Çevik Model Kullanan Projelere Ait Veri Seti ile İş Gücü Kestirimi

Şeyma Nur Kolak[1], Gizem Çavus[1], Muaz Gültekin[1], Oya Kalıpsız[1]

[1] Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği,
İstanbul, Türkiye
{seymanur.kolak, gizem,cavus,
muaz.gultekin}@std.yildiz.edu.tr ,oya@ce.yildiz.edu.tr

Özet. Bilgi teknoloji alanında yapılan birçok projede bilgi yetersizliğinden ve müşteri ile yeteri kadar iletişim halinde kalınmadığından dolayı birçok proje başarısızlıkla sonuçlanmaktadır. Burada önemli olan nokta projenin başlangıcında yazılımın büyüklüğünü, yazılımı geliştirmek için gereken kişi sayısını ve yazılımın maliyetini doğru bir şekilde öngörüp planlama yapabilmektir. Yazılım projelerinin kaynak tahminin doğruluğu ve güvenilirliği yazılım projesinin gidişatı için çok önemlidir. Yazılım geliştirme teknolojisinin sürekli değişen senaryolar içinde olması efor tahminini daha zorlu hale getirmektedir. Yazılım projelerinin efor tahminin doğruluğu ve güvenilirliği yazılım şirketlerinin rekabeti açısından önemlidir. Modern yazılım projeleri, yazılım geliştirme sürecinin her yinelemesi için hızlı, artan bir şekilde teslim edilmesini gerektirdiğinden, yinelemenin yürütülmesini izlemek ve iterasyon ilerledikçe kaliteli ürünler sunmak basariyi arttıracaktır. Bugüne kadar yazılım maliyet kestirimi için birçok model geliştirilmiştir. Bu modellerin çoğu yazılım geliştirme sürecinden bağımsızdır. Bu nedenle yazılım geliştirme metodolojisine bağlı olarak efor ve maliyet tahminin yapılması projenin basarisini arttıracaktır. Bu çalışmada scrum metodolojisi kullanılarak geliştirilen yazılım projeleri için maliyet tahmini yapmaya çalıştık. Geliştirdiğimiz model de gradient boosting algoritmasını kullanarak oluşturulan scrum veri seti üzerinde başarılı sonuçlar elde ettik.

Anahtar Kelimeler: Çevik Süreçler, Scrum Metodolojisi, Maliyet Kestirimi**,** Makine Öğrenmesi, Regresyon, Sınıflandırma

Software Effort Estimation in Agile Environment

Şeyma Nur Kolak[1], Gizem Çavus[1], Muaz Gultekin[1], Oya Kalipsiz[1]

[1]Yıldız Technical University, Computer Engineering,
Istanbul, Turkey

{seymanur.kolak, gizem,cavus, muaz.gultekin}@std.yildiz.edu.tr
,oya@ce.yildiz.edu.tr

Abstract. Many projects in the field of information technology are failing due to lack of information and they also fail because project decision makers are not in contact with the customer. The important point is predicting the size of the software, predicting the number of people required to develop the software and predicting the cost of the software accurately at the beginning of the project. The accuracy and reliability of the software resource estimate for the project is very important for the progress of software projects. The fact that software development technology is in constantly changing scenarios makes effort estimation more challenging. The accuracy of the effort estimates of software projects and the reliability of the software projects are so important for software companies because of the competitiveness. As modern software projects need to be delivered quickly and incrementally for each iteration of the software development process, it will be more successful to monitor the execution and provide quality products as the iteration progresses. To date, many models have been developed for software cost estimation. Most of these models are independent of the software development process. For this reason, making effort and cost estimation based on software development methodology will increase the success of the project. In this study, we tried to estimate effort for software projects which is developed in scrum environment. We have achieved successful results on the scrum dataset created using the gradient boosting algorithm.

Keywords: Agile, Scrum Methodologies, Effort Estimation**,** Machine Learning, Regression, Clustering

1 Giriş

Yazılım ölçümü, yazılım mühendisliğinin önemli bir bileşenidir. Tasarımın yüksek kalitede olup olmadığını ve uygulamaya geçişten önce, gereksinimlerin tutarlı ve eksiksiz olduğunu görebilmek için yazılımların özellikleri ölçülmelidir. Etkin proje yönetimi gerçekleştirmek amacıyla yazılım teslim süresi ve maliyet kestirimi için süreç

ve ürünlerin nitelikleri ölçülür. [1] Tahmin riski, kaynaklar, maliyet ve program için belirlenen niceliksel tahminlerdeki belirsizlik derecelendirme ile ölçülür. [3] Proje kapsamı yeterince anlaşılmamışsa veya proje gereksinimleri değişiyorsa, belirsizlik ve tahmin riski tehlikeli derecede yüksek olur.

