T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BULUT BİLİŞİM ÜZERİNDE ANLAMSAL WEB SERVİSLERİNİN KULLANIMI İÇİN MİMARİ TASARIM

GÖKAY BURAK AKKUŞ

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Doç. Dr. Erdem UÇAR

EDİRNE-2017

T.Ü. Fen Bilimleri Enstitüsü onayı

Prof.Dr. Murat YURTCAN
Fen Bilimleri Enstitüsü Müdür V.

Bu tezin Doktora tezi olarak gerekli şartları sağladığını onaylarım.

Doç.Dr. M. Tolga SAKALLI
Anabilim Dalı Başkanı

Bu tez tarafımca okunmuş, kapsamı ve niteliği açısından bir Doktora tezi olarak kabul edilmiştir.

Doç. Dr. Erdem UÇAR
Tez Danışmanı

Bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir Doktora tezi olarak oy birliği ile kabul edilmiştir.

Jüri Üyeleri	İmza
Doç.Dr. Erdem UÇAR	
Doç.Dr. Rafet AKDENİZ	
Prof.Dr. Tahir ALTINBALIK	
Doç.Dr. Erdinç UZUN	
Yrd.Doç.Dr. İlhan UMUT	

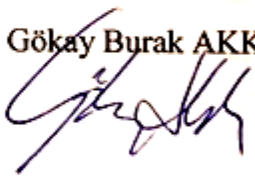
Tarih: 05/07/2017

T.Ü. FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ DOKTORA PROGRAMI
DOĞRULUK BEYANI

İlgili tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin kaynak gösterilerek ilgili tezde yer aldığını beyan ederim.

06/06/2017

Gökay Burak AKKUŞ



Doktora Tezi

Bulut Bilişim Üzerinde Anlamsal Web Servislerinin Kullanımı İçin Mimari Tasarım

T.Ü. Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Bulut bilişimi, sanal kaynakları standartlara dayalı şekilde yönetmeyi sağlayan teknoloji evrimini ifade etmektedir. Bu sanal kaynaklar, mantıksal olarak geniş bir havuzda toplanır. Öngörülemeyen kaynak ihtiyaçlarının olağan bir durum olarak kabullenildiği tipteki uygulamaların ihtiyaçlarının karşılanması açısından önemli bir fırsat doğmaktadır. Öngörülemeyen ihtiyaçlara sahip uygulamalara örnek olarak yoğun zamanlarda milyarlarca istek karşılayan veri yoğun web servislerini gösterebiliriz. Burada yer alan çalışmanın amacı, veri yoğun web servislerinin bulut bilişimi teknolojisi ile desteklenebilmesi ve uygun bir kataloglama ile özelliklerine uygun ortamlarda çalışmasının sağlanması için bir altyapı oluşturmaktır. Web Servisinin işleyeceği veri miktarı, harcayacağı kaynaklar ve sonucu dönmek üzere kullanacağı bant genişliği, bu paradigma değişikliği ile birlikte maliyetlendirilmesi, servisin kullanımından önce servisi kullanmak isteyen tarafa bildirilmesi gereken bir unsur olarak karşımıza çıkmaktadır. Bu çalışmada, bu tip servislerin sınıflandırılması ve sorgulamanın kriterlerinin değerlendirilmesi ile yaklaşık bir maliyet hesaplamasına yönelik temel oluşturulması hedeflenmektedir.Bu tahminleme sayesinde ileriye dönük kaynak kullanım ihtiyaçlarının da öngörülmesi sağlanabilecek ve bu şekilde ölçekleme konusunda katma değerli servislerin sağlayıcılarına yol gösterici bir araç sunulabilecektir.

Yıl : 2017

Sayfa Sayısı : 99

Anahtar Kelimeler : Web Servisleri, Bulut Bilişim, Servis Maliyet Öngörüsü

Doctoral Thesis
An Architectural Design For The Use Of Semantic Web Services On Cloud Computing
Trakya University Institute of Natural Sciences
Computer Engineering

ABSTRACT

Integration on the web became a reality since cloud computing, site-to-site integrations and “Software as a Service” approach have been widely accepted as industry standards. It would be possible to see the true potential of such a distributed infrastructure when these services can be combined together and executed as parts of a singular workflow. The collective composition of web services requires adaptive and dynamic systems where all parties may be subject to unexpected changes. It also requires intelligent systems that are able to make use of services as building blocks that meets user requirements with specific functional and non-functional attributes. The decision is to be made by selecting the appropriate service from a set of discovered candidates. The vision is to provide computing services to users on demand and charge them based on their usage and Quality of Service (QoS) expectations. In the cloud environments, when there are multiple services meeting the requirements, the main attribute to be considered becomes the price.

In this research, we present a general framework that guides service discovery and service composition by providing a predicted price against the requester’s requirements in the data driven web services domain that are hosted on a cloud environment.

Year : 2017
Number of Pages : 99
Keywords : Web Services, Cloud Computing, Service Cost Estimation

TEŞEKKÜR

Tez çalışmam sırasında kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösterici ve destek olan değerli danışman hocam sayın Doç. Dr. Erdem UÇAR'a, ilgisini ve önerilerini göstermekten kaçınmayan sayın Prof. Dr. Yılmaz KILIÇARSLAN'a sonsuz teşekkür ve saygılarımı sunarım.

Tez ilerleme süreci boyunca yardım, bilgi ve tecrübeleri ile bana sürekli destek olan Prof. Dr. Tahir ALTINBALIK'a ve tezin ilerleyen aşamalarında izleme komitesine katılan Yrd. Doç. Dr. İlhan UMUT'a teşekkür ederim.

Tez savunma aşamasındaki değerli katkıları için Doç. Dr. Rafet AKDENİZ'e ve Doç. Dr. Erdinç UZUN'a teşekkür ederim.

Çalışmalarım boyunca akademik değerlendirmelerimi paylaştığım değerli arkadaşım Dr. Yaşar SAFKAN'a teşekkürü bir borç bilirim.

Çalışmalarım boyunca maddi manevi destekleriyle beni hiçbir zaman yalnız bırakmayan başta eşim Nihan AKKUŞ olmak üzere tüm aileme de sonsuz teşekkür ederim.

Edirne **Gökay Burak AKKUŞ**

ÖZGEÇMİŞ

KİŞİSEL BİLGİ

Soyadı, Adı: Gökay Burak AKKUŞ

Uyruğu: TC

Doğum Tarihi: 12 Şubat 1981

Telefon: +90 532 412 1036

Email: gokayakkus@trakya.edu.tr

EĞİTİM

Derece	Kurum	Mezuniyet Yılı
Yüksek Lisans	Boğaziçi Üniversitesi Bilgisayar Mühendisliği	2006
Lisans	Ege Üniversitesi Bilgisayar Mühendisliği	2002

YABANCI DİLLER

İngilizce : Çok İyi, Almanca: Başlangıç

YAYINLAR

Akkuş, G. B., Uçar, E., Ünlü, N. (2015). Bulut Bilişim Servisleri Maliyet Tahminlemesi. 2. Ulusal Yönetim Bilişim Sistemleri Kongresi, Sıra No:189-Bildiri No:256

Akkuş, G.B. ve Uçar E., "Bulut Bilişimi: Bulutlarda Çalışacak Servislerin Tasarımı ve Sınıflandırılması", 4. Mühendislik ve Teknoloji Sempozyumu, Ankara, 2011

Akkuş, G.B., Uçar, E., INISTA 2010 konferansı dahilinde "International Symposium on INnovations in Intelligent SysTems and Applications" bildiri kitapçığındaki "Network Algorithms for Automation of Web Services Composition", 214-218 pp., Kayseri, Türkiye, Haziran 2010

Akkuş, G. B., Semantic Web Services Composition: A Network Analysis Approach. ICDE Workshops 2007: 937-943

İlhan, E. S., Akkuş, G. B., Bener, A. B., Improved Service Ranking and Scoring: Semantic Advanced Matchmaker (SAM). ENASE 2007: 95-102

İlhan, E. S., Akkuş, G. B., Bener, A. B., SAM: Semantic Advanced Matchmaker. SEKE 2007: 698-703

İÇİNDEKİLER

BÖLÜM 1

GİRİŞ

Bulut bilişimi basit anlamda dağıtık kaynakların tek ve bütün bir kaynakmış gibi görünmesini sağlar. Böylece tüketici tarafında kaynağın bulunduğu yer, kullanım oranları, yönetime ait alt detaylar hissedilmez. Arka uçta bu dağıtık kaynakların yönetimi, otomasyonu sağlanmış olan sistem denetleme ve yönetim araçlarıyla sağlanır. Bulut üzerinde platform, uygulama veya servisler sunulabilmektedir.

Web Servisleri, kurumlar arasında veya kurumların kendi iş kolları arasında karşılıklı ve birlikte çalışmayı sağlayan modüler web tabanlı uygulamalardır. Kurumları bu servis tabanlı entegrasyon yöntemini kullanmaya iten iki temel eğilim öne çıkmaktadır.

Bu eğilimlerden ilki, müşteri ihtiyaçlarına zamanında cevap verebilmek adına destekleyici bir ortam oluşturmaktır. Bu durum ancak bilgi sistemleri arasında veri akışının sağlanmasıyla gerçekleşebilmektedir. Veri akışı kurumların iç sistemleri arasında olabileceği gibi, kurumun dışında ortaklıklar kurulan farklı kurumlara doğru da olabilmektedir. Belli bir iş sürecini tamamlayabilmek için bu şekilde etkileşim içinde olan sistemler arasında entegrasyon ihtiyacı doğmaktadır. Bu ihtiyacın karşılanması, standartlara dayalı bir arayüz ve doğru ortamlarda erişilebilirliğin sağlanması ile mümkün olmaktadır. Standart arayüz olarak XML Web Servisleri geçerliliğini ortaya koyarken, erişilebilirlik için gerekli altyapıyı sağlayan ortam olarak Bulut Bilişimi öne çıkmaktadır.

Geçmişte kurumlar daha çok iç yapılarına odaklanarak kendi sistemleri arasında etkileşimi sağlamaya çalışmışlardır. Ancak, bilgi sistemleri tasarımındaki gelişmeler veri akışının yönünü içeriden dışarıya çevirmiştir. Bu değişim, kurumlar arası süreçlerin oluşmasına neden olmuştur. Süreçlerdeki paydaşlar sürekli olarak değişebilmektedir ve bu nedenle her kurum ihtiyacı olan işlevselliği sağlayan servisleri arayıp bulmak

ihtiyacındadır. Arama ve bulma işini başarabilmek için servislerin ve işlevlerinin anlamsal olarak tanımlanabilmesi gerekmektedir. Anlamsal tanımlama olan üstveriyi kullanan yazılım etmenleri, farklı servisleri bir arada kullanarak ölçek büyütme veya sistemleri entegre etme yoluna gidebilmektedir. Ölçek büyütme konusunda ana tıkanma noktası, servislerin uygulama parçacıkları şeklinde derlenerek bir süreç haline getirilmesidir. Bu durum servislerin otomatik olarak bulunup kullanılması problemini adreslemektedir.

Diğer bir eğilim ise servisleri süreç bölümleri olarak modellemektir. Servisleri Internet üzerinde merkezileşmeden kurtararak dağıtık bir şekilde farklı iletişim aygıtlarından erişilebilir kılmak mümkündür. Kurumlar karmaşık, yavaş ve pahalı yazılım entegrasyonu yükünden kurtarılarak kendi ana konuları olması gereken, sunacakları ürünler ve kritik görevlere odaklanabileceklerdir [1].

Bu çalışmada, servislerin modellenmesi ve bu sayede bulut bilişimi için uygun bir altyapı tasarımına temel oluşturulması planlanmaktadır.

BÖLÜM 2

BULUT BİLİŞİMİ (CLOUD COMPUTING) VE WEB SERVİSLERİ

Bulut bilişiminin henüz kabul görmüş ortak bir tanımı yoktur [2]. Ulusal Standartlar ve Teknoloji Enstitüsü (National Institute of Standards and Technology - NIST) [3] bulut bilişimin beş temel özelliğini tanımlamıştır. Bunlar, isteğe dayalı self-servis, geniş ağ erişimi, kaynakların ortak kullanımı, hızlı genişlemeye uygun esneklik ve ölçülmüş servistir. Aynı zamanda, bulut bilişim, dinamik ve sıklıkla kolay genişletilen, kullanıcılara İnternet üzerinden şeffaf sanallaştırılmış kaynalar sunan servis olarak da tarif edilmiştir [4]. Bulut bilişim mimarisi üç katmandan oluşur: (i) Servis olarak yazılım (Software as a Service - SaaS); (ii) Servis olarak platform (Platform as a service - PaaS) ve (iii) Servis olarak altyapı (Infrastructure as a Service - IaaS) [5]. Bulut yapıları, istemci, uygulamalar, platformlar, altyapılar ve servislerden oluşan beş bilişimli birer mimari olarak da görülür [6]. Mevcut bulut bilişim yapıları dört farklı şekilde yayılırlar: (a) fiziksel altyapının servis sağlayıcıya ait olduğu ve onlar tarafından yönetildiği açık bulutlar; (b) fiziksel altyapının kuruluşların oluşturduğu bir konsorsiyuma ait olduğu topluluk bulutları; (c) altyapının belli bir organizasyona ait olduğu özel bulutlar ve (d) sayılan üç tipin kombinasyonlarından oluşan hibrid bulutlar [7].

Bulut bilişimin kavramsal altyapısının oluşması, “mainframe” tabir edilen makinaların ilk üretildiği zamana kadar geriye gider. Mainframe makinalar, zamanına göre yüksek kapasiteye sahip makinalardı ve bunların işleme zamanı gibi kaynakları, terminaller üzerinden pek çok kullancı tarafından paylaşılarak kullanılırdı. Bulut bilişim, bu modele benzerlik gösterir. Bulut bilişimde, terminaller yerine, İnternet tabanlı sanallaştırma teknolojileri kullanılır. Bu teknolojilerin sayesinde, kullanıcılar veya istemciler, dünyanın farklı bölgelerinde bulunan bilgisayar kümelerini tek bir süper

bilgisayar kullanıyormuşçasına paylaşabilirler [8]. Elektrik kullanırken, elektriğin hangi yöntemle üretildiğini veya hangi özel tesisten geldiğini bilmediğimiz gibi, bulut bilişim kaynaklarını kullanırken de, servisi veren makinelerin nerede bulunduğunu bilmemize gerek yoktur.

Bulutlar, hem kişiler hem de girişimler için olağanüstü faydaları ortaya çıkartırlar. Bulutlar, maddi tasarruf, dış kaynak mekanizmaları, kaynak paylaşımı, her yerde ve her zaman erişim, isteğe bağlı ölçeklenme sağlarlar. Bulutlar, yazılım güncelleştirmeleri ve yükseltmeleri, lisanslar ve bakım gibi teknik ayrıntıları maskeleyerek, kullanıcıların işini kolaylaştırırlar. Bulutlar, tekil sunucu kurulumlarına göre, bilgi güvenliği açısından da avantajlı olabilirler. Bulutlar çok miktarda kaynağı bir araya getirerek yönettikleri için, bulut sağlayıcıları, tekil şirketlerin bilgi güvenliğinde sınırlı bilgi sahibi bir ağ yöneticisi yerine, bilgi güvenliği için bünyelerinde uzmanlar bulundurabilirler. Benzer şekilde, bulutlar büyük boyları ve kaynak esnekliği sayesinde, DDoS (Distributed Denial of Service attack) saldırılarına karşı daha dayanıklı da olabilirler. Sanallaştırma teknolojileri sayesinde, bulutlarda sanal makinalar bir fiziksel makinadan diğerine kolaylıkla aktarılabilir. DDoS saldırılarına karşı koymanın yanısıra, bu şekilde mobilleşen makinalar sayesinde, tek bir sistem yöneticisinin bütün kaynaklara tek başına hükmetmesi durumundan da kaçınılmış olur [9].

2.1. Bulut Bilişim Servisleri

Bulut bilişime dair servisler üç ana başlık altında [10] toplanmaktadır:

SaaS – Software as a Service (Yazılımın Servis Olarak Sunulması)

Bu kapsamda yer alan servisler, uç kullanıcıya veya üçüncü partilere doğrudan iş gören servislerin sunulması şeklindedir. Yazılım kiralama modelleri bu tip servislere örnektir.

PaaS – Platform as a Service (Platformun Servis Olarak Sunulması)

Bu kapsamda yer alan servisler, yazılım veya sistem geliştirmeyi sağlayan araçları sunmaktadır. Hedefinde kendi oluşturacağı yazılımı belli bir ortamda çalıştırmayı

planlayan üçüncü partiler vardır. Oluşturdukları servisleri kendileri kullanmak veya müşterilerine sunmak üzere bu platformdan faydalanmayı amaçlarlar.

IaaS – Infrastructure as a Service - (Altyapının Servis Olarak Sunulması)

Bu kapsamda yer alan servisler, daha çok donanım yönetimi sunar. Sunucu, ağ ve depolama aygıtları kullanım bazında servisler olarak sunulmaktadır. Temel olarak sunulan işlev sunucu parkının provizyonu ve bakım otomasyonudur.

Bu servislerin popüler sağlayıcıları Google App Engine, Amazon Elastic Compute Cloud (Amazon EC2), Microsoft Azure ve Salesforce olarak sıralanabilir. Bu servisler (IaaS, Paas ve SaaS) kamuya açık (public cloud), veya özel kullanım (private cloud) için sınırlandırılabilir. Bazen de bu servisler kamuya açık ve özel bulut bileşimi olabilecek bir melez bulut (hybrid cloud) üzerinde barındırılabilir.

2.2 Web Servisleri

Web servisleri "kendi kendine yeten, kendini açıklayan modüler uygulamalardır". World Wide Web Consortium - W3C sözlüğünde ise Web Servisleri "dışa açık arayüzleri ve bağlantıları XML (eXtensible Markup Language) olarak tariflenen ve bir URI (Uniform Resource Identifier) ile tanımlanan yazılım sistemi" olarak ifade edilmektedir [11]. Web Servisleri, uygulamalar arası entegrasyonun daha az maliyetle ve kolayca yapılmasını sağlamaktadır. Web Servisleri mimarisi servis sağlayıcı, servis istemci, servis kayıtlanması olmak üzere üç temel role dayanır.

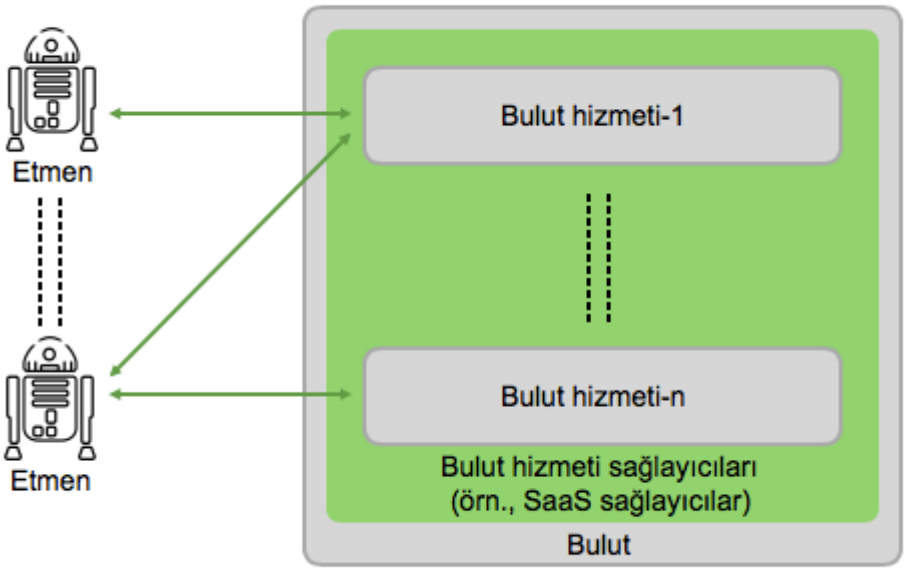
Servis sağlayıcı, servisin ve sağladığı verinin sahibidir. Servis istemci ise ihtiyacı olan işlevselliği sunan servisi çağrım yapmak yoluyla kullanan uygulama veya iş birimidir. Servis kayıtlanması ise servislere ait bilgilerin bulunduğu kayıtçı mekanizmasıdır. Bu kayıtlanmalarda anlamsal olarak servisin neler sunduğu, hangi parametrelerle çalıştığı, hangi veri yapısını sunduğu ve kullanım detayları yer almaktadır. Bu üç rolün bileşimi mevcut Web Servisleri mimarisini oluşturmaktadır [1].

