Makine Öğrenmesi Algoritmalarına Dayalı Yazılım Hata Tahmini: Çevresel Metriklerin Etkisi

Bu tez Bilgi Güvenliği Mühendisliği'nde
Tezli Yüksek Lisans Programının bir koşulu olarak

Merve ODABAŞI

tarafından

Fen Bilimleri Enstitüsü'ne
sunulmuştur.

İSTANBUL
ŞEHİR
ÜNİVERSİTESİ

Bu tezi okuduk, kapsam ve nitelik açısından Bilgi Güvenliği Mühendisliği alanında Yüksek Lisans derecesi için tümüyle uygun olduğu görüşüne vardık.

ONAYLAYANLAR:

Prof. Dr. Ensar GÜL .....
(Tez Danışmanı)

Prof. Dr. Selim ZAİM .....

Prof. Dr. Nizamettin BAYYURT .....

Bu tez İstanbul Şehir Üniversitesi, Fen Bilimleri Enstitüsü tarafından belirlenen tüm koşullara uygundur.

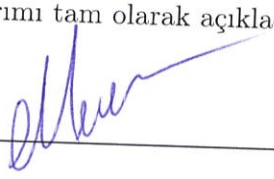
ONAY TARİHİ: 08.07.2019

MÜHÜR/İMZA:

Yazarlık Beyanı

Ben, Merve ODABAŞI, başlığı,'Makina Öğrenmesi Algoritmalarına Dayalı Yazılım Hata Tahmini: Çevresel Metriklerin Etkisi' olan tezin ve içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu tezin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Tezin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Esaslı yardım aldığım bütün kaynaklara teşekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

İmza: 

Tarih: 08.07.2019

Software Fault Prediction Based on Machine Learning Algorithms: The Effect of Environmental Metrics

Merve ODABAŞI

Abstract

According to the Word Quality report, the use of machine learning studies in the industrial field is rare due to the severe lack of quality data. Software metrics are generally utilized during software fault prediction in this field. In this study, besides the software metrics, the environmental metrics are also explored to see whether they also affect the results of machine learning and if they could, what would be the success rates and which environmental metrics are more effective on the results. The data set for this study was generated from combining various data from 10 projects that were produced a team of 4 analysts, 8 software engineers, and 5 test experts. 36 metrics and 6676 test cases in total were evaluated. The errors occurred in the test cases are not just considered as an error, their priority and cases that can not be tested are also taken into consideration. 9 fault level are employed in models. During the pre-processing phase, the PCA analysis is conducted, out of 36 metrics 12 are effective to the results. Models are created with four different algorithms which have achieved a success rate of; 89% by the decision tree algorithm, 87% by the nearest neighbors algorithm, 88% by the random forests algorithm and 91% by the Naive Bayes algorithm. In conclusion, it was observed that environmental metrics are indeed effective in software fault prediction and when applied with machine learning algorithms a high rate of success can be achieved. ...

Keywords: Software Testing, Software Fault Prediction, Machine Learning, Environmental Metrics

Makine Öğrenmesi Algoritmalarına Dayalı Yazılım Hata Tahmini: Çevresel Metriklerin Etkisi

Merve ODABAŞI

Öz

Bu çalışmada 2012-2019 yılları arasında makine öğrenmesi algoritmaları ile yazılım hata tahminleme konusunda, akademik ve sektörel alanda yapılan çalışmalar incelenmiştir. Yapılan inceleme sonucunda akademik alanda, yazılım test alanında makine öğrenmesi çalışmalarının trend olduğu görülmüştür. Word Quality Report'ta kaliteli verinin olmamasından dolayı sektörde makine öğrenmesi çalışmalarının kullanımının az olduğu belirtilmiştir. İlgili çalışmalarda, tahminlemeler esnasında sadece yazılımsal metriklerin kullanıldığı görülmüştür. Bu çalışmada ise, Yazılımsal metriklerin yanında, çevresel metriklerin de sonuç üzerinde etkili olup olmadığı, çevresel metriklerin kullanılarak makine öğrenmesi algoritmaları ile başarılı sonuç tahminleme yapılıp yapılamayacağı, hangi çevresel etkenlerin sonuç üzerinde ne kadar etkili olduğu araştırılmıştır. Çalışma için veri seti, 4 analist, 8 yazılım mühendisi ve 5 test uzmanından oluşan bir ekibinin yapmış olduğu 10 adet projenin çıktıları ile oluşturulmuştur. 36 adet metrik ile 6676 adet test durumu değerlendirilmiştir. Hatayı sadece hata olarak kabul etmenin yanında, hatanın önem derecesi ve test edilemeyecek olan senaryolarda hesaba katılmıştır. Toplamda dokuz adet sonuç tahminlemesi modelde kullanılmıştır. Ön işleme aşamasında PCA analizi yapılmış, 36 adet metrik içerisinden 12 adedinin sonuca etkili olduğu görülmüştür. Dört farklı algoritma ile modeller oluşturulup; karar ağacı algoritmasında 89%, en yakın komşular algoritmasında 87%, rastgele ormanlar algoritmasında 88% ve naive bayes algoritmasında 91% başarı elde edilmiştir. Sonuç olarak çevresel metriklerin de yazılım test sonucu tahminlemesinde etikili olduğu ve makine öğrenmesi algoritmaları ile kullanıldığında, yüksek oranda başarılı sonuç tahminlemesi yapılabileceği görülmüştür.

Anahtar Sözcükler: Yazılım Test, Makine Öğrenmesi, Çevresel Metrikler, Yazılım Test Trendleri, Yazılım Hata Tahmini

Teşekkür

Bu çalışmanın gerçekleştirilmesinde, üç yıl boyunca değerli bilgilerini bizlerle paylaşan saygıdeğer ve çok kıymetli; Prof. Dr. Ensar GÜL'e, çalışmam boyunca benden bir an olsun yardımlarını esirgemeyen arkadaşlarım; Abdulvehhab Ağın, Mustafa Aker, Okan Konur ve Yahya Koç'a, çalışma süresince tüm zorlukları benimle göğüsleyen ve hayatımın her evresinde bana destek olan değerli eşim Volkan Odabaşı'na sonsuz teşekkürlerimi sunarım...

İçindekiler

Şekil Listesi

Tablo Listesi

Kısaltmalar

ISTQB	**I**nternational **T**esting **Q**ualifications **B**oard
SDLC	**S**oftware **D**evelopment **L**ife **C**ycle
WQR	**W**orld **Q**uality **R**eport
TTB	**T**urkish **T**esting **B**oard
IOT	**I**nternet **O**f **T**hings
PCA	**P**rincibal **C**omponent **A**nalysis
PC	**P**rincibal **C**omponent

Bölüm 1

Giriş

Dünya Kalite Raporu 17-18 ve Test Durumu Anketi 2017-2018'e göre, yazılım dünyasın yeni ilgi alanları; yazılım test otomasyon, daha kısa yazılım yaşam döngüleri, mobil hibrit uygulamalar ve makine öğrenmesidir [1] [2] [3] [4]. Günümüzde, yazılım dünyasında her alanda makine öğrenmesi ile ilgili çalışmalar yapılmaktadır. Sosyal ağlardan, internet arama kayıtlarımızdan kişisel bilgilerimiz toplanarak gösterilecek ilgili reklamları seçmek için makine öğrenmesi kullanılmaktadır. Telefonumuzun açık olan GPS özelliği sayesinde konum bilgilerimizden iş yeri bilgilerimiz, hangi saatte evden çıktığımız, hangi saatte eve gittiğimiz gibi bilgilerle, bize ilgili saatlerde rota tavsiyesi bildirimleri gönderilmektedir. Makine öğrenmeleri artık günlük hayatımızın büyük bir parçası haline gelmiştir. Makine öğrenmesi, sistemlere insan müdahalesi veya açık programlama olmadan otomatik olarak öğrenme yeteneği sağlamak için yapay zekâyı uygulamakta ve çok büyük miktardaki verilerin analizini sağlamaktadır. Yazılım test dünyasında ise, makine öğrenmesi teknolojisinden beklenen, test sonuçları ve test otomasyon deneyimlerinden öğrenmesi, verilere otomatik olarak erişmesi ve onunla testler yapmasıdır. Bu döngü bittiğinde de sonuçları yeniden öğrenmesi ve test döngüsünü iyileştirmesidir.

Makine öğrenmesi, daha kısa sürede daha doğru sonuçlar vermektedir. Bu ve bunun gibi nedenlerden dolayı yazılım test ve kalite değerlendirme süreçlerinde, makine öğrenmesinin çok daha fazla kullanılması beklenmektedir[5]. Test uzmanlarının bu değişimi yakalayabilmeleri, becerilerinin teknolojik ilerlemelerle birlikte gelişmesi gerekmektedir. Bu becerilere örnek verecek olursak; ileri makine öğrenimi ve yapay zekâda test etme, DevOps'u destekleyen kalite mühendisliği, Performans mühendisliği, Bulut tabanlı test

araçları kullanımı, Dijital dönüşüm, IoT cihaz testleri, Güvenlik testi, Büyük veri ve veri analitiğinin test edilmesi, Mobil Ödeme ve Güvenliğin Test Edilmesi vb[6].

Makine öğrenmesi yetenekleri kullanılarak, yazılım test süreci gelişimine katkıda bulunma, yazılım test sonuç tahminleme yapma konularında günümüze kadar birçok çalışma yapılmıştır. Yapılan çalışmalar incelendiğinde, sonuca etki eden metriklerin seçiminde yazılımsal metriklerin seçildiği görülmüştür. Yazılımsal metriklerin yanında, çevresel metriklerin de sonuç üzerinde etkili olup olmadığı, çevresel metriklerin kullanılarak makine öğrenmesi algoritmaları ile başarılı sonuç tahminleme yapılıp yapılamayacağı, hangi çevresel etkenlerin sonuç üzerinde ne kadar etkili olduğu soruları üzerinde çalışma yapılmıştır.

Yazılım test sürecinde, hataya ne kadar erken ulaşılırsa, çözüm maliyeti o kadar düşük olmaktadır. Makine öğrenmesi yetenekleri kullanılarak, yazılım hata tahminlemesi yapıldığında hata alınması en muhtemel olan senaryodan başlanarak, hataya ulaşma süresi en aza indirilmektedir. Yazılım henüz başlamadan, test durumları belirlenip tahminleme yapılır ise, hata alınması en muhtemel senaryoların yazılımlarında en yüksek dikkat gösterilerek hatanın daha yapılmadan önlenmesi mümkün olmaktadır. Yöneticiler tarafından ekip oluşturma aşamasında, ilgili tahminleme yapılarak, en uygun ekip oluşturulması sağlanmaktadır.

Word Quality Report'a göre makine öğrenmesi çalışmalarının sektörde kullanımının az olmasının sebebi kaliteli verinin olmamasıdır. Bu nedenle bu çalışmada hazır veri kullanılmamış, yeni veri oluşturulmuştur. Yeni veri oluşturulmasında yazılım yaşam döngüsü uygulanan bir ekibin on adet projesi ve 6676 adet durum incelenmiştir. Çalışma esnasında toplamda 36 adet metrik değerlendirilmiştir. Test sonucu olarak sadece başarılı-başarısız değerlendirmesi yerine, hata seviyelerine göre değerlendirme yapılmıştır. Bunun yanı sıra test edilemeyecek olan senaryolar da tahminlemeye dâhil edilmiştir. Toplamda 9 adet sonuç tahminlemesi modelde kullanılmıştır.

Modelleme aşamasında dört adet teknik kullanılmıştır. Algoritmalar, günümüzde en çok kullanılan sınıflandırma algoritmalardan seçilmiştir. Karar Ağacı Uygulaması, En Yakın Komşular Algoritması, Rastgele Ormanlar Algoritması ve Naive Bayes Algoritması kullanılmıştır. Oluşturulan modellerin başarı yüzdeleri hesaplanmıştır.

İkinci bölümde, bu güne kadar yapılmış, ilgili çalışmalar hakkında bilgi verilmiştir. Üçüncü bölümde, yazılım test süreçleri, test stratejileri, yazılım test türleri, yazılım test teknikleri, yazılım test yaşam döngüsü hakkında bilgiler verilmiştir. Yine aynı bölümde kronolojik olarak, Türkiye ve Dünya test eğilimleri incelenmiştir. Dördüncü bölümde, makine öğrenmesi ile ilgili kavramlar incelenerek, ön işleme, karar ağacı algoritması, en yakın komşular algoritması, rastgele ormanlar algoritması ve naive bayes algoritmaları hakkında bilgi verilmiştir. Beşinci bölümde, yapılan uygulama, verilerin toplanma aşaması, analiz etme aşaması, işleme ve algoritmalar ile değerlendirme aşamasına yer verilmiştir. Altıncı bölümde ise sonuçlara yer verilmiştir.

Bölüm 2

İlgili Çalışmalar

Mevcut erken sorun giderme çabası, önemsiz ve ciddi problemler arasında eşit bir şekilde aynıdır. Adam A. Porter and Richard W. Selby'in 1990 yılında yapmış olduğu çalışmada, bunu bir sınıflandırma problemi yapan çözümü incelemişlerdir. Önerilen yaklaşım, ölçülebilir niteliklerine ve gelişim süreçlerine dayanarak, sorunlu bileşenlerin modellerini türetmektedir. Bu ölçüme dayalı modeller, hangi bileşenlerin büyük olasılıkla hataya yatkın olma veya yüksek geliştirme maliyetine sahip olma gibi yüksek riskli özellikleri paylaşabileceğini tahmin etmek için bir temel sağlamaktadır. Ortaya çıkan modellerin oluşturulması ve yorumlanması için Sınıflandırma Ağacı'nı veri olarak ise Nasa ve Hughes Hava Yolları'nın verilerini kullanmışlardır. Yöntem düzeyindeki ölçümler kullanılmış, verilerle 79,3% başarılı bir model oluşturmuşlardır [7].

Yazılım hatası tahmini için şu anda mevcut olan regresyon ağacı algoritmalarının kapsamlı değerlendirmesinden (birkaç büyük vaka çalışmasıyla birlikte) bir vaka çalışması sunulmaktadır. Taghi M. Khoshgoftaar Naeem Seliya, Dizayn ölçümlerini dikkate alarak şu algoritmaları kullanmışlardır: CART-LS (en küçük kareler), S PLUS ve CARD-LAD (en az mutlak sapma). Sunulan vaka çalışması, yaklaşık 13 milyon kod satırından oluşan geniş bir ağ telekomünikasyon sisteminden toplanan yazılım tasarım ölçümlerinden oluşmaktadır [8].

Yazılım metriklerine dayalı kalite sınıflandırma modelleri, bir yazılım modülünü hataya eğilimli veya hataya eğilimli değil olarak tahmin etmektedir. Bu tür modellerin zamanında uygulanması, yazılım kalite testi ve geliştirme kaynaklarının maliyeti etkin bir şekilde kullanılmasıyla kalite geliştirme çabalarının operasyonlar sırasında hataya meyilli

olması muhtemel olan modüllere yönlendirilmesine yardımcı olabilmektedir. Khoshgoftaar ve Seliya tarafından yedi sınıflandırma tekniğinin ve / veya araçlarının göreceli performanslarının kapsamlı bir değerlendirmesi yapılmıştır. Bunlar lojistik regresyon, vaka temelli muhakeme, sınıflandırma ve regresyon ağaçları (CART), S-PLUS ile ağaç bazlı sınıflandırma ve Sprint-Sliq, C4.5 ve Treedisc algoritmalarını içermektedir. Beklenen yanlış sınıflandırma maliyetinin kullanımı, farklı yazılım kalitesi sınıflandırma modellerinin performanslarını karşılaştırmak için tekil ve tek bir ölçüm olarak sunulmuştur [9].

Büyük bir yazılım sisteminin bir sonraki sürümünde hangi dosyaların en çok sayıda hata içerdiği konusunda bilgi sahibi olmak çok değerli bir varlık olabilir. Thomas J. Ostrand, Elaine J. Weyuker, and Robert M. Bell bunu başarmak için, bir sistemin, bir sonraki sürümünün, her bir dosyasındaki beklenen hata sayısını tahmin etmek için negatif bir binom regresyon modeli geliştirerek kullanmıştır. Tahminler, geçerli sürümdeki dosyanın koduna ve dosyanın önceki sürümlerdeki hata ve değişiklik geçmişine dayanmaktadır. Model, biri 4 yıl boyunca ardı ardına 17 sürüm yayınlayan, diğeri 2 yıl boyunca dokuz sürüm yayınlayan iki büyük endüstriyel sisteme uygulanmıştır. İki sistemin her sürümünde, tahmin edilen en yüksek hata sayısına sahip dosyaların yüzde 20'si, tespit edilen hataların 71% ile 92% 'sini oluştururken, toplam ortalama 83% olmuştur [10].

Hata ciddiyetini dikkate alırken hatayı tahmin etmede, özellikle Chidamber ve Kemerer grubunun bir alt kümesi olan nesne yönelimli tasarım ölçümlerinin faydasını araştırmak için lojistik regresyon ve makine öğrenme yöntemlerini kullanılmış, NASA veri seti ile çalışılmıştır. Hatayı sadece hata olarak kabul etmenin yanında, hatanın önem derecesini de hesaba katılmıştır. Yuming Zhou and Hareton Leung, Naive Bayes, Rastgele Ormanlar, En Yakın Komşular algoritmaları kullanılmıştır. Düşük öneme sahip hataların tahmin edilmesinin, yüksek önemli hataların tahmin edilmesinden daha kolay olduğunu ortaya koymuşlardır [11].

1990-2009 yılları arasında, Cağatay Çatal, yayınlanan ve Yazılım Hata Öngörme hakkında olan makaleleri incelemiş, yapılan çalışmalar hakkında özet bilgiler vermiştir [12].

Makine öğrenmesi algoritmalarına dayalı yazılım hata tahmini modeli sunulmaktadır. Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, Fatima Alsarayrah, bu çalışmada üç metrik ile tahminleme çalışması yapılmıştır. Bir modülün arızalarının sayısı, test uzmanlarının sayısı, yazılım projesinin ne kadar sürdüğü bilgileri ile oluşturulan veri

seti üzerine çalışılmıştır. Naive Bayes, Yapay zeka ağları ve Karar Ağacı algoritmalarını kullanmışlardır. 89% ile 99% arasında değişen başarı elde etmişlerdir [13].

Kullanılan yazılım dilinin sonuca etki edip etmediği araştırılmıştır. C / C ++ , Java ve Python dilleri ile aynı çalışmaları tekrar etmiş ve sonuçları karşılaştırmıştır. Asıl amaç ise Python projeleri için umut verici sonuçlar elde edilip edilemeyeceğini bulmaktır. Bartłomiej Wójcicki, Robert Dabrowski sonuç olarak ise, Python dilinin diğer diller kadar başarılı sonuçlar verdiğini görmüşlerdir. Veri seti olarak NASA'nın veri setlerini kullanmışlardır. Algoritma olarak Naive Bayes algoritması üzerinde çalışmışlardır [14]. Son zamanlarda çokça önerilen Özet Heterojen kusur tahmini (HDP), projelerini incelemişlerdir. Bununla birlikte, bu yöntemlerin, kusur verilerinin iki özelliğini yeterli şekilde içermediğini düşünmüşler. (1) veriler doğrusal olarak ayrılamaz ve (2) veriler oldukça dengesiz olabilir. Bu iki veri özelliği, etkili bir HDP modeli oluşturmayı zorlaştırmaktadır. Zhiqiang Li ve ekibi bu yazıda, HDP'ye iki aşamalı yeni bir İki Aşamalı Topluluk Öğrenme yaklaşımı (TSEL) yaklaşımı önermişlerdir.Bu öneriler şunlardır; topluluk çok çekirdekli etki alanı uyarlama (EMDA) aşaması ve topluluk veri örnekleme (EDS) aşaması. 30 kamu projesinde yapılan kapsamlı deneyler yapmışlar, önerilen TSEL yaklaşımının diğer yöntemlere göre, AUC'de 20%,14–33,92, f-ölçüsünde 36%, 05–54,78 ve 5%,48-19,93 arasında iyileşme sağlamıştır[15].

Sınıf dengesizliği birkaç kusurlu bileşenlerin yaratmış olduğu tutarsız sonuçlar araştırılmıştır. Sorunun temel özelliklerini keşfetmek için kapsamlı bir deney yapmışlardır. Qinbao Song, Yuchen Guo and Martin Shepperd dengesiz öğrenmenin etkisi, veri dengesizliği, sınıflandırma türü, girdi ölçümleri ve dengesiz öğrenme yöntemi ile etkileşimlerini incelemişlerdir. 27 veri setini, 7 sınıflayıcıyı, 7 tip girdi metrikini ve 17 dengesiz öğrenme yöntemini (hiçbir şey yapmamak dahil) sistematik olarak değerlendirmişlerdir. Veri setinin büyük bir çoğunluğunun (87%) orta veya düşük düzeyde dengesizlik sergilediğini tespit etmişlerdir ve bir diğer sonuç olarak Düşük dengesizlik seviyelerinin dışındaki herhangi bir şey, SDP için geleneksel öğrenmenin performansına açıkça zarar verdiğini görmüşlerdir[16].

Yazılım hata tahmini için makine öğrenme tekniklerini kullanan, literaturdeki 1991 ve 2013 tarihleri arasındaki çalışmaların sistematik bir incelemesi yapmılmıştır. Ruchika

Malhotra makine öğrenimi tekniklerinin yazılım hatası tahmini için mevcut araştırmalardaki performans özelliklerini değerlendirilmiştir. Makine öğrenme tekniklerinin performansını istatistiksel tekniklerle ve diğer makine öğrenme teknikleriyle karşılaştırılmıştır. Ayrıca, makine öğrenme tekniklerinin güçlü ve zayıf yönleri özetlenmiştir. Bu yazıda 64 temel çalışma ve makine öğrenme tekniklerinin yedi kategorisini tanımlamışlardır. Makalede bulunan sonuca göre, modül sınıfı hataya eğilimli veya hataya eğilimli olarak sınıflandırmamak için makine öğrenme tekniklerinin öngörme kabiliyetini kanıtlamaktadır. Yazılım hatası eğilimini tahmin etmek için makine öğrenme tekniklerini kullanan modeller, geleneksel istatistiksel yöntemlerden daha başarılıdır[17].

Bölüm 3

Yazılım Test

Yazılım test, bir görevin doğru şekilde gerçekleştirilip gerçekleştirilmediğini belirleme işlemidir. Yazılımcı tarafından yazılan kodun, işlevsel olarak çalışır olmasının yanında, işlevsel olmayan özelliklerinin de tatmin edici olması gerekmektedir.

Yazılım testi, yazılım yaşam döngüsü boyunca yapılmalıdır. Yazılım yaşam döngüsü başında, analizin kontrol edilmesi ile başlar, yazılım bittikten sonra son kullanıcıya teslim edilene kadar devam etmektedir[18].

3.1 Test Stratejileri

Test stratejileri birim testi, entegrasyon testi, sistem testi, kabul testi olmak üzere dörde ayrılmaktadır. Birim Testi, yöntemlerin veya sınıfların mantıksal davranışını kontrol etmek için otomatik olarak çalıştırılabilecek küçük bir kod parçasıdır. Bir birim testi, sadece belirli yöntemler veya işlevler içindeki kodu test etmemize olanak verir[19]. Birim testi diğer test stratejilerine göre daha uygun maliyetlidir. Kod parçası küçük olduğundan, hata bulmak çok daha kolaydır ve yazılımın diğer kod parçalarının yazılmasının bitmesini beklemeden test edilebilir[20]. İncelenen projelerde birim testleri yazılımcılar tarafından gerçekleştirilmektedir bu nedenle birim testi çalışmamızda yazılımcı testi olarak incelenmiştir.

Entegrasyon Testi, uygulamanın farklı bileşenlerinin birlikte uyum içinde çalışıp çalışmadığının test edildiği süreçtir. Entegrasyon testi, birim testlerinin bitmesinin ardından

yapılır. Ayrıca, hedeflenen sistem ile sistemin iletişim kurduğu harici uygulamalar arasında da gerçekleştirilir[19]. Entegrasyon testi incelenen projeler içerisinde gerçekleştirilen test stratejilerinden biridir. Sistem Testi, belirli gereksinimleri doğrulamak için yapılan testtir. Test sonucu bulunan hataların düzeltilmesi sonucu, yeniden başa dönen sürekli bir test döngüsüdür[19].

Kabul Testi, yazılımın tamamen işlevsel olup olmadığını ve müşterinin gereksinimlerini ve beklentilerini karşılayıp karşılamadığını kontrol etmek için piyasaya sürülmeden veya son kullanıcıya teslim edilmeden önce gerçekleştirilen son testtir[19]. Kabul testi incelenen projelerde, analistler tarafından son kullanıcıya yaptırılan testlerdir. Bu nedenle çalışmamızda bu sonuçlar, kapsam dışı bırakılmışlardır.

