PERAKENDE SEKTÖRÜNDE ANORMAL FİRE TESPİTİ İÇİN BİR KARAR DESTEK SİSTEMİ

Hilal ÇELİKMAKAS

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

PERAKENDE SEKTÖRÜNDE ANORMAL FİRE TESPİTİ İÇİN BİR KARAR DESTEK SİSTEMİ

Hilal ÇELİKMAKAS
000-0001-5736-7565

Prof. Dr. Hüseyin Cenk ÖZMUTLU
000-0003-2540-9657
(Danışman)

YÜKSEK LİSANS
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2020

TEZ ONAYI

Hilal ÇELİKMAKAS tarafından hazırlanan "PERAKENDE SEKTÖRÜNDE ANORMAL FİRE TEPİTİ İÇİN BİR KARAR DESTEK SİSTEMİ" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Prof. Dr. Hüseyin Cenk ÖZMUTLU

Başkan : Prof. Dr. Hüseyin Cenk ÖZMUTLU
000-003-2540-9657
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi,
Endüstri Mühendisliği Anabilim Dalı
İmza

Üye : Dr. Öğr. Üyesi Cengiz TOĞAY
000-001-5739-1784
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi,
Bilgisayar Mühendisliği Anabilim Dalı
İmza

Üye : Doç. Dr. Turgay Tugay BİLGİN
000-002-9245-5728
Bursa Teknik Üniversitesi,
Mühendislik ve Doğa Bilimleri Fakültesi,
Bilgisayar Mühendisliği Anabilim Dalı
İmza

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
…./…./….

Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

06/02/2020

Hilal ÇELİKMAKAS

ÖZET

Yüksek Lisans Tezi

PERAKENDE SEKTÖRÜNDE ANORMAL FİRE TESPİTİ İÇİN BİR KARAR DESTEK SİSTEMİ

Hilal ÇELİKMAKAS

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Hüseyin Cenk ÖZMUTLU

Karar destek sistemleri, büyük verilerin daha iyi tanınması sağlayarak daha doğru ve etkili karar verilmesine destek olan sistemlerdir. Günümüz teknoloji dünyasında veri çok hızlı bir şekilde toplanmaktadır. Elde edilen veriler artık insan tecrübesi ile anlaşılmayacak boyutlara ulaşmaktadır. Verilerin anlamlı bir hale getirilerek yorumlanabilmesi için veri madenciliği teknikleri kullanılmaktadır.

Bu çalışmada, Bursa, Eskişehir, Kütahya ve Bilecik'te 35 şubesi bulunan bölgesel bir alışveriş merkezinin fire verileri üzerinde çalışma yapılmıştır. Fireye sebep olan birçok faktör şirket kayıtlarında tutulmaktadır. Fire maliyetlerini azaltmak için, bu verileri işleyerek kullanıcıya karar verme sürecinde destek olacak bir uygulama geliştirilmiştir. Geliştirilen uygulama veri madenciliği algoritmalarını kullanarak, fireye sebep olabilecek ya da fire oluşumuna hiç etkisi olmayan durumları kullanıcıya raporlamaktadır. Uygulamanın ürettiği sonuçlar yorumlanarak fire oluşumuna etkileri tartışılmıştır.

Anahtar Kelimeler: Karar destek sistemleri, yazılım mimarisi, veri madenciliği, perakende, fire

2020, vii + 100 sayfa.

ABSTRACT

MSc Thesis

A DECISION SUPPORT SYSTEM FOR ABNORMAL WASTE IN RETAIL INDUSTRY

Hilal ÇELİKMAKAS

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Hüseyin Cenk ÖZMUTLU

Decision support systems are systems that help more accurate and effective decision making by providing a much better recognition of big data. In today's technology world, data is collected very quickly. The data obtained can not be perceived by human experience alone. Now reaches levels that cannot be understood by human experience. Data mining techniques are used to make the data meaningful and hence interpretable.

In this study, an analysis was conducted on the waste data of a regional shopping center with 35 shops in Bursa, Eskişehir, Kütahya and Bilecik. Many factors that cause wastage are kept in company records. To reduce wastage costs, an application has been developed to support the user in the decision-making process by processing this data. The developed application creates reports that include potential wastage scenarios and circumstances irrelevant to wastage by data mining algorithms. The results produced by the application are interpreted and its effects on the formation of waste are discussed.

Key words: Decision support system, software architecturedata mining, reatil, waste

2020, vii + 100 pages.

TEŞEKKÜR

Yüksek lisans tez çalışmamın gerçekleşmesinde değerli bilgi ve tecrübelerini benimle paylaşan saygıdeğer danışman hocam; Prof. Dr. Hüseyin Cenk ÖZMUTLU'ya teşekkürlerimi sunarım.

Hayatımın her evresinde bana destek olup yanımda olan çok kıymetli aileme teşekkürlerimi sunarım.

Tez çalışmam süresince yardımlarını esirgemeyen Ayşe BAŞTUĞ KOÇ'a, Zülal ÇELİKMAKAS'a, Enes ÇELİKMAKAS'a, Y.Yiğit GÜNAYDIN'a ve Serhat TUĞAN'a teşekkürlerimi sunarım.

Hilal ÇELİKMAKAS
06/02/2020

İÇİNDEKİLER

Sayfa
ÖZET....i
ABSTRACT....ii
TEŞEKKÜR....iii
SİMGELER ve KISALTMALAR DİZİNİ....v
ŞEKİLLER DİZİNİ....vi
ÇİZELGELER DİZİNİ....vii
1. GİRİŞ....1
2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI....3
2.1. Karar Destek Sistemleri....3
2.1.1. Karar Destek Sisteminin Özellikleri....4
2.1.2. Karar Destek Sistemi Bileşenleri....4
2.2. Veri Madenciliği....6
2.2.1. Verilerin Hazırlanması....7
2.2.2. Veri Madenciliği Modelleri....8
2.3. Yazılım Geliştirme....13
2.3.1. Yazılım Geliştirme Süreçleri....13
2.3.2. Yazılım Yaşam Döngüsü Temel Süreçleri....15
3. MATERYAL ve YÖNTEM....17
3.1. Karar Destek Sistemi Veri Madenciliği....18
3.1.1. Temel Bileşenler Analizi....19
3.1.2. Ki Kare Analizi....21
3.1.3. Lineer Regresyon....22
3.1.4. Karar Ağaçları....23
3.1.5. K-Means Kümeleme Algoritması....25
3.1.6. Apriori Algoritması....27
3.2 Karar Destek Yazılımı Geliştirme Süreçleri....28
4. BULGULAR....40
5. TARTIŞMA ve SONUÇ....55
KAYNAKLAR....57
EKLER....59
EK1 Karar Destek Sistemi Uygulama Kodları....60
ÖZGEÇMİŞ....100

SİMGELER ve KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
VM	Veri Madenciliği
KDS	Karar Destek Sistemleri
YSA	Yapar Sinir Ağları
UYOL	Uygun Olmayan Ürün
SQL	Structured Query Language
Z-skor	Standart skor
CHAID	Chi-Squared Automatic Interaction Detector
C&RT	Classification and Regression Trees
ID3	Iterative Dichotomiser 3
C4.5	Extension of ID3 Algorithm
SDR	Standart Deviation Reduction
CSV	Comma Seperated Values
ERP	Enterprise Resource Planning
PCA	Temel Bileşenler Analizi
JSON	JavaScript Object Notation
Weka	Waikato Environment for Knowledge Analysis

ŞEKİLLER DİZİNİ

Sayfa

Şekil 2.1. KDS'nin temel öğeleri 5
Şekil 2.2. Üç katmanlı yapay sinir ağı modeli 11
Şekil 3.1. K-means kümeleme adımları 26
Şekil 3.2. Karar destek sistemi yazılım mimarisi 29
Şekil 3.3. Karar destek sistemi uygulaması sınıf diyagramı 30
Şekil 3.4. Karar destek sistemi kullanıcı arayüzü 31
Şekil 3.5. Öznitelik tür kontrolü 32
Şekil 3.6. Veri seti algoritma uyum kontrolü 32
Şekil 3.7. PCA test veri seti 33
Şekil 3.8. PCA test uygulama sonucu 33
Şekil 3.9. Regresyon test veri seti 34
Şekil 3.10. Regresyon test uygulama sonucu 34
Şekil 3.11. K-means test veri seti 35
Şekil 3.12. K-means test uygulama sonucu 35
Şekil 3.13. Apriori test veri seti 36
Şekil 3.14. Apriori test uygulama sonucu 36
Şekil 3.15. Ki kare test veri seti 37
Şekil 3.16. Ki kare test uygulama sonucu 37
Şekil 3.17. KDS seçenekler 38
Şekil 3.18. Veri seti türleri 38
Şekil 4.1. SQL View görünümü 41
Şekil 4.2. PCA veri seti ve türleri 43
Şekil 4.3. PCA KDS uygulaması sonuçları 45
Şekil 4.4. Lineer regresyon veri seti ve türleri 47
Şekil 4.5. Lineer regresyon KDS uygulaması sonuçları 47
Şekil 4.6. K-means veri seti ve türleri 48
Şekil 4.7. K-means KDS uygulaması sonuçları 48
Şekil 4.8. Ki kare veri seti ve türleri 49
Şekil 4.9. Ki kare KDS uygulaması sonuçları 49
Şekil 4.10. Apriori veri seti ve türleri 50
Şekil 4.11. Apriori KDS uygulaması sonuçları 50
Şekil 4.12. Otomatik KDS nümerik veri seti 52
Şekil 4.13. Otomatik KDS nümerik sonuçlar 53
Şekil 4.14. Otomatik KDS nominal veri seti 53
Şekil 4.15. Otomatik KDS nominal sonuçlar 54

ÇİZELGELER DİZİNİ

Sayfa

Çizelge 2.1. Çağlayan modelli yazılım metodolojisi 16
Çizelge 4.1. Öznitelikler ve türleri 40
Çizelge 4.2. Sayısal tür – kategorik tür dönüşümü 42
Çizelge 4.3. PCA korelasyon matrisi 44
Çizelge 4.4. PCA sonucu ilişkili öznitelikler 45
Çizelge 4.5. PCA sonucu en az etkili öznitelikler 46
Çizelge 4.6. Apriori algoritması sonucu oluşan kurallar 51

1. GİRİŞ

Şirketler karar verme süreçlerinde, kaliteli kararlar verebilmek için kendi tecrübelerine güvenebilir ya da yönetim bilişim sistemlerinden elde edilen bilgiden başka bir bilgiye bakmaya gerek duymayabilir. Yüksek başarı hedefi olan, taktik ve stratejik karar vericiler ise karmaşık faktörlerin insan algısının ötesinde olan daha farklı zor durumlarla karşılaşırlar. Bu durumlarda yönetim bilişim sistemlerinden daha esnek ve farklı durumlar için karar vericiye destek sunan Karar Destek Sistemlerinin (KDS) uygulanması uygundur (Çetinyokuş ve ark. 2002, Gökçen 2007).

Karar destek sistemleri, toplanmış bilgilerden yararlanarak, karar vermeyi kolaylaştıran bilgisayar tabanlı sistemlerdir. Karar vericinin daha rahat ve doğru bir şekilde kararını verebilmesi için, ham veriyi kaynaktan alıp, işleyip, karar için düzenli hale getirir. Sonrasında bu işlenmiş veri üzerinde veri işleme metotları, Veri Madenciliği (VM) uygulamaları gibi yöntemler kullanılarak veri üzerinde insan tecrübesinin fark edemeyeceği gizli örüntüleri, farklı durumları olası senaryoları karar vericiye sunabilir (Şeker 2014).

Karar verme süreçlerinin önemli olduğu perakende sektörü, tedarikçilerden aldığı ürünleri müşterilere ulaştıran, düşük kâr marjı ile çalışan sektör grubudur. Maliyetlerini ürünler üzerindeki çok küçük karlılıklar ile karşılamaya çalışan bu sektörde, mağazalarda ve depolarda olan kayıp kaçaklar ve fireler büyük emekler ile toplanan cirolar kadar önemlidir. Ne yazık ki uygulamada öncelik her zaman satış cirolarına verilir ve o hedefe koşulur. Tam ölçülemeyen onlarca kayıp türünden önemli bölümünün engellenmesi durumunda, net karı artırmak mümkündür.

Sektörde kayıplar müşteri hırsızlığı, çalışan hırsızlığı, tedarikçi hırsızlığı ve kurum içi hatalar olarak ayrılır. Ülke ve perakendecilik türü ayrımı yapılmaksızın kayıpların satış rakamlarına oranı %1,5-2 civarındadır. Kayıplar için de genelde müşteri hırsızlığı ilk sırayı almasına rağmen perakende sektöründe bu durumun değiştiği tespit edilmiştir. Kayıpların onlarca çeşidinden sadece bir tanesi müşteri hırsızlığıdır ve hırsızlığın yarıya indirilmesi toplam kaybı sadece %10 iyileştirir (Tunçalp 2012).

Hırsızlığın dışında kurum içi hatalar vardır ki kayıp hanesine ilave edilir ve önlemini almak gerekir. Bunlar, sayım hataları, kodlama hataları, zimmet hataları, yanlış yüklemeden kaynaklanan hatalar olarak sınıflandırılabilir. Deprem, yangın, su baskını, toplumsal hareketler gibi olaylardan da kaynaklanabilecek zararların dikkate alınması ve kayıpların telafisi için önlem alınması gerekmektedir (Tunçalp 2012).

Perakende sektöründe marketlerde ürünler raflara konduktan sonra ya da daha konmadan, çeşitli sebeplerle değer kaybına uğrayabilir. Buna müşteri, personel, tedarikçi ya da yönetim sebep olabilir. Değer kaybına uğrayarak satılamayacak duruma gelen ya da fiyatı düşürülerek satışa sunulan bu ürünler maliyet açısından şirket için ciddi zararlar oluşturmaktadır. Tez çalışması kapsamında fireyi azaltıcı önlem kararları alınırken sadece insan tecrübesinden değil kaydı tutulan fire kayıtlarının tasarlanan bir karar destek sistemi ile desteklenmesi amaçlanmıştır. Bu sayede fireye sebep olabilecek birçok etmen göz ardı edilmeksizin, geliştirilebilir mimariye sahip tasarlanan bir program ile VM modelleri ile işlenecek ve kullanıcıya anlaşılabilir bir arayüz ile sunularak en doğru kararı vermesi sağlanacaktır.

Literatürde, esnek bir yazılım mimarisine sahip, KDS olarak VM yöntemlerini kullanarak fireye sebep olan etkileri inceleyerek tavsiyelerde bulunan bir çalışma mevcut değildir. Fireyi etkileme ihtimali olan veriler kayıt altına alınarak ve VM yöntemleri ile işlenmesi sonucu anlamlı bir bilgi çıkması beklenmektedir. İşlenebilecek gerçek, ham verilerin olması VM çalışmaları için uygun olup fire oranlarının tahmini ve önlem alınması konusunda KDS uygulanması oluşturacaktır. Bu sayede fire azaltıcı önlemler daha bilimsel bir yaklaşımla ortaya çıkan sonuçlar neticesinde daha isabetli ve daha hızlı karar alınması sağlanarak kısa zamanda minimum fireye indirgenecektir.

Bu tez çalışması kapsamında perakende sektöründe, farklı şehirlerde 35 şubesi bulunan bir alışveriş merkezinin geçmiş yıllara ait anormal fire kayıtları VM yöntemleri ile analiz edilerek fireye sebep olan nedenlerin etki değerleri tespit edilecektir. Elde edilen sonuçlar değerlendirilerek gelecek fire öngörüleri ve alınacak önlemler için karar alma sürecinde destek sağlayacak karar destek uygulaması geliştirilecektir.

2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

2.1. Karar Destek Sistemleri

KDS, karar verici konumundaki kullanıcıların karar vermelerine destek olan sistemlerdir. Başka bir ifadeyle, verilecek gereken kararlar ile ilgili verileri iyi tanıyarak daha etkili karar seçeneklerinin oluşturulmasını sağlayan ve kararın daha doğru verilme olasılığını artıran destek sistemlerdir (Gökçen 2007). KDS, karar aşamasında karar vericiye destek sağlayabilen, ancak yöneticinin yerini alamayan sistemler olarak tanımlanabilir (Ordukaya 2011).

KDS, karar verici rolündeki kişilere, bilgi, model, hesaplama, yazılım ve analiz gibi desteklerin sağlanmasını amaçlayan bir yönetim bilgi sistemi çeşididir (Ordukaya 2011). KDS'nin tanımları şu şekilde yapılabilir:

- Bir KDS, yarı yapısal veya yapısal olmayan karar verme durumlarında kullanıcıya destek olmak amacı ile verilere ve karar modellerine erişmeyi kolaylaştıran sistemlerdir.
- KDS, karar veren kişi yerine geçmekten ziyade alınan kararların doğruluğunu onaylayan, yapısal olmayan ve yarı yapısal problem çözümlerinde destek sağlayan etkileşimli sistemlerdir (Yıldız ve ark. 2008).

KDS'nin tanımları içinde kullanılan farklı karar durumları şu şekilde açıklanabilir:

Yapısal Karar: Konunun yapısına göre geliştirilmiş kurallar ile alınabilen kararlardır. Algoritmik kararlar olarak da adlandırılabilirler.

Yarı-Yapısal Karar: Bu karar durumda sorunun bazı aşamalarına belirli algoritmalar uygulansa da bu kadarı yeterli olmaz. Algoritma uygulanamayan kısımlar bazı varsayımlara dayandığı için yönetici kararları özel olarak değerlendirmesi ve yorumlaması gerekir.

Yapısal Olmayan Kararlar: Bir duruma özel, standart çözümü olmayan ve kişisel değerlendirme gerektiren kararlardır.

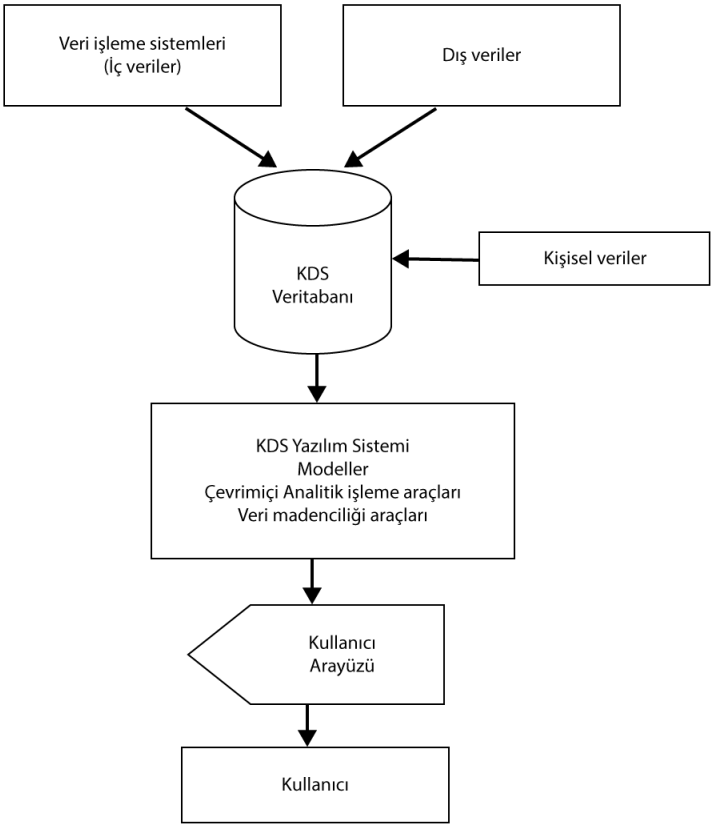
Model ve veri odaklı olmak üzere KDS'nin iki çeşidi mevcuttur. Model odaklı KDS veriler üzerinde farklı analizlerin uygulanması için özel yöntemler kullanan, bilgi sistemlerine bağlı olmayan, tek başına kullanılan sistemlerdir. Bu KDS genellikle merkezi bilgi sistemi kontrolünde olmayan gruplar tarafından geliştirilir. Bu sistemlerinin analiz verimliliği, modelin anlaşılmasını kolaylaştıracak kullanışlı ve kolay anlaşılabilir kullanıcı arayüzüyle ilişkilidir. Veri odaklı KDS, sistemlerdeki büyük verileri analiz eden sistemlerdir. Bu yapılarda büyük miktarlardaki verilerde gizli kalmış faydalı ve önemli bilgiler çıkarılarak, karar verme desteği sağlanır (Yıldız ve ark. 2008).

2.1.1. Karar Destek Sisteminin Özellikleri

KDS'nin genel özellikleri şu şekilde sıralanabilir (Çetinyokuş ve ark. 2002, Gökçen 2007); yapısal olmayan veya yarı yapısal kararlarda kullanılabilir, karar vericinin yerini alamaz, karar verme sürecine destek olur, kullanıcı ile etkileşimlidir, karar verme süreci aşamalarını destekler, model kullanırlar, kullanıcı kontrolündedir, çok sayıda bağımlı veya bağımsız karar durumları için destek sağlayabilir, farklı düzeylerdeki yöneticiler için, düzeyler arası ilişkiyi de önem vererek, karar verme yardımı sağlar, uygulamalarda kolaylık ve esneklik sağlar, bireylere veya gruplara karar verme desteği sağlar.

2.1.2. Karar Destek Sistemi Bileşenleri

KDS'nin temel öğeleri, KDS veri tabanı, KDS yazılım sistemi ve kullanıcı arayüzleridir. KDS veri tabanı, büyük bir veri deposu olabileceği gibi kişisel bilgisayara sığabilen küçük bir veri tabanıda olabilir. Bir KDS'nin temel öğeleri Şekil 2.1'de gösterilmektedir (Yıldız ve ark. 2008).



Şekil 2.1. KDS'nin temel öğeleri

KDS veri tabanı, uygulamalardan ve işlem hareketlerinden elde edilmiş verilerden oluşmaktadır. KDS yazılımı, veriyi analiz etmek için kullanılacak olan yazılım araçlarını içerir. KDS, kullanıcıların kolayca erişebileceği matematiksel, analitik modellerden veya VM modellerinden oluşmaktadır. VM yöntemleri, büyük verilerdeki gizli kalmış ilişkileri bulur, bu ilişkilerden gelecek davranış tahminleri için kurallar oluşturarak karar verme sürecinde yol göstermektedir (Yıldız ve ark. 2008).

Kullanıcı arayüzleri, karar vericilerin, KDS'ye ulaşmasını sağlar. Kullanıcılar, arayüz araçları yardımıyla KDS'yi kontrol edebilmekte ve kendi yorumları doğrultusunda probleme uygun çözümü arama pozisyonundadırlar. Mevcut problemin doğrultusunda KDS'yi kullanarak oluşturulan sonuç raporlarına göre, çözümler içerisinden en uygunu bulmaya çalışırlar (Yıldız ve ark. 2008).

2.2. Veri Madenciliği

VM son dönemlerde sıklıkla duyulan bir teknoloji olmasına rağmen 1700'lerde Bayes Teoremi ve 1800'lerde regresyon analizi gibi veriyi inceleme metotları ile başlayan, yaklaşık 300 yıllık geçmişi olan bir teknolojidir. Teknolojinin ilerlemesiyle ucuz ve güçlü bilgisayarların piyasaya çıkarak verilerin toplanmaya başlaması, insan tecrübesi ile artık bazı analizlerin yapılamayacak aşamaya gelmesi, verilerin büyümesi ve karmaşıklaşması VM ihtiyacını daha da çok hissettirmiştir (Anonim 2018a).

1990'ların başında, VM'nin bir alt adımı olarak, "Veri Tabanlarında Bilgi Keşfi" adı verilen daha kapsamlı bir süreç başlamıştır. Bu sürecin kullanılan en yaygın tanımı; "çözülmesi zor olan verilerdeki yeni, geçerli, nihai olarak anlaşılır ve potansiyel olarak yararlı olan kalıpları tanımlama süreci"dir (Fayyad 1996).

VM 1990'ların sonlarında, müşteri sadakat kartlarının piyasada yaygınlaşmasından sonra firmalar tarafından çok fazla popülerliğe ulaşan VM, günümüzde çok daha farklı alanlarda verimli bir şekilde kullanılmaktadır (Anonim 2018a).

Veri kayıtlarının düzenli olarak tutulduğu, büyük veri kaynaklarının bulunduğu her alanda, karar verme sürecine destek olunması istenen birçok sektörde VM kullanılmaktadır. Perakende ve pazarlama sektöründe; müşteri gruplandırma, satın alma davranışlarının belirlenmesi, raf düzenleme, indirim yapılacak ürün/kampanya seçimi, müşteri sadakat uygulamalarında. Tıpta; gen haritası analizlerinde, hastalık tespitlerinde. Uzay ve uçak bilimlerinde; galaksilerin sınıflandırılması, uydu ve uzay istasyonlarında hata tespitlerinde. Diğer uygulamalarda; belgeler arası benzerlik, sahtekârlık tespiti, risk analizleri, öğrenci başarı kriterleri gibi birçok alanda kullanımı mümkünüdür.

VM'nin uygulanması planlanan durum ile alakalı ön bilgi ve amaçlar açık bir şekilde ortaya konmalıdır. Bu kararlara göre VM adımları uygulanarak işlemin performansı artırılmalıdır. İşlemin sonucu ne kadar başarılı olursa, karar alma sürecine de katkısı aynı seviyede olacaktır.

2.2.1. Verilerin Hazırlanması

Veri Toplama: Geliştirilecek olan yöntemlerde kullanılacak olan veriler tek bir kaynakta ya da farklı kaynaklarda olabilir. Örneğin bir şirketin işlem verileri birden veri fazla veri tabanında tutulabilir. Bu aşamada hangi kaynakların kullanılacağına karar verilir.

Veri Birleştirme: Kullanılmasına karar verilen kaynakların amaca uygun bir şekilde oluşturulması ya da mevcutta var olan verilerin düzenli bir şekilde bir araya getirilerek veri kümesi oluşturulmasıdır.

Veri Temizleme: Birleştirilen veriler arasında analizi bozabilecek, eksik, tekrarlı, aykırı ya da aykırı değer bulunduran kayıtların düzenlemesidir.
Eksik Veri: Eksik verileri olan kayıtları silme, eksik verileri dikkate almadan çalışma yapma, eksik veriyi tahmin etme (Sabit bir değer, öznitelik değerinin ortalaması, aynı sınıfta yer alan nesnelerin öznitelik değerlerinin ortalaması vb.)
Gürültü: Bining, regresyon, kümeleme, filtreleme, bilgisayar ve kullanıcı kontrolü yöntemleri ile gürültü azaltma.

Veri Önişleme: VM'nin kalitesini artırmak, madenciliği kolaylaştırmak, verimliliği artırmak için yapılması gereken işlem adımlarıdır.
Veri Bütünleştirme: Birbiri ile bağımlı iki ya da daha fazla özniteliğin ya da nesnenin tek bir özellik ya da nesne olarak birleştirilmesidir. Sayısal veriler için "Korelasyon", Kategorik veriler için "Ki kare testi".
Veri Dönüştürme: Kullanılacak model ve algoritmaya göre verilerin belirli bir aralıkta ölçeklendirilmesi, tanımlama veya türünün değiştirilmesi gerekebilir. Veri dönüştürme işleminde veri, madencilikte uygun olan türlere dönüştürülür. Normalizasyon; veri bütünlüğünü bozacak şekilde verilerin, farklı ölçeklerle kaydedildiği durumlarda kullanılan bir yöntemdir. Verileri belirli bir aralıkta ölçeklendirmeyi sağlar. Min-max normalizasyon, Z-skor normalizasyon, Ondalıklı ölçek.

Veri Azaltma:

Veri Kübü Oluşturma: Verilerin her bir kaydını tek tek yapmak yerine planlanmış çok boyutlu modelleme ile verinin bağlantıları daha hızlı tespit edilirken, verinin çok boyutta yorumlanması ve incelenmesini sağlar.

Veriyi Sıkıştırma: Temel bileşenler analizi, resresyon ve log-doğrusal modeller, histogram, kümeleme, örnekleme

Veriyi Kesikli Hale Getirme: Bining, histogram analizi, kümeleme, ki kare analizi ve aralık birleştirme, bölümlendirme

Öznitelik Seçimi: İşlenmemiş ham veri setini temsil edecek en anlamlı altkümenin oluşturulması olarak açıklanabilir. Öznitelik seçimi için uygulanan algoritmalardın sonucuna göre en iyi n adet özniteliğin seçilmesi işlemidir (Budak 2018). Öznitelik seçimindeki amaç mevcut problemin çözümünde kullanılacak algoritmalar için performansı en iyi hale getirecek en faydalı özelliklerin seçilerek öznitelik sayısının azaltılmasıdır.