Genel mimari henüz tamamlanmamış olsa da, Web Servisleri belli bir tanım kümesine dayanır ve çeşitli iş etkileşimlerini desteklemektedir. Web Servisleri farklı sistemler arasında XML-tabanlı mesajların değişimine dayanmaktadır.

2.3. Servislerin Fiyatlandırılması

Servis olarak sunulan bu kaynaklar genelde kullandıkça öde (pay-as-you-go) planı ile sipariş edilir. Önceleri şirket içinde bulunan servisler gittikçe daha fazla şirket tarafından bulut servislerine geçirilmektedir. Böylelikle hem maliyet düşmekte hem de bakım işi azaltılmaktadır [12].

Üç katmanlı bulut hizmeti provizyon yapısı üç bileşenden oluşur: son kullanıcı, bulut servis sağlayıcı ve bulut altyapı sağlayıcıları [13]. Son kullanıcı, iş hizmeti sağlayıcısı ile bir iş ilişkisi içerisinde olan ve bulut servisi talep eden, bir kişi veya kuruma karşılık gelen bir bulut kullanıcısıdır. Bir bulut servisi sağlayıcısı, iş uygulamasını kiralanmiş bulut altyapısında yayınlayan ve çalıştıran bir iş hizmeti sağlayıcısıdır. Böylelikle, bu bulut hizmetleri son kullanıcılara ağ erişimi üzerinden sunulur. Bulut altyapı satıcısı ise işlemci, bellek, ağ ve diğer ana bilgi işleme kaynaklarını müşterilerine kullandıkça öde yöntemi ile sağlar [14].



Şekil 2.1. Son kullanıcıların, SaaS sağlayıcıları ile etkileşim şekli. Son kullanıcılar, etmenler aracılığıyla bulut hizmet sağlayıcılarının servislerine erişirler. SaaS sağlayıcıları, bulut altyapı satıcılarından hizmet alır ve hizmeti son kullanıcılara satar.

Hizmet provizyon prosedürü kısaca şu şekilde özetlenebilir:

1. Son kullanıcı, bir SaaS sağlayıcının hizmet kataloğunu tarar ve ilgili hizmet için bulut servis sağlayıcısına bir istek gönderir.
2. Bulut servis sağlayıcısı, son kullanıcının isteğini kabul eder ve talep üstüne sanal kaynaklar için altta kullandığı bulut altyapı satıcısına iletir.
3. Bulut altyapı satıcısı, kaynak kira isteğine cevap verir. İlgili SaaS sağlayıcısına son kullanıcının isteğini yerine getirebilmesi için Sanal Makine (VM – Virtual Machine) tahsis eder.
4. Son olarak, SaaS sağlayıcısı son kullanıcıdan hizmetleri karşılığında ücret talep eder ve bulut altyapı satıcısına Sanal Makinelere karşılık ödeme yapar.

Aradaki etkileşim Şekil 2.1'de gösterilmektedir.

Daha önce yapılmış olan çalışmalarda, Macias ve Guitart [15] bulut bilişim pazarlarında fiyatlama politikası konusunu ortaya atmışlardır. Ayrıca, bir diğer çalışmalarında [16] servis sağlayıcıların fiyatlarını rekabet şartları, zaman aralığı ve servis seviyesi anlaşması gibi parametrelere göre düzenlediklerinde, ciro artırma gibi iş

hedeflerine daha iyi ulaşabildiklerini göstermişlerdir. Ancak, önerdikleri model fazlaca katıdır ve pazardaki diğer katılımcıların da her zaman rasyonel davrandığını varsayar. Yine bu çalışmada, fiyatlama esnekliğinin yanısıra, fiyatlama fonksiyonlarına da esneklik getirilmiştir. Böylece, fiyatlama fonksiyonları kendi kendilerini değiştirerek değişen pazar şartlarına daha iyi uyum sağlayabilirler.

Ferguson et al. ise, çalışmalarında [17] ekonomik teorilerin kaynak yönetimine uygulanmasını tartışır. Aiber et al. ise, çalışmalarında [18] otonom öz optimizasyon için bir mimari sunarlar. Müşteri sınıflandırmanın, müşteri özelliklerine bağlı fiyat ayrıştırma gibi bazı öğelerinden ise diğer makalelerde bahsedilmiştir (Newhouse et al. ve Buyya) [19, 20]. Ancak, bu çalışmalarda diğer ayrıştırma öğelerinden bahsedilmemiştir. Chicco et al. tarafından yayınlanan makalede ise [21] elektrik dağıtım şebekelerinde müşteri sınıflandırması için veri madenciliği yöntemlerinden bahsedilir, ancak konu birbirine benzer davranışlar sergileyen müşterileri bulmak üzerine odaklanmıştır.

e-ticaret sitelerine giriş kontolü için kullanılmak üzere, kullanıcıların oturumlarını bir ürün satın alma niyetini esas alarak önceliklendirmeye yönelik bir mimari Poggi et al. tarafından iki ayrı çalışmada önerilmiştir [22, 23]. Bu çalışmalarda, müşterilerin bir e-ticaret sitesindeki gezinme tıklamaları gibi davranışları analiz edilmiştir. Bu davranışlara dayalı olarak, kullanıcının bir ürünü satın alma niyetinin olup olmadığını öngörmenin mümkün olduğunu göstermişlerdir. Sonuç olarak, servis kalitesi ve önceliklendirme, kullanıcıların niyetlerine göre ayarlanabilmiştir.

Bouhton et al. makalelerinde [24] iş yükü sınıfının, düşük seviyede kaynak ayrılması problemine uygulanabileceği konusundaki araştırmalarını sunmuşlardır. Odak noktaları, veritabanı üzerinde kaynaklar için rekabet içinde olan iş yükleridir. Bu şartlar altında, iş yüklerinin göreli önem seviyelerinin, kaynakların verimli bir şekilde kullanılmasını sağlayacak kuralları tanımlamışlardır.

Uçtan uca Servis Kalitesi sağlamaya bir yaklaşım da, Ayırma ve Paylaştırma için Globus Mimarisidir (Foster et al.) [25]. Bu yaklaşım, Servis Kalitesi sağlamak için önceden ayırma yöntemini kullanır. Otonom Servis Kalitesi sağlamak için kaynak yönetimi yapmanın bir diğer yolu, performans modellerini kullanır (Kounov et al.) [26]. Bu çalışmada, servis seviyesi anlaşmalarının sağlanabilmesi için, herhangi bir işin performansa olan etkisini hesaplayabilecek ve kaynak paylaşımını düzenleyebilecek

kaynak yöneticileri tasarlanması için bir çerçeve sunarlar. Her iki yaklaşımda da, kısmi kaynak arızası durumunda Servis Kalitesi sağlanması kaygısı bulunmamaktadır.

Risk yönetiminin bulut bilişime dahil edilmesi (Dejamme at al. 2006) [27] sayesinde, Service Level Aggreement – SLA (Hizmet Seviyesi Anlaşması) kullanımına daha dinamik bir yaklaşım mümkün hale gelmiştir. Bu sayede, SLA'e uyulamaması riski modellenebilir hale gelir. Böyelece, servis sağlayıcı farklı risk profillerine sahip SLA'ler önerebilir. Ancak, bu şekildeki bir risk modellemesi çok karmaşık hale gelebilir. Bu modellemenin yapılabilmesi için, hata sebeplerinin ve her bir sebebin ortaya çıkma olasılıklarının bilinmesi gerekir. Müşterilerin, servis sağlayıcıların ortaya koyduğu modelleri doğrulayacak imkanlara sahip olmaları ve risklerin oluşturulacak SLA'ler içinde modellenmesi gerekir.

Internet Servis Sağlayıcılar için ciro yönetiminden kavramların kullanılması, Nair ve Bapna tarafından araştırılmıştır (2001) [28]. Bu çalışmada, müşterilerin kabul veya reddedilme kararı incelenmiş, fakat farklı servis tipleri veya ileri rezervasyon hesaba katılmamıştır.

Küme sistemleri için ciro yönetimi kavramlarını inceleyen ilk çalışmalardan biri, Dube et. al. tarafından yayınlanmıştır (2005) [29]. Önerilen model, bir kaynağı farklı fiyarlarla sunar. Müşteri davranışının logit bir model takip ettiğini varsayarak, yazarlar az sayıda fiyat kategorisi için optimizasyon modeli oluşturmuşlar ve nümerik sonuçlarını yayınlamışlardır.

Sulisto et. al. (2008) [30] ise, fazla rezervasyon stratejilerinin, iptal ve vazgeçme durumlarının etkilerini azaltmak için nasıl kullanilabileceğini ve bu sayede cironun nasıl artırılabileceğini analiz etmişleridir. Simülasyonlar kullanarak, farklı fazla rezervasyon yöntemleri karşılaştırılmış ve denenmiştir. Urgaonkar et. al. (2002) [31] fazla rezervasyonun payaşımlı barındırma platformlarında faydalı olabileceğini göstermişlerdir; ancak optimizasyon için sadece bir tek çıktı hızı hesaplanmıştır.

Ciro yönetiminin dağıtımlı hesaplama alanına uygulanması için bir çatı Anandasivam ve Neumann tarafından (2009) [32] hazırlanmıştır. Bu çalışmada, teorik bir model ve kaynak gruplarının fiyatlanması için belli gereksinimler sunulur.

Bu çalışmada sadece son kullanıcı ve Saas servis sağlayıcıları arasındaki etkileşim konu edilmektedir. Son kullanıcının bakış açısından bir iş uygulamasına olan istek her

zaman performans ihtiyaçlarını belirten bir Hizmet Seviyesi Anlaşması (HSA) ile beraber gelir.

SaaS sağlayıcının bakış açısından operasyonel amaç, bulut servisinin son kullanıcının istediği hizmet seviyesinde bulut servisini takip ederken olabildiğince az sanal kaynak kiralamaktır. SaaS sağlayıcısının karı, son kullanıcıdan sağlanan ciro ile altyapı kirası arasındaki fark kadardır.

HSA, biri son kullanıcı diğeri hizmet sağlayıcısı olan iki parti arasında müzakere edilmiş bir anlaşmadır. Formal veya informal bir sözleşme olabilir. HSA fiyat, zaman ve iş gibi bir küme servis kalite (QoS) kısıtlarını tanımlar ve hizmetin kullanıcı tarafından nasıl kullanıldığını belirler. Bu anlaşmaya, müzakere sonucunda varılır.

2.3.1 Fiyatlandırma Planları

Bu bölümde servislerin fiyatlandırılması açısından detaylara girmeden önce, genel fiyatlama yaklaşımları dört başlık altında kısaca açıklanmaktadır.

2.3.1.1 Sabit Tekrarlayan Fiyatlandırma

Kullanıcılar her ay aynı miktarda kaynağa sahip olabilecekleri bir plan oluşturur. Bu bugün bulut bilişimde mevcut olan sözleşme planlarına benzerdir.

2.3.1.2 Kaynak Tüketimine Göre Değişken Fiyatlandırma

Bu durumda tek seferde kullanılan kaç tane kaynak olduğuna göre fiyat farklılık gösterebilir. Bu kullanıcıların ihtiyaçları olan kaynak sayısı temelinde ödeme yaptıkları isteğe bağlı plana göre ayarlanmıştır.

2.3.1.3 Süreye Göre Değişken Fiyatlandırma

Fiyat istemcinin kaynakları ne kadar süreyle kullandığına göre farklılık gösterir. Kullanım süresi daha uzun oldukça teklif edilen fiyat da daha azdır. Bu Amazon tarafından sağlanan spot fiyatlandırmada uygulanmıştır.

2.3.1.4 Maliyet Çarpanları

Maliyet çarpanları sağlayıcıların fiyatı bir faktöre göre arttırmasına yardımcı olur. Bulut işlem fiyatlandırma şemalarının hiçbirinde uygulanmamış olduğu görülmektedir. Çoğunlukla sigorta şirketleri tarafından kullanılırlar.

2.3.2 Fiyatın Müzakere Edilmesi

Aşağıdaki kısımda tüketici ve sağlayıcı arasında temel müzakereye olanak sağlayan farklı fiyatlandırma tiplerini açıklanmaktadıre. Değerlendirdiğimiz sağlayıcılar Amazon ve Microsoft Azure'dir.

2.3.2.1 İsteğe Bağlı Örnekler

İsteğe bağlı seçeneği, kullanıcıların herhangi bir uzun süreli taahhüt olmadan kullandıkları kadarı için ödeme yapmalarına olanak sağlar. Bu kullanıcının donanımı düşük bir maliyetle kiralamasına ve tutmasına olanak sağlar. Ancak Amazon saatlik ücretlendirir, tüketici bir buçuk saat kullanmış olmasına rağmen iki tam saat ücreti ödeyecektir. Bu tip fiyatlandırma çoğunlukla kısa işlemler gerektiren uygulamalar için kullanılır. Kullanıcıların uygulamayı bir yıl gibi çok uzun bir süre boyunca çalıştırması gerektiğinde uzun süreli bir taahhüt seçmeleri çok daha iyi olacaktır. Bunun tam bir yıl boyunca isteğe bağlı fiyatlandırma kullanmaktan daha ucuz olduğu kanıtlanmıştır. Bu durumda her bir oluşumun fiyatlandırması oluşum karmaşıklığına göre belirlenir. Grafik işlemciler (Graphics Processing Unit – GPU) veya çok işlemci (CPU) içermeyen oluşumlar çoğunlukla yüksek peformans için değil de basit geliştirme için kullanıldığından otomatik olarak daha ucuz olacaktır fakat aynı zamanda piyasada yüksek talep gören kaynaklar hizmeti kullanmayı isteyen birçok tüketici olacağından daha pahalı olacaktır.

2.3.2.2 Spot Fiyatlandırma Örnekleri

Kullanıcıların oluşumları çalıştırmak için ödemeye razı oldukları en büyük fiyatı teklif etmek zorunda oldukları spot fiyatlandırma kullanıcıların isteğe bağlı

fiyatlandırmadan çok daha düşük bir ücrete bulutta kullanılmayan alanı bir süreliğine rezerve etmesine olanak sağlar.

2.3.2.3 Rezerve Sunucular

Bu fiyatlandırma modeli uzun süreli taahhüt sağlar. Bu kullanıcıların istedikleri her zaman kaynaklarını sağlamaları içindir. Bir sözleşme kullanılması kullanıcıya çok miktarda paradan tasarruf edilmesinin yanı sıra bu kaynakların her zaman hazır olması avantajını sağlar. Yani, uzun süreli bir taahhütte bulunan kullanıcılar kullanmak istedikleri oluşumlar için daha düşük bir ücret öderler.

2.4 Etmen Temelli Müzakere

Etmen temelli otomatik müzakere sabit fiyatlandırma yerine esnek fiyatlandırmaya olanak tanır. Aynı zamanda hem alıcının hem de satıcının gelirinin maksimize olmasını sağlar. Etmen temelli otomatik müzakere için temelde üç yaklaşım bulunmaktadır: Bu yaklaşımlar, karar teorisi, oyun teorisi ve müzakere analizidir. Otomatik müzakere hala gelişime açık bir alandır, çünkü alanda bir takım zorluklar bulunmaktadır. Birincisi ontoloji problemi, ikincisi etmenlerin stratejisi, üçüncüsü de iletişim protokolüdür. [33]

2.5 Akademik araştırma konuları

Bulut bilişimi üzerine yapılan araştırmalar kısaca aşağıdaki başlıklarda toplanmaktadır [34].

- Ekonomik modeller ve uzmanlık
- Modern bilişim ortamlarında heterojenlik ve ölçek
- İletişime dair sınırlamalar
- Ölçekleme yöntemleri
- Programlama modelleri
- Kaynak yönetimi / Kullanım davranışları

Bu çalışmada özellikle kaynak yönetimine odaklanırken, ekonomik modelleme yapılması ve kullanılan kaynakların maliyetine dayalı şekilde fiyatlamaya temel oluşturacak bir tahminleme yapılması amaçlanmaktadır.

BÖLÜM 3

YAPAY ZEKA VE BÜYÜK VERİ

Bu çalışmaya temel olan fikirler yapay zekanın gelişimi ve büyük veriyi işleyebilme yöntemleri sayesinde meydana çıkmıştır. Günümüzde kullanmakta olduğumuz ve aslında henüz başlangış seviyesinde olarak tanımlayabileceğimiz kişisel asistanlardan (Siri veya Cortana gibi), mevcut uygulamaların akıllı bileşenlerle donatılmasına varan yelpazede yenilikçi yaklaşımlara ihtiyaç duyulmaktadır. Yapay zeka ve büyük veri sayesinde işlenen veriler bilgiye dönüştürülmekte ve belli bir bağlam içinde bir karar oluşturulurken kullanılmaktadır. Bu çalışmada, önceki bölümde sözü edilen web servislerin bir otomasyon içinde daha önceden yapılmamış ancak gerçek zamanlı olarak otonom etmenlerin bir pazarlık sonucu oluşturduğu sözleşmeler çerçevesinde kullanılmasını sağlayan çatı mimari tanımlanmaktadır.

Bu mimarinin detaylarına giriş yapmadan önce yapay zekanın tanımlanması faydalı olacaktır. Yapay zeka, normalde insan zekası gerektiren işlerin, makineler tarafından yapılmasını sağlamak için üretilen teoriler ve yapılan geliştirme çalışmalarıdır.

Yani, aslında, “mavi trenler sağa, kırmızı trenler sola gidecek” kararını veren basit bir sistem bile, “yapay zeka” olarak adlandırılabilir. Yalnız, hem tanımından hem de insan algısından ileri gelen sebeplerden, “yapay zeka” denen uygulama ve çalışmaların sınırları, çalışmalar ve zaman ilerledikçe, daralmıştır.

Bir yapay zeka uygulaması, yeterince süre kullanıldıktan ve nasıl çalıştığı net olarak anlaşıldıktan sonra, insanlar (ki bunlara bu alanda çalışanlar da dahildir) bu uygulamalara artık “yapay zeka” olarak değil, “sıradan bir bilgisayar uygulaması” gözüyle bakmaya başlarlar.

Yani, teknik olarak her ne kadar iki seçenekten birini seçen bir sistem dahi yapay zeka sayılsa da, “yapay zeka” pratikte herhangi bir anda, o an itibarıyla halen insan zekası ile yapılan işlerin, makineler tarafından yapılmasını sağlamaya çalışmaya denir hale gelmiştir.

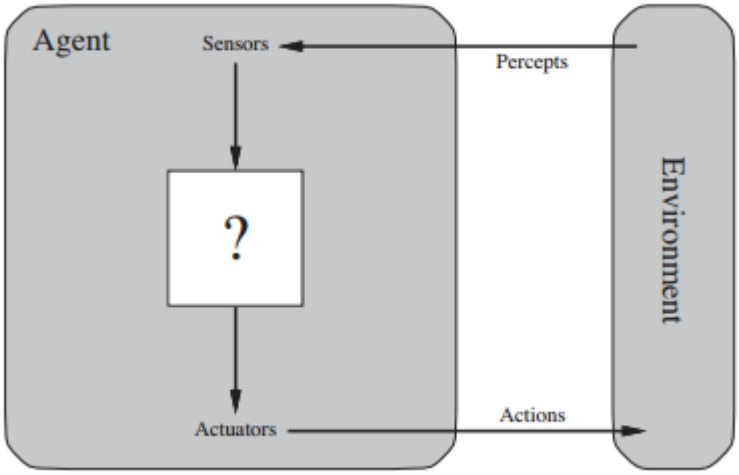
Diğer bir deyişle, her devir için yapay zeka, makinelerin yapma limitinde olduğu, ancak henüz tamamen insanların ellerinden alamadığı işleri yapmasına denir.

2017 yılı itibarıyla, uçaklardaki otomatik pilot sistemlerine yapay zeka gözüyle bakmıyoruz. Buna karşılık, dünya Go şampiyonunu yenen AlphaGo uygulamasına ise, yapay zeka gözüyle bakıyoruz.

3.1. Yapay Zekanın Çeşitleri

En basit şekilde karar veren sistemden, en karmaşık sistemlere kadar, insan zekasının yerini alacak her sistemi yapay zeka olarak tanımladık. Ancak, her yapay zeka sistemi de, aynı türden değildir ve aynı özelliklere sahip değildir.

Yapay zeka dediğimiz zaman, oluşturmaya çalıştığımız şey aslında "intelligent agent" yani "zeki etmen"lerdir. Oluşturmaya çalıştığımız tüm yapay etmenlerin çalışma prensibi, şekilde görüldüğü gibidir. Etmen, algılayıcıları (sensor) yardımıyla çevreden algıları toplar. Sonra da, devindiricileri (actuator) ile eylemlerde bulunur. Her zeki etmen, olası tüm algılara karşılık yapacağı eylemlerle tam olarak tanımlanabilir.