3.2 Yazılım Test Türleri

Yazılım test türleri, fonksiyonel test ve fonksiyonel olmayan test olmak üzere ikiye ayrılmaktadır.

Fonksiyonel test, ISTQB Yazılım Testi Terimler Sözlüğüne göre bir yazılımın istenilen işlevselliği gerçekleştirip gerçekleştirmediğini test etme sürecidir. Fonksiyonel testlerde uygulama, işlevsel gereksinimlere karşı test edilir. Testi yapan kişi uygulamanın yapması gerekeni yapıp yapmadığını ve beklendiği gibi çalışıp çalışmadığını test eder. Testten beklenenin ne olduğuna bağlı olarak fonksiyonel testlere farklı bakış açılarından yaklaşılabilir. Fonksiyonel testte amaç, yazılımda ortaya çıkabilecek en kritik hataları ortaya çıkaracak testlere öncelik vermektir[21]. Fonksiyonel testler somut testlerdir. İncelenen projelerde yoğun olarak fonksiyonel test kullanıldığı görülmüştür.

Fonksiyonel olmayan test, ISTQB Yazılım Testi Terimler Sözlüğüne göre "Bir bileşen veya sistemin fonksiyonalitesiyle ilgili olmayan niteliklerinin testi; örneğin güvenilirlik, verimlilik, kullanılabilirlik, sürdürülebilirlik ve taşınabilirlik." Fonksiyonel olmayan testlerde, uygulamanın davranışı test edilmektedir. Fonksiyonel olmayan testler, yazılımın yapması gerekeni yapıp yapmadığına odaklanmaz, daha çok yazılımın kalitesine odaklanmaktadır[21].

- Güvenilirlik: Gürbüzlük, Hata toleransı, Tamir edilebilirlik ve Uyum
- Kullanılabilirlik: Etkileyicilik ve Uyum, Anlaşılırlık, Hâkimiyet, Öğrenilebilirlik
- Etkinlik: Performans, Kaynak Yönetimi ve Uyum
- Bakım yapılabilirlik: Değiştirilebilirlik, Test Edilebilirlik ve Uyum, Analiz edilebilirlik, Stabil çalışma
- Taşınabilir olma: Yüklenebilme, Adapte olabilme, Yer Değiştirilebilir Olma ve Uyum Eş zamanlı çalışabilme.

İncelenen projelerde fonksiyonel olmayan test türlerinin de kullanıldığı görülmüştür.

3.3 Test Teknikleri/ Test Metodolojisi

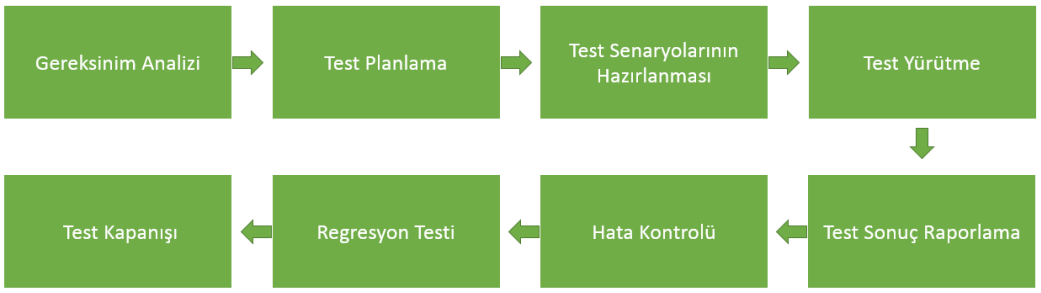
Test teknikleri, inceleme şekillerine göre üç'e ayrılmaktadır. Kara kutu testi, gri kutu testi ve beyaz kutu testi olarak adlandırılmaktadır.

Kara kutu testi, uygulamanın gerekliliklerine ve özelliklerine göre planlandığı bir tekniktir. Testi yapan kişi bu teknikle sadece yazılımın çıktılarına göre test yapabilmektedir[21]. ISTQB Yazılım Testi Terimler Sözlüğü 'ne göre "Test edilecek sistemin iç çalışma mantığı dikkate alınmamaktadır. Kara kutu test tekniğinde fonksiyonel veya fonksiyonel olmayan şekillerde testler yapılmaktadır. "Bu tekniğin dezavantajı, isterlerin subjektif olması nedeniyle otomatik hale getirilmesi zor olmasıdır. Bu nedenle, genellikle manuel olarak test edilmektedir[22].

Gri kutu testi, test yapan kişinin yazılımın temel yapısını bildiği, ancak ayrıntılarını bilmediği, beyaz ve kara kutu testlerinin bir karışımıdır. Testi yapan kişi, uygulamanın nasıl işlediği hakkında biraz bilgi sahibidir, ancak kaynak koduna erişimi yoktur[21].

Beyaz kutu testi, ISTQB Yazılım Testi Terimler Sözlüğü 'ne göre "Bir sistemin veya bir bileşenin içyapısının analizine bağlı olarak test etme yöntemidir." Beyaz kutu testi yapısal test veya kod temelli test olarak da kabul edilebilir. Beyaz kutu testi genellikle yazılımın programcıları tarafından yapılır ve yazılım geliştirmenin başından itibaren başlatılabilir. Beyaz kutu testinde amaç çoğu zaman mümkün olduğunca kodun kapsanmasıdır[21].

3.4 Yazılım Test Yaşam Döngüsü



Şekil 3.1: Yazılım yaşam döngüsü

Şekil-3.1'de verilen yazılım test yaşam döngüsünün ilk aşamasında, gereksinim analizi yapılmaktadır. Gereksinim analizi yapılmasında ki amaç analistler tarafından belirlenen isterlerin test edilebilir durumda olduğundan emin olmaktır. Subjektif durumlar test edilemez. Yazılan isterlerin net ve herkes tarafından aynı şekilde anlaşılır olması gerekmektedir.

Test Planlama, Yazılım Test Yaşam Döngüsü'nün ikinci ve en önemli aşamasıdır, çünkü tüm test stratejisinin tanımlandığı adımdır.

Test Senaryolarının Hazırlanması aşaması, test senaryolarının geliştirildiği ve test planlama faaliyetinin durdurulduğu aşamadır. Uygun test senaryoları test uzmanı tarafından elle yazılır veya bazı durumlarda otomatik test senaryoları oluşturulur.

Test Yürütme Aşaması, hazırlanan test senaryoların yürütülmesinden oluşur. Herhangi bir hata alınmaz ise, test başarılır olur. Test aşamasında hata ile karşılaşılır ise, sonuç raporlama aşamasına geçilir.

Test Raporlama, test senaryolarının yürütülmesinden sonra belirlenen hataların rapor edilmesidir ve hata raporu hataların düzeltilebilmesi için geliştirme ekibine iletilir.

Hata Kontrolü; bu aşamada geliştirme ekibinden dönen yazılımda, ilgili hatanın düzletilip düzeltilemediğine bakılır.

ISTQB Yazılım Testi Terimler Sözlüğü' ne göre "Regresyon testi, Yazılım da yapılan değişiklik veya düzeltme sonrasında bu değişiklik veya düzeltmenin yazılımın başka yerlerinde

sebep olabileceği hataları bulmaya yönelik olarak yazılımın değiştirilmeyen veya düzeltilmeyen taraflarının tekrar test edilmesidir. Yazılım veya yazılımın ortamı değiştiğinde uygulanan testlerdir. “

Regresyon test sürecinde, yüksek öncelikli test durumları daha erken yürütülür. Test uzmanları, ilk testte hata alınan test senaryolarına öncelik vererek, daha hızlı şekilde, testin erken aşamalarında hata tespit etme olasılığını artırır[22].

Hata Kodları/Test Sonucu

- **Katastrofik (Ölümcül) Hata:** Sistemin çökmesi veya kilitlenmesine sebep olan ve testin yapılmasına engel olan yıkıcı hatalardır.
- **Büyük Hata:** Gereksinimin bazı durumlarda karşılanıp bazı durumlarda karşılanmadığı hatalardır. Örn; bir gereksinimin sınır değerde karşılanıp sınır dışı değerde karşılanmadığı hatalardır.
- **Kritik Hata:** Sistemin çökmesi veya kilitlenmesine sebep olmayan ve testin yapılmasına engel olmayan fakat ilgili gereksinimi karşılamayan hatalardır.
- **Küçük Hata:** Gereksinimin karşılanmadığı durumda ilgili işlevi gerçekleştirmenin başka yolunun olduğu, etkisi az olan hatadır. Örn; Copy+paste.
- **Kozmetik Hata:** Gereksinimlerin karşılanmasını etkilemeyen, görünüm ve benzeri görsel hatalardır.
- **Başarılı:** Gereksinimlerin doğru ve tam bir şekilde karşılandıkları durumdur.
- **Analist ile Görüşme:** İsterin yanlış veya test edilemez olduğu durumlarda kullanılan hata kodudur.
- **Öneri:** Yazılımın veya analizin yanlış olmadığı, fakat daha iyi olabileceği düşünülen durumlarda test uzmanının analist veya yazılımcıya öneride bulunduğu hata kodudur.
- **Test Edilemedi:** Yazılımın herhangi bir nedenle test edilemediği durumlarda kullanılan hata kodudur.

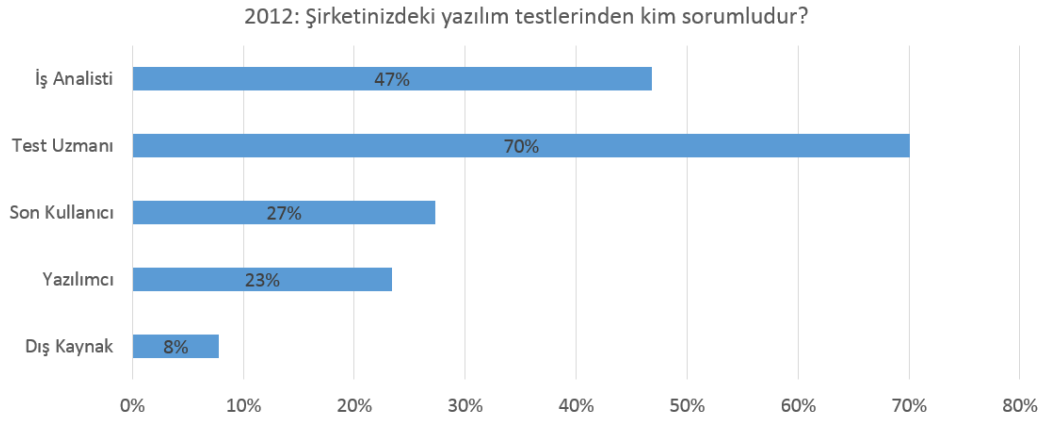
3.5 Yazılım Test Trendleri

3.5.1 Türkiye

Yazılım test tarihçesi farklı bir açıdan incelenmiştir. Tukish Testing Board (TTB), 2011 yılından beri Türkiye'de "Turkey Software Quality Report" başlığı altında, Türkiye'de ki yazılım sektörünün gidişatını, yapmış olduğu anketler ile raporlamaktadır. Anketler Türkiye'de yazılım sektöründe çalışan binlerce kişi tarafından cevaplanmaktadır. Her konuda olduğu gibi test dünyasında da yönelimler yıldan yıla değişmektedir. World Quality Report 2012 yılından itibaren yayınlanan benzeri bir rapordur. Kıyaslama yapabilmek adına, 2012 yılından itibaren konular incelenmiştir. Türkiye'de rapor konuları yıllara göre şu şekilde değişmiştir;

- **2012**: Test Organizasyonu ve Süreçleri, Test Eğitimleri, Test Otomasyon Araçları[23]
- **2013**: Gelecekte Test, Yeni Test Teknikleri ve Methodolojileri [24]
- **2014**: Mobilleşme[25]
- **2015**: İş odaklı Performans Testi[26]
- **2016**: Test Veri Yönetimi[27]
- **2017**: Çevik (Agile) Test[28]
- **2018**: Yeni Trendler, DevOps, Hata önleme, Sürekli Test, Yapay Zeka (Makine Öğrenmesi), Statik Test, Test Otomasyon[29]

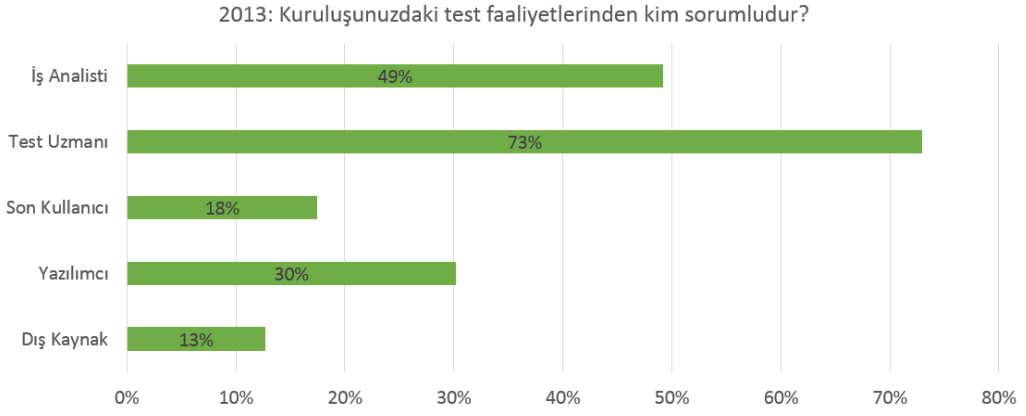
Anketlerde yer alan sorular yıl içerisinde önemli olan başlıklar hakkında çok önemli bilgiler vermektedir. İncelediğimiz raporlarda yıllara göre konular ve sorular ciddi değişiklikler gösterirken, az değişkenlik gösteren tek soru, yazılım testi projesini kimin gerçekleştiriyor olduğudur. Yıllara göre değişen soruları ve cevapları incelendiğinde, 2012 yılında test organizasyonu ve süreçleri, test eğitimleri, test otomasyon araçları konularının popüler olduğu görülmüştür. İlgili ankette katılımcılara "Şirketinizdeki yazılım testlerinden kim sorumludur?" sorusu sorulmuştur [23]. Şekil-3.2'de anket sonuçları verilmiştir.



Şekil 3.2: 2012- Şirketinizdeki yazılım testlerinden kim sorumludur?

Bir yıl öncesinin raporunda Test Uzmanları, 54% yer alırken, 2012 yılında ciddi bir artış göstererek 70% seviyelerine çıkmıştır. Aynı zamanda 2012 yılı raporunda, bu senelerde yazılım test uzmanlarının en büyük probleminin, alan bilgisinin az olduğu söylenmiştir. Bu nedenle de firmalarda analist ve yazılım test uzmanı ayrışıma gidileceği öngörülmüştür. 5 yıl içerisinde ise, yazılımcıların sadece birim test yapacağı, son kullanıcının ise, kullanıcı kabul testi dışında test yapmayacağı belirtilmiştir[24].

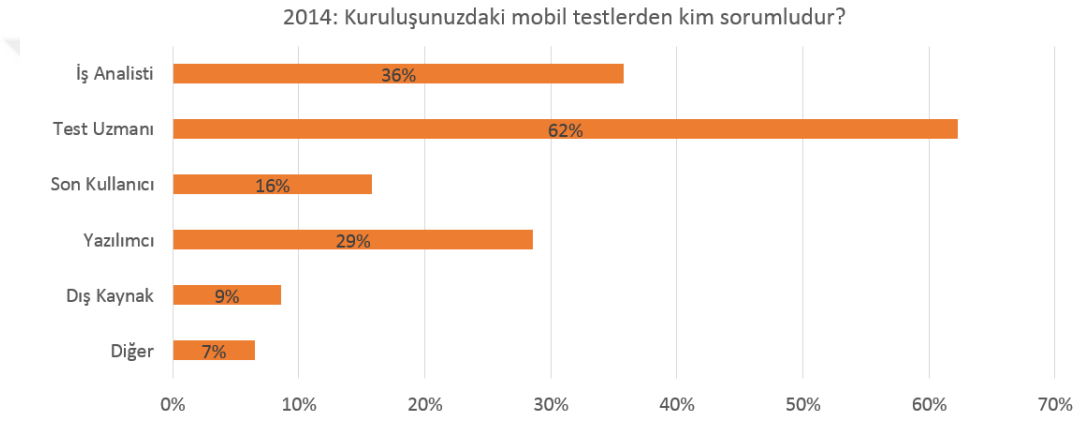
2013 yılında gelecekte test, yeni test teknikleri ve metodolojileri konularının popüler olduğu görülmüştür. İlgili ankette katılımcılara "Kuruluşunuzdaki test faaliyetlerinden kim sorumludur?" sorusu sorulmuştur[24]. Şekil-3.2'de anket sonuçları verilmiştir.



Şekil 3.3: 2013- Şirketinizdeki yazılım testlerinden kim sorumludur?

Bir yıl öncesinin test raporunda son kullanıcılar 27% oranında yazılım test işlerini yaparken, bu yıl ciddi bir düşüşle 18% oranlarına gerilemiştir. Bu gösteriyor ki, yazılım firmaları daha çok test uzmanı ile çalışmaya ve son kullanıcıya gitmeden testlerini kendi içlerine yapmaya başlamıştır. 73% olan yazılım test uzmanı oranının ise ilerleyen yıllarda 100'e yaklaşmasını beklediklerini belirtmişlerdir [24].

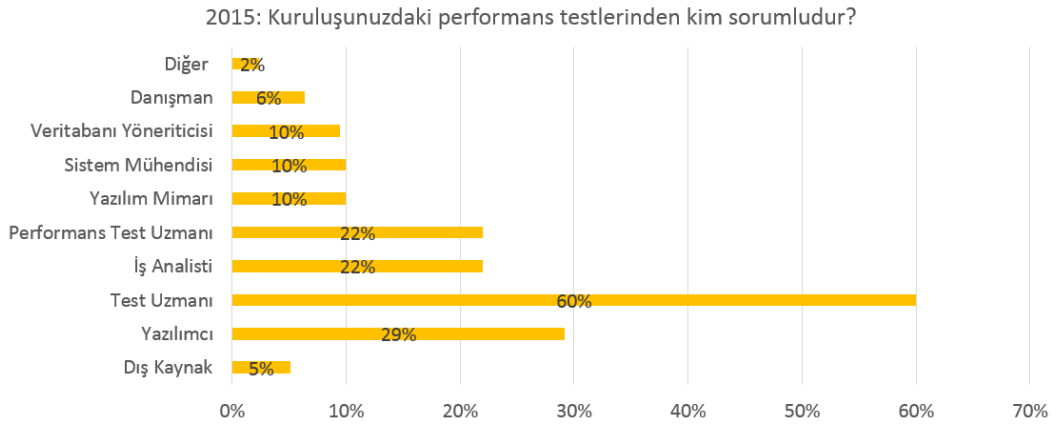
2014 yılında mobilleşme konusunun popüler olduğu görülmüştür. İlgili ankette katılımcılara "Kuruluşunuzdaki mobil testlerden kim sorumludur?" sorusu sorulmuştur[25]. Şekil-3.4'te anket sonuçları verilmiştir.



Şekil 3.4: 2014- Şirketinizdeki yazılım testlerinden kim sorumludur?

Son kullanıcının 15% değer ile hala anketlerde yer almasının sonucunun, mobil uygulamaların, son kullanıcı deneyimleri üzerine geliştirilmesinden kaynaklı olduğu belirtilmiştir. Mobil teknolojilerde de yüksek oranda test uzmanı çalışıyor olması, geleneksel yaklaşımın devam ettiğini ve birçok firmanın kendi bünyesinde test uzmanı çalıştırdığını göstermektedir[25].

2015 yılında iş odaklı performans testi konusunun popüler olduğu görülmüştür. İlgili ankette katılımcılara "Kuruluşunuzdaki performans testlerinden kim sorumludur?" sorusu sorulmuştur[26]. Şekil-3.5'te anket sonuçları verilmiştir.

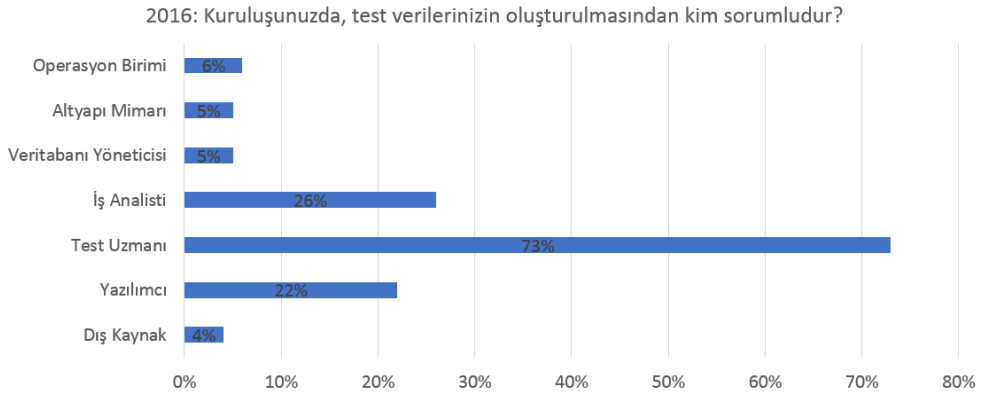


Şekil 3.5: 2015- Şirketinizdeki yazılım testlerinden kim sorumludur?

Teknolojinin gelişmesi ile birlikte, performans testi yapmak ciddi manada zorlaşmıştır. Performans testi yapabilmek için, teknik bilgiye sahip olunması, araç kullanımının bilinmesi, donanım bilgisi olması gibi ek özellikler gerekmektedir. Bu nedenle performans testi, geleneksel olarak test yaklaşımından ayrışmıştır.

Anketlerden çıkan sonuca göre, performans testi için kendi alanında uzmanlaşmış test uzmanları bu görevi yapmaktadır. Test kavramının gelişmesi ile birlikte geleneksel testçilerinin yanında, uzmanlaşmış, dallanmış farklı test uzmanları da görev yapmaya başlamıştır. Bu yazılım test dünyası için çok önemli bir gelişmedir[26].

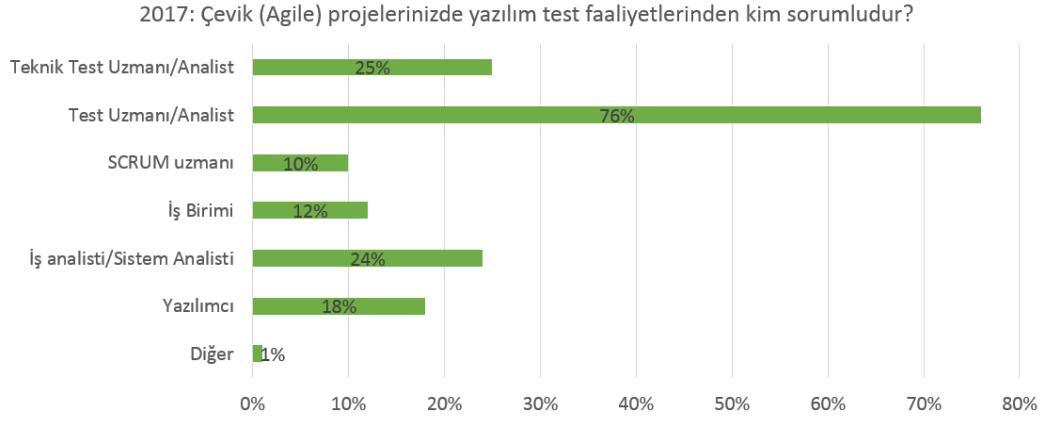
2016 yılında test veri yönetimi konusunun popüler olduğu görülmüştür. İlgili ankette katılımcılara "Kuruluşunuzdaki performans testlerinden kim sorumludur?" sorusu yöneltilmiştir[27]. Şekil-3.6'da anket sonuçları verilmiştir.



Şekil 3.6: 2016- Şirketinizdeki yazılım testlerinden kim sorumludur?

Sonuçlara göre test uzmanları, şirketlerinde test verilerini yönetiminde ciddi bir farkla önde gitmektedirler[27].

2017 yılında çevik (agile) test konusunun popüler olduğu görülmüştür. İlgili ankette katılımcılara "Çevik (Agile) projelerinizde yazılım test faaliyetlerinden kim sorumludur?" sorusu sorulmuştur[28] Şekil-3.7'de anket sonuçları verilmiştir.