Yazılım projeleri maliyet ve zamanlamayı aşma eğilimindedir. 2012 yılında *McKinsey* ve *Oxford* Üniversitesi tarafından, 5400 büyük ölçekli Bilgi Teknoloji (BT) projelerine yapılan çalışmada, bu projelerin ortalama %66' sının maliyet kestiriminin üzerinde olduğu ve %33' ünün planlanan zamanın üzerine çıktığı tespit edilmiştir. Yazılım projelerinin %82' sinin planlanan program ve zamanda tamamlanamadığı tespit edilmiştir. Bu tür problemlerin nedeni, proje yöneticilerinin maliyet kestirimini ve riskleri doğru yönetememesidir. [2]

Geç teslimat ve maliyet aşımları uzun yıllar yazılım projelerinde sıkça karşılaşılan sorunlardan olmuştur. Modern yazılım geliştirme, geliştiricilerinin önceki bölümlerin veya sürümlerin geliştirilmesi sırasında öğrenilenlerden faydalanmasına olanak sağlayan, artımlı ve yinelemeli bir yaklaşıma dayanmaktadır. Yazılımı artan ve yinelemeli geliştirmek, Birleştirilmiş Süreç (Rational), XP, Scrum ve diğer çevik yazılım geliştirme yöntemleri gibi birçok popüler yazılım geliştirme metodolojisinin önemli parçalarıdır [3]. Bu çalışmada, çalışma miktarı, bir iterasyonun sonunda teslim edilip edilemeyeceği konusunda teslimat yeteneğini tahmin etmek hedeflenmektedir. *Apache* ortamında geliştirilmiş açık kaynak projesine ait veri seti ile çalışma gerçekleştirilmiştir. Bu projede, her iterasyonda, bir dizi iş (konu, *issue*) tamamlanması gerekir. Projenin tamamlanmasına göre 4 ayrı zaman dilimi bulunmaktadır. Bu zaman dilimleri projenin başlangıç halindeki, projenin %30-%50-%80 lik kısmının bittiği andan sonraki iterasyonlar ve iterasyonlara ait konuları içermektedir. Bu zamanlarda yer alan iterasyonlara ait konuların hikâye noktası (story point) tahminleri ayrı ayrı hesaplanmıştır. Yazılım projelerinin maliyet kestiriminde kullanılan yöntemler makine öğrenme algoritmaları, veri madenciliği algoritmaları ve regresyon tabanlı algoritmalar olarak 3 başlık altında toplanmıştır. Bu çalışmada bir iterasyona ait konuların ölçümlerini tahmin edebilecek doğru modeller geliştirilmiştir. Tahmine dayalı modellerden regresyon tabanlı yöntemler seçilmiş ve *Stokastik Gradient* Artırma Makineleri (Gradient Boosting Maching, GBM) örneklenmiştir. Bu yöntem, hız farkını ve hikâye noktasını (story point) tahmin etmeye dayanır. Tahmin modellerini oluşturmak için hız farkı ve hikâye noktası parametreleri gerekecektir. Bu bildirinin yazım organizasyonu şu şekildedir; Bölüm 2'de; maliyet tahmini konusunda yapılan benzer çalışmalar anlatılacaktır. Bölüm 3'te; belirlenen regresyon tabanlı öğrenme algoritmalarının tanımlaması yapılacak ve bu algoritmalar üzerine optimizasyon işleminin nasıl gerçekleştirildiği açıklanacaktır. Bölüm 4'te; uygulama ve testler incelenecektir. Bölüm 5'te; başarılı bir şekilde sonlanan uygulamanın sonuçları incelenecek ve gelecek araştırmalar için tavsiyeler ele alınacaktır.

2 İlgili Çalışmalar

Yapılan çalışmaya benzer literatür taraması yapılmıştır. Yazılım geliştirmede iş gücü tahmin için makine öğrenimi algoritmaları kullanan çalışmalar ele alınmıştır.

2.1 Makine Öğrenimi Algoritmalarını Kullanarak Yazılım Geliştirmede Is Gücü Tahmin Yöntemleri

Yazılım boyutu tahmini, bir yazılım ürünü geliştirmek için gereken çabayı belirlemek için önemli bir özelliktir. Yetersiz, sorgulanabilir ve usulsüz veriler ışığında kalkınma görevlerini oluşturmak veya sürdürmek için gerekli olan en pratik efor ölçüsünü (bireysel saat ya da sermaye olarak iletilen) öngörme metodolojisidir. Yazılım Çaba Tahminleri (Software Effort Estimation-SEE), yazılımı geliştirmek veya sürdürmek için gereken çabanın en makul şekilde kullanılmasını öngörme prosedürüdür. SEE, bir yazılım projesini tamamlamak için gereken toplam çabayı tahmin etme faaliyetidir. Bu çalışmada web uygulama projeleri üzerinde iş gücü tahmini yapılmıştır.