2.2.2. Veri Madenciliği Modelleri

VM modelleri, tanımlayıcı modeller ve tahmin edici modeller olarak iki grupta incelenmektedir. VM modelleri; denetimli (supervised) ve denetimsiz (unsupervised) olmak üzere iki kategoriye ayrılmaktadır (Koyuncugil ve ark. 2009). Denetimli öğrenme, sonuçları bilinen kayıtlı veriler üzerinden tahmin modeli oluşturularak, yeni kayıt verilerinin sonucunun tahmin edilmesini sağlayan modellerdir. Bu öğrenme şeklinde, önceden belirlenen kriterlere göre sınıflar ayrılır, her sınıfa ait örnekler verilir. Buradaki amaç, örnek kayıtlardan yola çıkılarak her bir sınıfın kendi içerisindeki kuralını belirlemek ve daha sonraki verilerin dahil olacağı sınıfları bu kurallar sayesinde belirlemektir. Denetimsiz öğrenmede, denetimli öğrenmeden farklı olarak sınıf etiketi bulunmayan veriler üzerinde sıfırdan kümeleme modeli üretilir. Veriler arasında gizli örüntülerin tespit edilerek karar verme sürecine dahil edilmesi sağlanır (Sağın 2018). Denetimli öğrenme modelleri; Sınıflandırma ve Regresyon Tabanlı Yöntemler, Denetimsiz öğrenme modelleri; Kümeleme, Birliktelik Analizi olarak ayrılır (Hastile ve ark. 2004).

VM modelleri, Sınıflandırma ve Regresyon Tabanlı Yöntemler, Kümeleme ve Birliktelik Analizleri olarak 3 gruba ayrılabilir.

Sınıflandırma ve Regresyon Tabanlı Yöntemler: Sınıflama ve regresyon modelleri, gelecek sınıfları tahmin etmek için, mevcut veriden önemli veri sınıflarının oluşmasını modelleyen veri analiz yöntemleridir. Oluşturulan modelin sınama kümesi ile başarımı belirlenir. Yeni veriler, oluşturulan model kullanılarak sınıflandırılır. Sınıflandırma teknikleri sahtekârlık davranışları, karakter tanıma, ses tanıma, müşteri davranışları belirleme, hastalık teşhisi gibi çok geniş ve farklı alanlarda kullanılabilmektedir.

Örnek Tabanlı Sistemler (K-Komşu Algoritması): VM yöntemleri içerisindeki en kolay sınıflama modellerinden biridir. Bu algoritmada sınıflandırma, her nesnenin diğer nesnelere olan uzaklığının hesaplanması ile gerçekleştirilir. Sınıflandırılacak nesne, özniteliklerine göre kendisine en yakın olan komşularının sınıfına yerleştirilir (Beyer ve ark. 1999). K-komşu algoritma ismindeki '*k*', sınıflandırmada kullanılacak olan komşu sayısını ifade etmektedir. Örnek olarak k=5 küme sayısı için yeni bir veri sınıflandırılmak istenir ise daha önce oluşturulmuş sınıflarda bulunan verilerin en yakın 5 tanesine bakılır. Uzaklık hesaplamalarında Öklid uzaklığı, Minkowski uzaklığı, Mahalanobis uzaklığı gibi metriklerden yararlanılabilir (Sağın 2018).

Genetik Algoritmalar: Genetik algoritmalar iyi olanın hayatta kalması mantığı üzerine oturtulmuş sınıflandırma modelleri geliştirmektedir. Bu algoritmalarda veriler gen olarak değerlendirilir, bu genler üzerinde bilgisayar ortamında çaprazlama ve mutasyon işlemlerinin gerçekleştirilmesi ve bu sırada istenilen sonuca uygun şekilde amaç fonksiyonun verilmesi ile sonuca ulaşma mantığı ile çalışmaktadırlar.

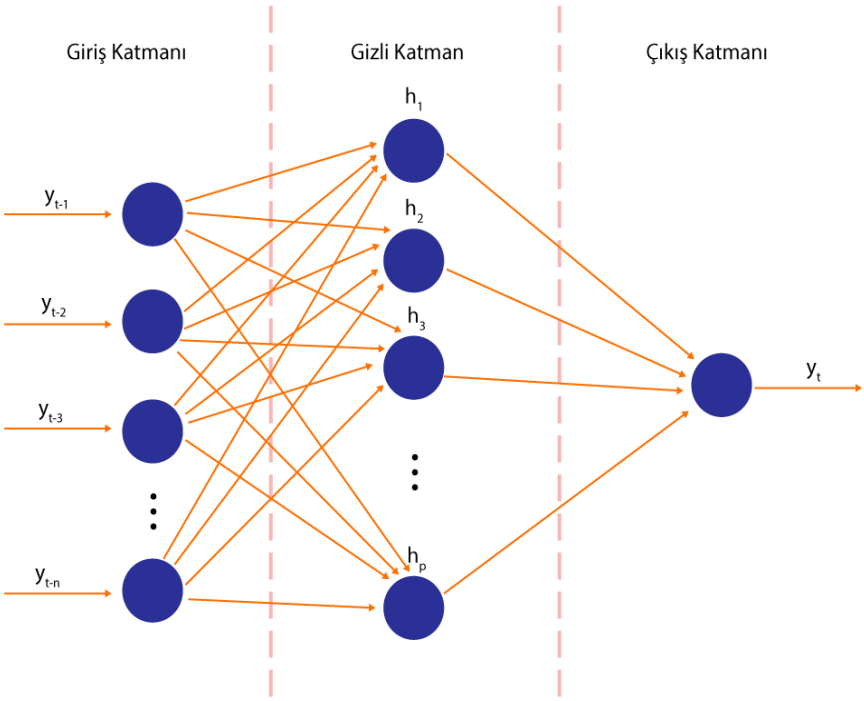
Genetik algoritmalarda ilk olarak çözümlerde, popülasyon olarak adlandırılan bir çözüm grubu oluşturulur. Çözümlerin kodları kromozom olarak isimlendirilir. Popülasyondaki birey sayısı bu adımda belirlenir. Mevcut jenerasyon kullanılarak yeni bir jenerasyon oluşturulur. Amaç, oluşturulan yeni jenerasyonun önceki nesilden daha iyi olmasıdır. Durdurma kriterine kadar yeni jenerasyonların oluşturmaya devam edilir (Silahtaroğlu 2016).

Bayesçi Sınıflandırıcılar (Naïve Bayes Sınıflandırma Algoritması): Bayes teoremine dayanan bir istatistiki sınıflandırma yöntemidir. Bu teknikte sınıf etiketi bulunan veriler kullanılarak belirli bir oranda öğrenme işlemi gerçekleşir. Yeni gelen verinin hangi sınıfa dahil olacağı, öğretilmiş veriler üzerinde yapılan olasılık işlemleri ile hesaplanır. Naive Bayes bazı kabullerde bulunulur. Bunlar; niteliklerin birbirlerinden bağımsız olması ve niteliklerin aynı önem derecesine sahip olmasıdır. Bu yöntemde eğitim verisinin az olması tahmin olasılığını düşürürken, çok olması aşırı öğrenme durumunu ortaya çıkarabilir (Han ve ark. 2006).

Karar Ağaçları Algoritması: Karar ağaçları, bir sınıflandırma yöntemidir. Düşük maliyetli olması sebebi ile en çok kullanılan sınıflandırma yöntemlerindendir. Karar ağaçları, kök, dal ve yapraklardan oluşur. Yapısı itibari ile akış şemalarına benzerler. Öznitelikler düğümler ile ifade edilir. Ağaç yapısının elemanları; dallar ve yapraklardır. Ağaç yapısı, ters çevrilmiş bir ağaca benzer. En yukarıdaki düğüm kök, en alttaki düğüm yaprak ve arada kalan düğümler ise dal olarak isimlendirilir. Kök düğümünden başlanarak her hiyerarşide veri dizisi belirli kriterlere göre bölünür (Özkan 2016). En yaygın kullanılan karar ağacı algoritmaları; CHAID, C&RT, ID3, C4.5'tir (Sağın 2018).

Yapay Sinir Ağları: İnsan beyninin soyut bir hesaplama modeli olan Yapay Sinir Ağları (YSA), gelişmiş matematiksel modellerin hesaplanmasını sağlayan, beyin işlevi görmek için tasarlanmış sistemlerdir. YSA, katmanlar şeklindedir ve ağırlıklı bağlantılar olarak adlandırılan tek yönlü bağlantılar ile birbirleri ile haberleşirler.

YSA tek katmanlı, iki katmanlı veya *n* katmanlı olabilir. Üç katmanlı YSA; girdi katmanı, gizli katman ve çıktı katmanından oluşmaktadır. İki katmanlı sinir ağında gizli katman bulunmamaktadır. Beyinde öğrenme işlemi, nöronların arasındaki sinaptik bağlantılar ile gerçekleşmektedir. YSA'da ise eğitme algoritması, girdi ve çıktı verilerini kullanarak, bağlantı ağırlıklarının bir yakınsama sağlanıncaya kadar, tekrarlı bir şekilde güncellenmesi ile gerçekleşir. Üç katmanlı bir YSA modeli Şekil 2.2'de gösterilmiştir (Sağın 2018).



Şekil 2.2. Üç katmanlı yapay sinir ağı modeli

Regresyon Tabanlı Yöntemler:

Lojistik Regresyon: Binom dağılıma uygun bağımlı değişkenlerin bulunduğu durumlarda, bağımlı değişken üzerindeki önemli risk faktörü olan, bağımsız değişkenleri belirlemek için lojistik regresyon kullanılır. Başka bir deyişle; bağımlı değişken ikili veya sıralı olduğunda, bağımlı ve bağımsız değişken değerlerinin birbirlerine olan etkileri arasındaki neden sonuç ilişkisinin belirlenmesinde kullanılır. Lojistik regresyon, verilerin olasılık kurallarına göre sınıflandırılmasını sağlar (Kaya ve ark. 2011). Bu yöntemin amacı, en az değişken ile en iyi uyumu sağlayacak şekilde, değişkenler arasındaki ilişkiyi tanımlayan bir model oluşturmaktır (Bircan 2004). Lojistik regresyon, bağımlı değişkenin değerini tahmin etmek yerine bağımlı değişkenin belirli bir değere sahip olma olasılığını tahmin etmeye çalışır (Kantardzic 2002).

Kümeleme: Kümeleme, mevcut veri içerisinde birbirine benzeyen verileri ayırma işlemidir. Sınıflandırmadaki gibi veri grupları oluşturulur. Sınıflandırmadan farkı; kümeleme modellerinde sınıf etiketlerinin önceden belirli olmamasıdır. Kaç farklı küme oluşturulacağı, verilerin benzerliklerine göre belirlenir (Dumhan 2003). Kümelemede amaç, eğitim verilerini kullanarak, nesnelerin birbirleri ile olan benzerlik ve farklılıklarına göre kümelere ayrılmasıdır. Başlangıç aşamasında veri kayıtlarının hangi kümelerde bulunacağı veya kümelemenin hangi özniteliklere göre yapılacağı bilinmediğinden, kümeleme algoritmaları denetimsiz öğrenme modelleridir.

Kümeleme yöntemlerinde veriler arasındaki uzaklıklara göre kümelere ayırma işlemi yapılır. Genellikle Euclid, Manhattan, Mahalanobis, Minkowski veya Canberra uzaklık hesaplama yöntemleri kullanılır (Arslan 2008). Veriler arasındaki uzaklıklara göre uygun algoritmalar kullanılarak veri alt kümelere ayrılır. Aynı kümede bulunan veriler ait oldukları kümeyi diğer kümelerden ayırt eden ortak özelliklere sahiptir. Bu VM yöntemi biyoloji, tıp, pazarlama, telekomünikasyon, ekonomi gibi birçok farklı alanda kullanılmaktadır. Kullanım alanı ve amaca göre kümeleme yöntemleri; model oluşturma, doğru türlerin belirlenmesi, veri indirgeme, gruplara dayalı tahmin, veri inceleme gibi amaçlar için kullanılabilir (Çakmak ve ark 2005).

Kümeleme yöntemleri, hiyerarşik kümeleme ve hiyerarşik olmayan kümeleme yöntemleri olmak üzere ikiye bölünür. Hiyerarşik kümeleme, tüm verinin ana küme olarak değerlendirilmesi ve sonrasında aşama aşama alt kümelere ayrılması veya alt kümeler olarak alınan kümelerin birleştirilmesi olarak tanımlanır. Hiyerarşik olmayan yöntemlerde ise n adet nokta, önceden bilinen k küme sayısına bölünür. Hiyerarşik kümelemenin aksine belirlenen bazı kriterlere uygun kümeler oluşturulurken, küme sayısı önceden bilinir (Guidici 2004).

Birliktelik Analizi: Büyük veri kümelerindeki gizli kalmış örüntüleri tespit etmek için kullanılan birliktelik analizleri; kümeleme modelleri gibi denetimsiz VM yöntemlerindendir. Bu yöntem ile büyük veriler üzerine birliktelik analizi algoritmaları uygulanarak, veriler arasındaki ilişkileri tespit eden kurallar oluşturulur.

Birliktelik kuralları genellikle karar verme amacı ile kullanılır. Bu yöntemin kullanıldığı bazı uyulama alanları; öneri sistemleri oluşturulması, katalog tasarımı, pazar sepet analizi, kayıp analizleridir. Müşteri sepet analizinde, müşteri satın alma modellerini oluşturmak için çaba gösterilir. Örneğin bir alışveriş merkezinde yumurta ve biberin birlikte satın alınıyor olması insan tecrübesi tahmin edilebilir. Ancak birliktelik kuralları modelleri ile hangi yumurta markası ile hangi biber çeşidinin sıklıkla tercih edildiği tespit edilebilir.

2.3. Yazılım Geliştirme

2.3.1. Yazılım Geliştirme Süreçleri

Yazılımın tanımı "bir bilgisayarda donanıma hayat veren ve bilgi işlemede kullanılan programlar, yordamlar, programlama dilleri ve belgelemelerin tümü" olarak yapılabilir (Yazılım 2020). Ayrıca yazılım, mevcut bir problemin çözümü için farklı cihazların birbirleriyle iletişimini sağlayan ve bu cihazların kullanılabilirliklerini geliştirmeyi sağlayan bilgisayar dilleri kullanılarak yapılandırılmış anlamlı ifadeler olarak da açıklanabilir. Yazılımla ilgili açıklamalar kod yazma süreci ile ilgilenirken yazılım geliştirme, bu süreçten çok daha fazlasını ifade etmektedir. Sadece kullanıcı ekranı tasarımı, bu arayüzlerin veri tabanı ile ilişkilendirilmesi için yazılan kodlar yazılım geliştirme sürecini tanımlamamaktadır. Yapılan bu işlemler sürecin bir parçası olup yazılım geliştirme aşamaları kod yazmaktan daha fazla işlemi içermektedir (Kartın 2008). Yazılım geliştirme süreci, yazılıma neden ihtiyaç duyulduğu sorusundan başlayarak üretim, geliştirme ve kullanım aşamalarının tamamını olarak tanımlanabilir (Özbilgin ve ark. 2010).

İş akış şemaları gibi yazılım geliştirilmesinde kullanılan yöntemler ihtiyaçları tam olarak karşılayamadıkları için etkinliklerini kaybetmişlerdir. Yazılım sürekli değişebilir ve genişleyebilir nitelikte olabileceği için sürekli bir döngü içerisinde ele alınmalıdır. Bu şekilde döngü içerisinde geriye dönmek veya tekrar bir değişikliğe gitmek söz konusu olabilir.

Yazılım geliştirmede farklı model ve süreçlerden söz edilebilir. Yazılım mühendisliğindeki diğer modellerin temelini oluşturan ve yazılım yaşam döngüsü modeli olan çağlayan modeli; analiz, tasarım, kodlama, test ve bakım aşamalarından oluşmaktadır.

Analiz: Yazılımların neyi nasıl yapacağının belirlendiği yani problem tanımının yapıldığı analiz şamasıdır. Kodun başarı kriteri, isteneni doğru bir şekilde yerine getirmesi ile açıklanır.

Bu sebeple öncelikle yazılımın hangi sorulara cevap vereceği doğru bir biçimde tanımlanmalıdır. Analiz aşamasının adımları şu şekilde sıralanabilir; sistem gereksinimleri, donanım, personel, kullanıcı gereksinimleri, sistem kısıtlamaları ve bu bilgilerin kullanıcılar tarafından doğrulanarak proje planının oluşturulmasıdır (Özbilgin ve ark. 2010).

Tasarım: Analiz aşamasında oluşturulan gereksinimlere göre yazılım temellerinin planlandığı adımdır. Yazılım tasarımı, bir sistemi oluşturacak gereksinimleri belirlemek için kullanılacak olan teknikler, gösterimler, stratejiler ve desenleri içermektedir. Bu aşama yazılım kullanıcı arayüzlerini, yazılım mimari tasarımını, veri tasarımını ve tasarım araçlarını kapsamaktadır. Tasarım süreci hem kullanıcı arayüzlerini hem de programın temelini oluşturmaktadır. Planlanacak tasarımın, yazılımın fonksiyonel gereksinimlerini karşılaması dışında güvenlik, performans ve kaynak gibi gereksinimleri de düşünülerek gerçekleştirilmelidir (Özbilgin ve ark. 2010).

Kodlama: Kodlama adımı, tasarım aşamasında yapılan plan doğrultusunda yazılımın oluşturulması aşamasıdır. Bu adım, kodlamanın yanı sıra verinin kullanıcıya ulaştırılması ve yazlımın geliştirilmesi sürecindeki tüm çalışmaları içerir. Analiz adımında belirlenen isteklerin sunulması için arka planda gerçekleşen algoritmalar bu adımda uygulanmaktadır. Yazılım geliştirme ortamı, kodlama dili, yazılım geliştirme araçları, veri tabanı yönetim sistemi seçimleri bu adımda gerçekleştirilir (Özbilgin ve ark. 2010).

Test: Test aşaması, kodlanan yazılımı sınama ve doğrulama sürecidir. Yazılımın istenen kriterlere uygun olup olmadığı, gereksinimlere cevap verip vermediğini kontrol etmek için çeşitli test tekniklerinin kullanıldığı adımdır. Test teknikleri statik ve dinamik olarak ayrılabilir. Statik teknikler, yazılım yaşam döngüsünden çıkarılan sonuçların analiz ve kontrolünü sağlarken, dinamik teknikler yalnızca yapılmış sistemi barındırır. Yazılımın ilk testleri genellikle yazılımcı tarafından yapılır. Geri bildirim ve gerçek hata ayıklama işlemleri test birimi tarafından yapılır. Yazılım kullanıldığı süre zarfında testler ve geri bildirimler devam eder (Özbilgin ve ark. 2010).

Bakım: Yazılım son kullanıcılar tarafından kullanılmaya başladıktan sonraki süreçte hata giderme ve güncellemeler yapma adımıdır. Yazılım destek sürecini kapsar, eksiklerin giderilmesi, performans artırıcı iyileştirilmelerin yapılması gibi aşamaları içerir.

2.3.2. Yazılım Yaşam Döngüsü Temel Süreçleri

Çekirdek süreçler olarak da isimlendirilen temellerinin gerçekleştirilmesi amacı ile kullanılan yöntemlerde belirtim modelleri; çekirdek sürece ait fonksiyonları uygulamak için kullanılan yöntemler iken süreç modelleri; döngüde belirtilen süreçlerin iyileştirme ve geliştirme aşamalarının hangi düzen veya sırada yapılacağı, geliştirmelerin nasıl uygulanması gerektiğini tanımlayan modellerdir (Anonim 2018b).

Belirtim Yöntemleri: Süreçler arası ilişkileri ve iletişimi sağlayan yöntemler; veri akış diyagramları, nesne ve sınıf şemaları, yapısal şemalardır. Süreçlerin iç akışını belirtmek için kullanılan yöntemler; algoritmalar, karar ağaçları, karar tabloları ve anlatım dilidir. Süreçler arasında kullanılan veri tanımlamaları için ise nesne ilişki modelleri, veri tabanı tabloları ve veri sözcükleri kullanılır (Anonim 2018b).

Süreç Modelleri: Yazılım geliştirme işleminin genel olarak yapılma şekline ilişkin yöntemlerdir (Anonim 2018b). Başlıca modeller şunladır; gelişigüzel model, barok modeli, çağlayan modeli (Çizelge 2.1), V modeli, helezonik model, evrimsel model, artırımsal model, araştırma tabanlı model.

Çizelge 2.1. Çağlayan modelli yazılım metodolojisi (Anonim 2018b)

Aşama	Kullanılan Yöntem ve Araçlar	Amaç	Çıktı
Planlama	Veri Akış Şemaları, Süreç Belirtimleri, Görüşme, Maliyet Kestirim Yöntemi, Proje Yönetim Araçları	Süreç İnceleme, Kaynak Kestirimi, Proje Yönetimi	Proje Planı
Analiz	Veri Akış Şemaları, Süreç Belirtimleri, Görüşme, Nesne İlişki Şemaları, Veri	Süreç Analizi, Veri Analizi	Sistem Analiz Raporu
Analizden Tasarıma Geçiş	Akışa Dayalı Analiz, Süreç belirtimlerinin program tasarım diline dönüştürülmesi, Nesne ilişki şemalarının veri tablosuna dönüştürülmesi	Başlangıç Tasarımı, Ayrıntılı Tasarım, Başlangıç Veri Tasarımı	Başlangıç Tasarım Raporu
Tasarım	Yapısal Şemalar, Program Tasarım Dili, Veri Tabanı Tabloları	Genel Tasarım, Ayrıntılı Tasarım, Veri Tasarımı	Sistem Tasarım Raporu

3. MATERYAL ve YÖNTEM

Bu çalışmada bölgesel bir alışveriş merkezinin mağazalarında ve depolarında oluşan firelere sebep olan faktörlerin tespit edilmesi için bir KDS oluşturulmuştur. Şirket bünyesinde yazılan özel bir el terminali ve masaüstü programları arayüzü ile SQL veri tabanında tutulan aylık fire verileri işlenecektir.

KDS olarak düşünülecek olursa birçok çeşitli faktörden dolayı fire oluşabilir. Oluşan fireyi etkileyen faktörler doğrudan veya dolaylı olabilir. Ya da birden fazla faktörün bir araya gelmesi fireye sebep olabilir. Bunlar insan tecrübesinin dışında olan durumlardır. Bu durumların tespiti için KDS geliştirilmiştir. Geliştirilen KDS kayıtlı veride, etkili ve etkisi olmayan öznitelik seçimi, birbiri ile ilişkili olan özniteliklerin tespiti ve bu işlemlerden sonra hangi yöntemin uygulanmasının daha doğru olacağını kullanıcıya sunarak karar verme sürecinde en sağlıklı yöntemin seçilmesinde tavsiyelerde bulunacaktır.

KDS bileşenleri bölüm 2.1.2 de açıklandığı üzere ana hatları ile veri tabanı, yazılım ve kullanıcı arayüzünden oluşmaktadır. Bu çalışmada farklı mağazalardaki kullanıcılardan el terminalleri ile aylık periyotlarda gelen fire verileri SQL veri tabanında kayıt altında tutulmaktadır. Veri tabanındaki verilere SQL tablo, JSON veya Excel formatında ulaşılarak yazılım katmanında işlem yapılabilir. KDS yazılım modeli, belirlenen amaç doğrultusunda, karar sistemini oluşturmak için VM tekniklerinin kullanıcı arayüzü ile kolay bir şekilde kullanımını sağlayacaktır. Geliştirilen KDS uygulaması aynı zamanda Weka yazılımı ile doğrulanarak, öncelikle Weka'dan alınan sonuçlar sonrasında geliştirilen yazılımda kodlanarak yorumlanmıştır.

Geliştirilen uygulamada kullanılacak veri bağımsızdır, belirli bir sektöre veya alana ait olmayabilir. Gerektiğinde farklı sektör ve alanlardaki veri işlemelerinde de kullanılabilmektedir. Mevcut yazılımda perakende sektörü veri kayıtları kullanılmıştır. Bu kayıtlar aylık raporlar için sisteme girilen fire yani çöpe giden ürünler olarak isimlendirilebilecek şirket ciro kaybına sebep olan ürün kayıtlarıdır. Bu kayıtlar veri tabanına çeşitli öznitelikler ile kayıt edilir. Her bir ürün için fire kaydı haricinde farklı

raporlarda farklı bilgilerde mevcuttur. Örneğin fireye ayrılmış bir ürünün satış raporunda ay içerisinde mevcut şubede kaç adet satıldığı, ürünün hangi kategoriye ait olduğu, şubedeki sorumlu kişi gibi faktörler elde edilebilir. Farklı raporlardan edinilen sonuçlar ve fire kayıtları birleştirildiğinde oluşan kayıtlarda, VM yöntemlerinden yararlanılarak çeşitli sonuçlar ve analizler elde edilmiştir.

Mevcut uygulamada kullanılacak olan verilerin işlenip istenen sonuca ulaşabilmesi için öncelikle bölüm 2.2.1 de açıklanan verilerin hazırlanması adımı için, gerekli işlemler yapılmıştır. Hazırlanan verilere uygun VM yöntemleri uygulanarak çeşitli sonuçlar elde edilmiştir.

3.1. Karar Destek Sistemi Veri Madenciliği

Veri Toplama: KDS'de kullanılacak olan perakende verileri şirket bünyesindeki sunucularda kurulu olan SQL ve Oracle veri tabanlarında tutulmaktadır. Birçok sistemde olduğu gibi veri tabanı üzerinden verilere ulaşmak mümkün olsa da, şirkette kullanılan Erp programı ve terminal yazılımları için farklı şirketler tarafından tasarımı yapılmış veri tabanı modellerindeki kayıtların anlaşılması güçtür. Bu sebeple işlenmesi planlanan verilerin öncelikli olarak gerek ERP programından gerekse diğer şirket raporlama yazılımlarından elde edilmesi gerekmektedir. Uygulamada kullanılan veri setleri belirtildiği şekilde öncelikle dışarı anlaşılabilir bir şekilde aktarılarak sonrasında bu uygulama için tasarlanmış SQL veri tabanına istenen formatta aktarılmıştır.

Veri Birleştirme: Fire veri girişleri aylık olarak raporlanarak değerlendirilmektedir fakat mağaza personeli bu girişleri ayda bir kere yapmak yerine fırsat buldukça yapmaktadır. Bu sebeple aynı ürünün ayın farklı günlerinde aynı sebeple birden fazla defa girişi bulunmaktadır. Veri birleştirme sürecinde örnekteki durumda olan veriler her ayda bir kaydı tutulacak şekilde gerek miktarsal gerek fiyatsal toplanarak birleştirilmiştir.

Veri Temizleme: Fire veri girişi mağaza personelleri tarafından el terminalleri ile yapıldığı için veri giriş sürecinde insan kaynaklı ya da terminal kaynaklı hataların

olması muhtemeldir. Aynı firenin gün içerisinde iki defa girilmesi, terminalin yazılım hatasından kaynaklı çift kayıt, hatalı bilgi girişi gibi durumlar mevcuttur. Veri temizleme ile böyle kayıtlar tespit edilerek silinmiştir.

Veri Önişleme: Veri tabanında tablolarda tutulan veriler farklı türlerdedir. Tarih verisi, sayısal veri, kategorik veri ve metin gibi. Metin tipinde tutulan kayıtlar gerek gizlilik(mağaza müdürü adı), gerekse algoritmalarda uygulama zorluğundan dolayı kategorik veri tipine çevrilerek benzersiz kodlar ile işleme alınacaktır.

Öznitelik Seçimi: VM uygulamalarında bağımlı öznitelikler sonuca ulaşmada aynı etkiyi gösterirken karmaşıklığı artırarak işlem adımlarını zorlaştırmaktadır. Bu durumun önüne geçmek için birbiri ile ilişkili olan özelliklerin tespit edilerek veride boyut azaltmaya gidilmesi gerekmektedir. Aynı şekilde bağımlı olmasa bile sonuca etkili olmayan öznitelikler bulunabilir. Bu durumda aynı şekilde karmaşıklığı artırır. Sonuca etkisi olmayan bu faktörlerin çıkarılması gerekir. Bağımlı ve etkisiz öznitelik tespitleri için VM'de farklı teknikler mevcuttur.