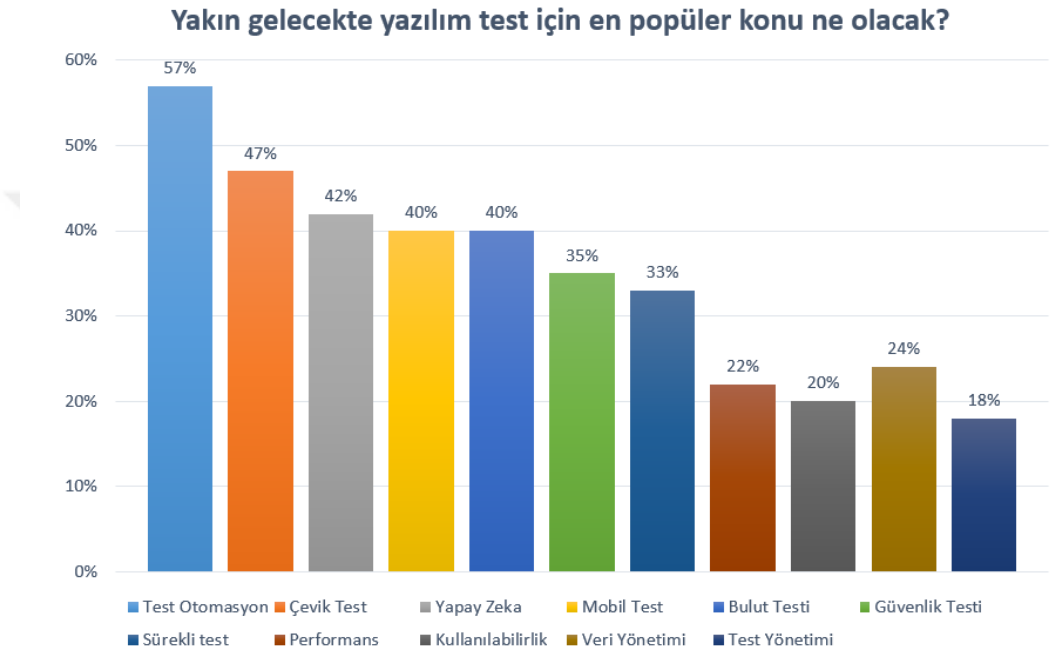


Şekil 3.7: 2017- Şirketinizdeki yazılım testlerinden kim sorumludur?

Çevik (Agile) yazılım yaşam döngüsünde Çevik (Agile) yazılım yaşam döngüsünde yazılım test uzmanı rolü bulunmamaktadır. SCRUM uzmanı, ürün sahibi ve yazılımcı rolleri

ile yürüyen sistemde nasıl olurda bu denli yüksek rakamlarda test uzmanı çıkar? Bu organizasyonlarda halen yazılım test uzmanı rolünün olmasından kaynaklanmaktadır[28].

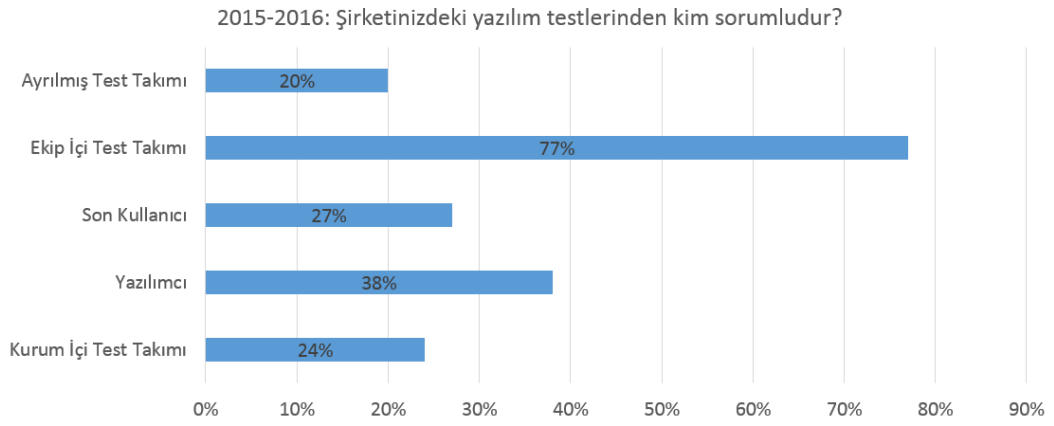
2018 yılında Yeni Trendler, DevOps, Hata önleme, Sürekli Test, Yapay Zekâ (Makine Öğrenmesi), Statik Test, Test konularının popüler olduğu görülmüştür. İlgili ankette katılımcılara "Yakın gelecekte yazılım test için en popüler konu ne olacak?" sorusu sorulmuştur[29]. Şekil-3.8'de anket sonuçları verilmiştir.



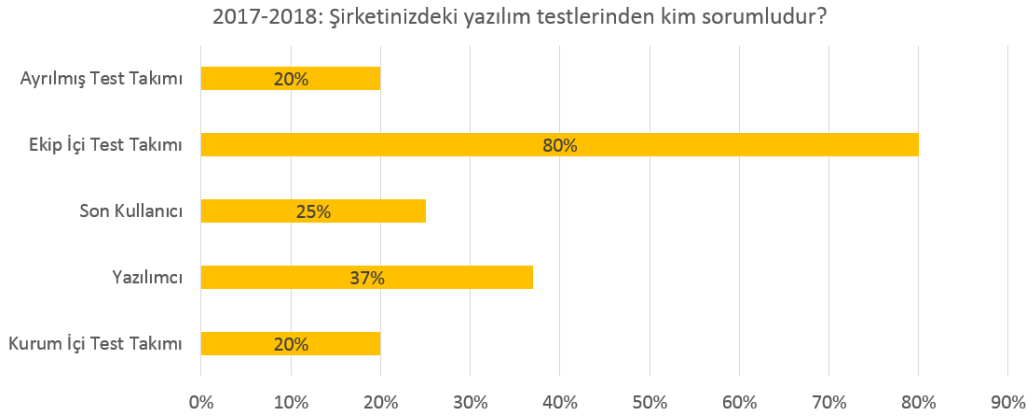
Şekil 3.8: Yakın gelecekte yazılım test için popüler konu ne olacak?

2018 raporunda en dikkat çeken soru, yakın gelecekte en popüler konuların neler olacağı sorusudur. Artık testi kimin yaptığı sorusundan 2018 yılı itibari ile vazgeçilmiştir. Türkiye'de testi, test uzmanı yapar kavramının oturduğu sonucuna ulaşılmaktadır. Anket sonuçlarında yapay zekânın, yazılım test ile birlikte kullanılacağı konusunda ciddi bir rakam çıkmaktadır. Anketi cevaplayanların 42%'si yapay zekânın test uzmanları tarafından kullanılacağı konusunda hemfikirdir. En yüksek oranda test otomasyon, çevik test ve yapay zekânın seçilmesinin sebebi, yazılım dünyasında trendin, daha ucuz, daha hızlı, daha kaliteli yazılımlar olmasıdır. Özel olarak yapay zekânın tercih edilme sebeplerinden bahsedecek olursak, yapay zekâ ile birlikte testler daha ulaşılır, daha ucuz ve daha kolay uygulanabilir olacaktır. Yakın gelecekte ise bu sıralamanın çok değişmeyeceği düşünülmektedir[29].

3.5.2 Dünya



Şekil 3.9: 2015-2016 Şirketinizdeki yazılım testlerinden kim sorumludur?



Şekil 3.10: 2017-2018 Şirketinizdeki yazılım testlerinden kim sorumludur?

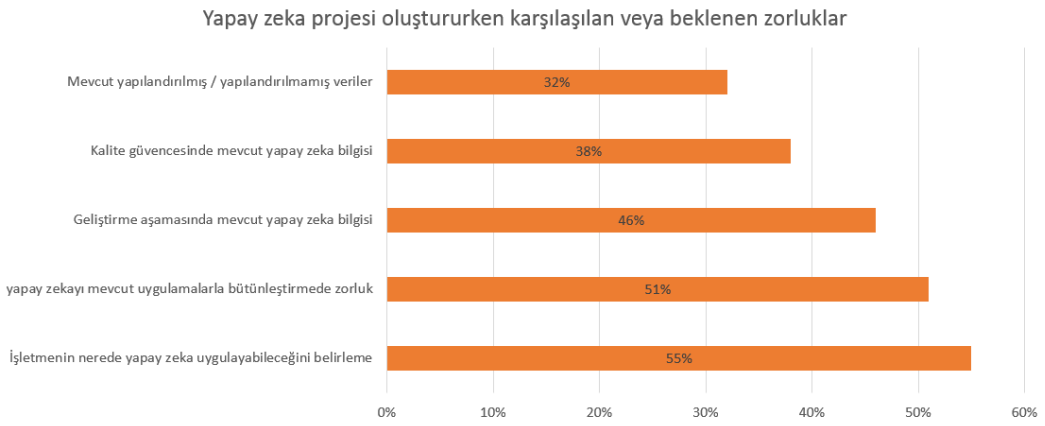
International Software Testing Qualification Board'ın yayınlamış olduğu, Worldwide Software Testing Report benzer sorulara dünya çapındaki yazılım ekiplerinin verdikleri cevaplar incelenmiştir. Türkiye'den farklı olarak Test Uzmanı kavramının çok daha fazla oturduğu görülmektedir. Türkiye'de testleri analistler de yaparken, yurtdışında bu kavramdan hiç söz edilmediğini görülmüştür. Son kullanıcı ve yazılımcı tarafından yapılan testlerin ise oranının azaldığını görülmüştür[30][31].

World Quality Report'u incelersek, yıllara göre dünyadaki test eğilimlerinin, konularının nasıl değiştiğini gözlemleyebiliriz.

- **2012**: Mobil Test, Bulut Teknolojileri, Mükemmellik Merkezli Test, Kalite Güvence, Test Organizasyonu [32]
- **2014**: Test Bütçeleri, Test Organizasyonu, Bulut Teknolojileri, Mobil Test, Çevik Test, Test Otomasyon [33]
- **2015**: Güvenlik Testi, Çevik Test, Test Veri Yönetimi, Test Bütçeleri, Dijital Dönüşüm [34]
- **2016**: Dijital Dönüşüm, Nesnelerin İnterneti, Çevik Yazılım ve DevOps, Mükemmellik Merkezli Test [35]
- **2017**: Dijital Dönüşüm, Çevik Yazılım ve DevOps, Test Veri Yönetimi, Nesnelerin İnterneti [1]
- **2018**: Yapay Zekâ (Makine Öğrenmesi), Otomasyon, Test Bütçeleri ve Maliyet, Test Verileri ve Test Ortamları [2]

2018 World Quality Report yapmış olduğu test sonucunda, katılımcıların 57%'sinin yapay zekâyı, test işlemlerinde kullanmak üzere, şimdiden yıllık planlarına aldıkları sonucuna ulaşmıştır. Rapora göre; yapay zekânın gelişmesinden, yazılım dünyası iki farklı şekilde etkilenecek. Bunlardan ilki yapay zekâ ile birlikte, tamamen kendini oluşturan kodlar, kendini test sonuçlarına veya değişen kodlara adapte edebilen kodlar, kendini çalıştırabilen kodları oluşturmak mümkün olacaktır. İkincisi, yapay zekâ kullanımının henüz başlangıç aşamaları olması sebebiyle, ilk olarak organizasyonlar, kritik kararları alabilmek veya hataların önlenmesi için, yapay zekâyı kullanabilirler. Yapay zekânın hâlihazırda gelişen bir teknoloji olması nedeni ile bu teknolojiyi yazılım testine uygulamak için gereken bilgi, uzmanlık ve olgunluk hala birçok organizasyonda bulunmamaktadır. Organizasyonlarda yapay zekâ ve makine öğrenmesi kullanımı konusunda çok ciddi bir heyecan söz konusu olup bu konuda birçok deneme ve çalışma yapılmaktadır.

Yapay zekâ ile yazılım test alanında hızla ilerleme kaydetme yeteneği, bir kuruluşun madencilik verilerindeki olgunluğuna bağlıdır. Bu nedenle, veriler testlerde yapay zekânın benimsenmesi konusunda en büyük zorluklardan biridir. Sadece onunla çalışma yeteneği önemli değil, aynı zamanda mevcut verilerin kaliteli olması da gerekmektedir. Ek olarak, testlerde yapay zekâ kullanımı, tipik test uzmanı profillerinden çok farklı roller oluşturur. Bu roller test etme, geliştirme, doğal dil işleme, veri bilimi, matematik, algoritmik bilgi ve makine öğrenimi gibi farklı beceriler karışımı gerektirir[2].

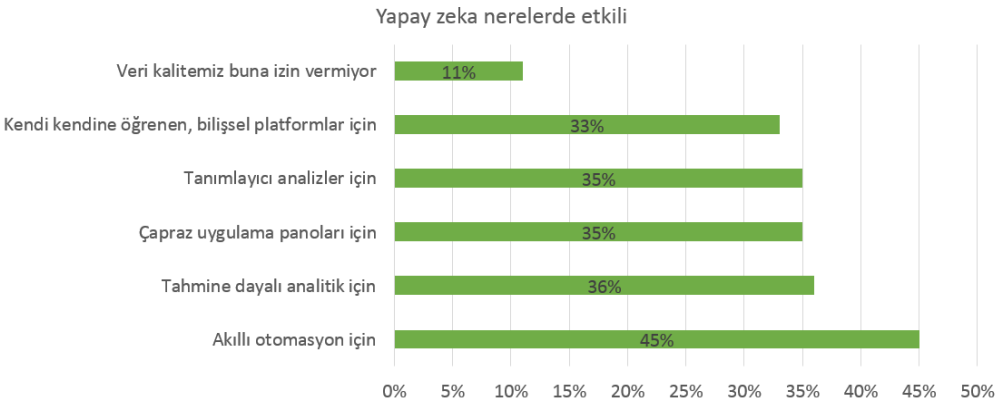


Şekil 3.11: Yapay zeka projelerinde karşılaşılan/ beklenen zorluklar

Rapora göre; Ancak, bu potansiyeli gerçeğe dönüştürmek her zaman kolay değildir. Yapay zekâ ile olan ilişkilerinde çoğu kuruluş, makine öğrenmesi, sinir ağları, bulanık mantık, robotik veya derin öğrenme gibi yapay zekâ teknolojilerini kullanmaktan ziyade, hala veri analitiği düzeyinde sıkışıp kalmaktadır.

Şekil 11'de yapay zekâ projelerini uygulamada karşılaştıkları zorluklar nelerdir sorusunun sonuçları verilmiştir. Katılımcılarım 55%'i "işletmenin yapay zekâyı nerede uygulayabileceklerini belirlemede zorluk yaşadıklarını" bildirmiştir.

Anket sonuçları, yukarıdaki zorlukların yanı sıra "yapay zekayı mevcut uygulamalarla entegre etmekte zorluk çektiğini" (cevap verenlerin 51%'i) ve "(kalkınmada mevcut yapay zeka bilgisinin yokluğunu" iki büyük sorun olarak belirtti. Ek olarak, uzman görüşü ayrıca, yapay zekânın kalite güvence ve test için daha fazla benimsenmesini engelleyen en büyük engellerden biri olarak verilerin niceliği ve kalitesi ile ilgili sorunlara işaret etmektedir.

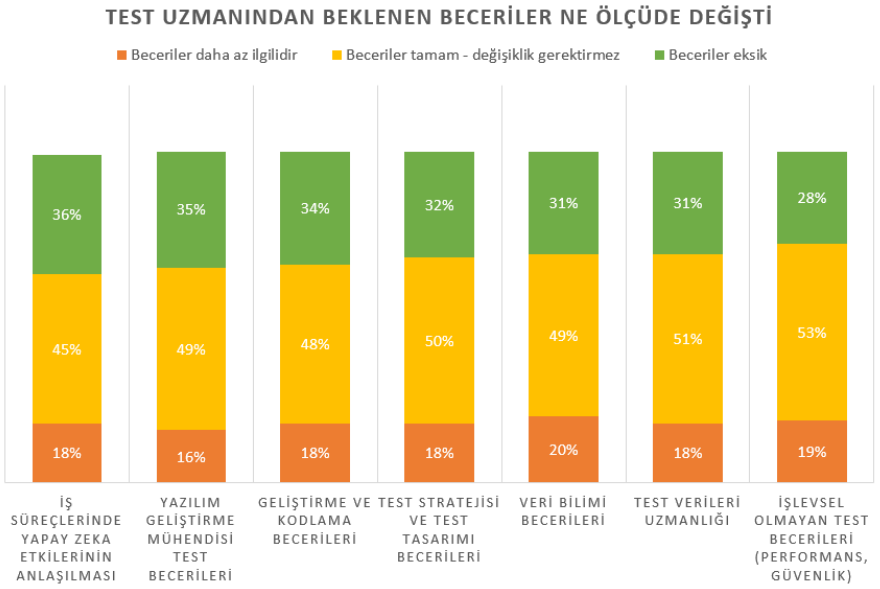


Şekil 3.12: Yapay zeka nelerde etkili?

Şekil 3.12'de yapay zekâ nerelerde etkilidir? Sorusunun sonuçları verilmiştir. Kuruluşlar yazılım kalitesi için yapay zekâ kullanarak bugün nasıl test ediyor? 2018 WQR anketi sonuçlarına göre, yanıt verenlerin 45% 'i akıllı otomasyon için AI kullandıklarını, 36%'sı tahmine dayalı analitik analizler için kullandıklarını belirtmişlerdir.

Tabloda belirtilen eğilimler, öngörülen analitik, robotik otomasyon, bilişsel otomasyon ve bu yılki ankette ortaya çıkan trendler olarak tanımlanan makine öğrenmesi gibi bazı trendlerle doğrudan ilişkilidir. Yapay zekâ kullanımı zor ve çok farklı becerileri de barındırmak gerektiren bir yöntemdir.

Anket katılımcılarına bu tür akıllı uygulamaları nasıl test ettikleri sorulduğunda, 57%'si yeni bir test yaklaşımı denediklerini, 45%'i ise yeni bir test yaklaşımı araştırdığını ve 35%'inin yeni bir test yaklaşımı kullandığını söyledi. Sonuçlardan, henüz hiçbir standart sürecin geliştirilmediği ve farklı organizasyonların farklı şekillerde deney yaptıkları çıkmaktadır[2].



Şekil 3.13: Test uzmanından beklenen beceriler ne ölçüde değişti?

Deneyim, yapay zekânın test uzmanlarının ihtiyaç duyduğu becerileri değiştirdiğini ortaya koymaktadır. Yapay zeka ile çalışmak, test etme, matematiksel optimizasyon, nörodilsel programlama, , iş zekası becerileri ve algoritmik bilgi gibi çeşitli yetkinliklere sahip profesyoneller gerektirdiğinden, geleneksel test yaklaşımı artık yeterli değildir.

Tabloda belirtildiği üzere şu anda, bu beceri kombinasyonunu bulmak zordur ve uzmanlar gelecekte daha fazla sayıda kuruluşun yapay zekâ ile deney yapmaya başlamasından dolayı nitelikli profesyonellerin mevcudiyeti ile ilgili zorlukların artacağını öne sürmektedir. Şekil-3.13'te verilen anket sonuçları yukarıdaki bilgilere ulaşmamızı sağlamaktadır[2].

Bölüm 4

Makine Öğrenmesi

Makine öğrenmesi, sistemlerden programlamadan, otomatik olarak öğrenme ve deneyimleri geliştirme becerisi sağlayan yapay zekânın bir uygulamasıdır. Makine öğrenmesi, verilere erişebilen ve kendileri için öğrendiklerini kullanabilen bilgisayar programlarıdır. Birincil amaç, bilgisayarların insan müdahalesi veya yardım olmadan otomatik olarak öğrenmesini sağlamak ve eylemleri buna göre ayarlamaktır.

Makine öğrenmesi, büyük miktarda veri analizini mümkün kılar. Kârlı fırsatları veya tehlikeli riskleri belirlemek için genellikle daha hızlı, daha doğru sonuçlar verirken, uygun bir şekilde eğitmek için ise ek zaman ve kaynak gerektirebilir. Makine öğrenimini yapay zekâ ve bilişsel teknolojilerle birleştirmek, büyük hacimli bilgilerin işlenmesinde daha da etkili olabilir.

Makine öğrenmesi yaklaşımlarının, bir kısmı tahmin(prediction) ve kesitirim (estimation), bir kısmı da sınıflandırma (classification) yapmaktadır.

- Tahmin (prediction): Veriden öğrenen modellerde sistem çıkışının nicel olması durumunda kullanılan yöntemlerin ürettiği değerlerdir.
- Sınıflandırma (classification): Giriş verisine yapay zeka çıkışların nitel olduğu durumlarda kullanılan yöntemlerin her veri örneğinin hangi sınıfa yapay zekat olduğunu belirlemesidir.
- Kümeleme (clustering) : Niteliklerinin bazı benzerlik kolaylıklarına göre sınıflandırılmasıdır. Önceden belirlenmiş bir sınıf kümesi olmadığı için sınıflandırmadan farklıdır[36].

Makine öğrenmesinde veriden bahsedecek olursak, verilerde yapısına göre ikiye ayrılmaktadır.

- Danışmanlı (Supervised) Öğrenme: Sınıf nitelik değerini tahmin etmek için veriyi kullanır.
- Danışmansız (Unsupervised) Öğrenme: Veri kümesinin sınıf nitelik değeri yoktur. Benzer örnekler bulunarak gruplanır.

4.1 Ön İşleme

4.1.1 Etiket Kodlaması (Label Encoding)

Bazı istisnalar dışında makine öğrenme algoritmaları için sayısal değerler gerekmektedir. Programa sayısal olmayan değerler sunulduğunda, özel bir yazılım kullanmadıkça program ne yapacağını bilememektedir. Birçok veri setinde isimler ve kategorik özellikler bulunmaktadır. Bu nedenle de bu verileri sayısal değerlerle değiştirmemiz gerekmektedir. Bu yöntemde her farklı değerin bir etiket olduğunu düşünülmektedir. Daha sonra her etiket bir tam sayı etiketi kodlamasıyla değiştirilmektedir [37].

Yazılım testi sonucunda, hataları gruplandırma sekiz farklı sayısal olmayan değerdir. Etiket kodlaması yöntemi ile bu hata kodları sayısal değerlere çevrilmiştir.

Tablo 4.1: Etiket kodlaması örnek tablosu

Hata durum adı	kod
Katastrofik (Ölümcül) Hata	0
Büyük Hata	1
Kritik Hata	2
Küçük Hata	3
Kozmetik Hata	4
Başarılı	5
Analist ile Görüşme	6
Test Edilemedi	7

4.1.2 Ölçekleme (Scaling)

Farklı özelliklerin değerleri, büyüklük sırasına göre değişebilmektedir. Bu, büyük değerler daha küçük değerleri anlamsız kılmaktadır. Bu durum kullanılan algoritmaya bağlı olarak değişen bir durumdur. Bazı algoritmaların düzgün çalışması için verileri ölçeklendirmemiz gerekmektedir. Öklid mesafesi ölçüsüne sahip en yakın komşular algoritması, değer büyüklüklerine duyarlıdır ve bu nedenle tüm özelliklerin eşit ağırlıkta olması için ölçeklendirilmelidir. Temel Bileşen Analizi (PCA) yaparken de ölçeklendirme kritiktir. PCA özellikleri, maksimum varyansla elde etmeye çalışır ve varyans, yüksek değerdeki büyüklükteki özellikler için yüksektir. Bu PCA'yı yüksek büyüklük özelliklerine doğru eğriltir. Ağaç tabanlı modeller mesafeye dayalı modeller değildir ve çeşitli özellik aralıklarını idare edebilirler. Bu nedenle, ağaçları modellerken Ölçeklendirme gerekli değildir. Naive Bayes algoritmaları, bu durumun üstesinden gelmek için tasarlanan algoritmalardır ve buna göre özelliklere ağırlıklar verir. Bu algoritmalarda ölçeklendirme özelliklerinin gerçekleştirilmesi fazla bir etkiye sahip olmayabilir [38].

x veri değeri için, bu verinin standardize edilmiş değeri z, z = (x - u) / s formülü ile hesaplanır. U, eğitim örneklerinin ortalamasıdır. S ise, eğitim örneklerinin standart sapmasıdır. Eğitim setindeki her veri için, bu hesaplama bağımsız olarak gerçekleştirilir [38].

4.1.3 Temel Bileşenler Analizi (PCA)

Temel Bileşen Analizi (PCA), bir veri setindeki gürültüyü veya boyutsallığı azaltmak için kullanılır. PCA en büyük varyanslara sahip doğrusal kombinasyonları arar ve en önemli bilgiyi elde etmek için en büyük varyansın en yüksek bileşen tarafından yakalandığı Asıl Bileşenler (PC) olarak ayırır [39].

Bir diğer anlatım şekli ile çok gözlemli olan bir veriyi, daha az boyut kullanarak nasıl ifade ederizi anlatmaktadır. PCA bir boyut azaltma tekniğidir. PCA analizi; veri boyutunu azaltma, tahminleme yapma, veri setini bazı analizler için görüntülemek olmak üzere üç temel amaç için kullanılabilir.