Bu genel yapı içinde, zeki etmenin çalışma şekline göre, yapay zeka sistemlerini bir kaç ana gruba ayırabiliriz.

3.1.1. Refleks Yapay Zeka

Refleks etmenler, içinde bulunduğu durumu, direkt olarak eyleme çeviren etmenlerdir. Yani, yaptıkları sadece algıları anlık olarak değerlendirip, buna uygun eyleme çevirmektir.

Basit olsalar da, refleks etmenler işe yararlar. Mesela, "uzman sistem" dediğimiz sistemler, bu şekilde çalışırlar - ilgili durumda uzman bir insanın alacağı aksiyonları taklit ederler.

Pratikte, refleks etmenler, ya programlama yoluyla ya da kural motorları kullanılarak meydana getirilirler. Programlama yoluyla meydana getirilen bir refleks etmenin davranışını değiştirmek için, programda değişiklik yapmak gerekir. Buna karşılık, bir kural motoru kullanılarak bildirimsel (declarative) hale getirilen refleks etmenlerin davranışını değiştirmek için, geçerli kuralları değiştirmek yeterli olur.

3.1.2. Problem Çözen Yapay Zeka

Problem çözen etmenler, muhtelif algoritmaları kullanarak (arama, kısıtlama sağlama) oluşan duruma uygun eylemleri hesaplar ve uygularlar. Yani, bu tarz bir uygulamada, eylemler direkt olarak algılara karşılık gelmezler, algıların oluşturduğu problem etmen tarafından bir hedef veya hedeflere uygun olarak çözülür ve bulunan çözüme uygun olarak eylem üretilir.

3.1.3 Bilgi Temelli Yapay Zeka

İnsan zekasının en önemli özelliklerinden biri, akıl yürütme ve mantıklı düşünmedir (reasoning). Buradaki temel özellik, insanın sadece algıları üzerinden karar vermeyip, mevcut bilgileri kullanarak akıl yürütme ve mantık yoluyla yeni bilgileri oluşturup, bunları da kararlarında kullanabilmesidir.

Bu özellikleri içeren etmenlere, bilgi temelli etmenler diyoruz. Bu etmenlerde, kural kümeleri veya önceden programlanmış algoritmalar yerine, bilgi bulunur. Bu bilgi, bilgi cümleleri şeklindedir. Bilgi temelli etmen, bilgileri mantık ve çıkarım yöntemleri ile kullanır ve algılara karşılık gelen eylemleri hesaplar.

Bilgiler, etmene baştan sabit bir şekilde verilmediyse ve algılar, eylem üretmenin yanısıra veya sadece bilgi kümesini genişletmek ve güncellemek için kullanılıyorsa, o

zaman elimizde öğrenen bir etmen var demektir. Bu tarz uygulamalar, yaygın olarak "makina öğrenmesi" olarak da bilinir.

3.2. Öğrenme

Öğrenmeyi bir kaç çeşide ayırabiliriz:

- Pekiştirme ile öğrenme (reinforcement learning):
  Bu tarz öğrenmede, etmen, eylemlerinin oluşturduğu sonuçlar için ödüllendirilir veya cezalandırılır. Bu sonuçtan, hangi eylemlerin sorumlu olduğunu bulmak etmene kalır.
- Gözetimli öğrenme (supervised learning):
  Burada, etmene belli sebep-sonuç ilişkileri hazır olarak verilir. Yani, öğrenme çevre ile etmenin direkt ilişkisi ile değil, bir "öğretici" aracılığıyla olur. Buradaki varsayım, "öğretici"nin gereken bilgiye önceden sahip olduğudur.
- Gözetimsiz öğrenme (unsupervised learning):
  Gözetimsiz öğrenmede, etmen, herhangi bir öğreticiden faydalanmadan, algılarını analiz ederek potansiyel olarak faydalı olabilecek şekilde sınıflandırmaya ve bunlarla elde edilen sonuçlar arasında bağıntı kurmaya çalışır.

Öğrenen etmenler, sıklıkla bu tanımlardan birine direkt olarak oturmazlar - aralardaki çizgiler çok net değildir. Mesela, bazı durumlarda, belli sebep-sonuç ilişkileri etmene sağlandıktan sonra, veriler üzerinden gözetimsiz öğrenme yapılıyor olabilir ya da sonrasında sadece sonuçlar üzerinden değerlendirme yapılarak pekiştirmeli öğrenme uygulanabilir.

3.2. Büyük Veri

“Büyük Veri” ve “Sıradan Veri” arasındaki fark genellikle 4-V olarak adlandırılan temel özelliklerin bulunup bulunmadığıyla ayrıştırılır. Bunlar hacim (volume), çeşitlilik (variety), hız (velocity) ve gerçeklik (veracity) olarak listelenebilir [35].

IBM ‘e göre hergün ($2.5x10^{18}$) byte veri oluşmaktadır. Bu ana kadar oluşan verinin %90’nından fazlası son iki yılda meydana gelmiştir. Verinin oluştuğu yerlere örnek olarak hava durumu bilgisini toplamak için kullanılan algılayıcılar, elektronik postalar,

dijital resimler ve videolar, çevrimiçi alışveriş sitelerindeki anlık işlem kayıtları, cep telefonu sinyalleri, araçlardan toplanan anlık konum bilgileri verilebilir.

Büyük veri tarafında yaşanan temel değişim, yeni yaklaşımlar sayesinde bu verinin artık işlenebilir hale gelmesidir. Özellikle bulut teknolojileri bunu daha mümkün hale getirmiştir. Artık küçük bir firma ya da bir birey yalnızca bir web sayfasını kullarak onlarca ve hatta yüzlerce makine üzerinde çalışacak uygulamalar geliştirebilmekte ve bunları ön yatırım yapmadan, makine sahibi olmadan çalıştırabilmektedir.

Sonuç olarak geliştirilen servisler artık daha güçlü ve iş görür hale gelmektedir. Yeni durumda, bu servislerin aranıp bulunması, bir iş akışı çerçevesinde kullanılması, bir sonuç üretmesi ve bu sonucun ihtiyaçlara yönelik kullanılması sağlanabilecektir. Burada önemli olan bu servislerin gerçek zamanlı olarak hem de yerel kaynaklardan bulunabilmesi ve kullanılabilmesi olacaktır.

Yine büyük verinin işlenebilmesi sayesinde, geçmişte sadece örneklem veri üzerinde yapılan analizler veya indirgenmiş kümeler üzerinde çalıştırılan algoritmalar artık orijinal veri üzerinde bir başka deyişle kaynağında oluştuğu şekliyle korunan ve defalarca üzerinde işlem yapılan bir hale dönüşmekte, bu da algoritmaların üreteceği sonuçların keskinliğini artırmakta ve ayrıca daha önce hiç yapamadığımız ölçeklerde çok sayıda veri kümesi ile bunların arasındaki ilişkilerle uğraşma imkanı tanımaktadır.

BÖLÜM 4

PROBLEMİN TANIMI VE VERİ YOĞUN WEB SERVİSLERİ İÇİN FİYATLAMA MODELİ

Bulut bilişimi ile sağlanacak olanaklardan mümkün olduğunca erken yaralanabilmek ve sistemleri bu ortama hızlı şekilde taşıyabilmek, etkin bir BT yönetimi için yakın dönemde en fazla üzerinde çalışılan konu olacaktır. Servis sağlayıcılar özelinde, bulut tarafından sağlanan olanaklarla, kaynak kullanımının öngörülmesi, sınıflandırılması ve yeniden fiyatlanması önem taşımaktadır. Bu fiyatlama, altyapıda kullanılan farklı kaynakların hangi oranda ve süre ile kullanıldığı, verilen servisin kalitesi ve istemcinin servise ulaşım öncelikleri veya abonelik anlaşmasına dayanacaktır.

Bu model içinde servis sağlayıcıların kendi servislerinin hangi durumlarda hangi maliyetlerle ve kar oranlarıyla çalışacaklarını önceden tahminlemesi ve bu tahminlemeyi istemcilerine servis çağrımı öncesinde sunabilmesi gerekmektedir. Bunun sağlanması için, servislerin maliyet hesaplarının dinamik olarak yapılabilmesi ve etmen temelli müzakere ile her bir çağrı öncesinde maliyet bazlı olarak fiyatlanabilmesi önerilmektedir. Dinamik fiyatlama yapılabilmesi için, hem servisin önceki maliyet bileşenlerine dayanan bir fiyat modeli olması; hem de anlık yük parametrelerine bağlı olarak değişen maliyetin belirlenebilmesi gerekmektedir.

Web Servisleri, veri yoğun servisler ve çalıştığı ortamda değişiklik yaratan servisler olarak iki temel kategoriye ayrılabilmektedir. Veri yoğun servislerin odak noktasında veride değişiklik yapmaktan çok veriyi sunmak vardır. Veriye erişim sağlayan, veriyi işleyerek farklı biçimlerde sunan bu servisler parametre veya algılayıcılardan elde edilen verilerin sunulması, iş zekâsı sistemlerinin veya uzman sistemlerin oluşturulması, farklı kaynaklardan gelen verinin işlenerek birleştirilmesi ve sunulması gibi çeşitli kullanım alanlarına sahiptir.

Veri yoğun web servisleri aynı zamanda sundukları içeriğin fiyatlanabilmesi için de temel bir altyapı sunabilecektir. Bu fiyatlama, bir tahminleme mekanizmasının servis kayıtlarına veya aynı servise ait simülasyona eklenmesi ile sağlanabilecektir. Bu sayede servisi kullanma niyetinde olan tüketici servis alma maliyetini önceden ve yüksek keskinlik derecesinde öğrenebilecektir. Sözkonusu maliyet analizi hem veriye uygun bir tahminlemeye hem de geçmiş servis kullanımlarına dair izleri karşılaştıran istatistik bilgilerine dayanmaktadır.

Oluşturulacak modelin kullanılmasıyla, dinamik şekilde olası parametrelere göre bir fiyat belirlenebilecek ve etmenlerin anlık servis çağrımı yapmaları için bir seçim parametresi olarak fiyat değeri kullanılabilecektir.

4.1. Veri Paylaşımı İçin Web Servisleri

Kurumlar arası entegrasyon ya da büyük veri üzerinde iş zekası algoritmalarından elde edilen konsolide çıktıların sunulması, web servislerinin bulut üzerinde kullanımı için gelişme gösteren alanlar olarak ortaya çıkmaktadır.

Doğru verinin doğru zamanda ve aynı zamanda sağlayıcı ile tüketici arasında uzlaşılmış bir fiyat üzerinden gerçekleştirilmesi bu çalışmanın odağını oluşturmaktadır.

Belli bir işlevi sağlayan servisler farklı ortamlarda, farklı servis sağlayıcılar tarafından sunulabilmektedir. Aynı işlevi sağlayan bu farklı servislerden en uygununu, ya da bir başka deyişle kalite ve maliyet olarak en avantajlısını seçmek problemin temelini oluşturmaktadır.

İşlev olarak en uygun servisi seçmek aynı zamanda yan faktörlere de bağlıdır. Bu faktörler maliyet, servisin güvenirliği ve farklı kullanıcıları tarafından değerlendirmesi, arkadaş tavsiyesi, vaad ettiği servis hızı olarak özetlenebilir. Tüm bu bileşenlerin bir arada değerlendirilmesi ile servis kullanım kararı oluşmaktadır. Bu kararın içindeki en etkin bileşen maliyet olarak öne çıkmaktadır. Maliyetin belirlenmesinde etkin olan unsurlar ise servis çağrımındaki parametreler, servis çağrımı sırasında oluşabilecek problemler ve istemcinin kimliğidir. Servisin çağrımı sırasında oluşabilecek bir hata veya zaman aşımı servis sağlayıcının oluşturacağı fiyat politikasına göre ortalama gelirini doğrudan etkileyecektir. Ayrıca istemci bazında bir abonelik durumu söz konusu ise, her bir servis çağrımında farklı marjlarda kar sistemi uygulanması mümkündür. Son olarak fiyatı

belirleyen en büyük unsur, servisin belli koşullarda gereken servisi sağlamak için bulut üzerinde kullanacağı kaynakların kendisine olan maliyetidir. Bu maliyeti hesaplamak servisin iç dinamikleriyle doğrudan ilgili bir durumdur.

Web Servislerinin bulut üzerinde kullanılması, servisi sağlayan taraf açısından maliyetlerin yönetilmesi anlamında kullandığın kadar öde modelini desteklemeyi gerekli kılmaktadır. Bu sebeple erişilen veri miktarı, ağ üzerinde iletilen veri miktarı, kullanılan işlemci gücü ve servis tüketicisinin kimliği üzerinden bir hesaplamaya ihtiyaç doğmaktadır.

4.2 Fiyat Bileşenleri

Servis kullanımı için fiyata ait bileşenler bulut tarafında kaynak kullanımından doğan maliyet, olası hata durumlarının zarara yol açmaması için hata payı, tüketiciye uygun uygulanacak kar payı olarak belirlenmektedir. Bunlardan tahminlenmesi en karmaşık olanı ve aynı zamanda fiyatın belirlenmesinde en büyük paya sahip olan bileşen kaynak kullanımından doğan maliyet olacaktır.

4.3 Maliyet Modellemesi

Bu tahminleme, servisler arasında ayrım yapma amacıyla kullanılabileceği gibi, yazılım etmenlerinin servisler için açık artırma veya eksiltme gibi metotlarla kullanıcı ihtiyacını karşılanmasının en iyilenmesini de mümkün kılacaktır.

Bu çalışmada servis kullanımının dinamik olarak fiyatlandırılması için aşağıda verilen parametreler ile olası maliyet modellenmiştir:

Tablo 4.1. Servis Maliyet Hesaplaması

Model	**İhtiyaç**	**Maliyet Hesaplaması**
Md	Veri depolama	Büyüklük(toplam)*Abonelik çarpanı*birim fiyat (depolama)
Mb	Bilgi işleme	Maliyet(makine)

Mti	Bulut içine veri transferi	Maliyet(içeri transfer)
Mtd	Bulut dışına veri transferi	Maliyet(dışarı transfer)
Mht	Beklenmeyen hata veya Zaman aşımı olasılığı	İstek zaman aşımına P olasılığı ile uğrarsa veya istemci için beklenmeyen bir sonuç ya da hata alınması durumunda farklı bir fiyatlama geçerli olacaktır.
MAb	İstemci kimliğine bağlı kar	Servis isteminde bulunan tarafın abonelik durumuna göre belirlenen kar parametresi, bu parametre ile hata veya zaman aşımı durumlarının dengelenmesi de sağlanmalıdır

Model parametreleri kullanılarak yaklaşık bir maliyet hesaplaması bir formülle sağlanabilecektir.

$$(1+ Mht) * (Md + Mb + Mti + Mtd) + Mab \qquad (4.1)$$

Formülde yer alan parametreler farklı bulut bilişim sağlayıcılarına göre değişiklik gösterebilecektir. Ayrıca her bir model parametresi farklı bir ağırlıklandırma ile servisin iç dinamiklerini yansıtmak üzere değiştirilebilecektir.

4.3.1 Servis Çağrımında Parametrelerin ve Çıktının Rolü

Servis çağrımında kullanılan parametreler gerçekleşecek hesaplama ve üretilecek çıktı üzerinde doğrudan etkiye sahiptir. Maliyet hesaplaması için çağrım parametreleri tahminleme analizinde temel birim olarak kullanılmaktadır.

4.3.2 Parametrelerin İşaretlenmesi

Parametreler, f_m fonksiyonu ile (içine ve örten) başka bir nümerik kümeye adreslenmektedir (5.2).

$$\forall i \; f_i(p_i) = c_i \qquad (4.2)$$

f_i fonksiyonu ile her bir parametre için karşılık gelen nümerik parametre sınıfı belirlenir. Bu fonksiyon içine ve örten bir fonsiyondur:

Örneğin p_1 = "Ali" olması durumunda c_1 = 1 şeklinde parametre sınıflandırması yapılır ve karşılık olarak bulunan bu nümerik değer tahminlemede kullanılır.

Tahminleme, belli bir servisin geçmişe ait kullanım detaylarının kaydedilmesi ve sonrasında bu verilerden yola çıkarak daha sonraki olası çağrımların maliyetinin ne kadar olacağını bulmayı amaçlar. Bu analiz kapsamında hem üzerinde çalışılan verinin büyüme trendi, hem de her bir girdinin hesaplama ve sonuçlar üzerindeki etkisi değerlendirmede göz önünde bulundurulur.

Sınıflandırma sonrası servis çağrım parametreleri, iki farklı tahminleme için kullanılır:

1. Çıktı kestirimi (output prediction), bu kestirim tüketiciye iletilecek veri miktarını da belirleyeceği için kaynak kullanım kestirimi için de kullanılacaktır.

2. Kaynak kullanımı kestirimi (CPU, bellek, disk, bant genişliği-ağ kullanımı)

4.3.2.1 Parametre İşaretlemede Doğruluk ve Tahminleme Üzerindeki Etkisi

Parametre sınıflarının doğru ölçekte ayarlanması tahminleme üzerinde önemli bir etkiye sahiptir. Farklı sınıflandırma algoritmaları değerlendirmeye alınmıştır:

- Naive Bayesian Classifier
- Support Vector Machines
- Feature Vector Construction

4.3.2.2 Geçmiş Veri Kullanımının Belirlenmesi (Pencereleme) ve Yerellik Prensibinin İncelenmesi

Pencereleme: Veri yoğun servislerin zaman içinde önceden tahminlenebilir bir desende davranış değiştirdiği varsayılmaktadır. Bu desen ne büyüklükteki bir log verisinin tahminlemeye baz oluşturacağını belirlemek üzere de kullanılabilir.

Veri penceresinin büyüklüğü algoritmanın eğitilmesi sırasında kullanılacağı sırada tahminleme kalitesini etkileyebilir. Aynı zamanda bu veri penceresinin içeriği log verisinin seçiciliği ile de ilişkilidir. Seçicilik ölçüsü kalitenin de önemli bir bileşenidir.

İstemci tarafından gönderilen girdi parametrelerinin sınıflandırılması bilişim teorisinde geniş yer bulan yerellik prensibine de tabi olabilir. Girdi parametrelerinin doğası yerellik derecesi konusunda fikir verebilir. Eğer yerellik yüksek değerlerde ölçülüyorsa, bu daha az disk erişimi anlamına gelir ve fiyata etkisi önemli olabilir.

4.4 Etmen Temelli Müzakere Protokolü

Probleme istemci tarafından bakıldığında, birden fazla servis sağlayıcı olduğunda, servis seçimi yapabilmek için, fiyat ve süre bilgisini edinebilmesi gerekir. Bunun için önerdiğimiz protokol, aşağıdaki gibidir:

Elimizdeki çıplak servisin (**S**), bir web servisi olduğunu varsayalım. Bu servisin çağırılması için gereki parametreler de $\mathbf{P_i}$ olsun. Müzakere protokolünde, istemci etmenin, bu servisi çağırmadan önce, fiyat, süre ve olası veri büyüklüğü bilgisini edinmesi gerekir. Bunu edinebilmesi için, giydirilmiş bir servis değerleme isteminde bulunması gerekir.

4.4.1 Servis Değerleme İstemi

Servis değerleme isteminde bulunması gereken parametreler, çıplak S servisinin çağırılması için gereken Pi parametrelerinin yanısıra, istemcinin kimlik ve/veya ödeme bilgilerinden oluşur. Eldeki model, abonelik tarzı bir model ise, burada iletilecek bilgi, kimlik bilgisinden ve önceden paylaşılmış bir güvenlik anahtarından ibaret olacaktır. Model anlık anonim kullanıma yönelik ise, sistem için anlamlı olan bir ödeme yönteminin bilgilerinin iletilmesi gerekli olacaktır. Bunu aşağıdaki şekilde özetleyebiliriz:

Tablo 4.2 Servis Değerleme İstemi Parametreleri

Servis Değerleme İstemi	
Çıplak servis parametreleri $\mathbf{P_i}$	Kimlik bilgisi ve güvenlik anahtarı **K** veya Ödeme bilgisi **O**

4.4.2 Servis Değerleme Yanıtı

Servis değerleme istemini alan servis, aşağıdaki işlemleri yapar:

1. *Kimlik veya ödeme bilgisi doğrulaması*: Sistem, eğer kimlik bilgisi sağlanmışsa, kimlik bilgisi ve güvenlik anahtarı kontolünü yapar. Modelde, kimlik bilgisi yerine ödeme bilgisi verilmiş ise, sistem bu aşamada ödeme bilgisini doğrular. Eğer kimlik kontrolü veya ödeme bilgisi kontrolü başarılı olmazsa, sistem ilgili durumu anlatan bir hata mesajıyla yanıt döner.
2. *Parametre doğrulaması*: Servis, çıplak servis parametrelerini, çıplak servis ile aynı şekilde çalışan parametre doğrulama işleminden geçirir. Bu adım önemlidir; çünkü maliyet hesabını yapan sistem gerçekte servisi çalıştırmayacaktır. Nihai olarak servisin kullanamayacağı parametreler için de bir maliyet çıkarımı mümkün olabilir. Bu durumda, maliyet hesabı yapılmış bir servis çağrısı, sonradan başarısız olacaktır. Doğru olan, bunun bu aşamada belirlenmesidir.