Web uygulamalarının çabuk tahmin edilmesi için geçmiş web geliştirmenin veri kümesi projeler ISBSG veri kümesinden toplanmıştır. Benzer şekilde, çevik projeler söz konusu olduğunda, Story Point Approach (SPA), bir kullanıcı hikayesini uygulamak için gereken çabayı ölçmek için kullanılır. Bir iterasyon sırasında bitirilen kullanıcı hikayelerinin tahminlerini ekleyerek (hikâye noktası tekrarı), proje hızı elde edilir. CPA, UCP, Web ve SPA veri setleri kullanılarak elde edilen modellerin verimliliği, üzerlerinde belirli akıllı teknikler kullanılarak geliştirilebilir. Önerilen araştırma çalışması, EBM, UCP, Web üzerinden Karar Ağacı (DT), Stokastik Gradient Yükseltme (SGB), Rasgele Orman (RF) ve Destek Vektör Regresyon (SVR) kernel yöntemleri gibi çeşitli makine öğrenimi (ML) tekniklerinin uygulanmasını ele almaktadır. Bu veri kümeleri, veri kümesi üzerinde iş gücü tahmini sürecini kullanmak için içeriklerine ve alaka düzeyine göre seçilir. Makine öğrenme tekniklerini uyguladıktan sonra elde edilen çeşitli modellerin sonuçları, performanslarını değerlendirmek amacıyla, literatürdeki mevcut sonuçların yanı sıra birbirleriyle karşılaştırılmıştır. [4]

3 Metodoloji

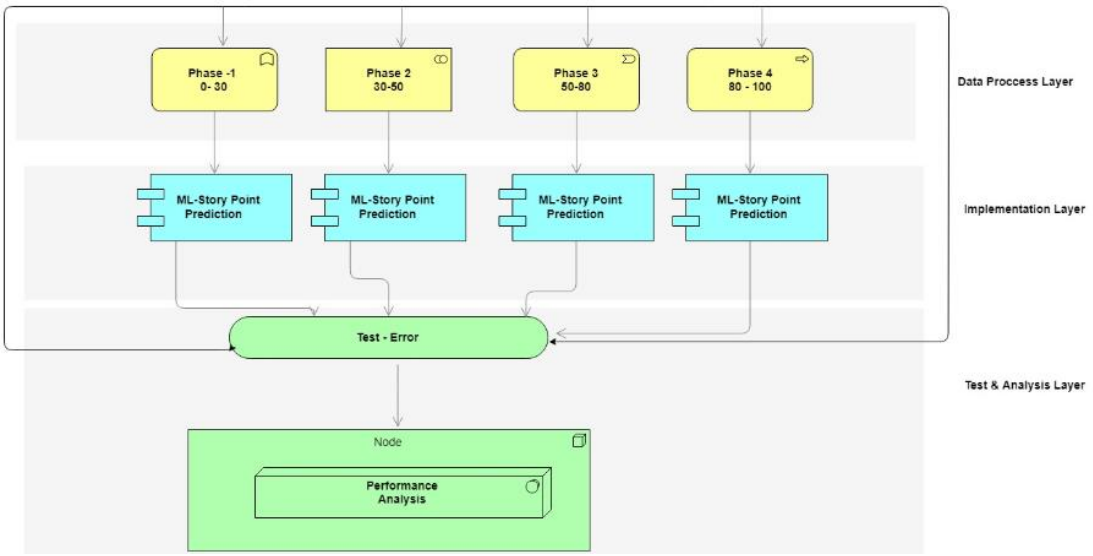
Bu çalışmada, Scrum ile geliştirilen projeye ait veri seti, iş gücü tahmini için kullanılmıştır. Scrum projelerinde geçen hikâye noktası kavramı, konuyu çözmeye yönelik tüm iş gücü, karmaşıklığını, kesinlik olmamayı ve riskleri içerir. Her bir iterasyonda bir konuya hikâye noktası atanır. Tüm iş bittiğinde o iterasyondaki tüm hikâye noktaları toplamı da hız(velocity) olarak adlandırılır. Veriler, kullanım için sınıflandırıcılar tarafından anlamlı hale getirilmiştir. Bu amaçla veri setinde bazı düzenlemeler yapılmıştır. Böylece, veri madenciliği yöntemleri ile bir model elde edilir.Geliştirilen uygulamada, model oluşturmak için veri kümesi üzerinde çalışılacak algoritmalar incelenmiştir. Bu algoritmalardan Stokastik Gradient Yükseltme (GBM) algoritması seçilmiştir. Uygulama için yazılım tasarımı Bölüm 3.1’ de, seçilen algoritma ile ilgili kısa açıklamalar Bölüm 3.2 de, veri seti tasarımı ise Bölüm 3.3’ de yer almaktadır. Sistemin genel mimarisi ise Şekil 1’ de yer almaktadır.

3.1 Yazılım Tasarımı

İş gücü tahmini için gereken tüm özellikler tek bir dosyada birleştirilir ve üzerinde algoritmalar çalıştırılır. Bu işlem için adımlar aşağıda listelenmiştir.