PCA Uygulama adımları; verileri hazırlama, kovaryans/korelasyon matrisini oluşturma, temel bileşenleri (pc) oluşturma, yeni verisetini oluşturmadan meydana gelmektedir [40]

$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}$$

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

Şekil 4.1: Standart sapma formülü

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

Şekil 4.2: Varyans formülü

Standart sapma, verilerin nasıl yayıldığına dair bilgi verir. Standart sapma varyansın karekökü̈dür.

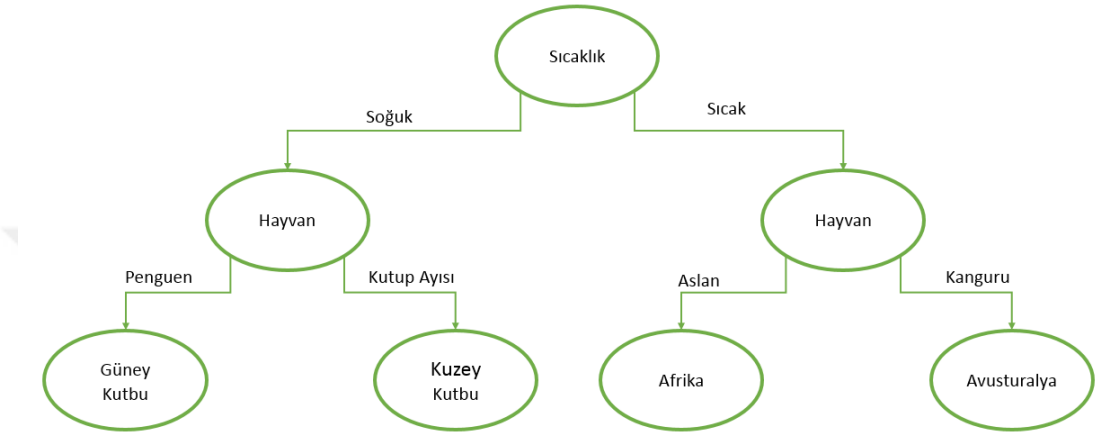
4.2 Karar Ağacı (Decision tree)

Karar Ağacı Öğremesi Makine Öğrenmesinde en çok kullanılan algoritmalardan birisi olarak karşımıza çıkmaktadır. Karar ağaçlarının bu kadar çok kullanılmasının sebeplerinden biri, diğer öğrenme metotlarına göre daha etkili olmasıdır. Ağaç şeklinde gösterilen öğrenilmiş sınıflandırma modelidir. Karar Ağaçları yöntemi aynı zamanda danışmanlı öğrenmenin bir alt dalıdır [41].

Karar Ağacı, en yaygın öğrenme yöntemlerinden biri olan sınıflandırma tekniğidir. Bir Karar Ağacı, belirli bir durma kriterine ulaşılıncaya kadar, verileri mümkün olduğu kadar farklı mevcut sınıflara ayırır ve özellik üzerinde ayarlanan verilerin yinelemeli olarak bölünmesiyle oluşturulur. Karar ağaçları, çok kolay ve anlaşılır şekilde görselleştirilebilir. Bu nedenle kullanıcılar tarafından kolayca anlaşılabilir. Karar ağacı ile ilgili iki adet algoritma mevcuttur. Bu algoritmaları İteratif Dichotomiser 3 (ID3) ve C4.5, 1986 ve 1993'te Ross Quinlan geliştirmiştir [41].

Karar Ağaçlarında, düğümler ayrıca kök olarak adlandırılan düğümlere, iç düğümlere ve son düğümlere ayrılabilir. Kök düğüm karar destek sürecinin başlangıcını temsil eder[41].

Karar ağacı yapısının arkasındaki fikir, örnekleri mümkün olduğu kadar homojen olan gruplara ayırmaktır. Tipik olarak, bir veri kümesinin en anlamlı özelliği, veriler bölündüğü zaman, en homojen veri gruplarına bölendir. Çoğu durumda, ağacın ilk dalı olarak bu özellik seçilir. Verilerin daha fazla bölümlenmesi, genellikle homojen grubun veya tek bir sınıftan örneklerden oluşan grupların elde edilmesi için gereklidir ve bu, daha az anlamlı özelliklerin değerlerine dayanarak yapılır [42] .



Şekil 4.3: Karar ağacı örneği

Yukarıdaki karar ağacı örneğinde, havanın sıcak veya soğuk olması durumunda ve bölgede görülen hayvan bilgisi ile birlikte hangi bölgede olduğumuzu tahmin etmek istiyoruz. Bu durumda Sıcaklık =Sıcak ve Hayvan=Aslan ise Afrika bölgesinde olduğumuz sonucuna varırız. Araştırmada ID3 algoritması kullanılmıştır. ID3 algoritması ise entropi ve bilgi kazanım formülleri kullanmaktadır. Bilgi kazanım değeri en yüksek olan değer kök değer olarak seçilmektedir.

Entropi formülünde S veri setini, C veri seti içerisinde bulunan sınıfları, p(c) ise c sınıfı içerisinde yer alan elementlerin oranını temsil etmektedir. Bilgi kazanım formülünde H(s) s veri setinin entropisini, T ise S kullanılarak oluşturulan alt veri setini, p(t) t alt veri seti içerisindeki elementlerin oranını, H(t) ise alt veri setinin entropisinin değerini temsil etmektedir.

$$H(S) = \sum_{c \in C} -p(c) \log_2 p(c)$$

Şekil 4.4: Entropi Formülü

$$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

Şekil 4.5: Bilgi kazanım formülü

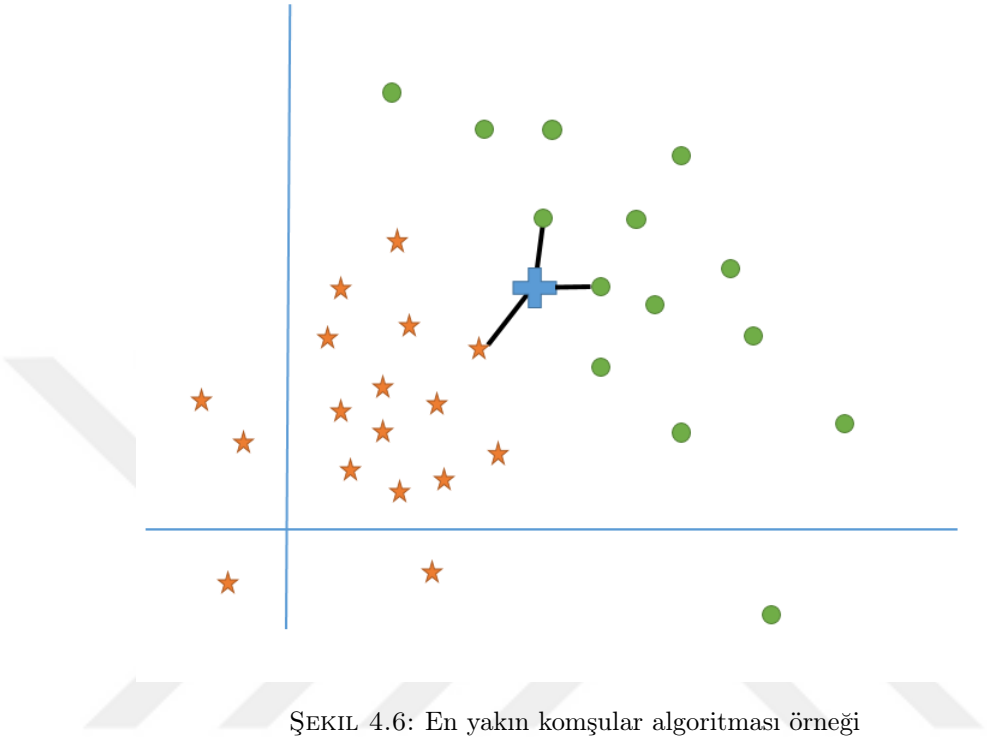
Budama işlemi, ağacın gereksiz uzamasına engel olmasının yanında, verideki gürültüyü de azaltmak için yapılır. Diğer dallara oranla daha az öneme sahip olan veri kaldırılır. Budama işleminin en basit yöntemi yapraklarda başlar ve bu yaprağın en popüler sınıfa sahip her bir düğümünü kaldırır; bu değişiklik doğruluk bozulmazsa tutulur. Buna hata düzeltme denir.

Budama yöntemleri ikiye ayrılır. Erken budama ve Geç budama. Erken budama da ağaç henüz tamamlanmamışken işlem gerçekleştirilir. Geç budamada ise ağaç tamamlandıktan sonra budama işlemi gerçekleştirilir.

4.3 En Yakın Komşular Algoritması (K Nearest Neighbour)

En Yakın Komşular algoritmaları, basit ama güçlü sınıflandırıcılar üretme yeteneklerinden dolayı en akıcı 10 veri madenciliği algoritmasından biri olarak tanımlanmıştır. Basitliğine rağmen, En Yakın Komşular kuralı genellikle rekabetçi sonuçlar verir. Bununla birlikte, En Yakın Komşular'ın en büyük dezavantajlarından biri tembel bir öğrenci olmasıdır, yani çalışma zamanında tüm eğitim verilerini kullanır. Algoritma, eğitim aşamasındaki her eğitim örneğine negatif olmayan bir ağırlık atar ve yalnızca pozitif ağırlıktaki eğitim örnekleri (prototipler olarak) test aşamasında tutulur. Bununla birlikte, algoritma, sınıflandırma amacıyla özel olarak tasarlanmıştır ve regresyon için kullanılamaz [43].

Aşağıdaki dağılıma yeni bir nokta eklensin ve bu nokta sınıflandırmak istensin. K=3 alınırsa, sınıflandırılmış olan elemanlardan 3 tanesini alır. Bu elemanlar ile aralarındaki mesafeye bakar. Kendisine en yakın olan sınıfa dahil olur.



Şekil 4.6: En yakın komşular algoritması örneği

K=3 olarak aldığımız için kendisine en yakın olan 3 eleman ile arasındaki mesafeye bakılacaktır. K için en uygun değeri seçmek için veriler ve sonuçlar çok iyi analiz edilmelidir. K değerini büyük bir değer seçersek, ortamdaki gürültü çok artacağından sonuç sağlıklı olmayacaktır. Genel olarak en sağlıklı k değeri 3-10 arasında seçilmelidir. Aradaki mesafeyi hesaplamak için çeşitli uzaklık formülleri vardır. Bunlardan en çok kullanılanı Öklid algoritmasıdır.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Şekil 4.7: Öklid algoritması formülü

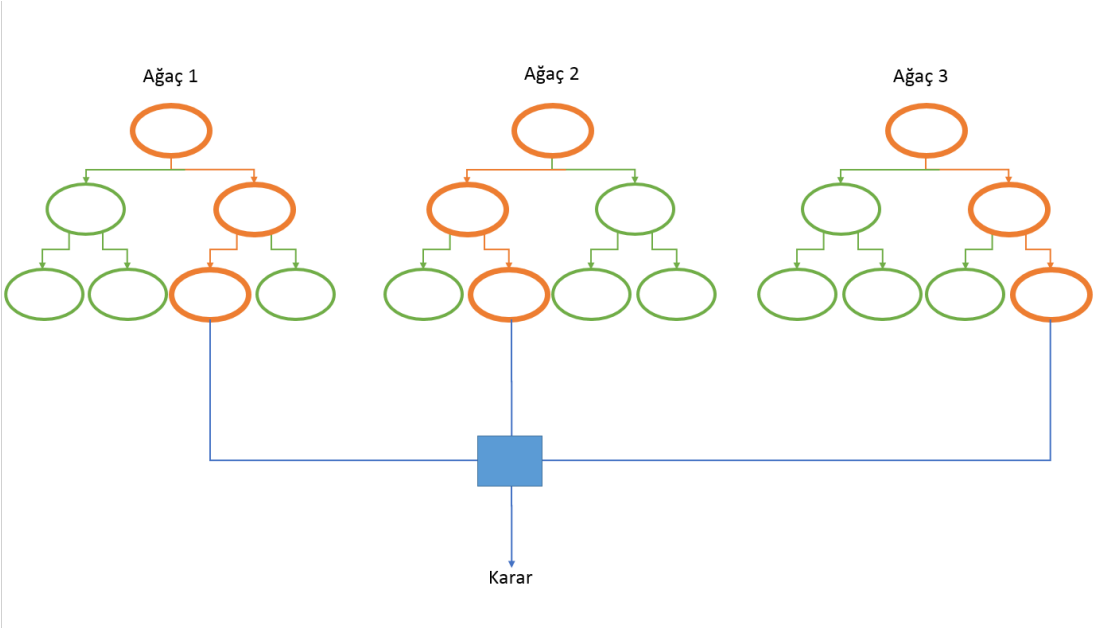
Örnekte en yakın komşuları yeşil elemanlar olduğundan, yeni eklediğimiz elemanımız da yeşil kümeye dâhil olmuştur. Söz konusu değişkenler kategorik olduğunda ise, farklı bir

mesafe ölçümü olan Hamming mesafe ölçümü metodu kullanılmıştır.

4.4 Rastgele Ormanlar (Random Forest)

Denetimli Rastgele Ormanlar, bagging ve ID3 ilkelerinin birlikte kullanarak oluşturulur. Eğitim seti verildiğinde, ID3 fikrine benzeyen rastgele bir alt küme örneklenir (değiştirilir). Ancak, ağacı yetiştirmek için bu önyükleme örneğindeki her durum kullanılmaz. Önyükleme işleminin yaklaşık üçte biri dışarıda bırakılır ve paket dışı veriler olarak kabul edilir. Her ağaçta rastgele bir seçim seçimi değerlendirilir. Paket dışı veriler, ormana ağaçlar eklendikçe bir sınıflandırma hata oranı elde etmek ve girdi değişkeninin (özellik) önemini ölçmek için kullanılır. Orman tamamlandıktan sonra, örneklem toplanma fikrini andıran ormandaki tüm ağaçlar arasında çoğunluk oyu alarak bir sınıflandırma yapılabilir [44].

Rastgele ormanın çalışma prensibi, birden fazla karar ağacı oluşturup, bu ağaçların ortalamasını kullanmaktadır. Bilindiği üzere karar ağaçlarının en büyük problemi aşırı öğrenme problemidir. Rastgele orman algoritmasının her değeri kullanmaması, öğrenme verisi olarak kullandığı verileri değiştirmesi ve n tane ağacın ortalamasını alması gibi özellikleri ile karar ağaçları kullanımından bu açıdan avantajlıdır.



Şekil 4.8: Rastgele ormanlar algoritması örneği

4.5 Naive Bayes Algoritması

Naive Bayes sınıflandırıcısı, esas olarak Bayes teoremi üzerine kurulu tahminlerde bulunma olasılığını kullanan basit bir sınıflandırıcıdır. Yaptığı varsayımlar güçlü değil, ancak birçok uygulamada oldukça iyi performans gösterdiği kanıtlanmıştır. Sınıflandırıcı ayrıca Idiot Bayes, Naive Bayes veya Simple Bayes olarak da adlandırılır. Naive Bayes sınıflandırıcısı, bir sınıfın belirli bir özelliğinin varlığının veya yokluğunun, başka herhangi bir özelliğin varlığı veya yokluğu ile ilgisi olmadığını varsayar. Örneğin, bir nesnenin yuvarlak, kabarık, yaklaşık 4 inç ila 8 inç çapında olması durumunda top olarak kabul edilebilir. Bu özellikler birbirlerine veya diğer özelliklerin varlığına bağlı olsalar bile, bir Naive Bayes sınıflandırıcısı, bu özelliklerin hepsinin, nesnenin bir top olma olasılığına bağımsız olarak katkı sağladığını düşünür [45].

Bu teorem bir rassal değişken için koşullu olasılıklar ile önsel (marjinal) olasılıklar arasındaki ilişkiyi gösterir. Naive bayes sınıflandırıcı basitçe bütün koşullu olasılıkların çarpımıdır.

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

Şekil 4.9: Naive bayes sınıflandırıcı formülü

p(x|Cj) sınıf j'den bir örneğin x olma olasılığı, P(Cj) Sınıf j'nin ilk olasılığı, p(x) herhangi bir örneğin x olma olasılığı, P(Cj|x) x olan bir örneğin sınıf j'den olması olasılığını temsil etmektedir.

Bölüm 5

Uygulama

5.1 Verilerin Toplanması

Yazılım otoritelerinin raporlarında da belirtildiği gibi, test ve makine öğrenmesi çalışmalarının önündeki en büyük engellerden biri, kaliteli verinin olmamasıdır. Hatta birçok kurumda analiz edilebilecek kadar bile test sonuç verisi bulunmamaktadır.

Yazılım yaşam döngüsü prensibi ile çalışan, küçük ve büyük ölçeklerde yazılım yapan bir ekibin yapmış olduğu çalışmalar incelendi. Ekip içerisinde çok donanımlı, iyi üniversitelerden mezun kişiler yer almaktadır. Yapılan incelemelerde görülmüştür ki analiz ekibi sadece işin doküman te etme kısmını yapan yüzeysel bir analiz ekibi değil uçtan uca süreçleri her yönüyle inceleyen bir ekiptir. Analiz ekibinde, farklı tecrübelerde dört adet analist yer almaktadır. Yazılım ekibi, farklı dillerde yazılım yapabilen, farklı yaş ve iş tecrübesine sahip bir ekip olduğu gözlemlenmiştir. Yazılım ekibinde ise sekiz kişi yer almaktadır. Test ekibi, yazılım test süreçleri, metodolojileri konusunda bilgili, yazılım testi konusunda eğitimli bir ekip olduğu gözlemlenmiştir. Testler yüzeysel son kullanıcı testi gibi yapılmamış, tüm test teknikleri kullanılarak testler büyük bir ciddiyetle gerçekleştirilmiştir. Test ekibinde farklı tecrübelere sahip beş adet test uzmanı yer almaktadır. Özetleyecek olursak, Analist Sayısı: 4, Yazılım Mühendisi Sayısı: 8 ve Test Uzmanı Sayısı: 5'dir.

Öncelikle, yazılım kalitesini etkileyebilecek olan tüm etkenlerin bir listesi çıkartıldı. Bu liste dört ana başlık olarak değerlendirildi. Bu başlıklar proje bilgileri, analiz bilgileri, yazılım bilgileri ve test bilgileridir.

5.1.1 Proje Bilgileri

- **Proje Tarihi**: Projenin hangi yıl içerisinde yapıldığının gösterildiği değerdir.
- **Projenin önemi**: Çok Yüksek, Yüksek, Orta ve Düşük olmak üzere dört kademede değerlendirilmektedir. Projenin önemi yönetim tarafından belirlenmektedir.
- **Projenin büyüklüğü:** Çok Büyük, Büyük, Orta ve Küçük olmak üzere dört kademede değerlendirilmektedir. Projenin önemi yönetim tarafından belirlenmektedir.
- **Proje ekibindeki yazılım mühendisi sayısı:** Yazılım projesinde aktif olarak yer alan yazılım mühendisi sayısının gösterildiği değerdir.
- **Proje ekibindeki analist sayısı:** Yazılım projesinde aktif olarak yer alan analist sayısının gösterildiği değerdir.
- **Proje ekibindeki test uzmanı sayısı:** Yazılım projesinde aktif olarak yer alan test uzmanı sayısının gösterildiği değerdir.

5.1.2 Analiz Bilgileri

- **Analistin yaşı**: İlgili projede çalışan analistin yaşının belirtildiği değerdir.
- **Analistin tecrübesi (yıl):** İlgili projede çalışan analistin toplam tecrübesinin belirtildiği değerdir.
- **Benzer proje tecrübesi var mı?** : İlgili projede çalışan analistin daha önce benzer bir projede çalışıp çalışmadığının gösterildiği değerdir.
- **Analiz için harcanan süre (Adam/Gün):** İlgili projede çalışan analistin, analiz için harcadığı toplam adam/gün sayısının belirtildiği değerdir.

5.1.3 Yazılım Bilgileri

- **Yazılım mühendisinin yaşı:** İlgili projede çalışan yazılım mühendisinin yaşının belirtildiği değerdir.
- **Yazılım mühendisinin tecrübesi (yıl)**: İlgili projede çalışan yazılım mühendisinin toplam tecrübesinin belirtildiği değerdir.

- **Benzer proje tecrübesi var mı?** : İlgili projede çalışan yazılım mühendisinin daha önce benzer bir projede çalışıp çalışmadığının gösterildiği değerdir.
- **Yazılım için harcanan süre (Adam/Gün):** İlgili projede çalışan yazılım mühendisinin, yazılım için harcadığı toplam adam/gün sayısının belirtildiği değerdir
- **Yazılım mühendisinin bildiği yazılım dili sayısı?:** Yazılım mühendisinin, projeden bağımsız olarak, kaç adet farklı yazılım diline hakim olduğunun belirtildiği değerdir.
- **Geliştirme metodolojisi:** İlgili projenin hangi yazılım metodolojisine göre geliştirildiğinin belirtildiği alandır. Seçenek olarak; Waterfall, Iterative, Scrum Lean ve Agile verilmiştir.
- **Yazılım dili:** İlgili projenin hangi dil kullanılarak yazıldığının belirtildiği alandır.
- **Kod kalite kontrol aracı kullanıldı mı?:** İlgili projede yazılım geliştirme aşamalarında, kod kalite değerlendirme aracı kullanıp, kullanılmadığının belirtildiği alandır.
- **Satır sayısı:** İlgili projenin kodunun kaç satırdan oluştuğunun belirtildiği alandır.
- **Hedef platformu:** İlgili projenin hangi platformda çalıştırılacağının belirtildiği alandır. Seçenek olarak; Web ve Desktop verilmiştir.
- **Birim test yazıldı mı?:** İlgili projede yazılım geliştirme aşamasında, birim testlerin yazılıp yazılmadığının belirtildiği alandır.
- **Yazılımcı testi yapıldı mı?:** İlgili projede yazılım geliştirme aşamasında, yazılımcı testlerinin yapılıp yapılmadığının belirtildiği alandır.
- **Kod gözden geçirme yapıldı mı?:** İlgili projede yazılım geliştirme aşamasında, kod gözden geçirmenin yapılıp yapılmadığının belirtildiği alandır.
- **Hazır platform üzerine ürün geliştirmesi mi?:** İlgili proje, sıfırdan mı yazıldı, yoksa hazır bir platform üzerinde geliştirme mi yapıldığının belirtildiği alandır.
- **Yazılımda danışmanlık alındı mı?:** İlgili projede, kurum ve birim dışından danışmanlardan yardım alınıp, alınmadığının belirtildiği alandır.

5.1.4 Test Bilgileri

- **Test mühensinin yaşı:** İlgili projede çalışan test uzmanının yaşının belirtildiği değerdir.
- **Test mühensinini tecrübesi (yıl):** İlgili projede çalışan test uzmanının toplam tecrübesinin belirtildiği değerdir.
- **Benzer proje tecrübesi var mı?:** İlgili projede çalışan test uzmanının daha önce benzer bir projede çalışıp çalışmadığının gösterildiği değerdir.
- **Test için harcanan süre (Adam/Gün):** İlgili projede çalışan test uzmanının, test için harcadığı toplam adam/gün sayısının belirtildiği değerdir
- **Test otomasyon kullanıldı mı?:** İlgili projede yazılım testi yapılırken, test otomasyon kodunun yazılıp yazılmadığının belirtildiği alandır.
- **Test otomasyon yüzdesi:** İlgili projede eğer yazılım test otomasyonu kodu yazıldı ise, toplam test durumlarının yüzdesel olarak ne kadarının otomatize edildiğinin belirtildiği alandır.
- **Kaç farklı tarayıcıda test edildi ?:** İlgili proje eğer web tabanlı bir proje ise, ilgili testin kaç farklı tarayıcıda yapıldığının belirtildiği alandır.
- **Durum kaç kere test edildi?:** İlgili proje kaç kere yazılımcı ve test uzmanı arasında gidip geldi, regresyon testi kaç kere yapıldı belirtilen alandır.
- **Test senaryo sayısı:** İlgili proje test edilirken, test uzmanı tarafından kaç farklı senaryoda test edildiğinin belirtildiği alandır.
- **Test Senaryo Gruplama:** Değerlendirmeye alınan projeler birbirinden bağımsız ve çok farklı projelerdir. Bu nedenle de yazılan test durumları da birbirinden farklılıklar göstermektedir. Tüm yazılım durumlarını ortak paydalarda birleştirme adına senaryo gruplama çalışması yapılmıştır. Her durum tek tek okunarak ilgili gruba eklenmiştir.
  - Login sayfası şifre ile giriş: ilgili test durumunda, login sayfası ile test senaryosu olması durumunda, kullanılan gruplama çeşididir.
  - Login sayfası şifre ile giriş: ilgili test durumunda, login sayfası ile test senaryosu olması durumunda, kullanılan gruplama çeşididir.

