyle tahmin eder.



Şekil 4.104. CostSensitiveClassifier Algoritması ve Parametreleri

AttributeSelectedClassifier

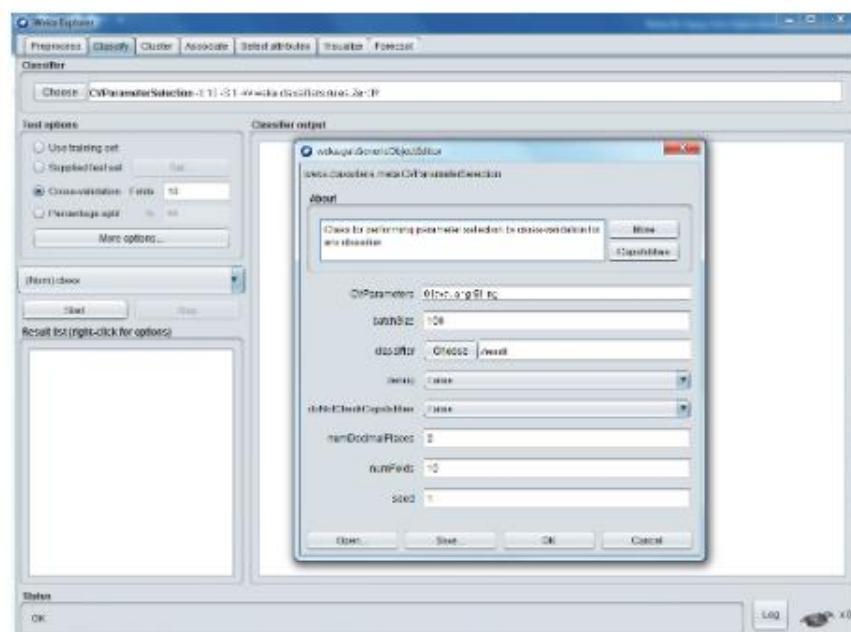
Beş sınıflandırıcı, temel sınıflandırıcının performansını optimize etmek için sarıcı teknğini kullanır. AttributeSelectedClassifier, nitelikleri seçer ve verilerin boyutlarını sınıflandırıcıya aktarmadan önce azaltır.



Şekil 4.105. AttributeSelectedClassifier Algoritması ve Parametreleri

CVParameterSelection

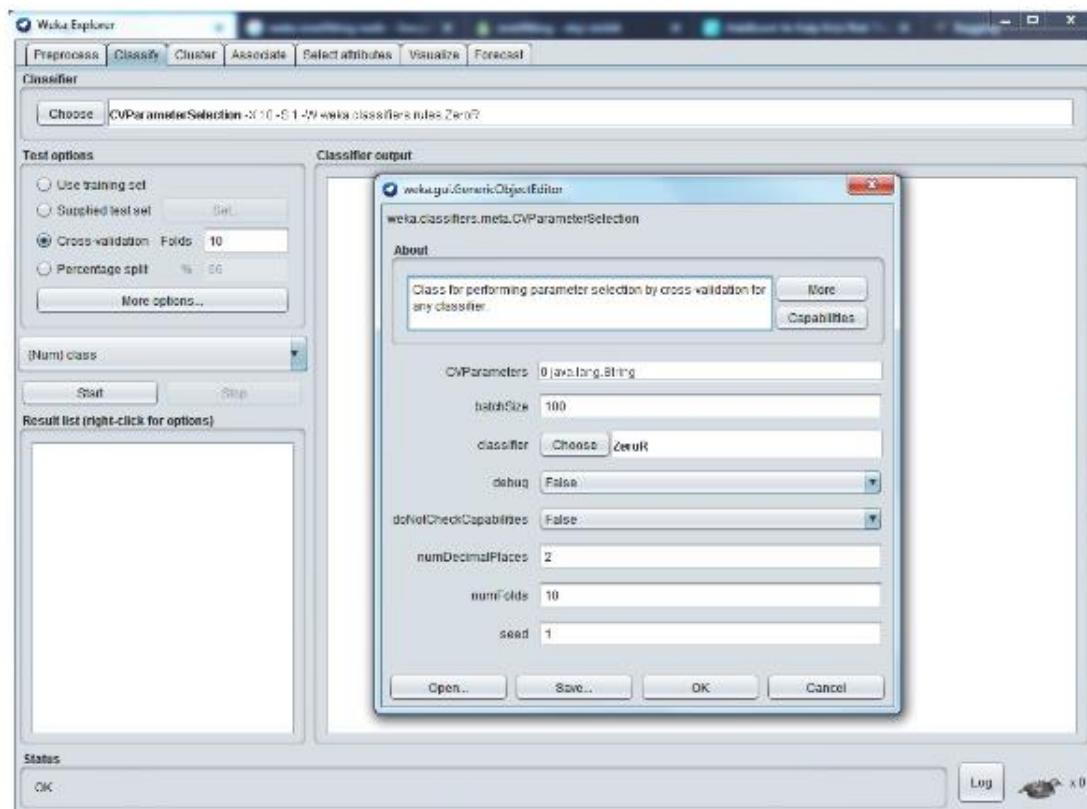
Parametreleri seçmek için çapraz doğrulamayı kullanarak performansı en iyi duruma getirir. Her parametre için, alt ve üst sınırlarını ve istenen artış sayısını içeren bir dize verilir. Kullanılacak çapraz doğrulama kat sayısını belirlenebilir.



Şekil 4.106. CVParameterSelection Algoritması ve Parametreleri

MultiScheme

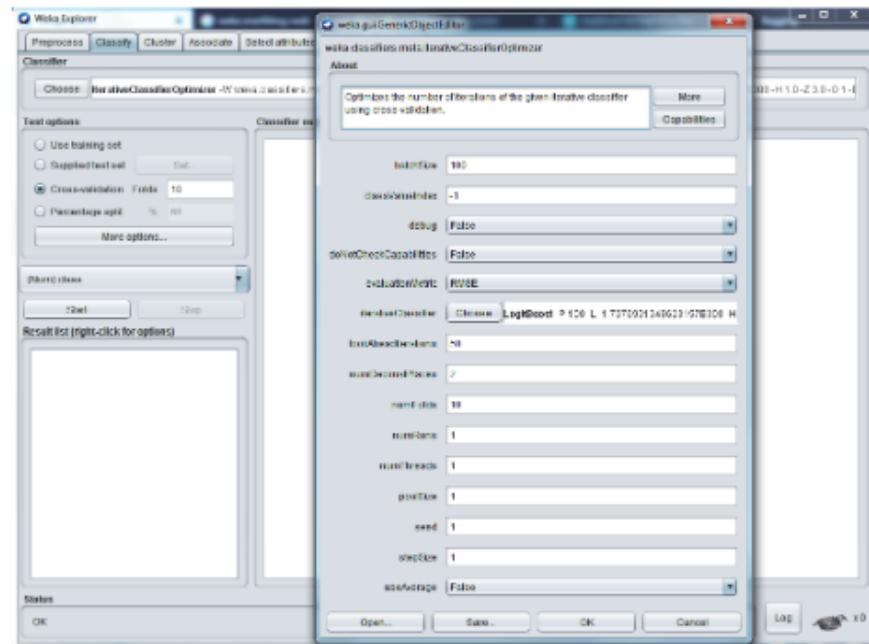
Yeniden yer değiştirme hatasını kullanarak veya eğitim verilerinde çapraz doğrulamayı kullanarak bir sınıflandırıcı seçer. Performans, sınıflandırma yüzdesel doğruluk ve regresyon için hata karelerinin ortalaması kullanılarak ölçülür.



Şekil 4.107. MultiScheme Algoritması ve Parametreleri

IterativeClassifierOptimizer

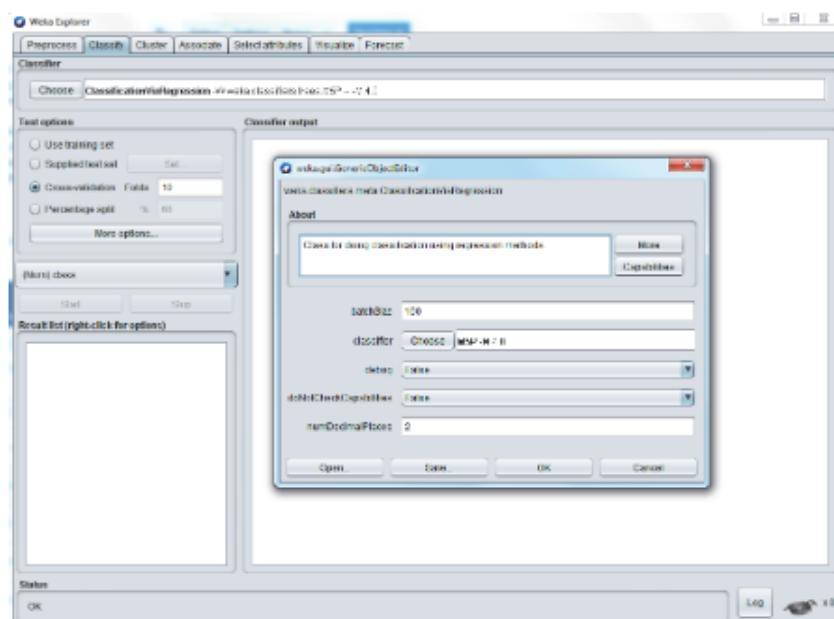
Eğitim verilerinde çapraz doğrulama kullanarak *Boosting* yöntemi gibi Tekrarlayan sınıflandırmalar ile iterasyonların sayısını optimize eder. Kullanıcı, ne kadar çapraz doğrulama katsayısının kullanılacağı, daha iyi bir çözüm bulmak için kaç iterasyonun ileriye bakılacağını ve değerlendirmenin ne sıklıkla gerçekleştirileceğini belirtebilir.



Şekil 4.108. IterativeClassifierOptimizer Algoritması ve Parametreleri

ClassificationViaRegression

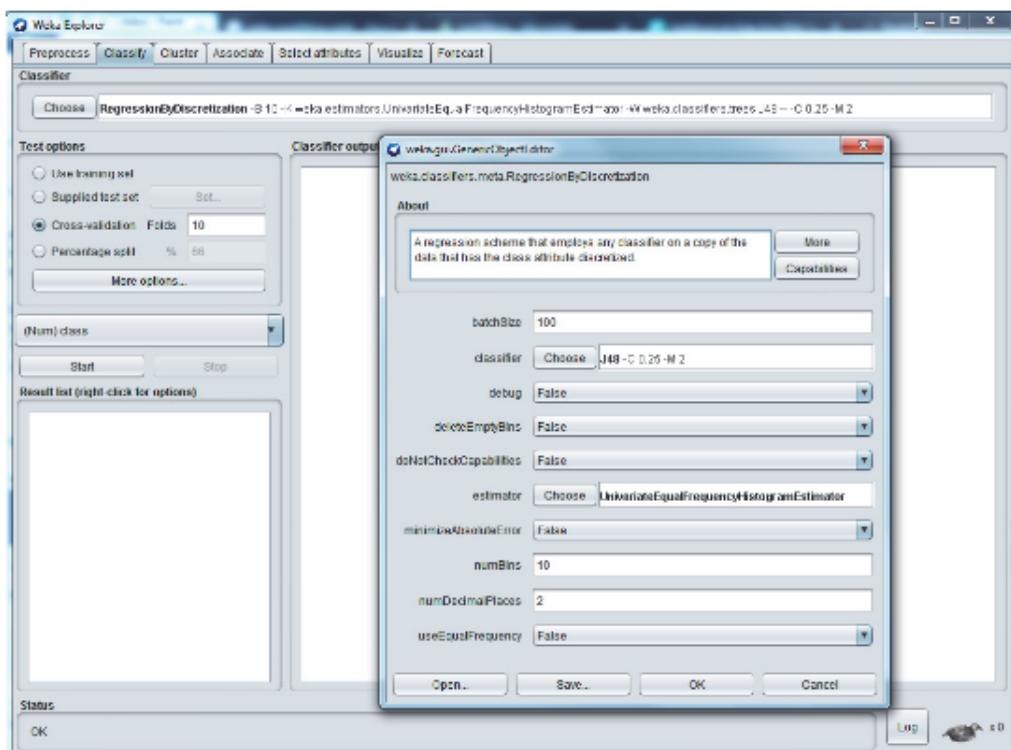
ClassificationViaRegression, *RegressionByDiscretization* ve *MultiClassClassifier*, bir tür görev için tasarlanan öğrenmeleri bir başkasına uyarlar. *ClassificationViaRegression*, sınıfı ikilileştirecek ve her değer için bir regresyon modeli oluşturarak bir regresyon yöntemi kullanarak sınıflandırma yapar.



Şekil 4.109. ClassificationViaRegression Algoritması ve Parametreleri

RegressionByDiscretization

Sınıf özniteliğini eşit genişlikli ayrıklıklaştırma kullanarak belirtilen sayıda kutuya ayıran ve sonra bir sınıflandırıcı kullanan bir regresyon şemasıdır. Tahminler, her bir ayrık aralık için ortalama sınıf değerinin ağırlıklı ortalaması olup, aralıklar için öngörülen olasılıklara dayanan ağırlıklardır.



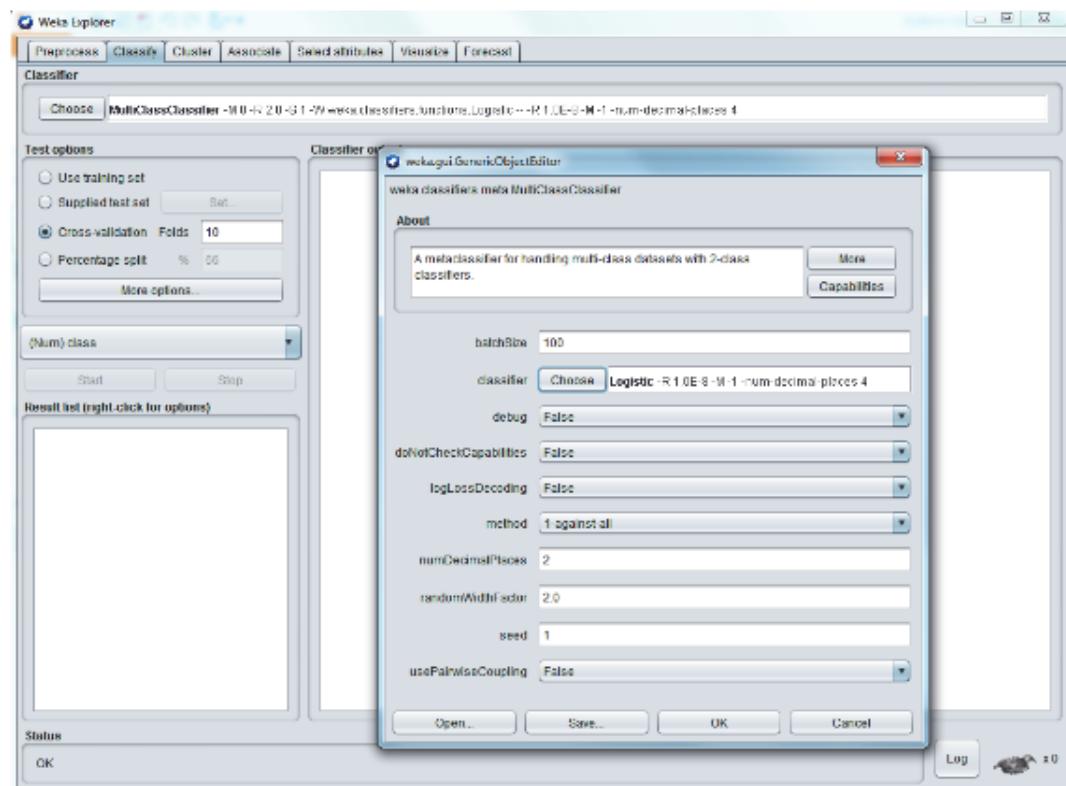
Şekil 4.110. RegressionByDiscretization Algoritması ve Parametreleri

MultiClassClassifier

MultiClassClassifier, burada anlatılan tüm yöntemlerden herhangi birini kullanarak, çok sınıflı sorunları iki sınıflı sınıflandırıcılar ile ele alır:

- Biri tüm diğerlerine karşı
- Tahmin etmek için oylama kullanarak çiftli sınıflandırma.
- Tamamlayıcı hata düzeltme kodları.
- Rastgele seçilen hata düzeltme kodları.

Rastgele kod vektörlerinin iyi hata düzeltme özelliklerine sahip olduğu bilinmektedir. Çift yönlü sınıflandırma için olasılık tahminlerinin çiftli birleştirilmesi açık olabilir.



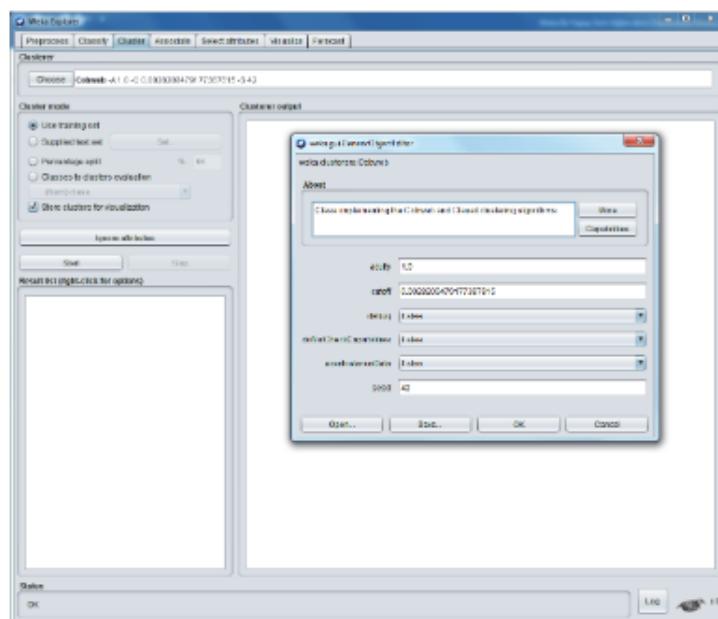
Şekil 4.111. MultiClassClassifier Algoritması ve Parametreleri

4.7. Kümeleme Algoritmaları (Clustering)

Explorer ekranındaki kümeleme sekmesi (*Cluster*), kümeleme algoritmalarına erişimi sağlar.

Cobweb

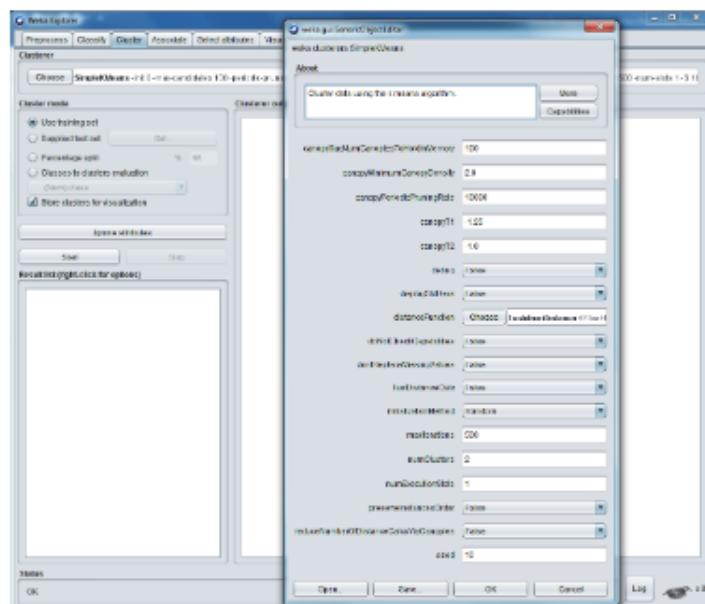
Türkçesi örümcek ağı olmasının nedeniyle de mantığı örümcek ağına çok benzeyen ağ şeması oluşturur. Artımlı kavramsal kümelemede popüler ve basit bir metottur. Bu metotta girdi nesneleri kategorik alan değer çiftleri olarak tanımlanır. Cobweb sınıflama ağacı formunda bir hiyerarşik kümeleme oluşturur. Sınıflama ağacındaki her bir nokta bir kavrama karşılık gelmektedir ve bu nokta altında sınıflama nesneleri özetleyen kavramın olasılıklı tanımlamasını içerir. Cobweb, nominal nitelikler için Cobweb algoritmasını ve sayısal nitelikler için Classit algoritmasını uygular.