- Bir projenin iş gücü tahmini için gerekli olan özellikler 3 veri seti üzerinden belirlenmiştir.
- Bu seçili özellikler tek bir veri setinde birleştirildi.
- Sınıflandırmada kullandığımız özellikler bu veri setinde belirlenir.
- İş gücünü tahmin etmek için kullandığımız algoritmalar belirlenir.
- Algoritmalar bu belirtilen özellikte çalışmaya başlar.

Her bir proje için, bir tahmin tablosu elde etmek için regresyon algoritmaları 4 farklı zamanda çalıştırılmıştır. Tahmin edilen değerler test verilerindeki gerçek değerler ile karşılaştırılarak hata sonuçları elde edildi. Bu modele genel bakış Şekil-1' de gösterilmiştir.



Sekil 1. Sistemin Genel Mimarisi

3.2 Gradyan Artırma Algoritması

Gradyan güçlendirme makineleri, çok çeşitli pratik uygulamalarda önemli başarı gösteren güçlü bir makine öğrenme tekniğidir. Farklı kayıp fonksiyonlarına göre öğrenmek gibi, uygulamanın belirli ihtiyaçlarına göre özelleştirilebilir. Teorik bilgi, gradyan artırma model tasarımının tüm aşamalarını kapsayan betimsel örnekler ve çizimlerle tamamlanır. Gradyan artırma, tahmin modelleri oluşturmak için en güçlü tekniklerden biridir.

Rasgele Orman algoritması bağımsız karar ağaçları yetiştirir (bu da paralel olarak yapılabilir) ve sadece bu ağaçların ürettiği tahminlerin ortalamasını nihai tahmin olarak alır. Öte yandan, gradyan takviye makineleri ağaçları üretir ve bunları sıralı bir şekilde topluluğa ekler. İlk ağaç, Rasgele Ormanlarda yapıldığı gibi üretilir. Buradaki temel fark, ilk ağacın ürettiği tahmin hatalarını en aza indirmeyi amaçlayan ikinci ağacın üretilmesidir. Hem birinci hem de ikinci ağaçlar topluluğa eklenir ancak her birine farklı ağırlıklar verilir. Bu süreç birden çok kez tekrarlanır: her adımda, öğrenilen tüm topluluğun hatalarına göre yeni bir ağaç eğitilir ve daha sonra topluluğa eklenir.

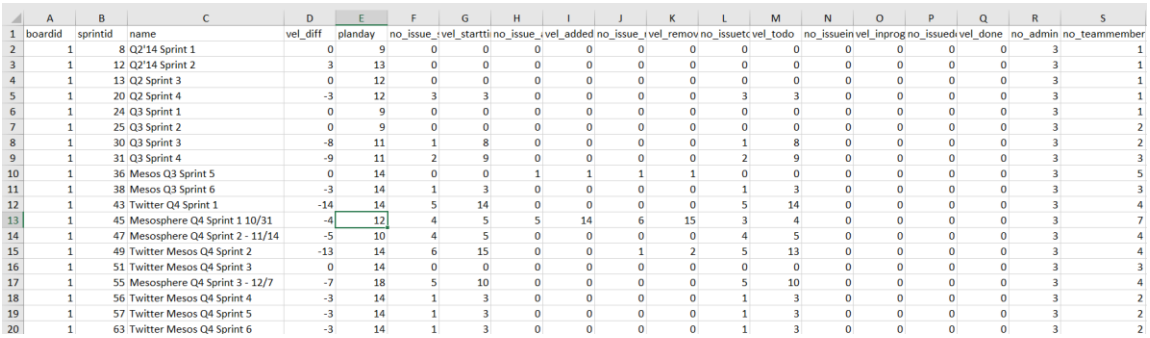
Son topluluk, yeni girdilerin sonucunu kestirmek için bir model olarak kullanılır. Rasgele Ormanların aksine, GBM'lerde düşük bir öğrenici sadece regresyon ağaçları değil, sinir ağları veya doğrusal regresyon gibi diğer regresyon öğrenme algoritmaları da olabilir. Uygulamada regresyon ağaçları düşük öğreniciler olarak kullanılmış ve 100 ağaç üretilmiştir