- Akış ilerletme: İlgili test durumunda, yapılan eylem sonucunda, yazılan akışın ilerlemesi senaryolarında, kullanılan gruplama çeşididir.
- Buton aktif etme: Test senaryosu kapsamında yapılan eylem sonucunda, bir butonun aktif olması senaryolarında, kullanılan gruplama çeşididir.
- Combobox veri seçimi: Test senaryosu kapsamında bir combobox'tan seçim yapılması durumunda, kullanılan gruplama çeşididir.
- Dosya indirme: Test senaryosu kapsamında bir linkten veya bir butona tıklayarak her tür yapıda dosya indirilmesi durumunda, kullanılan gruplama çeşididir.
- Dosya Yükleme: Test senaryosu kapsamında seçim yaparak, belirtilen ortama her türlü dosyanın yüklenmesi durumunda, kullanılan gruplama çeşididir.
- Ekran Tasarımı: Analizde belirtilen ekran tasarımlarının, mevcut ekran tasarımları ile karşılaştırıldıkları durumlarda, kullanılan gruplama çeşididir.
- Filtreleme: İlgili test senaryoları kapsamında, filtreleme özelliklerinin test edildiği durumlarda, kullanılan gruplama çeşididir.
- Grupbox veri seçimi: Test senaryosu kapsamında bir grupbox'tan seçim yapılması durumunda, kullanılan gruplama çeşididir.
- İptal: İlgili test senaryoları kapsamında, iptal butonlarının test edildiği durumlarda, kullanılan gruplama çeşididir.
- Listbox veri seçimi: Test senaryosu kapsamında bir listbox'tan seçim yapılması durumunda, kullanılan gruplama çeşididir.
- Mail gönderme: İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda mail gitmesi gereken durumlarda, kullanılan gruplama çeşididir.
- Otomatik veri getirilmesi: İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda, bazı alanlara otomatik olarak veri getirilmesi durumlarında kullanılan gruplama çeşididir.
- Radio button veri seçimi: Test senaryosu kapsamında bir radiobutton'dan seçim yapılması durumunda, kullanılan gruplama çeşididir.
- Sayfa geçişi: İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda, sayfanın değişmesi gereken durumunda kullanılan gruplama çeşididir.
- Sayısal veri girişi: İlgili test durumunda, sayısal bir veri girişi ile alakalı olan test durumlarında kullanılan gruplama çeşididir.

- Statü değişimi: : İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda, statünün değişmesi gereken durumunda kullanılan gruplama çeşididir.
- Tab geçişi: Test edilen ilgili sayfada, tab geçişi ile ilgili bir senaryo olması durumunda kullanılan gruplama çeşididir.
- Tarih veri girişi: Test edilen ekranda, tarih veri girişi yapılan bir alan olması durumunda kullanılan gruplama çeşididir.
- Tetikleme: İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda, başka bir akışın tetiklendiği durumunda kullanılan gruplama çeşididir.
- Text veri girişi: Test edilen ekranda, text veri girişi yapılan bir alan olması durumunda kullanılan gruplama çeşididir.
- Uyarı mesajı: İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda, ekrana uyarı mesajı verilmesi gereken durumunda kullanılan gruplama çeşididir.
- Veri değiştirme: İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda, ilgili veri başka bir veri ile değiştiriliyor ise kullanılan gruplama çeşididir.
- Veri doğruluk kontrolü: İlgili test senaryosunda, belirli bir alanda bulunan verinin, doğru olup olmadığının kontrolünün yapıldığı durumlarda kullanılan gruplama çeşididir.
- Veri eşleştirme: İlgili test senaryosunda, belirli bir alanda bulunan verinin, yine başka bir alanda bulunan veri ile karşılaştırılıp, uygun ise birleştirildiği durumlarda kullanılan gruplama çeşididir.
- Veri okuma: İlgili test senaryosunda, belirli bir alanda bulunan verinin okunduğu ve bu veri ile aksiyon alındığı durumlarda kullanılan gruplama çeşididir.
- Veri okuma ve yazma: İlgili test senaryosunda, belirli bir alanda bulunan verinin okunduğu ve belirtilen başka bir alana yazıldığı durumlarda kullanılan gruplama çeşididir.
- Veri oluşturma: İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda, bir verinin oluştuğu durumlarda kullanılan gruplama çeşididir.
- Veri silme: İlgili test senaryoları kapsamında, yapılan aksiyon sonucunda, bir verinin silindiği durumlarda kullanılan gruplama çeşididir.
- Veri yazma: İlgili test senaryosunda diğer verilerden ve aksiyonlardan bağımsız olarak veri yazma durumlarında kullanılan gruplama çeşididir.

- Yetki: İlgili test senaryosunda belirtilen senaryonun gerçekleştirilmesi için, test edilen kullanıcıda ilgili yetkinin olup olmadığının kontrol edildiği durumlarda kullanılan gruplama çeşididir.
- Zorunlu alan kontrolü: Analizde zorunlu alan olarak belirtilen alanların, gerçekten zorunlu alan olarak tanımlanıp, tanımlanmadığının kontrol edildiği durumlarda kullanılan gruplama çeşididir.

Hata Durumu: İlgili durumun test sonucunda hangi statüde olduğunun belirtildiği alandır. Dokuz farklı hata durumu bulunmaktadır.

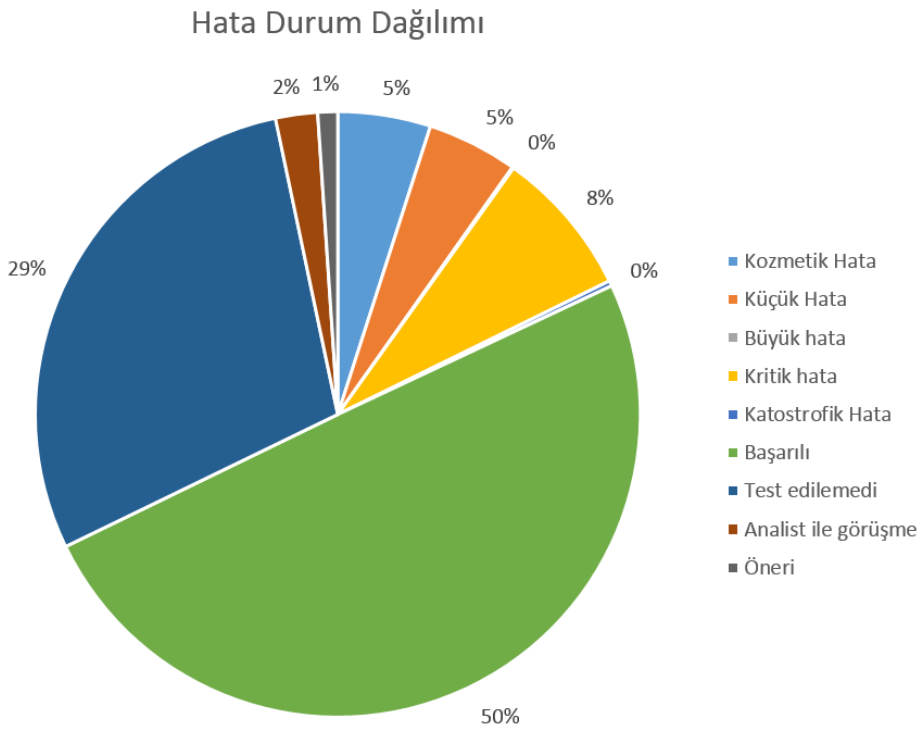
5.2 Verilerin İstatistikleri

Toplam 6676 adet durum incelenmiştir. Sonuçlar incelediğinde en düşük hata sayısı 5 adetle Büyük Hata ve 19 adetle Katostrofik Hata olduğu görülüştür. Bu hata sayılarının az olmasının sebebi çok ciddi durumlarda kullanılması, ölümcül ve yıkıcı hata türü olmasıdır. Durumların 71 adedine test uzmanı tarafından öneri yazılmıştır. Hem analist hem de yazılımcıyı ilgilendiren durum önerilerinde kullanılan bir hata türüdür. Bu toplam verinin yaklaşık olarak 1%'ine denk gelmektedir. 2%'lik pay ile arkasından Analist ile Görüşme hata tipi gelmektedir. 150 adet durumda test uzmanı, analist ile görüşme talep etmiştir. Bu durumlar yazılımcıyı ilgilendirmeyen fakat analizde değişiklik yapılmasını öneren durumlarda kullanılmaktadır. 323 adet durumda Küçük Hata ile karşılaşılmıştır. Bu toplam verinin yaklaşık olarak 5%'ine denk gelmektedir. 331 adet durumda ise Kozmetik Hata ile karşılaşılmıştır. Bu da toplam verinin yaklaşık olarak 5%'ine denk gelmektedir. Kozmetik hata görsel hatalardır. Önem derecesi olarak düşüktür. 525 adet durumda ise Kritik Hata ile karşılaşılmıştır. Bu toplam verimizin yaklaşık olarak 8%'ine denk gelmektedir. Kritik hata gereksinimin karşılanmadığı durumlarda kullanılan hata türüdür. Bu nedenle önem derecesi olarak yüksektir. İlgili testler esnasında 6676 adet durumun, 1928 adedi test edilememiştir. Bu toplam verimizin yaklaşık olarak 29%'una denk gelmektedir.

En yüksek test durum sonucu olarak Başarılı durum sonucu çıkmaktadır. 3324 adet durum sonucu olarak başarılı olmuştur. Bu da toplam verimizin yaklaşık olarak 50%'sine denk gelmektedir. Sadece Başarılı test durum sonucunda test geçti olarak sayılmaktadır. Diğer tüm durum sonuçlarında test başarısızdır.

Tablo 5.1: Hata durumu (Test Sonucu) ve Hata durum (Test Sonucu) sayısı

Hata Durumu	Hata Durum Sayısı	Hata Durum Yüzdesi
Kozmetik Hata	331	%5
Küçük Hata	323	%5
Büyük Hata	5	%0
Kritik Hata	525	%8
Katostrofik Hata	19	%0
Başarılı	3324	%50
Test edilemedi	1928	%29
Analist ile görüşme	150	%2
Öneri	71	%1
Toplam	6676	%100



Şekil 5.1: Hata durum dağılımı

Şekil-5.1'te verilen grafikte de görüldüğü gibi, 50% başarı ile ilerleyen test durumları mevcuttur. Kalan 50% ise diğer durumlar arasında paylaşılmaktadır. Dikkat çeken yükseklikte olan bir diğer değer 29% ile Test Edilemedi durumudur. Bu değerlerin yüksek

çıkmasının birkaç sebebi olabilir, test durumları oluşturulurken analiz tamamlanmış olduğundan, tüm senaryoların test durumları çıkartılıyor. Yazılımcı ise tüm yazılımı bitirmeden teste parça parça gönderirse, test edilemeyen senaryo sayısı yükseliyor. Bir diğer sebep ise; Katostrofik Hata ve Büyük Hata gibi testin devam etmesini engelleyen durumlarda diğer senaryolara geçilememesidir.

Yüksek sayıda çıkan test edilemeyen durumların da sayısı dikkat çekici derecede yüksektir. Yazılımcı testi denen kavramın önemi bu bilgi ile daha da artmaktadır. Tek neden bu olmasa da yazılımcı tarafından test edilmeden gönderilen yazılımlar test uzmanının işini uzatmakta, testin yazılımcı ile test uzmanı arasında gidiş geliş sayısını arttırmaktadır.

Test durumları ortak paydalarda değerlendirmek için gruplandırılmıştır. Bu gruplandırma bize hangi alanlarda daha çok hata yapıldığı konusunda bilgi vermektedir. Böylece test başlamadan önce ilgili senaryoda çok hata yapılan durumlardan başlanırsa, hataya ulaşma süresi kısılarak, test maliyeti azaltılabilir.

Tablo-5.2'de en yüksek paydaya sahip olan gruptan başlayacak olursak, bu zamana kadar incelenen senaryolarda sayısı az olmasına rağmen eposta atmada 100% başarısızlık yaşanmıştır. Bir sonraki testlerde mail senaryolarından başlarsak hata yakalama şansımız çok yüksek olur. Aynı şekilde, 100% ile 90% arasında olan değerleri inceleyecek olursak, 97% yetkilendirmeler, 91% tetiklemeler bu araya girmektedir. Yetkilendirme problemleri genel olarak her yazılımda çok karşılaşılan durumlardır. Sadece yazılım özelinde değil, network tarafında da yetki problemleri bu yüzdenin artmasına neden olmaktadır.

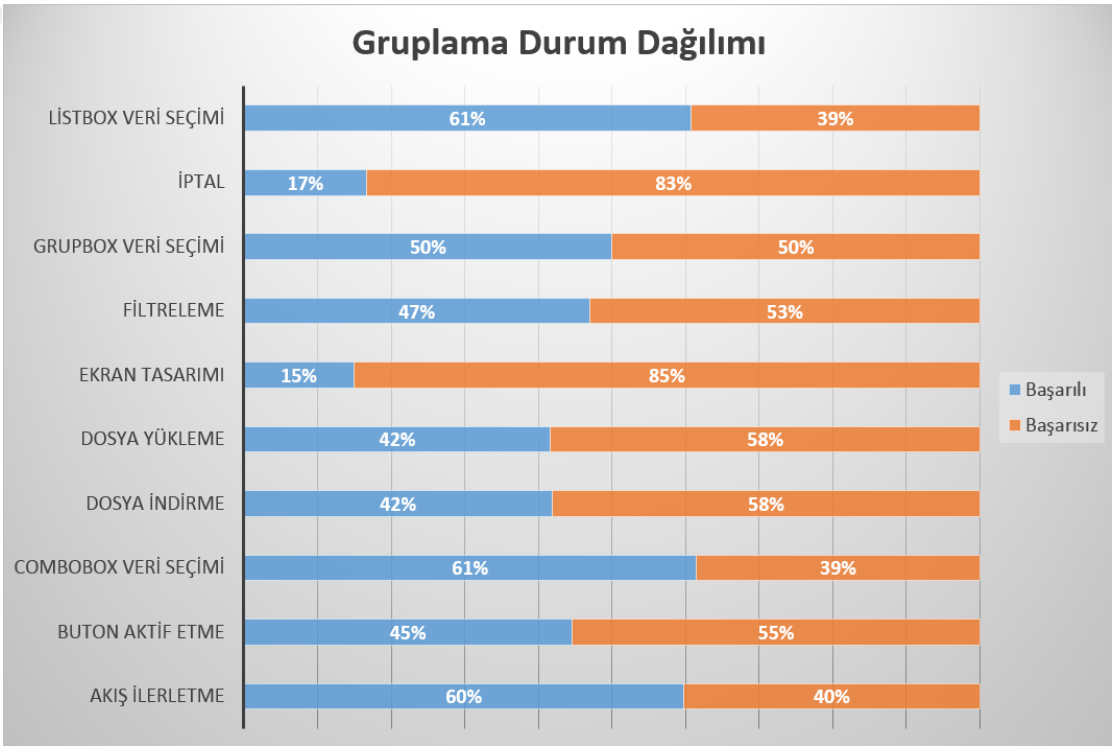
Bir diğer yaklaşım çeşidi, yönetimsel olarak, yazılımcıların bu alanlarda neden çok hata yaptıklarını tespit etmek olacaktır. Yazılımcıların karşılaştıkları zorlukları belirleyip çözüm bulmak, gerekli ise eğitim aldırmak, yetersiz ise ekipmanları ve donanımları geliştirmek veya en basit yöntemle yazılımcıların bu tarz yazılımları geliştirirken daha dikkatli davranmasını sağlamak, hata sayısını azaltacaktır.

Tablo 5.2: Yazılım gruplama senaryo sayıları ve başarı durumları

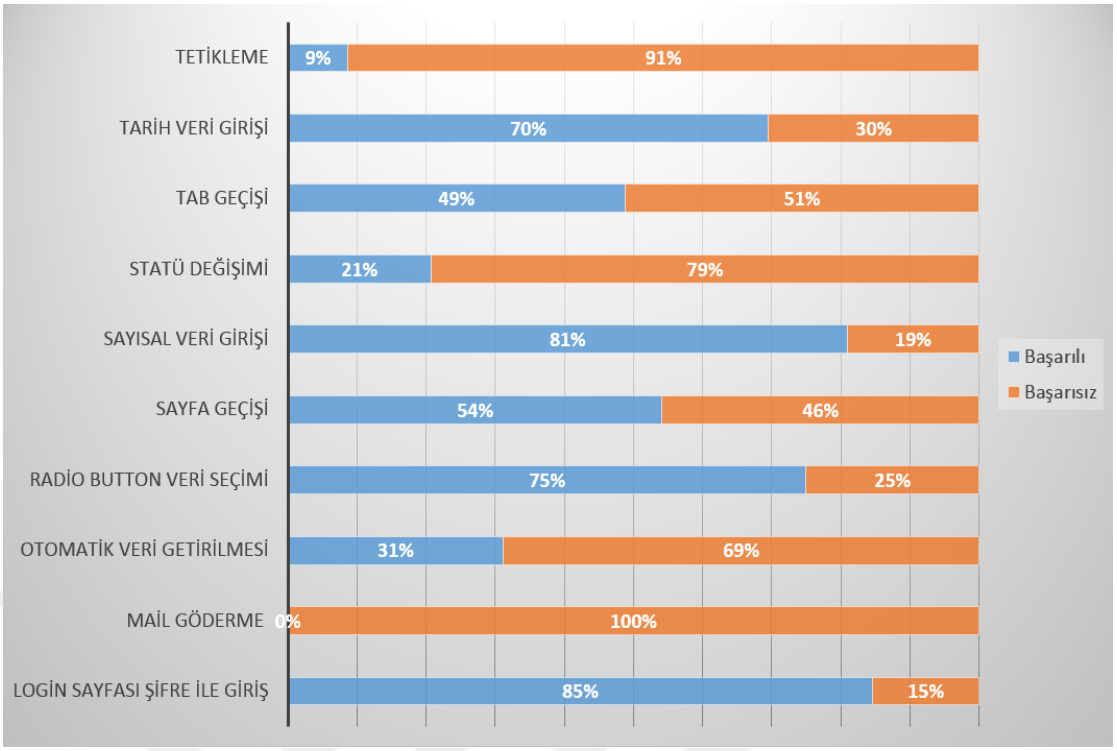
Yazılım Türü Gruplama	Toplam	Başarılı	Başarısız	% Başarılı	% Başarısız
Akış ilerletme	516	308	208	60%	40%
Buton aktif etme	202	90	112	45%	55%
Combobox veri seçimi	70	43	27	61%	39%
Dosya indirme	43	18	25	42%	58%
Dosya Yükleme	12	5	7	42%	58%
Ekran Tasarımı	980	146	834	15%	85%
Filtreleme	181	85	96	47%	53%
Grupbox veri seçimi	6	3	3	50%	50%
İptal	6	1	5	17%	83%
Listbox veri seçimi	28	17	11	61%	39%
Login sayfası şifre ile giriş	52	44	8	85%	15%
Mail göderme	2	0	2	0%	100%
Otomatik veri getirilmesi	186	58	128	31%	69%
Radio button veri seçimi	12	9	3	75%	25%
Sayfa geçişi	24	13	11	54%	46%
Sayısal veri girişi	42	34	8	81%	19%
Statü değişimi	227	47	180	21%	79%
Tab geçişi	41	20	21	49%	51%
Tarih veri girişi	46	32	14	70%	30%
Tetikleme	23	2	21	9%	91%
Text veri girişi	243	192	51	79%	21%
Uyarı mesajı	212	55	157	26%	74%
Veri değiştirme	375	64	311	17%	83%
Veri doğruluk kontrolü	1567	1070	497	68%	32%
Veri eşleştirme	29	12	17	41%	59%
Veri okuma	5	5	0	100%	0%
Veri okuma ve yazma	1065	419	646	39%	61%
Veri oluşturma	107	56	51	52%	48%
Veri silme	78	43	35	55%	45%
Veri yazma	150	105	45	70%	30%
Yetki	29	1	28	3%	97%
Zorunlu alan kontrolü	117	60	57	51%	49%

Tam tersi bakış açısı ile bakıldığında, 100% başarı ile testi geçen gruplama çeşidi veri okuma çeşididir. Test durum sıralamasından en son olarak veri okuma senaryolarına bakabiliriz. Hata çıkma olasılığı çok daha düşük olacaktır. Aynı şekilde, 100% ile 90% arasında olan başarılı değerleri incelendiğinde, bu aralıkta bir değer olmadığını görülmektedir. Şu sonucu çıkmaktadır, aslında çok nadir durumlar dışında her çeşit hata yapılabilir. Test sırasında bazı senaryo çeşitlerini sona bırakmak istesek bu tabloya göre, login ve sayısal veri girişi grupları seçilmelidir.

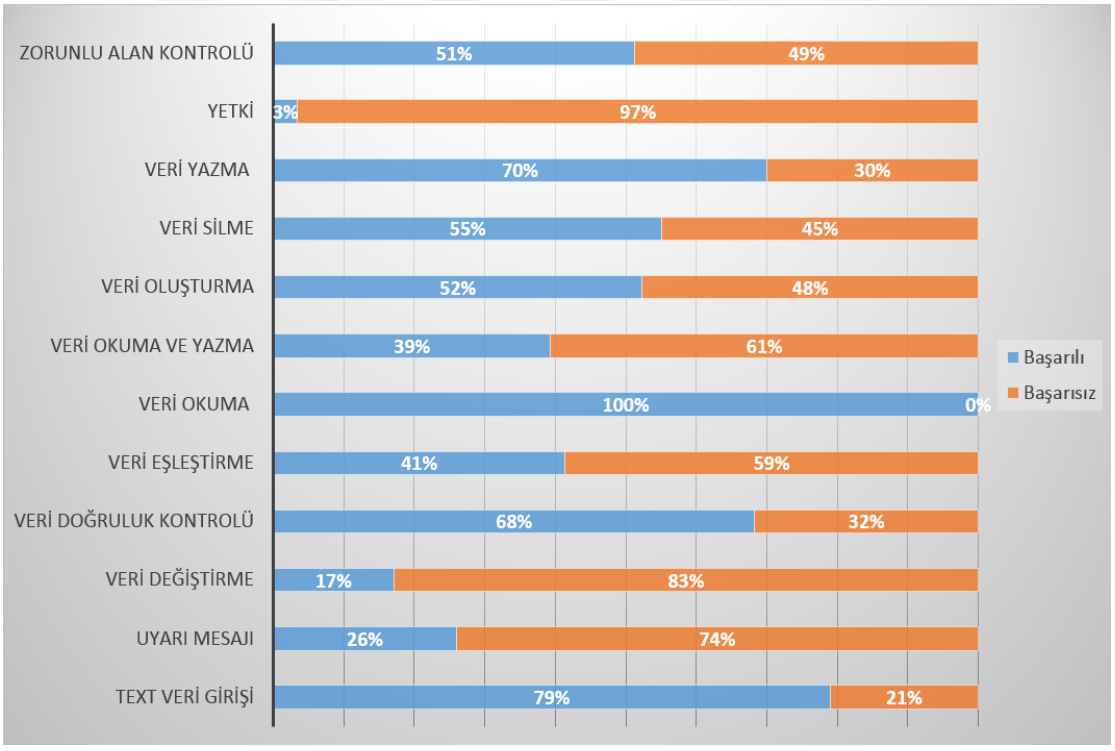
Bir diğer yaklaşım çeşidi, yönetimsel olarak, yazılımcıların bu alanlarda neden az hata yaptıklarını tespit etmek olacaktır. Yazılımcıların güçlü yanlarını öğrenmek ve bu alanlara daha fazla yatırım yapmak, başarı oranını artıracaktır.