Şekil 4.112. Cobweb Algoritması ve Parametreleri

SimpleKMeans

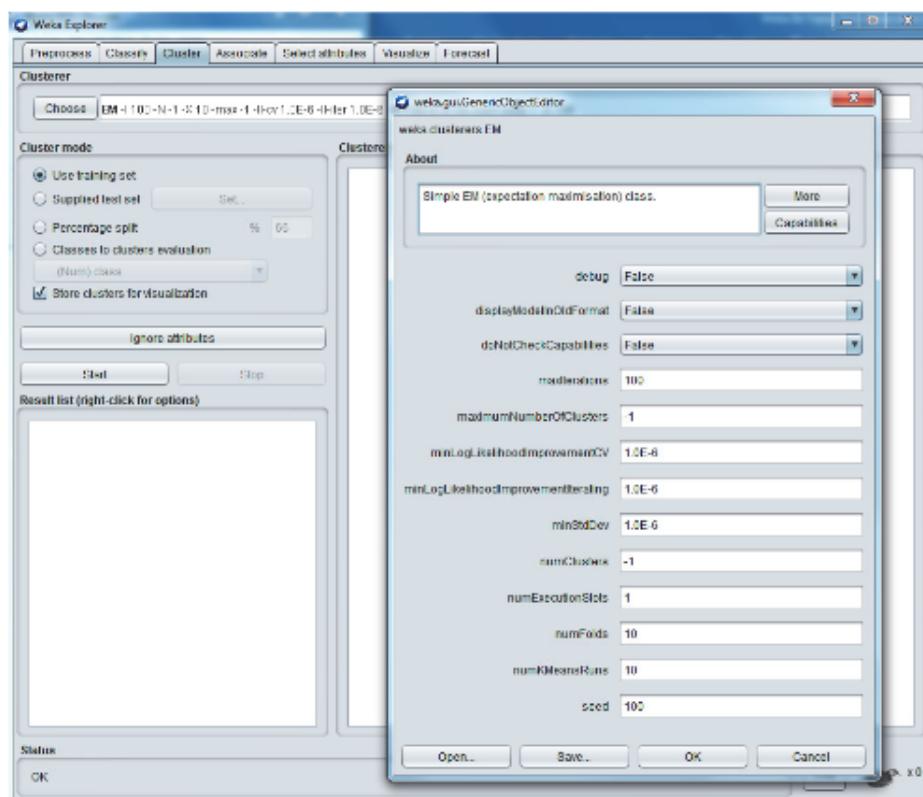
K ortalamaları yöntemi kullanılarak kümeleme yapar. N adet veri nesnesinden oluşan bir veri kümесini giriş parametresi olarak verilen K adet kümeye bölümlere (kümeleme) yapar. Kullanıcı, Öklit ve Manhattan mesafe metrikleri arasında seçim yapabilir. İkinci durumda, algoritma aslında karaçları yerine k-medyanlarıdır ve merkezler, küme içi mesafe fonksiyonunu en aza indirmek için araçlar yerine medyanlara dayanmaktadır.



Şekil 4.113. SimpleKMeans Algoritması ve Parametreleri

EM

Basit bekleni maksimizasyonu sınıfıdır. *EM*, her bir kümeye ait olma olasılığını belirten bir olasılık dağılımı atar. EM, çapraz doğrulama yoluyla kaç tane kümenin oluşturulacağına karar verebilir veya kaç tane kümenin oluşturulacağını muhtemel olarak belirtebilir. Kullanıcı tanımlı olarak maksimum iterasyon sayısını belirtebilir ve normal yoğunluk hesaplaması için izin verilen standart sapmayı ayarlayabilir. Kümeler, çapraz kovaryans matrisli Gauss dağılımlarıdır.



Şekil 4.114. EM Algoritması ve Parametreleri

Aşağıdaki şekilde ise varsayılan seçeneklerle birlikte, `cpu.arff` verileri için SimpleKMeans algoritması uygulandığındaki çıkışı göstermektedir. İki kume ve Öklid mesafesi kullanılmıştır. Kümelemenin sonucu, satırları nitelik adları olan ve sütunları kume merkezlerine karşılık gelen bir tablo olarak gösterilmektedir. Başlangıçta ek bir kume tüm veri kümelerini gösterir. Her kümeye örneklerin sayısı, sütununun üstündeki parantez içinde görünür. Her tablo girdisi, o sütundaki kume için karşılık gelen sayısal nitelikler için niteliğin ortalaması veya nominal nitelikler için niteliğin mod değeridir. Kullanıcılar sayısal nitelikler için standart sapmaları ve nominal nitelikler

icin frekans sayılarını da gösterebilir. Çıktının alt kısmı, öğrenilen kümeleme modelinin uygulanmasının sonucunu göstermektedir. Bu durumda, her bir eğitim örneğini, her bir sütunun en üstündeki parantez sayıları ile aynı sonucu gösteren kümelerden birine atanır.

```

==== Run information ====
Scheme:      weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Relation:    cpu
Instances:   209
Attributes:  7
              MYCT
              MMIN
              MMAX
              CACH
              CHMIN
              CHMAX
              class
Test mode:   evaluate on training data

==== Clustering model (full training set) ====

kMeans
=====
Number of iterations: 12
Within cluster sum of squared errors: 21.17961501115821

Initial starting points (random):

Cluster 0: 600,768,2000,0,1,1,16
Cluster 1: 59,4000,12000,32,6,12,113

Missing values globally replaced with mean/mode

Final cluster centroids:
  Cluster#
Attribute Full Data      0      1
             (209.0) (171.0) (38.0)
  -----
MYCT      203.823 238.9064 45.9474
MMIN     2867.9809 1721.0058 8029.3684
MMAX     11796.1531 7743.7778 30031.8421
CACH     25.2057 11.7544 85.7368
CHMIN    4.6986 2.7076 13.6579
CHMAX    18.2679 12.3626 44.8421
class    105.622 50.2632 354.7368

Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

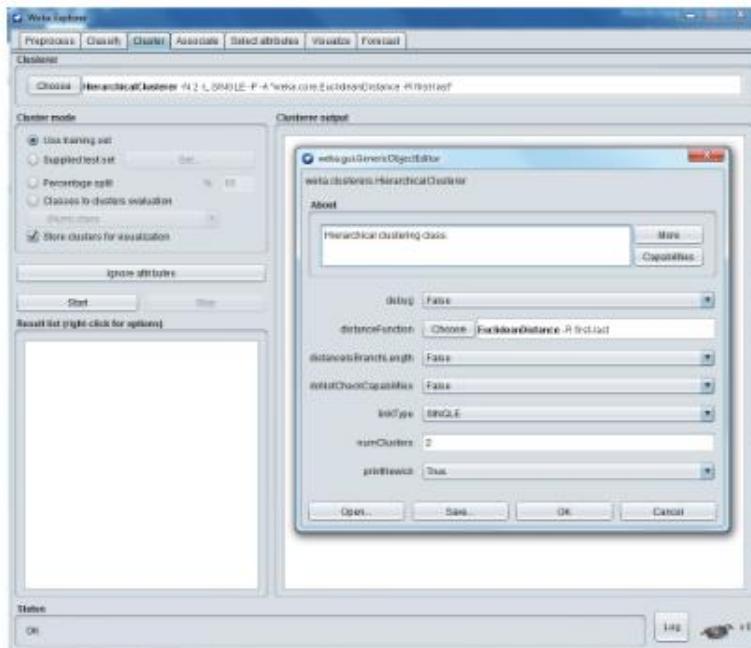
0  171 ( 82%)
1  38 ( 18%)

```

Şekil 4.115. Cpu.arff Dosyasına SimpleKMeans Algoritması Uygulama Sonuçları

HierarchicalClusterer

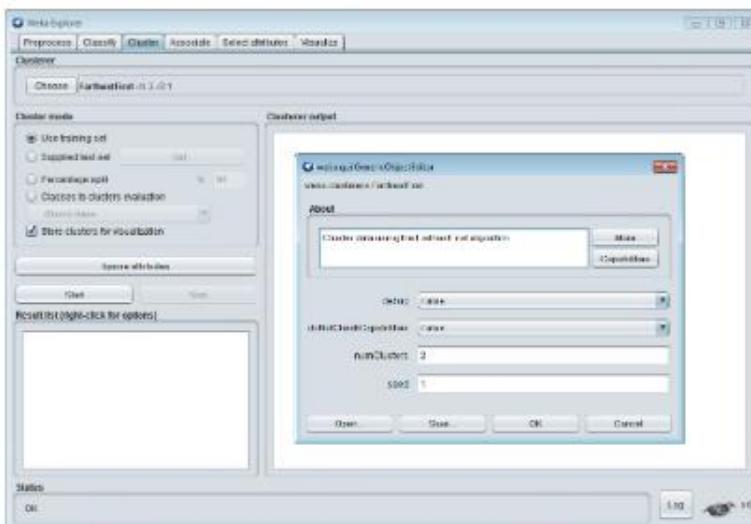
Öbek ağaçları olarak adlandırılan ağaçlar biçimindeki hiyerarşik kümelieme çözümleri, birçok uygulama alanı için büyük ilgi görmektedir. Hiyerarşik ağaçlar, farklı soyutlama düzeylerindeki verilerin bir görünümünü sağlar.



Şekil 4.116. HierarchicalClusterer Algoritması ve Parametreleri

FarthestFirst

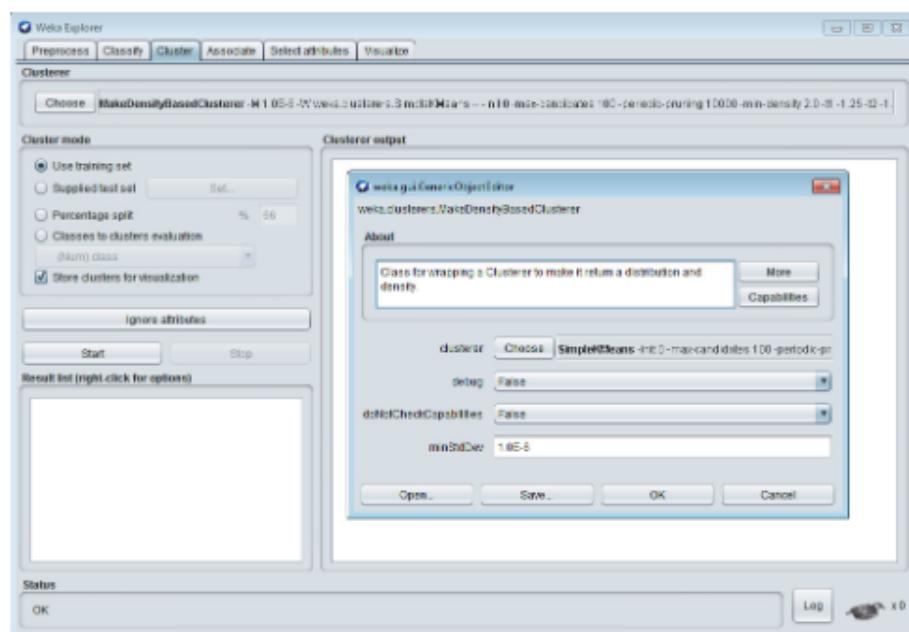
En uzak ilk çaprazlama algoritması uygulanır. K ortalama algoritması üzerine modellenerek hızlı, basit, yaklaşık bir küme aracı olarak oluşturulmuştur.



Şekil 4.117. FarthestFirst Algoritması ve Parametreleri

MakeDensityBasedClusterer

Olasılık dağılımını ve yoğunluğunu döndürmek için küme algoritmasını paketleyen meta kümeleme algoritmasıdır. Her bir küme ve nitelik için, ayrı bir dağılım veya simetrik bir normal dağılıma uyar. Normal dağılımın minimum standart sapması bir parametre olarak kullanıcı tarafından verilebilir.

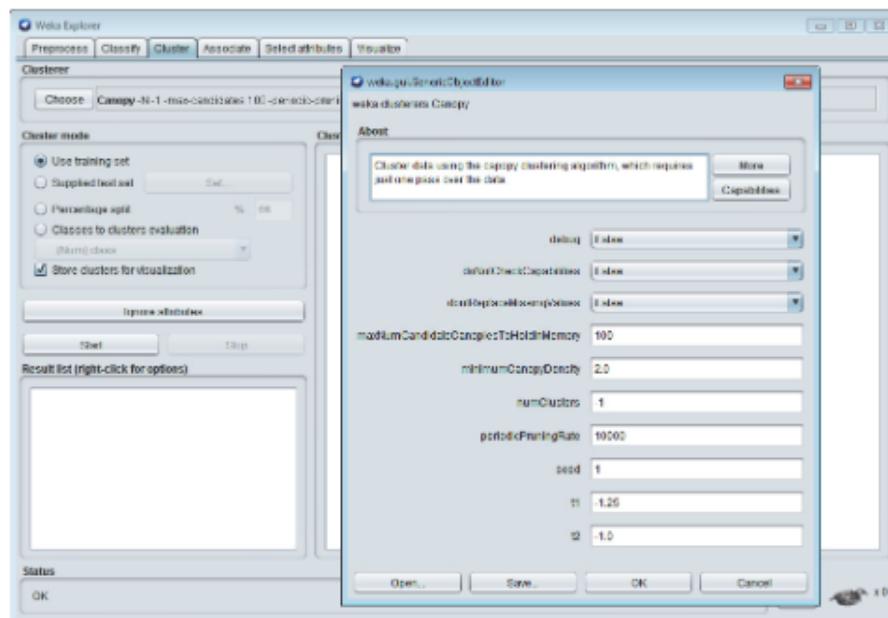


Şekil 4.118. MakeDensityBasedClusterer Algoritması ve Parametreleri

Canopi

Kubbe kümeleme algoritması algoritmasını uygular. Kubbe kümelenmesi, genellikle k-araçları için bir başlatma stratejisi olarak büyük veri setlerinde kullanılmak üzere hızlı bir kümeleme metodu olarak kullanılır. Burada başka bir algoritmanın doğrudan uygulanması pratik olmayabilir. Algoritma, giriş verisini, bölgenin merkezinden gevşek bir mesafe olan T_1 ile tanımlanan hipersferler biçiminde yakınlık bölgelerine (kubbeler) ayırır. Algoritma tarafından kaç kanopi oluşturulduğunu kontrol etmek için ikinci bir sıkı mesafe, T_2 (*burada $T_2 < T_1$*) kullanılır. Verilerin en fazla iki geçiş gereklidir. İlk geçiş, kanopileri oluşturur ve ilk kanopinin merkezini oluşturmak için rastgele bir örnek seçerek başlar. Bunu takiben, geri kalan örneklerin her biri, mevcut her bir kanopinin merkezine olan mesafesini hesaplayarak dikkate alınır. Eğer verilen bir örnek tüm kanopilere T_2 mesafesinin dışındaysa, yeni bir kanopi oluşturur, aksi halde atılır. Verilerin ikinci geçisi, kanopi örneklerine atar ve gevşek T_1 mesafesini kullanır. Kanopiler T_1 me-

safesine göre üst üste binebildiğinden, belirli bir örneğin birden fazla kanopiye ait olması mümkündür.



Şekil 4.119. Canopy Alogitması ve Parametreleri

4.8. Birliktelik Kural Öğrenmeleri

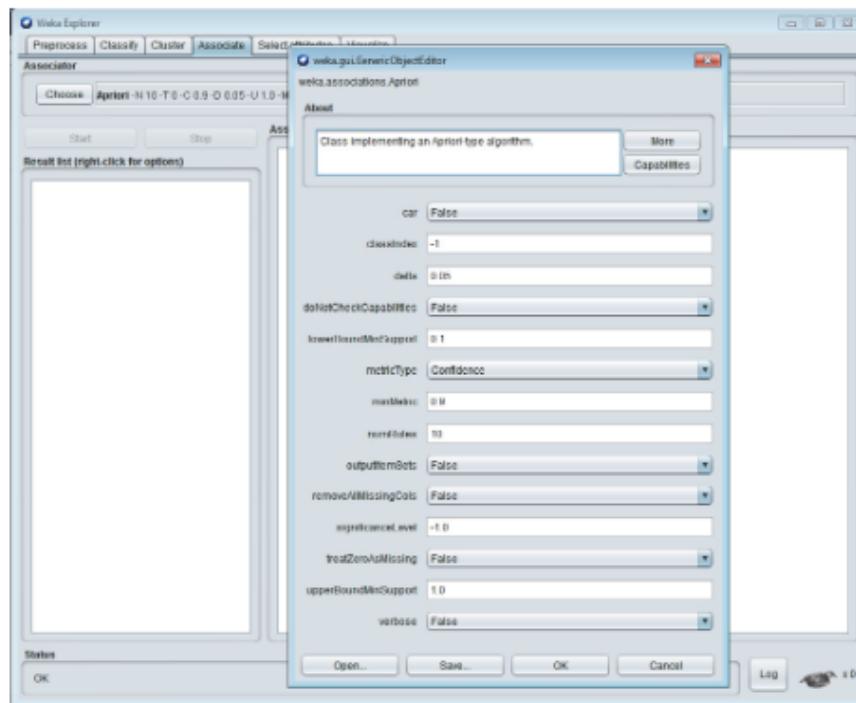
Apriori

Verilerin %100'lük minimum desteğiyle başlar ve bunu en az 10 kural oluncaya kadar (veya %10'luk alt sınıra ulaşınca kadar) %5'lük adımlarla azaltır. Buradaki varsayılan değerler kullanıcı tarafından parametre giriş ekranından değiştirilebilir. Sıralama kuralları için dört alternatif ölçüt vardır:

- Güven (*Confidence*): Sonuçta kapsanan öncülün örneklerle oranıdır.
- Asansör (*Lift*): Desteğin güvene (*Confidence*) bölünmesiyle elde edilir.
- Kaldıraç (*Leverage*): Hem öncül hem de sonuçların istatiksel olarak bağımsız olması durumunda beklenenlerin ötesinde, hem öncül hem de sonuçların kapsadığı ek örneklerin oranıdır.
- Kanaat (*Conviction*): Bir elemanın bir diğer eleman yokken ki görülmeye olasılığını hesaplar.

Kullanıcı tarafından bir önem seviyesi (*significance*) belirleyerek kuralların bu düzeyde önemlilik açısından test edilmesi sağlanabilir. Bu algorit-

manın, sonučta tek bir niteliçin değerini içeren kurallar ile sınırlama seçenekleri bulunmaktadır. Bunlara sınıf birliktelik kuralları denir.

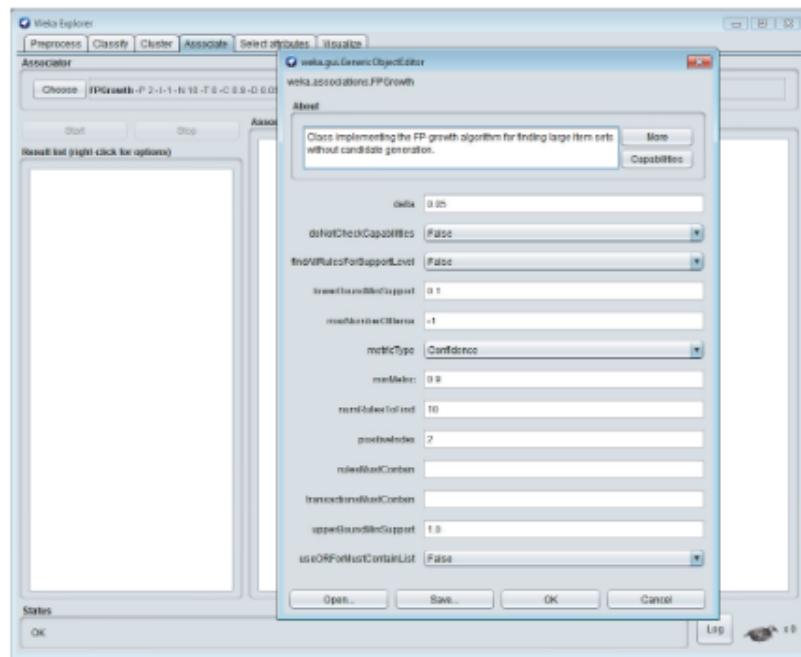


Şekil 4.120. Apriori Algoritması ve Parametreleri

Piyasadaki alışveriş verileri bu algoritma ile işlenmek istenirse, ARFF dosyaları belirli bir şekilde kodlanmalıdır. Özellikle alışveriş içinde bulunmayan ürünler ile ilgilenilmemiçi için bunlara karşılık gelen değerler tek değerli nominal olarak verilmelidir. Eksik değerler, bir ürünün alışverişte olmadığını göstermek için kullanılabilir.

