# Yazılım Hata Kayıtlarının Makine Öğrenmesi Yöntemleriyle Kümelenerek, Hataya Sebep Olan Bileşenlerin Tespit Edilmesi

Sinan POLAT

Aselsan A.Ş. SST-GGZYTM, P.K. 1 06172, Yenimahalle, Ankara

spolat@aselsan.com.tr

**Özet.** ASELSAN Savunma Sistem Teknolojileri Sektör Başkanlığı Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü bünyesinde geliştirilen yazılımlar SST Kalite Yönetim Sisteminin tanımlamış olduğu yazılım geliştirme sürecine uygun olarak geliştirilmektedir. Bu bağlamda geliştirilen yazılımlar yapılan geliştirme testleri sonrası, yeterlilik testlerine girmektedirler. Yeterlilik testleri sırasında yazılımda görülen hatalar için yazılım test mühendisleri, hata takip aracında kayıt açarak, ilgili sürüm hatalarını yazılım mühendislerine bildirmektedirler. Kaydı açılan hata kayıtları bazen beklediğimizden daha fazla bilgi içerebilmektedir. Öyle ki, hataya sebep olan bileşen ya da bileşenler hata açıklamasından çıkarılabilmektedir. Bu doğrultuda hata kayıtları üzerinden hataya sebep olan bileşenlerin otomatik olarak tespit edilmesi, hem zaman tasarrufu sağlayacak, hem de sorunun çözümünü kolaylaştıracaktır. Ayrıca, bileşenlerdeki hata yoğunluğunun tespit edilmesi gibi yazılım mühendisliği metriklerinin hesaplanmasına da yardımcı olacaktır.

Bu çalışmada, hataya sebep olan bileşenlerin tespiti amacıyla, hata kayıtlarının denetimsiz öğrenme teknikleri ile kümelenmesi[1] anlatılacaktır. Denetimsiz öğrenme[2], etiketli veriler olmaksızın, girdi veri kümeleri üzerinden çıkarsamalar yapmak için kullanılan bir makine öğrenme[3] yöntemidir. Çalışmanın değerlendirilmesinde, ASELSAN Savunma Sistem Teknolojileri Sektör Başkanlığı Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü bünyesinde geliştirilen iki farklı atış kontrol yazılımı için girilen yaklaşık 1200 hata kaydı veri kümesi olarak kullanılmıştır. Kümeleme için ise K-means ve Sonek Ağacı[4] Clustering denetimsiz öğrenme algoritmaları kullanılarak performansları değerlendirilmiştir. Deneysel çalışmalar sonucu elde edilen ölçümler oldukça umut verici olmuştur. Öyle ki, hata analizi sonucu hesaplanan doğruluk metriğine göre, hataların %78'inin başarılı bir şekilde hataya sebep olan bileşene göre sınıflandırıldığı görülmüştür.

**Anahtar Kelimeler:** Hata Kayıtları, Makine Öğrenmesi, Denetimsiz Öğrenme, Kümeleme, Yazılım Geliştirme.

[1] Clustering
[2] Unsupervised Learning
[3] Machine Learning
[4] Suffix Tree

# Detection of Components Causing Errors by Clustering Software Error Records with Machine Learning Methods

Sinan POLAT

Aselsan A.Ş. SST-GGZYTM, P.K. 1 06172, Yenimahalle, Ankara

spolat@aselsan.com.tr

**Abstract.** The software developed in the Embedded and Real Time Software Design Department of ASELSAN Defense Systems Technologies Business Sector is developed in accordance with the software development process defined by the SST Quality Management System.The softwares developed in this context is included in the sufficiency tests after the development tests. For errors seen in the software during the sufficiency tests, the software test engineers register the bug tracking and notify the software engineers of the relevant version errors. Recorded error logs sometimes contain more information than we expect. So that the component or components that caused the fault can be extracted from the error description. In this respect, the automatic identification of the faulty components via error logs will save time and facilitate the solution of the problem. It will also help in the computation of software engineering metrics, such as determining the error density in components.