Şekil 5.2: Gruplama durum dağılımı 1



Şekil 5.3: Gruplama durum dağılımı 2

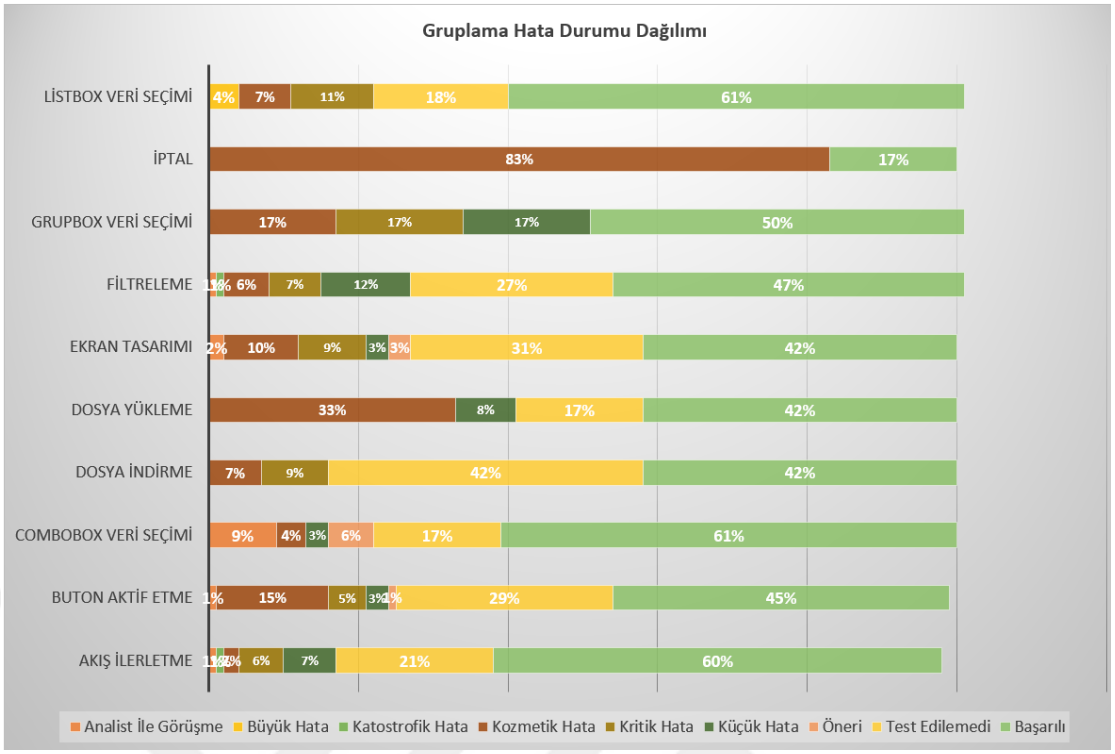


Şekil 5.4: Gruplama durum dağılımı 3

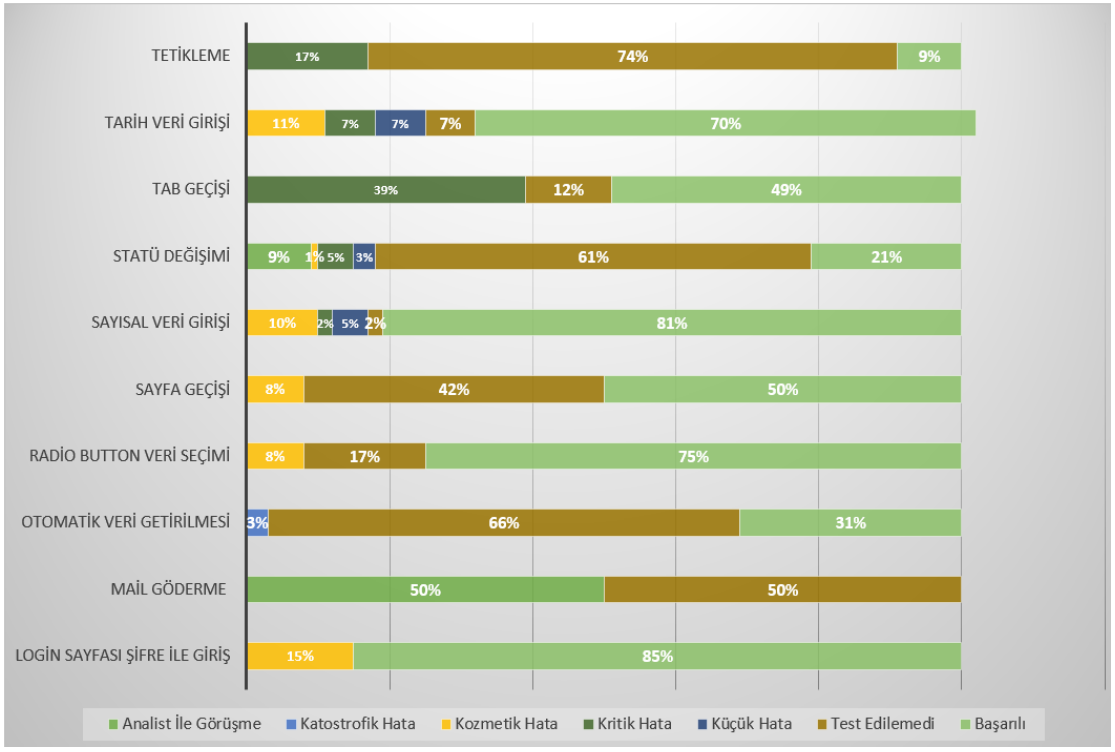
Tablo 5.3: Yazılım türüne göre hata sayıları

	Analist İle Görüşme	Büyük Hata	Katostrofik Hata	Kozmetik Hata	Kritik Hata	Küçük Hata	Öneri	Test Edilemedi	Başarılı
Akış ilerletme	7	1	6	12	33	37	2	110	308
Buton aktif etme	3	1	0	31	10	7	2	58	90
Combobox veri seçimi	6	0	0	3	0	2	4	12	43
Dosya indirme	0	0	0	3	4	0	0	18	18
Dosya Yükleme	0	0	0	4	0	1	0	2	5
Ekran Tasarımı	21	0	1	95	87	27	34	299	416
Filtreleme	1	0	1	11	13	21	0	49	85
Grupbox veri seçimi	0	0	0	1	1	1	0	0	3
İptal	0	0	0	5	0	0	0	0	1
Listbox veri seçimi	0	1	0	2	3	0	0	5	17
Login sayfası şifre ile giriş	0	0	0	8	0	0	0	0	44
Mail göderme	1	0	0	0	0	0	0	1	0
Otomatik veri getirilmesi	0	0	6	0	0	0	0	123	57
Radio button veri seçimi	0	0	0	1	0	0	0	2	9
Sayfa geçişi	0	0	0	2	0	0	0	10	12
Sayısal veri girişi	0	0	0	4	1	2	0	1	34
Statü değişimi	20	0	0	3	12	6	0	139	47
Tab geçişi	0	0	0	0	16	0	0	5	20
Tarih veri girişi	0	0	0	5	3	3	0	3	32
Tetikleme	0	0	0	0	4	0	0	17	2
Text veri girişi	5	0	0	6	11	13	0	16	192
Uyarı mesajı	4	0	2	10	38	5	15	83	55
Veri değiştirme	10	0	0	2	26	39	0	234	64
Veri doğruluk kontrolü	8	0	0	39	167	83	6	195	1069
Veri eşleştirme	3	0	0	0	1	5	0	8	12
Veri okuma	0	0	0	0	0	0	0	0	5
Veri okuma ve yazma	50	1	0	25	70	52	5	443	419
Veri oluşturma	3	0	3	6	4	3	2	30	56
Veri silme	3	0	0	5	19	1	0	7	43
Veri yazma	3	0	0	24	0	5	0	13	105
Yetki	0	0	0	17	0	1	0	10	1
Zorunlu alan kontrolü	2	1	0	7	2	9	1	35	60

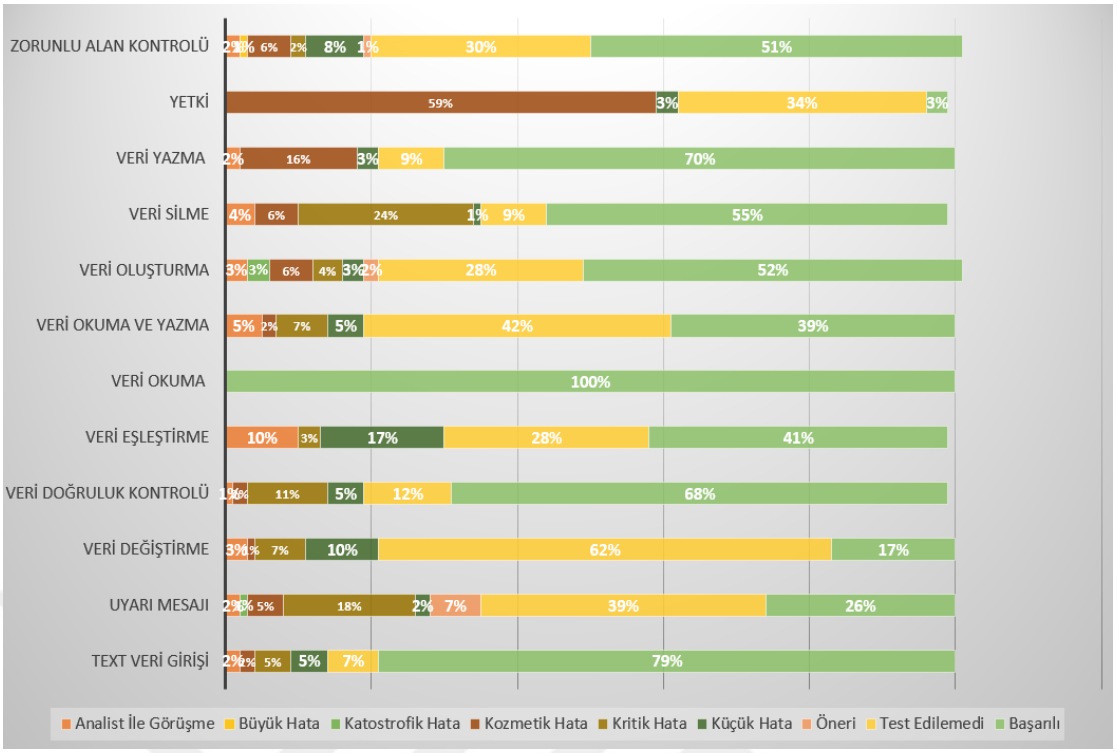
Tablo-5.3'te en ölümcül hata alınan durumları inceleyecek olursak, akış ilerletme'de toplamda 6 kere katostrofik hata alındığını görülmektedir. Akış mantığında bu hata çeşidi aslında beklenen bir hata çeşididir. Bu tarz yazılımlarda hata var ise akış ilerlemez ve test durur. Testin durduğu hata çeşidi ise katostrofik hatadır. Sayının bu kadar yüksek olmasının nedeni aslında işin mantığından kaynaklanmaktadır.



Şekil 5.5: Gruplama hata dağılımı 1



Şekil 5.6: Gruplama hata dağılımı 2



Şekil 5.7: Gruplama hata dağılımı 3

Bütün projelerin ilk 5 test başarı oranları ortalamaları Tablo-5.4'da verilmiştir.

Tablo 5.4: Test başarı ortalaması

	Tüm projelerin başarı ortalaması
1. Test	20%
2. Test	34%
3. Test	49%
4. Test	58%
5. Test	71%

Bütün projelerin ilk 5 test başarı oranları ortalamaları Tablo-5.4'da verilmiştir. Test başarı ortalaması, bütün projelerin (10 adet) ilk testlerinin sonuç başarılarının ortalamasından oluşmaktadır. Projeden bağımsız olarak ilk test sonucu ortalama 20% oranında başarılıdır. Projelerde yapılan test sayısı arttıkça başarı oranı da artmaktadır.

Tablo 5.5: Proje 1 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	31	34	65	48	Ortalama Üstüne Başarı
2. Test	42	22	64	66	Ortalama Üstüne Başarı
3. Test	47	17	64	73	Ortalama Üstüne Başarı
4. Test	54	9	63	86	Ortalama Üstüne Başarı
5. Test	57	6	63	90	Ortalama Üstüne Başarı

Proje 1 test sonuçları Tablo-5.5'te verilmiştir. Test sürecinin sonunda 90% başarılı olarak canlıya alınmıştır. 10% 'luk kısım düşük önem deresinde olduğundan canlıya çıkılmasını etkilememiştir. 48% 'den başlayan test başarısı 5 kere düzeltmeden sonra 90% 'a çıkmıştır.

48% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasının üstünde olduğunu göstermektedir. Yazılım başarı oranı doğrusal artışa yakın olarak yükselmiştir. Test içerisinde karşılaşılan hataların çok büyük veya testi engelleyen hatalar olmadığı sonucuna ulaşabiliriz. Tüm başarı oranları incelendiğinde her test için ortalama başarının üzerinde olduğunu görmekteyiz. Proje 1 'in önemli özellikleri şu şekildedir,

- Orta önem düzeyine sahip bir projedir.
- Proje büyüklük olarak orta derecede değerlendirilmektedir.
- 1 adet analist, 1 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken C# dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmamıştır.
- Yazılımcı testi yapılmamıştır.
- 3 farklı tarayıcıda web testleri gerçekleştirilmiştir.
- 10% test otomasyon yazılmıştır.
- 65 adet test durumu (senaryosu) yazılmıştır.

- Test yazılım mühendisi ve test uzmanı arasında 5 kere gidip geldikten sonra, canlıya alınmıştır.
- Test için 8 adam gün harcanmıştır.

Tablo 5.6: Proje 2 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	11	41	52	21	Ortalama Üstüne Başarı
2. Test	14	38	52	27	Ortalama Altında Başarı
3. Test	27	25	52	52	Ortalama Üstüne Başarı
4. Test	41	11	52	79	Ortalama Üstüne Başarı
5. Test	46	6	52	88	Ortalama Üstüne Başarı
6. Test	52	0	52	100	

Proje 2 test sonuçları Tablo-5.6'da verilmiştir. Test sürecinin sonunda 100% başarılı olarak canlıya alınmıştır. 21%'den başlayan test başarısı 6 kere düzeltmeden sonra 100%'e çıkmıştır. 21% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasını yakaladığını göstermektedir. Çok hızlı şekilde başarı oranının yükselmesi test içerisinde katostofik hata çeşidi olan hatanın mevcut olduğu göstermektedir. Dolayısı ile de test edilemeyen senaryo sayısı da, bu durumun bir sonucu olarak artmaktadır. 2. Testten sonra hızlıca başarı oranının yükselmesi bu testi engelleyen hatanın çözülmesi ile açıklanabilir.

Proje 2 'nin önemli özellikleri şu şekildedir,

- Orta önem düzeyine sahip bir projedir.
- Proje büyüklük olarak orta derecede değerlendirilmektedir.
- 1 adet analist, 1 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken C# dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmamıştır.
- Yazılımcı testi yapılmamıştır.

- 3 farklı tarayıcıda web testleri gerçekleştirilmiştir.
- Test otomasyon yazılmıştır.
- 52 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 6 kere gidip geldikten sonra, canlıya alınmıştır.
- Test için 15 adam gün harcanmıştır.

Tablo 5.7: Proje 3 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	8	53	61	13	Ortalama Altında Başarı
2. Test	40	21	61	66	Ortalama Üstüne Başarı
3. Test	49	12	61	80	Ortalama Üstüne Başarı
4. Test	56	5	61	92	Ortalama Üstüne Başarı
5. Test	61	0	61	100	Ortalama Üstüne Başarı

Proje 3 test sonuçları Tablo-5.7'de verilmiştir. Test sürecinin sonunda 100% başarılı olarak canlıya alınmıştır. 13%'den başlayan test başarısı 5 kere düzeltmeden sonra 100%'e çıkmıştır.

13% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasının altında kaldığını göstermektedir. Çok hızlı şekilde başarı oranının yükselmesi test içerisinde katostofik hata çeşidi olan hatanın mevcut olduğu göstermektedir. Dolayısı ile de test edilemeyen senaryo sayısı da, bu durumun bir sonucu olarak artmaktadır. 2. Testten sonra hızlıca başarı oranının yükselmesi bu testi engelleyen hatanın çözülmesi ile açıklanabilir.

Bu proje ile ilgili bir diğer durum yazılımın tam olarak yazılımcı testi yapılmadan teste gönderilmesidir. İlk olarak test uzmanı tarafından test edildikten sonra yazılımcıya gönderilmiş. Yazılımcı testi sonrasında başarı 66%'ya çıkmıştır. Bu da aslında yazılımcı testinin yazılım başarı oranında ne kadar önemli olduğunu göstermektedir.

Proje 3 'ün önemli özellikleri şu şekildedir,

- Orta önem düzeyine sahip bir projedir.

- Proje büyüklük olarak küçük derecede değerlendirilmektedir.
- 1 adet analist, 1 adet yazılım mühendisi, 2 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken C# dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmamıştır.
- Yazılımcı testi yapılmamıştır.
- 3 farklı tarayıcıda web testleri gerçekleştirilmiştir.
- Test otomasyon yazılmıştır.
- 61 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 5 kere gidip geldikten sonra, canlıya alınmıştır. Test için 13 adam gün harcanmıştır.

Tablo 5.8: Proje 4 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	5	34	39	13	Ortalama Altında Başarı
2. Test	12	27	39	31	Ortalama Altında Başarı
3. Test	18	21	39	46	Ortalama Altında Başarı
4. Test	18	21	39	46	Ortalama Altında Başarı
5. Test	31	8	39	79	Ortalama Üstünde Başarı
6. Test	39	0	39	100	

Proje 4 test sonuçları Tablo-5.8'de verilmiştir. Test sürecinin sonunda 100% başarılı olarak canlıya alınmıştır. 13%'den başlayan test başarısı 6 kere düzeltmeden sonra 100%'e çıkmıştır.

13% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasının altında kaldığını göstermektedir. Çok hızlı şekilde başarı oranının yükselmesi test içerisinde katostofik hata çeşidi olan hatanın mevcut olduğu göstermektedir. Dolayısı ile de test edilemeyen senaryo sayısı da, bu durumun bir sonucu olarak artmaktadır. 2. Testten sonra hızlıca başarı oranının yükselmesi bu testi engelleyen hatanın çözülmesi ile açıklanabilir.

Bu proje ile ilgili bir diğer durum da aynı proje 3'te olduğu gibi yazılımın tam olarak yazılımcı testi yapılmadan teste gönderilmesidir. Sonrasında devam eden testlerde oran linnere yakın bir şekilde yükselmiştir.

Proje 4 'ün önemli özellikleri şu şekildedir,

- Orta önem düzeyine sahip bir projedir.
- Proje büyüklük olarak küçük derecede değerlendirilmektedir.
- 1 adet analist, 1 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken C# dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmamıştır.
- Yazılımcı testi yapılmamıştır.
- 1 tarayıcıda web testleri gerçekleştirilmiştir.
- Test otomasyon yazılmıştır.
- 39 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 6 kere gidip geldikten sonra, canlıya alınmıştır.
- Test için 16 adam gün harcanmıştır.

Tablo 5.9: Proje 5 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	6	50	56	11	Ortalama Altında Başarı
2. Test	8	48	56	14	Ortalama Altında Başarı
3. Test	13	43	56	23	Ortalama Altında Başarı
4. Test	15	41	56	27	Ortalama Altında Başarı
5. Test	32	24	56	57	Ortalama Altında Başarı
6. Test	32	24	56	57	
7. Test	56	0	56	100	

Proje 5 test sonuçları Tablo-5.9'da verilmiştir. Test sürecinin sonunda 100% başarılı olarak canlıya alınmıştır. 11%'den başlayan test başarısı 7 kere düzeltmeden sonra 100%'e çıkmıştır.

11% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasının altında kaldığını göstermektedir. Bu proje ile ilgili bir diğer durum da aynı proje 3'te ve proje 4'te olduğu gibi yazılımın tam olarak yazılımcı testi yapılmadan teste gönderilmesidir. Sonrasında devam eden testlerde oran linnere yakın bir şekilde yükselmiştir.

Proje 5 'in önemli özellikleri şu şekildedir,

- Orta önem düzeyine sahip bir projedir.
- Proje büyüklük olarak küçük derecede değerlendirilmektedir.
- 1 adet analist, 1 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken C# dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmamıştır.
- Yazılımcı testi yapılmamıştır.
- 1 tarayıcıda web testleri gerçekleştirilmiştir.
- Test otomasyon yazılmıştır.
- 56 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 6 kere gidip geldikten sonra, canlıya alınmıştır.
- Test için 14 adam gün harcanmıştır.

Tablo 5.10: Proje 6 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	0	22	22	0	Ortalama Altında Başarı
2. Test	2	20	22	9	Ortalama Altında Başarı
3. Test	6	16	22	27	Ortalama Altında Başarı
4. Test	6	16	22	27	Ortalama Altında Başarı
5. Test	19	3	22	86	Ortalama Üstünde Başarı
6. Test	22	0	22	100	

Proje 6 test sonuçları Tablo-5.10'da verilmiştir. Test sürecinin sonunda 100% başarılı olarak canlıya alınmıştır. 0%'dan başlayan test başarısı 6 kere düzeltmeden sonra 100%'e çıkmıştır.

0% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasının altında kaldığını göstermektedir. İlk testte 0% başarı oranı, yazılımın çalışmadığının göstergesidir. Bu oran entegrasyon problemi veya test ortamının yeterli seviyede ayakta olmamasından kaynaklanabilir. Bu seviyede yaşanan bir düşük başarı durumunun yazılımcı testi ile alakası yoktur. Fakat 2. Testte alınan 9% ve 3. Testte alınan 27%'lik oran yazılımcı testine yeterince önemin verilmediğinin, veya yazılım tamamlanmadan teste gönderilmesinin sonucudur.

Proje 6 'nın önemli özellikleri şu şekildedir,

- Orta önem düzeyine sahip bir projedir.
- Proje büyüklük olarak küçük derecede değerlendirilmektedir.
- 1 adet analist, 1 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken C# dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmamıştır.
- Yazılımcı testi yapılmamıştır.
- 1 tarayıcıda web testleri gerçekleştirilmiştir.

- 30% oranında test otomasyon yazılmıştır.
- 22 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 6 kere gidip geldikten sonra, canlıya alınmıştır.
- Test için 12 adam gün harcanmıştır.

Tablo 5.11: Proje 7 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	17	49	66	26	Ortalama Üstünde Başarı
2. Test	40	34	74	54	Ortalama Üstünde Başarı
3. Test	53	21	74	72	Ortalama Üstünde Başarı
4. Test	65	9	74	88	Ortalama Üstünde Başarı
5. Test	65	9	74	88	Ortalama Üstünde Başarı
6. Test	74	0	74	100	

Proje 7 test sonuçları Tablo-5.11'de verilmiştir. Test sürecinin sonunda 100% başarılı olarak canlıya alınmıştır. 26%'dan başlayan test başarısı 6 kere düzeltmeden sonra 100%'e çıkmıştır.

26% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasının üstünde olduğunu göstermektedir. Yazılım başarı oranı doğrusal artışa yakın olarak yükselmiştir. Test içerisinde karşılaşılan hataların çok büyük veya testi engelleyen hatalar olmadığı sonucuna ulaşabiliriz. Tüm başarı oranları incelendiğinde her test için ortalama başarının üzerinde olduğunu görmekteyiz.

Proje 7 'nin önemli özellikleri şu şekildedir,

- Orta önem düzeyine sahip bir projedir.
- Proje büyüklük olarak orta derecede değerlendirilmektedir.
- 1 adet analist, 1 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.

- Yazılım geliştirilirken Java dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmamıştır.
- Yazılımcı testi yapılmamıştır.
- 3 farklı tarayıcıda web testleri gerçekleştirilmiştir.
- 30% oranında test otomasyon yazılmıştır.
- 74 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 5 kere gidip geldikten sonra, canlıya alınmıştır.
- Test için 10 adam gün harcanmıştır.

Tablo 5.12: Proje 8 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	39	17	56	70	Ortalama Üstünde Başarı
2. Test	43	15	58	74	Ortalama Üstünde Başarı
3. Test	72	15	87	83	Ortalama Üstünde Başarı
4. Test	72	15	87	83	Ortalama Üstünde Başarı
5. Test	84	3	87	97	Ortalama Üstünde Başarı

Proje 8 test sonuçları Tablo-5.12'de verilmiştir. Test sürecinin sonunda 97% başarılı olarak canlıya alınmıştır. 70%'dan başlayan test başarısı 5 kere düzeltmeden sonra 100%'e çıkmıştır.

70% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasının çok üstünde olduğunu göstermektedir. Yazılım başarı oranı doğrusal artışa yakın olarak yükselmiştir. Test içerisinde karşılaşılan hataların çok büyük veya testi engelleyen hatalar olmadığı sonucuna ulaşabiliriz. Tüm başarı oranları incelendiğinde her test için ortalama başarının üzerinde olduğunu görmekteyiz. Aslında 70% başarı ile başlayan bir testin daha kısa sürede tamamlanması beklenirken diğer projeler gibi 5 kere test edilmesinin sebebi incelenebilir.

Proje 8 'in önemli özellikleri şu şekildedir,

- Orta önem düzeyine sahip bir projedir.
- Proje büyüklük olarak orta derecede değerlendirilmektedir.
- 1 adet analist, 1 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Iterative geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken Java dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmıştır.
- Yazılımcı testi yapılmamıştır.
- 3 farklı tarayıcıda web testleri gerçekleştirilmiştir.
- Test otomasyon yazılmamıştır.
- 87 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 5 kere gidip geldikten sonra, canlıya
- alınmıştır.
- Test için 33 adam gün harcanmıştır.