FPGrowth

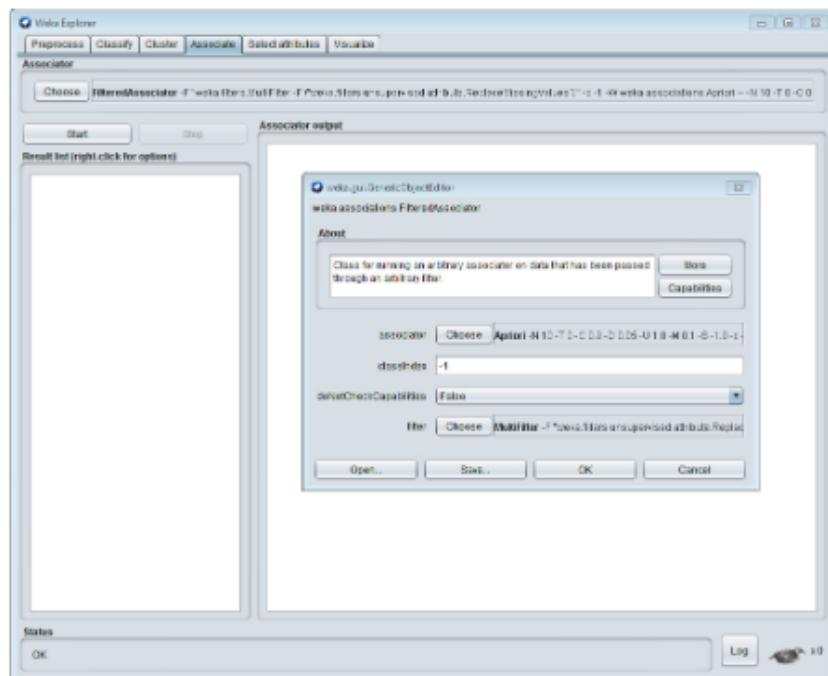
Sık desen ağacı madenciliği algoritmasını uygular. Alışveriş sepeti verisi için tasarlanan Apriori'nin bu tür veriler için özel kodlaması burada uygulanmamıştır. Tüm katılımların ikili nominal değerler olması beklenir ve kullanıcı, iki değerin hangisinin pozitif olarak ele alınacağını belirleyebilir, yani, sepette varlığını belirtir (*varsayılan değer ikinci değeri kullanmaktadır*). *FPGrowth*, standart veya seyrek durumlarda çalışabilir. Parametrelerinin çoğu, Apriori ile aynıdır. İstenen sayıda kuralı aynı şekilde bulur (*minimum desteği yinelemeli olarak azaltarak*). İsteğe bağlı olarak, kullanıcı, minimum desteğiin alt sınırını karşılayan tüm kuralları ve sıralama ölçütü için belirlenen minimum değeri (*güven, asansör, kaldırıç veya kanaat*) bulabilir.



Şekil 4.121. FPGrowth Algoritması ve Parametreleri

FilteredAssociator

Verilerin bir birleştiriciye ulaşmadan önce bir filtreden geçirilmesini sağlar. Hem filtre hem de temel birleştirici, kullanıcının yapılandırabileceği seçeneklerdir.



Şekil 4.122. FilteredAssociator Algoritması ve Parametreleri

4.9. Nitelik Seçimi

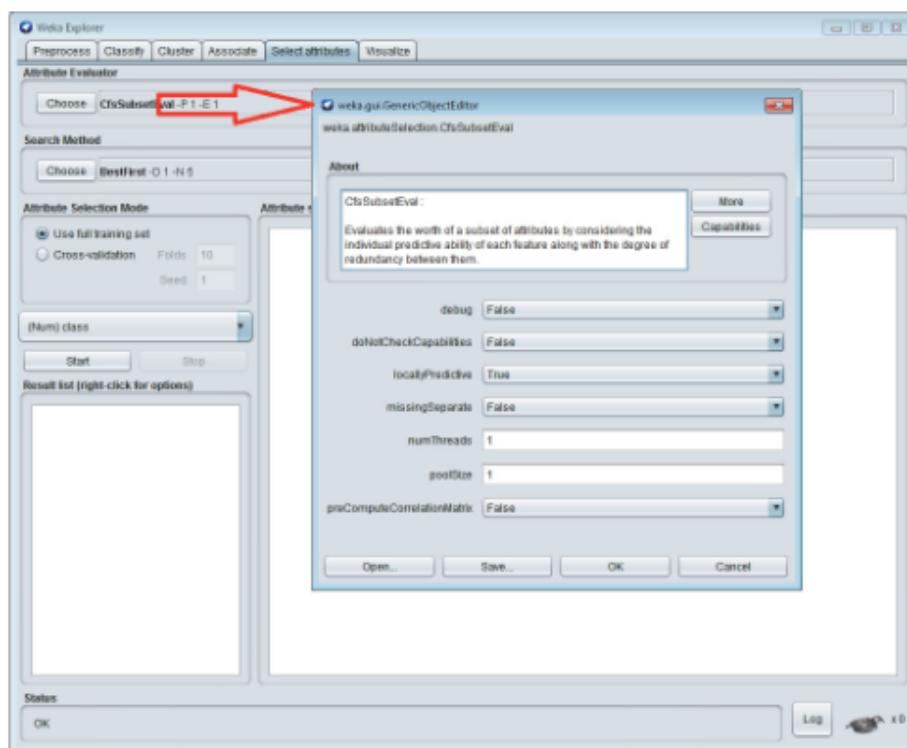
Nitelik seçimi, normal olarak, her birinin değerlendirildiği nitelik alt kümelerinin boşluğu aranarak yapılır. Potansiyel olarak daha hızlı ancak daha az doğru olan bir yaklaşımdır. Nitelikleri tek tek değerlendirmek ve bunları sıralamak, seçilen bir kesme noktasının altına düşen nitelikleri atmakta.

4.9.1. Nitelik Değerlendiriciler

CfsSubsetEval

CfsSubsetEval ve *WrapperSubsetEval* algoritmaları alt küme değerlendirmecileri olarak bir alt nitelik kümesi alır ve aramaya kılavuzluk eden sayısal bir ölçü döndürür. Diğer WEKA nesneleri gibi yapılandırılmışlardır.

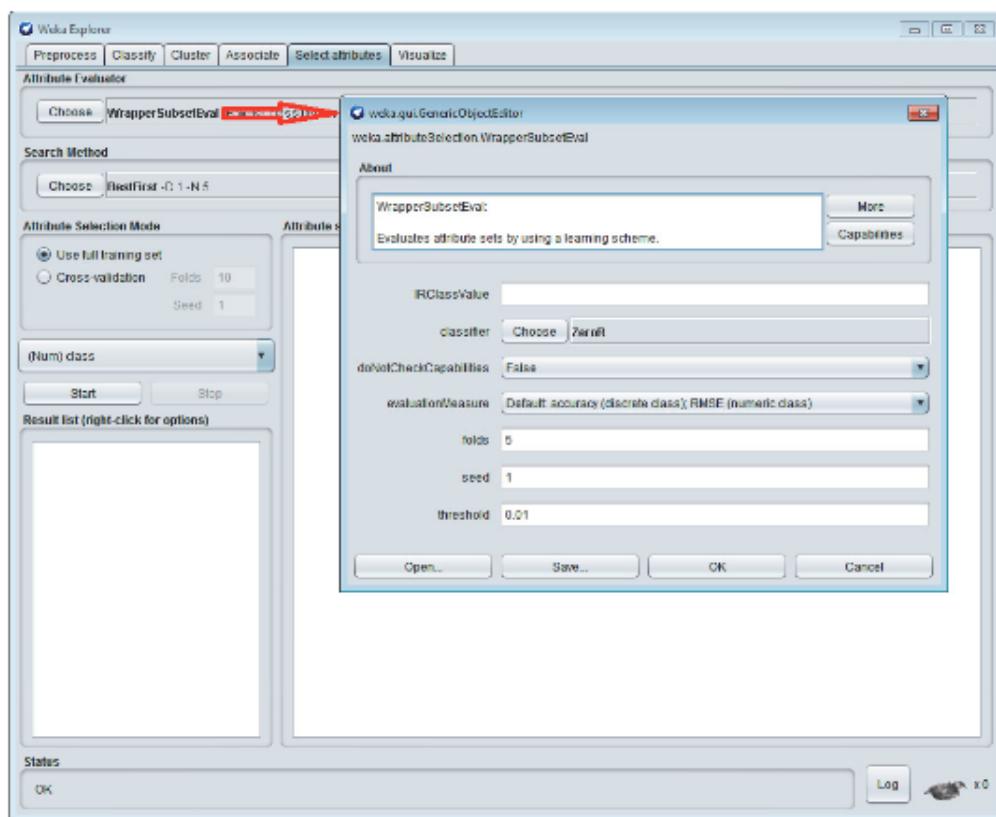
CfsSubsetEval her bir niteliğin öngörü yeteneğini ve bunlar arasındaki fazlalık derecesini ayrı ayrı değerlendirir. Sınıfla yüksek düzeyde ilişkili olan ancak düşük karşılıklı korelasyona sahip nitelik kümelerini tercih eder. Eksik değerler, ayrı bir değer olarak ele alınabilir veya sayımları, frekanslarına oranla diğer değerler arasında dağıtılabılır.



Şekil 4.123. CfsSubsetEval Algoritması ve Parametreleri

WrapperSubsetEval

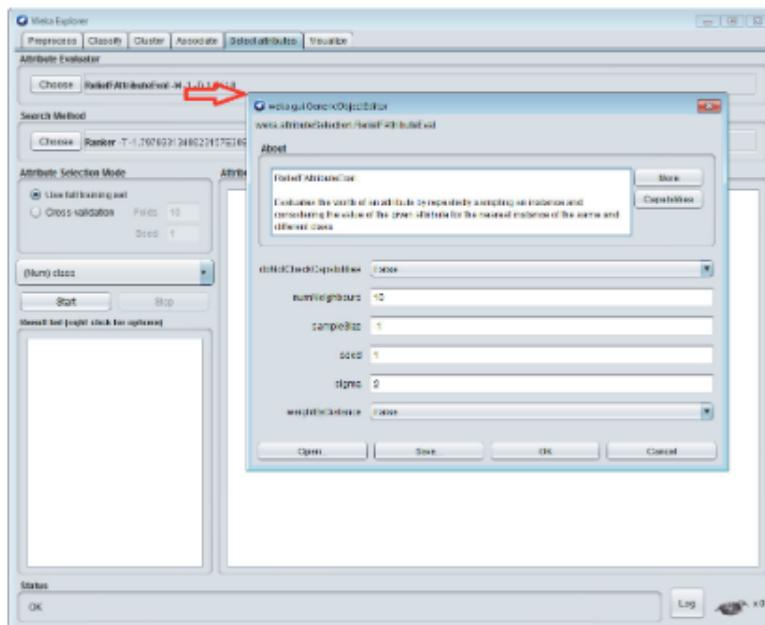
CfsSubsetEval, nitelik seçimine yönelik bir filtre yöntemidir. *WrapperSubsetEval* sarmalayıcı yöntemini uygular. *WrapperSubsetEval*, nitelik kümesini değerlendirmek için bir sınıflandırıcı kullanır ve her bir kümenin öğrenme şemasının doğruluğunu tahmin etmek için çapraz doğrulamayı kullanır.



Şekil 4.124. WrapperSubsetEval Algoritması ve Parametreleri

ReliefFAttributeEval

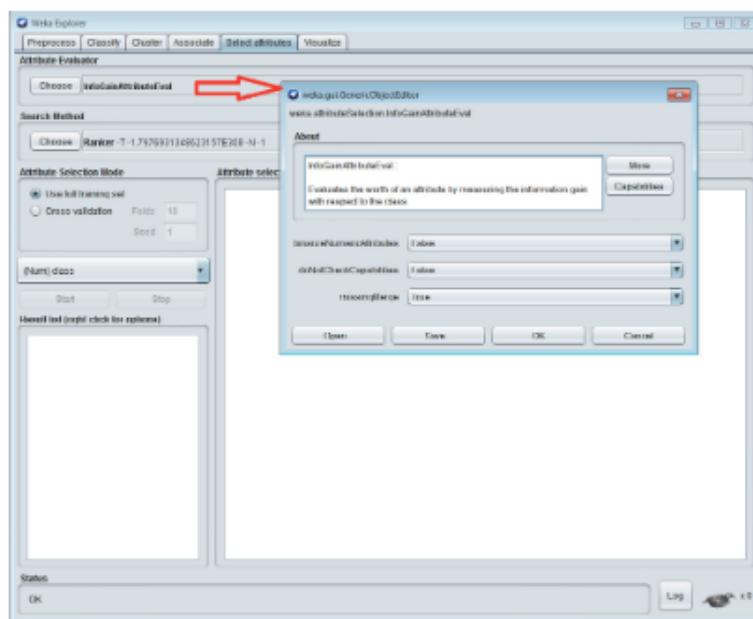
Tek nitelikli değerlendirciler Ranker arama yöntemiyle Ranker'in belirli bir sayıyı attığı bir sıralı liste oluşturmak için kullanılır. *ReliefFAttributeEval* örnek tabanlıdır: örnekleri rastgele örnekler ve aynı ve farklı sınıfların komşu örneklerini kontrol eder. Kesikli ve sürekli sınıf verileri üzerinde çalışır. Parametreler, örneklenen örnek sayısını, kontrol edilecek komşuların sayısını, komşuların mesafelere göre ağırlıklarını ve mesafelerin ne kadar hızlı kaçılıklarının bozulduğunu belirleyen üstel bir fonksiyonu belirtir.



Şekil 4.125. ReliefFAttributeEval Algoritması ve Parametreleri

InfoGainAttributeEval

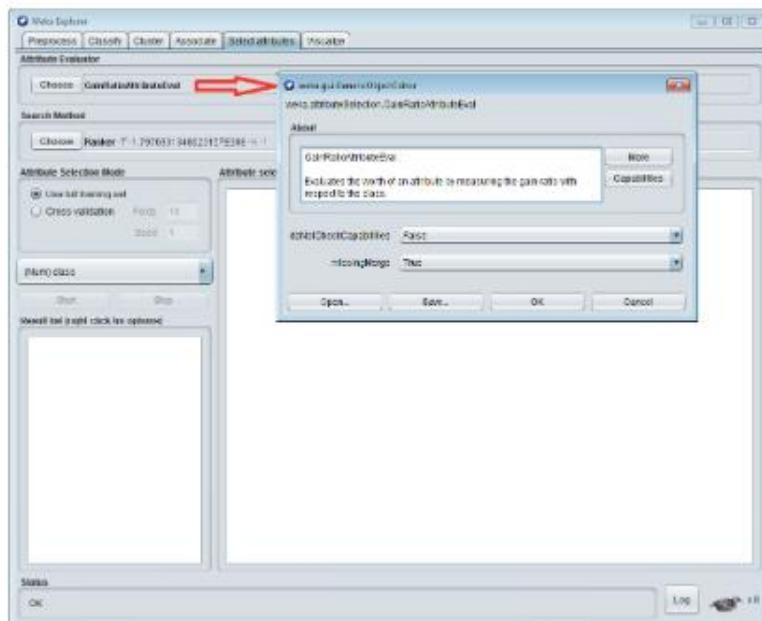
Nitelik kazanımlarını sınıfla ilgili olarak ölçerek değerlendirir. Önce MDL tabanlı discretizasyon yöntemini kullanarak sayısal nitelikleri ayırr (*bunun yerine bunları ikiye ayırmak için ayarlanabilir*). Bu yöntem, sonraki üçü ile birlikte, eksik değerleri ayrı bir değer olarak işleyebilir veya sayımları frekansları ile orantılı olarak diğer değerler arasında dağıtabilir.



Şekil 4.126. InfoGainAttributeEval Algoritması ve Parametreleri

GainRatioAttributeEval

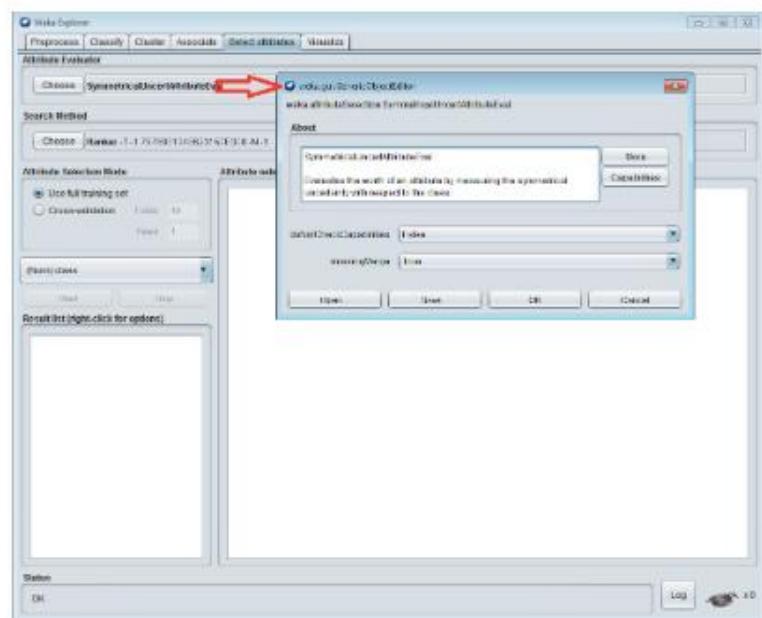
Niteliklerini sınıf ile ilgili olarak kazanma oranlarını ölçerek değerlendirir.



Şekil 4.127. GainRatioAttributeEval Algoritması ve Parametreleri

SymmetricalUncertAttributeEval

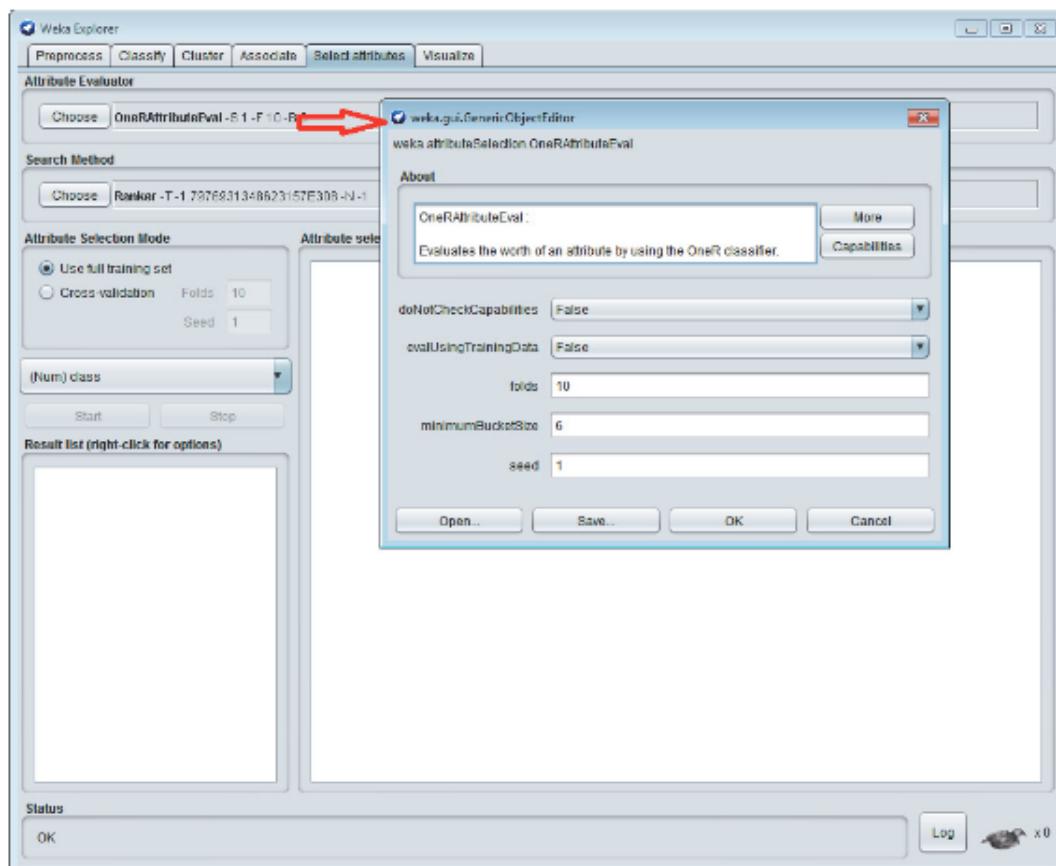
Sınır göre simetrik belirsizliği ölçerek bir niteliği değerlendirir.