3.3 Veri Seti Tasarımı

Scrum Veri Kümesi, Apache'den gelen veri kümesini içerir. Bu veri kümesi .csv uzantılı dosya türünde tutulur. Her proje için 3 ayrı veri seti mevcuttur. Bunlardan ilki, her iterasyona ait konuların genel özelliklerini ve hikâye noktası bilgisini tutar. İkinci veri seti, projedeki her bir iterasyonun özellikleri ve hız farkı (velocity different) değeri tutulur. Sonuncusunda ise konular arasındaki ilişkiyi gösteren değerler mevcuttur. Bu veri setlerinin örnekleri Şekil 2, 3 ve 4'te gösterilmiştir.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	boardid	sprintid	type	priority	no_comm	no_affect	no_fixvers	no_issueli	no_blockir	no_blocke	no_fixvers	no_priorit	no_des_ch	gunning_fog
2	1	8	Task	Minor	1	0	0	0	0	0	1	0	0	medium
3	1	8	Improvem	Major	1	0	0	0	0	0	1	0	0	medium
4	1	8	Improvem	Major	1	0	0	2	0	0	1	0	0	medium
5	1	12	Bug	Major	0	1	0	1	0	0	0	0	0	hard
6	1	12	Bug	Minor	0	0	0	1	0	0	1	0	0	hard
7	1	13	Bug	Minor	0	0	0	0	0	0	0	0	0	medium
8	1	20	Bug	Major	0	0	5	4	0	0	0	0	0	medium
9	1	20	Task	Minor	1	0	0	0	0	0	2	0	0	medium
10	1	20	Improvem	Major	1	0	0	0	0	0	2	0	0	medium
11	1	20	Bug	Minor	3	0	0	0	0	0	2	0	0	medium
12	1	25	Bug	Major	1	1	0	0	0	0	1	0	0	hard
13	1	25	Improvem	Major	4	0	0	0	0	0	0	0	0	hard
14	1	25	Bug	Major	0	0	0	0	0	0	9	0	0	hard
15	1	30	Bug	Major	2	0	0	4	0	0	0	0	0	hard

Sekil 2. Apache Issue Veri Seti



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	boardid	sprintid	name	vel_diff	planday	no_issue_s	vel_startti	no_issue_a	vel_added	no_issue_r	vel_remov	no_issuetc	vel_todo	no_issuein	vel_inprog	no_issued	vel_done	no_admin	no_teammember
2	1	8	Q2'14 Sprint 1	0	9	0	0	0	0	0	0	0	0	0	0	0	0	3	1
3	1	12	Q2'14 Sprint 2	3	13	0	0	0	0	0	0	0	0	0	0	0	0	3	1
4	1	13	Q2 Sprint 3	0	12	0	0	0	0	0	0	0	0	0	0	0	0	3	1
5	1	20	Q2 Sprint 4	-3	12	3	3	0	0	0	0	3	3	0	0	0	0	3	1
6	1	24	Q3 Sprint 1	0	9	0	0	0	0	0	0	0	0	0	0	0	0	3	1
7	1	25	Q3 Sprint 2	0	9	0	0	0	0	0	0	0	0	0	0	0	0	3	2
8	1	30	Q3 Sprint 3	-8	11	1	8	0	0	0	0	1	8	0	0	0	0	3	2
9	1	31	Q3 Sprint 4	-9	11	2	9	0	0	0	0	2	9	0	0	0	0	3	3
10	1	36	Mesos Q3 Sprint 5	0	14	0	0	1	1	1	1	0	0	0	0	0	0	3	5
11	1	38	Mesos Q3 Sprint 6	-3	14	1	3	0	0	0	0	1	3	0	0	0	0	3	3
12	1	43	Twitter Q4 Sprint 1	-14	14	5	14	0	0	0	0	5	14	0	0	0	0	3	4
13	1	45	Mesosphere Q4 Sprint 1 10/31	-4	12	4	5	5	14	6	15	3	4	0	0	0	0	3	7
14	1	47	Mesosphere Q4 Sprint 2 - 11/14	-5	10	4	5	0	0	0	0	4	5	0	0	0	0	3	4
15	1	49	Twitter Mesos Q4 Sprint 2	-13	14	6	15	0	0	1	2	5	13	0	0	0	0	3	4
16	1	51	Twitter Mesos Q4 Sprint 3	0	14	0	0	0	0	0	0	0	0	0	0	0	0	3	3
17	1	55	Mesosphere Q4 Sprint 3 - 12/7	-7	18	5	10	0	0	0	0	5	10	0	0	0	0	3	4
18	1	56	Twitter Mesos Q4 Sprint 4	-3	14	1	3	0	0	0	0	1	3	0	0	0	0	3	2
19	1	57	Twitter Mesos Q4 Sprint 5	-3	14	1	3	0	0	0	0	1	3	0	0	0	0	3	2
20	1	63	Twitter Mesos Q4 Sprint 6	-3	14	1	3	0	0	0	0	1	3	0	0	0	0	3	2