In this study, the clustering of error records with unsupervised learning techniques will be explained in order to identify the components causing the error. Unsupervised learning is a machine learning method that is used to make inferences over input data sets, without labeled data. In the evaluation of the work, approximately 1200 error logs were used as the data set which entered for the two different fire control software developed in the Real Time Software Design Department of ASELSAN Defense Systems Technologies Business Sector. For clustering, performance was evaluated using K-means and Suffix Tree Clustering unsupervised learning algorithms. According to the accuracy metric calculated by the error analysis, 78% of the errors were successfully classified according to the faulty component.

**Keywords:** Error Records, Machine Learning, Unsupervised Learning, Clustering, Software Development.

# 1 Giriş

Yazılım projesi, süreçlerin doğru ve profesyonel şekilde yönetilmesiyle hayata geçebilmektedir. Başarılı bir yazılım geliştirme süreci için her aşamanın ayrıntılı bir şekilde planlaması yapılmalı, süreçlerin en sağlıklı şekilde yönetilmesi gerekmektedir. Bu bağlamda, SST Grubu bünyesinde geliştirilen yazılımlar, müşterilerinin ihtiyaçlarına ve beklentilerine etkin ve verimli olarak cevap verebilmek için gerekli olan tasarım ve geliştirme süreçleri doğrultusunda geliştirilmektedir [1]. Temel olarak bir yazılım geliştirme döngüsü dört temel aşamada gerçekleşmektedir. Bu adımlar; gereksinimlerin belirlenmesi, yazılım tasarımının yapılması, kodlama ve test olarak sıralanabilir. Bu adımların her biri tanımlı olan yönergeler doğrultusunda eksiksiz olarak yapılmalıdır. Öyle ki, herhangi bir aşamada atlanan ya da görmezden gelinen bir husus sonraki aşamalarda probleme sebep olabilmektedir.

Yazılım yeterlilik testleri bu anlamda en az tasarım ve kodlama kadar önemli bir role sahiptir. Öyle ki, geliştirme süreci sırasında oluşan bir hatanın testler sırasında bulunması, hem yazılımı daha kaliteli hale getirecek, hem de son kullanıcıdan gelebilecek olumsuz bir geri bildirimin önüne geçilecektir. Bu amaçla testler sırasında bulunan hatalar için yazılım geliştiricilere bildirimler açılır. Bu bildirimler, hatanın hangi gereksinim test edilirken oluştuğu, ilgili yazılımın sürümü, kritiklik seviyesi ve hata açıklaması gibi bölümlerden oluşmaktadır.

Bildirilen hatalara ait hata açıklamaları bazen beklediğimizden daha fazla bilgi içerebilmektedir. Hata açıklamasında bulunan bazı anahtar sözcükler hataya neden olan yazılım bileşenin belirlenmesini sağlayabilmektedir. Bu bağlamda, bir yazılımın hataya sebep olan ya da hataya açık bileşenlerinin hata kayıtlarının açıklamalarından otomatik olarak bulunması, hem sorunu çözmek için zaman tasarrufu sağlayacak hem de yazılım bileşenleri üzerindeki hata yoğunluğu gibi yazılım mühendisliği metriklerinin bulunabilmesini sağlayacaktır.

Bu çalışmada denetimsiz öğrenme tekniklerini kullanarak hata kayıtlarını hataya neden olan yazılım bileşenlerine kümeleyen bir yazılım geliştirilmiştir. Denetimsiz öğrenme, etiketli veriler olmaksızın, girdi veri kümeleri üzerinden çıkarsamalar yapmak için kullanılan bir makine öğrenme yöntemidir. Denetimsiz öğrenme, kümeleme, aykırılık tespiti, sinir ağları ve gizli değişken modeller öğrenilmesi gibi yaklaşımları içermektedir. Fakat en yaygın denetimsiz öğrenme yöntemi, gizli modelleri bulmak veya verileri gruplamak için kullanılan kümeleme analizidir. Yapılan bu çalışmada olabilecek en iyi sonucu elde edebilmek için farklı kümeleme modelleri ve algoritmaları kullanılmıştır. Bu bağlamda Sonek Ağacı Kümeleme (STC) [2] ve K-Means Kümeleme [3] algoritmaları kullanılmıştır. Bu kümeleme algoritmalarının gerçekleştirilmesinde Carrot[2] Java API'lerinden yardım alınmıştır [4].