Şekil 4.128. SymmetricalUncertAttributeEval Algoritması ve Parametreleri

OneRAttributeEval

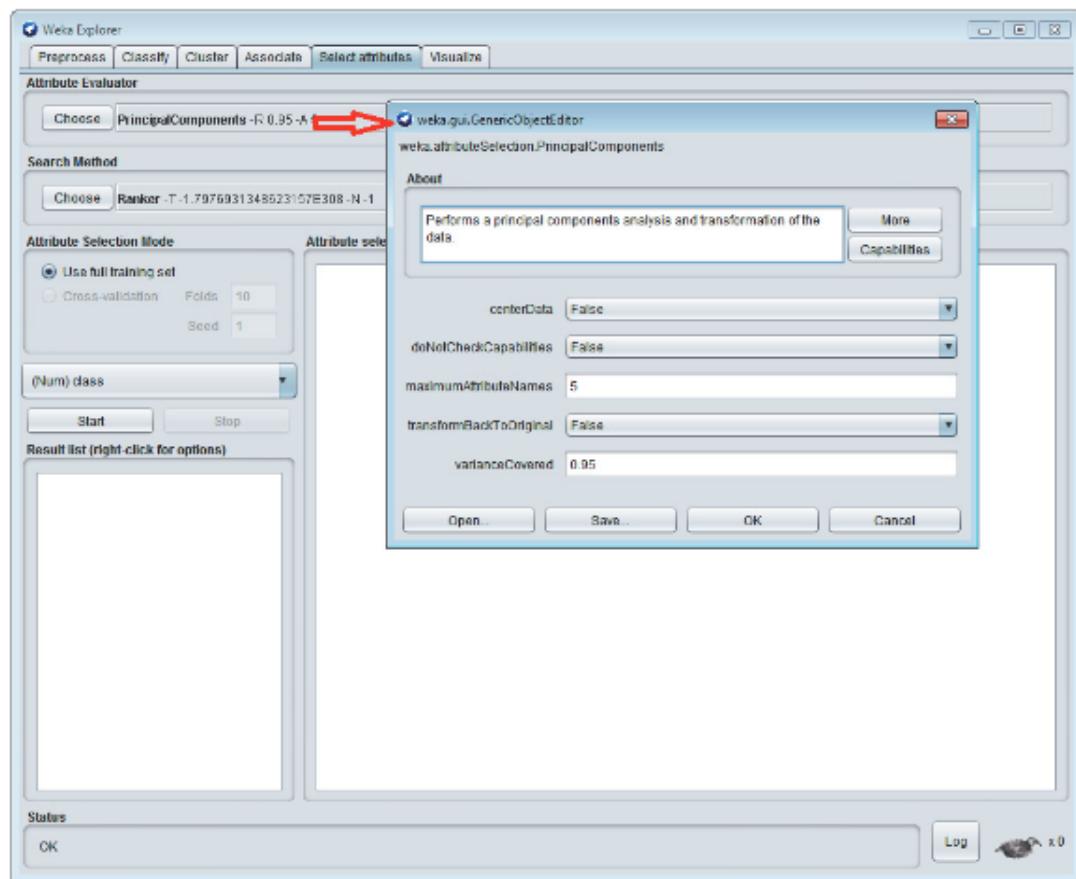
OneR sınıflandırıcısı tarafından benimsenen basit doğruluk ölçümünü kullanır. OneR'nin yaptığı gibi değerlendirme için eğitim verilerini kullanabilir veya dahili çapraz doğrulamayı uygulayabilir. Kat sayı değeri (*folds*) kullanıcı tanımlı bir parametre olarak verilir. OneR'nin basit ayriklaştırma yöntemini benimser. Minimum kepçe boyutu (*minimumBucketSize*) bir parametre olarak girilebilir.



Şekil 4.129. OneRAttributeEval Algoritması ve Parametreleri

PrincipalComponents

Diğer tek nitelikli değerlendirmelerin aksine, PrincipalComponents nitelikler kümesini dönüştürür. Yeni nitelikler, özdeğerleri sırasına göre sıralanır. İsteğe bağlı olarak, belirli bir oranın varyansını (*varsayılan olarak %95*) hesaba katmak için yeterli özvektörler seçerek bir alt küme oluşturulur. Son olarak, azaltılmış veriler orijinal alana geri dönüştürülebilir.



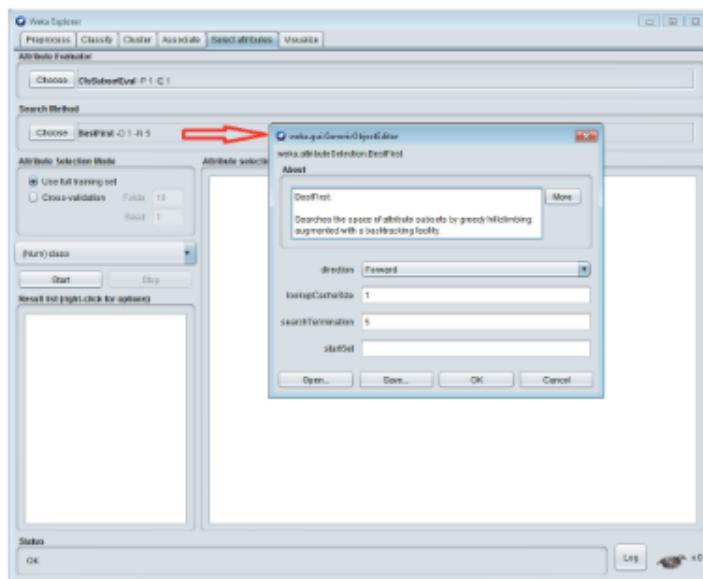
Şekil 4.130. PrincipalComponents Algoritması ve Parametreleri

4.9.2. Arama Yöntemleri

Arama yöntemleri, iyi bir alt kümesi bulmak için özellik alanını çaprazlar. Kalite, seçilen özellik alt kümesi değerlendircisi tarafından ölçülür. Her arama yöntemi WEKA'nın nesne editörü ile yapılandırılabilir.

BestFirst

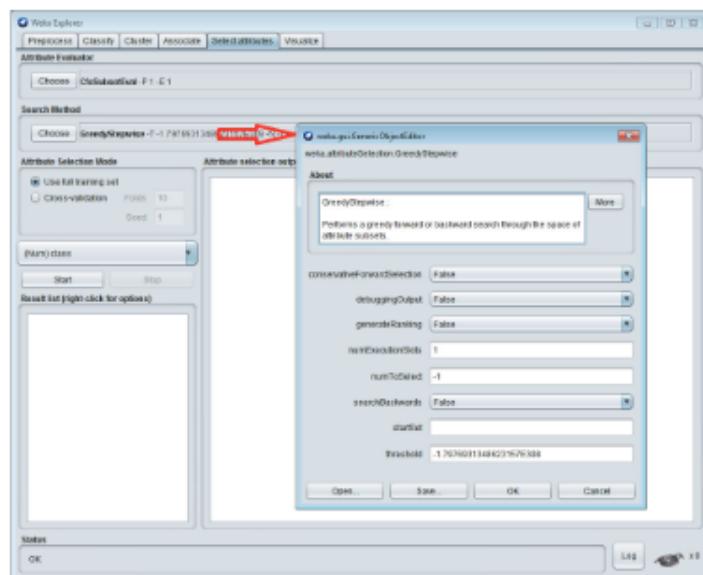
Backtracking ile açgözlü tepe tırmanışı gerçekleştirir. Sistem geri yüklemelerinden önce ardışık olmayan onaylanmamış düğümlerin kaçının bulunması gereği belirtilebilir. Boş nitelik kümesinden ileriye doğru, tam kümeden geriye doğru arama yapabilir veya bir ara noktasında (*nitelik dizinlerinin bir listesiyle belirtilir*) başlayabilir ve olası tüm tek nitelikli eklemler ve silme işlemlerini dikkate alarak her iki yönde arama yapabilir. Değerlendirilen alt kümeler verimlilik için önbelleğe alınır. Önbellek boyutu kullanıcı tarafından değiştirilebilen bir parametredir.



Şekil 4.131. BestFirst Algoritması ve Parametreleri

GreedyStepwise

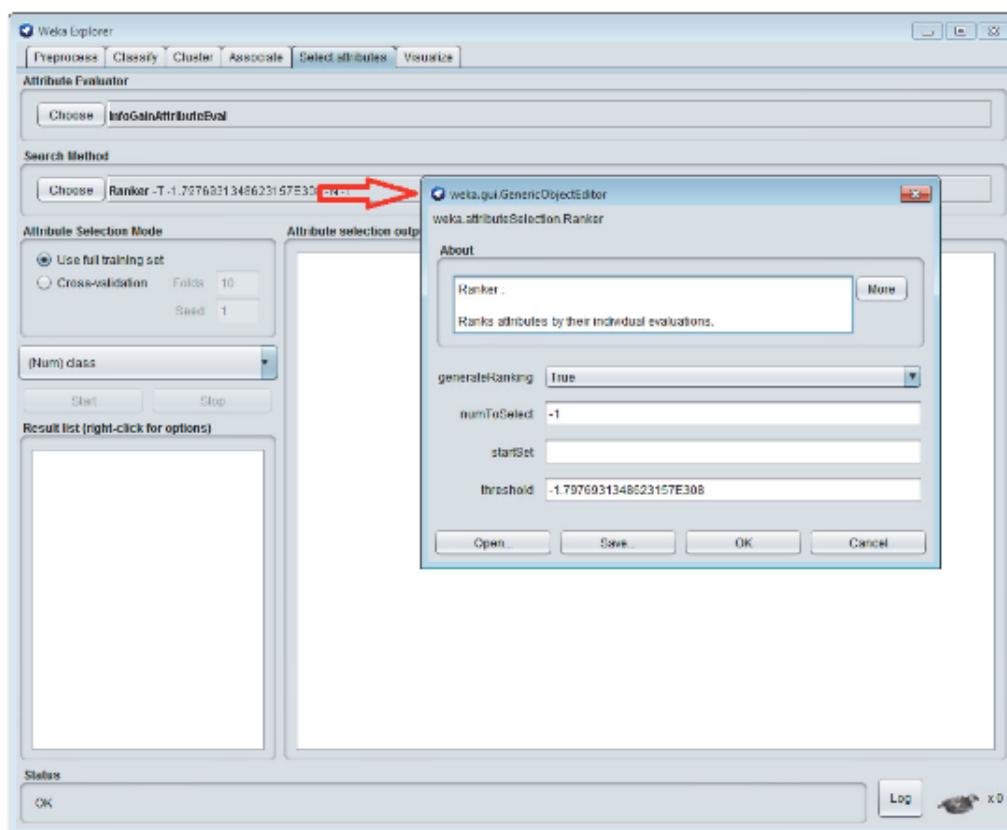
Nitelik alt kümelerini uzayda açgözlülükle arar. BestFirst gibi, boş kümeden ileriye veya tam kümeden geriye doğru ilerleyebilir. BestFirst'ten farklı olarak, geriye dönüş yapmaz, kalan en iyi niteliğin eklenmesi veya silinmesi, değerlendirme metriğini azalttığı anda sona erer. Alternatif bir modda, boşluğu boşluktan boşluğa (veya tam tersi) geçerek ve niteliklerin seçildiği sırayı kaydederek nitelikleri sıralar. Niteliklerin atıldığı bir eşigi korumak veya ayarlamak için niteliklerin sayısı kullanıcı tarafından belirtilebilir.



Şekil 4.132. GreedyStepwise Algoritması ve Parametreleri

Ranker

Daha önce belirtildiği gibi, nitelik alt kümeleri için bir arama yöntemi değil, bireysel nitelikler için bir sıralama düzenidir. Nitelikleri kendi değerlendirmelerine göre sıralar ve bir nitelik alt kümesi değerlendircisi değil, tek nitelikli değerlendircilerinden biri ile birlikte kullanılmalıdır. Sıralayıcı sadece nitelikleri sıralamakla kalmaz, aynı zamanda alt sıradakileri kaldırarak nitelik seçimini gerçekleştirir. Niteliklerin atıldığı bir kesme eşiği veya kaç tane niteliğin saklanacağı belirlenebilir. Sıralamalarından bağımsız olarak saklanması gereken belirli nitelikler belirtilebilir.

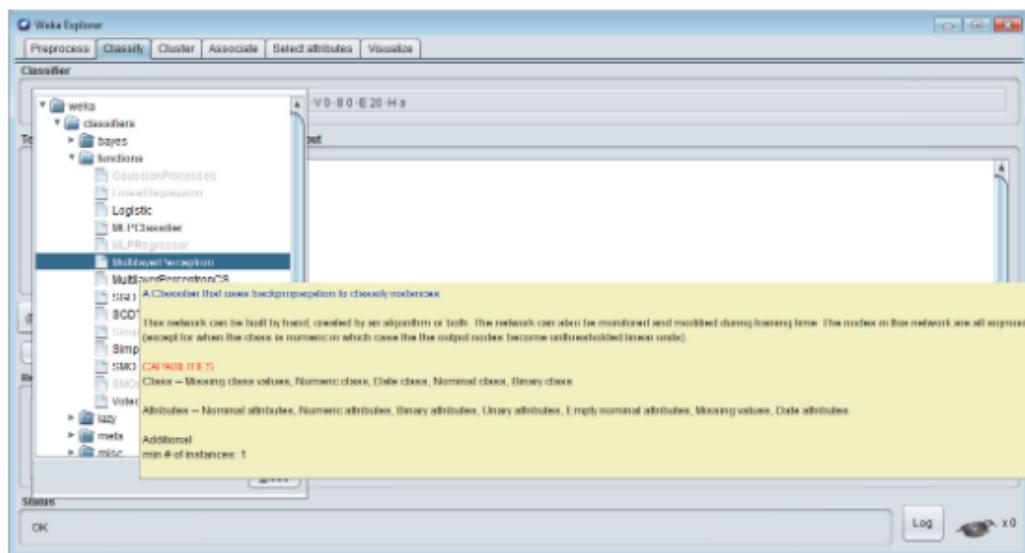


Şekil 4.133. Ranker Algoritması ve Parametreleri

5

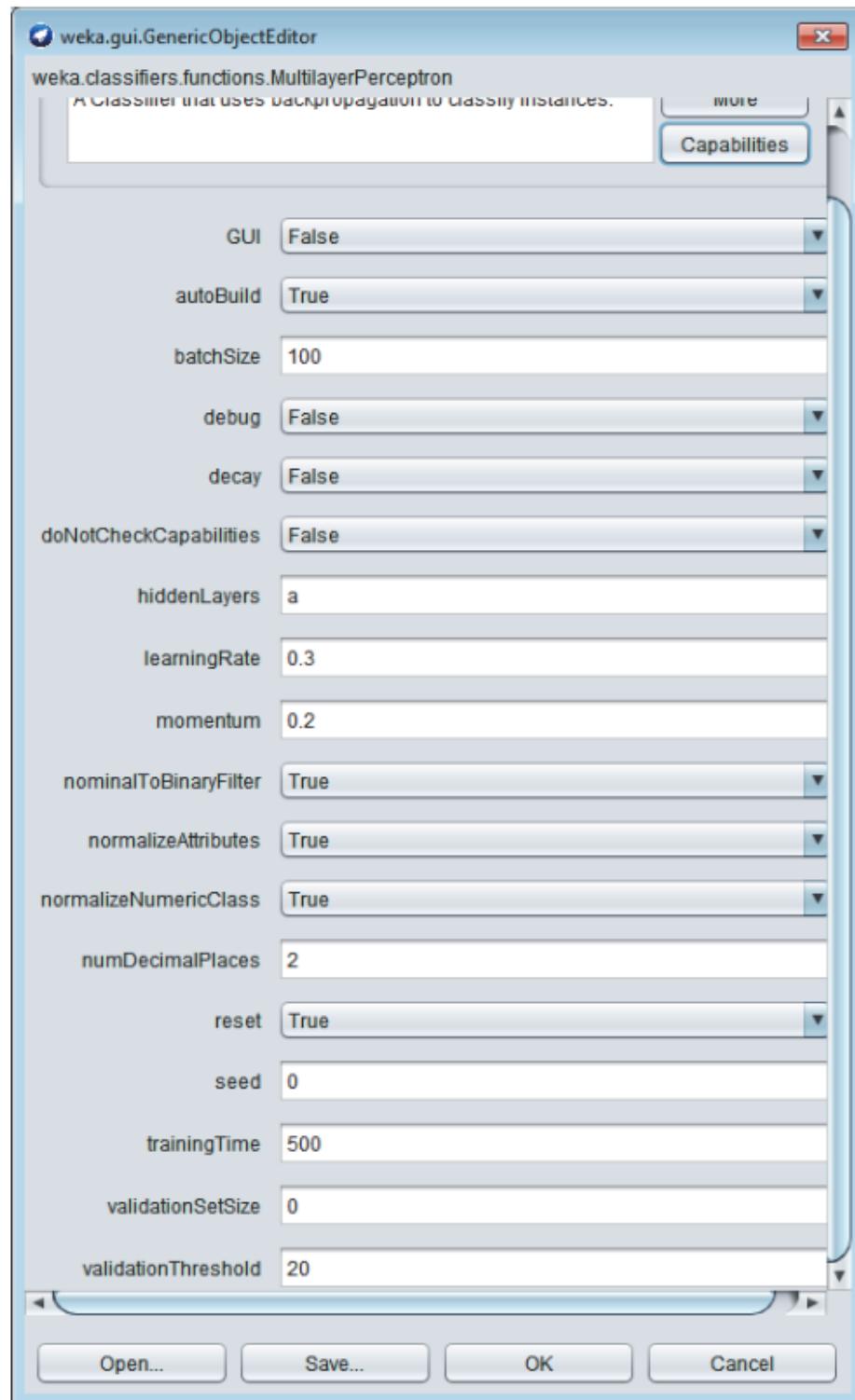
Weka ile Yapay Sinir Ağları

Çok katmanlı geri yayılımlı yapay sinir ağları (*Multilayer Perceptron*) sınıflandırma sekmesi içinde fonksiyonlar (*functions*) grublaması altında yer almaktadır. Aşağıdaki şekilde çok katmanlı yapay sinir ağlarının seçimi gösterilmiştir.