3.1.1. Temel Bileşenler Analizi (Principal Component Analysis)

Temel bileşenler analizi; veri azaltmak, öznitelik seçmek için kullanılan önemli bir tekniktir. Veri setinin varyans-kovaryans yapısını, özniteliklerin doğrusal birleşimleri ile açıklayarak, boyut azaltılması ve yorumlanmasını sağlayan çok parametreli bir yöntemdir.

Temel bileşenler analizinde, bağımlılık içeren, ölçüm sayısı n olan p adet öznitelik; birbirinden bağımsız, ortogonal ve doğrusal olan özellikleri bulunan k adet yeni özelliklere dönüştürülebilir. n durum ve p özellikten oluşan veri matrisi X_{pxn} olsun. Burada X_{pxn} işlenmemiş olan veri matrisi bu şekliyle kullanılabileceği gibi Z_{pxn} şeklinde ifade edilen standartlaştırılmış öznitelikler matrisi de kullanılabilinir. Ham veri matrisinin X_{pxn} tercih edilmesi durumunda temel bileşenlerin bulunması için varyans-kovaryans matrisi, standartlaştırılmış veri matrisinin Z_{pxn} tercih edilmesi durumunda ise korelasyon matrisi uygulanmaktadır. Çok farklı sonuçlar oluşturan bu iki yöntemden

hangisinin tercih edileceği konusunda belirleyici etmen, verilerin ölçü birimleridir. Verilerin varyansları ve ölçü birimleri birbirine yakın ise kovaryans matrisi, birbirine uzak korelasyon matrisi seçilir. Gerçekte özniteliklerin ölçü birimlerinin yakınlığı genelde olası olmadığından Z_{pxn} standart matrisi kullanılır (Yaycılı 2006).

Temel bileşenler analizinin, bağımlı öznitelikleri tespit etmek ve boyut azaltmak için kullanıldığından bahsedilmiştir. Z matrisinin değişkenleri arasında tam bağımsızlı ya da çok az benzerlik olması durumunda, bağımsızlaştırma bir amaç olamayacağı gibi, boyut azaltmadan da önemli bir birleştirme sağlanamaz. Çünkü

$$\hat{R} = \frac{1}{n-1}\sum_{i=1}^{N} z_j z_j^l = ZZ^1 = I \tag{3.1}$$

özelliği nedeniyle z_j'lerin y_j'lere dönüştürülmesi ile tekrar birim ilişki matrisine ulaşılacaktır. Bu durumda tek değişkenli analizlerdeki varyans karşılığı olan genelleştirilmiş varyans ifadesi ve aşağıdaki testler (1.1, 1.2) önerilmektedir (Tatlıdil 1996).

$$X_h^2 = -\left\{(n-1) - \frac{1}{6}(2p+5)\right\} log|R| \tag{3.2}$$

veya

$$X_h^2 = -\left\{(n - \frac{1}{6}(2p+11)\right\} log\left|\hat{R}\right| \approx X^2_{\frac{1}{2}p(p-1)} \tag{3.3}$$

Küresellik olarak tanımlanan test için;

$$H_0: R = 1 \tag{3.4}$$

$$H_1: R \neq 1 \tag{3.5}$$

hipotezi kurulur. H_0 kabul edilmezse temel bileşenler analizinin kullanılması tavsiye edilir (Tatlıdil 1996).

Temel bileşenler analizinin kullanılmasında iki amaç olduğu açıklanmıştır. Bunlardan biri bağımlılıkların tespit edilerek kaldırılması ve diğeri de boyut indirgemedir. Öncelikle $[R\hat{} - lI]=0$ ifadesinden öznitelikler bulduktan sonra aralarında önemli olan m adet öznitelik sayısına karar verebilmek için birçok metot bulunmaktadır. Bu metotlardan birincisi, $R\hat{}$ matrisi kullanıldığı zaman birden (1'den) büyük özniteliklerin sayısı, *m* değerini verir. Kullanılacak eşitlik;

$$\frac{\sum_{j=1}^{m} \lambda j}{p} \geq \frac{2}{3} \tag{3.6}$$

temel bileşenlerin sayısı eşitsizliği sağlayan en küçük *m* değeri olur (Yaycılı 2006).

3.1.2. Ki Kare Analizi (Chi Square Analysis)

Ki kare analizi beş farklı alanda kullanılabilir (Anonim 2018c):

- Veri setindeki özelliklerinin birbiri ile bağımlı olup olmadığının tespiti (ki kare bağımsızlık testi);
- Veri işleme sonuçlarının açıkça teorik bir ihtimal dağılımı ile orantılı olup olmadığının belirlenmesi (ki kare uygunluk testi);
- İki ya da daha fazla özelliğin aynı noktadan gelip gelmediğinin araştırılması (bağdaşıklık testi);
- Veri seti varyanslarının tahmin ve testleri;
- İkiden fazla veri setinin oranının birbirine eşit olup olmadığının tespiti.

Ki Kare Bağımsızlık Testi: Veri setinde bulunan özelliklerin bağımlı veya ilişkili olup olmadıklarını tespit etmek için kullanılır. Testi uygulamak için öncelikle bir tablo oluşturulur ve özelliklerden biri sütuna diğeri satıra yerleştirilir. Testin aşamaları şu şekildedir (Anonim 2018c):

1. Hipotezin oluşturulması: İki değişken arasındaki bağımlılık durumu test edileceği için hipotezler:

H_0: İki değişken birbirinden bağımsızdır

H_1: İki değişken birbirine bağımlıdır.

ifadeleri ile oluşturulur.

2. α: Anlamlılık seviyesinin belirlenmesi

3. χ^2: Bağımsızlık test istatistiğinin hesaplanması

Bundan dolayı;

$$\chi^2 = \sum_{i=1}^{n} \frac{(f_i - N_{pi})^2}{N_{pi}} \tag{3.7}$$

$$= \sum (f_i - f_i')^2 / f_i' \tag{3.8}$$

formülü kullanılır. Formülde;

f_i: Gerçek frekansları;

f_i': Teorik frekansları

ifade etmektedir.

χ^2 istatistiği hesabı yapıldıktan sonra karar verme adımına geçilir. Özellik sayısı 30'dan küçükse χ^2 tablosundan, 30'dan büyükse normal dağılım (Z tablosundan) elde edilen değerler alınarak analiz yapılır ve karar verilir.

$$\chi^2 < \chi_\alpha^2 \tag{3.9}$$

ise H_0 reddedilemez (bağımsızlık vardır),

$$\chi^2 > \chi_\alpha^2 \tag{3.10}$$

ise H_0 reddedilir (bağımsızlık yoktur).

3.1.3. Lineer Regresyon

Lineer regresyon, bir veya birden fazla bağımsız öznitelik ile başka bir bağımlı öznitelik arasındaki ilişkiyi modellemek için kullanılan bir metottur. Lineer regresyonun modelinin kullanım amacı verilmiş olan x ve y değerlerini kullanarak w değerlerini

bulmaktır. *w* değeri bulununca, *y* değeri verilmeyen fakat *x* değeri belirtilmiş olan bir veri ile *y* değeri hesaplanabilmektedir. Bu sebeple lineer regresyon aşağıdaki biçimde ifade edilebilir (Peker ve ark. 2017):

Her zaman $x_0 = 1$ olmak koşuluyla;

$$f(x) \sum w_i x_i = wx \tag{3.11}$$

3.1.4. Karar Ağaçları (Non- Lineer Regresyon-M5P)

M5P metodu, karar ağaçlarından türetilen bir metottur. Birçok karar ağaçlarının oluşturularak sonuçlarının bir araya toplanması ile regresyon, sınıflandırma gibi işlemlerde kullanılır. Tahmin etme durumlarında kullanılan basit ve kullanışlı bir yöntemdir. Karar ağaçlarında kesin sonuç belirtilir ve karar ağacını oluşturan kuralların oluşturulma şekli açıklanır. Karar ağaçlarının diğer yöntemlere göre daha çok kullanılmasının sebebi daha anlaşılabilir yapıda olmasıdır (Mitchell 2009).

Karar ağaçları metodunda öncelikle karar ağacının oluşturulabilmesi için eğitim kümesi kullanılır. Ağaçların oluşturulmasında CART, C4.5, ID3, CHAID gibi metotlar kullanılabilir. Bu metotların bilgi kazanımını ölçerken farklı yöntemleri kullandıkları için birbirinden farklıdır. Karar ağaçlarının oluşturulurken uygulanan temel adımlar şu şekildedir (Garip 2017):

- *E*, eğitim kümesi olsun.
- *E* kümesinin örneklerini en iyi bölen öznitelik *k* entropiye göre belirlenir. *k* değeri ağacın yeni bir düğümünü oluşturur.
- *k*'da çocuk düğümlere ilişkin alt veri kümesinin örnekleri bulunur.
- Her bir alt veri kümesi için;
    a. Örnekleri bölecek nitelikler bitmişse
    b. Örneklerin tümü aynı sınıfta ise
    c. Geriye kalan niteliklerin değerini taşıyan başka bir örnek yoksa işlemi sonlandır.

- Diğer durumda alt veri kümesini bölmek için, ikinci adıma dönülerek entropiye göre yeni *k* düğümü bulunur.

Karar ağaçlarının oluşturulmasında öznitelik seçimi entropiye göre yapılır. Entropi, belirsizliği, rastgeleliği ve beklenmeyen durumların oluşması durumlarını ifade eder. Bilgi kazanımı en yüksek olan öznitelik seçilir.

Entropinin hesaplanması aşağıda gösterilmiştir. Örneğin *S* kümesi, *i = {1, ..., m}* olmak üzere C_i sınıfı s_i satır içersin. Buradaki bilgi kazanımı *I* hesabı şu şekildedir:

$$I(s_1, s_2, \dots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{si}{s} \tag{3.12}$$

Bir *A* değişkenin $\{a_1, a_2, \dots, a_v\}$ değerleri ile düzensizliği *E(A)* (entropi)

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \cdots + s_{mj}}{s} I(s_{1j}, \dots, s_{mj}) \tag{3.13}$$

A değişkeni kullanılarak ağacın dallanmasıyla kazanılan bilgi *Kazanç* (*A*) ise şu şekilde hesaplanır:

$$Kazanç(A) = I(s_1, s_2, \dots, s_m) - E(A) \tag{3.14}$$

Karar ağacı oluşturulduğunda kökten yaprağa doğru gidilir. Düğümler aşama aşama test edilir. Sınıfların oluşturulduğu dal uçlarına yapraklar konur. Düğümler ve ağacın köküne sorular etiketlenir. Etiketlenen sorular cevaplandıkça dal ucuna bir yaprak eklenir. Ağacın kökündeki sorun dallara bölünerek küçük parçalara ayrılmış olur. Sorunun çözümü karar ağaçları olmasa bile, karar ağaçları ile daha anlaşılabilir duruma gelmiş olur. Karar ağaçları genellikle sınıflandırma, tahmin ve kümeleme sorunlarının çözümünde tercih edilir.

Problemde öznitelik sayısı arttıkça ve problem karmaşıklaştıkça düğüm sayıları artmaktadır. Bu ağaçların anlaşılabilirliği zor olacağından ağacın budanması gerekir. Karar ağaçlarındaki yaprak sayısı kural sayısı kadardır. Karar ağaçlarının daha gelişmiş

şekli olan rastgele orman yöntemi, eğitim kümesinden birçok karar ağacı üretip, sonuçların birleştirilmesi mantığı ile çalışır.

M5P metodu, karar ağaçlarının her birinin girdi vektöründen bağımsız olarak örneklenmesiyle oluşturduğu ağaçlar topluluğudur. Bu metot, M5 metodunun geliştirilmesi ile oluşturulmuştur. Topluluk öğrenme metodu olup, sürekli değişkenleri tahmin etmek için geliştirilmiştir. M5 karar ağaçları 3 aşamada oluşturulur (Garip 2017):

- Karar ağacının geliştirilmesi
- Geliştirilen ağacın budanması
- Budanan ağacın düzleştirilmesi

M5 karar ağacı oluşturulurken amaç, SDR değerinin maksimuma getirilmesidir.

$$SDR: Sd(T) \vdash \sum_i \frac{|T_i|}{|T|} XSd(T_i) \tag{3.15}$$

T: Şartlar
T_i: *i* durumundaki şartlar
Sd (T): *T*'nin standart sapması
Sd (T_i): T_i standart hata sapması

M adet karar ağacı oluşturularak aşırı öğrenmenin önüne geçebilmek için budama işlemi gerçekleştirilir. Budama yapılırken ağacın değişkenleri birer birer kaldırılarak düğümler çözülür ve sonra ağaçta düzleştirme işlemi yapılır.

3.1.5. K-Means Kümeleme Algoritması

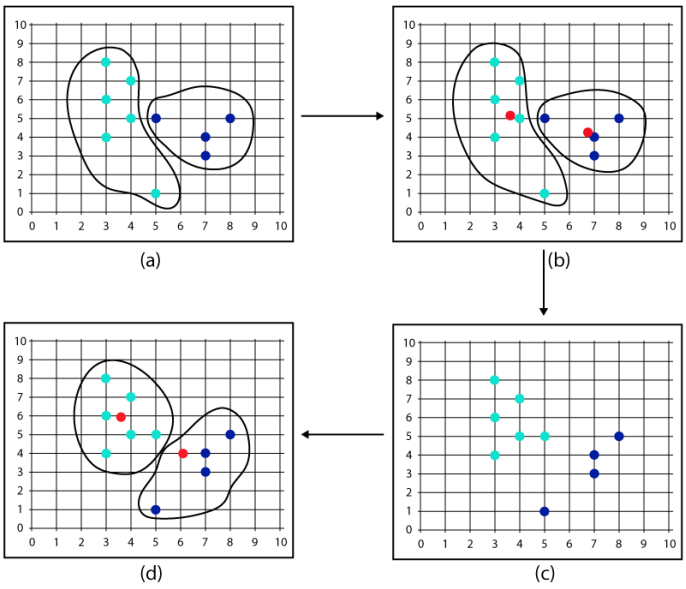
Denetimsiz öğrenme metotlarından en çok tercih edilen bir yöntemdir. K-means'in kümeleme algoritması her bir örneğin sadece bir kümede olmasına izin verir (Evans 2005). Bu sebeple, keskin bir kümeleme algoritmasıdır. K-means algoritmasının temel mantığı veri kümesinde *n* adet veri nesnesi olan bir kümenin, girişte belirtilen *k* adet kümeye bölünmesidir. Amaç, kümelere bölme işleminin sonucunda oluşturulan

kümelerin, aynı kümedeki nesnelerin benzerliğinin maksimum ve farklı kümelerdeki nesnelerin benzerliğinin minimum olmasını sağlamaktır. Mevcut çalışmada, (3.16, 3.17) numaralı Öklid uzaklığı kullanılarak kümeleme işlemi yapılmaktadır (Dinçer 2006).

$p=(p_1, p_2, \ldots, p_n)$ ve $q=(q_1, q_2, \ldots, q_n)$ (3.16)

$$\sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \quad (3.17)$$

K-means algoritması rastgele seçilen k (küme sayısı) merkez noktayla (küme) işleme başlar. Veri setindeki her bir nokta kendisine en yakın olan merkez noktanın kümesine atanır. Küme merkezinin değerleri kendisine ait tüm noktaların ortalaması alınarak güncellenir (Şekil 3.1). Bu işlem merkez noktasının değerleri değişmeyinceye kadar devam eder (Amasyalı ve ark. 2008).



Şekil 3.1. K-means kümeleme adımları

K-means algoritmasının başlıca adımları aşağıdaki gibidir:

1.Adım: k adet küme merkezlerini temsil eden nesneyi rastgele seç. M_1,M_2,..M_k örnek orta noktaları aşağıdaki gibi hesaplanır (Gersho ve ark. 1991).

$$M_k = \frac{1}{n_k}\sum_{i=1}^{n_k} x_{ik} \quad (3.18)$$

2.Adım: Küme içi değişimlerini (3.19) karesel hata formülü ile hesapla. $e_1, e_2, \dots, e_k$ (Linde vd., 1980).

$$e_i^2 = \sum_{i=1}^{n_k} (x_{ik} - M_k)^2 \tag{3.19}$$

K kümesini içeren bütün küme uzayı için karesel-hata, küme içindeki değişimlerin toplamıdır. O halde kare-hata değeri formül (3.20) deki gibi hesaplanır.

$$E_k^2 = \sum_{k=1}^{K} e_k^2 \tag{3.20}$$

3.Adım: Her bir veri nesnesini kendisine en yakın kümeye ata.
4.Adım: Verilerin tümü en yakın kümeleye atandığında tekrar *k* adet küme için merkezleri hesapla.
5.Adım: Küme merkezlerinde bir değişiklik olmayıncaya kadar 2. ve 3. Adımları tekrarla.

3.1.6. Apriori Algoritması

Apriori algoritması, 1994 yılında Agrawal ve Srikant tarafından geliştirilmiştir (Agrawal 1994). En çok tercih edilen birliktelik kuralı algoritmasıdır. Apriori algoritmasındaki temel mantık şu şekildedir: ” Eğer *k*-öğe kümesi minimum destek değerini sağlıyorsa, bu kümenin alt kümeleri de minimum destek değerini sağlar”.

Algoritma sonucunda tespit edilen her kural, destek ve güven kriterleri ile açıklanır. Destek değeri, öğelerin arasındaki birlikteliklerin sıklığını ifade ederken güven değeri ise bu birliktelik kurallarının doğruluğunu ifade eder. *A* ve *B*, birbirinden farklı öğe kümesi olsun. *A* kümesi için destek değeri, *A* öğe kümesini kapsayan kümelerin tüm öğe kümelerine oranıdır ve (3.21) denklemi ile hesaplanır. *A* ve *B* öğe kümeleri için destek değeri, tüm kümeler içerisinde birlikte bulunma olasılığıdır ve (3.22) denklemi ile hesaplanır. *B* öğe kümesinin hangi olasılık değeri *A* öğe kümesi içerinde bulunacağı güven değeri ile ifade edilir ve (3.23) ve (3.24) denklemlerinden herhangi biri ile hesaplanabilir (Güngör ve ark. 2013).

$$Destek(A) = \frac{A \text{ öğe küme sayısı}}{Toplam \text{ öğe küme sayısı}} \quad (3.21)$$

$$Destek(A,B) = \frac{(A,B) \text{ öğe küme sayısı}}{Toplam \text{ öğe küme sayısı}} \quad (3.22)$$

$$Güven(A,B) = \frac{(A,B) \text{öğe küme sayısı}}{A \text{ öğe küme sayısı}} \quad (3.23)$$

$$Güven(A,B) = \frac{Destek(A,B)}{Destek(A)} \quad (3.24)$$

Algoritma sonucunda elde edilen kuralların güvenilirliği ve doğruluğu destek ve güven değerleri ile belirlenir. Kümeler arası birlikteliğin yapılması için hem güven hem de destek değeri yüksek olmalıdır. Minimum destek ve güven değerleri algoritmanın başlangıcından önce belirlenmelidir (Güngör ve ark. 2013).

3.2 Karar Destek Yazılımı Geliştirme Süreçleri

Tez kapsamında geliştirilen uygulama dışarıdan farklı formatlarda da alınabilen veri setine veri madenciliği yöntemlerini uygulayarak kullanıcıya karar verme sürecinde destek olabilecek yorumlar sunmaktadır. Arayüz yardımı ile son kullanıcıya sadece başlıca kontrolleri sunan uygulama kolay anlaşılabilir ve geliştirilebilir özelliklerdedir.

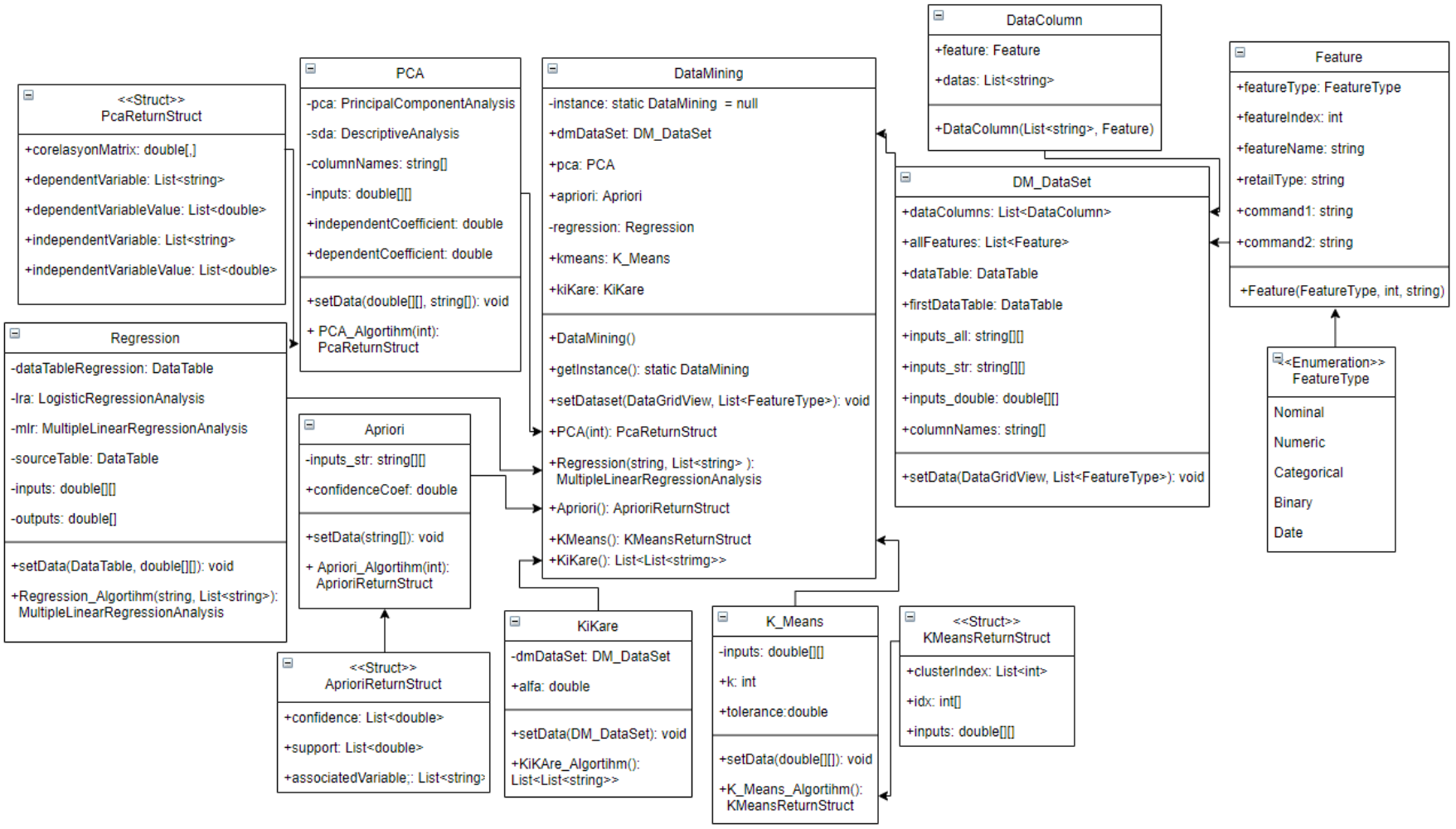
Analiz: *Sistem gereksinimleri;* Kullanıcının dışarıdan veri setlerini alabileceği, veri setindeki özniteliklerin türünü seçebileceği ve gerektiğinde çıkarabildiği, uygulamak istediği algoritmayı ve algoritmanın dışarıdan aldığı değişkenleri seçebileceği, algoritma uygulandıktan sonra sonuçlarının gösterildiği bir uygulama geliştirilmiştir.

Donanım; Yürütülebilir bir uygulama olarak tasarlanan program birçok kullanıcı bilgisayarında çalışabilir durumdadır. Veri setini tanıyan her kullanıcı uygulamayı kullanarak algoritma sonucuna ait yorumları anlayabilir şekilde arayüz tasarımları gerçekleştirilmiştir.

Sistem kısıtlamaları; Uygulamada Windows7 ve üzeri işletim sistemi bulunduran bilgisayarlarda çalışmaktadır. Veri madenciliği algoritmaları için kullanılan *Accord.NET* kütüphanesi için *.NET Framework 4.6* ve üzeri versiyonları gerekmektedir.

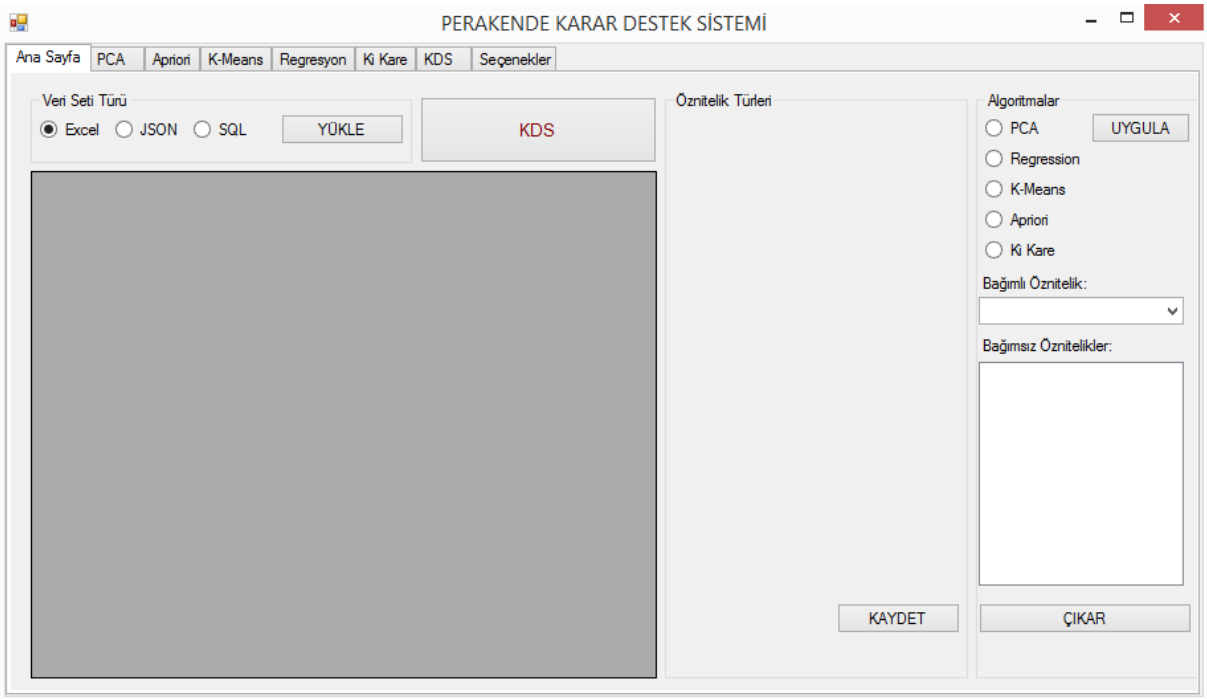
Tasarım: Yazılım mimarisi üç katmandan oluşmaktadır. Alt katman olarak veri madenciliği algoritmaları ve kütüphanelerini içeren "Algoritma Katmanı", veri setini öznitelikleri, türlerini, veri seti sütunlarının ve veri madenciliği sonuçlarının tutulduğu "Veri Katmanı" ve kullanıcı ara yüzü ve araçların olaylarındaki işlemleri içeren "Arayüz Katmanı"dır (Şekil 3.2). Uygulamanın sınıf diyagramı Şekil 3.3'teki gibidir.

Şekil 3.2. Karar destek sistemi yazılım mimarisi



Şekil 3.3. Karar destek sistemi uygulaması sınıf diyagramı

Karar destek sistemi uygulamasının kullanıcı arayüzü Şekil 3.4'teki gibidir. Kullanıcı Excel, JSON formatlarında ve SQL komutu ile veri yükleyebilir. Yüklenen verideki öznitelik sayısına göre "*Öznitelik Türleri*" bölümünde dinamik olarak etiket ve açılan kutu (*combobox*) araçları eklenmektedir. Bu araçlardan öznitelik tür seçimleri yapılmaktadır. Türleri seçilen öznitelikler kaydedilerek algoritma seçimi yapılarak seçilen algoritma uygulanabilmektedir. Çıkartılmak istenen öznitelikler işaretlenebilir liste (*checkedlist*) aracından seçilerek çıkarılabilir.