   Eğer servis parametresi doğrulama adımı başarısız olursa, servis, istemciye, servis parametrelerindeki hatayı bildiren bir yanıt döner. Bu durumda, herhangi bir maliyet tahminlemesi yapılmaz ve yanıt olarak dönülmez.
3. *Tahminleme*: Ödeme/kimlik doğrulaması ve parametre doğrulama adımları başarılı olarak geçildikten sonra, eldeki tahminleme modeli kullanılarak, tahminlemede bulunulur. Tahminleme sonucunda, üç önemli parametre vardır. Tahmini çalışma süresi, tahmini veri miktarı ve tahmini maliyet. Bunlardan başka, bir istem anahtarı da oluşturulup, yanıt içinde dönülür. Yanıtta, ayrıca ilgili tahminin geçerlilik süresi de bulunur.

Tablo 4.3. Servis Değerleme Yanıtı Parametreleri.

Servis Değerleme Yanıtı	
Parametre	**Açıklama**
Tahmini çalışma süresi **T**	Sorgu yapıldığında yanıtın gerçek zamanda oluşturulma süresi
Tahmini veri miktarı **D**	Sorgu yapıldığında, yanıtın boyutu. Eldeki probleme göre byte cinsinden veya veri satırı cinsinden olabilir.
Tahmini maliyet **C**	Sorgu yapıldığında, maliyeti bu kadar olacaktır.
İstem anahtarı **A**	Sorgu, bu anahtarla beraber yapıldığında, bu maliyetle yapılacaktır.
İstem anahtarı bitim zamanı **t**	Verilen maliyetler, sistem yüküne göre zamanla değiştiği için, her istem anahtarı belli bir süre geçerlidir. Bu eldeki sistemin değişim hızına göre, saniyelerden saatlere kadar olabilir.

Sistem, servis değerleme yanıtını döndükten sonra, **A** istem anahtarını ve buna karşılık gelen **Pi** çıplak servis parametrelerini, **t** zamanına kadar kayıtlı tutup hatırlamak durumundadır. (Veri yoğun sistemlerde, bunun oluşturacağı ek yük ve maliyet gözardı edilebilir.)

4.4.3 Servis İstemi

Servis değerleme yanıtını alan istemci, eğer sonuç başarılıysa, bu aşamada, istemini gerçekleştirebilir. İstemi gerçekleştirmek için, **t** anına kadar, **A** istem anahtarı ve **P_i** çıplak servis parametreleriyle, servis isteminde bulunabilir.

Sistem tarafından servis istemi alındığında, öncelikle verilen **A** istem anahtarıyla, depolanmış olan çıplak servis parametreleri **P_i** depolandığı yerden çağırılır. Çıplak servis **S** bu parametrelerle çağırılır, ve çıplak servisin sonucu dönülür.

Her ne kadar parametreler doğrulamadan geçirilmiş olsa da, bu aşamada hata oluşması mümkündür. Hata halinde, çıplak servisin hata sonucu dönülür.

Herhangi bir hata oluşmadan çağrının tamamlanması halinde, istemciye sonuç dönüldükten sonra, önceden üretilmiş olan maliyet **C** kadar ücretlendirilir.

BÖLÜM 5

MAKİNE ÖĞRENMESİ VE DENEY ORTAMI

Burada amacımız, bir servisin, gerçek çağrıyı yapmadan, o servis çağrısının maliyetini hesaplayabileceği, öğrenen bir program yaratmaktır. Yani, "servis değerlemesi" olarak bahsettiğimiz işlemin, deneysel ortamda da gerçeklemek için, bir ortam hazırlamak ve bunun sonuçlarını analiz etmektir.

5.1 Makine Öğrenmesi

Tanım: Deneyim E olarak, tanımlı işler T olarak, performans değeri de P olarak tanımlı olsun. Bir bilgisayar programına öğrenen program denebilmesi için, T işleri gerçekleştirilirken performans değeri P'nin, deneyim E ile artıyor olması gerekmektedir.

5.2 Deney Ortamı

Deney ortamı için, çok yüksek performanslı olmayan (ki bu durumda performans ölçümü anlam taşımayacaktır) ancak günün sunucu standartlarına da uygun olan bir makineyi kiralayarak kullanılmıştır. Bu sebepten, deney ortamı, i7 işlemciye sahip bir makinede 16 GB bellek ve 256 GB SSD disk ile oluşturulmuştur. Makinanın üzerinde, web servisi için Python programlama dili ve Django çatısı kullanılmıştır. Veri tabanı olarak da, Postgresql kullanlmıştır. Kullanılan uygulama ve özellikleri aşağıda detaylandırılmaktadır.

5.2.1 WEKA

Makine öğrenmesi yöntemlerinin uygulanmasında WEKA adlı araç kullanılmıştır. Açılımı "Waikato Environment for Knowledge Analysis (WEKA)" Java dilinde

geliştirilmiş bir veri madenciliği yazılımıdır. İçinde, SVM, NN ve LR gibi çeşitli makine öğrenme algoritmalarına ait işlevler barındırır [36].

Weka makine öğrenme yöntemlerini bir arada toplayan bir projedir, kullanıcı dostu grafik ara yüzü ile çeşitli veri madenciliği yöntemlerini sağlar. Bu tezde, özellikle Weka Bayes ve SVM fonksiyonları kullanılmıştır. Veriler bir veritabanında saklanır ve WEKA'ya özel Özellik İlişkileri Dosyası (Arff) oluşturularak sınıflandırma işlemnleri gerçekleştirilir. Arff biçiminde temel iki bölüm vardır, birincisi ilişki ismi, nitelikleri ve öznitelik değeri tiplerinin ifade edildiği başlık bölümü, ikincisi ise birer satır şeklinde öznitelik değerlerini tutan veri bölümüdür.

WEKA seçilen öğrenme algoritması için çapraz doğrulama sağlar. Veriler rastgele ayrılmıştır ve N adet alt-grup içinden, N-1 alt-grup (Bu çalışmada %50 olarak seçilmiştir) eğitim verisi olarak kullanılır, diğer alt grup ise test etmek için kullanılır. Bu süreç n defa çalıştırılır. N farklı süreçten geçerli çıktısı olanların ortalaması alınır.

5.2.1.1 ARFF Dosya Yapısı

ARFF dosya yapısı, Weka'ya özel geliştirilmiş bir dosya yapısıdır ve esas olarak bir metin dosyasıdır. İnsan tarafından okunabilir ve yazılabilir olması önemli bir özelliğidir. Dosyanın ilk satırı, dosyadaki ilişki tipini (relation) gösterir. İkinci satırdan itibaren, dosyanın içeriğinde bulunan veri kümesindeki özellikler (attributes) her biri bir satırda olmak üzere belirtilir. Özelliklerin hemen ardından, her bir veri kümesi bir satıra gelecek şekilde veriler yer alır. Her bir satırdaki veri kümesi içindeki her bir özellik, virgül ile birbirinden ayrılır.

5.2.2 Kullanılan Veriler ve Programlama Detayları

Bu bölümde, deney için kullanılmış olan veriler, bunların kaynağı ve deney için yapılan programlama çalışmasının teknik ayrıntıları anlatılmaktadır.

5.2.2.1 Deney İçin Kullanılan Veri Kaynağı

Deney için kullanılan veriler, artık aktif olmayan "İşler Burada" (www.islerburada.com) adlı sitenin geçmiş verilerinden alınmıştır. Bu site, 2009-2010

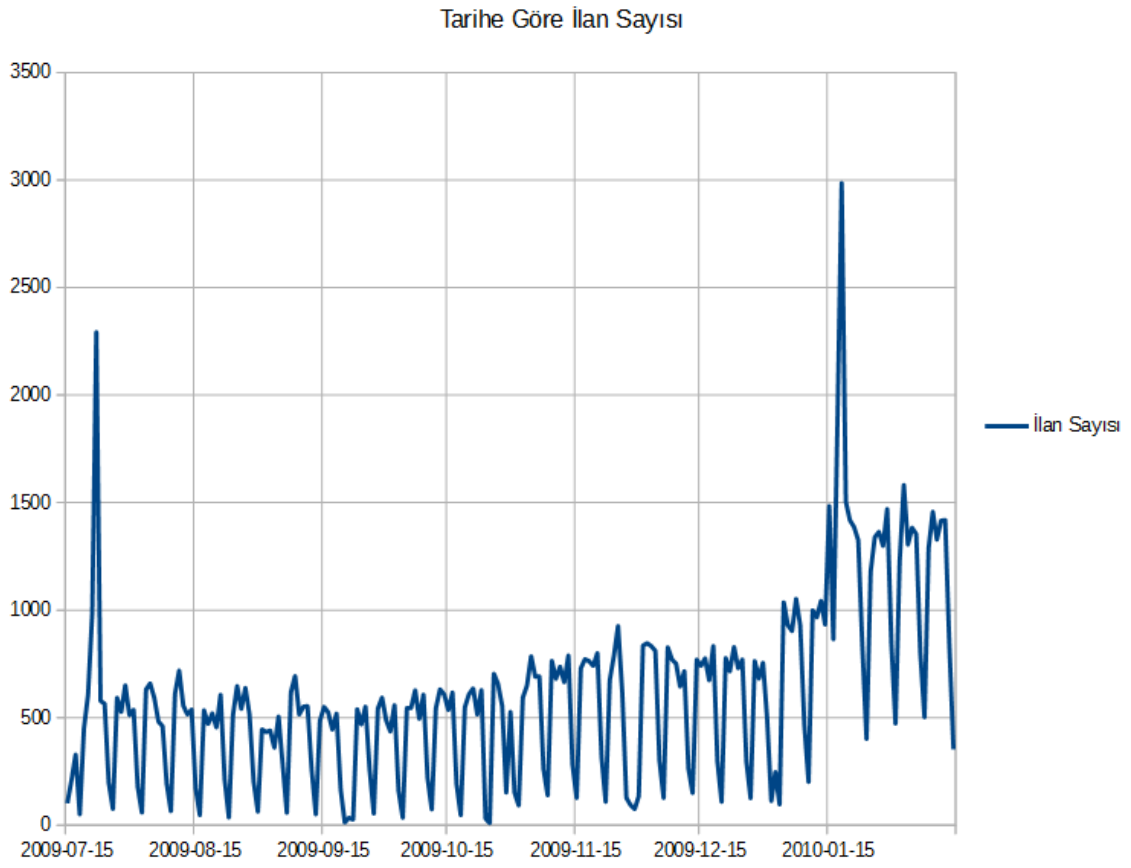
yıllarında çalışmış bir sitedir. Site, ikinci dereceden bir iş arama sitesidir. İş ilanı yayınlanan kariyer sitelerini gezinerek, tüm iş ilanlarını bir veritabanında toplar. Sonra, bu toplanan verileri ayrıştırarak, her iş ilanını analiz eder. Sonrasında, bu iş ilanlarını indeksleyerek, site kullanıcılarının aramasına sunar.

Veri tabanında bulunan 130.000 kadar ilan içeriğinden 10 kadarı aşağıda örnek olarak verilmiştir. Aslında, ilan içeriklerinin ağırlıklı kısmı "İş Tanımı" ve "Aranan Nitelikler" kısımlarındadır, ancak hem uzunluk sebebiyle hem de analiz esnasında ayrıştırma gücünün yüksek bulunmaması sebebiyle verilerin bu kısımları atlanmıştır.

Tablo 5.1. İş İlanı Örnekleri.

Tarih	**Şirket Adı**	**Pozisyon**	**Şehir**
2009-08-03	UNIVERSAL HOSPITALS GROUP	CERRAHİ SERVİS HEMŞİRESİ	İstanbul
2009-08-03	Metropol İnsan Kaynakları	BÖLGE MÜDÜR YARDIMCISI/SİGORTA/ ADANA	Adana
2009-08-03	ACCOR Otelleri Türkiye	Demi Chef de Partie	Gaziantep
2009-08-03	İnproda Danışmanlık	SATIŞ UZMANI (BAKIM VE DESTEK HİZMETLERİ)	Ankara
2009-08-03	RGS GROUP	Depo Elemanı	İstanbul
2009-08-03	Ali Raif İlaç San. A.Ş.	Tıbbi Satış Mümessili	Samsun
2009-08-03	UNIVERSAL HOSPITALS GROUP	Ebe Hemşire	İstanbul
2009-08-03	Erol Hukuk Bürosu	SEKRETER	İstanbul Anadolu İstanbul Avrupa
2009-08-03	egoks kimya	satış danışmanı	Rize
2009-08-03	ÜSTAY - Ç.OVA - CAC - JV LİBYA	Muhasebe ve Mali İşler Sorumluları	Libya

Tarihe göre ilan sayısı değişimi, aşağıdaki şekilde görülmektedir. Çıkan iş ilanı sayıları, normal bir haftalık salınım göstermektedir; bunun sebebi ise hafta sonları ilan sayılarının düşmesidir. Bu periyodik değişim gözardı edildiğinde, ilan sayılarının zaman içinde büyük bir değişim göstermediği gözlemlenebilir. İncelenen aralığın son kısmındaki yükselme ise, tamamen daha çok sitenin indesklenmeye başlamış olması ile ilgili bir durumdur.



Şekil 5.1. Tarihe göre günlük ilan sayısı değişimi.

İş ilanı sayısının, şehirlere göre kırılımı da, dikkate değer bir durumdur. İstanbul'da diğer ilanlara göre ezici miktarda iş ilanı verilmektedir. En çok iş ilanı verilen şehirlerin kırılımı aşağıdaki tabloda gösterilmiştir:

Tablo 5.2. En çok iş ilanı çıkan şehirler.

Şehir	**İş İlanı Sayısı**
İstanbul	68042
Ankara	16735
İzmir	5629
Bursa	4192
Antalya	3261
Akdeniz	2063
Kocaeli	1916
Adana	2135
Tekirdağ	1312
Konya	1115
Gaziantep	1014
Kayseri	961
Muğla	726
Denizli	627
Libya	617
Samsun	536
Eskişehir	506
Sakarya	452
Manisa	404
Erzurum	350

Diyarbakır	344
Balıkesir	342
Aydın	335
Malatya	324

Sözkonusu toplam ilan sayısının 130.000 kadar olduğu düşünülürse, İstanbul ilan sayısının yarısından fazlasına denk düşmektedir. Ardından gelen bir kaç büyük il de dahil edildiğinde, toplam ilanların %80'i civarına varılmaktadır. Arada, şehir dışında adına yoğunluklu ilan verilen iki yer daha dikkat çekmektedir. "Akdeniz" turizm ile ilgili işler dolayısıyla öne çıkmıştır. "Libya" ise ilgili yıllarda inşaat işleriyle öne çıktığından bu listede kendisine yer bulmuştur.

5.2.2.2 Programlama Ortamı

Deney amaçları için, basit ve pratik olması amacıyla, programlama dili olarak Python seçilmiştir. Python, nesne yönelimli, basit, modüler ve yüksek seviyeli bir programlama dilidir. Python dili tüm popüler platformları destekler.

Gereken web servisinin yazılabilmesi için, Django çatısı kullanılmıştır. Django, Python programlama dili ile yazılmış, MVC (Model-View-Controller) desenini destekleyen bir web programlama çatısıdır.

5.2.2.3 Web Servisi

Web servisi, web üzerinden gelecek istemi, veri tabanını sorgulayacak şekilde düşünülmüş ve hazırlanmıştır. Aşağıdaki kod, web üzerinden alınan istemin, veri tabanı sorgusuna dönüştürülüp, sonuçların üretilmesi işini yapmaktadır.

Kod Örneği 5.1. İstemi sorguya dönüştürme

```
1 from django.shortcuts import  render_to_response
2 from search.forms import SearchForm, AdvancedSearchForm
3 from django.db.models import Q
4 import re
5 from search.models import AdContent
```

```
 6 from django.template.context import RequestContext
 7 from measurement.decorators import measurer
 8
 9 import logging
10
11 logger = logging.getLogger("django")
12
13 # Gelen istemi karşılar. Eğer istem metodu GET ise, sorgu
14 # formunu gösterir. POST ise, sorgulamayı yapar ve sonuçları
15 # gösterir
16
17 def home(request):
18     search_str = None
19     found_entries = []
20
21     if request.method == "POST" and ('search_box' in request.POST) \
22             and request.POST['search_box'].strip():
23         search_str = request.POST['search_box']
24
25         found_entries = _simple_search_query(search_str)
26
27     form = SearchForm()
28
29     return render_to_response("index.html", {'search_form': form,
30                                              'search_str': search_str,
31                                              'ad_contents': found_entries },
32                                               context_instance=RequestContext(request))
33
34 # İlan içeriklerinden, verilen parametrelere göre arama yapar
35 # ve sonuçları döner. home() forksiyonu tarafından kullanılır.
36
37 @measurer
38 def _simple_search_query(search_str):
39
40     query = _get_query(search_str, ['company_name', 'job_position', 'cities', 'qualification', 'sector_name'])
41
```

```
42    return AdContent.objects.filter(query).order_by('-created')
43
44 # Gelen sorguyu, veri tabanı üzerine çaıştırabilecek hale
45 # getirir ve döner.
46
47 def _get_query(query_string, search_fields):
48    query = None
49    terms = normalize_query(query_string)
50
51    for term in terms:
52       or_query = None # Query to search for a given term in each field
53       for field_name in search_fields:
54          q = Q(**{"%s__icontains" % field_name: term})
55          if or_query is None:
56             or_query = q
57          else:
58             or_query = or_query | q
59       if query is None:
60          query = or_query
61       else:
62          query = query & or_query
63
64    return query
```

Diğer çözümlenmesi gereken sorun, her bir sorgu esnasında, kullanılan kaynakların ölçümlenmesi sorunudur. Standard web sunucularının logları, böyle bir ölçüm için yeterli bilgiyi vermezler. Normal şartlar altında gelen istemin içeriğini de depolamazlar; bunu yapmak mümkündür, ancak elde edilecek sonuç, gelen sorgunun ne kadar zamanda tamamlandığından öte bilgi içermez. Bunu çözümlemek için, Python ile sorgulama yapan fonksiyonu sarmalayarak (wrap) sistemden harcanan kaynakları almak mümkündür. Yukarıdaki kod parçasında "@measurer" ifadesi, ilgili fonksiyonu sarmalayan fonksiyonu ifade etmektedir. Bu sarmalayan fonksiyonun tanımı aşağıda verilmiştir:

Kod Örneği 5.2. Kaynak ölçümleme ve kayıt tutma

```
1 import time
2 import psutil
3 from functools import wraps
4
5 import logging
6 import json
7 from measurement.models import MeasurementLog
8 from django.forms.models import model_to_dict
9 from datetime import datetime
10
11 logger = logging.getLogger("django")
12
13 # measurer, bir decorator tanımıdır. Başına @measurer konan
14 # fonksiyon çağırıldığında, prosesin, o fonksiyon çalışırken
15 # kullandığı kaynakları ölçer ve bu sonuçları döner.
16
17 def measurer(function):
18     @wraps(function)
19     def timer_function(*args, **kwargs):
20         logger.info("TIMER called")
21         process = psutil.Process()
22
23         cpu_info_start = process.cpu_times()
24         io_counters_start = process.io_counters()
25         memory_info_start = process.memory_info_ex()
26         num_ctx_switches_start = process.num_ctx_switches()
27         t_start = time.time()
28
29         result = function(*args, **kwargs)
30
31         t_end = time.time()
32         cpu_info_end = process.cpu_times()
33         io_counters_end = process.io_counters()
34         memory_info_end = process.memory_info_ex()
```

```
35        num_ctx_switches_end = process.num_ctx_switches()
36
37        start_dict = __convert_to_info_dict(cpu_info_start, io_counters_start, memory_info_start, num_ctx_switches_start, t_start)
38        end_dict = __convert_to_info_dict(cpu_info_end, io_counters_end, memory_info_end, num_ctx_switches_end, t_end)
39
40        info = dict()
41        info["start"] = start_dict
42        info["end"] = end_dict
43        info["result_count"] = len(result)
44
45        if type(args[0]) is str:
46            info["request_params"] = {"search_str": args[0]}
47        else:
48            info["request_params"] = model_to_dict(args[0])
49
50            if args[1]:
51                info["request_params"]["begin_date"] = args[1]
52            if args[2]:
53                info["request_params"]["end_date"] = args[2]
54
55        info_json = json.dumps(info)
56
57        measurement_log = MeasurementLog()
58        measurement_log.log = info_json
59        measurement_log.function_name = function.__name__
60        measurement_log.result_count = len(result)
61        measurement_log.save()
62
63        logger.info("Log %s: %s" % (function.__name__, info))
64        return result
65    return timer_function
66
67
68 # Bu bir yardımcı fonksiyondur. Kullanılan kaynakları, bir "dictionary"
69 # içine toplar ve döner.
70
```

```
71 def __convert_to_info_dict(cpu_info, io_counters, memory_info, num_ctx_switches, t):
72     info_dict = dict()
73     info_dict["cpu_info"] = cpu_info._asdict()
74     info_dict["io_counters"] = io_counters._asdict()
75     info_dict["memory_info"] = memory_info._asdict()
76     info_dict["num_ctx_switches"] = num_ctx_switches._asdict()
77     info_dict["time"] = datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')
78     info_dict["long_time"] = time.time()
79
80     return info_dict
```