Tablo 5.13: Proje 9 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	13	78	91	14	Ortalama Altında Başarı
2. Test	28	63	91	31	Ortalama Altında Başarı
3. Test	166	48	214	78	Ortalama Üstünde Başarı
4. Test	277	8	285	97	Ortalama Üstünde Başarı
5. Test	325	4	329	99	Ortalama Üstünde Başarı

Proje 9 test sonuçları Tablo-5.13'de verilmiştir. Test sürecinin sonunda 99% başarılı olarak canlıya alınmıştır. 14% 'dan başlayan test başarısı 5 kere düzeltmeden sonra 99% 'a çıkmıştır. 14% başarı ile testin başlaması, bu yazılımın, ekip başarı ortalamasının altında kaldığını göstermektedir. Fakat 3. Testten sonra ekip başarı oranı yakalanmıştır. 14% 'den 31% 'e, 31% 'den 78% 'e çıkan başarı oranı testi engelleyen bir durumun olduğunun

göstergesidir. Bu durum yine diğer projelerde olduğu gibi yazılımcı testinden veya yazılımın tam olarak tamamlanmadan teste gönderilmiş olmasından kaynaklanabilir. Proje 9, 3. Testte başarı ortalamasını yakalamıştır.

Proje 9 'un önemli özellikleri şu şekildedir.

- Orta önem düzeyine sahip bir projedir.
- Proje büyüklük olarak orta derecede değerlendirilmektedir.
- 1 adet analist, 2 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken Java dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmıştır.
- Yazılımcı testi yapılmıştır.
- 1 tarayıcıda web testleri gerçekleştirilmiştir.
- Test otomasyon yazılmamıştır.
- 329 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 9 kere gidip geldikten sonra, canlıya alınmıştır.
- Test için 25 adam gün harcanmıştır.

Tablo 5.14: Proje 10 test sonucu senaryo değerlendirmesi

	Başarılı	Başarısız	Toplam	Başarı %	Ortalama Değerlendirmesi
1. Test	0	109	109	0	Ortalama Altında Başarı
2. Test	1	110	111	1	Ortalama Altında Başarı
3. Test	1	109	110	1	Ortalama Altında Başarı
4. Test	2	109	111	2	Ortalama Altında Başarı
5. Test	10	103	113	9	Ortalama Altında Başarı
6. Test	17	121	138	12	
7. Test	22	125	147	15	
8. Test	48	106	154	31	
9. Test	51	105	156	33	
10. Test	50	98	148	34	
11. Test	57	91	148	39	
12. Test	52	96	148	35	
13. Test	57	92	149	38	
14. Test	56	84	140	40	
15. Test	57	83	140	41	
16. Test	42	98	140	30	
17. Test	43	97	140	31	
18. Test	42	98	140	30	
19. Test	63	83	146	43	
20. Test	53	89	142	37	
21. Test	55	87	142	39	
22. Test	65	79	144	45	
23. Test	72	73	145	50	

Proje 10 test sonuçları Tablo-5.14'te verilmiştir. En büyük proje olma özelliğini taşımaktadır. Bu nedenle de test edilen senaryo sayısı diğer projelere göre ciddi farklılık göstermektedir. Diğer projelerden farklı değerlendirilmelidir. Bu denli büyük projelerde başarı yüzdesinin başlangıçta düşük olması, analizin tamamlandıktan sonra test senaryolarının yazılmasından ve yazılımın parça parça teste gönderilmesinden kaynaklanmaktadır.

Bu projede göze çarpan bir diğer husus ise, testin 50% başarı seviyesinde olmasına rağmen canlıya alınmasıdır. Bu şekilde testi tamamlanmadan canlıya alınan projelerin hata maliyetleri ciddi şekilde yüksek olmaktadır. Tespit edilen fakat çözümlemeyen 73 adet hata son kullanıcı karşısında ciddi itibar ve maddi kayıplara neden olacaktır. 50% başarı ile canlıya alınan bir projede mevcut hataların ölümcül olmadığını varsayabiliriz. Göz ardı edilebilecek, küçük hata, kozmetik hata veya önerilerle canlıya alınmıştır. Yine çok büyük ihtimalle test edilemeden canlıya alınmış yazılımlar bu proje içerisinde mevcuttur.

Proje 10'un önemli özellikleri şu şekildedir,

- Yüksek önem düzeyine sahip bir projedir.
- Proje büyüklük olarak büyük derecede değerlendirilmektedir.
- 2 adet analist, 3 adet yazılım mühendisi, 1 adet test uzmanı projede çalışmıştır.
- Waterfall geliştirme metodu ile geliştirilmiştir.
- Yazılım geliştirilirken Java dili kullanılmıştır.
- Kod kalite kontrol aracı kullanılmıştır.
- Yazılımcı testi yapılmıştır.
- 1 tarayıcıda web testleri gerçekleştirilmiştir.
- Test otomasyon yazılmamıştır.
- 329 adet test durumu (senaryosu) yazılmıştır.
- Test yazılım mühendisi ve test uzmanı arasında 23 kere gidip geldikten sonra, canlıya alınmıştır.
- Test için 50 adam gün harcanmıştır.

5.3 Uygulama

Uygulamada Spyder geliştirme ortamı kullanılmıştır. Yazılım dili olarak Python kullanılmıştır. Gerekli olan yükleme işlemleri aşağıda belirtilmiştir. Yazılım için scikit-learn kütüphaneleri kullanılmıştır.

5.3.1 Ön İşleme

Verileri makine öğrenmesi uygulamalarında kullanılmak üzere makinenin anlayacağı dile çevirmek gerekmektedir. Öncelikle verinin sisteme yüklenmesi gerekmektedir. Yükleme sonucunda örnek verinin nasıl göründüğü Şekil-5.8'de gösterilmiştir.

dataset - DataFrame

Index	Numara	Proje Tarihi	yuklugu (Buyuk, K	nemi (Yuksek, Or	cibindeki yazilimc	ekibindeki analist	ekibindeki testci	Analistin yasi	alistin tecrubesi (y	proje tecrubesi v	harcanan sure (A	Yazilimcinin yasi	imc
0	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
1	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
2	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
3	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
4	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
5	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
6	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
7	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
8	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
9	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
10	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
11	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
12	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
13	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
14	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
15	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
16	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
17	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
18	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3
19	1	2016	Kucuk	Orta	1	1	1	33	5	Evet	15	30	3

Şekil 5.8: Örnek veri

Etiket kodlaması, veri yüklemesi tamamlandıktan sonra, etiket kodlama işlemi ile sayısal olmayan veriler, sayısal verilere çevrilmiştir. Verilerin sayısal verilere çevrilmesi ile birlikte bu değerler artık makine tarafından kullanılır hale gelmiştir.

Bir diğer ön işleme metodu ise, ölçekleme işlemidir. Verilerin birbiri arasında anlamsız fark yaratmaması açısından bu işlem uygulanmaktadır.

PCA analizi sonucu Tablo-5.15'de verilmiştir. Sonuca etki eden aslında on iki adet metrik mevcut. Diğer metrikler çok az sonucu etkilemektedir. Sadece bir adet metrikle 42% sonuç tahmin edilebilir. 42%, 21% ve 10% olan üç adet metrik değeri ile 73% başarı oranı elde

edebilir. On ikinci metrikten sonraki değerler sıfıra çok yakın olduğundan aslında sonuca etkisi de yok denecek kadar az olmaktadır.

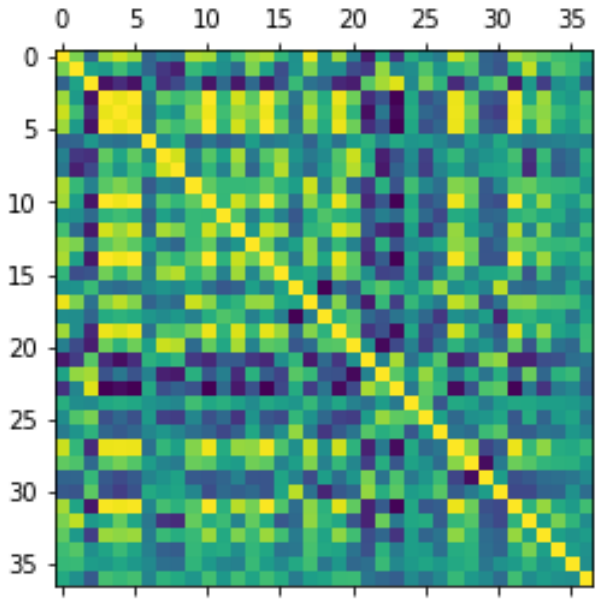
Tablo 5.15: PCA Analiz sonucu

Metrik No	PCA Analiz Sonucu	Metrik No	PCA Analiz Sonucu
0. Metric	0,423	10. Metric	0,011
1. Metric	0,218	11. Metric	0,010
2. Metric	0,106	12. Metric	0,004
3. Metric	0,054	13. Metric	2.15817e-29
4. Metric	0,050	14. Metric	1.98787e-29
5. Metric	0,034	15. Metric	1.31624e-29
6. Metric	0,026	16. Metric	3.22086e-30
7. Metric	0,023	17. Metric	2.60478e-30
8. Metric	0,019	18. Metric	1,15381e-30
9. Metric	0,016	.....	.....

PCA analiz algoritması içerisinde korelasyon analizi barındırmaktadır. Korelasyon analizi, iki değişken arasındaki doğrusal ilişkiyi veya bir değişkenin iki ya da daha çok değişken ile olan ilişkisini test etmek, varsa bu ilişkinin derecesini ölçmek için kullanılan istatistiksel bir yöntemdir [46].

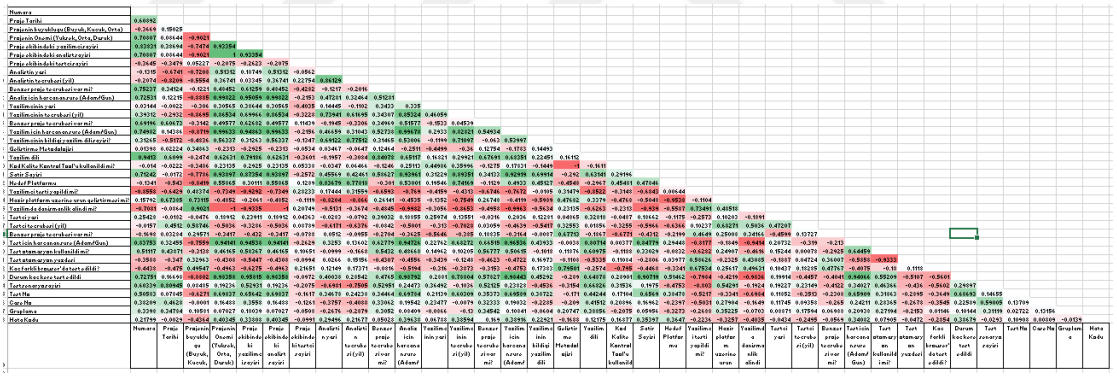
Verinin korelasyon analizi sonucu ortaya çıkan ısı haritası Şekil-5.9'da verilmiştir. Korelasyon haritasında köşegen çizgisindeki değerlerin çok yüksek olması, köşegenden tam simetrik olması, analizimizin doğru olduğunu göstermektedir.

Isı haritasında sarı değerler çok yüksek korelasyon değerlerini göstermektedir. Doğrulsal çizgide aynı metrikler değerlendirildiği için en yüksek değer olan 1 değeri çıkmaktadır karşımıza. Harita üzerinde yer alan renkleri takip ederek metrikler üzerinde yer alan korelasyonları yorumlayabiliriz.



Şekil 5.9: Korelasyon Matrisi Isı Haritası

Verinin korelasyon analizi sonucu ortaya çıkan korelasyon matrisi Şekil-5.10'da verilmiştir.



Şekil 5.10: Korelasyon Matrisi

Korelasyon katsayısı yorumu;

r<0.2 ise çok zayıf ilişki ya da korelasyon yok

0.2-0.4 arasında ise zayıf korelasyon

0.4-0.6 arasında ise orta şiddette korelasyon

0.6-0.8 arasında ise yüksek korelasyon

0.8> ise çok yüksek korelasyon olduğu yorumu yapılır.

Tablo 5.16: PCA Analiz sonucu

Korelasyon katsayısı	Metrik adı	Korelasyon yorumu
0.403452223	Projenin önemi	Orta şiddet
0.403452223	Proje ekibindeki analist sayısı	Orta şiddet
0.396375248	Analiz için harcanan süre (Adam/Gun)	zayıf
0.389155809	Yazılım için harcanan süre (Adam/Gun)	zayıf
0.38678971	Durum kaç kere test edildi	zayıf
0.385943617	Yazılımcının tecrübesi (yil)	zayıf
0.364697417	Hedef Platformu	zayıf
0.353969212	Satır Sayısı	zayıf
0.340819742	Test için harcanan sure (Adam/Gun)	zayıf
0.338079828	Proje ekibindeki yazılımcı sayısı	zayıf
0.294956348	Analistin yaşı	zayıf
0.229209504	Yazılımcının bildiği yazılım dili sayısı	zayıf
0.217987484	Numara (Proje sırası)	zayıf
0.216768418	Analistin tecrübesi (yil)	zayıf
0.168996275	Benzer proje tecrübesi var mı? (Yazılım)	korelasyon yok
0.168765578	Kod Kalite Kontrol Tool'u kullanıldı mi?	korelasyon yok
0.121745374	Yazılım dili	korelasyon yok
0.109081522	Test No	korelasyon yok
0.079052743	Test otomasyon kullanıldı mı?	korelasyon yok
0.067880908	Yazılımcının yaşı	korelasyon yok
0.050284971	Benzer proje tecrubesi var mı? (Analiz)	korelasyon yok
0.008090821	Case No	korelasyon yok
-0.002866436	Proje Tarihi	korelasyon yok
-0.013890421	Gruplama	korelasyon yok
-0.029297006	Test senaryo sayısı	korelasyon yok
-0.043415629	Testci yaşı	korelasyon yok
-0.047167624	Test otomasyon yüzdesi	korelasyon yok
-0.056927976	Benzer proje tecrubesi var mı?(Test)	korelasyon yok
-0.099131062	Proje ekibindeki testci sayısı	korelasyon yok
-0.168765578	Geliştirme Metodolojisi	korelasyon yok
-0.223550477	Yazılımcı testi yapıldı mı?	zayıf
-0.249505789	Testçi tecrübesi (yıl)	zayıf
-0.28539746	Kaç farklı tarayıcıda test edildi?	zayıf
-0.325687288	Hazır platform geliştirmesi mi?	zayıf
-0.403452223	Yazılımda danişmanlık alındı mı?	Orta şiddet
-0.436406025	Projenin büyüklüğü	Orta şiddet

Tablo-5.16'da verilen değerler incelenecek olursa, projenin önemi ve projede çalışan analist sayısının sonuç ile pozitif yönde bir korelasyonu mevcuttur. Bu koralasyon şiddeti orta şiddetlidir. Projenin büyüklüğü ve danışmanlık alınmış olması ise, negatif yönlü ve orta şiddette etki etmiştir.

Analiz için harcanan süre, Yazılım için harcanan süre, Durumun kaçıncı kez test edildiği, Yazılımcının tecrübesi, Hedef Platformu, Satır Sayısı, Test için harcanan süre, Proje ekibindeki yazılımci sayısı, Analistin yaşi, Yazılımcının bildiği yazılım dili sayısı, Analistin tecrübesi ise zayıf şiddetle pozitif yönde etki etmektedir. Yazılımci testi yapıldı mı, Hazır platform gelistirmesi mi, Kaç farkli tarayıcıda test edildi, Testçi tecrübesi ise zayıf şiddette negatif yönlü korelasyon içermektedir.

Tablo 5.17: PCA Analiz sonucu etkili metrikler

Korelasyon katsayısı	Metrik adı
-0.436406025	Projenin buyuklugu (Buyuk, Kucuk, Orta)
0.403452223	Projenin Onemi (Yuksek, Orta, Dusuk)
0.403452223	Proje ekibindeki analist sayısı
-0.403452223	Yazılımda danismanlik alindi mi?
0.396375248	Analiz için harcanan sure (Adam/Gun)
0.389155809	Yazılım için harcanan sure (Adam/Gun)
0.38678971	Durum kac kere test edildi
0.385943617	Yazılımcının tecrubesi (yil)
0.364697417	Hedef Platformu
0.353969212	Satir Sayısı
0.340819742	Test için harcanan sure (Adam/Gun)
0.338079828	Proje ekibindeki yazılımcı sayısı

PCA analizine göre sonuç için Tablo-5.17'de verilen on iki adet metriği kullanmamız yeterli olacaktır.

5.3.2 Algoritma Uygulamaları

Algoritmalar ve veri ile oluşturulan modellerin başarısının ölçülmesi için, karmaşıklık matrisi kullanılmıştır. Bütün modellerde toplam verinin %80'i ile model eğitilmiş, kalan %20'si ile test edilmiştir.

Karmaşıklık matrisi etiket tanımlamaları aşağıdaki gibidir;

Tablo 5.18: Etiket tanımlamaları

Definition	Label
Analist ile Görüşme	0
Başarılı	1
Büyük Hata	2
Katostrofik Hata	3
Kozmetik Hata	4
Kritik Hata	5
Küçük Hata	6
Öneri	7
Test Edilemedi	8

5.3.2.1 Karar Ağacı Uygulaması

Ön işleme aşamasından sonra karar ağacı algoritması ve veriler ile model oluşturulmuştur. Karar ağacı sınıflandırmasında, verinin 80%'i eğitim seti olarak kullanılmıştır. Kalan 20%'lik veri ile eğitilen model test edilmiştir.

Karar ağacı algoritmasında max depth değeri iki olarak seçilmiştir. Sırayla, artan bir şekilde farklı dept değerleri denenmiştir. İki değerinden daha büyük değerler verildiğinde başarı oranının değişmediği gözlemlenmiştir. İki kademeden sonra oluşacak olan değerler budanmıştır. Budama işlemini gerçekleştirilmesinin nedeni, gereksiz işlem yükünden kaçınmaktır.

```
[[ 26   0   0   0   0   0   0   0   0]
 [  0 685   0   0   0   0   0   0   0]
 [  0   0   0   0   0   1   0   0   0]
 [  0   0   0   0   0   7   0   0   0]
 [  0   0   0   0   0  70   0   0   0]
 [  0   0   0   0   0 113   0   0   0]
 [  0   0   0   0   0  53   0   0   0]
 [  0   0   0   0   0  13   0   0   0]
 [  0   0   0   0   0   0   0   0 368]]
0.8922155688622755
0.8922155688622756
```

Şekil 5.11: Hata matrisi sonuç-Karar Ağacı

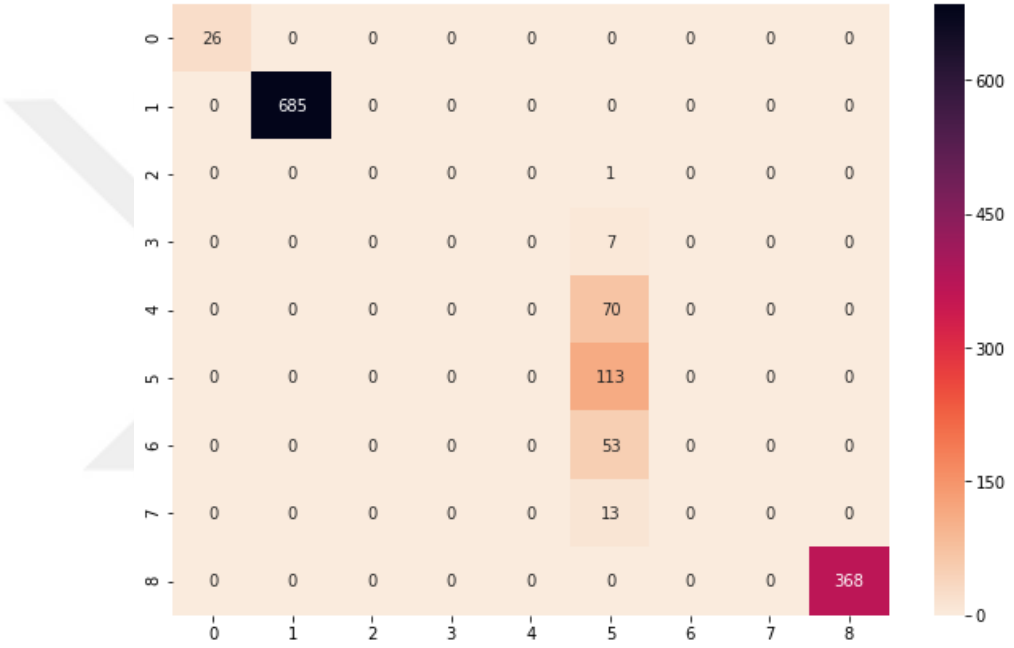
Hata matrisinde köşegenler üzerinde bulunan değerler doğru tahmin edilmiş değerlerdir. Köşegen üzerinde olmayan değerler ise, yanlış tahmin edilmiş değerlerdir. Harici bölgelerdeki sayısal değerler ise a olması gerekirken b şeklinde yapılan tahminleri ifade etmektedir.

Şekil-5.11'de bulunan hata matrisi sonucuna göre, 89% başarı ile hata tahminlemesi yapılmıştır. Bu tahminleme yapılırken, test aşamasında denenen değerlerin sonucu şu şekildedir.

- 26 adet sonuç Analist ile Görüşme olarak tahmin edilmiştir, doğru sonuç Analist ile Görüşmedir ve tahmin başarılıdır.
- 685 adet sonuç Başarılı olarak tahmin edilmiş, doğru sonuç Başarılıdır ve tahmin başarılıdır.
- 1 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Büyük Hatadır ve tahmin başarılı değildir.
- 7 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Katostrofik Hatadır ve tahmin başarılı değildir.
- 70 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Kozmetik hatadır ve tahmin başarılı değildir.
- 113 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Kritik Hatadır ve tahmin başarılıdır.
- 53 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Küçük Hatadır ve tahmin başarılı değildir.

- 13 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Öneridir ve tahmin başarılı değildir.
- 368 adet sonuç Test Edilemedi tahmin edilmiş, doğru sonuç Test Edilemedidir ve tahmin başarılıdır.

Test seti, toplam verinin 20% olarak seçildiğinden, 1336 adet tahminleme yapıldı, toplam veri sayısı 6676 adettir. 1336 sonuçtan 1192 adet sonuç başarılı tahmin edilmiştir. Bu sonuçlara göre başarı oranı 89%'dur.



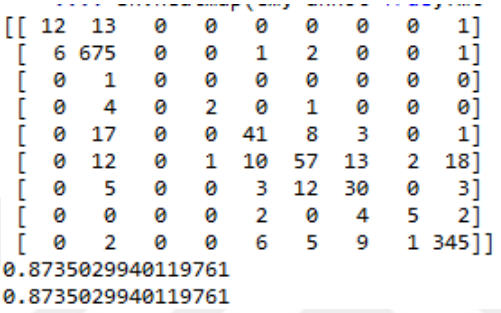
Şekil 5.12: Hata matrisi ısı haritası -Karar Ağacı

Şekil 5.12'de hata matrisinin ısı haritası verilmiştir. Isı haritasında, yukarıda belirtilen değerlerin dağılım oranları görülmekte

5.3.2.2 En Yakın Komşular Algoritması Uygulaması

Ön işleme aşamasından sonra en yakın komşular algoritması ve veriler ile model oluşturulmuştur. En yakın komşular algoritmasında da, verinin 80%'i eğitim seti olarak kullanılmıştır. Kalan 20%'lik veri ile eğitilen model test edilmiştir.

Knn algoritmasında n_neigbours değeri en yakındaki 5 komşu nokta ile çalışıldığını ifade etmektedir. Bunun sebebi, veri setine en uygun n_neighbours değerinin 5 olarak tespit edilmesidir. Artan bir şekilde komşu sayısı değiştirilmiş ve en uygun değer olarak 5 değeri belirlenmiştir. 5 değerinden daha yüksek değer verdiğimizde başarı oranı artmamaktadır.