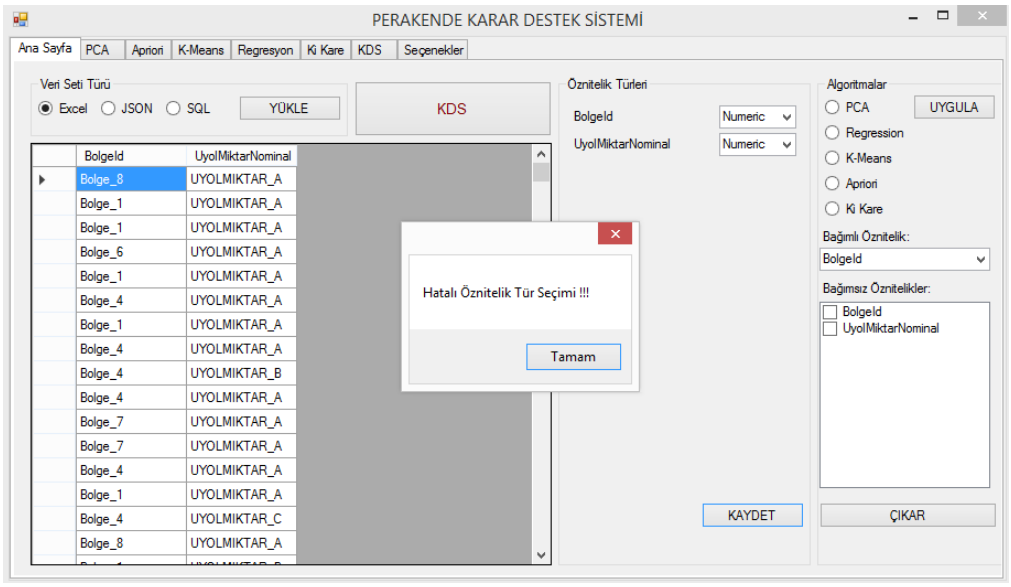


Şekil 3.4. Karar destek sistemi kullanıcı arayüzü

Kodlama: Uygulama Microsoft Visual Studio Enterprise 2017 platformunda C# programlama dili ile geliştirilmiştir. PCA, Regresyon, K-menas, Apriori ve Ki kare algoritmaları için *Accord.NET* (Anonim 2020) makine öğrenmesi ve istatistik kütüphanesi kullanılmıştır.

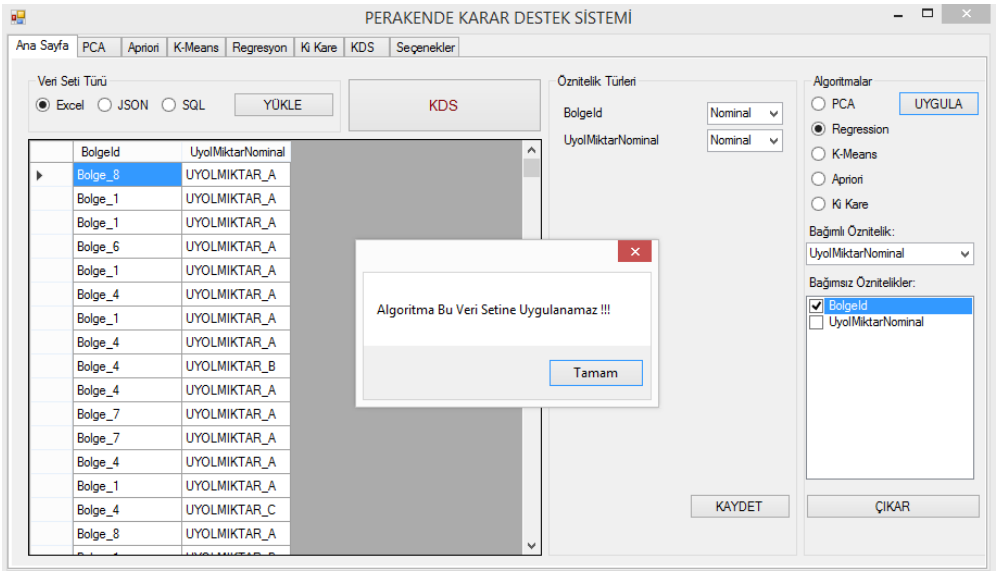
Test: Uygulama içerisinde bazı kontrollerin ve yapılan arka plan işlemlerinin doğruluğunu test etmek için algoritmalar farklı veri setleri üzerinde test edilmiştir.

- *Nominal/Nümerik Öznitelik Tür Kontrolü:* Veri seti yüklendikten sonra tür seçimi alanında geçersiz seçimini önlemek için verinin tür kontrolü yapılmaktadır. Hatalı tür seçimi olması halinde kullanıcı uyarılmaktadır (Şekil 3.5).



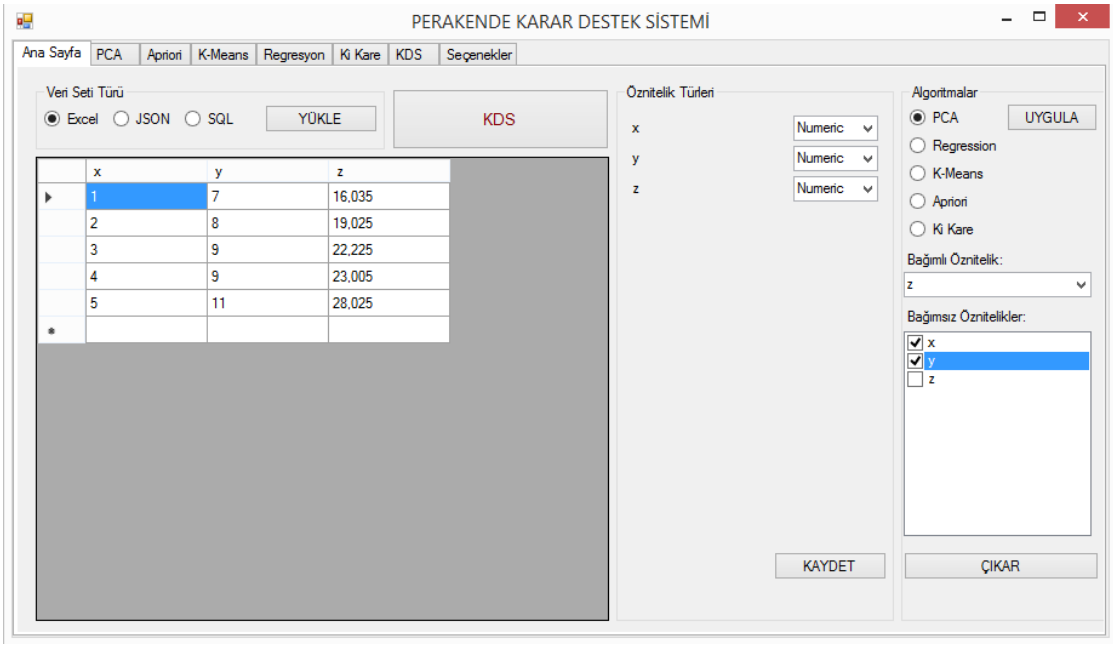
Şekil 3.5. Öznitelik tür kontrolü

- *Veri Seti Algoritma Uyumu Kontrolü:* Bazı algoritmalar sayısal veri türünü desteklerken bazıları metin ya da kategorik veri türünü desteklemektedir. Eğer bir veri setine desteklenmeyen bir algoritma uygulanmak istenirse kullanıcı uyarılmaktadır (Şekil 3.6).

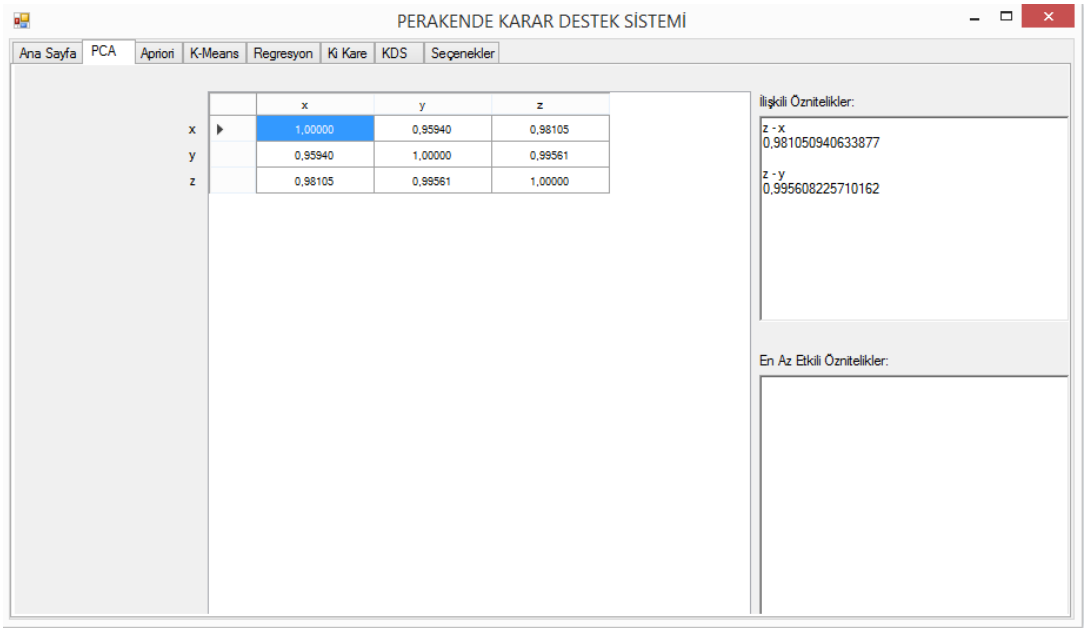


Şekil 3.6. Veri seti algoritma uyum kontrolü

- *PCA Algoritması:* Test veri setindeki özniteliklerin birbirleri ile ne derecede ilişkili olduğunu veya "*Bağımlı Öznitelik*" açılır kutusundan (Şekil 3.7) seçilen bağımlı değişkeni an az etkileyen diğer öznitelikleri açıklamaktadır (Şekil 3.8). İlişkili özniteliklerin güven alt limiti 0.7, en az etkili özniteliklerin güven üst limiti 0.1'dir. Bu değişkenler *"Seçenekler"* sekmesinden değiştirilebilir.

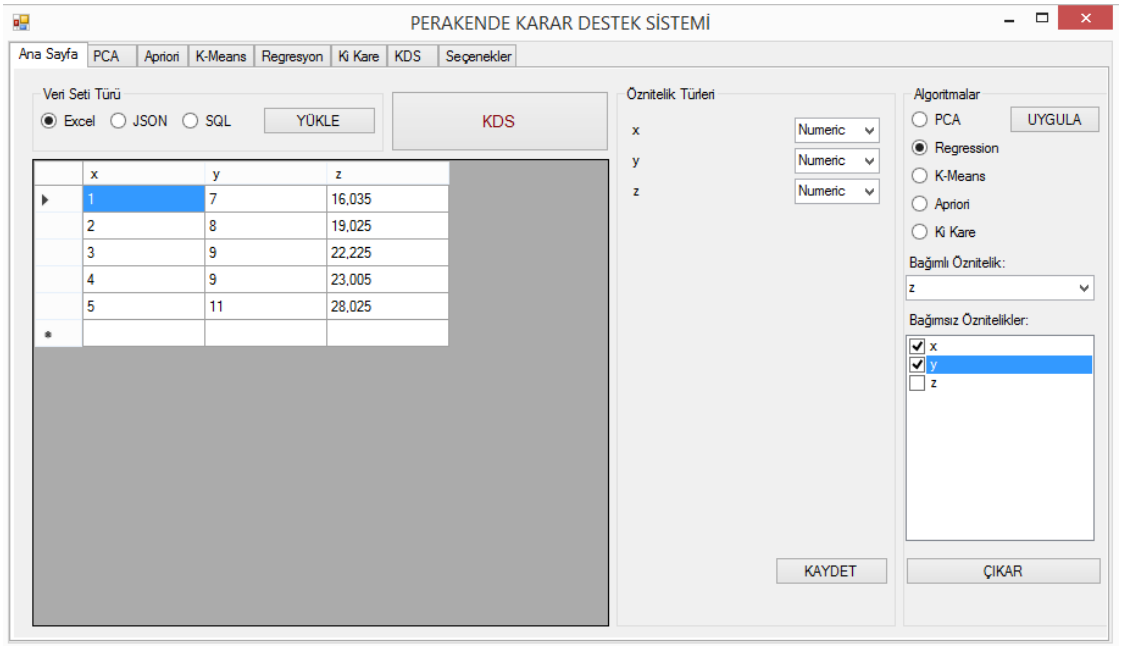


Şekil 3.7. PCA test veri seti

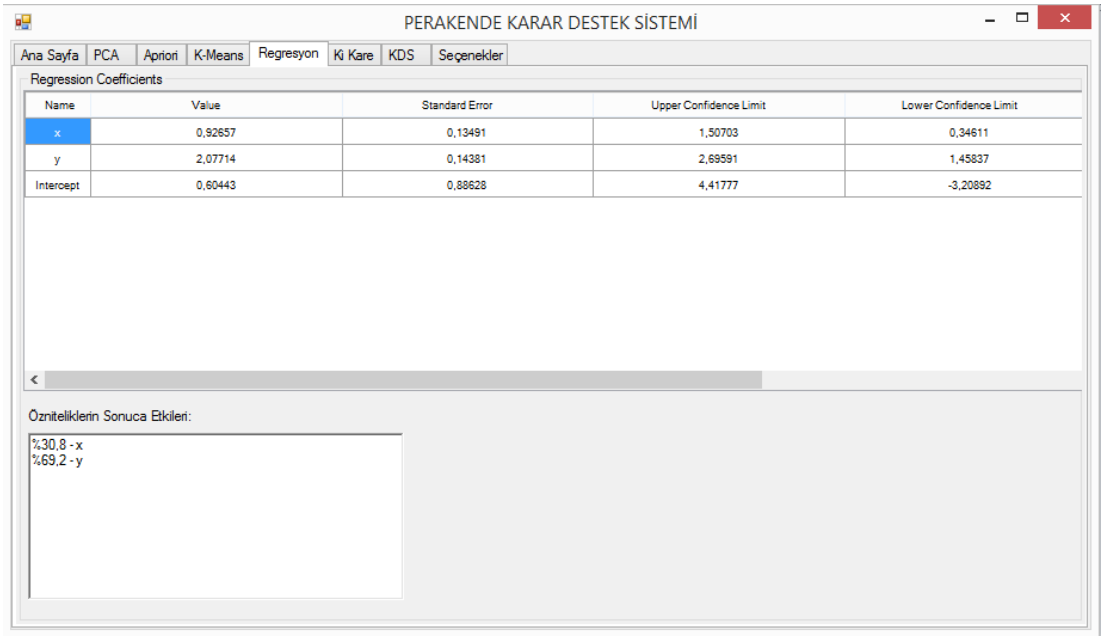


Şekil 3.8. PCA test uygulama sonucu

- *Lineer Regresyon Algoritması:* Test veri seti üzerinde bağımsız özniteliklerin bağımlı özniteliği etkileme oranını kullanıcıya sunan adımdır. Şekil 3.9'daki örnekte *z* özniteliği bağımlı, *x* ve *y* öznitelikleri bağımsız olarak seçilmiştir. Algoritma uygulandığında *x* ve *y* değişkenlerinin *z*'yi hangi oranda (%) etkilediği sonucu elde edilmektedir (Şekil 3.10).

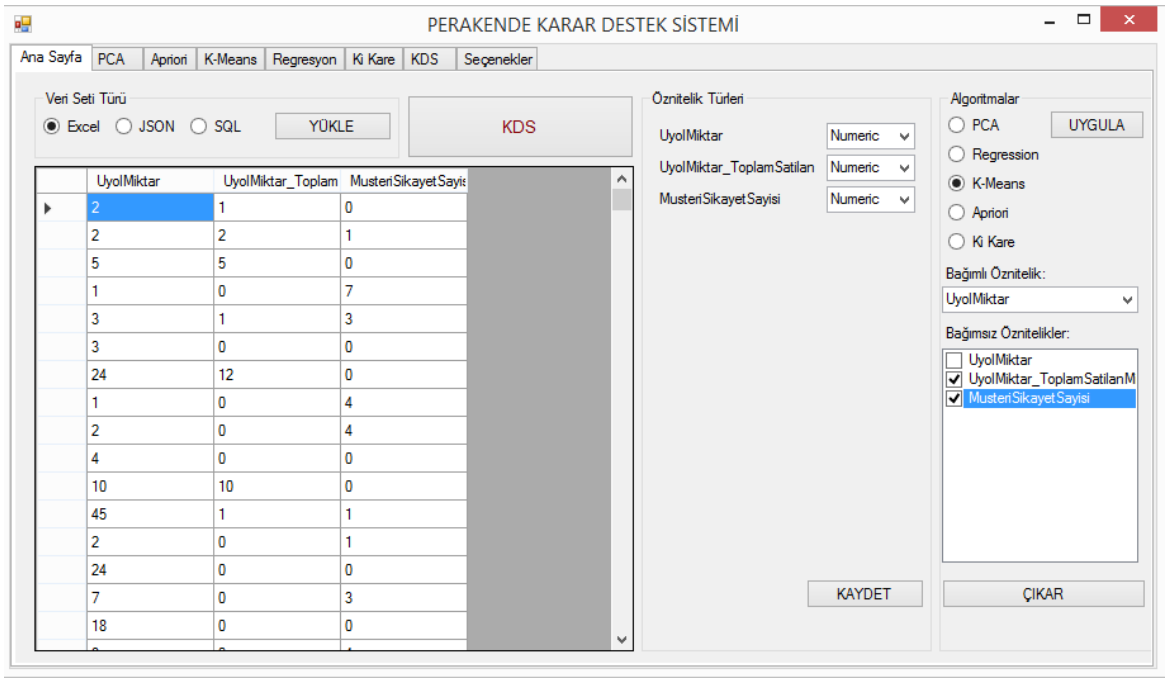


Şekil 3.9. Regresyon test veri seti

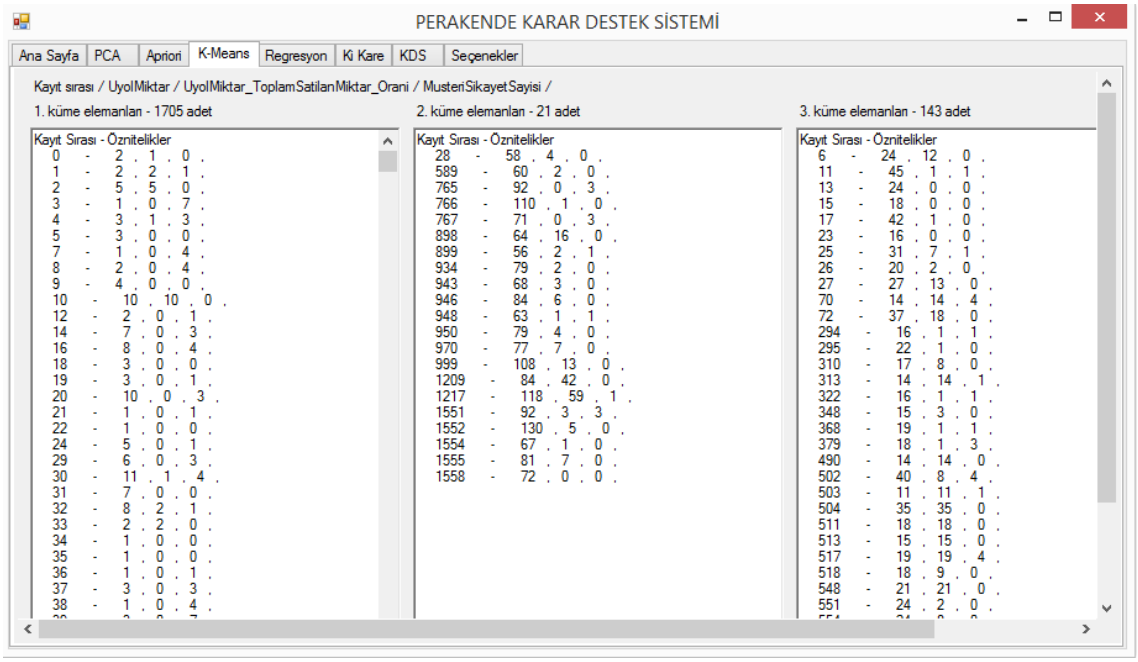


Şekil 3.10. Regresyon test uygulama sonucu

- *K-Means Algoritması:* Gerçek veri setinin bir bölümü alınıp uygulamaya yüklenerek "*K-Means*" algoritması uygulanmıştır (Şekil 3.11). Varsayılan olarak küme sayısı 3 olarak belirlenmiştir. Elde edilen kümeler ve kümelere ait kayıt sayıları Şekil 3.12'de gösterilmiştir. *"Seçenekler"* sekmesinde *k-küme* sayısı değiştirilebilir bir değişkendir.

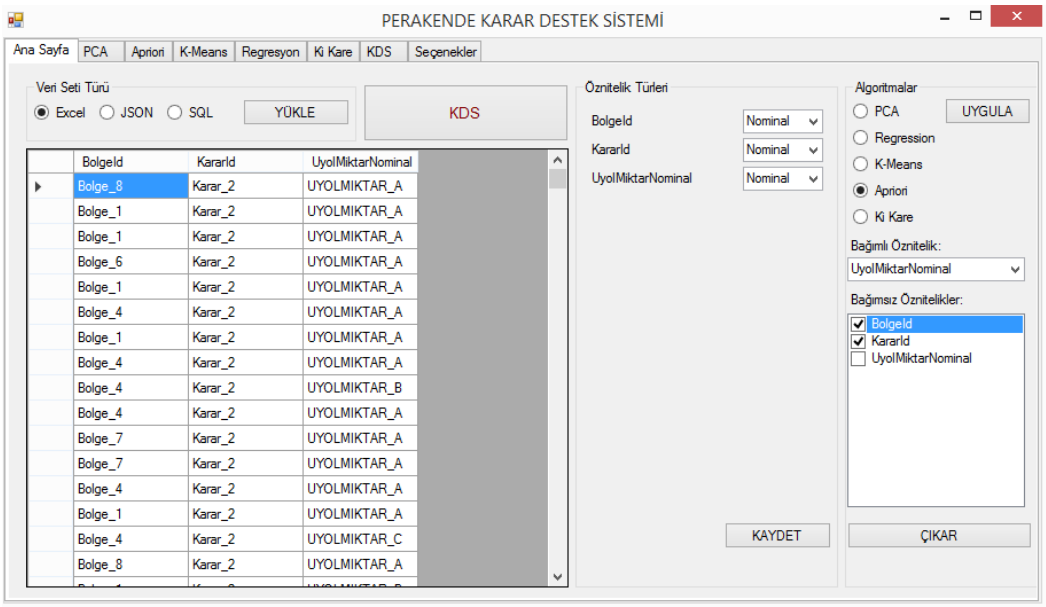


Şekil 3.11. K-means test veri seti

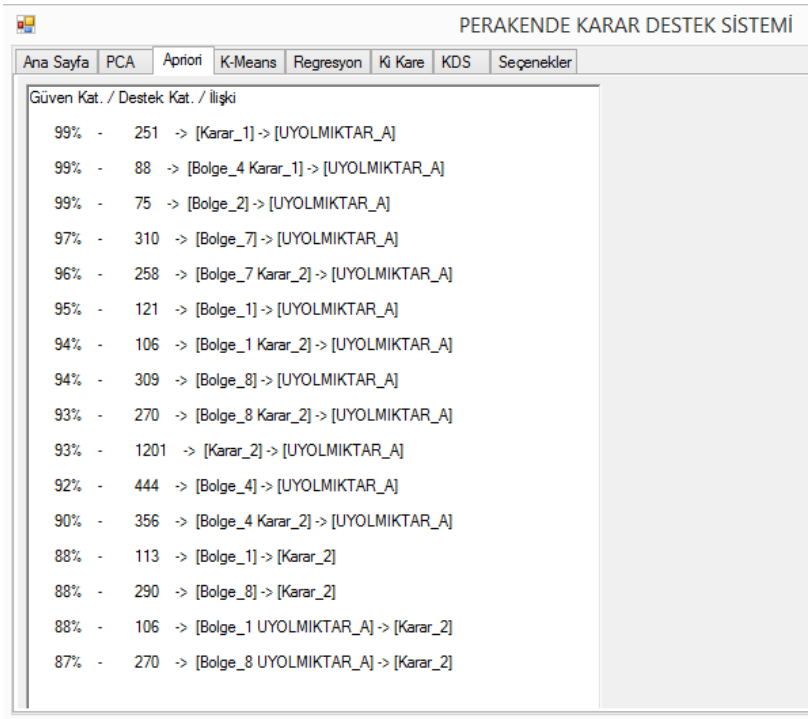


Şekil 3.12. K-means test uygulama sonucu

- *Apriori Algoritması:* Gerçek veri seti üzerinden alınan bir bölüm veri uygulamaya yüklenmiştir. Apriori algoritması öz nitelik türleri "*Nominal*" veya "*Categorical*" türünde olmalıdır (Şekil 3.13). Bu algoritmada birbiri ile bağlantılı olan özellikler, güven ve destek katsayıları ile birlikte kullanıcıya raporlanmaktadır (Şekil 3.14).

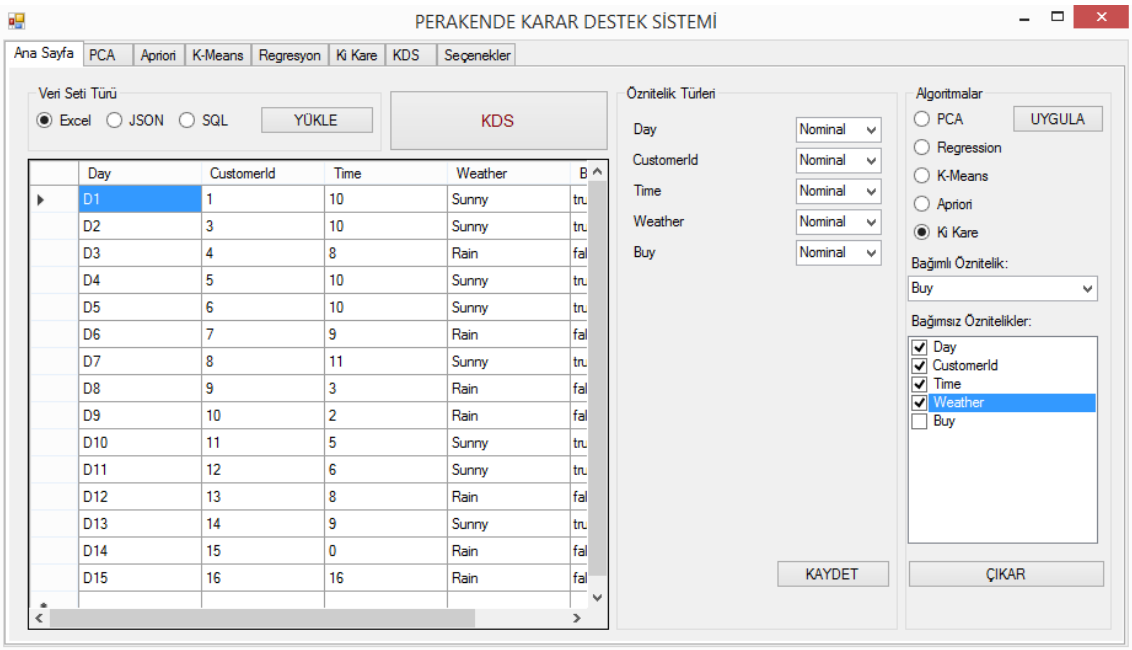


Şekil 3.13. Apriori test veri seti

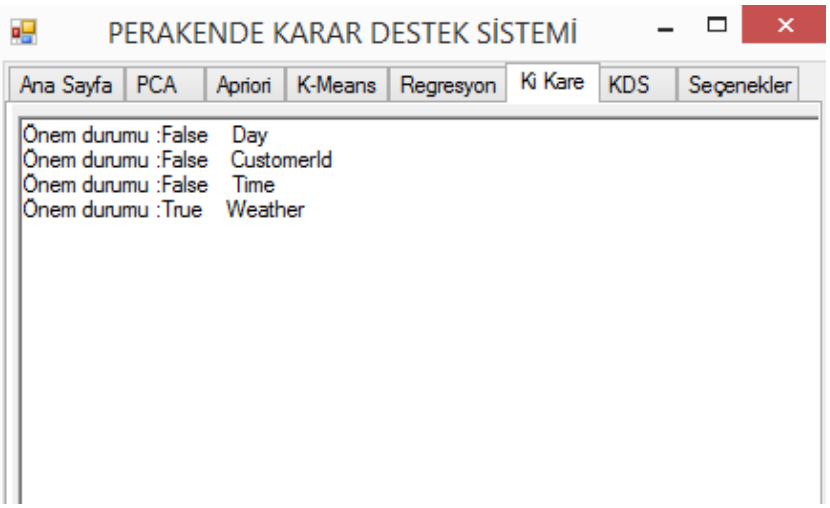


Şekil 3.14. Apriori test uygulama sonucu

- *Ki Kare Algoritması:* Test veri seti üzerinde bağımsız özniteliklerin bağımlı öznitelik üzerinde etkili olup olmadığını kullanıcıya sunan adımdır. Şekil 3.15'teki örnekte *Buy* özniteliği bağımlı, diğer öznitelikleri bağımsız olarak seçilmiştir. Algoritma uygulandığında bağımsız özniteliklerin bağımlı özniteliği etkileyip etkilemediği sonucu raporlanmıştır (Şekil 3.16). Ki kare algoritmasının "*anlamlılık değeri(α)*" varsayılan olarak 0.05'tir. *"Seçenekler"* sekmesinden değer değiştirilebilir.