Şekil 5.1. Çok Katmanlı Yapay Sinir Ağları Seçimi

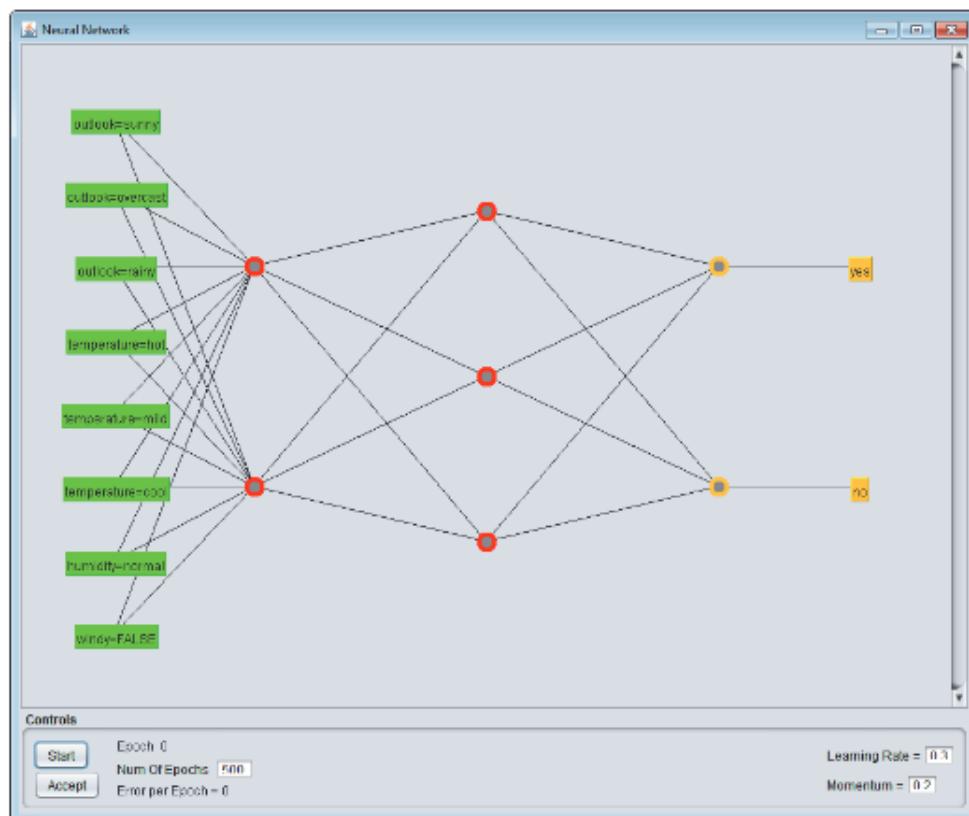
2 numaralı başlık içerisinde yapay sinir ağları ile ilgili detaylı bilgi verilmiştir. Yapay sinir ağları ile ilgili detaylı bilgi edinmek için o bölümün okunması önem arz etmektedir. Bu algoritmanın birçok parametresi bulunmaktadır. Bu parametrelerin neler olduğu ile ilgili aşağıdaki şekil incelenebilir.



Şekil 5.2. Geri Yayımlı Yapay Sinir Ağları Parametreleri

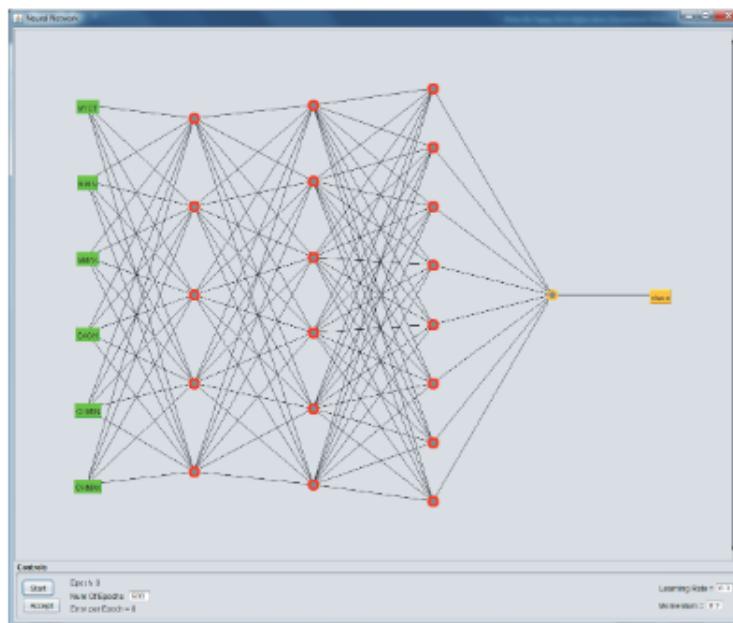
5.1. GUI Ekranı

Yapay sinir ağlarının modeli görsel olarak oluşturulmak istenirse bunun için GUI adlı parametre “True” olarak seçildiğinde Başlat (*Start*) butonuna basılıncaya iç katmanların kullanıcı tarafından tasarlabilirceği bir ekran gösterilecektir. Ekran aşağıdaki gibidir.



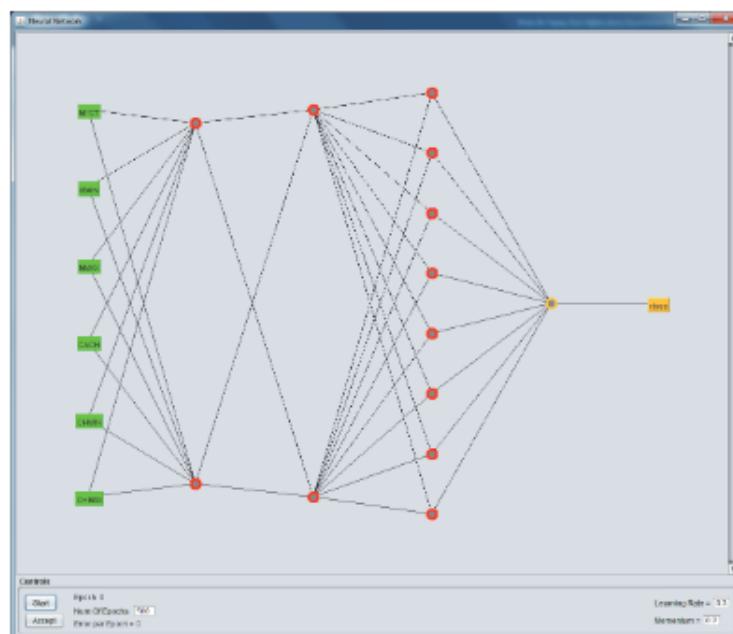
Şekil 5.3. Yapay Sinir Ağları GUI Ekranı

Yukarıdaki şekilde iki katmanlı bir modelin birinci katmanında iki adet sinir veya diğer bir adıyla düğüm ikinci katmanında ise üç adet sinir bulunmaktadır. Kullanıcı tarafından bu ekrandaki sınırlar fare imleci ile istenildiği gibi tasarlabilir. Yeşil dikdörtgen kutular için alınmış olanlar giriş katmanının ifade ederken turuncu renkteki dikdörtgenler içine alınan ve en sonda bulunanlar ise çıkış katmanıdır. Kategorik değişkenlerin tahmin edilmesi durumunda kategori elemanlarının her biri ayrı bir çıkış olarak gösterilir. Dolayısıyla çıkış katmanındaki sinir sayısı kategorideki eleman sayısı ile aynı olacaktır. Sayısal değerdeki tahminlerde ise çıkış katmanındaki sinir yalnızca bir tane olur. Aşağıdaki şekilde bu durum gösterilmiştir.



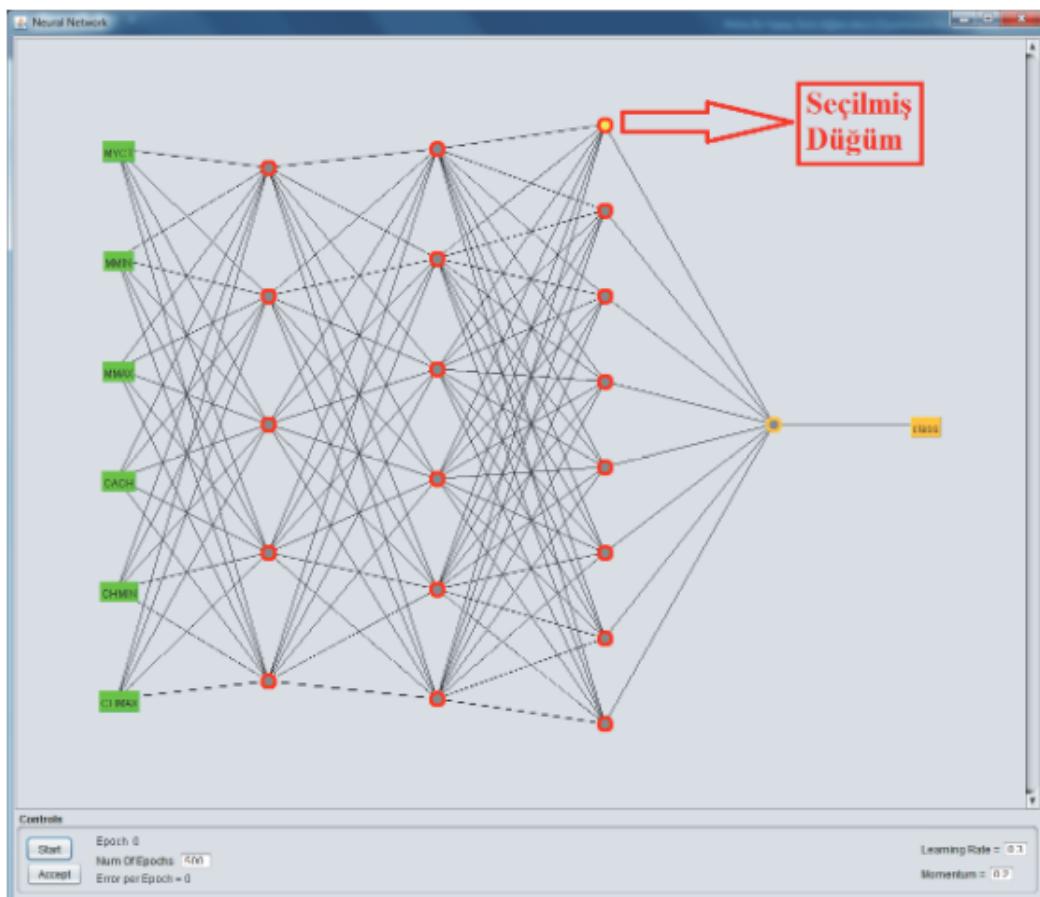
Şekil 5.4. Sayısal Türde Tahminler İçin Çıkış Katmanı

Açılan ekranda sinirlerin ve bağlantılarının yapısı Başlat (*Start*) butonuna basılmadan önce istenilen şekilde değiştirilebilir. Sinirler seçilebilir veya seçilmiş olanlar kaldırılabilir. Seçim kaldırılmak istenirse farenin sağ tıklanması yeterlidir. Aşağıdaki şekilde bir önceki şekilde görünen ağ yapısının ortasındaki kırmızı yuvarlaklar ile görünen düğümler seçilmiş ve sağ tıklanarak kaldırılmıştır. Sonuca aşağıdaki ağ yapısı ortaya çıkmıştır.



Şekil 5.5. Bazı Düğümleri Silinmiş Ağ Yapısı

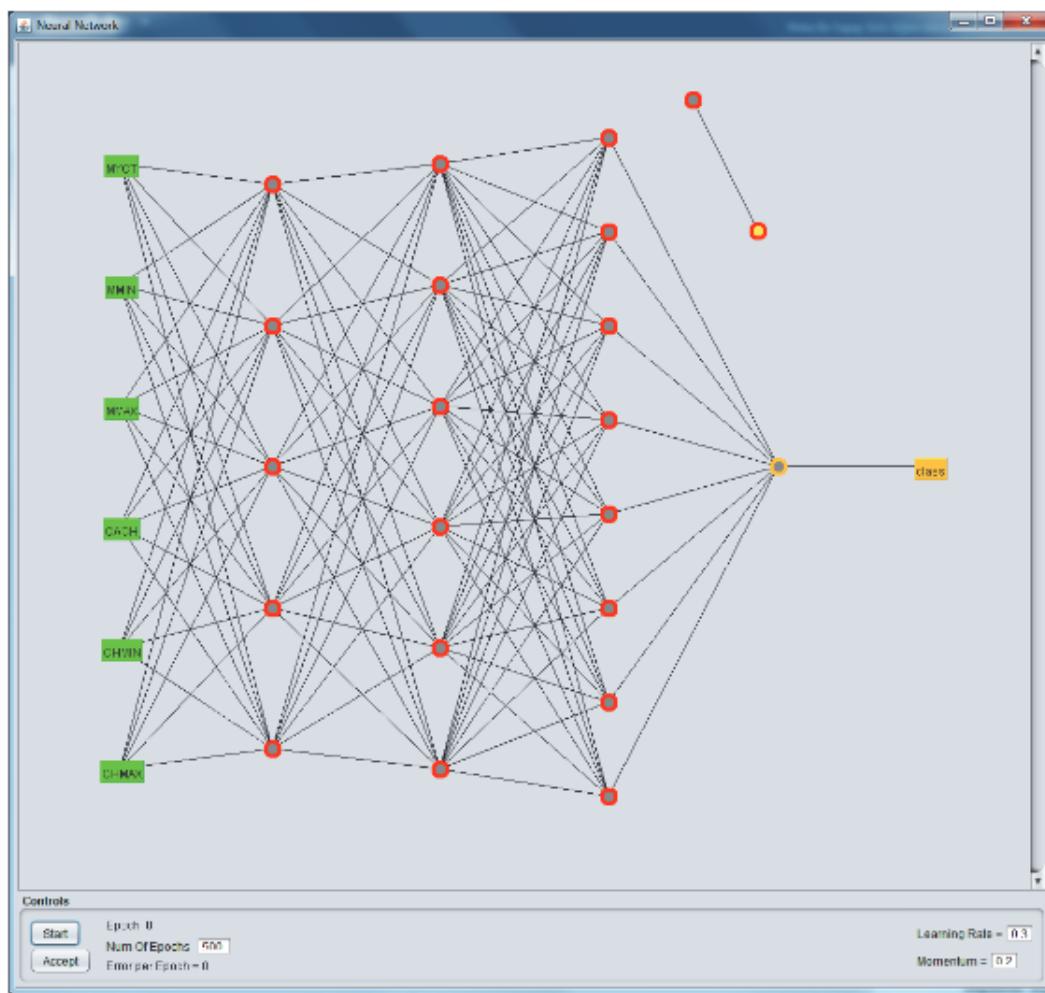
Ağ yapısı içerisindeki kırmızı renkte çerçevesi olan ve gri renkli içi dolu yuvarlak olarak gösterilenler her bir siniri veya diğer bir adıyla düğümü göstermektedir. Düğümler arasındaki bağlantılar ise siyah renkte çizgiler ile gösterilmektedir. Herhangi bir düğümü seçmek için kırmızı yuvarlaklara fare ile sol tıklamak yeterlidir. Seçimin yapıldığı yuvarlığın içindeki gri rengin sarı renk olması ile görselleştirilir. Bu seçim kaldırılmak istenirse ekranın boş bir noktasında fare ile sağ tıklamak gereklidir. İçi gri renkli olan düğümler seçili olmadıklarını gösterir. Aşağıdaki şekilde bir düğümün seçilmiş hali gösterilmiştir.



Şekil 5.6. Ağ Yapısında Herhangi Bir düğümün Seçilmiş Hali

Düğüm ekleme istenirse öncelikle ekranın herhangi bir boş kısmına sağ tıklanıp herhangi bir seçim yapılmaması sağlanır. Sonrasında ekranın herhangi bir boş kısmına fare ile sol tıklandığında yeni bir düğüm eklenecektir ve eklenen yeni düğüm seçili geleceğinden dolayı içi sarı renkte olacaktır. İki düğüm arasında bağlantı oluşturmak istenirse başlangıç düğümü fare ile sağ tıklanarak seçilir ve ardından bir sonraki bağlanacak düğüm fare

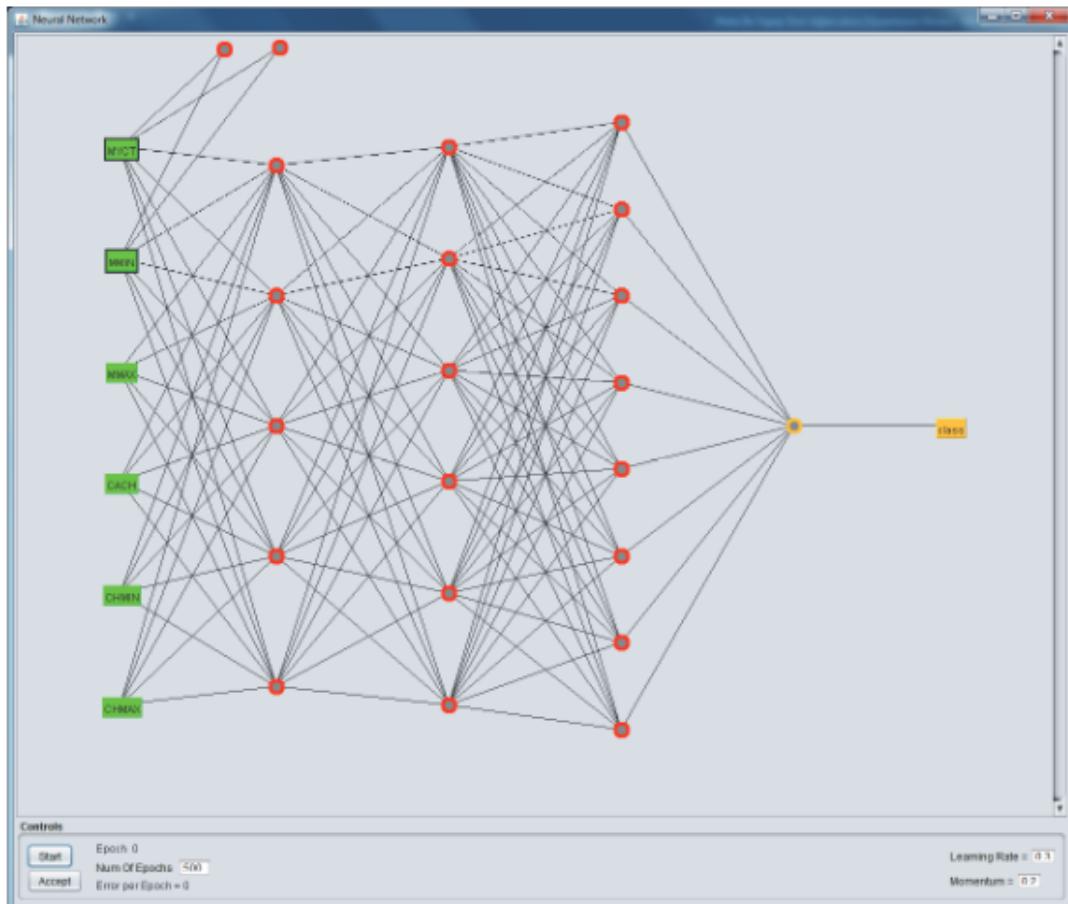
ile sol tıklanır. Aşağıdaki şekilde gösterildiği gibi iki düğüm birbirine bir çizgi aracılığıyla bağlanmış olacaktır. Bu işlem sonrası ekranın herhangi bir boş kısmına fare ile sol tıklama yeni bir düğüm eklemeye ve onun da en son seçili olan düğüme bağlanmasına neden olur.