Sekil 3. Apache Iterasyon Veri Seti

	A	B	C	D
1	idissuelink	issuekey	linkedIssueKey	name
2	1	DRILL-1274	DRILL-1267	contains
3	2	DRILL-1274	DRILL-1267	is duplicated by
4	3	DRILL-1277	DRILL-1206	contains
5	4	DRILL-1278	DRILL-1288	duplicates
6	5	DRILL-1271	DRILL-1240	duplicates
7	6	DRILL-1267	DRILL-1274	Is contained by
8	7	DRILL-1267	DRILL-1274	duplicates
9	8	DRILL-1245	DRILL-1243	is superceded by
10	9	DRILL-1240	DRILL-1271	is duplicated by
11	10	DRILL-1206	DRILL-1277	Is contained by
12	11	DRILL-1185	DRILL-1451	is duplicated by
13	12	DRILL-891	DRILL-1039	is duplicated by
14	13	TEZ-722	TEZ-743	relates to
15	14	JENA-621	JENA-525	relates to
16	15	TEZ-716	TEZ-68	is duplicated by
17	16	TEZ-717	TEZ-1304	depends upon
18	17	TEZ-717	TEZ-1298	depends upon
19	18	HELIX-349	HELIX-281	duplicates
20	19	HELIX-349	HELIX-281	is related to
21	20	TEZ-718	TEZ-761	relates to
22	21	CB-5747	CB-6969	relates to
23	22	TWILL-29	TWILL-18	is duplicated by
24	23	JENA-619	JENA-685	relates to
25	24	TEZ-705	TEZ-698	is depended upon by

Sekil 4. Apache Issue link Veri Seti

Şekil 5 ‘te regresyon tabanlı makine öğrenmesi tekniklerinin denendiği veri seti görülmektedir. Burada dikkat edilmesi gereken nokta tüm özelliklerin nümerik olması gerektiğidir. Bu sebeple de Şekil 6’da görülen işlemler yapılarak veri setinde düzenlemeye gidilmiştir. Bu veri setine ait özelliklerin bir projede nasıl kullandıklarını anlamak önemlidir. Bunun için veri setinin özellikleri ile ilgili açıklamalar aşağıda maddeler halinde açıklanmıştır.

- *Type*: Her bir konuya, konunun çözülmesiyle ilgili görevin niteliğini belirten bir tür (ör. Görev (Task), Hata (Bug), Yeni özellik (new feature), İyileştirme (Improvement) ve İşlev Testi (Function Test)) atanır.
- *Priority: Konunun* önceliğini, bir konunun diğer konulara ilişkin olarak katılması gereken sırayı göstermektedir. Örneğin, engelleyici öncelikli konular,

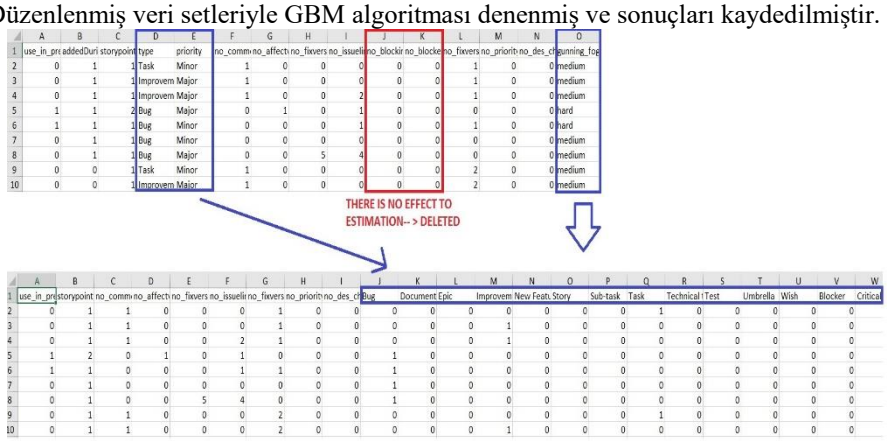
büyük veya küçük öncelikli konulardan daha fazla endişelendirmelidir. Engelleyici önceliği, tamamlanacak diğer sorunları engelleyen bir sorundur. Bu çalışmada, beş öncelik seviyesini dikkate alınmıştır: önemsiz, küçük, büyük, kritik ve engelleyici.

- *Number of Comment*:Tahmin sırasında geliştiricilerin yorum sayısı ekip işbirliği derecesinin göstergesidir.
- *Number of FixVersion*: Her bir konudaki düzeltme sürümü alanı, sorunun giderildiği (veya alınacağı) sürümü gösterir. Çok sayıda düzeltme sürümü olan sorunlar, geliştirme, test etme ve bütünleştirme açısından daha fazla dikkat gerektirir.
- *Number of Affect Version*: Bir konunun bulunduğu sürümleri belirtir.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	use_in_pre	addedDuri	storypoint	type	priority	no_comm	no_affecti	no_fixvers	no_issueli	no_blockir	no_blocke	no_fixvers	no_priorit	no_des_ch	gunning_fog
2	0	1	1	Task	Minor	1	0	0	0	0	0	1	0	0	medium
3	0	1	1	Improvem	Major	1	0	0	0	0	0	1	0	0	medium
4	0	1	1	Improvem	Major	1	0	0	2	0	0	1	0	0	medium
5	1	1	2	Bug	Major	0	1	0	1	0	0	0	0	0	hard
6	1	1	1	Bug	Minor	0	0	0	1	0	0	1	0	0	hard
7	0	1	1	Bug	Minor	0	0	0	0	0	0	0	0	0	medium
8	0	1	1	Bug	Major	0	0	5	4	0	0	0	0	0	medium
9	0	0	1	Task	Minor	1	0	0	0	0	0	2	0	0	medium
10	0	0	1	Improvem	Major	1	0	0	0	0	0	2	0	0	medium