Burada tanımlanan fonksiyon sayesinde, sarmalanan fonksiyonun tükettiği kaynakları ölçümlemek mümkün olmaktadır. Burada, Python'un psutil kütüphanesi bu ölçümler için kullanılmıştır. Ölçümleme yapabilmek için, öncelikle psutil.Process() çağrısı yapılarak, o anda çalışan proses (ki bu web servisi veren prosestir) ölçümlerini alabileceğimiz obje referansı elde edilir. Bundan sonra, bu obje üzerinden, ölçümde kullanılacak değerler, sarmalanan fonksiyonun öncesinde ve sonrasında çağırılır. Bunun dışında, daha basit olarak zamanlama bilgisi için time.time() çağrısı da yapılır.

process.cpu_times() çağrısı, ilgili prosesin ve alt proseslerinin kendileri ve sistem çağrıları esnasında kullandıkları işlemci sürelerini verir. Bu çağrının tipik bir sonucunun gösterimi aşağıdaki gibidir:

pcputimes(user=0.04, system=0.02, children_user=0.0, children_system=0.0)

process.io_counters() çağrısı, ilgili prosesin okuma ve yazma işlem sayılarını ve toplam byte cinsinden okuma yazma miktarını verir. Bu çağrının tipik bir sonucunun gösterimi aşağıdaki gibidir:

pio(read_count=609, write_count=173, read_bytes=0, write_bytes=0)

process.memory_info_ex() çağrısı, ilgili prosesin hafıza kullanım değerlerini verir. Bu çağrının tipik bir sonucunun gösterimi aşağıdaki gibidir:

pmem(rss=14213120, vms=74743808, shared=5246976, text=2867200, lib=0, data=8630272, dirty=0)

process.num_ctx_switches() çağrısı, ilgili prosesin istemli ve istemsiz bağlam geçiş sayılarını verir. Bu çağrının tipik bir sonucu aşağıdaki gibidir:

pctxsw(voluntary=194, involuntary=58)

5.2.3 Deneye ait Detaylar ve Ortam Kurulumu

Bu bölümde adım adım deneyin hazırlanışına ait detaylar anlatılmaktadır.

Bu tez kapsamındaki yaklaşımımız beş ana adımdan oluşmaktadır; veri toplama, verinin ön analizi, özellik seçimi, özellik vektörü oluşturulması ve sınıflandırma. Temel ayrıntılar aşağıdaki gibidir:

1. Veri toplama analiz için gereken verinin alınarak işlenmeye uygun olanların alınmasıdır. Bu aşama log verilerinin örnek uygulamalardan alınmasını ve WEKA girdi dosyalarına (arff tipinde)çevrilmesini içerir.
2. Ön işleme: Verinin anlamlı en küçük parçalarının ve önemli özelliklerinin belirlenmesi ve sınıflandırılmasını içerir.
3. Özellik seçimi: değişik bütün olası özellikler belirlendikten sonra analiz kapsamında etkili olabilecek (bizim durumumuzda fiyata etki edebilecek) özelliklerin etki değerine uygun şekilde seçilmesini içerir. Weighted Log Likelihood Ratio (WLLR) skorlaması kullanılmaktadır.
4. Özellikler tf ve tf-idf metotları kullanılarak ağırlıklandırılır.
5. Sınıflandırma ise eğitim ve test aşamalarının tümüdür. Farklı sınıflandırma algoritmaları kullanılır, bu sayede en iyilenmiş parametre kalibrasyonuna ulaşılmaya çalışılır.

5.2.3.1 Veri Kümesi ve Veri Toplama

Veri toplama amacıyla, iş ilanı sitelerini indeksleyen (artık kullanımda olmayan) bir web sitesinin (www.islerburada.com) verileri kullanılarak yeni bir örnek uygulama yaratılmıştır. Bir iş ilanı, şirket adı, pozisyon adı, istenen nitelikler ve iş tanımı bilgilerini içerir. Aynı zamanda, ilanın yayına çıkış tarihi de kayıt altındadır.

5.2.3.2 Parametre Seçimi ve Tahminleme Gücü

Orijinal web sitesinde, iş ilanlarının her alanı üzerinden arama yapmak mümkündür. Diğer bir deyişle, sorgulamada tüm alanlar üzerinden sorgu yapmak mümkündür. Ancak, her alanı sorgu maliyetini tahminlerken kullanmak, tahmini aynı oranda iyileştirmez.

Bir parametre, sınıflandırmaya tabi tutulduğunda, ayrıldığı sınıfların dağılımına göre bir enformasyon içeriği oluşturur. Pek çok değer alan bir **P** parametresi, herhangi bir yöntemle, **N** elemanlı bir kümeye adreslenirse, bu değerlerin sıklıklarına göre, ilgili adresleme kümesinde bir dağılım oluşturur. Bu adresleme kümesinin elemanlarına $\mathbf{A_i}$ dersek, oluşan dağılımın enformasyon içeriği şu formülle belirlenir:

$$I = \sum_i \; p(A_i)\, log_2(1/p(A_i)) \qquad (5.1)$$

Burada, p(**Ai**), adresleme kümesinde oluşan sıklık, dolayısıyla olasılık dağılımıdır. Birbirine yakın sıklıklar, daha yüksek bilgi içeriği taşır. Dağılımın "dengesiz" olması ise, bilgi içeriğinin düşük olması sonucunu verir.

Diğer bir önemli değer, değişimin maliyete olan etkisidir. Adresleme kümesinde oluşan ortalama maliyet farklarının, orantılı olarak yüksek çıkması durumunda, bilgi içeriği düşük olmasına rağmen, ortalamaya etkisi yüksek olabilir.

Bu şekilde değerlendirildiğinde, aşağıdaki durumlar gözlenmiştir:

Şirket adı parametresi, neredeyse hiç sorgulanmamıştır. Sorgulandığında ise, maliyete olan etkisi, şirket ismine göre olan dağılım çok dengeli olduğundan, maliyete etkisinin yüksek olmadığı gözlenmiştir. Bu sebepten, şirket adı parametresi dışarıda bırakılmıştır.

İstenen nitelikler parametresi, uzun bir metinden oluştuğundan, sorgulanırken başına ve sonuna mutlaka joker karakterler eklenerek sorgulanmıştır. Bu yüzden, sınıflandırmada bilgi içeren bir sınıflandırma yapmak mümkün olmamıştır. Deneysel sonuçlarda da, herhangi bir şekilde modele eklenerek bir iyileşme elde edilemediğinden, çalışma dışı bırakılmıştır.

İş tanımının parametresinin durumu, istenen nitelikler ile neredeyse aynıdır. Bu parametre de uzun bir metin alanı sorguladığından, tahminleme gücü olarak düşük bulumuş, dolayısıyla çalışma dışında bırakılmıştır.

Aşağıdaki parametrelerin ise, tahminleme gücünün yüksek olduğu görüldüğünden, bu parametreler kullanılarak iş ilanı arama işlevini yerine getiren bir web servisi oluşturulmuştur:

I. Arama başlangıç tarihi

II. Arama bitiş tarihi

III. Pozisyon adı (joker karakter içerebilir veya boş olabilir)

IV. Şehir (boş olabilir veya bir şehir seçilebilir)

Oluşturulan bu servis, verilerin alındığı web sitesinin kullanım loglarından çıkartılan bir sorgu kümesi kullanılarak çağırılmıştır. Asıl servis fonksiyonu, bir loglama fonksiyonu içinde çağırılmış, bu sayede her servis çağrısı için, aşağıdaki bilgiler toplanmıştır:

I. Sorgu esnasında makina üzerindeki işlemci yükü

II. Sorgu sonucunda dönülen iş ilanı sayısı

III. Toplam gönderilen veri miktarı

IV. Sorgu için makina tarafından yapılan toplam giriş / çıkış miktarı

V. Sorgu için kullanılan işlemci süresi

VI. Sorgu için kullanılan toplam gerçek zaman

5.2.3.3 Sınıflandırma Algoritması

Sınıflandırma algoritması, bu araştırmanın önemli alanlarından biridir. Sınıflandırma, tahminleme algoritmalarının beklenen bir performans aralığında çalışmasının omurgasını oluşturur. Bu çalışmada, netlik ve hataları azaltmak açısından, manuel yönlendirilmiş bir sınıflandırma algoritması kullanılmıştır. Burada, sınıflandırma bir alan uzmanı tarafından tanımlanan kurallar ile, parametrelerin her değeri tahminleme algoritmasında kullanılacak bir değerle eşleştirir.

Sınıflandırma algoritmasının, sonraki deneylerde otomatik hale getirilmesi mümkündür. Bunun için, yapay sinir ağları, bulanık yaklaşımlar gibi araçlar kullanılabilir.

5.2.3.4 Özellik Sınıfları (Attributes)

Bu çalışma kapsamında özellik sınıfları aşağıdaki şekilde manuel olarak belirlenmiş ve ön işlemden geçirilmiştir.

Yapılan gözlemlerde, sorgu başlangıç ve bitiş tarihlerinden bağımsız olarak sadece sogulanan gün sayısının bir tahminleme değeri taşıdığı anlaşıldığından, başlangıç ve bitiş tarihi parametreleri, tek bir "gün sayısı" parametresine adreslenmiştir. Günlük iş ilanı sayısı fazla değişim göstermediğinden, bu yaklaşım genel geçerliliğe sahiptir.

Pozisyon adı parametresi ise, aşağıdaki şekilde dört farklı değer içeren bir kümeye adreslenmiştir:

I. 0: Boş (bu alan arama için doldurulmamıştır)
II. 1: % karakteri (joker karakter) içermez
III. 2: % karakteri ile biter
IV. 3: Son karakter dışında bir yerde % karakteri içerir

Şehir parametresi, aşağıdaki şekilde üç değer içeren bir kümeye adreslenmiştir:

I. 0: Boş (şehir seçilmemiştir)
II. 1: Büyük şehir seçişmiştir (en çok ilan çıkan dört şehirden biri)
III. 2: Başka bir şehir seçilmiştir

Bunlara ek olarak, servis veren makina üzerindeki işlemci yükü de sayısal bir değer olarak parametrelere dahil edilmiştir. Bu değer aslında eldeki sorgudan bağımsızdır; ancak sorgu tahminlemeden (kısa) bir süre sonra yapıldığında, sorgunun tamamlanma süresinin belirlenmesinde önem taşır. Ayrıca, bu çalışmada oluşturulan tahminleme modelinin dinamik özelliğini ortaya koyması açısından da ortaya konmuştur. Yani, aynı

sorgu, farklı zamanlarda yapıldığında farklı sürelerde ve farklı maliyetlerde sonuç verebilir.

5.2.3.5 Analiz Yöntemi

Tahminleme algoritmasının dört adet parametresi vardır. Bunlardan iki tanesi "alan uzmanı" tarafından manuel olarak bir değer kümesine adreslenerek sınıflandırılmış parametreler, diğer ikisi ise sayısal olarak kullanılan parametrelerdir. Bu parametreler Tablo 5.3'te gösterilmiştir.

Tablo 5.3. Tahminleme algoritması girdi parametreleri ve tipleri.

Tahminleme Algoritması Girdi Parametreleri	
Parametre Adı	**Tipi**
Pozisyon adı	Sınıflandırılmış
Şehir	Sınıflandırılmış
Gün sayısı	Sayısal
İşlemci yükü	Sayısal

Tahminleme sonucu olarak da, dört adet değer elde edilecektir. Bunların tamamı, sayısal değerlerdir. Tahminleme sonuç değerleri Tablo 5.4'te gösterilmiştir.

Tablo 5.4. Tahminleme algoritması çıktı değerleri ve tipleri.

Tahminleme Algoritması Çıktı Değerleri	
Değer Adı	**Tipi**
Toplam süre	Sayısal
Toplam işlemci süresi	Sayısal
Giriş/çıkış miktarı	Sayısal
Toplam transfer edilen veri	Sayısal

Eldeki girdi parametrelerindeki özel durum, sınıflandırma yapılmış değerler ile sayısal değerlerin bir arada bulunuyor olmasıdır. Yani, eldeki problem sadece sınıflandırma problemi ya da sadece bir regresyon / korealasyon problemi değildir. Dolayısıyla, bu çalışmada, bu iki tip tahminleme disiplinini bir araya gelmiş olmaktadır.

Tahminleme modeli için, WEKA üzerinde M5P algoritması kullanılmıştır. Bu algoritma, aşağıda anlatıldığı şekilde çalışır:

Öncelikle, sınıflandırılmış değerler üzerinden, bir karar ağacı oluştulmuş, sınıflandırılmış değerlerin kombinasyonları için sayısal parametreler üzerinden lineer modeller oluşturulmuştur. Sonra, parametreler değiştirilerek, karar ağacı budanmış, benzer lineer modeller bir araya toplanarak hem sınfılandırılmış değerleri hem de sayısal değerleri kullanabilen tahminleme modeli oluşturulmuştur.

5.2.3.6 Eğitim Kümesi

Toplanan verinin %50'lik bölümü sınıflandırma algoritmalarının eğitimi amacıyla kullanılmıştır. Verinin geri kalan kısmı, tahminlemelerin doğrulanması ve farklı gruplama ve tahminleme algoritmalarının değerlendirilmesi için kullanılmıştır.

Kullanılan arff dosyasından kısa bir örnek aşağıda verilmiştir:

```
@relation 'log'
@attribute start numeric
@attribute end numeric
@attribute days numeric
@attribute job class
@attribute city class
@attribute cpuload numeric
@attribute rows numeric
@attribute data numeric
@attribute io numeric
@attribute cpu numeric
@attribute time numeric
@data
171,178,8,1,1,0.338,58,79788,81117,14.853,31.053
23,24,2,3,2,0.281,8,12300,13350,9.221,29.359
```

3,10,8,3,2,0.203,10,10745,11725,28.392,31.651

34,60,27,2,0,0.051,282,335993,350421,34.64,51.692

91,98,8,1,2,0.496,7,10538,12529,0.607,19.947

6,21,16,3,1,0.274,194,254149,260940,107.767,136.981

119,202,84,2,3,0.324,763,952450,952810,831.809,969.256

5.2.3.7 Test Kümesi

Toplanan verinin kalan %50'lik kısmı ise test amaçlı olarak kullanılmıştır. Bu, toplam 97.200 veri satırının, 48.600 adedi anlamına gelmektedir.

BÖLÜM 6

SONUÇLAR VE SONUÇLARIN ANALİZİ

Sonuç olarak, adı geçen dört adet çıktı için, sonuçlar elde edilmiştir. Bu dört adet çıktının her biri için elde edilmiş sonuçlar aşağıdaki bölümlerde verilmiştir. Her bir çıktı için örnek tahminleme sonuçları ve daha sonra da istatistiksel sonuçlar sunulmuştur. Son bölümde de, sonuçların istatiskleri toplu halde sunulmuş ve yorumlanmıştır.

Sonuçlarda, hem bir karar ağacı yapısı, bu ağaçların yapraklarında da lineer modeller bulunmaktadır. Bunun sebebi, tahminleme girdilerinde, hem sınıfsal verilerin, hem de nümerik verilerin bulunmasıdır. Sınıflandırılmış veriler, sayısal olarak anlam taşımazlar; bunlar karar ağacında hangi yaprağın seçileceğini etkilerler. Sayısal veriler ise, lineer model içinde hesaba katılırlar. Kullanılacak sistemlerde, genel durumda böyle olacağı kabul edilebileceği için, her türlü veri ile işe yarayacak bir yöntem kullanılmıştır.

6.1 Toplam Süre Tahmini Sonuçları

Toplam süre tahmini için elde edilen sonuçlardan ilk yirmi beş adedi aşağıdaki tabloda verilmiştir.

Tablo 6.1. Toplam süre tahmini ve gerçekleşen
sürelerin karşılaştırıldığı örnek sonuçlar.

Veri Noktası	Gerçek Süre	Tahminlenen Süre	Hata
1	24.859	21.693	-%13
2	28.059	30.276	%8
3	13.083	14.616	%12
4	60.453	55.914	-%8
5	106.081	103.911	-%2
6	547.047	576.017	%5
7	42.741	46.092	%8
8	679.424	669.703	-%1
9	34.304	58.682	%71
10	804.747	786.023	-%2
11	10.542	12.51	%19
12	150.276	166.312	%11
13	587.752	582.526	-%1
14	14.384	17.611	%22
15	8.957	6.323	-%29
16	4.725	4.704	%0
17	19.503	21.211	%9
18	27.392	27.143	-%1
19	114.524	109.545	-%4
20	91.112	90.254	-%1
21	19.14	17.603	-%8
22	12.704	13.032	%3

Veri Noktası	Gerçek Süre	Tahminlenen Süre	Hata
23	15.973	17.406	%9
24	170.99	157.106	-%8
25	145.02	149.059	%3

Tabloda ilgi çeken özellik, durumların önemli bir kısmında gerçeğe yakın ve orantısal olarak oldukça gerçekçi sonuçlar alınmış olmasıdır. Ancak, bazı sayıca az durumlarda büyük hatalar söz konusu olmuştur. Oluşan sonuçların istatistiksel özeti aşağıdaki tabloda sunulmuştur:

Tablo 6.2. Toplam süre tahmini istatistiksel özeti.

Korelasyon katsayısı	0.9415
Ortalama mutlak hata	10.85
Hatanın standart sapması	29.45
Göreli mutlak hata	%6.19

6.2 İşlemci Süresi Tahmini Sonuçları

İşlemci süresi tahmini için elde edilen sonuçlardan ilk yirmi beş adedi aşağıdaki tabloda verilmiştir.

Tablo 6.3. İşlemci süresi tahmini ve gerçekleşen sürelerin karşılaştırıldığı örnek sonuçlar

Veri Noktası	Gerçek İşlemci Süresi	Tahminlenen İşlemci Süresi	Hata
1	6.464	7.327	%13
2	17.816	21.322	%20
3	2.905	3.256	%12
4	40.517	41.661	%3
5	80.335	81.095	%1
6	420.238	445.429	%6
7	31.325	34.826	%11
8	547.579	493.16	-%10
9	20.432	18.504	-%9
10	465.458	550.509	%18
11	0.081	0.059	-%27
12	128.45	143.527	%12
13	517.282	509.98	-%1
14	6.464	7.338	%14
15	2.498	3.085	%23
16	0.92	0.877	-%5
17	11.86	18.253	%54
18	18.631	15.807	-%15
19	79.211	77.446	-%2
20	75.759	75.939	%0
21	5.587	7.338	%31
22	10.21	7.035	-%31

Veri Noktası	Gerçek İşlemci Süresi	Tahminlenen İşlemci Süresi	Hata
23	8.687	7.154	-%18
24	140.682	118.814	-%16
25	122.228	112.719	-%8

Bu tabloda da, toplam süre tahminine benzer olarak, durumların çoğunda yakın ve düşük hatalı tahminler, az bir kısmında ise büyük hatalar olduğu gözlemlenmektedir. Oluşan sonuçların istatistiksel özeti aşağıdaki tabloda sunulmuştur:

Tablo 6.4. İşlemci süresi tahmini istatistiksel özeti.

Korelasyon katsayısı	0.9436
Ortalama mutlak hata	6.90
Hatanın standart sapması	19.89
Göreli mutlak hata	%4.21

6.3 Giriş/Çıkış Miktarı Tahmini Sonuçları

İşlemci süresi tahmini için elde edilen sonuçlardan ilk yimi beş adedi aşağıdaki tabloda verilmiştir.

Tablo 6.5. Giriş/çıkış miktarı tahmini ve gerçekleşen miktarların karşılaştırıldığı örnek sonuçlar.