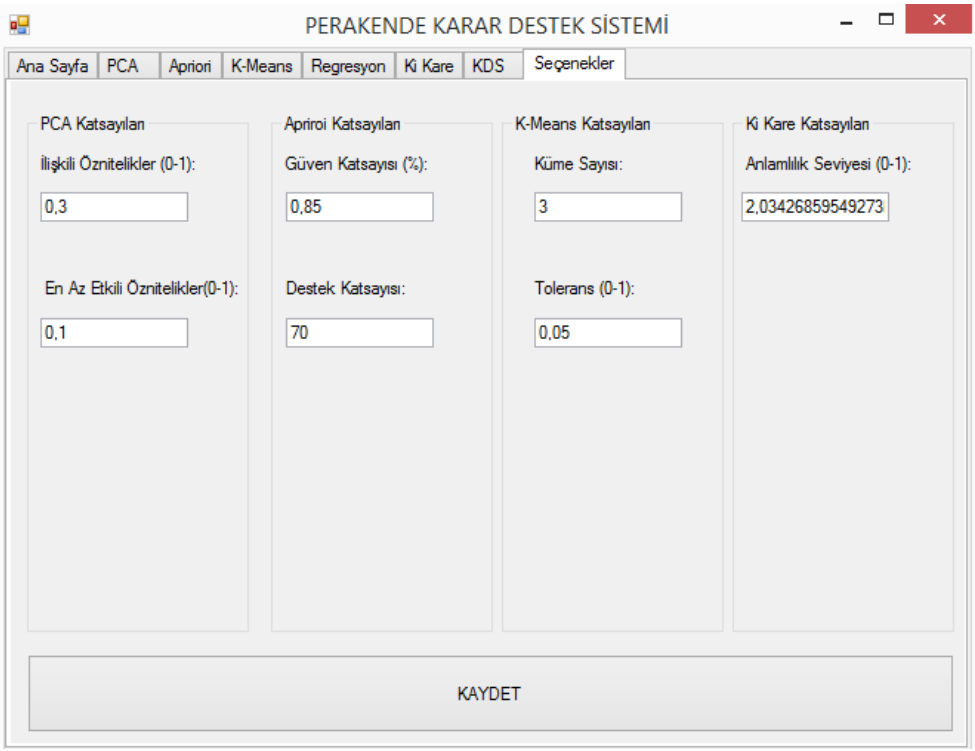


Şekil 3.15. Ki kare test veri seti



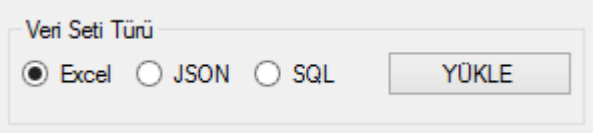
Şekil 3.16. Ki kare test uygulama sonucu

- *Varsayılan Algoritma Katsayılarının Değiştirilmesi:* PCA, apriori, k-means ve ki kare algoritmalarında kodun içerisinde varsayılan olarak girilen katsayıların kullanıcı tarafından değiştirilebilmesini sağlayan bölümdür (Şekil 3.17). PCA için Şekil 3.8'deki *"İlişkili Öznitelikler"* ve *"En Az Etkili Öznitelikler"* için sınır değerleri, apriori algoritmasında *"Güven Katsayısı"* ve *"Destek Katsayısı"*, K-means algoritmasında *"Küme Sayısı"* ve *"Tolerans"* değerlerinin, ki kare analizi için ise *"Anlamlılık Seviyesi (α)"* değerlerinin değiştirilebilmesini sağlar.



Şekil 3.17. KDS seçenekler

- *Excel - JSON - SQL Veri Seti Türleri:* Uygulama testleri Excel(*.xls*) ve JSON formatlarını, SQL komutunu desteklemektedir (Şekil 3.18).



Şekil 3.18. Veri seti türleri

Bakım: KDS uygulaması arayüz katmanı, veri katmanı ve algoritma katmanından oluşmaktadır. Her katman birbirinden bağımsızdır. Katmanlar ayrıldığında farklı uygulamalarda da kullanılabilir yapıdadır. Tasarımda katman birbirinden bağımsız olduğu için bakım ve geliştirme aşamalarında katmanlar birbirinden minimum düzeyde etkilenecektir.

Uygulamanın yeni sürümünde yapılması gereken bazı güncellemeler gerekecektir. İlk olarak K-means algoritmasının daha kolay anlaşılabilir olması ve görselleştirilebilmesi için veri 2 boyutlu grafiklere ayrılarak kümeler gösterilebilir. Mevcut arayüzde bu algoritma sonucu liste şeklinde kullanıcılara sunulmaktadır. Diğer bir iyileştirme uyarı ekranlarında yapılabilir. Uyarı türü ve uyarı metinleri bir sınıfta tutularak, gerekli durumlarda uyarı türü ile çağrılan uyarılar tek formatta daha düzenli ve kolay bir kullanım sağlanabilir. Önemli diğer bir planlama ise veri katmanındaki veri seti formatında olacaktır. "*DataTable*" formatında tutulan kayıtlar, okunup yazılabilmesi daha kolay ve esnek olan JSON formatına dönüştürülebilir.

4. BULGULAR

Mevcut çalışmadan kullanılan ham veriler şirketin ERP programından ve şirket raporlama yazılımlarından toplanmıştır. Elde edilen veriler SQL veri tabanına aktarılmıştır. Şubelerin düzensiz olarak girdiği mağaza fire verileri şirket bünyesinde ayda 1 defa raporlanarak değerlendirilmektedir. Bu sebeple aynı yıl ve ay içerisinde farklı günlerde stok kodu, depo, nedeni, yönlendirme adımı ve kararı aynı olan kayıtların fire miktarları toplanarak yeni kayıt verileri oluşturulmuştur. Oluşturulan öznitelikler Çizelge 4.1'deki gibidir.

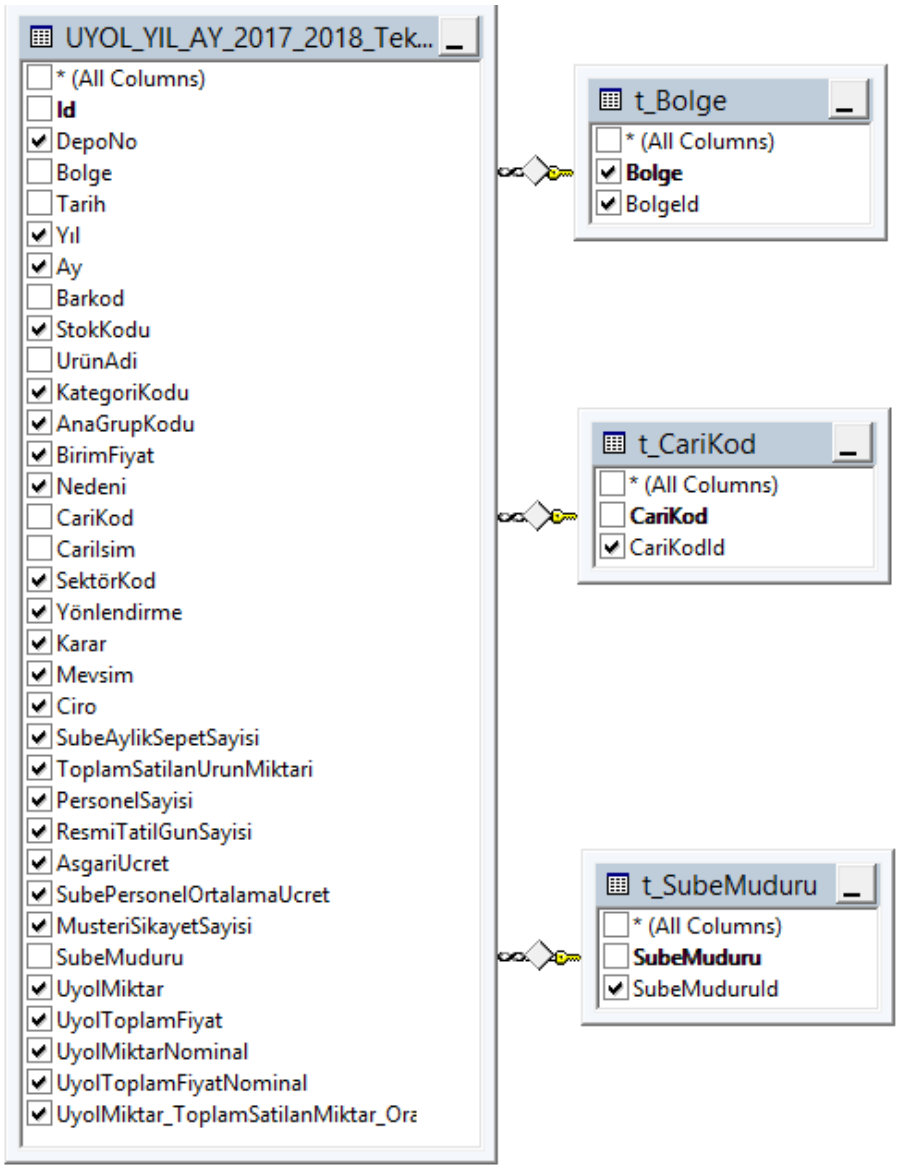
Çizelge 4.1. Öznitelikler ve türleri

Öznitelik	**Tür**
Id	int
DepoNo	nvarchar(3)
Bolge	nvarchar(50)
Tarih	datetime
Yıl	nvarchar(50)
Ay	nvarchar(50)
Barkod	nvarchar(MAX)
StokKodu	nvarchar(50)
UrünAdi	nvarchar(MAX)
KategoriKodu	nvarchar(50)
AnaGrupKodu	nvarchar(50)
BirimFiyat	money
Nedeni	nvarchar(50)
CariKod	nvarchar(50)
CariIsim	nvarchar(MAX)
SektörKod	nvarchar(MAX)
Yönlendirme	nvarchar(50)
Karar	nvarchar(50)
Mevsim	nvarchar(50)
Ciro	money
SubeAylikSepetSayisi	int
ToplamSatilanUrunMiktari	int
PersonelSayisi	int
ResmiTatilGunSayisi	int
AsgariUcret	money
SubePersonelOrtalamaUcret	money
MusteriSikayetSayisi	int
SubeMuduru	nvarchar(100)
UyolMiktar	int
UyolToplamFiyat	money

Çizelge 4.1'deki durumda

- Bölge
- Şube Müdürü
- Cari

öznitelikleri gizlilik sebebi ile kategorik veriye dönüştürülmüştür. Bu işlem için SQL veri tabanında kategorik olması planlanan nitelikler için tablolar oluşturularak yabancı anahtar ile ana tabloya bağlanmıştır. Yapılan işlemlerin tek bir tabloda görünebilmesi için SQL *View* oluşturulmuştur (Şekil 4.1).



Şekil 4.1. SQL View görünümü

Son olarak VM yöntemleri uygulanırken etiket olarak kabul edilecek olan öznitelik "*UyolMiktar*" sütunudur. Bu öznitelik tabloda nümerik türünde tutulmaktadır. Sınıflandırma algoritmalarının bazılarında etiketi nominal olarak kullanma durumu istenmektedir. Böyle durumlar için "*UyolMiktar"* ve "*UyolToplamFiyat"* özniteliklerini nominale çevirme işlemi yapılmıştır (Çizelge 4.2).

Çizelge 4.2. Sayısal tür – kategorik tür dönüşümü

UyolMiktarı	**Nominal**
1-20	A
21-40	B
41-60	C
61-80	Ç
81-100	D
101-120	E
121-140	F
141-160	G
161-180	Ğ
181-200	H
201-220	I
221-240	İ
241-260	J
261-280	K
281-300	L
301-320	M
321-340	N
341-360	O
361-380	Ö
381-400	P
401-420	R
421-440	S
441-460	Ş
461-480	T
481-500	U
501-520	Ü
521-540	V
541-560	Y
561-580	Z
581-600	W

UyolToplamFiyat	**Nominal**
0-50	A
51-100	B
101-150	C
151-200	Ç
201-250	D
251-300	E
301-350	F
351-400	G
401-450	Ğ
451-500	H
501-550	I
551-600	İ
601-650	J
651-700	K
701-750	L
751-800	M
801-850	N
851-900	O
901-950	Ö
951-1000	P
1001-1050	R
1051-1100	S
1101-1150	Ş

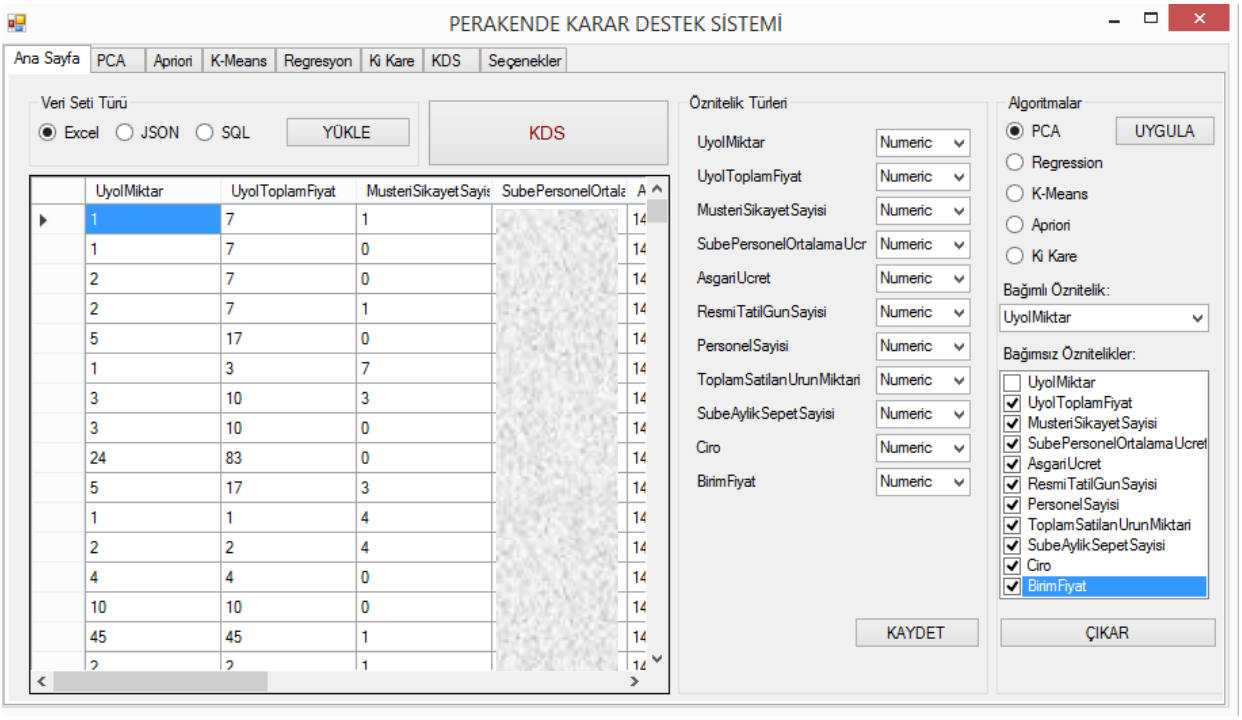
Son veri önişlemeden sonra tablodaki öznitelikler ve türleri aşağıdaki gibidir.

Nominal Öznitelikler: *DepoNo, BolgeId, Yıl, Ay, StokKodu, KategoriKodu, AnaGrupKodu, Nedeni, CariKodId, SektörKod, Yonlendirme, Karar, Mevsim, SubeMuduruId, UyolMiktarNominal, UyolToplamFiyatNominal*

Nümerik Öznitelikler: *UyolMiktar, UyolToplamFiyat, UyolMiktar_ToplamSatilan UrunMiktari, MusteriSikayetSayisi, SubePersonelOrtalamaUcret, AsgariUcret, ResmiTatilGunSayisi, PersonelSayisi, ToplamSatilanUrunMiktari, SubeAylikSepet Sayisi, Ciro, BirimFiyat*

Veri madenciliği algoritmalarının uygulanması aşamalarında KDS uygulamasına destek olması açısından Weka yazılımından yararlanılmıştır. Weka ile veri seti üzerinde test edilerek verimli sonuç alınan algoritmalar yazılıma dahil edilmiştir.

PCA: Nümerik verilere "UyolMiktar" etiket olarak kabul edilerek PCA uygulandığında (Şekil 4.2) Çizelge 4.3'teki korelasyon matrisi elde edilmiştir.

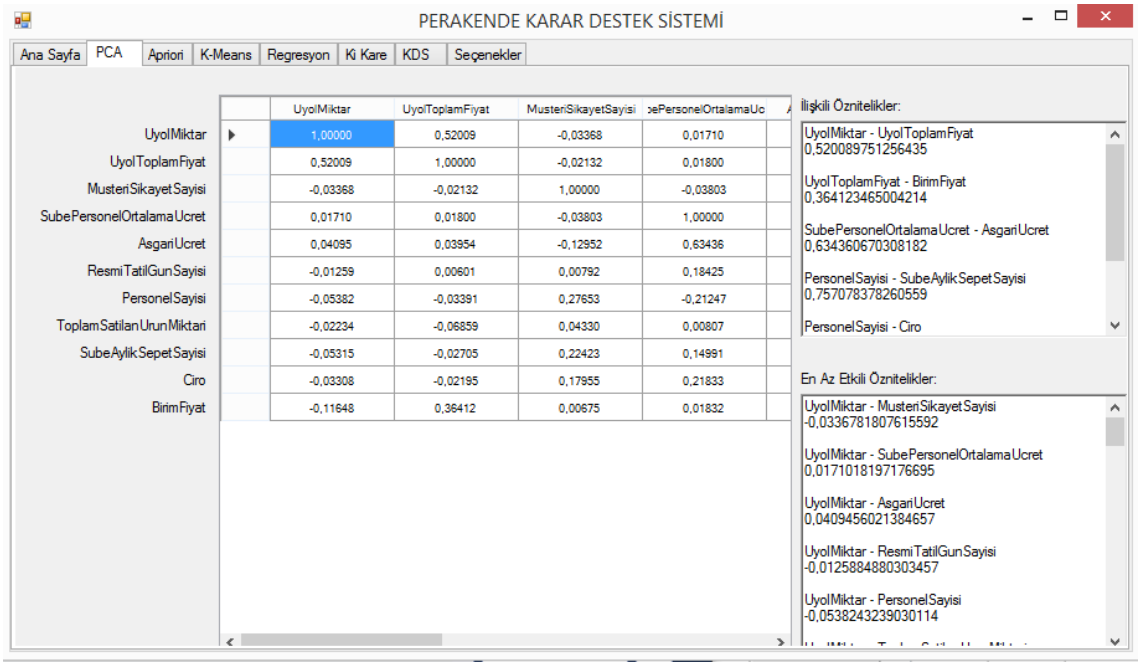


Şekil 4.2. PCA veri seti ve türleri

Çizelge 4.3. PCA korelasyon matrisi

	Uyol Miktar	Uyol Toplam Fiyat	UyolMiktar ToplamSatilanMiktar Orani	Müşteri Şikayet Sayısı	Şube Personel Ortalama Ücret	Asgari Ücret	Resmi Tatil Gün Sayısı	Personel Sayısı	Toplam Satılan Ürün Miktarı	Şube Aylık Sepet Sayısı	Ciro	Birim Fiyat
Uyol Miktar	1	0,51	0,49	-0,03	0,02	0,04	-0,01	-0,05	-0,01	-0,05	-0,03	-0,12
Uyol Toplam Fiyat	0,51	1	0,3	-0,02	0,02	0,04	0	-0,03	-0,06	-0,02	-0,02	0,37
UyolMiktar ToplamSatilanMiktar Orani	0,49	0,3	1	-0,03	0	0,03	-0,01	-0,06	-0,06	-0,06	-0,05	-0,04
Müşteri Şikayet Sayısı	-0,03	-0,02	-0,03	1	-0,04	-0,12	0,01	0,26	0,04	0,2	0,16	0
Şube Personel Ortalama Ücret	0,02	0,02	0	-0,04	1	0,66	0,13	-0,23	0,01	0,14	0,22	0,03
Asgari Ücret	0,04	0,04	0,03	-0,12	0,66	1	0,02	-0,23	-0,01	0,02	0,16	0,03
Resmi Tatil Gün Sayısı	-0,01	0	-0,01	0,01	0,13	0,02	1	-0,02	0	-0,01	0,02	0,01
Personel Sayısı	-0,05	-0,03	-0,06	0,26	-0,23	-0,23	-0,02	1	0,13	**0,76**	**0,72**	0,03
Toplam Satılan Ürün Miktarı	-0,01	-0,06	-0,06	0,04	0,01	-0,01	0	0,13	1	0,13	0,14	-0,07
Şube Aylık Sepet Sayısı	-0,05	-0,02	-0,06	0,2	0,14	0,02	-0,01	0,76	0,13	1	**0,8**	0,04
Ciro	-0,03	-0,02	-0,05	0,16	0,22	0,16	0,02	0,72	0,14	0,8	1	0,03
Birim Fiyat	-0,12	0,37	-0,04	0	0,03	0,03	0,01	0,03	-0,07	0,04	0,03	1

KDS uygulaması PCA analizi sonuçları Şekil 4.3'teki gibidir. Birbiri ile ilişkili öznitelikler ve sonucu %1 in altında etkileyen *"En Az Etkili Öznitelikler"* raporlanmaktadır.



Şekil 4.3. PCA KDS uygulaması sonuçları

KDS uygulamasının korelasyon matrisine göre belirlenen sınırın üzerindeki faktörler (Şekil 4.3) ilişkili özniteliklerdir (Çizelge 4.4). Bu ilişkili öznitelikler birbiri ile orantılı olduğu için sonuç üzerindeki etkileri aynıdır. Bu faktörlerden herhangi birinin alınması karar destek sisteminin ileriki aşamalarında karmaşıklığı önleyecektir.

Çizelge 4.4. PCA sonucu ilişkili öznitelikler

Öznitelikler	Oran
PersonelSayisi - SubeAylikSepetSayisi	0,748
PersonelSayisi - Ciro	0,727
SubeAylikSepetSayisi - PersonelSayisi	0,748
SubeAylikSepetSayisi - Ciro	0,794

Aynı şekilde Çizelge 4.5'te "*UyolMiktar*" bağımlı değişkenini en az miktarda etkileyen öznitelikler uygulamanın sonucu olarak verilmiştir. Bu özniteliklerin var olması sonucu

çok az etkilediği için veri setinden çıkarılması sonucun yorumlanması açısından gereklidir.

Çizelge 4.5. PCA sonucu en az etkili öznitelikler

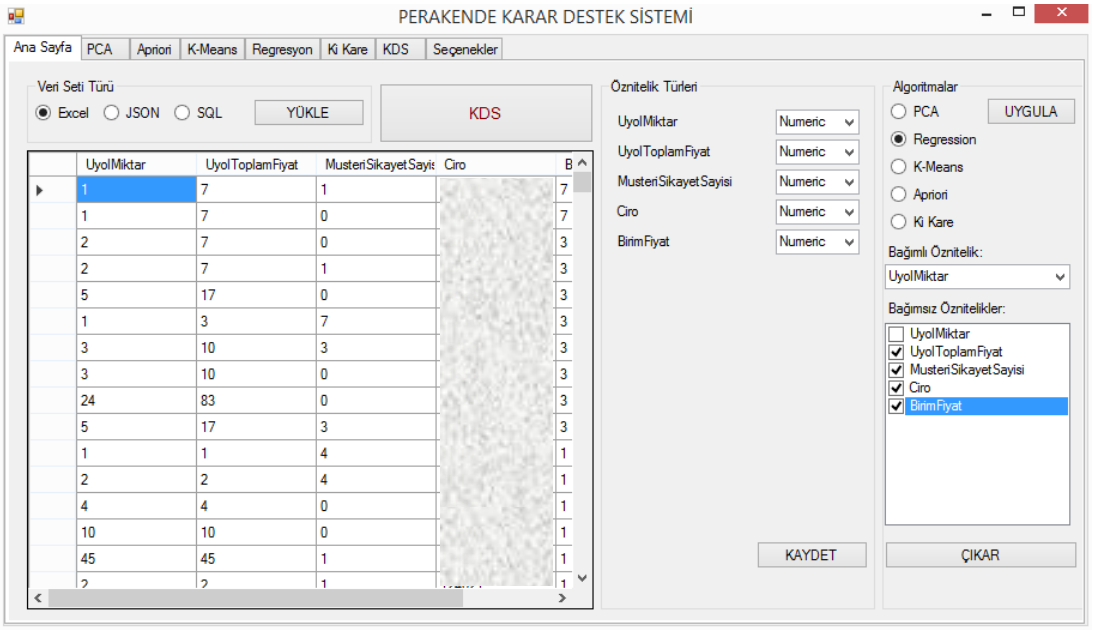
Öznitelikler	Oran
UyolMiktar - MusteriSikayetSayisi	-0,035
UyolMiktar - SubePersonelOrtalamaUcret	0,012
UyolMiktar - AsgariUcret	0,033
UyolMiktar - ResmiTatilGunSayisİ	-0,016
UyolMiktar - PersonelSayisi	-0,041
UyolMiktar - ToplamSatilanUrunMiktari	-0,0005
UyolMiktar - SubeAylikSepetSayisi	-0,0425
UyolMiktar - Ciro	-0,024
UyolMiktar - BirimFiyat	-0,114
UyolToplamFiyat - MusteriSikayetSayisi	-0,0249
UyolToplamFiyat - SubePersonelOrtalamaUcret	0,029
UyolToplamFiyat - AsgariUcret	0,045
UyolToplamFiyat - ResmiTatilGunSayisi	0,0002
UyolToplamFiyat - PersonelSayisi	-0,017
UyolToplamFiyat - ToplamSatilanUrunMiktari	-0,052
UyolToplamFiyat - SubeAylikSepetSayisi	-0,004
UyolToplamFiyat - Ciro	0,004
UyolMiktar_ToplamSatilanMiktar_Orani - MusteriSikayetSayisi	-0,03
UyolMiktar_ToplamSatilanMiktar_Orani - SubePersonelOrtalamaUcret	0,002
UyolMiktar_ToplamSatilanMiktar_Orani - AsgariUcret	0,030
UyolMiktar_ToplamSatilanMiktar_Orani - ResmiTatilGunSayisi	-0,008
UyolMiktar_ToplamSatilanMiktar_Orani - PersonelSayisi	-0,063
UyolMiktar_ToplamSatilanMiktar_Orani - ToplamSatilanUrunMiktari	-0,062
UyolMiktar_ToplamSatilanMiktar_Orani - SubeAylikSepetSayisi	-0,066
UyolMiktar_ToplamSatilanMiktar_Orani - Ciro	-0,0558
UyolMiktar_ToplamSatilanMiktar_Orani - BirimFiyat	-0,0518

Bağımlı Öznitelikler; *UyolMiktarNominal - Uyol Toplam Fiyat Nominal*

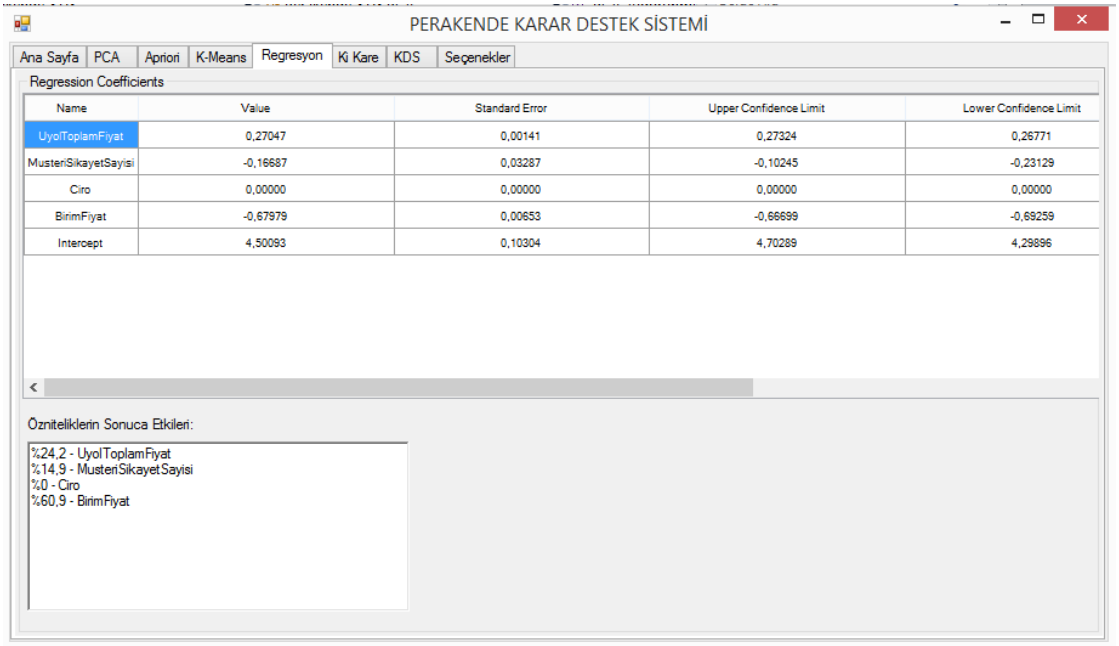
Etkisiz Öznitelikler; *Mevsim, Ay, Yıl, Yönlendirme*

Seçilen Öznitelikler; *UyolMiktar, UyolToplamFiyat, MusteriSikayetSayisi, Ciro, BirimFiyat*

Lineer Regresyon: Nümerik veriler, geliştirilen karar destek sistemi uygulamasında Şekil 4.4'teki gibi "*UyolMiktar*" özniteliği sınıf etiketi (bağımlı öznitelik), diğer değişkenler bağımsız öznitelik olarak seçilerek lineer regresyon algoritması uygulandığında Şekil 4.5'teki sonuçlar elde edilmektedir. Bağımsız özniteliklerin sonucu hangi oranda (%) etkilediği gözlemlenmektedir.