Şekil 5.7. İki Düğümün Birbirine Bağlanması

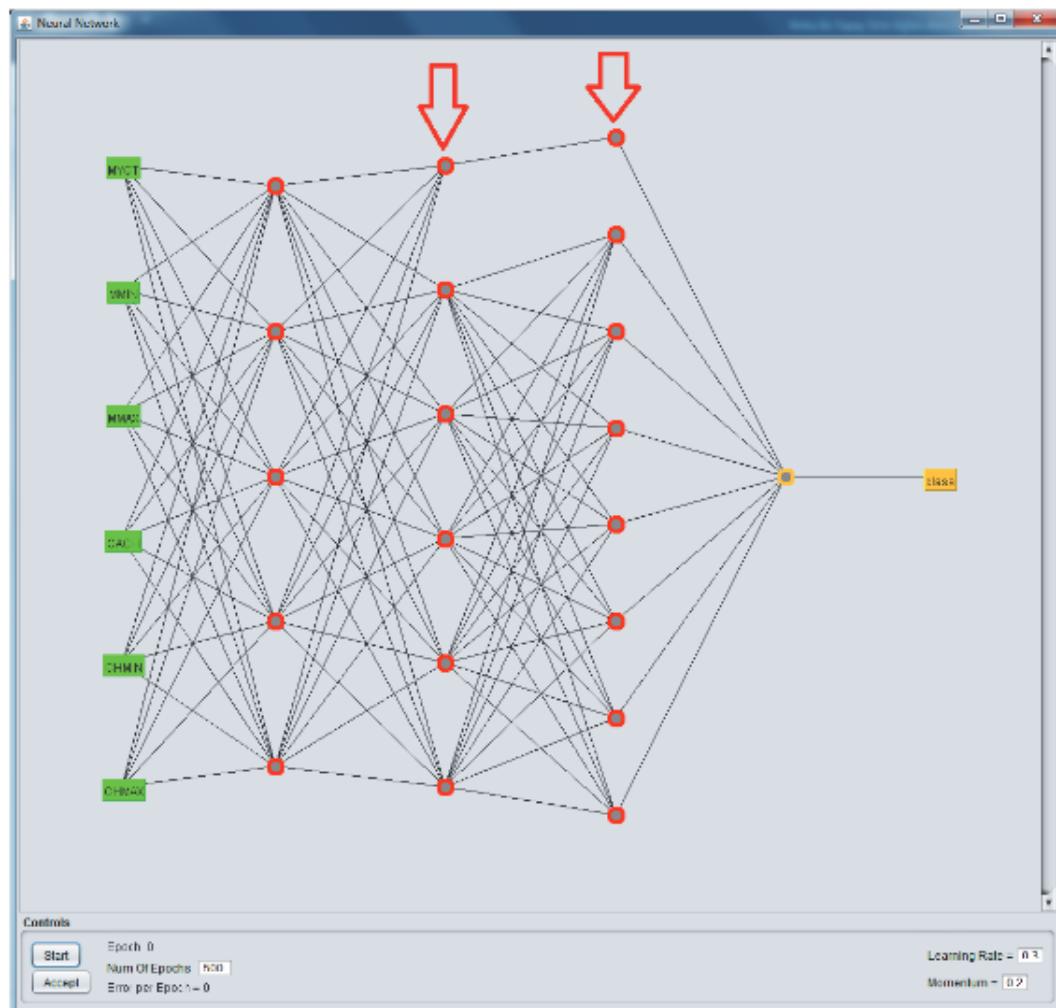
Buradaki iki düğümün birbirine bağlanması gibi bu bağımsız düğümler diğer var olan giriş katmanındaki, çıkış katmanındaki veya gizli katmandaki herhangi bir düğüme de bağlanabilir. CTRL tuşu basılı iken farenin sol tıklanması birden fazla düğümün seçilmesini sağlar. Sonrasında herhangi bir düğüme tıklanması ise seçili olan tüm düğümlerin son tıklanan düğüme bağlanması sağlanır. Eğer birden fazla seçili düğümden sonra ekranın boş bir yerine fare sol tıklanırsa oluşturulacak yeni düğüme seçili olan düğümlerin tümü bağlanır. Aşağıdaki şekilde giriş katmanındaki ilk iki düğüm CTRL tuşu basılı iken seçilmiş ve sonrasında eklenen iki yeni düğüme tıklanarak

ikisinin de bu düğümlere bağlanması sağlanmıştır. Bu şekilde istenirse tüm gizli katman baştan tasarlabilir.



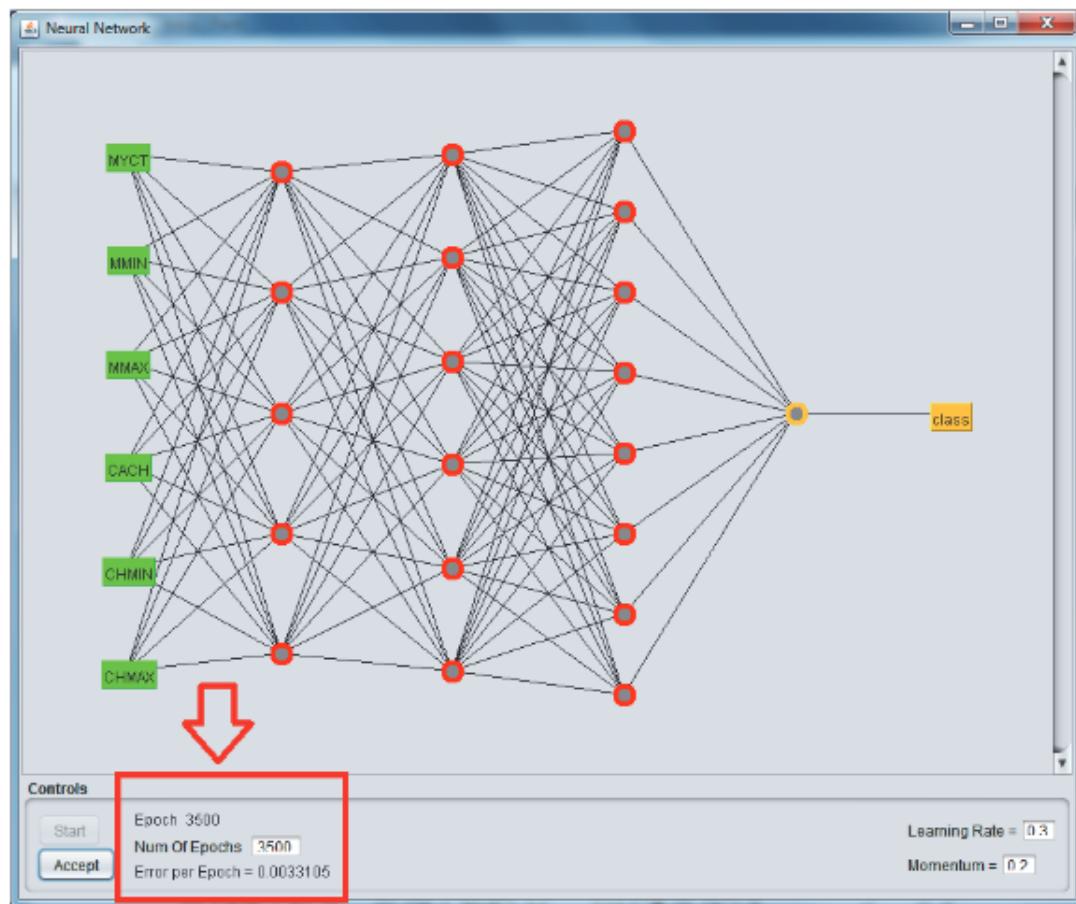
Şekil 5.8. Giriş Katmanındaki Düğümlerin Yeni Düğümlere Bağlanması

Bağlantılar arasında yön gösterilmemesine rağmen bağlantılar yönlendirmeli özelliğidir. Herhangi bir düğüm kaldırılmak istenirse öncelikle hiçbir düğüm seçili olmaması durumunda ilgili düğümü fare ile sağ tıklandığında hem bu düğümün kendisi hem de sahip olduğu tüm bağlantılar kaldırılır. Yalnızca iki düğüm arasındaki bağ kaldırılmak istenirse öncelikle birinci düğüm seçilir ve ardından bağlantısı olan düğüme fare ile sağ tıklanır. Bu yalnızca ilgili iki düğümün bağını kaldıracaktır. Aşağıdaki şekilde gösterilen iki düğümün diğer düğümler ile olan tüm bağlantıları kaldırılmış ve yalnızca kendi aralarındaki bağlantı bırakılmıştır.



Şekil 5.9. İki Düğüm Arasındaki Bağları Kaldırmak

Ağ yapısının oluşturulduğu bu ekranada istenirse öğrenme oranı (*LearningRate*), momentum katsayısı (*momentum*) ve verilerin kaç defa işleneceği anlamına gelen devir sayısı (*epochs*) parametre olarak verilebilir. Başlat (*Start*) butonuna tıklanmasıyla ağ yapısı öğrenme işlemini başlatır. Devir sayısı (*epochs*) ve her bir devirdeki (*epochs*) hata (*Error per Epochs*) ekranın sol alt köşesinde işlem boyunca güncellenerek gösterilir. Aşağıdaki şekilde 3500 devir sayısı (*epochs*) için son değerler gösterilmiştir. Buradaki değerlerin her bir devir sayısı için gösterildiği ve en son gösterilen değerlerin ise en son devir sayısına (aşağıdaki şekil için 3500. devir) ait olduğu gözden kaçmamalıdır.



Şekil 5.10. Devir Sayısı ve Her Devirdeki Hata Gösterimi

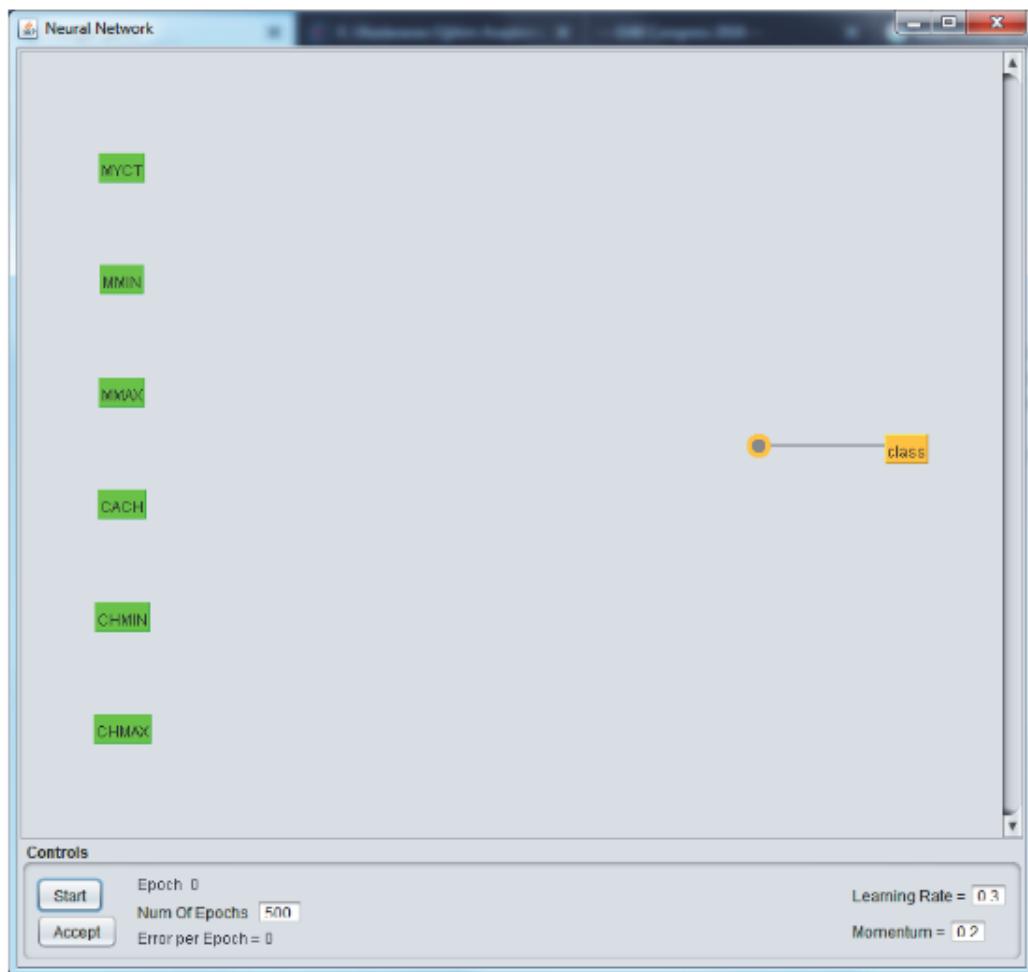
Sayısal değerler tahmin edilirken değerlerin normalize edilip edilmemesi hata değerini doğrudan etkiler. Ağ yapısı belirlenen devir sayısına ulaştığında durur. Bu durumda kullanıcı oluşan ağ yapısını kabul ederse Kabul Et (*Accept*) butonuna basabilir veya devir sayısı ve diğer parametreler istenilen şekilde değiştirildikten sonra tekrar Başlat (*Start*) butonu ile öğrenme işlemi tekrarlanabilir. Eğitim ve test verilerinin ayırtılması çapraz doğrulama katsayısı (*folds*) ile yapılmış ise her bir çapraz doğrulama için Başlat (*Start*) butonu ve Kabul Et (*Accept*) butonu kullanılmalıdır.

5.2. Parametre Değerleri

5.2.1. Otomatik Oluşturma (*AutoBuild*)

Ağın otomatik olarak oluşturulup oluşturulmayacağı veya kullanıcıya bırakılmış olup olmayacağı belirler. Yani kullanıcı tarafından GUI arayüzünden oluşturulan iç katman yapısının mı yoksa otomatik olarak oluş-

turulan ağ yapısının mı çalışacağı belirlenir. Bu özelliğin false yapılması durumunda GUI arayüzü aktif edilirse açılan ekranda hiçbir iç katmanın olmadığı ve kullanıcı tarafından tasarlanmasının bekleniği görülecektir. Aşağıdaki şekilde GUI özelliği aktif olarak seçili iken otomatik oluşturulma (*AutoBuild*) özelliği pasif olarak seçilmiş bir ağ yapısı gösterilmiştir.

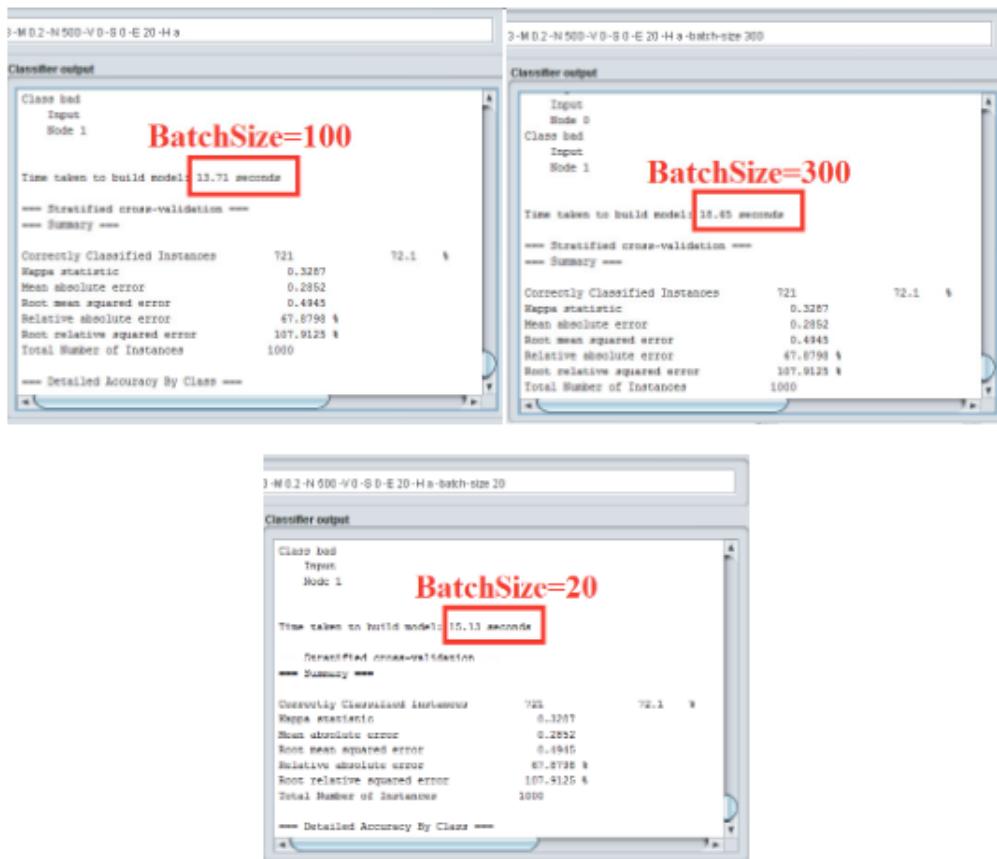


Şekil 5.11. Otomatik Oluşturma (*AutoBuild*) Özelliği Pasif Yapılmış Ağ Yapısı

5.2.2. Yiğin Boyutu (BatchSize)

Yiğin tahmini gerçekleştirildiğinde işleminden geçirilecek örnek sayısını ifade eder. Daha fazla veya daha az örnek sağlanabilir. Ancak bu durumun uygulamanın öğrenme süresine doğrudan bir etkisi olduğu unutulmamalıdır. Aşağıdaki şekilde 1000 örnek sayısına sahip credit-g.arff dosyasına ait yiğin boyutunun 100 ile 300 olması arasındaki öğrenme süreleri gösterilmiştir. Yiğin boyutunun öğrenme işleminin başarısına bir etkisi olmamasına rağmen öğrenme süresine doğrudan bir etkisi olduğu

görmektedir. Fakat bu etki yığın miktarı arttıkça sürenin uzayacağı anlamına gelmemektedir.



Şekil 5.12. Yığın Boyutu Karşılaştırması

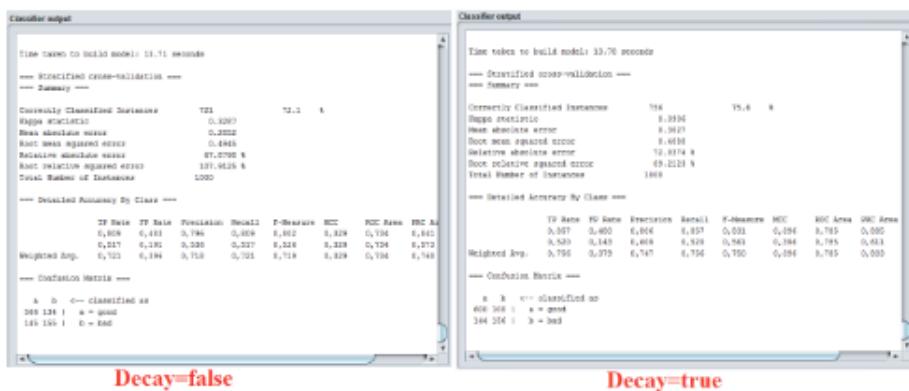
5.2.3. Ayıklama (debug)

Bu parametre varsayılan olarak pasif (*false*) olarak ayarlanmıştır. Aktif (*true*) edilmesi explorer ekranında herhangi bir değişiklik sağlamaz. Fakat konsol (*console*) dediğimiz kod ekranı ile çalışıldığında ve bu parametre aktif (*true*) olarak seçilmiş ise sınıflandırcı ekranda ek bilgileri gösterebilir.

5.2.4. Zayıflama (decay)

Aktif (*true*) edilmesi ile öğrenme oranının (*learningRate*) azalmasını sağlar. Bu, mevcut öğrenme oranının ne olacağını belirlemek için başlangıç öğrenme oranını (*learningRate*) devir sayısına (*epoch*) bölecektir. Bu durum ağın çıkış hedefinden uzaklaşmasını engellemeye yardımcı olmanın yanı sıra ağın ağın genel performansını iyileştirir. Öğrenme oranının azaltılması GUI ekranında gösterilmeyeceği unutulmamalıdır. Yalnızca orijinal öğrenme

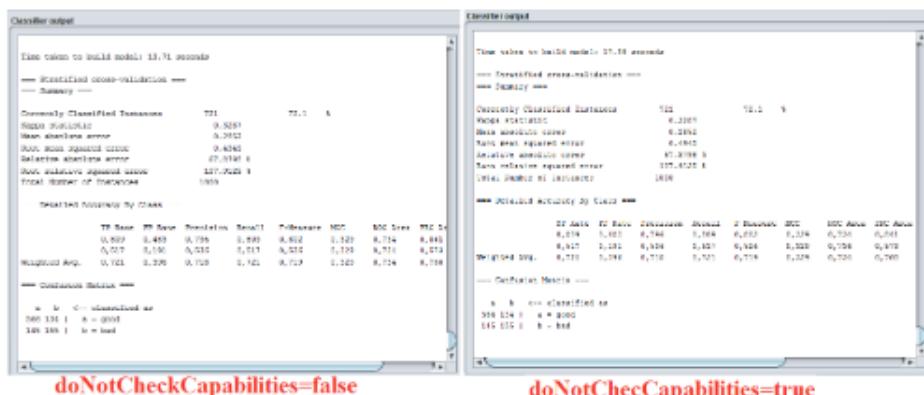
oranı azaltılacaktır. Öğrenme oranının GUI arayüzünde değiştirilmesi yalnızca başlangıç öğrenme oranını değiştirir. Aşağıdaki şekilde credit-g.arff dosyası üzerinde zayıflama (*decay*) özelliğinin aktif edilmesi ve pasif edilmesi sonrası yapılan analiz sonuçları gösterilmiştir. Hem öğrenme süresinin hem de başarı oranının etkilendiği görülmektedir.