Veri Noktası	Gerçek Giriş/Çıkış Miktarı	Giriş Çıkış Miktarı Tahmini	Hata
1	35718	29892.65	-%16
2	329122	339864.8	%3
3	25684	27121.11	%6
4	53311	57770.75	%8
5	1993789	1645054	-%17
6	554637	690928	%25
7	92349	100949.6	%9
8	706229	624212.4	-%12
9	100561	95688.7	-%5
10	6156914	4044702	-%34
11	2671	2236.79	-%16
12	210884	225976.8	%7
13	1026241	1149675	%12
14	30448	28933.97	-%5
15	24925	25389.05	%2
16	9337	8518.25	-%9
17	37950	41642.39	%10
18	62822	45342.29	-%28
19	241411	214200.5	-%11
20	1533249	1552087	%1
21	25030	22933.97	-%8

Veri Noktası	Gerçek Giriş/Çıkış Miktarı	Giriş Çıkış Miktarı Tahmini	Hata
22	122383	107875.2	-%12
23	238889	225532	-%6
24	370777	336823.3	-%9
25	233291	216450.6	-%7

Bu tabloda, önceki tablolardan farklı olarak, tahminlerdeki hataların daha büyük olduğunu gözlemlemekteyiz. Aynı durum, bir sonraki bölümdeki toplam veri miktarı tahminlerinde de görülecektir. Bunun temelinde yatan sebep, toplam giriş çıkış miktarının ve toplam veri miktarının, sıklıkla sorgunun zorluğunun yanısıra, sorgunun sonucuna da bağlı olmasıdır. Yapılan modellemede, bunu direkt olarak hedef alan bir düzenleme olmadığından, bu beklenen bir durum olarak kabul edilebilir. Oluşan sonuçların istatistiksel özeti aşağıdaki tabloda sunulmuştur:

Tablo 6.6. Giriş/çıkış miktarı tahmini istatistiksel özeti.

Korelasyon katsayısı	0.7361
Ortalama mutlak hata	31.124
Hatanın standart sapması	127.836
Göreli mutlak hata	%11.27

6.4 Toplam Veri Miktarı Tahmini Sonuçları

Toplam veri miktarı tahmini için elde edilen sonuçlardan ilk yirmi beş adedi aşağıdaki tabloda verilmiştir.

Tablo 6.7. Toplam veri miktarı tahmini ve gerçekleşen miktarların karşılaştırıldığı örnek sonuçlar.

Veri Noktası	Gerçek Toplam Veri Miktarı	Tahminlenen Toplam Veri Miktarı	Hata
1	31958	28215.58	-%12
2	325240	330030.2	%1
3	24374	22387.77	-%8
4	48040	44445.7	-%7
5	1975061	1779486	-%10
6	538106	618953.2	%15
7	88936	97256.77	%9
8	703848	614514.2	-%13
9	93728	94024	%0
10	6139506	6103345	-%1
11	2034	2069.28	%2
12	201177	218227.5	%8
13	1025639	1136841	%11
14	28275	29256.05	%3
15	23025	24873.67	%8
16	8818	7970.65	-%10
17	34397	36784.67	%7
18	58918	51886.52	-%12
19	236510	218721.2	-%8
20	1532760	1537056	%0
21	23566	21256.05	-%10

Veri Noktası	Gerçek Toplam Veri Miktarı	Tahminlenen Toplam Veri Miktarı	Hata
22	122129	130791.1	%7
23	231084	215483.3	-%7
24	367357	379886.1	%3
25	232594	211250	-%9

Bu tabloda, ilk iki tablodan farklı ve giriş/çıkış miktarı tablosuna benzer olarak, tahminlerdeki hataların görece büyük olduğunu gözlemlemekteyiz. Oluşan sonuçların istatistiksel özeti aşağıdaki tabloda sunulmuştur:

Tablo 6.8. Toplam veri miktarı tahmini istatistiksel özeti.

Korelasyon katsayısı	0.8377
Ortalama mutlak hata	20.112
Hatanın standart sapması	92.447
Göreli mutlak hata	%8.67

6.5 Sonuçların Özeti

Elde edilen tüm tahminleme sonuçlarının özeti, aşağıdaki tabloda sunulmuştur:

Tablo 6.9. Tahminleme parametreleri sonuçları özeti

Tahminlenen Çıktılar / Maliyet Parametreleri			
Parametre	**Korelasyon Katsayısı**	**Ortalama Mutlak Hata**	**Kullanılan Veri Adedi**
Süre	0.9415	%6.19	48.600
İşlemci süresi	0.9436	%4.21	48.600
Giriş / çıkış	0.7361	%11.27	48.600
Veri	0.8377	%8.67	48.600

Tabloda görüleceği üzere, işlemci süresi ve toplam süre üzerine olan tahminlemeler, giriş/çıkış miktarı ve toplam süre üzerine olan tahminlemelere göre daha az hata oranına sahiptir. Bunun sebebine bakıldığında, sınıflandırmanın genel olarak başarılı olduğu, ancak özellikle pozisyon adı sorgulamasında, "basitlik ve sadelik" adına, sadece sorgunun zorluğuna bağlı bir sınıflandırma kullanıldığından, sorgunun vereceği sonuç sayısının tahminlenemesinden dolayı hatanın yükseldiği anlaşılmaktadır. Esas olarak, sadece sorguyu değerlendirerek, sorgunun gerçek ortamında çalıştırmadan bundan çok daha iyi bir sonuç elde etmek kuramsal olarak mümkün değildir.

Maliyet tahminlemesi açısından bakıldığında, maliyetin bir kaç, hatta bir kaç on kat değişkenlik gösterebildiği bir sorgu sisteminden bahsettiğimizi gözönüne aldığımız zaman, elde edilen hata oranlarıyla beraber tahminlenen sayıların oldukça kullanışlı olacağı sonucuna varılabilir.

Elde edilen sonuçlar hem çıktının hem de maliyet bileşenlerinin göreceli büyük hata payları olmadan makine öğrenmesi yöntemleri ile tahminlenebileceğini göstermektedir. Verinin sorgulanma karakteristiği (gelen sorguların çeşitliliği), büyüme trendleri, sorgulama algoritmalarının performans özellikleri sonuçlar üzerinde etkili olmakla birlikte ağırlık açısından önemli bir ölçüye ulaşmamaktadır. Bu sayede veri yoğun servisler için tahminlemenin kararlı şekilde yapılacağı varsayılmaktadır.

BÖLÜM 7

DEĞERLENDİRME

Bulut yaklaşımı için servis yönelimli bir mimariye sahip olmak ve buluta geçiş için ara adımları tasarlamak, mevcut sistemlerin bu yeni modele uyarlanması anlamında önem taşımaktadır. Sahip olunacak ya da tüketim unsuru olarak kullanılacak servislerin hazırlanması veya bu servislerin sistem entegrasyonlarında kullanılması yakın dönemde daha da önem kazanacaktır. Hem servis sağlayıcılar hem de servis tüketicileri için bu servislerin en uygun maliyetlerle yönetilmesi, yeni BT yönetiminin önemli bir parçası haline gelecektir. Bu nedenle, bu çalışmada planlanan servis sınıflandırması ve maliyet tahminlemesi için altyapı çalışması, dinamik maliyet hesaplamaları, simülasyonlar ve farklı bulut ortamlarının değerlendirilmesi için temel bir yaklaşım sunulmaktadır.

Büyük veri uygulamalarının gelişmesi ile birlikte, altyapıda servis sağlayıcıya ait bulut hizmetlerini kullanarak veriye dayalı analiz servislerinin sunulması yaygınlaşacaktır. Bulut servislerinin kullanılması verinin mevcut adanmış sunucularda işlenmesinin zorluğundan kaynaklanmaktadır. Büyük veri algoritmalarının karakteristik özellikleri de bu çalışmayı ilerletmek üzere gelecekte incelenebilir.

KAYNAKLAR

[1] Gökay Burak Akkuş, Erdem Uçar, ***Network Algorithms for Agent Based Automation of Web Services Composition***, INISTA, 2010.

[2] Jun-jie Wang, Sen Mu, ***Security Issues and Countermeasures in Cloud Computing***, In Proceedings of the 2011 IEEE International Conference on Grey Systems and Intelligent Services (GSIS), Nanjing, China, 15–18 September, pp. 843–846, 2011.

[3] ***Final Version of NIST Cloud Computing Definition Published***. Available online: http://www.nist.gov/itl/csd/cloud-102511.cfm (son erişim tarihi: 13.06.2016).

[4] Haoyong Lv, Yin Hu, ***Analysis and Research About Cloud Computing Security Protect Policy***, In Proceedings of the 2011 International Conference on Intelligence Science and Information Engineering (ISIE), Wuhan, China, 20–21 August, pp. 214–216, 2011.

[5] Peter Mell, Timothy Grance, ***The NIST Definition of Cloud Computing***, NIST: USA, 2009. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[6] Pritesh Jain, Dheeraj Rane, Shyam Patidar, ***A Survey and Analysis of Cloud Model-Based Security for Computing Secure Cloud Bursting and Aggregation in Renal Environment***, In Proceedings of the 2011 World Congress on Information and Communication Technologies (WICT), Mumbai, India, 11–14 December, pp. 456–461, 2011.

[7] Bhagyaraj Gowrigolla, Sathyalakshmi Sivaji, M. Roberts Masillamani, ***Design and Auditing of Cloud Computing Security***, In Proceedings of the 2010 5th International Conference on Information and Automation for Sustainability (ICIAFs), Colombo, Sri Lanka, 17–19 December, pp. 292–297, 2010.

[8] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, Anand Ghalsasi, ***Cloud Computing — The Business Perspective***, Decision Support Systems, Vol. 51, Issue 1, pp. 176-189, 2011.

[9] Issa M. Khalil, Abdallah Khreishah, Muhammad Azeem, ***Cloud Computing Security: A Survey***, Computers 2014, 3(1), pp. 1-35. doi:10.3390/computers3010001

[10] Marin Litoiu, Murray Woodside, Johnny Wong, Joanna Ng, Gabriel Iszlai, ***A Business Driven Cloud Optimization Architecture***, Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 380-385, 2010.

[11] Allen Brown, Hugo Haas, ***Web Services Glossary***, World Wide Web Consortium (W3C), <http://www.w3.org/TR/ws-gloss/>, HTML, 2002.

[12] Seokho Son, Sung Chan Jun, ***Negotiation-Based Flexible SLA Establishment with SLA-driven Resource Allocation in Cloud Computing***, Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on, IEEE, pp. 168-171, 2013.

[13] Young Choon Lee, Chen Wang, Albert Y. Zomaya, Bing Bing Zhou, ***Profit-Driven Scheduling for Cloud Services with Data Access Awareness***, Journal of Parallel and Distributed Computing 72.4, April, pp. 591-602, 2012.

[14] Zhipiao Liu, Shangguang Wang, Qibo Sun, Hua Zou, Fangchun Yang, ***Cost-Aware Cloud Service Request Scheduling for SaaS Providers***, The Computer Journal (2013): bxt009, 2013.

[15] Mario Macias, Jordi Guitart, ***Using Resource-Level Information into Nonadditive Negotiation Models for Cloud Market Environments***, In 12th IEEE/IFIP Network Operations and Management Symposium (NOMS'10), Osaka, Japan, April 19–23, pp. 325–332, 2010.

[16] Mario Macías, Jordi Guitart, ***A Genetic Model for Pricing in Cloud Computing Markets***, In Proceedings of the 2011 ACM Symposium on Applied Computing (SAC'11), ACM, New York, NY, USA, pp. 113-118, 2011.

[17] Donald F. Ferguson, Christos Nikolaou, Jakka Sairamesh, and Yechiam Yemini, ***Economic Models for Allocating Resources in Computer Systems***, Market Based Control of Distributed Systems. World Scientific Conference, pp. 156–183, 1996.

[18] Sarel Aiber, Dagan Gilat, Ariel Landau, Natalia Razinkov, Aviad Sela, Segev Wasserkrug, ***Autonomic Self-Optimization According to Business Objectives***, Proceedings of the International Conference on Autonomic Computing (ICAC'04), USA: IEEE Computer Society, pp. 206–213, 2004.

[19] Steven Newhouse, Jon MacLaren, Katarzyna Keahey, ***Trading Grid Services within the UK e-Science Grid***, Grid Resource Management: State of the Art and Future Trends, pp. 479–490, 2004.

[20] Rajkumar Buyya, ***Economic-Based Distributed Resource Management and Scheduling for Grid Computing***, Ph.D. thesis, Monash University, 2002.

[21] Gianfranco Chicco, R. Napoli, Federico Piglione, ***Comparisons Among Clustering Techniques for Electricity Customer Classification***, IEEE Transactions on Power Systems 21(2), pp. 933– 940, 2006.

[22] Nicolas Poggi, Toni Moreno, Josep Lluis Berral, Ricard Gavaldà, Jordi Torres, ***Web Customer Modeling for Automated Session Prioritization on High Traffic Sites***,

Proceedings of the 11th International Conference on User Modeling, Corfu, Greece, 2007.

[23] Nicolas Poggi, Toni Moreno, Josep Lluis Berral, Ricard Gavaldà, Jordi Torres, ***Self-Adaptive Utility-Based Web Session Management***, Computer Networks: The International Journal of Computer and Telecommunications Networking, Vol. 53 Issue 10, pp. 1712-1721, 2009.

[24] Harley Boughton, Pat Martin, Wendy Powley, Randy Horman, ***Workload Class Importance Policy in Autonomic Database Management Systems***, Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06) ,Washington, DC, USA: IEEE Computer Society, pp. 13–22, 2006.

[25] Ian Foster, Carl Kesselman, Craig Lee, Bob Lindell, Klara Nahrstedt, Alain Roy, ***A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation***, Proceedings of the 7th International Workshop on Quality of Service (IWQoS 1999), London, UK, pp. 62–80, 1999.

[26] Samuel Kounev, Ramon Nou, Jordi Torres, ***Building Online Performance Models of Grid Middleware with Fine-Grained Load-Balancing: A Globus Toolkit Case Study***, The 4th European Engineering Performance Workshop (EPEW 2007), Berlin, Germany, pp. 125-140, 2007.

[27] Karim Djemame, Iain Gourlay, James Padgett, Georg Birkenheuer, Matthias Hovestadt, Odej Kao, Kerstin Voß, ***Introducing Risk Management Into The Grid***, The 2nd IEEE International Conference on e-Science and Grid Computing (e-Science'06), Amsterdam, Netherlands, p. 28, 2006.

[28] Suresh Nair, Ravi Bapna, ***An Application of Yield Management for Internet Service Providers***, Naval Research Logistics, Vol. 48, Issue 5, pp. 348–362, 2001.

[29] Parijat Dube, Yezekael Hayel, Laura Wynter, ***Yield Management for IT Resources on Demand: Analysis and Validation of a New Paradigm for Managing Computing Centres***, Journal of Revenue and Pricing Management, Vol.4, Issue 1, pp. 24-38, 2005.

[30] Anthony Sulistio, Kyong Hoon Kim, Rajkumar Buyya, ***Managing Cancellations and No-Shows of Reservations with Overbooking to Increase Resource Revenue***, Proceedings of the 2008 8th International Symposium on Cluster Computing and the Grid (CCGRID), IEEE Computer Society, pp. 267-276, 2008.

[31] Bhuvan Urgaonkar, Prashant Shenoy, Timothy Roscoe, ***Resource Overbooking and Application Profiling in Shared Hosting Platforms***, OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation. New York, NY, USA, Vol. 36, Issue SI, pp. 239–254, 2002.

[32] Arun Anandasivam, Dirk Neumann, ***Managing Revenue in Grids***, Hawaii International Conference on System Sciences, Proceedings of the 42nd Annual, Springer-Verlag GmbH, 2009.

[33] Liu Kexing, ***A Survey of Agent Based Automated Negotiation***, Network Computing and Information Security (NCIS), 2011 International Conference on, Vol. 2. IEEE, 2011.

[34] Lutz Schubert, Keith Jeffery, ***Advances in Clouds: Research in Future Cloud Computing***, Expert Group Report, Public Version 1.0, 2012.

[35] Schroeck, M., Shockley, R., Smart, J., Romero-Morales, D., Tufano, P. , ***Analytics: The real world use of big data***, IBM Institute for Business Value, 2012.

[36] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten, ***The WEKA Data Mining Software: An Update***, SIGKDD Explorations, Vol. 11, Issue 1, pp. 10-18, 2009.