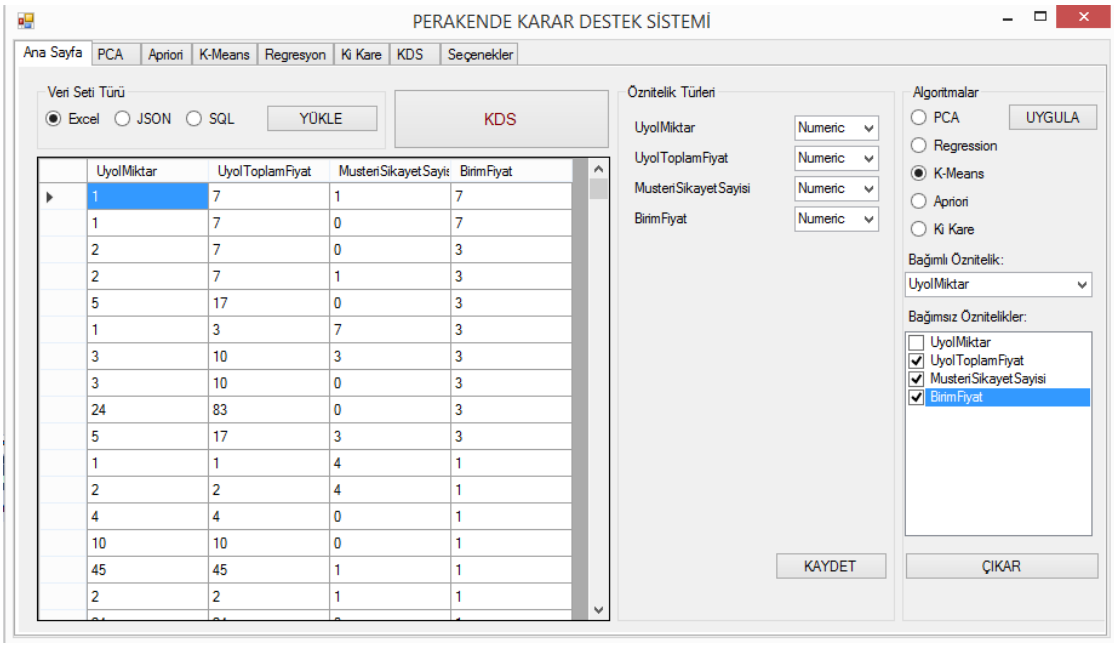


Şekil 4.4. Lineer regresyon veri seti ve türleri

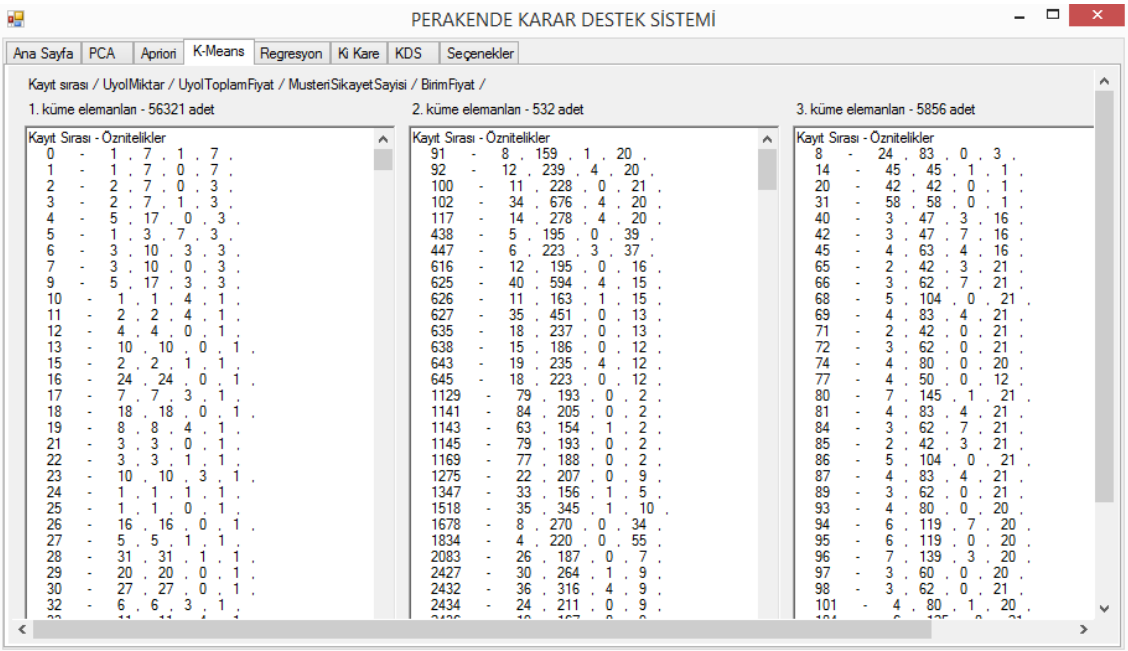


Şekil 4.5. Lineer regresyon KDS uygulaması sonuçları

K-Means: KDS uygulamasında veri setine uygulanan regresyon algoritmasının sonuçlarına göre öznitelik seçimi yapılarak kalan özniteliklere (Şekil 4.6) K-means algoritması uygulandığında veri seti öznitelik özelliklerine göre kümelenmiştir (Şekil 4.7).



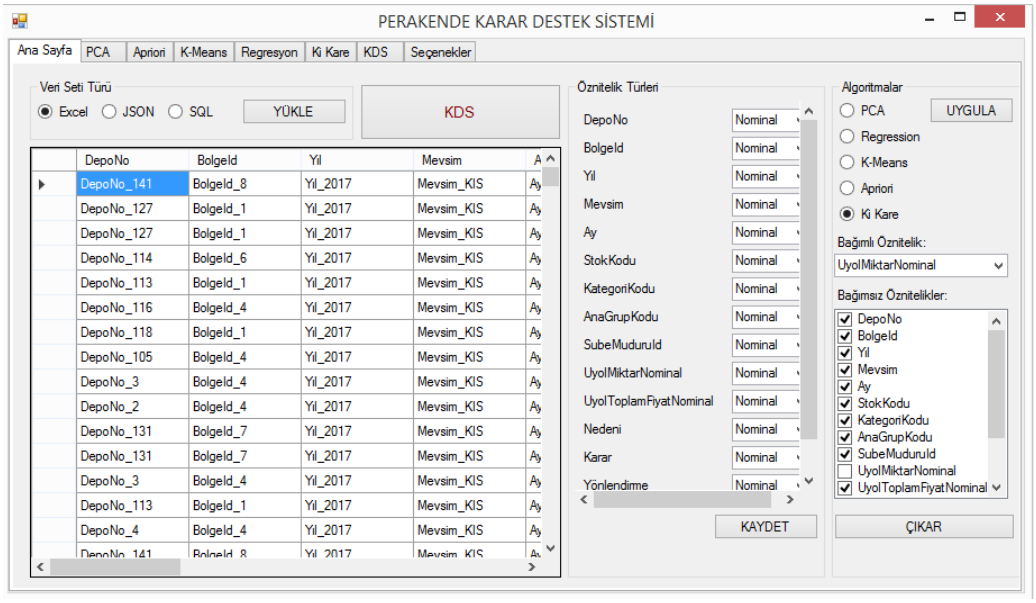
Şekil 4.6. K-means veri seti ve türleri



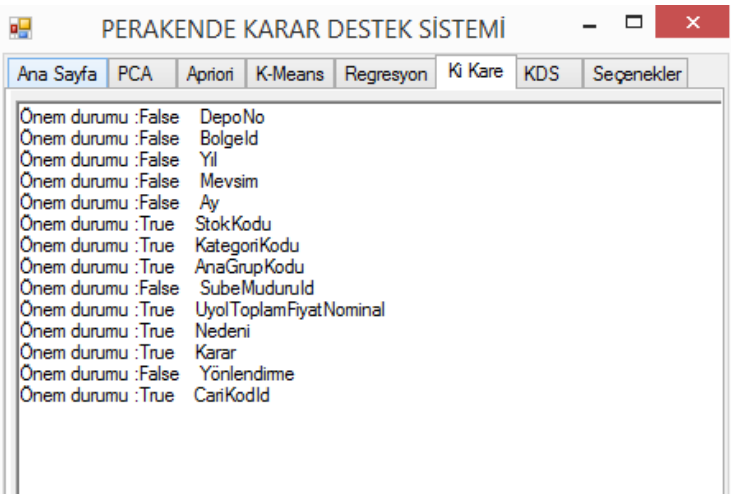
Şekil 4.7. K-means KDS uygulaması sonuçları

M5P: Veri setinde bir karar ağacı algoritması olan M5P yöntemi kullanılmıştır. Algoritma veri seti üzerinde yeterince açıklayıcı olmamıştır.

Ki Kare Analizi: Nominal veri setine (Şekil 4.8) öznitelik seçimi için ki kare algoritması uygulandığında Şekil 4.9'daki sonuçlar elde edilmiştir. "*UyolMiktarNominal*" sınıf etiketi olarak seçilmiştir. Özniteliklerin sonuca etkileri olup olmadığı gözlemlenmektedir (Şekil 4.9).

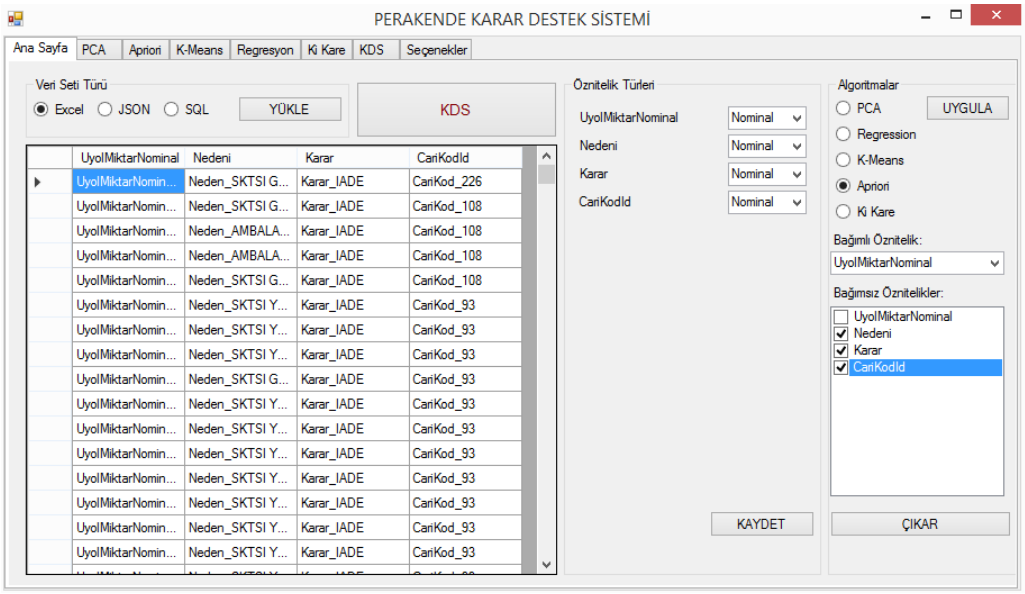


Şekil 4.8. Ki kare veri seti ve türleri

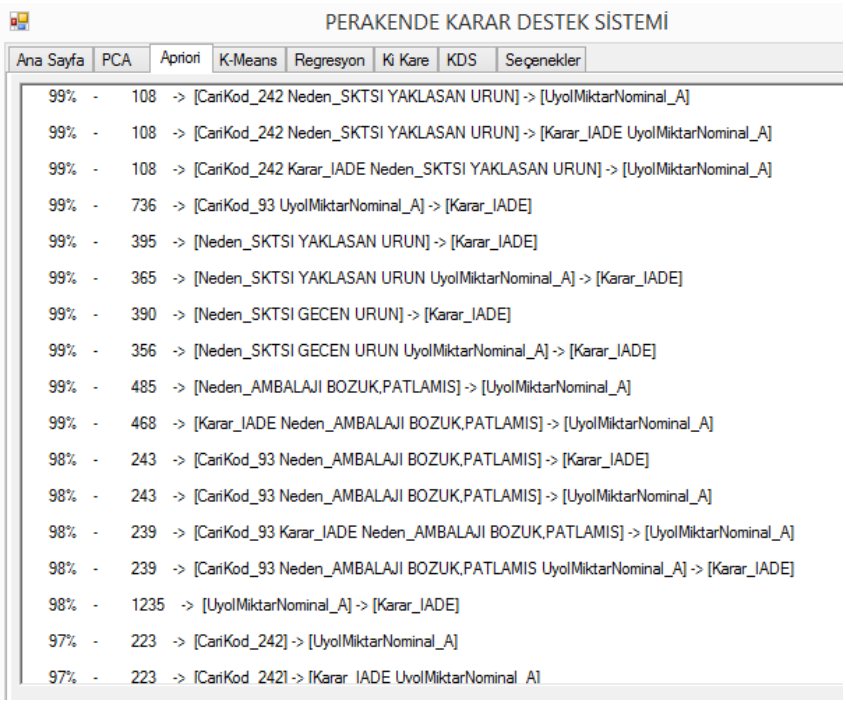


Şekil 4.9. Ki kare KDS uygulaması sonuçları

Apriori: Apriori algoritması KDS uygulamasında nominal verilerin tamamına uygulandığında sonuç üretememiştir. Bu sebeple veri seti ana grup kodu ile filtrelenerek apriori algoritması uygulanmıştır (Şekil 4.10). Güven değeri %85 ve üzeri olan kurallar listelenmiştir (Şekil 4.11). Tüm nominal verilere Weka ile apriori algoritması uygulandığında Çizelge 4.6'daki sonuçlar elde edilmiştir.



Şekil 4.10. Apriori veri seti ve türleri



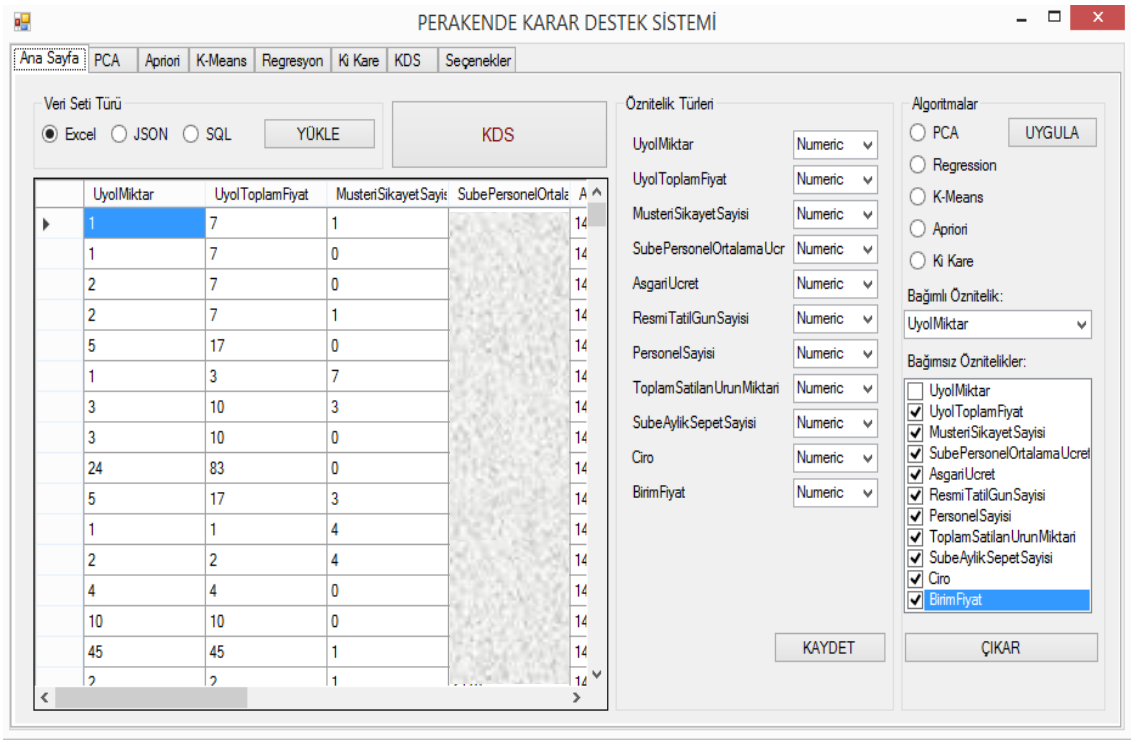
Şekil 4.11. Apriori KDS uygulaması sonuçları

Çizelge 4.6. Apriori algoritması sonucu oluşan kurallar

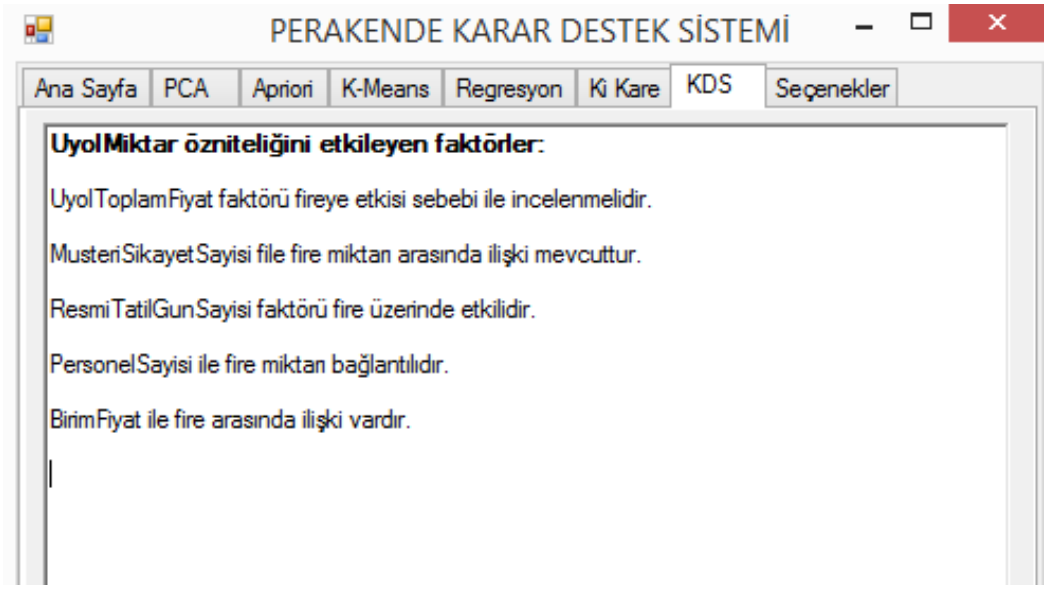
No	Kural	Güven Değeri
1	Nedeni=AMBALAJI BOZUK,PATLAMIS ==> UyolMiktarNominal=A	1
2	BolgeId=4 Nedeni=AMBALAJI BOZUK,PATLAMIS Karar=IADE ==> UyolMiktarNominal=A	1
3	BolgeId=4 Nedeni=AMBALAJI BOZUK,PATLAMIS==> UyolMiktarNominal=A	1
4	Nedeni=AMBALAJI BOZUK,PATLAMIS Karar=IADE ==> UyolMiktarNominal=A	1
5	Karar=FIRE ==> UyolMiktarNominal=A	0,99
6	AnaGrupKodu=25 ==> UyolMiktarNominal=A	0,98
7	AnaGrupKodu=22 ==> UyolMiktarNominal=A	0,98
8	AnaGrupKodu=22 Karar=IADE ==> UyolMiktarNominal=A	097
9	AnaGrupKodu=22 ==> Karar=IADE	0,97
10	AnaGrupKodu=22 UyolMiktarNominal=A ==> Karar=IADE	0,97
11	Nedeni=SKTSI YAKLASAN URUN ==> Karar=IADE	0,96
12	BolgeId=7 Karar=IADE ==> UyolMiktarNominal=A	0,96
13	BolgeId=4 ==> UyolMiktarNominal=A	0,96
14	BolgeId=8 ==> UyolMiktarNominal=A	0,95
15	BolgeId=4 Karar=IADE ==> UyolMiktarNominal=A	0,95
16	BolgeId=8 Karar=IADE ==> UyolMiktarNominal=A	0,95
17	AnaGrupKodu=22 ==> UyolMiktarNominal=A Karar=IADE	0,95
18	Karar=IADE ==> UyolMiktarNominal=A	0,95
19	Nedeni=SKTSI GECEN URUN ==> Karar=IADE	0,95
20	UyolMiktarNominal=A Nedeni=SKTSI GECEN URUN ==> Karar=IADE	0,94
21	Nedeni=SKTSI GECEN URUN ==> UyolMiktarNominal=A	0,91
22	BolgeId=8 ==> Karar=IADE	0,9
23	BolgeId=8 UyolMiktarNominal=A==> Karar=IADE	0.9
24	Nedeni=SKTSI YAKLASAN URUN ==> UyolMiktarNominal=A	0.9
25	BolgeId=7 ==> Karar=IADE	0.9
26	Nedeni=SKTSI YAKLASAN URUN Karar=IADE ==> UyolMiktarNominal=A	0.9
27	UyolMiktarNominal=A ==> Karar=IADE	0.9
28	BolgeId=7 UyolMiktarNominal=A ==> Karar=IADE	0.89
29	BolgeId=4 ==> Karar=IADE	0.88
30	BolgeId=4 UyolMiktarNominal=A ==> Karar=IADE	0.88
31	Nedeni=SKTSI GECEN URUN ==> UyolMiktarNominal=A Karar=IADE	0.86
32	BolgeId=8 ==> UyolMiktarNominal=A Karar=IADE	0.86

Otomatik KDS: Tez kapsamında geliştirilen, perakende sektöründe fire miktarına etki eden faktörlerin tespit edilme aşamasında, karar vericiye destek sağlaması planlanan uygulamada VM algoritmalarının otomatik olarak uygulandığı ve yorumlandığı bölümdür. Arayüzünden veri seti yüklendikten ve veri türleri seçildikten sonra *"KDS"* düğmesine (*buton*) basıldığında (Şekil 4.12, Şekil 4.14) başka hiçbir işleme gerek kalmadan kullanıcıya Şekil 4.13 ve Şekil 4.15'teki gibi otomatik rapor sunulmaktadır.

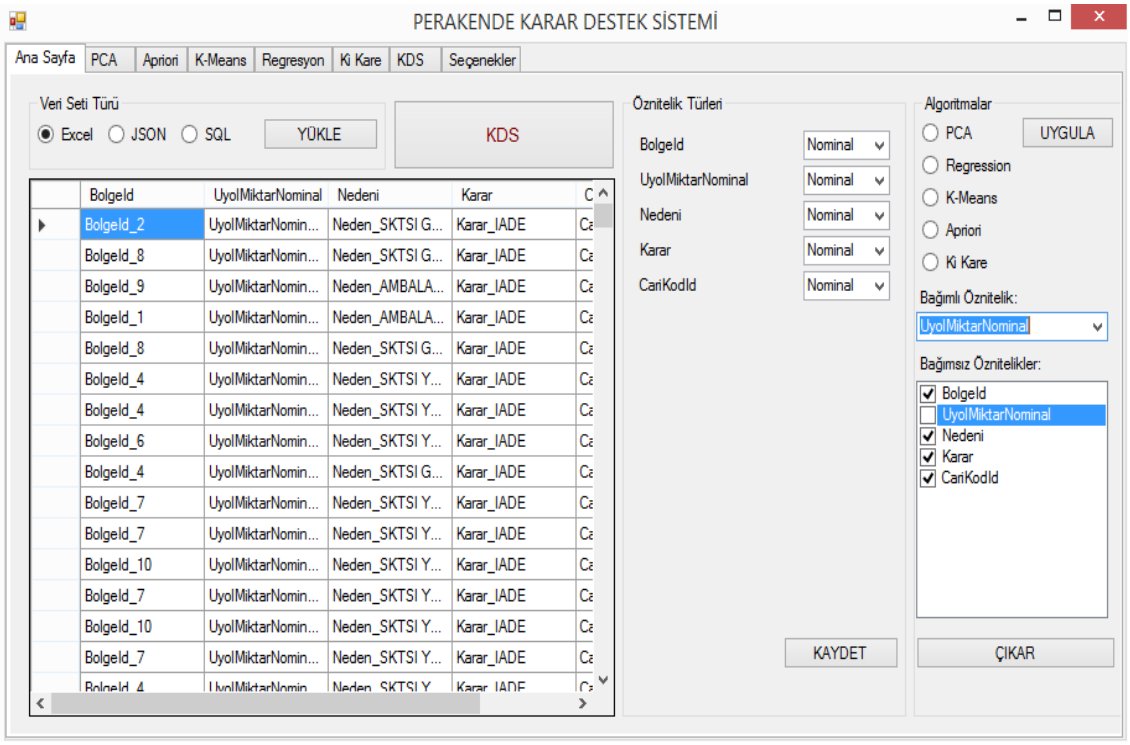
Nominal veri seti için PCA ve regresyon, nümerik veri seti için ki kare ve apriori algoritmaları otomatik olarak uygulanmaktadır. Sonuçların son kullanıcı tarafından anlaşılabilmesi için perakende sektörü fireyi etkileyen faktörler, algoritma sonuçlarına göre yorumlanmaktadır (Şekil 4.13, Şekil 4.15).