Şekil 5.13. Zayıflama Özelliği Aktif ve Pasif Yapılmış Analiz Sonuçları

5.2.5. Verileri Kontrol Etme (*doNotCheckCapabilities*)

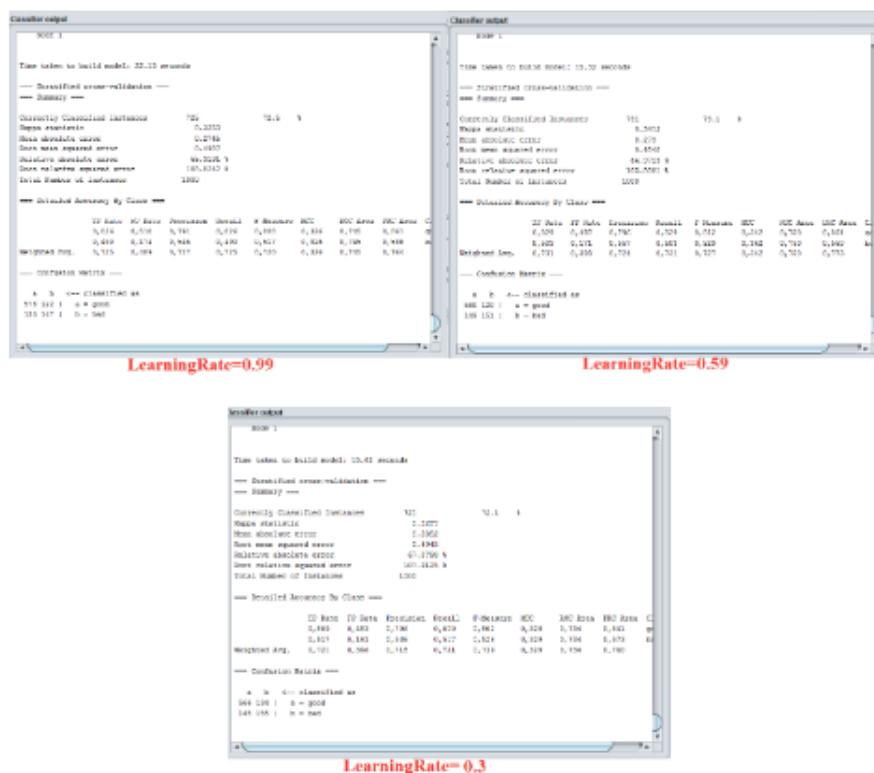
Bu değerin aktif edilmesi durumunda (*true*) sınıflandırıcı çalıştırılmadan önce yapılan niteliklerin ve verilerin kontrolünün yapılmaması sağlanır. Bu durum çalışma zamanını azaltır fakat dikkatli kullanılması önerilir. Aşağıda-ki şekilde credit-g.arff dosyası üzerinde verileri kontrol etme (*doNotCheckCapabilities*) özelliğinin aktif edilmesi ve pasif edilmesi sonrası yapılan analiz sonuçları gösterilmiştir. Değerin aktif edilmesi durumunda analizin öğrenme süresinin negatif olarak etkilendiği ama başarı oranının etkilenmediği görülmektedir.



Şekil 5.14. Verileri Kontrol Etme Özelliği Aktif ve Pasif Yapılmış Analiz Sonuçları

5.2.6. Öğrenme Oranı (*learningRate*)

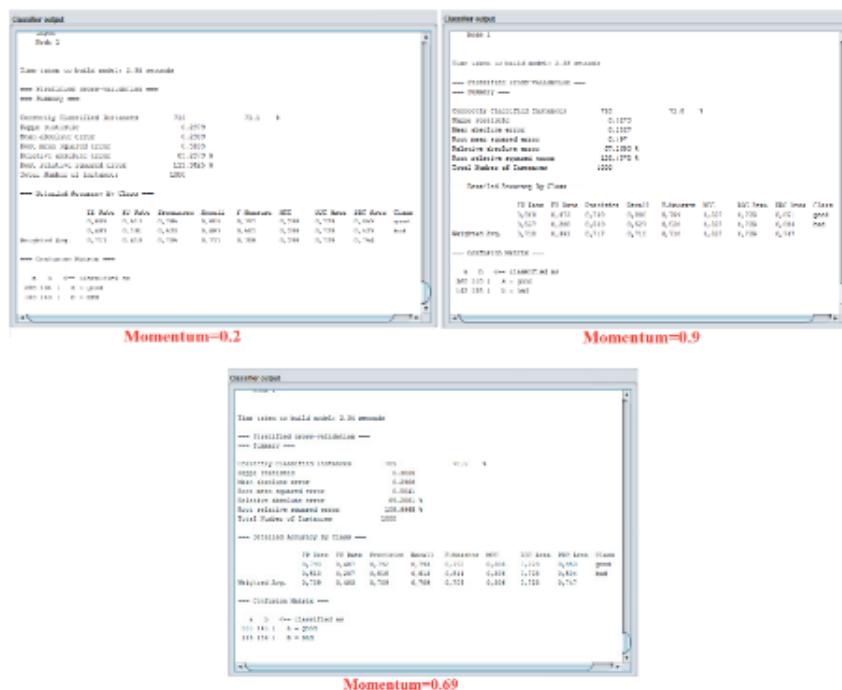
Ağırlık değerleri burada belirtilen oran ile güncellenir. Bu değere her yerden erişildiği ve birden fazla yerde kullanıldığı unutulmamalıdır. Dolayısıyla bu değer tüm ağın çalışmasını etkiler. 0'dan büyük ve 1'den küçük değerleri alır. Yüksek iterasyon sayısı ile düşük öğrenme oranının kullanılması daha ölçülü olacaktır. Aşağıdaki şekilde credit-g.arff dosyası üzerinde 0.3, 0.59 ve 0.99 öğrenme oranı değerleri için analiz sonuçları gösterilmiştir. Dikkat edilirse öğrenme oranı değerlerinin değişmesi hem ağın öğrenme süresini hem de ağın başarı oranını verilen değerler doğrudan etkilemektedir.



Şekil 5.15. Farklı Öğrenme Oranı Değerleri İçin Analiz Sonuçları

5.2.7. Momentum (*momentum*)

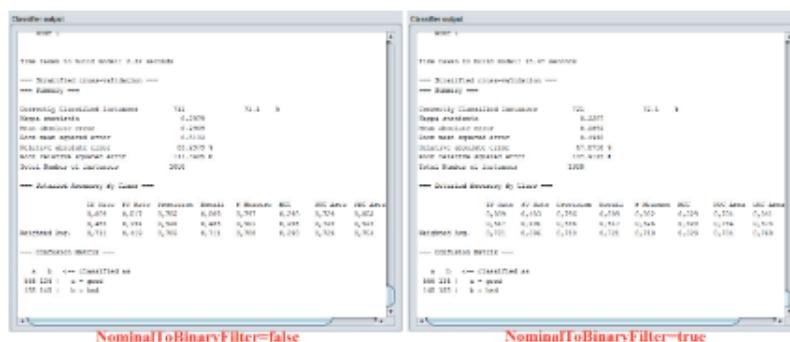
Ağın sinirlerindeki ağırlık değerleri güncellenirken momentum değeri de kullanılır ve bu şekilde ağırlık değeri güncellenir. Aşağıdaki şekilde credit-g.arff dosyası üzerinde 0.2, 0.9 ve 0.69 momentum değerleri için analiz sonuçları gösterilmiştir. Öğrenme süresini kısaltmak için ikili dönüşüm özelliği kapatılmıştır. Dikkat edilirse momentum değerlerinin değişmesi hem ağın öğrenme süresini hem de ağın başarı oranını verilen değerler doğrudan etkilemektedir.



Şekil 5.16. Farklı Momentum Değerleri İçin Analiz Sonuçları

5.2.8. Kategorik Verileri İkili Verilere Dönüşürme (*nominalToBinaryFilter*)

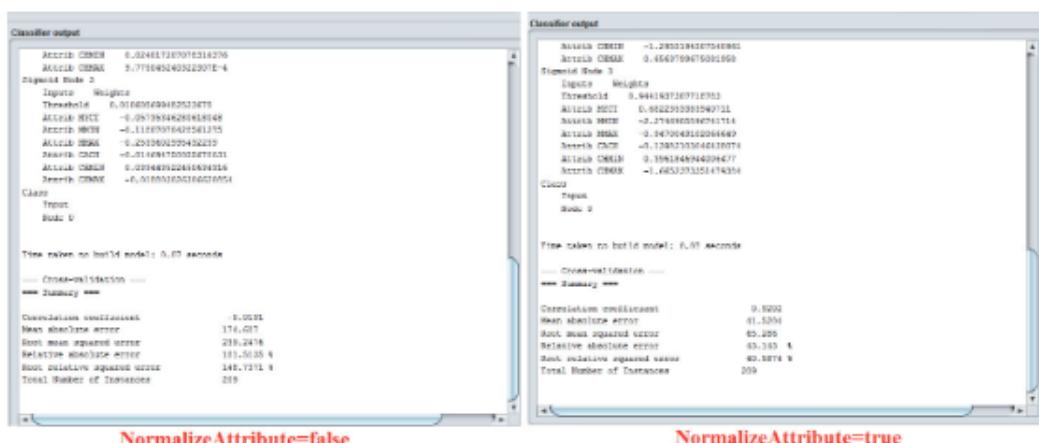
Bu özelliğin aktif edilmesi (*true*) durumunda veriler içerisinde kategorik (*nominal*) özellikte veriler bulunması durumunda bunları ikili formlara dönüştürecek bir filtre ekler. Ağ çalışmadan önce bu filtre çalıştırılarak veriler ön işlemeye tabi tutulur. Kategorik değerlerin ikili değerlere dönüştürülmesi ağın performansına da pozitif katkısı olacaktır. Aşağıdaki şekilde creditg.arff dosyasına ikili türde dönüştürme uygulanmış ve uygulanmamış analiz sonuçları gösterilmiştir. Dikkat edilirse ikili türde dönüştürülme özelliği kapatılırsa (*false*) ağın başarısı düşmüştür. Fakat ağın başarısının düşmesinin yanısıra ağın öğrenme süresinin çok kısalığı da görülmektedir.



Şekil 5.17. Kategorik Verilerin İkili Türe Dönüştürülmüş ve Dönüştürülmemiş Analiz Sonuçları

5.2.9. Verileri Normalize Etme (*normalizeAttributes*)

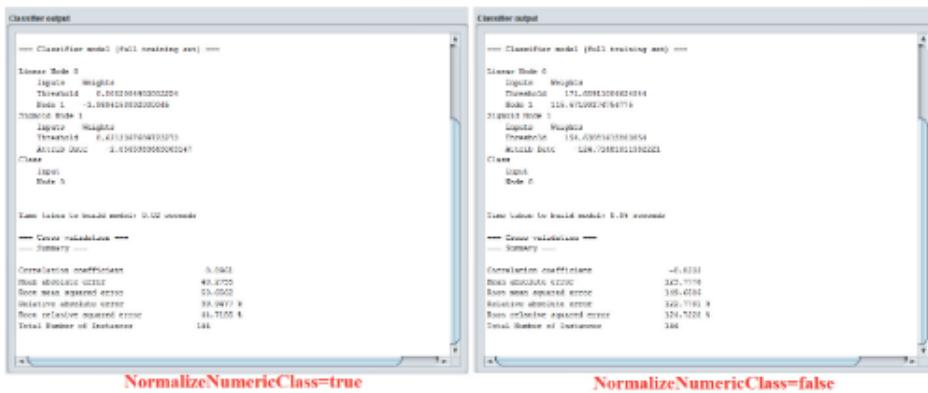
Aktif edilmesi (*true*) durumunda niteliklerin değerlerini -1 ile +1 aralığına çekerek normalize eder. Bu durum ağın performansını artırmaya yardımcı olabilir. Yalnızca sayısal değerler normalize edilmez. Aynı zamanda kategorik (*nominal*) değerler de ikili değerlere dönüştürüldükten sonra normalleştirilecektir. Verilerin normalize edilmesi sayısal değer aralıklarının eşit olmaması durumunda üç değerlerin ağı yanlış yönlendirmesini de engeller. Normalize yapılmaması durumunda daha büyük değerler daha önemli değerler haline dönüşebilir. Aşağıdaki şekilde *cpu.arff* dosyası için normalize yapılması ve yapılmaması durumundaki ağ çıktıları görülmektedir. Dikkat edilirse hata değerleri normalize işlemi yapılmayınca çok yükselmektedir.



Şekil 5.18. Verileri Normalize Edilmiş ve Edilmemiş Analiz Sonuçları

5.2.10. Sayısal Tahmin Değerini Normalize Etme (*normalizeNumericClass*)

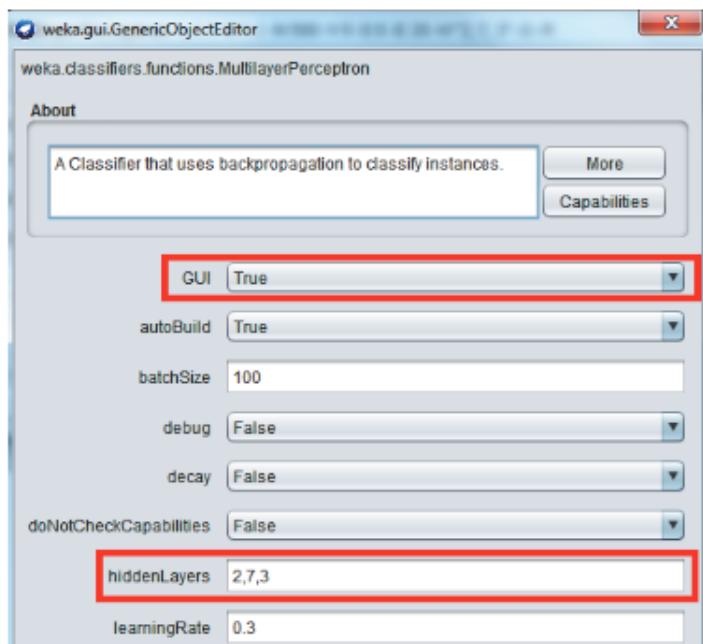
Tahmin edilecek sütunun (*niteliğin*) sayısal olması durumunda oradaki değerlerin normalize edilip edilmeyeceğini belirler. Bu durum ağın performansını doğrudan etkileyecektir. Aşağıdaki şekilde *airline.arff* dosyası için tahmin edilen sınıfın normalize edilmiş ve normalize edilmemiş analiz sonuçları gösterilmiştir. 144 tane verisi bulunan örnekte dikkat edilirse tahmin edilen sınıf değeri normalize edilmeyince hem ağın çalışma süresi hem de ağın başarı oranı negatif olarak etkilenmiştir. Fakat unutulmamalıdır ki bu nitelik yalnızca ağın öğrenme işlemine başlamadan önce normalize edilir ve en son olarak orijinal aralığına geri dönürülerek çıktıda sunulur.



Şekil 5.19. Tahmin Edilen Niteliğin Normalize Edilmiş ve Edilmemiş Analiz Sonuçları

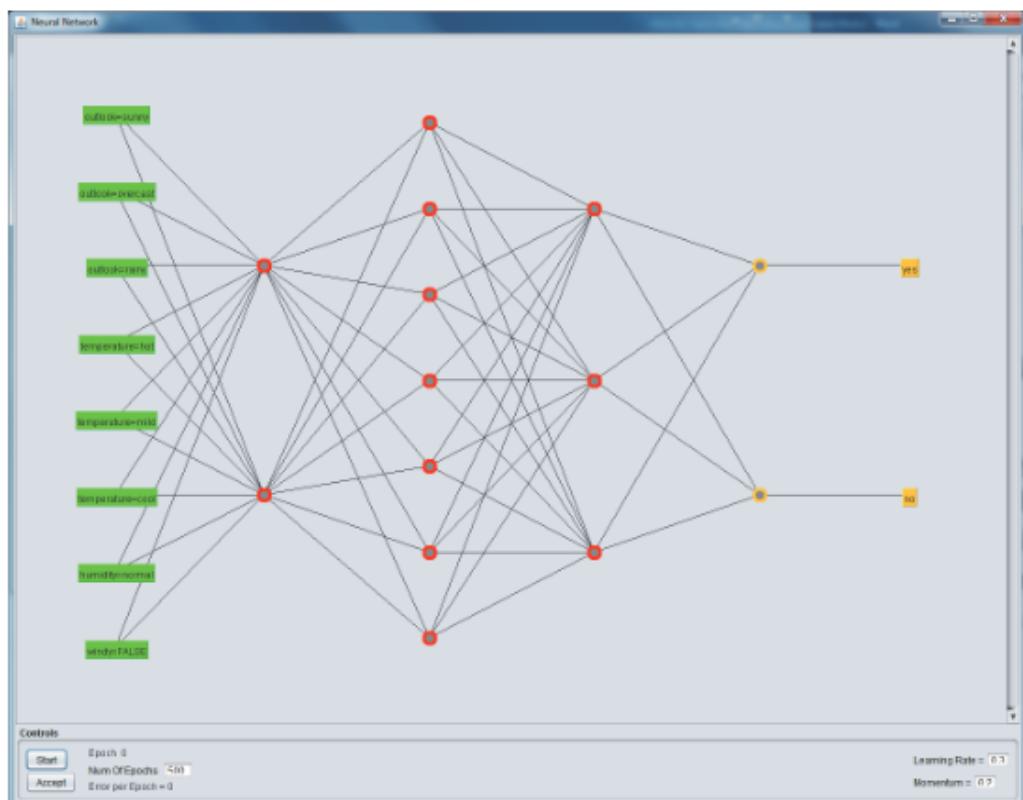
5.2.11. Gizli Katmanlar (*hiddenLayers*)

GUI Ekranı başlığı altında kullanıcıların ağ yapısının iç katman sayılarını ve bağlantılarını özel olarak ayarlayabileceği detaylı olarak anlatılmıştır. Weka parametreleri arasında iç katman sayılarının kullanıcı tarafından ve rilmesi ve bağlantılarının otomatik olarak yapılması mümkündür. Bunun için tek yapılması gereken gizli katman parametre (*hiddenLayers*) değerine iç katmanların sırayla ve aralarında virgül olacak şekilde yazılmasıdır. Aşağıdaki şekilde toplam üç katmanlı ve birinci katmanında iki, ikinci katmanında yedi, üçüncü katmanında ise üç sinir olacak bir ağ yapısı oluşturulmuştur.



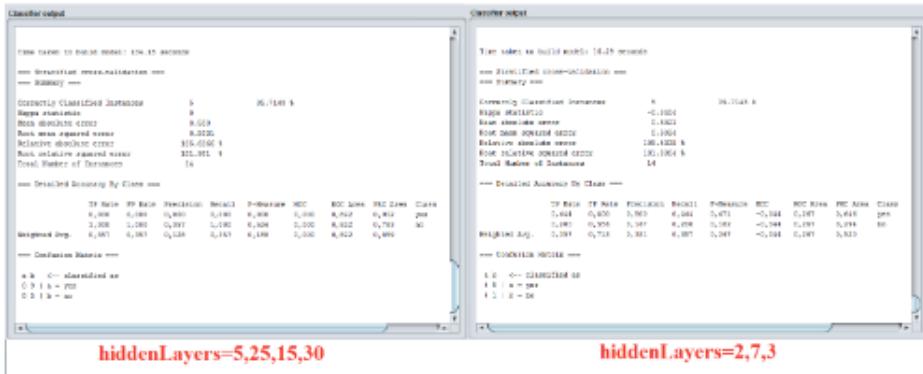
Şekil 5.20. Yapay Sinir Ağlarının İç Katman Sayılarının Girilmesi

Ağ yapısının iç katman sayıları bu şekilde verildikten sonra ağıın yapısı görsel olarak gösterilmek, incelenmek veya üzerinde ekleme/çıkarma yapmak istenirse GUI parametresinin de aktif edilmesi gereklidir. Burada weather.nominal.arff dosyası için oluşturulan ağ yapısı aşağıda gösterildiği gibi olacaktır.



Şekil 5.21. İç Katman Sayıları Girilmiş Ağ Yapısı

Ağın iç katman sayılarının değişmesi ağı tamamen değiştireceğinden dolayı ağı tümüyle etkilemektedir. Dolayısıyla iç katman sayısı ve her katmandaki sinir sayıları ağıın hem başarısını hem de öğrenme süresini doğrudan etkileyecektir. Fakat iç katman sayılarının değişiminin ağıın başarı oranını nasıl etkileyeceği tam olarak bilinemez. Ayrıca hangi veriler için hangi iç katman yapılarının kullanılacağı için genel bir öngörü yoktur. Ancak kullanıcının deneyimi bunu belirleyebilir. Dolayısıyla farklı iç katman sayılarında farklı denemeler yapıp başarı oranları ile birlikte öğrenme sürelerinin karşılaştırılarak en iyi performanslı ağıın tespit edilmesi gerekebilir. Aşağıdaki şekilde weather.nominal.arff dosyası üzerinde iç katman sayıları “2,7,3” olan ve “5,25,15,30” olan ağ yapılarının analiz sonuçları gösterilmiştir.