EKLER

EK-A Toplam Süre Tahminleme Modeli

Toplam süre tahminlemesi için Weka ile elde edilen M5P modeli aşağıda verilmiştir:

```
Scheme:       weka.classifiers.trees.M5P -U -M 100.0
Relation:     log-weka.filters.unsupervised.attribute.Remove-R1-2,7-10
Instances:    97200
Attributes:   5
              days
              job
              city
              cpuload
              time
Test mode:    split 50.0% train, remainder test

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

M5 pruned model tree:

days <= 21.5 :
|  days <= 10.5 :
|  |  days <= 4.5 : LM1 (15950/4.512%)
|  |  days >  4.5 :
|  |  |  city=2,3,1 <= 0.5 : LM2 (7407/2.797%)
|  |  |  city=2,3,1 >  0.5 :
|  |  |  |  job=0,3 <= 0.5 :
|  |  |  |  |  job=2,0,3 <= 0.5 : LM3 (5809/3.291%)
|  |  |  |  |  job=2,0,3 >  0.5 :
|  |  |  |  |  |  cpuload <= 0.645 :
|  |  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  |  city=3,1 <= 0.5 : LM4 (992/1.696%)
|  |  |  |  |  |  |  |  city=3,1 >  0.5 : LM5 (981/8.238%)
|  |  |  |  |  |  |  city=1 >  0.5 : LM6 (5977/6.943%)
|  |  |  |  |  |  cpuload >  0.645 :
|  |  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  |  city=3,1 <= 0.5 : LM7 (257/1.893%)
|  |  |  |  |  |  |  |  city=3,1 >  0.5 : LM8 (247/14.622%)
|  |  |  |  |  |  |  city=1 >  0.5 : LM9 (1486/13.049%)
```

```
|  |  |  |  job=0,3 >  0.5 : LM10 (13387/3.673%)
|  days >  10.5 :
|  |  city=3,1 <= 0.5 :
|  |  |  city=2,3,1 <= 0.5 : LM11 (2972/3.21%)
|  |  |  city=2,3,1 >  0.5 :
|  |  |  |  job=3 <= 0.5 : LM12 (817/2.165%)
|  |  |  |  job=3 >  0.5 : LM13 (667/4.359%)
|  |  city=3,1 >  0.5 :
|  |  |  cpuload <= 0.629 :
|  |  |  |  job=0,3 <= 0.5 :
|  |  |  |  |  job=2,0,3 <= 0.5 : LM14 (1647/5.547%)
|  |  |  |  |  job=2,0,3 >  0.5 : LM15 (2797/18.545%)
|  |  |  |  job=0,3 >  0.5 :
|  |  |  |  |  days <= 15.5 : LM16 (1752/4.267%)
|  |  |  |  |  days >  15.5 : LM17 (2190/5.422%)
|  |  |  cpuload >  0.629 :
|  |  |  |  job=0,3 <= 0.5 :
|  |  |  |  |  job=2,0,3 <= 0.5 : LM18 (404/11.208%)
|  |  |  |  |  job=2,0,3 >  0.5 : LM19 (695/36.027%)
|  |  |  |  job=0,3 >  0.5 :
|  |  |  |  |  cpuload <= 2.575 :
|  |  |  |  |  |  days <= 15.5 : LM20 (274/7.131%)
|  |  |  |  |  |  days >  15.5 : LM21 (304/7.115%)
|  |  |  |  cpuload >  2.575 : LM22 (428/10.364%)
days >  21.5 :
|  city=2,3,1 <= 0.5 :
|  |  days <= 50.5 :
|  |  |  job=2,0,3 <= 0.5 : LM23 (504/4.696%)
|  |  |  job=2,0,3 >  0.5 :
|  |  |  |  cpuload <= 0.627 : LM24 (1588/5.794%)
|  |  |  |  cpuload >  0.627 : LM25 (403/10.315%)
|  |  days >  50.5 :
|  |  |  job=2,0,3 <= 0.5 : LM26 (717/7.688%)
|  |  |  job=2,0,3 >  0.5 :
|  |  |  |  cpuload <= 0.649 :
|  |  |  |  |  job=0,3 <= 0.5 : LM27 (964/17.993%)
|  |  |  |  |  job=0,3 >  0.5 : LM28 (1286/2.645%)
|  |  |  |  cpuload >  0.649 : LM29 (566/22.192%)
|  city=2,3,1 >  0.5 :
|  |  days <= 50.5 :
|  |  |  cpuload <= 0.637 :
|  |  |  |  job=0,3 <= 0.5 :
|  |  |  |  |  job=2,0,3 <= 0.5 : LM30 (1586/14.915%)
|  |  |  |  |  job=2,0,3 >  0.5 :
|  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  city=3,1 <= 0.5 : LM31 (377/3.998%)
|  |  |  |  |  |  |  city=3,1 >  0.5 : LM32 (290/56.083%)
|  |  |  |  |  |  city=1 >  0.5 : LM33 (2068/50.683%)
|  |  |  |  job=0,3 >  0.5 :
|  |  |  |  |  days <= 34.5 :
|  |  |  |  |  |  days <= 27.5 : LM34 (778/6.47%)
|  |  |  |  |  |  days >  27.5 : LM35 (846/7.352%)
|  |  |  |  days >  34.5 :
|  |  |  |  |  |  days <= 42.5 :
|  |  |  |  |  |  |  city=1 <= 0.5 : LM36 (235/15.143%)
|  |  |  |  |  |  |  city=1 >  0.5 : LM37 (737/5.965%)
|  |  |  |  |  |  days >  42.5 :
```

```
|  |  |  |  |  |  |  city=1 <= 0.5 : LM38 (245/15.588%)
|  |  |  |  |  |  |  city=1 >  0.5 : LM39 (794/7.401%)
|  |  |  cpuload >  0.637 :
|  |  |  |  job=2,0,3 <= 0.5 :
|  |  |  |  |  city=1 <= 0.5 : LM40 (104/14.378%)
|  |  |  |  |  city=1 >  0.5 :
|  |  |  |  |  |  days <= 38.5 : LM41 (174/20.174%)
|  |  |  |  |  |  days >  38.5 :
|  |  |  |  |  |  |  cpuload <= 2.653 : LM42 (73/28.535%)
|  |  |  |  |  |  |  cpuload >  2.653 : LM43 (50/38.602%)
|  |  |  |  job=2,0,3 >  0.5 :
|  |  |  |  |  cpuload <= 1.883 : LM44 (627/46.138%)
|  |  |  |  |  cpuload >  1.883 :
|  |  |  |  |  |  days <= 33.5 : LM45 (411/51.478%)
|  |  |  |  |  |  days >  33.5 :
|  |  |  |  |  |  |  job=3 <= 0.5 :
|  |  |  |  |  |  |  |  city=1 <= 0.5 : LM46 (65/118.895%)
|  |  |  |  |  |  |  |  city=1 >  0.5 : LM47 (178/120.415%)
|  |  |  |  |  |  |  job=3 >  0.5 : LM48 (311/32.127%)
|  |  days >  50.5 :
|  |  |  cpuload <= 0.639 :
|  |  |  |  job=0,3 <= 0.5 :
|  |  |  |  |  job=2,0,3 <= 0.5 :
|  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  city=3,1 <= 0.5 : LM49 (255/2.799%)
|  |  |  |  |  |  |  city=3,1 >  0.5 : LM50 (282/35.359%)
|  |  |  |  |  |  city=1 >  0.5 : LM51 (1660/34.408%)
|  |  |  |  |  job=2,0,3 >  0.5 :
|  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  city=3,1 <= 0.5 : LM52 (452/7.806%)
|  |  |  |  |  |  |  city=3,1 >  0.5 : LM53 (450/133.678%)
|  |  |  |  |  |  city=1 >  0.5 : LM54 (2772/109.053%)
|  |  |  |  job=0,3 >  0.5 :
|  |  |  |  |  days <= 68.5 :
|  |  |  |  |  |  city=1 <= 0.5 : LM55 (583/20.951%)
|  |  |  |  |  |  city=1 >  0.5 :
|  |  |  |  |  |  |  days <= 58.5 : LM56 (762/7.952%)
|  |  |  |  |  |  |  days >  58.5 : LM57 (1019/9.479%)
|  |  |  |  |  days >  68.5 : LM58 (2786/18.315%)
|  |  |  cpuload >  0.639 :
|  |  |  |  job=0,3 <= 0.5 :
|  |  |  |  |  job=2,0,3 <= 0.5 : LM59 (514/66.526%)
|  |  |  |  |  job=2,0,3 >  0.5 :
|  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  city=3,1 <= 0.5 :
|  |  |  |  |  |  |  |  cpuload <= 2.323 : LM60 (55/10.85%)
|  |  |  |  |  |  |  |  cpuload >  2.323 : LM61 (58/18.452%)
|  |  |  |  |  |  |  city=3,1 >  0.5 :
|  |  |  |  |  |  |  |  cpuload <= 3.237 : LM62 (80/208.987%)
|  |  |  |  |  |  |  |  cpuload >  3.237 : LM63 (33/356.318%)
|  |  |  |  |  |  city=1 >  0.5 : LM64 (728/213.565%)
|  |  |  |  job=0,3 >  0.5 :
|  |  |  |  |  cpuload <= 2.44 : LM65 (651/33.615%)
|  |  |  |  |  cpuload >  2.44 : LM66 (643/43.081%)

LM num: 1
time =
```

```
        + 17.6038

LM num: 2
time =
        + 17.099

LM num: 3
time =
        + 15.3121

LM num: 4
time =
        + 9.1592

LM num: 5
time =
        3.9391 * days
        + 7.2397 * cpuload
        - 1.9191

LM num: 6
time =
        6.5466 * days
        + 14.9767 * cpuload
        - 9.9114

LM num: 7
time =
        + 9.9001

LM num: 8
time =
        8.1609 * cpuload
        + 26.8359

LM num: 9
time =
        12.5239 * days
        + 17.1948 * cpuload
        - 58.2398

LM num: 10
time =
        8.4808 * days
        + 17.2813 * city=3,1
        + 8.4836 * city=1
        + 22.5843 * cpuload
        - 34.1155

LM num: 11
time =
        1.3312 * days
        + 19.168 * job=2,0,3
        - 2.2064 * job=0,3
        + 7.7194 * cpuload
```

- 13.2174

LM num: 12
time =
+ 11.6989

LM num: 13
time =
7.0895 * days
+ 35.9578 * cpuload
- 35.126

LM num: 14
time =
1.6195 * days
+ 8.1146 * cpuload
- 0.4776

LM num: 15
time =
7.5714 * days
+ 40.9704 * cpuload
- 33.6745

LM num: 16
time =
8.3697 * days
+ 40.6737 * cpuload
- 18.2546

LM num: 17
time =
8.5074 * days
+ 61.1717 * cpuload
- 27.9867

LM num: 18
time =
3.556 * days
+ 8.3285 * cpuload
- 30.2195

LM num: 19
time =
14.2575 * days
+ 46.9913 * cpuload
- 155.4173

LM num: 20
time =
11.0607 * days
+ 38.5435 * cpuload
- 50.7604

LM num: 21

time =
	11.8417 * days
	+ 59.7228 * cpuload
	- 86.1583

LM num: 22
time =
	19.7352 * days
	+ 51.5111 * cpuload
	- 199.5606

LM num: 23
time =
	+ 24.1459

LM num: 24
time =
	1.342 * days
	- 12.5256 * job=0,3
	+ 14.8155 * cpuload
	+ 10.0138

LM num: 25
time =
	2.1361 * days
	- 18.017 * job=0,3
	+ 19.5314 * cpuload
	- 16.0442

LM num: 26
time =
	0.5563 * days
	+ 12.9834 * cpuload
	- 7.7951

LM num: 27
time =
	1.9944 * days
	+ 59.607 * cpuload
	- 35.8184

LM num: 28
time =
	1.1901 * days
	+ 7.5804

LM num: 29
time =
	2.3829 * days
	- 60.9796 * job=0,3
	+ 38.7949 * cpuload
	- 34.8739

LM num: 30
time =

1.9891 * days
+ 38.9265 * city=3,1
+ 23.912 * city=1
+ 16.0278 * cpuload
- 66.7214

LM num: 31
time =
+ 23.8442

LM num: 32
time =
6.6188 * days
- 53.961

LM num: 33
time =
10.1352 * days
+ 143.5374 * cpuload
- 117.1264

LM num: 34
time =
8.6894 * days
+ 36.128 * city=3,1
+ 29.1324 * city=1
+ 82.6603 * cpuload
- 100.4818

LM num: 35
time =
8.6661 * days
+ 50.5973 * city=3,1
+ 27.3118 * city=1
+ 109.7702 * cpuload
- 118.9233

LM num: 36
time =
3.9921 * days
+ 66.9868 * city=3,1
+ 131.084 * cpuload
+ 39.1911

LM num: 37
time =
9.7065 * days
+ 145.0546 * cpuload
- 87.2347

LM num: 38
time =
7.4896 * days
+ 78.2842 * city=3,1
+ 125.059 * cpuload

- 93.6818

LM num: 39
time =
10.0506 * days
+ 167.8076 * cpuload
- 111.9752

LM num: 40
time =
55.1615 * city=3,1
+ 12.6552

LM num: 41
time =
3.4874 * days
- 11.2206

LM num: 42
time =
+ 130.7801

LM num: 43
time =
+ 246.2519

LM num: 44
time =
11.4154 * days
+ 90.6498 * job=0,3
+ 161.0153 * city=1
+ 81.2251 * cpuload
- 316.5324

LM num: 45
time =
17.4082 * days
+ 102.3307 * job=3
+ 178.8753 * city=1
+ 85.0384 * cpuload
- 506.7895

LM num: 46
time =
+ 241.2443

LM num: 47
time =
24.8355 * days
+ 132.3625 * cpuload
- 656.1141

LM num: 48
time =
20.8081 * days

```
        + 157.3791 * cpuload
        - 595.297

LM num: 49
time =
        + 16.5515

LM num: 50
time =
        1.7778 * days
        + 70.2844 * cpuload
        - 46.8246

LM num: 51
time =
        2.8476 * days
        + 56.7564 * cpuload
        - 57.4882

LM num: 52
time =
        0.8183 * days
        + 16.4364 * cpuload
        - 13.0916

LM num: 53
time =
        10.4054 * days
        + 227.2216 * cpuload
        - 387.9756

LM num: 54
time =
        12.5439 * days
        + 222.1515 * cpuload
        - 271.7821

LM num: 55
time =
        7.9151 * days
        + 103.487 * city=3,1
        + 201.7065 * cpuload
        - 149.2704

LM num: 56
time =
        10.1023 * days
        + 205.7518 * cpuload
        - 130.1588

LM num: 57
time =
        10.3209 * days
        + 251.7542 * cpuload
        - 160.395
```

LM num: 58
time =
	10.3628 * days
	+ 143.5013 * city=3,1
	+ 61.853 * city=1
	+ 309.1504 * cpuload
	- 389.185

LM num: 59
time =
	4.1322 * days
	+ 188.2578 * city=1
	+ 57.6249 * cpuload
	- 322.2371

LM num: 60
time =
	+ 65.6032

LM num: 61
time =
	+ 108.1543

LM num: 62
time =
	+ 589.9479

LM num: 63
time =
	+ 1277.1034

LM num: 64
time =
	21.4119 * days
	+ 266.2896 * cpuload
	- 876.8605

LM num: 65
time =
	13.9918 * days
	+ 190.2094 * city=1
	+ 258.866 * cpuload
	- 594.5632

LM num: 66
time =
	22.5001 * days
	+ 249.6991 * city=1
	+ 273.2702 * cpuload
	- 1266.5984

Number of Rules : 66

EK-B İşlemci Süresi Tahminleme Modeli

İşlemci süresi tahminlemesi için Weka ile elde edilen M5P modeli aşağıda verilmiştir:

```
Scheme:       weka.classifiers.trees.M5P -U -M 100.0
Relation:     log-weka.filters.unsupervised.attribute.Remove-R1-2,7-9,11
Instances:    97200
Attributes:   5
              days
              job
              city
              cpuload
              cpu
Test mode:    split 50.0% train, remainder test

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

M5 pruned model tree:

days <= 20.5 :
|  days <= 9.5 :
|  |  city=2,3,1 <= 0.5 : LM1 (10283/2.03%)
|  |  city=2,3,1 >  0.5 :
|  |  |  days <= 4.5 : LM2 (12794/4.252%)
|  |  |  days >  4.5 :
|  |  |  |  job=0,3 <= 0.5 :
|  |  |  |  |  job=2,0,3 <= 0.5 : LM3 (5600/2.677%)
|  |  |  |  |  job=2,0,3 >  0.5 :
|  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  city=3,1 <= 0.5 : LM4 (1202/0.433%)
|  |  |  |  |  |  |  city=3,1 >  0.5 : LM5 (1178/9.873%)
|  |  |  |  |  |  city=1 >  0.5 : LM6 (7173/8.325%)
|  |  |  |  job=0,3 >  0.5 : LM7 (12891/2.878%)
|  days >  9.5 :
|  |  city=3,1 <= 0.5 :
|  |  |  city=2,3,1 <= 0.5 : LM8 (2991/3.997%)
|  |  |  city=2,3,1 >  0.5 :
|  |  |  |  job=3 <= 0.5 : LM9 (809/1.45%)
|  |  |  |  job=3 >  0.5 : LM10 (669/3.137%)
|  |  city=3,1 >  0.5 :
|  |  |  job=0,3 <= 0.5 :
|  |  |  |  job=2,0,3 <= 0.5 : LM11 (2033/6.242%)
|  |  |  |  job=2,0,3 >  0.5 : LM12 (3544/21.643%)
|  |  |  job=0,3 >  0.5 : LM13 (4928/5.022%)
days >  20.5 :
|  city=2,3,1 <= 0.5 :
|  |  days <= 50.5 :
|  |  |  job=2,0,3 <= 0.5 : LM14 (558/3.935%)
|  |  |  job=2,0,3 >  0.5 :
|  |  |  |  days <= 32.5 : LM15 (972/4.712%)
|  |  |  |  days >  32.5 : LM16 (1226/8.158%)
|  |  days >  50.5 :
|  |  |  job=2,0,3 <= 0.5 : LM17 (717/7.671%)
|  |  |  job=2,0,3 >  0.5 :
```

```
|  |  |  |  job=0,3 <= 0.5 : LM18 (1206/22.516%)
|  |  |  |  job=0,3 >  0.5 : LM19 (1610/0.322%)
|  city=2,3,1 >  0.5 :
|  |  days <= 49.5 :
|  |  |  job=2,0,3 <= 0.5 :
|  |  |  |  days <= 33.5 : LM20 (1060/11.608%)
|  |  |  |  days >  33.5 :
|  |  |  |  |  city=1 <= 0.5 : LM21 (256/16.259%)
|  |  |  |  |  city=1 >  0.5 : LM22 (843/22.196%)
|  |  |  job=2,0,3 >  0.5 :
|  |  |  |  days <= 31.5 :
|  |  |  |  |  job=3 <= 0.5 :
|  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  city=3,1 <= 0.5 : LM23 (232/2.689%)
|  |  |  |  |  |  |  city=3,1 >  0.5 : LM24 (180/42.092%)
|  |  |  |  |  |  city=1 >  0.5 : LM25 (1153/38.7%)
|  |  |  |  |  job=3 >  0.5 :
|  |  |  |  |  |  days <= 24.5 : LM26 (945/5.952%)
|  |  |  |  |  |  days >  24.5 : LM27 (1110/7.878%)
|  |  |  |  days >  31.5 :
|  |  |  |  |  job=3 <= 0.5 :
|  |  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  |  city=3,1 <= 0.5 : LM28 (296/4.748%)
|  |  |  |  |  |  |  city=3,1 >  0.5 :
|  |  |  |  |  |  |  |  days <= 44.5 : LM29 (180/68.368%)
|  |  |  |  |  |  |  |  days >  44.5 : LM30 (64/93.975%)
|  |  |  |  |  |  city=1 >  0.5 : LM31 (1635/70.222%)
|  |  |  |  |  job=3 >  0.5 :
|  |  |  |  |  |  days <= 41.5 :
|  |  |  |  |  |  |  city=1 <= 0.5 : LM32 (373/14.162%)
|  |  |  |  |  |  |  city=1 >  0.5 : LM33 (1122/7.181%)
|  |  |  |  |  |  days >  41.5 : LM34 (1257/11.368%)
|  |  days >  49.5 :
|  |  |  job=0,3 <= 0.5 :
|  |  |  |  job=2,0,3 <= 0.5 :
|  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  city=3,1 <= 0.5 : LM35 (324/2.979%)
|  |  |  |  |  |  city=3,1 >  0.5 : LM36 (344/46.228%)
|  |  |  |  |  city=1 >  0.5 : LM37 (2097/42.941%)
|  |  |  |  job=2,0,3 >  0.5 :
|  |  |  |  |  city=1 <= 0.5 :
|  |  |  |  |  |  city=3,1 <= 0.5 : LM38 (576/9.615%)
|  |  |  |  |  |  city=3,1 >  0.5 : LM39 (580/169.559%)
|  |  |  |  |  city=1 >  0.5 : LM40 (3597/137.567%)
|  |  |  job=0,3 >  0.5 :
|  |  |  |  days <= 68.5 :
|  |  |  |  |  days <= 58.5 : LM41 (1439/13.648%)
|  |  |  |  |  days >  58.5 :
|  |  |  |  |  |  city=1 <= 0.5 : LM42 (408/28.322%)
|  |  |  |  |  |  city=1 >  0.5 : LM43 (1270/11.316%)
|  |  |  |  days >  68.5 : LM44 (3475/22.936%)

LM num: 1
cpu =
          + 5.5724
```

LM num: 2
cpu =
	+ 8.5621

LM num: 3
cpu =
	+ 5.661

LM num: 4
cpu =
	+ 1.2498

LM num: 5
cpu =
	3.3765 * days
	- 0.8372 * cpuload
	- 5.8061

LM num: 6
cpu =
	5.4703 * days
	- 9.6185

LM num: 7
cpu =
	6.2021 * days
	+ 12.3773 * city=3,1
	+ 6.2288 * city=1
	+ 0.0707 * cpuload
	- 18.2033

LM num: 8
cpu =
	+ 14.1297

LM num: 9
cpu =
	+ 2.773

LM num: 10
cpu =
	5.1645 * days
	- 11.6097

LM num: 11
cpu =
	1.4193 * days
	- 5.1803

LM num: 12
cpu =
	6.4123 * days
	- 22.1389

LM num: 13

cpu =
	7.0657 * days
	+ 12.5553 * city=1
	- 19.5441

LM num: 14
cpu =
	+ 11.456

LM num: 15
cpu =
	1.2179 * days
	- 6.1501 * job=0,3
	+ 1.559

LM num: 16
cpu =
	1.0189 * days
	- 12.7457 * job=0,3
	+ 12.9626

LM num: 17
cpu =
	0.3924 * days
	- 3.241

LM num: 18
cpu =
	1.6041 * days
	- 12.8383

LM num: 19
cpu =
	1.0012 * days
	+ 0.8924

LM num: 20
cpu =
	1.5256 * days
	+ 21.1733 * city=3,1
	+ 14.0542 * city=1
	- 38.6094

LM num: 21
cpu =
	38.9498 * city=3,1
	+ 3.3185

LM num: 22
cpu =
	2.4021 * days
	- 32.3285

LM num: 23
cpu =

+ 8.8167

LM num: 24
cpu =
5.7939 * days
- 53.3303

LM num: 25
cpu =
7.6635 * days
- 36.117

LM num: 26
cpu =
7.0207 * days
+ 27.9244 * city=3,1
+ 19.0814 * city=1
- 51.0939

LM num: 27
cpu =
7.5327 * days
+ 36.4632 * city=3,1
+ 24.1331 * city=1
- 74.5793

LM num: 28
cpu =
+ 17.0715

LM num: 29
cpu =
+ 148.6896

LM num: 30
cpu =
+ 234.6338

LM num: 31
cpu =
9.4545 * days
- 97.9686

LM num: 32
cpu =
6.728 * days
+ 52.7739 * city=3,1
- 58.737

LM num: 33
cpu =
8.0548 * days
- 29.8306

LM num: 34

cpu =
	8.0304 * days
	+ 64.2201 * city=3,1
	+ 32.1058 * city=1
	- 123.4884

LM num: 35
cpu =
	+ 7.8577

LM num: 36
cpu =
	1.7022 * days
	- 37.6409

LM num: 37
cpu =
	2.4081 * days
	- 37.5663

LM num: 38
cpu =
	0.7116 * days
	- 13.3271

LM num: 39
cpu =
	8.0244 * days
	- 206.1367

LM num: 40
cpu =
	10.4849 * days
	- 157.6815

LM num: 41
cpu =
	8.2129 * days
	+ 80.2768 * city=3,1
	+ 35.7388 * city=1
	- 150.5349