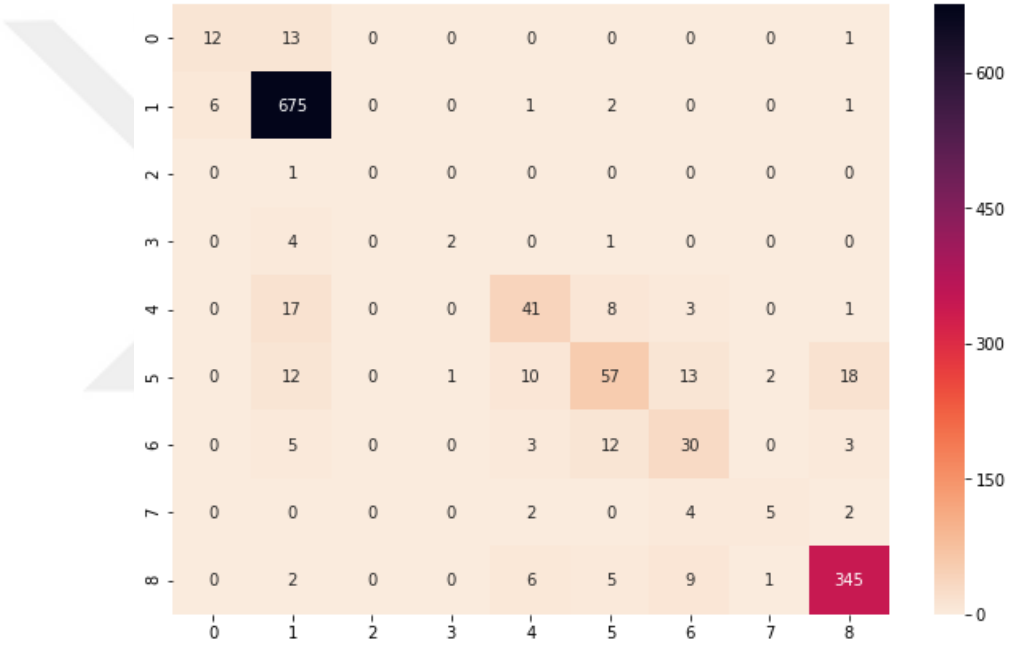
Şekil 5.13: Hata matrisi sonuç-En yakın komşular

Şekil-5.13'te bulunan hata matrisi sonucuna göre, 87% başarı ile hata tahminlemesi yapılmıştır. Bu tahminleme yapılırken, test aşamasında denenen değerlerin sonucu şu şekildedir.

- 12 adet sonuç Analist ile Görüşme tahmin edilmiş, doğru sonuç Analist ile Görüşmedir ve tahmin başarılıdır.
- 675 adet sonuç Başarılı tahmin edilmiş, doğru sonuç Başarılıdır ve tahmin başarılıdır.
- 2 adet sonuç Katostrofik Hatadır tahmin edilmiş, doğru sonuç Katostrofik Hatadır ve tahmin başarılıdır.
- 41 adet sonuç Kozmetik Hata tahmin edilmiş, doğru sonuç Kozmetik Hatadır ve tahmin başarılıdır.
- 57 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Kritik Hatadır ve tahmin başarılıdır.

- 30 adet sonuç Küçük Hata tahmin edilmiş, doğru sonuç Küçük Hatadır ve tahmin başarılıdır.
- 5 adet sonuç Öneri tahmin edilmiş, doğru sonuç Öneridir ve tahmin başarılıdır.
- 345 adet sonuç Test Edilemedi tahmin edilmiş, doğru sonuç Test Edilemedidir ve tahmin başarılıdır.

Test seti, toplam verinin 20% olarak seçildiğinden, 1336 adet tahminleme yapıldı, toplam veri sayısı 6676 adettir. 1336 sonuçtan 1167 adet sonuç başarılı tahmin edilmiştir. Bu sonuçlara göre başarı oranı 87%'dir.



Şekil 5.14: Hata matrisi algoritması ısı haritası -En yakın komşular

Şekil 5.14'te hata matrisinin ısı haritası verilmiştir. Isı haritasında, yukarıda belirtilen değerlerin dağılım oranları görülmektedir.

5.3.2.3 Rastgele Ormanlar Uygulaması

Rastgele ormanın çalışma presibi aslında, birden fazla karar ağacı oluşturup, bu ağaçların ortalamasını kullanmaktadır. Bu uygulama aşırı öğenmenin önüne geçmek içindir. Ön işleme aşamasından sonra rastgele ormanlar algoritması ve veriler ile model oluşturulmuştur. Rastgele Ormanlar algoritmasında da, verinin 80%'i eğitim seti olarak kullanılmıştır. Kalan 20%'lik veri ile eğitilen model test edilmiştir. Estimators değeri ana veri setinden kaç tane alt set oluşturup, oluşturulan alt setlerin rastgele ormanlar algoritmasına verileceğini belirler. Uygulamada estimators değeri 8 olarak belirlenmiştir. Artan bir şekilde komşu sayısı değiştirilmiş ve en uygun değer olarak 8 değeri belirlenmiştir. 8 değerinden daha yüksek değer verdiğimizde başarı oranı artmamaktadır.

```
[[ 10  16   0   0   0   0   0   0   0]
 [  0 685   0   0   0   0   0   0   0]
 [  0   1   0   0   0   0   0   0   0]
 [  0   3   0   0   1   1   0   0   2]
 [  0  20   0   0  33  14   0   0   3]
 [  0  18   0   0   2  89   1   0   3]
 [  0  16   0   0   1  18  16   0   2]
 [  0   3   0   0   0  10   0   0   0]
 [  0   7   0   0   8   0   0   0 353]]
0.8877245508982036
0.8877245508982037
```

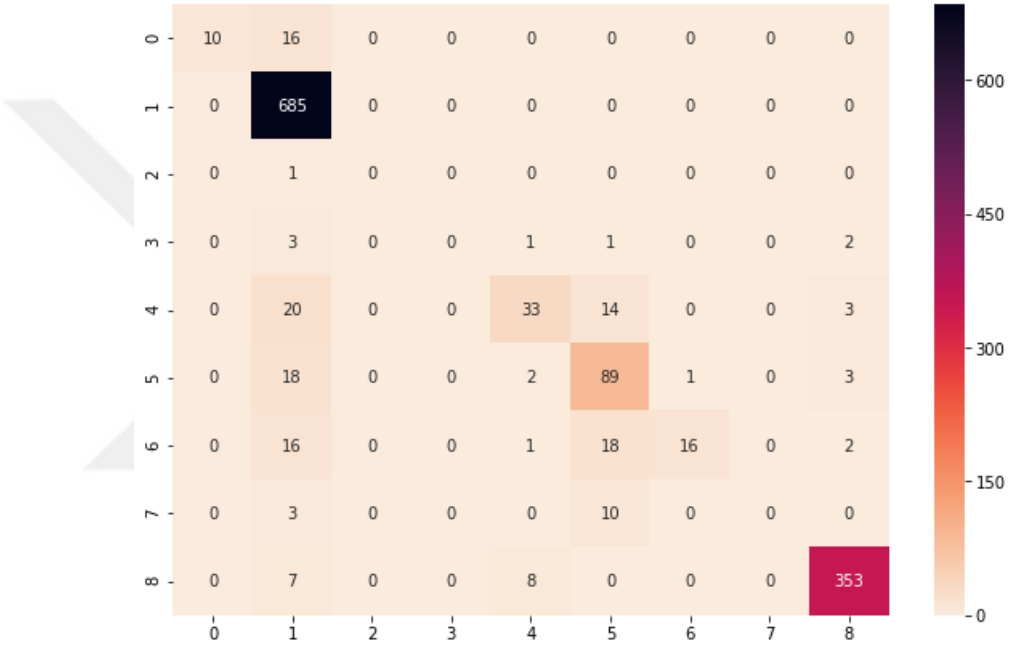
Şekil 5.15: Hata matrisi sonuç-Rastgele orman

Şekil-5.15'de bulunan hata matrisi sonucuna göre, 88% başarı ile hata tahminlemesi yapılmıştır. Bu tahminleme yapılırken, test aşamasında denenen değerlerin sonucu şu şekildedir.

- 10 adet sonuç Analist ile Görüşme tahmin edilmiş, doğru sonuç Analist ile görüşmedir ve tahmin başarılıdır.
- 685 adet sonuç Başarılı tahmin edilmiş, doğru sonuç Başarılıdır ve tahmin başarılıdır.
- 33 adet sonuç Kozmetik Hata tahmin edilmiş, doğru sonuç Kozmetik Hatadır ve tahmin başarılıdır.
- 89 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Kritik Hatadır ve tahmin başarılıdır.

- 16 adet sonuç Küçük Hata tahmin edilmiş, doğru sonuç Küçük Hatadır ve tahmin başarılıdır.
- 353 adet sonuç Test Edilemedi tahmin edilmiş, doğru sonuç Test Edilemedidir ve tahmin başarılıdır.

Test seti, toplam verinin 20% olarak seçildiğinden, 1336 adet tahminleme yapıldı, toplam veri sayısı 6676 adettir. 1336 sonuçtan 1186 adet sonuç başarılı tahmin edilmiştir. Bu sonuçlara göre başarı oranı 88%'dir.



Şekil 5.16: Hata matrisi ısı haritası -Rastgele Orman

Şekil 5.16'da hata matrisinin ısı haritası verilmiştir. Isı haritasında, yukarıda belirtilen değerlerin dağılım oranları görülmektedir.

5.3.2.4 Naive Bayes Uygulaması

Ön işleme aşamasından sonra rastgele ormanlar algoritması ve veriler ile model oluşturulmuştur. Naive Bayes algoritmasında da, verinin 80% 'i eğitim seti olarak kullanılmıştır. Kalan 20% 'lik veri ile eğitilen model test edilmiştir.

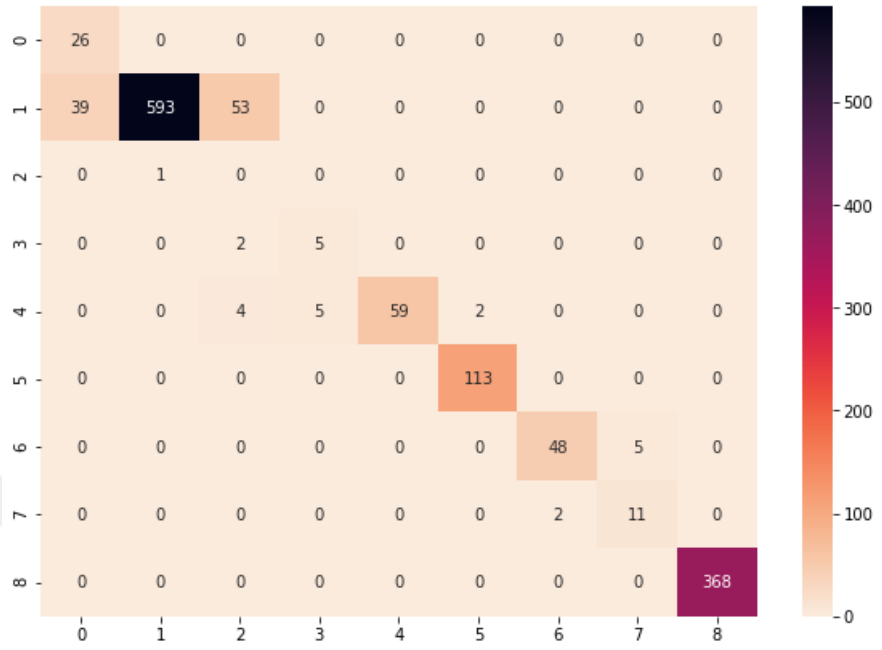
```
[ 39 593  53   0   0   0   0   0   0]
[  0   1   0   0   0   0   0   0   0]
[  0   0   2   5   0   0   0   0   0]
[  0   0   4   5  59   2   0   0   0]
[  0   0   0   0   0 113   0   0   0]
[  0   0   0   0   0   0  48   5   0]
[  0   0   0   0   0   0   2  11   0]
[  0   0   0   0   0   0   0   0 368]]
0.9154191616766467
0.9154191616766467
```

Şekil 5.17: Hata matrisi sonuç-Naive Bayes

Şekil-5.17'da bulunan hata matrisi sonucuna göre, 88% başarı ile hata tahminlemesi yapılmıştır. Bu tahminleme yapılırken, test aşamasında denenen değerlerin sonucu şu şekildedir.

- 26 adet sonuç Analist ile Görüşme tahmin edilmiş, doğru sonuç Analist ile Görüşmedir ve tahmin başarılıdır.
- 593 adet sonuç Başarılı tahmin edilmiş, doğru sonuç Başarılıdır ve tahmin başarılıdır.
- 5 adet sonuç Katostrofik Hata tahmin edilmiş, doğru sonuç Katostrofik Hatadır ve tahmin başarılıdır.
- 59 adet sonuç Kozmetik Hata tahmin edilmiş, doğru sonuç Kozmetik Hatadır ve tahmin başarılıdır.
- 113 adet sonuç Kritik Hata tahmin edilmiş, doğru sonuç Kritik Hatadır ve tahmin başarılıdır.
- 48 adet sonuç Küçük Hata tahmin edilmiş, doğru sonuç Küçük Hatadır ve tahmin başarılıdır.
- 11 adet sonuç Öneri tahmin edilmiş, doğru sonuç Öneridir ve tahmin başarılıdır.
- 353 adet sonuç Test Edilemedi tahmin edilmiş, doğru sonuç Test Edilemedidir ve tahmin başarılıdır.

Test seti, toplam verinin 20% olarak seçildiğinden, 1336 adet tahminleme yapıldı, toplam veri sayısı 6676 adettir. 1336 sonuçtan 1208 adet sonuç başarılı tahmin edilmiştir. Bu sonuçlara göre başarı oranı 91% 'dir.



Şekil 5.18: Hata matrisi ısı haritası-Naive Bayes

Şekil-37'de bulunan hata matrisi sonucuna göre, 91% başarı ile hata tahminlemesi yapılmıştır. Bu tahminleme yapılırken, test aşamasında denenen değerlerin sonucu şu şekildedir.

Bölüm 6

Sonuç

İlgili çalışmaların akademik olarak 1990 yılında yapılmaya başlanmasına rağmen, sektörde 2018 yılı itibari ile popüler olmaya başlandığı görülmüştür. Yapılan benzer çalışmalarda ise, genel olarak açık kaynak olarak NASA'nın KC1 ve KC2 verileri kullanılmaktadır. Bu verilerde, metrik olarak yazılım tabanlı metrikler kullanılmıştır. WQR'a göre makine öğrenmesi çalışmalarının sektörde kullanımının az olmasının sebebi kaliteli verinin olmamasıdır. Hem veri oluşturmak hem de farklı bir bakış açısı ile yaklaşmak adına hazır veri kullanmak yerine veri oluşturma için çalışma ve analizler yapılmıştır. Yapılan bu çalışma ile yazılım dışındaki metrikler de veri setine eklenmiştir. Yeni veri oluşturulmasında yazılım yaşam döngüsü uygulanan bir ekibin on adet projesi incelenmiştir. Toplam 6676 adet durum incelenmiştir. Çalışma esnasında toplamda kırk adet metrik değerlendirilmiştir. Veri üzerinde yapılan analiz ve çalışmalar ile, yazılım dışı metriklerin de sonuca etkili olduğu görülmüştür. Test sonucunu başarılı ve başarısız olarak değerlendirmek yerine hata seviyeleri de hesaba katılmıştır. Yuming Zhou and Hareton Leung[11]'ın yaptığı çalışmaya ek olarak test edilemeyen senaryolar da hesaba katılmış ve öngörülmeye çalışılmıştır.

PCA analizi sonucunda kırk adet belirlenen ilk metriklerden üç tanesi hiç etki etmediği için veri setinden çıkartılmıştır. Kalan otuz yedi metrik için yeniden PCA analizi yapıldığında on iki adet metriğin sonuç üzerinde 1%'den fazla değerde etkili olduğu, kalan metriklerin ise 1%"den daha az etkide sonuca etki ettiği görülmüştür. Bir adet metrikle 42%' sonuç tahmin edilebilirdir. 42%', 21%' ve 10%' olan üç adet metrik değeri ile 73%'

başarı oranı elde edebildiği görülmüştür. Projenin önemi ve projede çalışan analist sayısının sonuç ile pozitif yönde, orta şiddette korelasyonu, Projenin büyüklüğü ve danışmanlık alınmış olması ise, negatif yönlü ve orta şiddette korelasyonu olduğu görülmüştür. Analiz için harcanan süre, yazılım için harcanan süre, Durumun kaçıncı kez test edildiği, yazılımcının tecrübesi, Hedef Platformu, Satır Sayısı, Test için harcanan süre, Proje ekibindeki yazılımcı sayısı, Analistin yaşi, yazılımcının bildiği yazılım dili sayısı, Analistin tecrübesi ise zayıf şiddetle pozitif yönde etki ettiği görülmüştür. Yazılımcı testi yapıldı mı, Hazır platform geliştirmesi mi, Kaç farklı tarayıcıda test edildi, Testçi tecrübesi ise zayıf şiddette negatif yönlü korelasyonlu olduğu görülmüştür.

Modelleme aşamasında dört adet teknik kullanılmıştır. Algoritmalar günümüzde en çok kullanılan sınıflandırma algoritmalardan seçilmiştir. Karar Ağacı Uygulaması, En Yakın Komşular Algoritması, Rastgele Ormanlar Algoritması ve Naive Bayes Algoritması kullanılmıştır. Karar Ağacı Uygulaması 89%', En Yakın Komşular Algoritması 87%', Rastgele Ormanlar Algoritması 88%', Naive Bayes Algoritması 91%' başarı oranı ile çalışmıştır. Bu durumda Naive Bayes Algoritması bu veri seti için en uygun algoritmalar olarak görünmektedir.

Yazılım test aşamasında, hataya ne kadar kısa sürede ulaşılırsa, yazılım test maliyeti, dolayısı ile de yazılım maliyeti o kadar düşük olacaktır. Makine öğrenmesi kabiliyetleri ve eski proje verileri ile oluşturulan model, bize hangi test durumlarında hata alma olasılığımızın yüksek olduğunu söyleyecektir. Bu sayede hataya erken ulaşıp, hata çözüm süresini dolayısı ile de maliyeti düşürecektir.

Proje planlaması yapılma aşamasında, toplanan veriler ve farklı algoritmalar ile, analizin tamamlanma süresi, yazılımın tamamlanma süresi, testin tamamlanma süresi, dolayısı ile projenin tamamlanma süresi ön görülebilir. Proje planlamaları için çok büyük kolaylık sağlayacaktır. Aynı şekilde, projede çalışacak olan kişilerin bilgileri, çevresel metrikler ve yazılım metrikleri ile yapılacak olan çalışma ile, en uygun proje çalışanlarının seçilmesi sağlanacaktır. Hem zaman hem de maddi olarak kazanç sağlanacaktır.

Gelecekte yapılacak olan çalışmalarda, farklı projelerden de veriler eklenerek, veri seti güçlendirilecektir. Korelasyon katsayısı yüksek olan metrikler üzerine yoğunlaşıp, benzer etkide farklı metrik arama çalışmaları yapılacaktır. Metriklerin sadece sonuca olan ilişkileri değil, birbiri ile de olan ilişkileri incelenecektir. Yazılım test sonuç tahminleme için

en uygun algoritma çalışmaları devam edecek, gerekli ise yeni bir algoritma çalışması yapılacaktır.

Kaynakça

[1] Capgemini. World quality report 2017. Technical report, Capgemini, 2017.

[2] Capgemini. World quality report 2018. Technical report, Capgemini, 2018.

[3] Practitest. State of testing 2017. Technical report, Practitest, 2017.

[4] Practitest. State of testing 2018. Technical report, Practitest, 2018.

[5] L. Formhold. Machine learning the next big thing in qa and testing. 21 Ocak 2018 Erişildi: 21 Ocak 2019. URL `https://www.testbirds.com/blog/machine-learning/`.

[6] A. R. Chowdhury. Machine learning the next big thing in qa and testing. *https://www.lambdatest.com*, 25 Eylül 2018 Erişildi: 20 Ocak 2019. URL `https://www.lambdatest.com/blog/testing-trends-to-look-out-for-in-2019`.

[7] A. A. P. a. R. W. Selby. Empirically guided software development using metric-based classification trees. *IEEE*, pages 46–54, 1990.

[8] T. M. Khoshgoftaar ve N. Seliya. Tree-based software quality estimation models for fault prediction. *IEEE*, 2002.

[9] T. M. Khoshgoftaar ve N. Seliya. Comparative assessment of software quality classification techniques: An empirical case study. *IEEE*, page 229–257, 2004.

[10] E. J. W. a. R. M. B. Thomas J. Ostrand. Predicting the location and number of faults in large software systems. *IEEE*, 2005.

[11] Y. Zhou ve H. Leung. Empirical analysis of object-oriented design metrics for predicting high and low severity fault. *IEEE*, pages 771–789, 2006.

[12] Ç. Çatal. Software fault prediction: A literature review and current trends. *Elsevier*, 2011.

[13] M. Alnabhan ve F. Alsarayrah A. Hammouri, M. Hammad. Software bug prediction using machine learning approach. *International Journal of Advanced Computer Science and Applications*, pages 78–82, 2018.

[14] W. Bartłomiej ve R. Dabrowski. Applying machine learning to software fault. *e-Informatica Software Engineering Journal*, pages 199–216, 2018.

[15] X. Zhu H.Zhang B. Xu S. Ying Z. Li, X.Jing. Heterogeneous defect prediction with two-stage ensemble learning. *Automated Software Engineering, Springer Science Business Media*, 2019 04 Haziran 2019.

[16] Y. Guo Q. Song and M.Shepperd. A comprehensive investigation of the role of imbalanced learning for software defect prediction. *IEEE*, Nisan 2018.

[17] R. Malhotra. A systematic review of machine learning techniques for software fault prediction. *Elseiver Science Direct*, 2015.

[18] S. R. Schach. Testing principles and practice. *ACM Computing Survey*, 1996.

[19] L. Zhang. Software testing a case study of a small norwegian software team. 2012.

[20] T. Choudhury ve S. Sabitha J. Gaurl, A. Goya. A walk through of software testing techniques. Kasım 2016.

[21] M. Jeskanen ve J. Moilanen. Non-functional testing: security and performance testing. Kasım 2015.

[22] A. Pozo ve S. Vergilio A. R. Lens. Linking software testing results with a machine learning approach. 2013.

[23] Turkish Testing Board. Turkey software quality report 2012. Technical report, Turkish Testing Board, 2012.

[24] Turkish Testing Board. Turkey software quality report 2013. Technical report, Turkish Testing Board, 2013.

[25] Turkish Testing Board. Turkey software quality report 2014. Technical report, Turkish Testing Board, 2014.

[26] Turkish Testing Board. Turkey software quality report 2015. Technical report, Turkish Testing Board, 2015.

[27] Turkish Testing Board. Turkey software quality report 2016. Technical report, Turkish Testing Board, 2016.

[28] Turkish Testing Board. Turkey software quality report 2017. Technical report, Turkish Testing Board, 2017.

[29] Turkish Testing Board. Turkey software quality report 2018. Technical report, Turkish Testing Board, 2018.

[30] Testing Qualification Board. Worldwide software testing report. Technical report, Testing Qualification Board, 2015-2016.

[31] Testing Qualification Board. Worldwide software testing report. Technical report, Testing Qualification Board, 2017-2018.

[32] Capgemini. World quality report 2012. Technical report, Capgemini, 2012.

[33] Capgemini. World quality report 2014. Technical report, Capgemini, 2014.

[34] Capgemini. World quality report 2015. Technical report, Capgemini, 2015.

[35] Capgemini. World quality report 2016. Technical report, Capgemini, 2016.

[36] Y.l.Cheung. Tecniques in data mining. Haziran 2001.

[37] Y. Liu. Python machine learning byexample. *Packt Publishing Ltd*, 2019.

[38] S. Asaithambi. Machine learning the next big thing in qa and testing. *www.medium.com*, 4 Aralık 2017 Erişildi: 4 Nisan 2019. URL `https://medium.com/greyatom/why-how-and-when-scale-your-features-4b30ab09db5e`.

[39] J. Hogenboom. Princible component analysis and side-channel attacks. Ağustos 2010.

[40] S. Malatyali. Machine learning the next big thing in qa and testing. *www.bilgiinegi.com*, 08 Kasım 2018 Erişildi: 27 Mart 2019. URL `www.bilgiinegi.com/temel-bilesen-analizi-principal-component-analysis-nedir.htm`.

[41] M. Luckert ve M. Schaefer-Kehnert. Using machine learning methods for evaluating the quality of technical documents.

[42] W. D. Bennette. Instance selection for simplified decision trees through the generation and selection of instance candidate subsets. 2011.

[43] H. Dubey. Efficient and accurate knn based classification and regression. Mart 2013.

[44] B. E. Lowe. The random forest algorithm with application to multispectral image analysis. 2015.

[45] S. Hossain. Predicting subject area interest of a student using naive bayes. 2016.

[46] Akademikistatistik. Korelasyon analizi. *http://akademikistatistik.com*, 2019 Erişildi: 30 Mart 2019. URL `http://akademikistatistik.com/korelasyon-analizi/`.