Şekil 5.22. Farklı İç Katman Sayılarının Analiz Sonuçları

Yapay sinir ağları kullanılarak iyi bir model elde edilmek istenirse belki de defalarca bu algoritmanın (*multiLayerPerception*) çalıştırılması ve farklı iç katman sayıları ile denenmesi gerekecektir. İç katman sayılarının değiştirilmesi veri sayılarına bağlı olarak da çok uzun süre alabilir. Dolayısıyla belki de iyi bir model elde etmek haftalar ya da aylar alabilir. Yalnızca iç katman sayılarını değiştirmek değil belki de öğrenme oranı (*learningRate*) ve momentum değerlerini de değiştirmek gerekebilir.

Gizli katman sayıları kullanıcı tarafından istenilen sayıarda verilebileceği gibi Weka tarafından ön tanımlı bazı karakterlerin girilmesi ile de hesaplanarak oluşturulması sağlanabilir. Dikkat edilirse bu algoritma seçildiğinde (*multiLayerPerception*) varsayılan olarak gizli katman sayısı giriş kutusunda “a” yazmaktadır. Bu karakterlerin neler olduğu ve nasıl hesaplandığı aşağıdaki formüllerde verilmiştir.

$$a = \frac{n + c}{2}$$

$$i = n$$

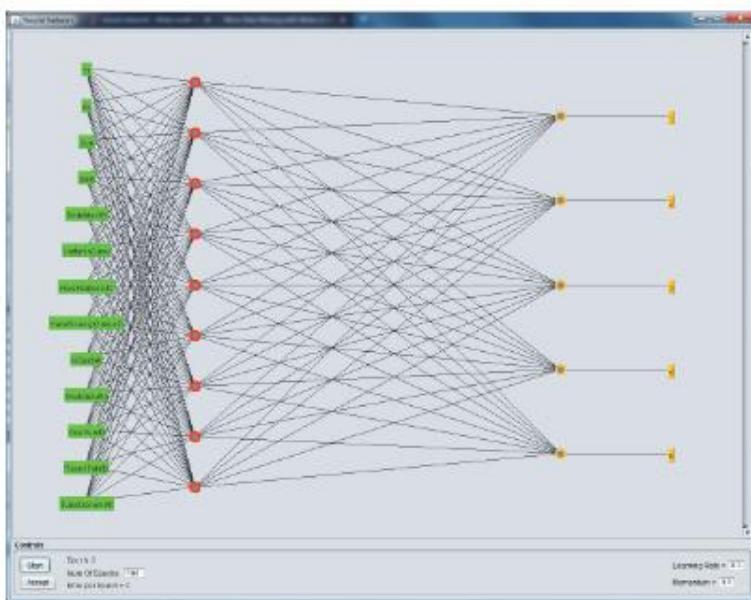
$$o = c$$

$$t = n + c$$

$$c = \text{Tahmin Edilen Sınıf}$$

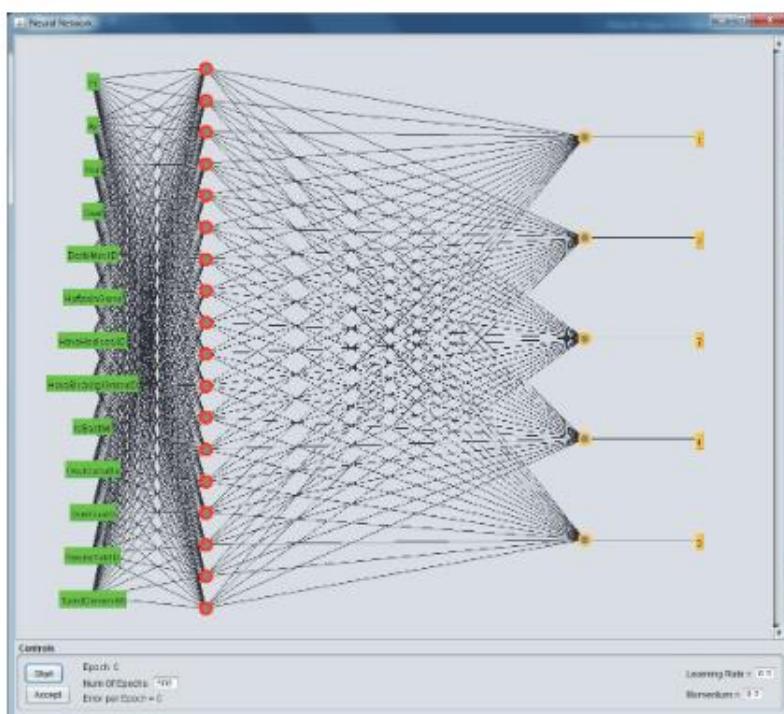
$$n = \text{niteliklerin sayısı (girdi sayısı)}$$

Aşağıdaki şekilde 13 adet girdi değeri ve çıkış sınıfında ise 5 farklı kategorik türde değer bulunmaktadır. Bu durumda iç katman sayısına (*hiddenLayers*) “a” yazılması durumunda; bu ikisinin toplamının ikiye bölümünü dokuza eşit olacağını dolayı ağı tek katmanlı dokuz sınırlı bir yapı olmuştur.



Şekil 5.23. İç Katman Sayılarının “a” Olarak Belirlenmesi

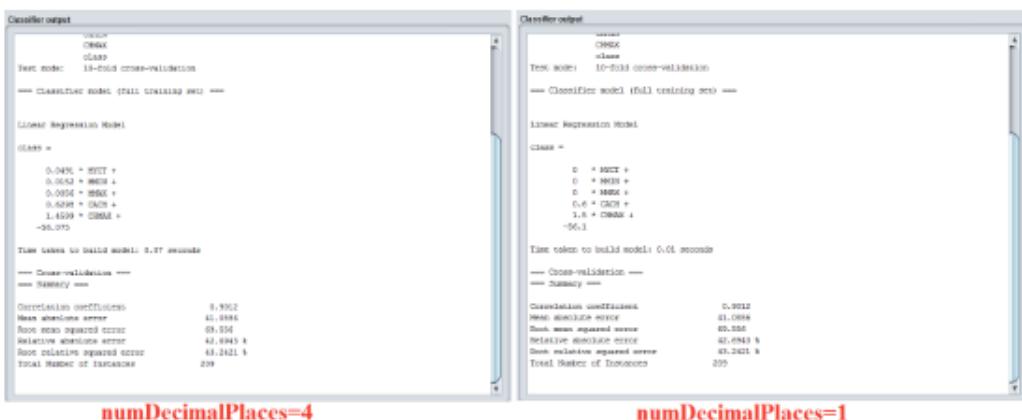
Aynı ağ yapısının iç katman sayısı (*hiddenLayers*) parametresine “t” değeri girilirse aşağıdaki şekilde ağ yapısı ortaya çıkacaktır. Bu durumda 13 adet girdi değeri ile beş adet çıktı değerinin toplamı olan 18 adet sinirden oluşan tek katmanlı bir ağ yapısı oluşturmaktadır.



Şekil 5.24. İç Katman Sayılarının “t” Olarak Belirlenmesi

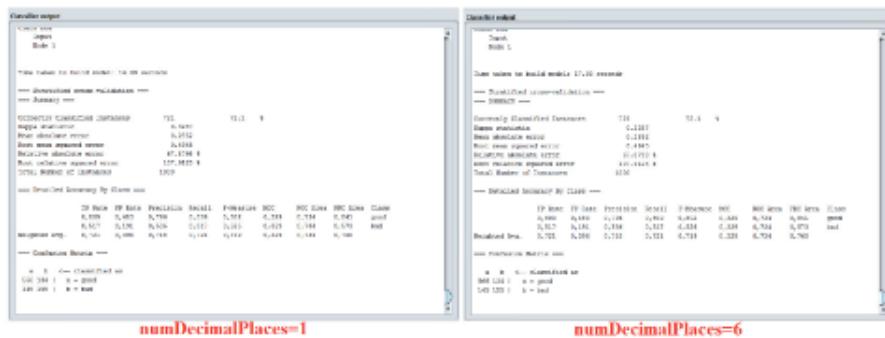
5.2.12. Ondalıklı Sayı Sınırını Belirleme (*numDecimalPlaces*)

Ondalıklı sayılarda virgülüden sonra kaç basamak kullanılacağını belirtir. Bu değer yapay sinir ağları (*multiLayerPerceptron*) algoritmasında çıktıda herhangi bir şeyin değişmediğini gösterebilir. Ancak çıktıya yansımayan değerlere etki ettiği bilinmelidir. Aşağıdaki şekilde doğrusal regresyon (*LinearRegresyon*) algoritmasında credit-g.arff dosyasına ondalıklı sayı sınırı (*numDecimalPlaces*) olarak farklı değerler verilmiş ve analiz sonuçları gösterilmiştir. Dikkat edilirse tahmin değeri hesaplanırken ki katsayıların ondalık sayı sınırları farklı olduğu gösterilmiştir. Ayrıca ondalıklı sayı sınırının düşük tutulmasının ağıın öğrenme süresine de etki ettiği işlem süresinin azalmasından da görülmektedir.



Şekil 5.25. Doğrusal Regresyon Algoritmasında Virgülden Sonra Farklı Sayıdaki Kısımları Alma

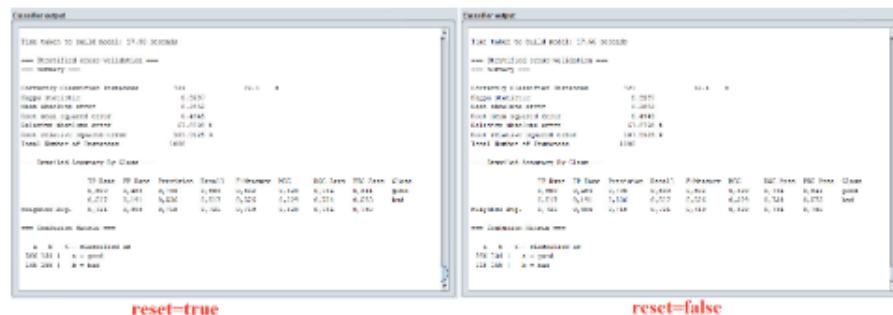
Ondalıklı sayı sınırının (*numDecimalPlaces*) değişmesi yalnızca doğrusal regresyon (*LinearRegresyon*) algoritmasında çıktı ekranında farklılığa sebep olmaktadır. Fakat yapay sinir ağları ve diğer algoritmalar da kullanıldığından çıktıda herhangi bir değişikliğe sebep olmasa da arka tarafta yapılan işlemlerde bir kısım değişikliklere sebep olduğu görülebilmektedir. Aşağıdaki şekilde yapay sinir ağları (*multiLayerPerceptron*) algoritmasında credit-g.arff dosyasına ondalıklı sayı sınırı (*numDecimalPlaces*) olarak farklı değerler verilmiş ve analiz sonuçları gösterilmiştir. Dikkat edilirse tahmin değeri hesaplanırken ki ağıın başarısı ve çıktıtı değişmezken ağıın öğrenme süresi kısmen etkilenmiştir.



Şekil 5.26. YSA Algoritmasında Virgülünden Sonra Farklı Sayıdaki Kısımları Alma

5.2.13. Baştan Başlatma (reset)

Varsayılan olarak aktif seçili olan bu parametre ağın daha düşük bir öğrenme oranına dönerken sıfırlanmasını sağlar. Eğer ağ verilen öğrenme oranından saparsa, otomatik olarak ağ sıfırlanarak daha düşük bir öğrenme oranı ile tekrar öğrenme işlemine başlar. Bu seçenek GUI parametresinin aktif edilmediği durumda kullanılabilir. Ağ öğrenme oranından saparsa ağın daha düşük bir öğrenme oranı ile tekrar öğrenmesine izin verilmez ve hata mesajını ekranda gösterir. Aşağıdaki şekilde credit-g.arff dosyasına baştan başlatma özelliğinin aktif ve pasif edilmesi durumunda ortaya çıkan analiz sonuçları gösterilmiştir. Öğrenme süresini etkilerken öğrenme başarısında herhangi bir farklılığa neden olmadığı görülmüştür.

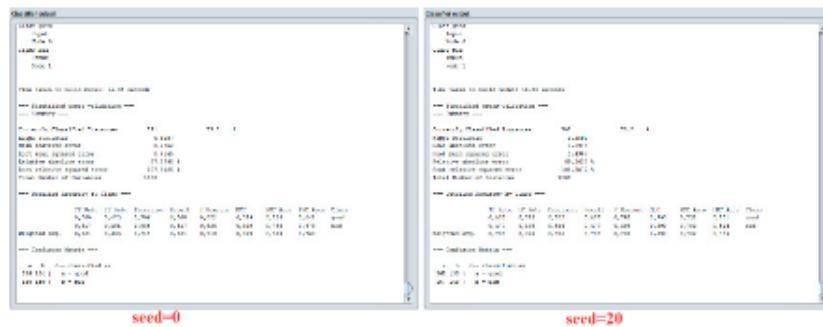


Şekil 5.27. Baştan Başlatma Parametresinin Aktif veya Pasif Olma Durumuna Göre Analiz Sonuçları

5.2.14. Başlangıç (seed)

Rastgele sayı üreticisinin başlangıç değerini belirler. Rastgele sayılar sınırlar arasındaki bağlantıların ağırlıklarını belirlemeye kullanılır. Rastgele sayılar ayrıca eğitim verilerini karıştırmak için de kullanılır. Aşağıdaki şekilde credit-g.arff dosyasına farklı başlangıç değerleri (seed) verilerek yapı-

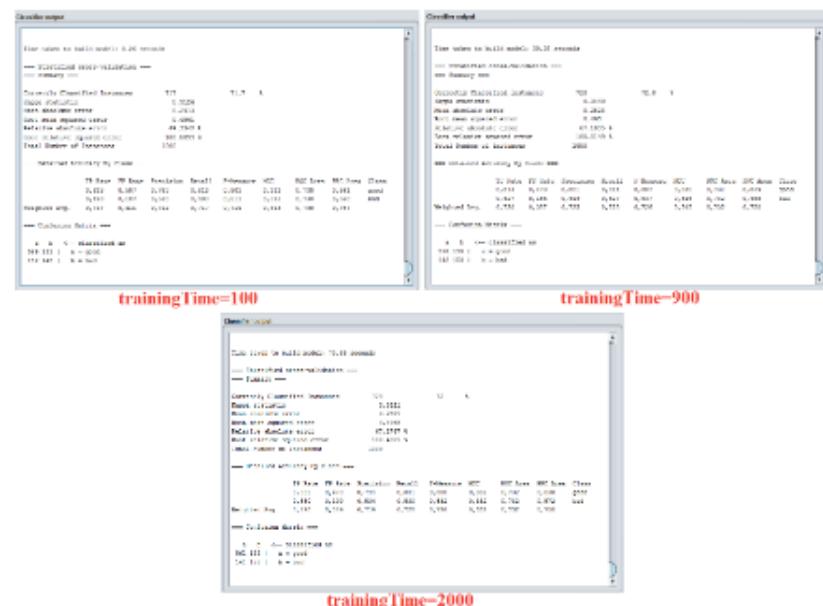
lan analiz sonuçları gösterilmiştir. Dikkat edilirse bu değerin değişimi hem ağın başarı oranını hem de ağın öğrenme süresini doğrudan etkilemektedir.



Şekil 5.28. Farklı Başlangıç Değerleri ile Yapılan Analiz Sonuçları

5.2.15. Devir Sayısı (TrainingTime)

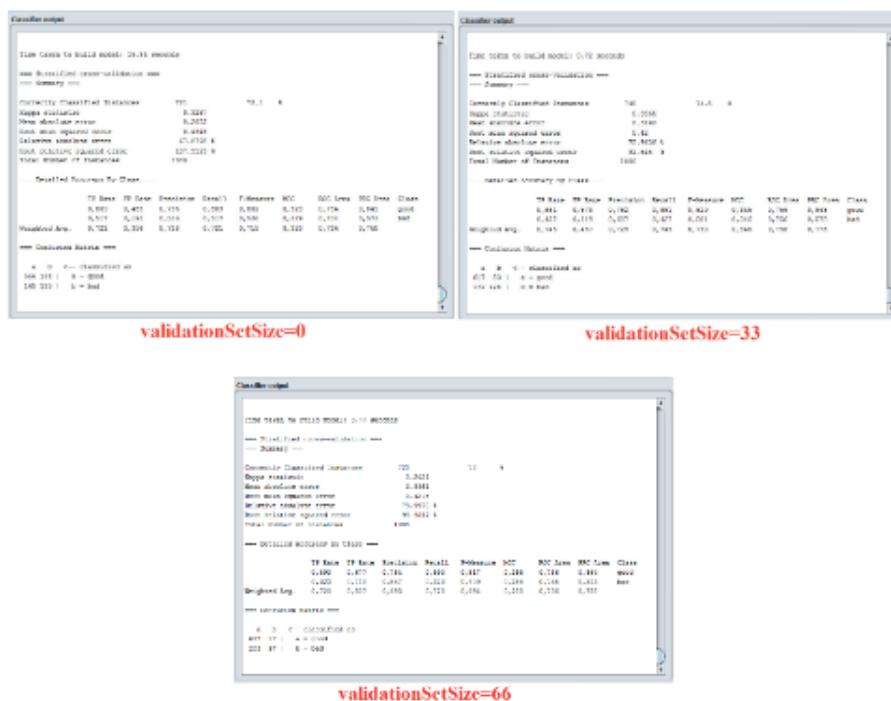
Verilerin kaç defa işleneceği anlamına gelen devir sayısının girildiği değerdir. Diğer bir adıyla geri yayılmış yapay sinir ağlarının performansını belirleyen iterasyon sayısı değeridir. GUI ekranı içerisinde de aynı değerin girişi yapılır. Orada iterasyonların sayısı (*Num of epochs*) olarak girilir. Oradaki konu başlığında da bilgi verilmiştir. Aşağıda üç farklı devir sayısı ile credit-g.arff dosyasına uygulanmış analiz sonuçları gösterilmiştir. Dikkat edilirse devir sayısının artması öğrenme süresini doğru orantılı olarak etkilerken öğrenmenin başarısını ise nasıl etkileyeceği kestirilememektedir.



Şekil 5.29. Farklı Devir Sayıları ile Yapılan Analiz Sonuçları

5.2.16. Doğrulama Kümesi Oranı (*validationSetSize*)

Doğrulama kümesinin yüzde boyutunu belirler. Eğitim, doğrulama setindeki hatanın sürekli olarak kötüleştiği veya eğitim süresine ulaşıldığı gözlemlenene kadar devam edecektir. Eğer bu değer sıfıra ayarlanırsa, hiçbir doğrulama seti kullanılmayacaktır ve bunun yerine ağ belirtilen sayıda devir sayısı için eğitilecektir. Bu parametre iki basamaklı sayılardan daha büyük bir sayı girişi yapıılırsa Tamam (*Ok*) tuşuna basıldığında ilk iki basamağını alıp kalan basamları otomatik olarak siler. Doğrulama seti yüzdesel olarak verilmesinden dolayı bu durum oluşur. Doğrulama setinin verilmesi öğrenmenin erken sonlanması neden olur. Doğrulama seti kullanılması hem öğrenme süresini hem de ağın başarısında farklılıklara sebep olacaktır. Aşağıda üç farklı doğrulama kümesi oranı ile credit-g.arff dosyasına uygulanmış analiz sonuçları gösterilmiştir.



Şekil 5.30. Farklı Doğrulama Kümesi Oranları ile Yapılan Analiz Sonuçları

5.2.17. Doğrulama Eşiği (*validationThreshold*)

Doğrulama testini sonlandırmak için kullanılır. Buradaki değer, eğitim sonlandırılmışdan önce, bir satırda kaç kez doğrulama kümesi hatasının daha kötü hale geleceğini belirler.