Bir diğer önemli nokta, kümeleme işleminden önce hata kayıtlarına bazı önişleme adımlarının uygulanması gerekliliğidir. Çünkü hata açıklamaları çeşitli yazım hataları, rakamlar, noktalama işaretleri ve bazı özel isimler içerebilmektedir. Kümeleme yapmadan önce bu tarz hatalı gruplamaya sebep olabilecek etmenlerin temizlenmesi başarımı doğrudan etkileyen bir hususdur.

Bu bildiride, hata nedeninin otomatik tespiti amacıyla geliştirilen yazılım anlatılacaktır. Bildirinin akışı 2. bölümde benzer çalışmalar, 3. bölümde projenin geliştirme süreci, 4. bölümde değerlendirme, 5. bölümde ise sonuçlar olacak şekilde ilerleyecektir.

## 2 Benzer Çalışmalar

Literatürde yapılan çalışmalar genellikle Dikey Hata Sınıflandırması (DHS) yöntemini esas almış olsa da çalışmamızla paralel olarak hataların sebeplerini bulmayı ve yazılım geliştirme sürecini iyileştirmeyi amaçlamıştır. DHS, hataları sınıflandırıp, analiz eden bir tekniktir.

1991 yılında Ram Chillarege [5] ve ekibi hataların anlamsal olarak sınıflandırılabileceğini ve yazılım güvenilirliği ile anlamsal olarak sınıflandırılmış hata türleri arasında ilişki kurulabileceğini belirtmişlerdir. 1992 yılında yayınladıkları çalışmada ise [6] Dikey Hata Sınıflandırması(DHS) yöntemini açıklamışlardır.

Kumaresh ve Baskaran [7], farklı projeler üzerinde hataları inceleyerek bu hataların türlerini bulmaya çalışmışlardır. Hata türlerini belirledikten sonra hata kök sebep analizi yaparak hataların sebeplerini bulmuşladır.

Falessi ve Cantone [8] ise yaptıkları deneyde hataları düzgün ve etkili bir şekilde sınıflandırma işleminin deneyimle alakalı olduğunu, öncelikle eğitim işleminin yapılmış olması gerektiğini belirtmişlerdir.

Söylemez, Tarhan ve Dikici'nin 2012 yılında yaptıkları çalışmada [9], hatalar analiz edilerek hata kayıtlarından geliştirme aşamasında anlamlı bilgiler çıkarmayı ve geliştiricilere süreç ilerlerken hızlı bir şekilde geribildirim vermeyi öneren Dikey Hata Sınıflandırması (DHS) tekniğinin özellikleri anlatılmıştır. Sonrasında ise DHS kullanarak yazılım geliştirme süreçlerini iyileştirmek için belirledikleri yöntem bağlamında, G222 bünyesinde geliştirilen Bütünleşik Sosyal Yardım Hizmetleri Projesi hataları analiz edilerek DHS tekniğinin uygulanabilirliği araştırılmış, oldukça başarılı sonuçlar elde edilmiştir.

## 3 Yapılan Çalışma

Bu bölümde çalışma kapsamında kullanılan veri setine ve temel proje aşamalarına değinilecek, alt bölümlerde ise bu aşamalar detaylandırılacaktır. Gerçekleştirilen proje genel anlamda ön işleme, analiz ve değerlendirme olmak üzere üç temel aşamadan oluşmaktadır.

- **Ön işleme**
  - Cümlelerin kelimelere parçalanması
  - Önemsiz kelimelerin (ing. Stop words) ve noktalama işaretlerinin silinmesi
  - Büyük-küçük harf uyumunun sağlanması ve kelime köklerinin bulunması
- **Analiz**
  - Kümeleme
  - Belirlenen kümelere etiket verilmesi ve bileşenlere atanması
- **Değerlendirme**

Çalışmada veri seti olarak ASELSAN Savunma Sistem Teknolojileri Sektör Başkanlığı Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü bünyesinde geliştirilen iki farklı atış kontrol yazılımı için girilen yaklaşık 1200 hata kaydı kullanılmıştır. Hata kayıtları, hata kaydı atama aracı üzerinden girilmektedir. Hata kayıtları bu araç üzerinden alınarak MS Office Excel ortamına aktarılmıştır. Alınan bu hata kayıtları, yazılım geliştiriciler tarafından kategorize edilmiş, hataların yaklaşık 10 farklı yazılım bileşeninden kaynaklandığı tespit edilmiştir.