Sekil 5. Kategorik Veri Seti

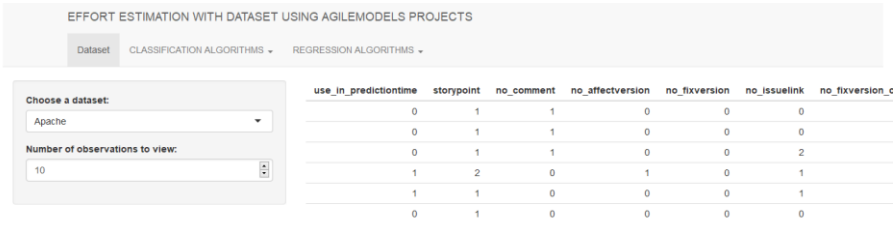
Düzenlenmiş veri setleriyle GBM algoritması denenmiş ve sonuçları kaydedilmiştir.



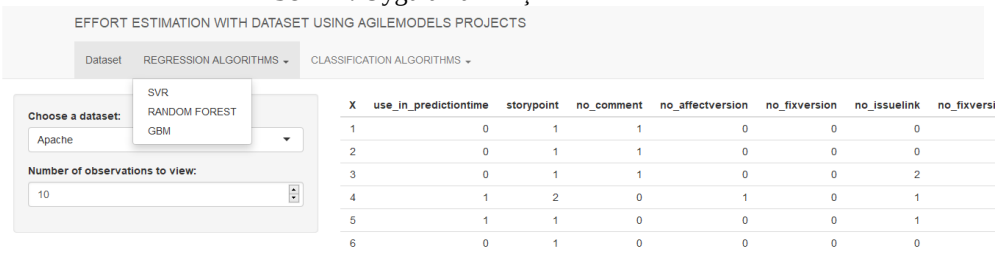
Sekil 6. Kategorik Verinin Düzenlenmiş Hali

4 Uygulama ve Test

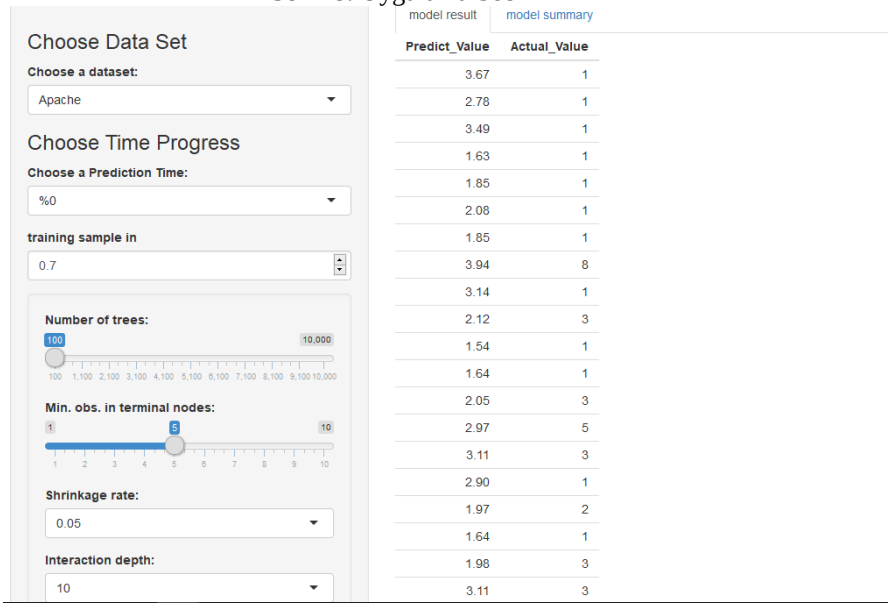
We Çalışmanın uygulaması R dili gerçekleştirilmiştir. Uygulamanın çalıştırılması için en az 2GB RAM ve 400MB Disk alanına ihtiyaç duyulmaktadır. Kullanıcı ara yüzü Şekil 7, Şekil 8 ve Şekil 9'da gösterilmiştir.