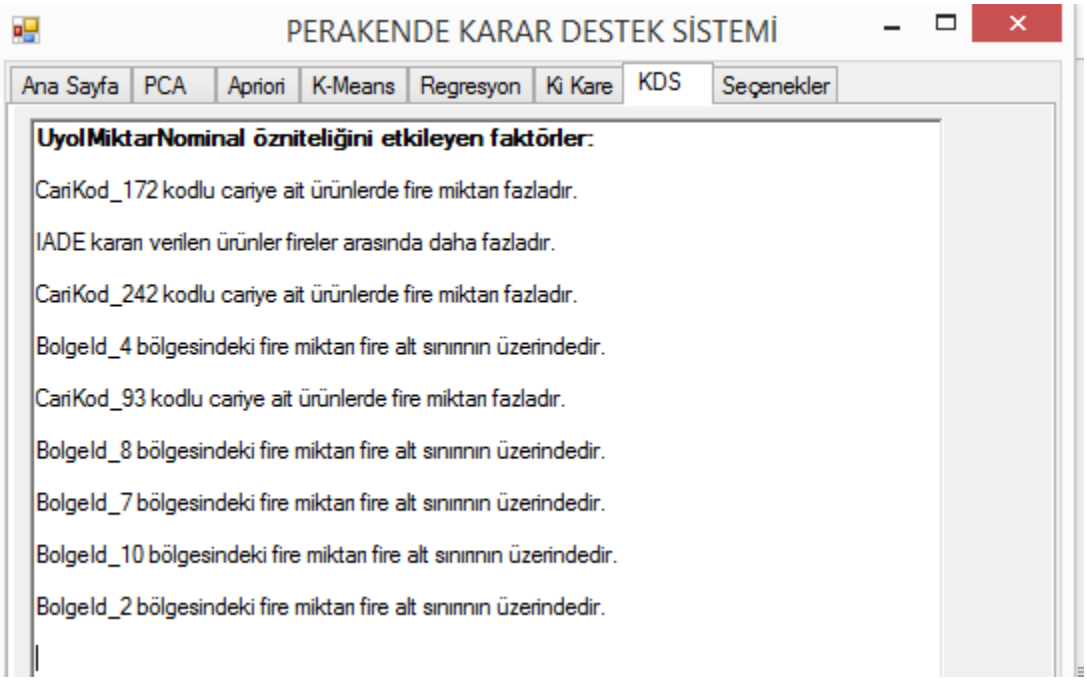
Şekil 4.12. Otomatik KDS nümerik veri seti



Şekil 4.13. Otomatik KDS nümerik sonuçlar



Şekil 4.14. Otomatik KDS nominal veri seti



Şekil 4.15. Otomatik KDS nominal sonuçlar

5. TARTIŞMA ve SONUÇ

Yüksek lisans tezinde Marmara, Ege ve İç Anadolu Bölgelerinde şubeleri bulunan alışveriş merkezine ait market ve depolardaki fire kayıtları incelenmiştir. Fireye sebep olabilecek doğrudan veya dolaylı çeşitli faktörler analiz edilmiştir. Elde edilen sonuçlar şirketin fire azaltıcı önlemler alma sürecinde karar vericiye destek sağlayacaktır.

KDS uygulaması dışarıdan aldığı verileri kullanıcının rahatlıkla uygulayabileceği bir şekilde VM algoritmalarını işleyerek, uygulanan algoritmaların sonucuna göre en doğru yöntemi ve adımı göstermektedir. Geliştirilen uygulamada kullanılacak veri seti farklı alanlar ve sektörlere ait olabilecek esneklikte modellenmiştir. Mevcut sorunun çözümünde perakende sektörü verileri kullanılmıştır ve algoritmalar bu veriler üzerinde denenerek yorumlanmıştır.

Veri setine uygulanan VM yöntemlerinde fireyi etkileyen nedenler incelenerek çeşitli sonuçlar elde edilmiştir. Bazı faktörlerin fireye hiçbir etkisi yok iken bazı faktörlerin doğrudan bazılarının ise dolaylı olarak etkisi gözlemlenmiştir. Bulgular bölümünde analiz edilen sonuçlara genel olarak bakıldığında aslında her firenin personel ya da müşteri kaynaklı olmadığı, yönetim kararları ve tedarikçi firmaların fireler üzerindeki nasıl etkili olduğu görülmektedir. Mağaza ve depolara satış kapasitenin üzerinde ürün doldurmak şubelerde son kullanma tarihini aşan firelere, fazla istifleme ile ambalaj bozukluklarına sebep olmaktadır şeklinde yorumlanabilir. Özellikle bazı tedarikçilerden temin edilen belirli ürünlerdeki sürekli fireler o tedarikçinin ürünler kullandığı ambalaj ile ilişkilendirilebilir. Farklı bölgelerde bulunan mağazalardaki veriler incelendiğinde fire oranlarında farklılıklar gözlemlenmiştir. Bu durum çeşitli şekillerde yorumlanabilir. Bölge müdürlerinin uygulamış olduğu farklı stratejiler ya da bölgenin sosyo-kültürel yapısı hakkında bilgiler verebilir. Bu sonuçların dışında öznitelikler arasındaki gizli kalmış ilişkilerde gözlemlenmiştir. Asgari ücretin artması ile müşteri sepetindeki ürün miktarının artması, müşteri şikâyet sayısının az olduğu mağazalarda cironun diğer mağazalara göre nispeten cironun daha fazla olması gibi farklı sonuçlarda elde edilmiştir.

Geliştirilen uygulama Visual Studio Enterprise 2017 platformunda, C# programlama dili ile kodlanmıştır. VM algoritmaları için Accord.NET framework'ünün matematik, veri madenciliği ve istatistik kütüphaneleri kullanılmıştır.

Yapılan çalışma sonucunda bir veri setini çeşitli VM algoritmaları ile işleyerek kullanıcıya sonuçları raporlayan ve önerilerde bulunan, geliştirilebilir, esnek ve dinamik bir karar destek sistemi uygulaması geliştirilmiştir. Karar verme ve veri analiz süreçlerinde bu öneri ve sonuçların göz önünde bulundurulması tavsiye edilmektedir.

KAYNAKLAR

Agrawal, R. ve Srikant, R. 1994. Fast Algorithms for Mining Association Rules in Large Databases. Proceedings of the 20th International Conference on Very Large Databases (VLDB), Santiago.

Amasyalı, F. M., Ersoy, O. 2008. Kümeleyici Topluluklarının Başarısını Etkileyen Faktörler. IEEE 16th Signal Processing and Communication Applications Conference, Aydın.

Anonim, 2018a. Veri Madenciliğinin Tarihi. https://www.exastax.com.tr/veri-analitigi/veri-madenciliginin-tarihi/-(Erişim tarihi: 20.01.2020).

Anonim, 2018b. Yazılım İsterleri Çözümlemesi. https://github.com/erbilnas/cs-sakaryauniversity-(Erişim tarihi: 25.11.2019).

Anonim, 2018c. Ki Kare Testleri. http://www.ekolar.com/ki-kare-testleri/-(Erişim tarihi: 22.01.2020).

Anonim, 2020. Accord.NET. http://accord-framework.net//-(Erişim tarihi: 22.02.2020).

Arslan, H. 2008. Sakarya Üniversitesi Web Sitesi Erişim Kayıtlarının Web Madenciliği İle Analizi. *Yüksek Lisans Tezi*, SAÜ Fen Bilimleri Enstitüsü, Elektronik Ve Bilgisayar Eğitimi Anabilim Dalı, Sakarya.

Beyer K., Goldstein J., Ramakrishnan R., Shaft U. 1999. When Is 'Nearest Neighbor' Meaningful?. International Conference of Database Theory. 15 January 1999, Madison.

Bircan, H. 2004. Lojistik Regresyon Analizi: Tıp Verileri Üzerinde Bir Uygulama. *Kocaeli Üniversitesi Sosyal Bilimler Enstitüsü Dergisi,* 2: 185–208.

Budak, H. 2018. Özellik Seçimi Yöntemleri ve Yeni Bir Yaklaşım. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, .

Çakmak, Z., Uzgören, N., Keçek, G. 2005. Kümeleme Analizi Teknikleri İle İllerin Kültürel Yapılarına Göre Sınıflandırılması ve Değişimlerinin İncelenmesi. *Dumlupınar Üniversitesi Sosyal Bilimler Dergisi,* 12: 15–36.

Çetinyokuş, T., Gökçen, H. 2002. Borsada teknik göstergelerle analiz için bir Karar Destek Sistemi. *Gazi Üni. Müh. Mim. Fak. Dergisi*, 17(1): 43-58.

Dinçer, E. 2006. Veri Madenciliğinde K-Means Algoritması ve Tıp Alanında Uygulanması. *Yüksek Lisans Tezi*, KOU Fen Bilimleri Enstitüsü, Kocaeli.

Dumhan, M. H. 2003. Data Mining: Introductory and Advanced Topics. Pearson, New Jersey.

Evans, S., Lloyd, J., Stoddard, G., Nekeber, J., Samone, M. 2005. Risk Factors For Adverse Drug Events. *The Annals of Pharmacotherapy,* 39: 1161-1168.

Fayyad, U., Shapiro, G., Smyth, P. 1996. From Data Mining To Knowledge Discovery In Databases. *AI Magazine,* 17(3): 37-54.

Garip, E. 2017. OEDCD Ülkelerindeki CO_2 Emisyonunun Makine Öğrenmesi İle Tahmin Edilmesi. *Yüksek Lisans Tezi,* IMU Fen Bilimleri Enstitüsü, Mühendislik Yönetimi Anabilim Dalı, İstanbul.

Gersho, A., Gray, R.M. 1991. Vector Quantization and Signal Compression. Kluwer Academic Publishers Norwell, USA, 738pp.

Gökçen, H. 2007. Yönetim Bilgi Sistemleri. Palme Yayıncılık, Ankara, 342s.

Guidici, P. 2004. Applied Data Mining: Statistical Methods for Business and Industry,(2nd ed.), John Wiley and Sons Ltd, Hoboken.

Güngör, E., Yalçın N., Yurtay, N. 2013. Apriori Algoritması İle Teknik Seçmeli Ders Seçimi Analizi. UZEM 2013 Ulusal Uzaktan Eğitim ve Teknolojileri Sempozyumu, Konya.
Han J., Kamber M. 2006. Data Mining: Concepts and Techniques (edited by 2nd.). Morgan Kaufmann, USA, 770pp.
Hastie T., Tibshirani R., Friedman J. 2004. The Elements of Statistical Learning: Data Mining, Inference and Prediction (edited by Second.). Sprinter, USA, 745pp.
Kantardzic, M. 2002. Data Mining Concepts, Models, Methods, and Algorithms. Wiley InterScience, IEEE, USA.
Kartın, E. 2008. Güvenli Yazılım Geliştirme, Beykent Üniversitesi Computer Based Business Application CEN 404 Seminer Notları, İstanbul.
Kaya, Y., Yeşilova, A. 2011. İki Durumlu Karışımlı Lojistik Regresyona İlişkin Bir Uygulama. *Bilişim Teknolojileri Dergisi,* 4(3): 53–58.
Koyuncugil, A. S., Özgülbaş N. 2009. Veri Madenciliği: Tıp ve Sağlık Hizmetlerinde Kullanımı ve Uygulamaları. *Bilişim Teknolojileri Dergisi*, 2(2): 21–32.
Mitchell, T. 2009. Machine Learning. Mcgraw-Hill Companies, New York.
Ordukaya, E. 2011. Bulanık Karar Verme Süreçlerinde Geri Bildirim Ve Mikro Öğretim Uygulaması. *Yüksek Lisans Tezi,* MÜ Fen Bilimleri Enstitüsü, Bilgisayar-Kontrol Eğitimi Programı, İstanbul.
Özbilgin, İ.G., ÖZLÜ, M. 2010. Yazılım Geliştirme Süreçleri Ve ISO 27001 Bilgi Güvenliği Yönetim Sistemi. Akademik Bilişim'10 - XII. Akademik Bilişim Konferansı. 10-12 Şubat 2010, Muğla Üniversitesi, Muğla.
Özkan, Y. 2016. Veri Madenciliği Yöntemleri. Papatya Bilim, İstanbul, 240 s.
Peker, M., Özkaraca, O., Kesimal, B. 2017. Enerji Tasarruflu Bina Tasarımı İçin Isıtma ve Soğutma Yüklerini Regresyon Tabanlı Makine Öğrenmesi Algoritmaları İle Modelleme. *Bilişim Teknolojileri Dergisi,* 10(4): 443–449.
Sağın, A.N. 2018. Veri Madenciliği Algoritmaları İle Birliktelik Kurallarının Belirlenmesi: Perakende Sektöründe Bir Uygulama. *Yüksek Lisans Tezi,* İTİCÜ Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, İstanbul.
Silahtaroğlu, G. 2016. Veri Madenciliği: Kavram ve Algoritmaları. Papatya Bilim, İstanbul, 304s.
Şeker, S. E., 2014. Karar Destek Sistemleri. http://mis.sadievrenseker.com/2014/02/karar-destek-sistemleri-kds-decision-support-systems-dss/-(Erişim tarihi: 10.01.2020).
Tatlıdil, H. 1996. Uygulamalı Çok Değişkenli İstatistiksel Analiz. Hacettepe Üniversitesi Fen Fakültesi İstatistik Bölümü, Akademi Yayınları, Ankara.
Tunçalp, E. 2012. Mağaza Kayıpları Kader Değildir. Eylül Yayın Grubu, İstanbul, 239s.
Yaycılı, A.Ö. 2006. Temel Bileşenler Analizi İçin Robust Algoritmaları. *Yüksek Lisans Tezi,* GÜ Fen Bilimleri Enstitüsü, İstatistik Anabilim Dalı, Ankara.
Yazılım, 2020. Türk Dil Kurumu Büyük Sözlük. https://sozluk.gov.tr/-(Erişim tarihi: 20.01.2020).
Yıldız, O., Dağdeviren, M., Çetinyokuş, T. 2008. İşgören Performansının Değerlendirilmesi İçin Bir karar Destek Sistemi Ve Uygulaması. *Gazi Üni. Müh. Mim. Fak. Dergisi,* 23(1): 239-248.

EKLER

EK 1 Karar destek sistemi uygulama kodları

EK1 Karar Destek Sistemi Uygulama Kodları

frmUserInterface.cs

```
using Accord.Controls;
using Accord.IO;
using Accord.Statistics.Analysis;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Perakende_KDS
{
   public partial class frmUI : Form
   {
      public static DataMining dataMining;
      public static CheckedListBox checkListBox;

      public frmUI()
      {
         InitializeComponent();
      }
```

```
private void btnDataSetImport_Click(object sender, EventArgs e)
{
   richTextBoxPcaDependent.Clear();
   richTextBoxPcaIndependent.Clear();
   richTextBoxApriori.Clear();
   richTextBoxRegresyon.Clear();
   richTextBoxKiKare.Clear();
   richTextBoxKDS.Clear();
   panel3.Controls.Clear();

   dataMining = DataMining.getInstance();
   if (radioExcel.Checked == true)
   {
      if (openFileDialog.ShowDialog(this) == DialogResult.OK)
      {
         string filename = openFileDialog.FileName;
         string extension = Path.GetExtension(filename);
         if (extension == ".xls")
         {
            ExcelReader db = new ExcelReader(filename, true, false);
            dataGridDataSet.DataSource = db.GetWorksheet(0);
            FillData(dataGridDataSet);
         }
      }
   }
   if (radioJSON.Checked == true)
   {
      if (openFileDialog.ShowDialog(this) == DialogResult.OK)
      {
         string filename = openFileDialog.FileName;
         string extension = Path.GetExtension(filename);
         if (extension == ".json")
```

```
            {
                StreamReader r = new StreamReader(filename);
                string json = r.ReadToEnd();
                dataGridDataSet.DataSource = JsonStringToDataTable(json);
                FillData(dataGridDataSet);
            }
        }
    }
    if (radioSQL.Checked == true)
    {
        SqlConnection con = new SqlConnection("Data Source=ASUS; Initial Catalog=TEZ;
Integrated Security=true");
        SqlCommand cmd = new SqlCommand("sp_numTable", con);
        cmd.CommandType = CommandType.StoredProcedure;

        SqlDataAdapter dr = new SqlDataAdapter(cmd);
        DataSet ds = new DataSet();
        dr.Fill(ds);
        dataGridDataSet.DataSource = ds.Tables[0];
        FillData(dataGridDataSet);
    }
}

private void btnDataTypeSave_Click(object sender, EventArgs e)
{
    List<FeatureType> dataType = new List<FeatureType>();

    foreach (Control p in panel1.Controls)
    {
        if (p is ComboBox)
        {
            FeatureType x = (FeatureType)Enum.Parse(typeof(FeatureType), p.Text, true);
            dataType.Add(x);
```

```
            }
        }

        List<string> selectedColumnNAme = new List<string>();
        richTextBoxRegresyon.Clear();
        foreach (string name in checkListBox.CheckedItems)
            selectedColumnNAme.Add(name);

        dataMining.setDataset(dataGridDataSet, dataType, selectedColumnNAme,
comboAttribute.Text);

    }

    private void btnApply_Click(object sender, EventArgs e)
    {
        if (radioPCA.Checked == true)
        {
            richTextBoxPcaDependent.Clear();
            richTextBoxPcaIndependent.Clear();

            if (comboAttribute.SelectedIndex != -1)
            {
                PcaReturnStruct pca = dataMining.PCA(comboAttribute.SelectedIndex);

                string[] columnName = dataMining.dmDataSet.columnNames.Where(x => x !=
null).Select(x => x.ToString()).ToArray();
                dgvStatisticCorrelation.DataSource = new ArrayDataView(pca.corelasyonMatrix,
columnName);
                for (int i = 0; i < pca.dependentVariable.Count; i++)
                {
                    richTextBoxPcaDependent.AppendText(pca.dependentVariable[i] + "\n");

richTextBoxPcaDependent.AppendText(pca.dependentVariableValue[i].ToString() + "\n");
```

```
            richTextBoxPcaDependent.AppendText("\n");
        }
        for (int i = 0; i < pca.independentVariable.Count; i++)
        {
            richTextBoxPcaIndependent.AppendText(pca.independentVariable[i] + "\n");
richTextBoxPcaIndependent.AppendText(pca.independentVariableValue[i].ToString() + "\n");
            richTextBoxPcaIndependent.AppendText("\n");
        }

        /****** Dynamik Label ******/
        int location_x = 0 + 1;
        int location_y = 0 + 25;
        Label[] labels = new Label[dataMining.dmDataSet.columnNames.Count];
        int j = 0;

        foreach (string col in columnName)
        {
            labels[j] = new Label();
            labels[j].Text = col;
            labels[j].Location = new Point(location_x, location_y);
            labels[j].Size = new Size(150, 20);
            labels[j].Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Regular,
GraphicsUnit.Point, ((byte)(0)));
            labels[j].TextAlign = ContentAlignment.TopRight;

            panel2.Controls.Add(labels[j]);
            location_y += 22;
            j++;
        }
        this.tabControl.SelectedTab = tabPagePCA;
        /************************/
    }
```

```
            else
            {
                MessageBox.Show("Lütfen bağımlı özniteliği seçiniz!");
            }
        }

        if (radioRegression.Checked == true)
        {
            richTextBoxRegresyon.Clear();

            richTextBoxRegresyon.Clear();
            MultipleLinearRegressionAnalysis reg =
dataMining.Regression((string)comboAttribute.SelectedItem,
dataMining.dmDataSet.inDependentFeatureName);
            dgvLinearCoefficients.DataSource = reg.Coefficients;

            double sumCoef = 0;
            for (int i = 0; i < reg.Coefficients.Count - 1; i++)
            {
                sumCoef += Math.Abs(reg.Coefficients[i].Value);

            }

            for (int i = 0; i < reg.Coefficients.Count - 1; i++)
            {
                double prop = Math.Abs(reg.Coefficients[i].Value * 100 / sumCoef);
                richTextBoxRegresyon.AppendText("%" + Math.Round(prop, 1) + " - " +
reg.Coefficients[i].Name + "\n");
            }
            this.tabControl.SelectedTab = tabPageRegresyon;
        }

        if (radioKMeans.Checked == true)
```

```
        {
            panel3.Controls.Clear();

            KMeansReturnStruct kmeans = dataMining.KMeans();
            /**********Dynamic label + richText**********/
            int location_x = 0 + 10;
            int location_y = 0 + 45;
            RichTextBox[] richText = new RichTextBox[kmeans.clusterIndex.Count];
            Label[] labels = new Label[kmeans.clusterIndex.Count];
            Label labels2 = new Label();

            labels2.Text += ("Kayıt sırası / ");
            for (int i = 0; i < dataMining.dmDataSet.columnNames.Count; i++)
            {
                labels2.Text += (dataMining.dmDataSet.columnNames[i] + " / ");
            }
            labels2.Location = new Point(location_x, location_y - 40);
            labels2.Size = new Size(1000, 20);
            labels2.Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Regular,
GraphicsUnit.Point, ((byte)(0)));
            panel3.Controls.Add(labels2);



            for (int j = 0; j < kmeans.clusterIndex.Count; j++)
            {
                labels[j] = new Label();
                labels[j].Text = (kmeans.clusterIndex[j] + 1).ToString() + ". küme elemanları - " +
"adet";
                labels[j].Location = new Point(location_x, location_y - 20);
                labels[j].Size = new Size(190, 20);
                labels[j].Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Regular,
GraphicsUnit.Point, ((byte)(0)));
                panel3.Controls.Add(labels[j]);
```

```
            richText[j] = new RichTextBox();
            richText[j].Location = new Point(location_x, location_y);
            richText[j].Size = new Size(300, 500);
            richText[j].Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Regular,
GraphicsUnit.Point, ((byte)(0)));
            richText[j].Name = "richTextKmeans_" + j;
            panel3.Controls.Add(richText[j]);

            location_x += 310;
        }

        /*****************************************/
        for (int j = 0; j < kmeans.clusterIndex.Count; j++)
        {
            int count = 0;
            RichTextBox rtb = panel3.Controls.OfType<RichTextBox>().FirstOrDefault(x =>
x.Name == "richTextKmeans_" + j);
            if (rtb != null)
            {
                rtb.AppendText("Kayıt Sırası - Öznitelikler");
                rtb.AppendText("\n");

                for (int i = 0; i < kmeans.idx.Length; i++)
                {
                    if (kmeans.idx[i] == kmeans.clusterIndex[j])
                    {
                        rtb.AppendText("     " + i + "       -       ");
                        for (int k = 0; k < kmeans.inputs[0].Length; k++)
                            rtb.AppendText(kmeans.inputs[i][k].ToString() + "   ,   ");
                        rtb.AppendText("\n");
                        count++;

                    }
```

```
                }
                rtb.AppendText("\n");
            }

            labels[j].Text = (kmeans.clusterIndex[j] + 1).ToString() + ". küme elemanları - " +
count + " adet";
        }
        this.tabControl.SelectedTab = tabPageKMeans;
    }

    if (radioApriori.Checked == true)
    {
        richTextBoxApriori.Clear();

        AprioriReturnStruct apriori = dataMining.Apriori();
        richTextBoxApriori.AppendText("Güven Kat. / Destek Kat. / İlişki\n\n");
        for (int i = 0; i < apriori.associatedVariable.Count; i++)
        {
            richTextBoxApriori.AppendText("      " + Math.Round(apriori.confidence[i], 2) *
100 + "%    -  ");
            richTextBoxApriori.AppendText("      " + apriori.support[i].ToString() + "    ->  ");
            richTextBoxApriori.AppendText(apriori.associatedVariable[i].ToString() + "\n");
            richTextBoxApriori.AppendText("\n");
        }
        this.tabControl.SelectedTab = tabPageApriori;
    }

    if (radioKiKare.Checked == true)
    {
        richTextBoxKiKare.Clear();
        //dataMining.kiKare.alfa =  2.03426859549273E-100;
        List<List<string>> kiKare = dataMining.KiKare();
        for (int i = 0; i < kiKare.Count; i++)
```

```
            {
                richTextBoxKiKare.AppendText("Önem durumu :" + kiKare[i][1] + "    " +
kiKare[i][0] /*+ "    -    " + x2.PValue.ToString()*/  + "\n");
            }

            this.tabControl.SelectedTab = tabPageKiKare;
        }

    }

    private void btnRemove_Click(object sender, EventArgs e)
    {
        foreach (string name in checkListBox.CheckedItems)
        {
            dataGridDataSet.Columns.Remove(name);
            (dataGridDataSet.DataSource as DataTable).Columns.Remove(name);
        }
        FillData(dataGridDataSet);
    }


    private void btnKDS_Click(object sender, EventArgs e)
    {
        richTextBoxKDS.Clear();
        this.tabControl.SelectedTab = tabPageKDS;
        List<string> feature = new List<string>();

        if (comboAttribute.SelectedIndex != -1)
        {
            if (dataMining.dmDataSet.allFeatures[0].featureType == FeatureType.Numeric)
            {
                richTextBoxKDS.SelectionFont = new Font(this.richTextBoxKDS.Font,
FontStyle.Bold);
```

```
            richTextBoxKDS.AppendText(dataMining.dmDataSet.dependentFeatureName + "
özniteliğini etkileyen faktörler: \n\n");


            richTextBoxKDS.SelectionFont = new Font(this.richTextBoxKDS.Font,
FontStyle.Regular);


            /***** pca *****/
            PcaReturnStruct pca = dataMining.PCA(comboAttribute.SelectedIndex);
            for (int i = 0; i < pca.dependentVariable.Count; i++)
            {
               for (int k = 0; k < dataMining.dmDataSet.dataColumns.Count; k++)
               {
                  if (dataMining.dmDataSet.dataColumns[k].feature.featureName ==
pca.dependentVariable[i].Split('-')[1].Trim())
                  {
                     feature.Add(pca.dependentVariable[i].Split('-')[1].Trim());
                     richTextBoxKDS.AppendText(pca.dependentVariable[i].Split('-')[1].Trim()
+ dataMining.dmDataSet.dataColumns[k].feature.command1 + "\n\n");
                  }
               }
            }


            List<string> independentName = new List<string>();
            richTextBoxRegresyon.Clear();
            foreach (string name in checkListBox.CheckedItems)
               independentName.Add(name);


            /***** regresyon *****/
            MultipleLinearRegressionAnalysis reg =
dataMining.Regression((string)comboAttribute.SelectedItem, independentName);
            dgvLinearCoefficients.DataSource = reg.Coefficients;


            double sumCoef = 0;
            for (int i = 0; i < reg.Coefficients.Count - 1; i++)
```

```
            {
                sumCoef += Math.Abs(reg.Coefficients[i].Value);


            }
            for (int i = 0; i < reg.Coefficients.Count - 1; i++)
            {
                double prop = Math.Abs(reg.Coefficients[i].Value * 100 / sumCoef);


                if (prop > 3)
                {
                    if (feature.IndexOf(reg.Coefficients[i].Name) == -1)
                    {


                        for (int k = 0; k < dataMining.dmDataSet.dataColumns.Count; k++)
                        {


                            if (dataMining.dmDataSet.dataColumns[k].feature.featureName ==
reg.Coefficients[i].Name)
                            {
                                feature.Add(reg.Coefficients[i].Name);
                                richTextBoxKDS.AppendText(reg.Coefficients[i].Name +
dataMining.dmDataSet.dataColumns[k].feature.command1 + "\n\n");
                            }
                        }
                    }
                }
            }




            if (dataMining.dmDataSet.allFeatures[0].featureType == FeatureType.Nominal)
            {
```

```
            /*** Ki Kare***/
            // dataMining.kiKare.alfa=0,1;
            richTextBoxKDS.SelectionFont = new Font(this.richTextBoxKDS.Font,
FontStyle.Bold);
            richTextBoxKDS.AppendText(dataMining.dmDataSet.dependentFeatureName + "
özniteliğini etkileyen faktörler:\n\n");
            richTextBoxKDS.SelectionFont = new Font(this.richTextBoxKDS.Font,
FontStyle.Regular);



            List<List<string>> kiKare = dataMining.KiKare();
            for (int i = 0; i < kiKare.Count; i++)
            {
               if (kiKare[i][1] == "True")
               {
                  if (feature.IndexOf(kiKare[i][0]) == -1)
                  {

                     for (int k = 0; k < dataMining.dmDataSet.dataColumns.Count; k++)
                     {
                        if (dataMining.dmDataSet.dataColumns[k].feature.featureName ==
kiKare[i][0].ToString())
                        {
                           feature.Add(kiKare[i][0]);

richTextBoxKDS.AppendText(kiKare[i][0].Replace(dataMining.dmDataSet.dataColumns[k].fe
ature.retailType + "_", "") + dataMining.dmDataSet.dataColumns[k].feature.command1 +
"\n\n");
                        }
                     }
                  }
               }
            }
            /*** Apriori***/
            AprioriReturnStruct apriori = dataMining.Apriori();
```

```
            for (int i = 0; i < apriori.associatedVariable.Count; i++)
            {
                if
(apriori.associatedVariable[i].Contains(dataMining.dmDataSet.dependentFeatureName))
                {
                    string str0 = apriori.associatedVariable[i].Split('>')[0].Trim().Trim('-
').Trim().Trim('[').Trim(']');
                    string str1 = apriori.associatedVariable[i].Split('>')[1].Trim().Trim('-
').Trim().Trim('[').Trim(']');

                    string[] oznitelikler0 = str0.Split(' '); //2 veya daha fazla faktör var ise
                    string[] oznitelikler1 = str1.Split(' '); //2 veya daha fazla faktör var ise

                    if (str1.Contains(dataMining.dmDataSet.dependentFeatureName))  //bağımlı
öznitelik etkileniyor ise  -> UyolMiktar
                    {
                        for (int j = 0; j < oznitelikler0.Length; j++)
                        {
                            if (feature.Contains(oznitelikler0[j]) == false)
                            {
                                for (int k = 0; k < dataMining.dmDataSet.dataColumns.Count; k++)
                                {
                                    if
(Convert.ToDouble(dataMining.dmDataSet.dataColumns[k].datas.IndexOf(oznitelikler0[j])) !=
(-1))
                                    {
                                        feature.Add(oznitelikler0[j]);

richTextBoxKDS.AppendText(oznitelikler0[j].Replace(dataMining.dmDataSet.dataColumns[k]
.feature.retailType + "_", "") + dataMining.dmDataSet.dataColumns[k].feature.command2 +
"\n\n");
                                    }

                                }
                            }
                        }
```

```
                }
            }
        }
    }
    this.tabControl.SelectedTab = tabPageKDS;
  }
  else
  {
    MessageBox.Show("Lütfen bağımlı özniteliği seçiniz!");
  }
}

private void tabControl_Click(object sender, EventArgs e)
{
  if (dataMining != null)
  {
    /* PCA */
    txtPCADependentCoef.Text = dataMining.pca.dependentCoefficient.ToString();
    txtPCAIndependentCoef.Text = dataMining.pca.independentCoefficient.ToString();

    /* Apriori */
    txtAprioriConfCoef.Text = dataMining.apriori.confidenceCoef.ToString();
    txtAprioriSuppCoef.Text = dataMining.apriori.suppCoef.ToString();


    /* K-Means */
    txtKMeansKCount.Text = dataMining.kmeans.k.ToString();
    txtKMeansTolerance.Text = dataMining.kmeans.tolerance.ToString();

    /* Ki Kare */
    txtKiKareAlfa.Text = dataMining.kiKare.alfa.ToString();
  }
```

```
    }

    private void btnSettingSave_Click(object sender, EventArgs e)
    {
        if (dataMining != null)
        {
            dataMining.pca.dependentCoefficient =
Convert.ToDouble(txtPCADependentCoef.Text);
            dataMining.pca.independentCoefficient =
Convert.ToDouble(txtPCAIndependentCoef.Text);

            /* Apriori */
            dataMining.apriori.confidenceCoef = Convert.ToDouble(txtAprioriConfCoef.Text);
            dataMining.apriori.suppCoef = Convert.ToDouble(txtAprioriSuppCoef.Text);

            /* K-Means */
            dataMining.kmeans.k = Convert.ToInt32(txtKMeansKCount.Text);
            dataMining.kmeans.tolerance = Convert.ToDouble(txtKMeansTolerance.Text);

            /* Ki Kare */
            dataMining.kiKare.alfa = Convert.ToDouble(txtKiKareAlfa.Text);
        }
    }

    public void FillData(DataGridView dataGridView)
    {
        string dependent_Name;
        string[] columnNames;

        DataTable sourceTable;
        int location_x = 0 + 5;
        int location_y = 0 + 10;
```

```
this.comboAttribute.Items.Clear();
this.checkedListAttribute.Items.Clear();
this.panel1.Controls.Clear();

/**************************************/
sourceTable = (dataGridView.DataSource as DataTable);
columnNames = new string[sourceTable.Columns.Count];

for (int j = 0; j < sourceTable.Rows.Count; j++)
{
   for (int k = 0; k < sourceTable.Columns.Count; k++)
   {
      columnNames[k] = sourceTable.Columns[k].ColumnName.Trim();
   }
}

/**************************************/
ComboBox[] comboBoxes = new ComboBox[columnNames.Length];
Label[] labels = new Label[columnNames.Length];
int i = 0;
foreach (string col in columnNames)
{
   this.comboAttribute.Items.Add(col);
   this.checkedListAttribute.Items.Add(col);

   labels[i] = new Label();
   labels[i].Text = col;
   labels[i].Location = new Point(location_x, location_y);
   labels[i].Size = new Size(135, 20);
   labels[i].Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Regular,
GraphicsUnit.Point, ((byte)(0)));

   comboBoxes[i] = new ComboBox();
```

```
            foreach (var item in Enum.GetValues(typeof(FeatureType)))
            {
               comboBoxes[i].Items.Add(item);
            }

            comboBoxes[i].Name = "comboBox" + col;
            comboBoxes[i].Location = new Point(location_x + 135, location_y - 3);
            comboBoxes[i].Size = new Size(70, 20);
            comboBoxes[i].Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Regular,
GraphicsUnit.Point, ((byte)(0)));
            comboBoxes[i].SelectedIndex = 0;
            panel1.Controls.Add(labels[i]);
            panel1.Controls.Add(comboBoxes[i]);

            location_y += 25;
            i++;
         }

         this.comboAttribute.SelectedIndex = 0;
         dependent_Name = (string)comboAttribute.SelectedItem;
         checkListBox = checkedListAttribute;
      }

      public DataTable JsonStringToDataTable(string jsonString)
      {
         DataTable dt = new DataTable();
         string[] jsonStringArray = Regex.Split(jsonString.Replace("[", "").Replace("]", ""),
"},");
         List<string> ColumnsName = new List<string>();
         foreach (string jSA in jsonStringArray)
         {
            string[] jsonStringData = Regex.Split(jSA.Replace("{", "").Replace("}", ""), ",");
            foreach (string ColumnsNameData in jsonStringData)
```

```
            {
                try
                {
                    int idx = ColumnsNameData.IndexOf(":");
                    string ColumnsNameString = ColumnsNameData.Substring(0, idx -
1).Replace("\"", "");
                    if (!ColumnsName.Contains(ColumnsNameString))
                    {
                        ColumnsName.Add(ColumnsNameString);
                    }
                }
                catch (Exception ex)
                {
                    throw new Exception(string.Format("Error Parsing Column Name : {0}",
ColumnsNameData));
                }
            }
            break;
        }
        foreach (string AddColumnName in ColumnsName)
        {
            dt.Columns.Add(AddColumnName);
        }
        foreach (string jSA in jsonStringArray)
        {
            string[] RowData = Regex.Split(jSA.Replace("{", "").Replace("}", ""), ",");
            DataRow nr = dt.NewRow();
            foreach (string rowData in RowData)
            {
                try
                {
                    int idx = rowData.IndexOf(":");
                    string RowColumns = rowData.Substring(0, idx - 1).Replace("\"", "");
```

```
                string RowDataString = rowData.Substring(idx + 1).Replace("\"", "");
                nr[RowColumns] = RowDataString;
            }
            catch (Exception ex)
            {
                continue;
            }
        }
        dt.Rows.Add(nr);
    }
    return dt;
  }
 }
}
```