**Tablo 1.** Örnek hata kaydı

| | |
|---|---|
| **ID** | SIVT00009071 |
| **Başlık** | Açılışta sürekli TRM02 hatası alınması |
| **Açıklama** | Yazılım her açıldığında TRM02 hatası alıyor, fakat aslında termal kamera ile iletişim var. |

### 3.1 Ön işleme

Hata kayıtları yazım hataları, rakamlar, noktalama işaretleri ve özel isimler içerebilmektedir. Analiz aşamasını doğrudan etkileyecek bu etmenleri ortadan kaldırmak için öncelikle kayıtların temizlenmesi gereklidir. Bu sebeple, öncelikle hata açıklamaları kelimelere parçalanarak, ön işleme kapsamında uygulanması planlanan değişiklikler uygulanmıştır. Bu değişiklikler aşağıda verilmiştir:

- Noktalama işaretleri, köşeli parantezler ve sayılar kaldırılır.
- Kesme işaretleri kaldırılır.
- Tireli sözcükler iki bölüme ayrılır.
- Harf içermeyen diğer parçacıklar silinir.(semboller)
- Kısaltmalar tek bir parça olarak ele alınır.
- Önemsiz kelimeler kaldırılır. Çeşitli bağlaç, zamir vs. içeren 231 kelimelik bir liste bu amaçla kullanılmaktadır [10].

Hata kayıtları temizlendikten sonra büyük-küçük harf uyumu ve kök bulma işlemleri gerçekleştirilir. Öncelikle tüm sözcükler küçük harfle gösterilir. Kelimeleri harf büyüklüğüne duyarsız hale getirdikten sonra tüm sözcüklerin "Zemberek" Türkçe için kök bulma aracı yardımıyla kökleri bulunur [11]. Türkçe zengin bir morfolojik yapıya sahiptir. İngilizce dört ya da beşten fazla ek almaz iken Türkçe dokuz ya da on ek alabilecek bir yapıya sahiptir. Bu nedenle genellikle Türkçe ve Türkçe gibi sondan eklemeli dillerde bir metni analiz etmek zor olabilmektedir. İki tür son ek vardır; çekim ekleri ve yapım ekleri. Çekim eklerini atmak basittir. Ancak, bir sözcük birden fazla yapım eki almış olabilir. Bu kelimelerin doğru kökünün bulunabilmesi de sorun teşkil edebilmektedir. Örneğin, *"işçilik"* sözcüğü iki yapım eki almıştır. Böyle durumlarda seçilecek kök tüm yapım eklerinin atılmasıyla bulunacak kök şeklinde bu çalışmada kullanılmıştır. Bu durumda *"işçilik"* sözcüğü için bulunacak kök *"iş"* olacaktır.

### 3.2 Analiz

En yaygın denetimsiz öğrenme yöntemi, gizli modelleri bulmak veya verileri gruplamak için kullanılan küme analizidir. Bu nedenle, küme verileri için gizli kalıplar bulmaya çalışan bir dizi kümeleme yaklaşımı geliştirilmiştir. Bu şekilde, farklı araştırmacılar farklı küme modelleri kullanır ve bu küme modellerinin her biri için yine farklı algoritmalar verilebilir. Tipik küme modelleri, hiyerarşik kümeleme, k-means ve k-medoid kümeleme, Gauss karma modelleri, en yakın komşular, gizli Markov modelleri vb. olarak verilebilir.

Objektif olarak değerlendirildiğinde tüm veri türleri üzerinde "doğru" çalışabilen bir kümeleme algoritması yoktur. Belli bir problem için en uygun kümeleme algoritması, bir küme modelini diğerine tercih etmek için matematiksel bir sebep olmadıkça deneysel olarak seçilmelidir. Bu nedenle, belli bir tür model için tasarlanmış bir algoritma, genellikle çok farklı türde bir model içeren bir veri kümesinde başarısız olur. Bu bağlamda bu çalışma kapsamında da en iyi sonucu elde edebilmek amacıyla Sonek Ağacı Kümeleme (STC) ve K-Means Kümeleme algoritmaları kullanılmıştır.