Sekil 7. Uygulama Giriş Ekranı



Sekil 8. Uygulama Secim Ekranı



Sekil 9. GBM Algoritmasının Sonuç Ekranı

Uygulamanın ilk açılan sayfası Şekil 7' de verilmiştir. Burada veri setinin içeriği görülmektedir. Daha sonra regresyon algoritma bölümünden GBM algoritması seçilir. GBM algoritmasının parametreleri kullanıcı tarafından girilecek şekilde ayarlanmıştır. Seçilen veri setinin 0-30-50-80 için ayrı ayrı veri setleri üzerine algoritma çalıştırılmıştır. Veri setinin ne kadarlık kısmı eğitim ne kadarlık kısmı test için kullanılacağı da kullanıcı tarafından belirlenecektir. En ideal olarak %70' i eğitim, %30' u test olarak seçilmiştir. Ağaç sayısı olarak 100 ağaç türetilmiştir. Her bir düğümdeki yaprak sayısı

5 olarak, öğrenme oranı 0.05, ağaç derinliği de 10 olarak seçilmiştir. Bu değerlere göre algoritmanın ürettiği hikâye noktaları Şekil 9' da görülmektedir.

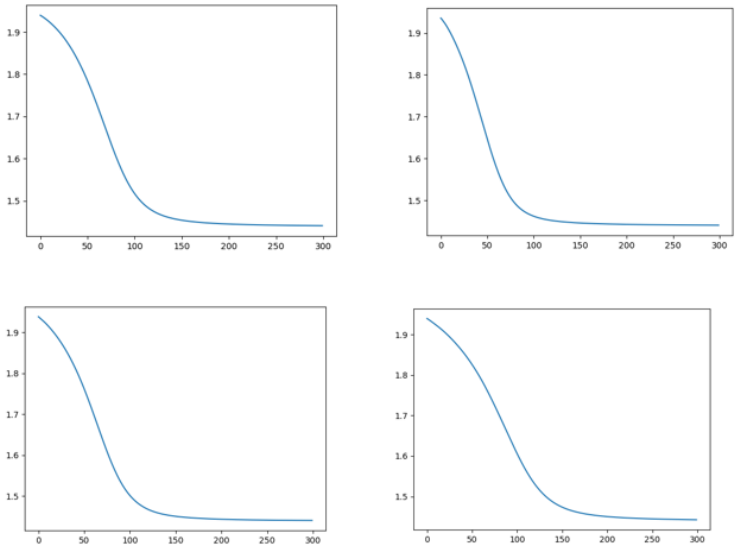
5 Sonuçlar ve Yapılacak Çalışmalar

Uygulanan GBM algoritmasının sonuçları Şekil 10'daki gibidir

DATASET	Prediction Time(%)	GBM
APACHE	0	1.62
	30	1.63
	50	1.7
	80	1.61

Sekil 10. GBM Algoritmasının RMSE Değerleri

Burada RMSE (Root Mean Square Error) değerleri esas alınmıştır. Regresyon tabanlı makine öğrenmesi teknikleri uygulanması durumunda doğruluk oranlarından çok hata oranlarının karesel değerleri bizi daha doğru sonuçlara götüreceğinden dolayı bu değerler kullanılmıştır. Şekil 11'de Apache ortamından elde edilmiş projelere ait RMSE değerlerinin zamanla iterasyona bağlı artışları görülmektedir.



Sekil 11. Apache Veri Seti Üzerinde RMSE Gösterimi

Bu grafiklerden de görüldüğü üzere RMSE değerleri iterasyon sayısı arttıkça düşmektedir. Eğitim veri setleriyle kendini eğiten sistem zamanla daha iyi öğrenir ve daha doğru tahminlerde bulunur. Burada işlemler 300 iterasyona kadar test edilmiştir. Bir

süre sonra sistemin eğitiminin durduğu iterasyon sayısının arttırımının gereksiz olduğu gözlenmiştir

Gelecek çalışmalarda regresyon tabanlı makine öğrenme tekniklerine devam edilebilir aynı zamanda Yapay Sinir Ağları modelleriyle de testler yapılabilir. Bir sonraki çalışma olarak Deep Learning ile Yapay Sinir Ağları teknikleri kullanılması öngörülmüştür

Kaynaklar

1. B. Flyvbjerg and A. Budzier “Why Your IT Project May Be Riskier Than You Think”, Harvard Business Review, 2011
2. B. Michael, S. Blumberg, and J. Laartz, “Delivering large-scale IT projects on time, on budget, and on value,” Tech. Rep., 2012.
3. Olcaysoy Buharalı, A. Kalıpsız, O.: Bilişim Projelerinde Yazılım Risk Yönetimi: Telekomünikasyon Örneği, 2015
4. Shashank Mouli Satapathy, Effort Estimation Methods in Software Development using Machine Learning Algorithm(2016)
5. L. Williams,"What agile teams think of agile principles”, Communications of the ACM,2012
6. M. Cohn,"Agile estimating and planning", 2005, Pearson Education.
7. Brischoux, Francois and Legagneux, Pierre "Don't Format Manuscripts”, The Scientist,2009
8. Roger S. Pressman ,"Software Engineering A Practitioner's Approach", McGraw.Hill International Edition
9. https://en.wikipedia.org/wiki/Gradient_boosting
10. https://www.cprime.com/resources/what-is-agile-what-is-scrum/
11. http://www.statisticshowto.com/rmse/