LM num: 42
cpu =
	6.0664 * days
	+ 95.9703 * city=3,1
	- 33.7928

LM num: 43
cpu =
	8.6328 * days
	+ 1.1792 * cpuload
	- 58.8211

LM num: 44

```
cpu =
        8.6806 * days
        + 121.4801 * city=3,1
        + 53.6653 * city=1
        - 235.0464

Number of Rules : 44
```

EK-C Giriş/Çıkış Miktarı Tahminleme Modeli

Giriş/çıkış miktarı tahminlemesi için Weka ile elde edilen M5P modeli aşağıda verilmiştir:

```
Scheme:       weka.classifiers.trees.M5P -M 100.0
Relation:     log-weka.filters.unsupervised.attribute.Remove-R1-2,7-8,10-11
Instances:    97200
Attributes:   5
              days
              job
              city
              cpuload
              io
Test mode:    split 50.0% train, remainder test

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

M5 pruned model tree:
(using smoothed linear models)

days <= 19.5 :
|  job=3,0 <= 0.5 :
|  |  days <= 5.5 : LM1 (10234/4.347%)
|  |  days >  5.5 :
|  |  |  days <= 11.5 :
|  |  |  |  job=2,3,0 <= 0.5 : LM2 (6979/4.789%)
|  |  |  |  job=2,3,0 >  0.5 : LM3 (11922/6.956%)
|  |  |  days >  11.5 :
|  |  |  |  job=2,3,0 <= 0.5 : LM4 (2092/8.29%)
|  |  |  |  job=2,3,0 >  0.5 :
|  |  |  |  |  city=1,0 <= 0.5 :
|  |  |  |  |  |  city=3,1,0 <= 0.5 : LM5 (371/1.875%)
|  |  |  |  |  |  city=3,1,0 >  0.5 : LM6 (373/15.062%)
|  |  |  |  |  city=1,0 >  0.5 : LM7 (2934/12.6%)
|  job=3,0 >  0.5 :
|  |  days <= 4.5 :
|  |  |  days <= 2.5 :
|  |  |  |  days <= 1.5 : LM8 (2039/4.769%)
|  |  |  |  days >  1.5 : LM9 (2018/6.237%)
|  |  |  days >  2.5 : LM10 (3223/11.009%)
|  |  days >  4.5 :
|  |  |  days <= 10.5 :
|  |  |  |  city=1,0 <= 0.5 :
```

```
|  |  |  |  |  city=3,1,0 <= 0.5 : LM11 (1666/2.657%)
|  |  |  |  |  city=3,1,0 >  0.5 : LM12 (1719/18.894%)
|  |  |  |  city=1,0 >  0.5 : LM13 (13422/24.087%)
|  |  |  days >  10.5 : LM14 (5752/40.744%)
days >  19.5 :
|  job=3,0 <= 0.5 :
|  |  days <= 46.5 :
|  |  |  job=2,3,0 <= 0.5 : LM15 (2697/17.595%)
|  |  |  job=2,3,0 >  0.5 :
|  |  |  |  city=1,0 <= 0.5 :
|  |  |  |  |  city=3,1,0 <= 0.5 : LM16 (502/3.643%)
|  |  |  |  |  city=3,1,0 >  0.5 : LM17 (418/28.916%)
|  |  |  |  city=1,0 >  0.5 : LM18 (3585/26.332%)
|  |  days >  46.5 :
|  |  |  job=2,3,0 <= 0.5 : LM19 (3754/36.791%)
|  |  |  job=2,3,0 >  0.5 :
|  |  |  |  days <= 68.5 :
|  |  |  |  |  city=1,0 <= 0.5 : LM20 (656/38.68%)
|  |  |  |  |  city=1,0 >  0.5 : LM21 (2575/45.392%)
|  |  |  |  days >  68.5 : LM22 (3203/60.502%)
|  job=3,0 >  0.5 :
|  |  days <= 60.5 :
|  |  |  days <= 36.5 : LM23 (4231/74.727%)
|  |  |  days >  36.5 : LM24 (4816/129.352%)
|  |  days >  60.5 : LM25 (6019/219.418%)

LM num: 1
io =
        18.8608 * days
        + 98.1274 * job=2,3,0
        + 57.2882 * job=3,0
        + 110.1974 * city=3,1,0
        + 74.1461 * city=1,0
        + 50.5404 * city=0
        + 25755.7702

LM num: 2
io =
        18.3858 * days
        + 188.956 * job=2,3,0
        + 57.2882 * job=3,0
        + 196.4516 * city=3,1,0
        + 27647.7884 * city=1,0
        + 10248.3334 * city=0
        + 16782.8751

LM num: 3
io =
        18.3858 * days
        + 153.4999 * job=2,3,0
        + 57.2882 * job=3,0
        + 46848.1022 * city=3,1,0
        + 29913.2468 * city=1,0
        + 19694.6262 * city=0
        + 10973.7025
```

LM num: 4
io =
	5073.0953 * days
	+ 720.4452 * job=2,3,0
	+ 57.2882 * job=3,0
	+ 41536.069 * city=3,1,0
	+ 30572.5216 * city=1,0
	+ 23225.3165 * city=0
	- 67946.8208

LM num: 5
io =
	444.0441 * days
	+ 492.5988 * job=2,3,0
	+ 57.2882 * job=3,0
	+ 5492.968 * city=3,1,0
	+ 1411.7517 * city=1,0
	+ 1059.8319 * city=0
	+ 165.9968 * cpuload
	+ 13675.6823

LM num: 6
io =
	9032.5956 * days
	+ 492.5988 * job=2,3,0
	+ 57.2882 * job=3,0
	+ 5475.9061 * city=3,1,0
	+ 1411.7517 * city=1,0
	+ 1059.8319 * city=0
	+ 7302.0802 * cpuload
	- 43120.525

LM num: 7
io =
	10590.0701 * days
	+ 492.5988 * job=2,3,0
	+ 57.2882 * job=3,0
	+ 916.1169 * city=3,1,0
	+ 602.2906 * city=1,0
	+ 41932.8937 * city=0
	+ 8.736 * cpuload
	- 2683.5807

LM num: 8
io =
	207.1114 * days
	+ 44.604 * job=2,3,0
	+ 61.0829 * job=3,0
	+ 296.6564 * city=3,1,0
	+ 17280.9045 * city=1,0
	+ 5744.9597 * city=0
	+ 4.4974 * cpuload
	+ 9633.7793

LM num: 9
io =

208.2366 * days
+ 44.604 * job=2,3,0
+ 61.0829 * job=3,0
+ 296.6564 * city=3,1,0
+ 29214.1394 * city=1,0
+ 6896.8935 * city=0
+ 1090.7785 * cpuload
+ 13815.6419

LM num: 10
io =
17138.8784 * days
+ 44.604 * job=2,3,0
+ 61.0829 * job=3,0
+ 30727.3693 * city=3,1,0
+ 28768.8954 * city=1,0
+ 15293.6642 * city=0
- 49355.5189

LM num: 11
io =
156.6106 * days
+ 44.604 * job=2,3,0
+ 61.0829 * job=3,0
+ 1161.3228 * city=3,1,0
+ 347.0586 * city=1,0
+ 235.4582 * city=0
+ 16166.3584

LM num: 12
io =
9559.0722 * days
+ 44.604 * job=2,3,0
+ 61.0829 * job=3,0
+ 1141.5104 * city=3,1,0
+ 347.0586 * city=1,0
+ 235.4582 * city=0
+ 16150.6323

LM num: 13
io =
18237.4217 * days
+ 44.604 * job=2,3,0
+ 61.0829 * job=3,0
+ 272.2258 * city=3,1,0
+ 182.5133 * city=1,0
+ 33587.4695 * city=0
+ 478.7419

LM num: 14
io =
14947.234 * days
+ 44.604 * job=2,3,0
+ 61.0829 * job=3,0
+ 140050.9643 * city=3,1,0
+ 84881.8341 * city=1,0

+ 65174.6188 * city=0
- 188734.817

LM num: 15
io =
4827.7347 * days
+ 1608.6983 * job=2,3,0
+ 368.9349 * job=3,0
+ 93511.7031 * city=3,1,0
+ 53854.4901 * city=1,0
+ 44710.4326 * city=0
- 134574.9277

LM num: 16
io =
312.2958 * days
+ 1281.5341 * job=2,3,0
+ 368.9349 * job=3,0
+ 8668.9939 * city=3,1,0
+ 2678.6276 * city=1,0
+ 2027.8187 * city=0
+ 29774.7478

LM num: 17
io =
6785.0857 * days
+ 1281.5341 * job=2,3,0
+ 368.9349 * job=3,0
+ 9580.7047 * city=3,1,0
+ 2678.6276 * city=1,0
+ 2027.8187 * city=0
- 15207.1702

LM num: 18
io =
10645.3968 * days
+ 1281.5341 * job=2,3,0
+ 368.9349 * job=3,0
+ 2022.0966 * city=3,1,0
+ 1331.9544 * city=1,0
+ 88677.9446 * city=0
- 15411.6561

LM num: 19
io =
4749.4706 * days
+ 1922.8696 * job=2,3,0
+ 368.9349 * job=3,0
+ 193273.5815 * city=3,1,0
+ 119513.0677 * city=1,0
+ 77930.8145 * city=0
- 283945.7863

LM num: 20
io =
3171.1653 * days

+ 1388.9587 * job=2,3,0
+ 368.9349 * job=3,0
+ 293620.7352 * city=3,1,0
+ 6797.8659 * city=1,0
+ 4214.7503 * city=0
- 104721.2997

LM num: 21
io =
9795.3053 * days
+ 1388.9587 * job=2,3,0
+ 368.9349 * job=3,0
+ 4752.5928 * city=3,1,0
+ 3252.5005 * city=1,0
+ 128064.7575 * city=0
+ 13489.3597

LM num: 22
io =
9666.7752 * days
+ 1388.9587 * job=2,3,0
+ 368.9349 * job=3,0
+ 442984.6421 * city=3,1,0
+ 260459.4342 * city=1,0
+ 198286.0283 * city=0
- 664202.6774

LM num: 23
io =
13074.4347 * days
+ 302.7237 * job=2,3,0
+ 415.0542 * job=3,0
+ 216546.3181 * city=3,1,0
+ 182268.8764 * city=1,0
+ 104380.1407 * city=0
- 282166.8816

LM num: 24
io =
12182.8094 * days
+ 302.7237 * job=2,3,0
+ 415.0542 * job=3,0
+ 443135.017 * city=3,1,0
+ 292746.5681 * city=1,0
+ 179005.0948 * city=0
- 490098.0385

LM num: 25
io =
16633.0598 * days
+ 302.7237 * job=2,3,0
+ 415.0542 * job=3,0
+ 733816.5969 * city=3,1,0
+ 424238.2745 * city=1,0
+ 302798.5391 * city=0
- 1097013.6061

Number of Rules : 25

EK-D Veri Miktarı Tahminleme Modeli

Veri miktarı tahminlemesi için Weka ile elde edilen M5P modeli aşağıda verilmiştir:

```
Scheme:       weka.classifiers.trees.M5P -M 100.0
Relation:     log-weka.filters.unsupervised.attribute.Remove-R1-2,7,9-11
Instances:    97200
Attributes:   5
              days
              job
              city
              cpuload
              data
Test mode:    split 50.0% train, remainder test

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

M5 pruned model tree:
(using smoothed linear models)

days <= 19.5 :
|  job=3,0 <= 0.5 :
|  |  days <= 5.5 : LM1 (10234/4.205%)
|  |  days >  5.5 :
|  |  |  days <= 11.5 :
|  |  |  |  job=2,3,0 <= 0.5 : LM2 (6979/4.731%)
|  |  |  |  job=2,3,0 >  0.5 : LM3 (11922/6.577%)
|  |  |  days >  11.5 :
|  |  |  |  job=2,3,0 <= 0.5 : LM4 (2092/8.141%)
|  |  |  |  job=2,3,0 >  0.5 :
|  |  |  |  |  city=1,0 <= 0.5 :
|  |  |  |  |  |  city=3,1,0 <= 0.5 : LM5 (371/1.786%)
|  |  |  |  |  |  city=3,1,0 >  0.5 : LM6 (373/14.882%)
|  |  |  |  |  city=1,0 >  0.5 : LM7 (2934/12.492%)
|  job=3,0 >  0.5 :
|  |  days <= 4.5 :
|  |  |  days <= 2.5 : LM8 (4057/5.481%)
|  |  |  days >  2.5 : LM9 (3223/10.905%)
|  |  days >  4.5 :
|  |  |  days <= 10.5 :
|  |  |  |  city=1,0 <= 0.5 :
|  |  |  |  |  city=3,1,0 <= 0.5 : LM10 (1666/2.58%)
|  |  |  |  |  city=3,1,0 >  0.5 : LM11 (1719/18.799%)
|  |  |  |  city=1,0 >  0.5 : LM12 (13422/24.004%)
|  |  |  days >  10.5 : LM13 (5752/40.742%)
days >  19.5 :
|  job=3,0 <= 0.5 :
|  |  days <= 46.5 :
|  |  |  job=2,3,0 <= 0.5 : LM14 (2697/17.415%)
|  |  |  job=2,3,0 >  0.5 :
```

```
|  |  |  |  city=1,0 <= 0.5 :
|  |  |  |  |  city=3,1,0 <= 0.5 : LM15 (502/3.53%)
|  |  |  |  |  city=3,1,0 >  0.5 : LM16 (418/28.739%)
|  |  |  |  city=1,0 >  0.5 : LM17 (3585/26.232%)
|  |  days >  46.5 :
|  |  |  job=2,3,0 <= 0.5 : LM18 (3754/36.636%)
|  |  |  job=2,3,0 >  0.5 :
|  |  |  |  days <= 68.5 :
|  |  |  |  |  city=1,0 <= 0.5 : LM19 (656/38.521%)
|  |  |  |  |  city=1,0 >  0.5 : LM20 (2575/45.348%)
|  |  |  |  days >  68.5 : LM21 (3203/60.478%)
|  job=3,0 >  0.5 :
|  |  days <= 60.5 :
|  |  |  days <= 36.5 : LM22 (4231/74.794%)
|  |  |  days >  36.5 : LM23 (4816/129.643%)
|  |  days >  60.5 : LM24 (6019/220.235%)

LM num: 1
data =
        18.4788 * days
        + 95.9843 * job=2,3,0
        + 56.6047 * job=3,0
        + 107.2662 * city=3,1,0
        + 72.3735 * city=1,0
        + 49.6811 * city=0
        + 23984.5217

LM num: 2
data =
        34.2344 * days
        + 184.7517 * job=2,3,0
        + 56.6047 * job=3,0
        + 190.042 * city=3,1,0
        + 29036.7529 * city=1,0
        + 85.4677 * city=0
        + 15277.1207

LM num: 3
data =
        9206.7662 * days
        + 150.1225 * job=2,3,0
        + 56.6047 * job=3,0
        + 45107.4448 * city=3,1,0
        + 28849.703 * city=1,0
        + 19323.0752 * city=0
        - 63392.7954

LM num: 4
data =
        4987.8362 * days
        + 706.7532 * job=2,3,0
        + 56.6047 * job=3,0
        + 40195.6706 * city=3,1,0
        + 29614.6875 * city=1,0
        + 22755.0334 * city=0
        - 67750.2864
```

LM num: 5
data =
	436.3472 * days
	+ 482.9818 * job=2,3,0
	+ 56.6047 * job=3,0
	+ 5362.9394 * city=3,1,0
	+ 1383.0345 * city=1,0
	+ 1043.633 * city=0
	+ 157.4403 * cpuload
	+ 12095.9387

LM num: 6
data =
	8872.8842 * days
	+ 482.9818 * job=2,3,0
	+ 56.6047 * job=3,0
	+ 5346.2763 * city=3,1,0
	+ 1383.0345 * city=1,0
	+ 1043.633 * city=0
	+ 6875.8972 * cpuload
	- 43948.0218

LM num: 7
data =
	10466.1417 * days
	+ 482.9818 * job=2,3,0
	+ 56.6047 * job=3,0
	+ 893.0995 * city=3,1,0
	+ 589.2892 * city=1,0
	+ 41308.4812 * city=0
	+ 8.4174 * cpuload
	- 5542.4818

LM num: 8
data =
	14505.3678 * days
	+ 43.7854 * job=2,3,0
	+ 60.3367 * job=3,0
	+ 287.9648 * city=3,1,0
	+ 21983.4666 * city=1,0
	+ 6177.0306 * city=0
	+ 609.8039 * cpuload
	- 10684.4067

LM num: 9
data =
	16776.3058 * days
	+ 43.7854 * job=2,3,0
	+ 60.3367 * job=3,0
	+ 29581.0655 * city=3,1,0
	+ 27853.1194 * city=1,0
	+ 14886.8994 * city=0
	- 49130.5173

LM num: 10

data =
	153.467 * days
	+ 43.7854 * job=2,3,0
	+ 60.3367 * job=3,0
	+ 1133.3536 * city=3,1,0
	+ 339.3141 * city=1,0
	+ 231.8945 * city=0
	+ 14690.7278

LM num: 11
data =
	9319.8199 * days
	+ 43.7854 * job=2,3,0
	+ 60.3367 * job=3,0
	+ 1114.0272 * city=3,1,0
	+ 339.3141 * city=1,0
	+ 231.8945 * city=0
	+ 14717.3931

LM num: 12
data =
	17945.2309 * days
	+ 43.7854 * job=2,3,0
	+ 60.3367 * job=3,0
	+ 266.0602 * city=3,1,0
	+ 178.5922 * city=1,0
	+ 33047.188 * city=0
	- 1729.532

LM num: 13
data =
	14769.0594 * days
	+ 43.7854 * job=2,3,0
	+ 60.3367 * job=3,0
	+ 137572.3837 * city=3,1,0
	+ 83354.7333 * city=1,0
	+ 64532.8573 * city=0
	- 188187.642

LM num: 14
data =
	4754.0334 * days
	+ 1589.8425 * job=2,3,0
	+ 366.2988 * job=3,0
	+ 91420.205 * city=3,1,0
	+ 52736.4266 * city=1,0
	+ 44196.4276 * city=0
	- 133849.6118

LM num: 15
data =
	307.7725 * days
	+ 1266.9926 * job=2,3,0
	+ 366.2988 * job=3,0
	+ 8518.3516 * city=3,1,0
	+ 2644.9828 * city=1,0

+ 2008.1242 * city=0
+ 27528.0238

LM num: 16
data =
6666.0892 * days
+ 1266.9926 * job=2,3,0
+ 366.2988 * job=3,0
+ 9413.8395 * city=3,1,0
+ 2644.9828 * city=1,0
+ 2008.1242 * city=0
- 16676.3277

LM num: 17
data =
10525.4738 * days
+ 1266.9926 * job=2,3,0
+ 366.2988 * job=3,0
+ 1989.8173 * city=3,1,0
+ 1315.3466 * city=1,0
+ 87775.6943 * city=0
- 18412.9833

LM num: 18
data =
4711.9675 * days
+ 1904.8437 * job=2,3,0
+ 366.2988 * job=3,0
+ 190344.7667 * city=3,1,0
+ 117683.9491 * city=1,0
+ 77188.8011 * city=0
- 283696.234

LM num: 19
data =
3107.7548 * days
+ 1375.7045 * job=2,3,0
+ 366.2988 * job=3,0
+ 290055.5534 * city=3,1,0
+ 6721.2748 * city=1,0
+ 4184.4419 * city=0
- 104498.6815

LM num: 20
data =
9759.5018 * days
+ 1375.7045 * job=2,3,0
+ 366.2988 * job=3,0
+ 4695.3122 * city=3,1,0
+ 3217.0191 * city=1,0
+ 127094.3626 * city=0
+ 6138.304

LM num: 21
data =
9601.623 * days

+ 1375.7045 * job=2,3,0
+ 366.2988 * job=3,0
+ 438291.114 * city=3,1,0
+ 258253.6274 * city=1,0
+ 197329.1076 * city=0
- 662866.3815

LM num: 22
data =
12947.9883 * days
+ 299.5337 * job=2,3,0
+ 412.1085 * job=3,0
+ 213091.6151 * city=3,1,0
+ 180444.7104 * city=1,0
+ 103679.1528 * city=0
- 281690.0314

LM num: 23
data =
12080.1896 * days
+ 299.5337 * job=2,3,0
+ 412.1085 * job=3,0
+ 438407.8629 * city=3,1,0
+ 290154.05 * city=1,0
+ 177634.3687 * city=0
- 488774.6969

LM num: 24
data =
16559.5138 * days
+ 299.5337 * job=2,3,0
+ 412.1085 * job=3,0
+ 727852.7474 * city=3,1,0
+ 421135.7392 * city=1,0
+ 301463.7055 * city=0
- 1096157.0194

Number of Rules : 24