**Sonek Ağacı Kümeleme Algoritması (STC)**

Sonek Ağacı Kümeleme (STC), lineer zamanlı çalışabilen bir kümeleme algoritmasıdır. Temel olarak metinler arasında ortak olarak geçen cümle parçalarının belirlenmesini amaçlamaktadır. Bu bağlamda bir cümle parçacığı bir veya daha fazla kelimenin sıralı bir dizilimini ifade etmektedir. Algoritmaya göre temel kümeler, ortak bir ifade paylaşan metinler kümesi olarak tanımlanabilir. Temel kümelerin tanımlanması, metin koleksiyonumuz için ters indeks yapısının oluşturulması olarak görülebilir. Sonek ağacı veri yapısı yardımıyla bu temel kümeler kolaylıkla belirlenebilmektedir. Bu yapı, koleksiyonun boyutuyla doğrusal olarak ilişkilidir ve metinler okunurken ilgili veri yapısı da eş zamanlı olarak oluşturulabilmektedir.
*M* uzunluğundaki bir *S* dizgesi için *T* Sonek ağacının sahip olduğu özellikler aşağıda verilmiştir:

- Köklü bir ağaçtır ve yönlüdür
- 1 ile *M* arasında etiketlenmiş *M* yaprağı vardır
- Ağaçtaki her bir dal *S* dizgesinin bir alt dizgesini oluşturur
- Kökten, *i*. yaprağa kadar etiketlenmiş bir yol üzerindeki kenarlar birleştirilebilir
- Kök olmayan her ara düğümün en az 2 yaprağı vardır
- Bir düğümden çıkan kenarlar farklı karakterler ile başlar

STC; metinlerin temizlenmesi, temel kümelerin tanımlaması ve bu temel kümeleri kümelere birleştirme olmak üzere üç temel aşamada işlemektedir. Temel kümelerin birleştirilmesi ile oluşan yeni kümeler, metinler arasındaki ortak cümle parçacıklarını da bizlere vermiş olacaktır.

Bizim çalışmamız açısından düşünüldüğünde ise, elde edilen bu ortak cümle parçacıkları ilgili hata kaydı açıklamalarında ortak geçen bölümler olacaktır. Bulunan ortak

cümle parçacıkları bileşenler ile eşleştirildiğinde ise o kümedeki hataların hangi bileşenden kaynaklı olduğu da tespit edilmiş olacaktır.

**K-means Kümeleme Algoritması**

K-means, bilinen kümeleme problemini çözen en basit denetimsiz öğrenme algoritmalarından biridir. K-means algoritmasının genel mantığı n adet veri nesnesinden oluşan bir veri kümesini, giriş parametresi olarak verilen k adet kümeye bölümlemektir. Amaç, gerçekleştirilen bölümleme işlemi sonunda elde edilen kümelerin, küme içi benzerliklerinin maksimum ve kümeler arası benzerliklerinin minimum olmasını sağlamaktır. Kullanılan matematiksel yöntem, her sınıf için merkez olarak belirlenen noktaya olan uzaklığa (aynı zamanda bu hata miktarıdır) göre yeni kümelerin yerleştirilmesidir.

Algoritma temel olarak 4 aşamadan oluşur:

1. Küme merkezlerinin belirlenmesi
2. Merkez dışındaki örneklerin mesafelerine göre sınıflandırılması
3. Yapılan sınıflandırmaya göre yeni merkezlerin belirlenmesi (veya eski merkezlerin yeni merkeze kaydırılması)
4. Kararlı hale (İng. stable state) gelinene kadar 2. ve 3. adımların tekrarlanması

K-Means algoritması rastgele seçilen K (küme sayısı) adet merkez noktayla başlar. Veri kümesindeki her nokta kendisine en yakın merkez noktanın kümesine atanır. Küme merkezinin değeri kendine ait noktaların ortalaması alınarak hesaplanır. Bu işlem merkezlerin değerleri değişmeyinceye kadar devam eder.