DataMining.cs

```
using Accord.Statistics.Analysis;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;

namespace Perakende_KDS
{
   public class DataMining
   {
      private static DataMining instance = null;
      public DM_DataSet dmDataSet;
      public PCA pca;
      public Apriori apriori;
      private Regression regression;
      public K_Means kmeans;
      public KiKare kiKare;


      public DataMining()
      {
         dmDataSet = new DM_DataSet();
         pca = new PCA();
         apriori = new Apriori();
         regression = new Regression();
         kmeans = new K_Means();
         kiKare = new KiKare();
      }

      public static DataMining getInstance()
```

```
    {
        if (instance == null)
        {
            instance = new DataMining();
        }
        return instance;
    }

    public void setDataset(DataGridView dgv, List<FeatureType> featureType, List<string>
selectedColumnName, string dependentAttribute)
    {
        dmDataSet.setData(dgv, featureType, selectedColumnName, dependentAttribute);
    }

    public PcaReturnStruct PCA(int labelIndex)
    {
        if (dmDataSet.dataColumns[0].feature.featureType == FeatureType.Numeric)
        {
            double[][] allData_double = dmDataSet.allData.Select(line => line.Select(item =>
Convert.ToDouble(item)).ToArray()).ToArray();
            pca.setData(allData_double, dmDataSet.columnNames);
            return pca.PCA_Algortihm(labelIndex);
        }
        else
        {
            MessageBox.Show("Algoritma Bu Veri Setine Uygulanamaz !!!");
            return new PcaReturnStruct();
        }
    }

    public AprioriReturnStruct Apriori()
    {
        if (dmDataSet.dataColumns[0].feature.featureType == FeatureType.Nominal)
```

```
        {
            string[][] allData_double = dmDataSet.allData.Select(line => line.Select(item =>
item.ToString()).ToArray()).ToArray();
            apriori.setData(allData_double);
            return apriori.Apriori_Algortihm();
        }
        else
        {
            MessageBox.Show("Algoritma Bu Veri Setine Uygulanamaz !!!");
            return new AprioriReturnStruct();
        }
    }

    public MultipleLinearRegressionAnalysis Regression(string _dependent_Name,
List<string> _independent_Names)
    {
        if (dmDataSet.dataColumns[0].feature.featureType == FeatureType.Numeric)
        {
            double[][] inputs_double = dmDataSet.inputsData.Select(line => line.Select(item
=> Convert.ToDouble(item)).ToArray()).ToArray();
            regression.setData(dmDataSet.allDataTable, inputs_double);
            return regression.Regression_Algortihm(_dependent_Name,
_independent_Names);
        }
        else
        {
            MessageBox.Show("Algoritma Bu Veri Setine Uygulanamaz !!!");
            return new MultipleLinearRegressionAnalysis();
        }
    }

    public KMeansReturnStruct KMeans()
    {
        if (dmDataSet.dataColumns[0].feature.featureType == FeatureType.Numeric)
```

```
        {
            double[][] allData_double = dmDataSet.allData.Select(line => line.Select(item =>
Convert.ToDouble(item)).ToArray()).ToArray();
            kmeans.setData(allData_double);
            return kmeans.K_Means_Algortihm();
        }
        else
        {
            MessageBox.Show("Algoritma bu verisetine uygulanamaz");
            return new KMeansReturnStruct();
        }
    }

    public List<List<string>> KiKare()
    {
        if (dmDataSet.dataColumns[0].feature.featureType == FeatureType.Nominal)
        {
            kiKare.setData(dmDataSet);
            return kiKare.KiKare_Algortihm();
        }
        else
        {
            MessageBox.Show("Algoritma Bu Veri Setine Uygulanamaz !!!");
            return new List<List<string>>();
        }
    }
}

public class DM_DataSet
{
    public List<Feature> allFeatures = new List<Feature>();
    public DataTable allDataTable { get; set; }
    public DataTable firstAllData { get; set; }
```

```
    public List<string> columnNames = new List<string>();
    public List<DataColumn> dataColumns { get; set; }
    public string dependentFeatureName;
    public List<string> inDependentFeatureName = new List<string>();
    public object[][] inputsData;
    public object[][] allData;



    public void setData(DataGridView _dgv, List<FeatureType> _featureTypes, List<string>
_inDependentFeatureName, string _dependentFeatureName)
    {
        dependentFeatureName = _dependentFeatureName;


        allDataTable = (_dgv.DataSource as DataTable);
        if (firstAllData == null)
        {
            firstAllData = new DataTable();
            firstAllData.Merge(_dgv.DataSource as DataTable);
        }


        for (int i = 0; i < _inDependentFeatureName.Count; i++)
        {
            inDependentFeatureName.Add(_inDependentFeatureName[i]);
        }


        /*column name*/
        for (int k = 0; k < allDataTable.Columns.Count; k++)
        {
            columnNames.Add(allDataTable.Columns[k].ColumnName);
        }


        /*all feature*/
        for (int i = 0; i < _featureTypes.Count; i++)
```

```
{
    Feature ftr = new Feature(_featureTypes[i], i, columnNames[i]);
    this.allFeatures.Add(ftr);

    if (_featureTypes[i] == FeatureType.Numeric || _featureTypes[i] ==
FeatureType.Binary)
    {
      if (i == 0)
      {
        ftr.retailType = "UyolMiktar";
        ftr.command1 = "";
        ftr.command2 = " ";
      }
      if (i == 1)
      {
        ftr.retailType = "UyolToplamFiyat";
        ftr.command1 = " faktörü fireye etkisi sebebi ile incelenmelidir.";
        ftr.command2 = "";
      }
      if (i == 2)
      {
        ftr.retailType = "MusteriSikayetSayisi";
        ftr.command1 = " file fire miktarı arasında ilişki mevcuttur.";
        ftr.command2 = "";
      }
      if (i == 3)
      {
        ftr.retailType = "SubePersonelOrtalamaUcret";
        ftr.command1 = " faktörü fire üzerinde etkilidir.";
        ftr.command2 = "";
      }
      if (i == 4)
      {
```

```
    ftr.retailType = "AsgariUcret";
    ftr.command1 = " ile fire arasında ilişki vardır.";
    ftr.command2 = "";
}
if (i == 5)
{
    ftr.retailType = "ResmiTatilGunSayisi";
    ftr.command1 = " faktörü fire üzerinde etkilidir.";
    ftr.command2 = "";
}
if (i == 6)
{
    ftr.retailType = "PersonelSayisi";
    ftr.command1 = " ile fire miktarı bağlantılıdır.";
    ftr.command2 = "";
}
if (i == 7)
{
    ftr.retailType = "ToplamSatilanUrunMiktari";
    ftr.command1 = " fire miktarında etkilidir.";
    ftr.command2 = "";
}
if (i == 8)
{
    ftr.retailType = "SubeAylikSepetSayisi";
    ftr.command1 = " fire üzerinde etkilidir.";
    ftr.command2 = "";
}
if (i == 9)
{
    ftr.retailType = "Ciro";
    ftr.command1 = " faktörü fire üzerinde etkilidir.";
```

```
                ftr.command2 = "";
            }
            if (i == 10)
            {
                ftr.retailType = "BirimFiyat";
                ftr.command1 = " ile fire arasında ilişki vardır.";
                ftr.command2 = "";
            }

        }
        if (_featureTypes[i] == FeatureType.Nominal || _featureTypes[i] ==
FeatureType.Categorical)
        {
            if (this.allDataTable.Columns[i].ColumnName.Contains("DepoNo"))
            {
                ftr.retailType = "DepoNo";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " nolu depo fire miktarı diğer depolara göre daha fazladır.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("BolgeId"))
            {
                ftr.retailType = "BolgeId";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır.Bu kategoride
kırılımlar incelenmelidir.";
                ftr.command2 = " kodlu bölgedeki fire miktarı fire alt sınırının üzerindedir.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("Bolge"))
            {
                ftr.retailType = "Bolge";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
```

```
                ftr.command2 = " bölgesindeki fire miktarı fire alt sınırının üzerindedir.";


            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("Yıl"))
            {
                ftr.retailType = "Yıl";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " yılında fire miktarı daha fazladır";


            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("Mevsim"))
            {
                ftr.retailType = "Mevsim";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " mevsiminde fire miktarları artmıştır.";


            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("Ay"))
            {
                ftr.retailType = "Ay";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır.Bu kategoride
kırılımlar incelenmelidir.";
                ftr.command2 = " ayında fire miktarları artmıştır.";


            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("StokKodu"))
            {
                ftr.retailType = "STK";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " stok kodlu ürünün fire miktarı üzerinde etiklidir.";
```

```
            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("KKod"))
            {
                ftr.retailType = "KategoriKodu";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " kategori kodunda fire miktarı daha fazladır.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("AnaGrupKod"))
            {
                ftr.retailType = "AnaGrupKod";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " ana grup kodlu ürünlerin fire miktarları fazladır.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("SubeMudur"))
            {
                ftr.retailType = "SubeMuduruId";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " kodlu şube müdürüne iat şubelerde fire oranı daha fazladır.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("UyolMiktarNominal"))
            {
                ftr.retailType = "UyolMiktarNominal";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " fire miktarı ";

            }
```

```
            if
(this.allDataTable.Columns[i].ColumnName.Contains("UyolToplamFiyatNominal"))
            {
               ftr.retailType = "UyolToplamFiyatNominal";
               ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
               ftr.command2 = " toplam fire fiyatı ile fire miktarı ilişkilidir.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("Neden"))
            {
               ftr.retailType = "Nedeni";
               ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
               ftr.command2 = " nedenine ait fire miktar oranı daha fazladır.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("Karar"))
            {
               ftr.retailType = "Karar";
               ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
               ftr.command2 = " kararı verilen ürünler fireler arasında daha fazladır.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("Yön"))
            {
               ftr.retailType = "Yönlendirme";
               ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
               ftr.command2 = " yönlendirme adımı fireler üzerinde daha fazla kullanılmıştır.";

            }
            if (this.allDataTable.Columns[i].ColumnName.Contains("CariKod"))
```

```
            {
                ftr.retailType = "CariKodId";
                ftr.command1 = " faktörünün fire miktarı üzerindeki etkisi fazladır. Bu
kategoride kırılımlar incelenmelidir.";
                ftr.command2 = " kodlu cariye ait ürünlerde fire miktarı fazladır.";
            }
        }
    }
    /*input*/
    inputsData = new object[allDataTable.Rows.Count][];
    allData = new object[allDataTable.Rows.Count][];
    for (int j = 0; j < allDataTable.Rows.Count; j++)
    {
        object[] row_input = new object[inDependentFeatureName.Count];
        object[] row_all = new object[allFeatures.Count];
        int count = 0;
        for (int k = 0; k < allDataTable.Columns.Count; k++)
        {
            if (_featureTypes[k] == FeatureType.Numeric || _featureTypes[k] ==
FeatureType.Binary)
            {
                if ((_dgv.DataSource as DataTable).Rows[j].ItemArray[k] is double == false)
                {
                    MessageBox.Show("Hatalı Öznitelik Tür Seçimi !!!");
                    return;
                }
            }
            row_all[k] = (object)allDataTable.Rows[j].ItemArray[k].ToString();
            if (allDataTable.Columns[k].ColumnName != dependentFeatureName)
            {
                row_input[count] = (object)allDataTable.Rows[j].ItemArray[k].ToString();
                count++;
            }
```

```
            }
            if (row_input[0] != null)
                inputsData[j] = row_input;
            if (row_all[0] != null)
                allData[j] = row_all;
        }

        /*data columns*/
        this.dataColumns = new List<DataColumn>();
        for (int j = 0; j < allDataTable.Columns.Count; j++)
        {
            List<string> _datas = new List<string>();
            for (int i = 0; i < allDataTable.Rows.Count; i++)
            {
                _datas.Add(allDataTable.Rows[i].ItemArray[j].ToString());
            }
            this.dataColumns.Add(new DataColumn(_datas, this.allFeatures[j]));
        }
    }
}

public class DataColumn
{
    public Feature feature { get; set; }
    public List<string> datas { get; set; }
    public DataColumn(List<string> _datas, Feature _feature)
    {
        feature = _feature;
        datas = _datas;
    }
}
```

```
    public class Feature
    {
        public FeatureType featureType { get; set; }
        public int featureIndex { get; set; }
        public string featureName { get; set; }
        public string retailType { get; set; }
        public string command1 { get; set; }
        public string command2 { get; set; }

        public Feature(FeatureType _featureType, int _featureIndex, string _featureName)
        {
            featureType = _featureType;
            featureIndex = _featureIndex;
            featureName = _featureName;
        }
    }

    public enum FeatureType
    {
        Nominal,
        Numeric,
        Categorical,
        Binary,
        Date
    };
}
```

DataMiningAlgorithm.cs

```
using Accord.MachineLearning;
using Accord.MachineLearning.Rules;
using Accord.Math;
using Accord.Math.Distances;
using Accord.Statistics.Analysis;
using Accord.Statistics.Filters;
using Accord.Statistics.Models.Regression;
using Accord.Statistics.Models.Regression.Linear;
using Accord.Statistics.Testing;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;


namespace Perakende_KDS
{
    public struct PcaReturnStruct
    {
        public double[,] corelasyonMatrix;
        public List<string> dependentVariable;
        public List<double> dependentVariableValue;
        public List<string> independentVariable;
        public List<double> independentVariableValue;
    }

    public struct AprioriReturnStruct
    {
        public List<double> confidence;
        public List<double> support;
        public List<string> associatedVariable;
    }

    public struct KMeansReturnStruct
    {
        public List<int> clusterIndex;
        public int[] idx;
        public double[][] inputs;
    }


    public class PCA
    {
        private PrincipalComponentAnalysis pca;
        private DescriptiveAnalysis sda;
        private string[] columnNames;
        private double[][] inputs;
        public double independentCoefficient { get; set; } = 0.1;
        public double dependentCoefficient { get; set; } = 0.3;
```

```
    public void setData(double[][] _inputs, List<string> _columnNames)
    {
        this.inputs = _inputs;
        this.columnNames = new string[_columnNames.Count];

        for (int i = 0; i < _columnNames.Count; i++)
        {
            this.columnNames[i] = _columnNames[i];
        }
    }

    public PcaReturnStruct PCA_Algortihm(int labelIndex)
    {
        PcaReturnStruct result = new PcaReturnStruct();
        result.independentVariable = new List<string>();
        result.independentVariableValue = new List<double>();
        result.dependentVariable = new List<string>();
        result.dependentVariableValue = new List<double>();

        sda = new DescriptiveAnalysis(columnNames).Learn(inputs);
        var method = (PrincipalComponentMethod)(AnalysisMethod.Center);
        pca = new PrincipalComponentAnalysis(method);
        pca.Learn(inputs);

        for (int i = 0; i < columnNames.Length; i++)
        {
            for (int j = 0; j < columnNames.Length; j++)
            {
                if ((i != j) && i == (labelIndex))
                // if (i != j)
                {
                    if (Convert.ToDouble(sda.CorrelationMatrix[i, j]) < independentCoefficient)
                    {
                        if (result.independentVariable.IndexOf((columnNames[j] + " - " +
columnNames[i])) == -1)
                        {
                            result.independentVariable.Add((columnNames[i] + " - " +
columnNames[j]));
                            result.independentVariableValue.Add(sda.CorrelationMatrix[i, j]);
                        }

                    }
                    else if (Convert.ToDouble(sda.CorrelationMatrix[i, j]) > dependentCoefficient)
                    {
                        if (result.dependentVariable.IndexOf((columnNames[j] + " - " +
columnNames[i])) == -1)
                        {
                            result.dependentVariable.Add((columnNames[i] + " - " +
columnNames[j]));
                            result.dependentVariableValue.Add(sda.CorrelationMatrix[i, j]);
                        }
                    }
```

```
            }
        }
    }

    result.corelasyonMatrix = sda.CorrelationMatrix;
    return result;
  }
}


public class Regression
{

  private DataTable dataTableRegression;
  private LogisticRegressionAnalysis lra;
  private MultipleLinearRegressionAnalysis mlr;
  private DataTable sourceTable;

  private double[][] inputs;
  private double[] outputs;

  public void setData(DataTable _dataTable, double[][] _inputs)
  {
    this.inputs = new double[_inputs.Length][];
    this.dataTableRegression = _dataTable;
    this.inputs = _inputs;
  }

  public MultipleLinearRegressionAnalysis Regression_Algortihm(string dependent_Name,
List<string> independent_Names)
  {
    sourceTable = dataTableRegression;
    String dependentName = dependent_Name;
    DataTable dependent = sourceTable.DefaultView.ToTable(false, dependentName);


    List<string> names = new List<string>();
    foreach (string name in independent_Names)
      names.Add(name);

    String[] independentNames = names.ToArray();
    DataTable independent = sourceTable.DefaultView.ToTable(false, independentNames);

    this.inputs = independent.ToJagged();
    outputs = dependent.Columns[dependentName].ToArray();

    var sda = new DescriptiveAnalysis()
    {
      ColumnNames = independentNames
    }.Learn(inputs);
```

```
        lra = new LogisticRegressionAnalysis()
        {
            Inputs = independentNames,
            Output = dependentName
        };

        mlr = new MultipleLinearRegressionAnalysis(intercept: true)
        {
            Inputs = independentNames,
            Output = dependentName
        };
        MultipleLinearRegression reg = mlr.Learn(inputs, outputs);

        return mlr;
    }
}


public class K_Means
{
    private double[][] inputs;
    public int k { get; set; } = 3;
    public double tolerance { get; set; } = 0.05;

    public void setData(double[][] _inputs)
    {
        this.inputs = _inputs;
    }


    public KMeansReturnStruct K_Means_Algortihm()
    {
        KMeansReturnStruct result = new KMeansReturnStruct();
        result.clusterIndex = new List<int>();

        KMeans kmeans = new KMeans(k, new SquareEuclidean())
        {
            Tolerance = tolerance
        };

        int[] idx = kmeans.Learn(inputs).Decide(inputs);
        result.idx = idx;
        result.inputs = inputs;

        for (int j = 0; j < kmeans.Clusters.Clusters.Length; j++)
        {
            result.clusterIndex.Add(kmeans.Clusters.Clusters[j].Index);
        }

        return result;
    }
}
```

```
public class Apriori
{
   private string[][] inputs_str;
   public double confidenceCoef { get; set; } = 0.85;
   public double suppCoef { get; set; } = 70;

   public void setData(string[][] _inputs)
   {
      this.inputs_str = _inputs;
   }

   public AprioriReturnStruct Apriori_Algortihm()
   {
      AprioriReturnStruct result = new AprioriReturnStruct();
      result.confidence = new List<double>();
      result.support = new List<double>();
      result.associatedVariable = new List<string>();
      AssociationRule<string>[] rules;

      var apriori = new Apriori<string>(threshold: 1, confidence: 0);
      AssociationRuleMatcher<string> classifier = apriori.Learn(inputs_str);
      rules = classifier.Rules;


      var orderRules = from rule in rules
                       orderby rule.Confidence descending
                       select rule;

      foreach (var item in orderRules)
      {
         if (item.Confidence > confidenceCoef && item.Support > suppCoef)
         {
            result.confidence.Add(item.Confidence);
            result.support.Add(item.Support);
            result.associatedVariable.Add(item.ToString().Split(';')[0]);
         }
      }

      return result;
   }
}


public class KiKare
{
   private DM_DataSet dmDataSet;
   public double alfa { get; set; } = 2.03426859549273E-100;

   public void setData(DM_DataSet _dmDataSet)
   {
```

```
            this.dmDataSet = _dmDataSet;
        }

        public List<List<string>> KiKare_Algortihm()
        {
            List<List<string>> result = new List<List<string>>();
            List<string> lrResult;
            var data = dmDataSet.allDataTable;
            var feature = new Codification() { };

            feature.Learn(data);
            string outputName;

            double[] outputs = feature.Apply(data,
dmDataSet.dependentFeatureName).ToVector(out outputName);
            string[] inDependentColumnName = dmDataSet.inDependentFeatureName.Where(x =>
x != null).Select(x => x.ToString()).ToArray();

            for (int i = 0; i < inDependentColumnName.Length; i++)
            {
                string[] inputNames;
                double[][] inputs = feature.Apply(data, inDependentColumnName[i]).ToJagged(out
inputNames);
                lrResult = new List<string>();

                var lra = new LogisticRegressionAnalysis()
                {
                    Inputs = inputNames,
                    Output = outputName
                };
                LogisticRegression regression = lra.Learn(inputs, outputs);
                lra.ChiSquare.Size = alfa;
                ChiSquareTest x2 = lra.ChiSquare;

                lrResult.Add(inDependentColumnName[i]);
                lrResult.Add(x2.Significant.ToString());

                //if (x2.Significant == true)
                result.Add(lrResult);
            }

            return result;
        }
    }

}
```

ÖZGEÇMİŞ

Adı Soyadı	: Hilal ÇELİKMAKAS
Doğum Yeri ve Tarihi	: Bursa, 06.02.1991
Yabancı Dil	: İngilizce
Eğitim Durumu	
Lise	: İnegöl Anadolu Lisesi
Lisans	: Dumlupınar Üniversitesi
Yüksek Lisans	: Bursa Uludağ Üniversitesi
Çalıştığı Kurum/Kurumlar	: Kuartek Proje Tasarım Yazılım Araştırma Ltd. Şti. Hadim Alışveriş Merkezleri Tic. A.Ş. DeltaV Uzay Teknolojileri A.Ş.
İletişim (e-posta)	: hcelikmakas@gmail.com
Yayınları	: -