Bizim çalışmamız açısından düşünüldüğünde ise buradaki n değeri hata kaydı açıklamalarının sayısını ifade ederken, k değeri ise yazılımda bulunan bileşen sayısını ifade edecektir. Kümelemede kullanılan mesafe ölçütü ise aynı bileşen ile ilgili kelimelerin sayısı olarak düşünülecektir. Bu doğrultuda, K-means algoritması yardımıyla bulunan kümeler aynı bileşenden kaynaklı oluşmuş olan hata kümelerini ifade etmiş olacaktır. Kümelerin merkezlerine en yakın hata kayıtları ise ilgili kümenin hangi bileşen ile eşleştiği hakkında bilgi vermektedir.

# 4 Değerlendirme

Yapılan çalışmanın değerlendirilmesi aşamasında daha önceden de belirtildiği gibi Aselsan SST GGZYTM bünyesinde geliştirilmiş olan iki farklı atış kontrol yazılımına ait hata kayıtları kullanılmıştır. Veri setinde yaklaşık 1200 hata kaydı ve 10 farklı bileşen kategorisi bulunmaktadır. Bu hata kayıtları yazılım geliştiricileri tarafından hataya neden olan bileşenlere göre işaretlenmiştir. Bu işaretlemeler sadece başarımın ölçülmesi aşamasında kullanılmış olup, kümeleme algoritmalarına girdi olarak verilmemiştir. Excel formatında olan hata kayıtlarının, değerlendirme aşamasında kullanılabilmesi için “ID”, “Konu” ve “Sorun_Istek_Tanimi” alanları seçilerek CSV formatına dönüştürülmüştür.

| id | Konu | Sorun_Istek_Tanimi | Aciklama | Bilesen_1 | Bilesen_2 | Bilesen_3 | Bilesen_4 | Bilesen_5 | Bilesen_6 |
|---|---|---|---|---|---|---|---|---|---|
| SIVT00002011 | Servo Controller Hatası | Servo ile Sym arasındaki güncelleme sıklığı nedeni ile Servo Ctrl hata veriyor. | Optimizasyon ve iyileştirme yapıldı. | | | x | | | |
| SIVT00002100 | Atışa yasak bölgede ateş ediliyor - KMTD Ref: 2.2.2.1 | Menüler ile tanımlanan ateşe yasak bölgeye taret yönlenince ateş edilmemesi gerekir. Fakat yapılan testlerde sistemin ateşe yasak bölgede ateş ettiği görülmüştür. | Yapıldı | | | x | | | |
| SIVT00002101 | Taretin yan ve yükseliş eksenlerindeki sıfır noktasının ayarlanması - KMTD: 2.2.2.2 | Taretin yan ve yükseliş eksenlerindeki sıfır noktasının ayarlanması ve korunması gerekir. | Çözüldü | | | x | | | |
| SIVT00002102 | Taretin yan ve yükseliş limitlerinin uygulanması KMTD: 2.2.2.2 | Taretin yan ve yükseliş limitlerinin belirlenemediği görülmüştür. | Limitler Uygulandı | | | x | | | |
| SIVT00003434 | Vhib Takip Penceresinin kaydırılması | Vhib Takip Penceresinin article ile aynı noktaya kaydırılmalı | Boresigh'ın değiştirildiği yerde (VHIBController) takip penceresinin pozisyonu değiştirildi. | | | | | | x |
| SIVT00003437 | Can arayüzünden çıkan bilinmeyen mesaj | Can arayüzünden çıkan bilinmeyen mesajın sebebinin bulunması ve düzeltilmesi | Motora gelen mesajlar static event yapıldıktan sonra bilinmeyen mesaj görülmedi. | | | x | | | |
| SIVT00003498 | giderme menüsünden İPTAL ile çıkılsa bile değerlerin kaydedilmesi | Otomatik kayma giderme menüsünden İPTAL ile çıkılsa bile değerler kaydediliyor. | Motor ile Menüler arasındaki çift yönlü bağımlılık koparıldı. | | | x | | | |
| SIVT00003565 | Bakım modu menüsünde iken TRM00 hatası çıkıyor | Bakım modu menüsünde iken TRM00 hatası çıkıyor | Çözüldü (Boot modunda (boot menüsünde) gelen invalid video sinyalleri değerlendirmeye alınmıyor.) | | | | x | | |

**Şekil 1.** Excel Formatında Hata Kayıtları

Bilgisayarda, virgülle ayrılmış değerler dosyası (CSV), düz metin halinde sekmeli verileri (sayıları ve metni) depolamaktadır. Şekil 1'de Excel formatında örnek hata kayıtları görülürken, Şekil 2'de CSV formatına dönüştürülmüş hata kayıtları verilmiştir.

```
1  SIVT00002011;Servo Controller Hatası;Bakım Modunda Video Trackin yapılmıyor;Servo ile Sym arasındaki güncelleme
   sıklığı nedeni ile Servo Ctrl hata veriyor.
2  SIVT00002100;Atışa yasak bölgede ateş ediliyor - KMTD Ref: 2.2.2.1;Menüler ile tanımlanan ateşe yasak bölgeye taret
   yönlenince ateş edilmemesi gerekir. Fakat yapılan testlerde sistemin ateşe yasak bölgede ateş ettiği görülmüştür.
3  SIVT00002101;Taretin yan ve yükseliş eksenlerindeki sıfır noktasının ayarlanması - KMTD: 2.2.2.2;Taretin yan ve
   yükseliş eksenlerindeki sıfır noktasının ayarlanması ve korunması gerekir.
4  SIVT00002102;Taretin yan ve yükseliş limitlerinin uygulanması KMTD: 2.2.2.2;Taretin yan ve yükseliş limitlerinin
   belirlenemediği görülmüştür.
```

**Şekil 2**. CSV formatına dönüştürülmüş hata kayıtları

Veri seti hazırlandıktan sonra hata kayıtları STC ve K-means kümeleme algoritmaları yardımıyla kümelenmiştir. Başarımın ölçülebilmesi için her iki algoritma için de doğruluk metriği, daha önceden geliştiriciler tarafından işaretlenmiş hata kayıtları kullanılarak hesaplanmıştır. Doğruluk, analiz sonucunun gerçek değere ne kadar yakın olduğunun bir ölçütüdür. Ölçümlerin aritmetik ortalamasının gerçek değere yakınlığı olarak tanımlanır ve hata olarak ifade edilir. Doğruluk formülasyonu Formül 1'de verilmiştir.

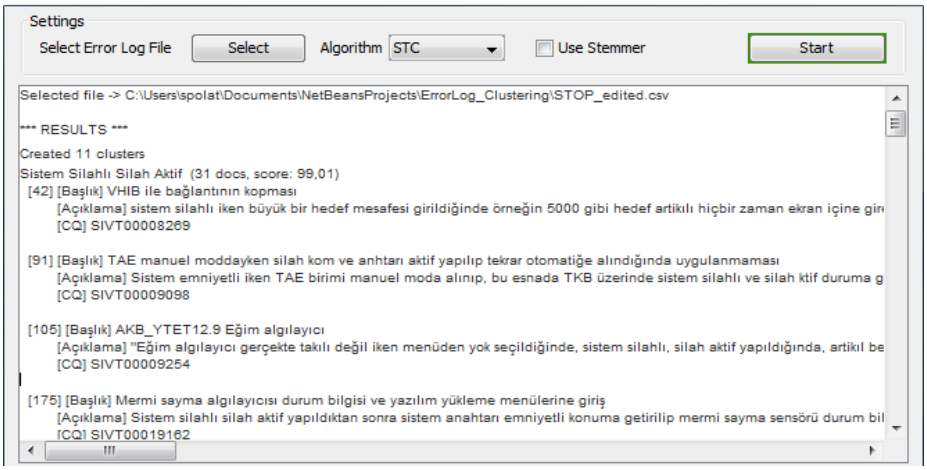
$$Doğruluk = \frac{Bileşene\ Göre\ Doğru\ Sınıflanmış\ Hata\ Kaydı\ Sayısı}{Toplam\ Hata\ Kaydı\ Sayısı} \quad (1)$$

Değerlendirme sonuçları Tablo 2'de verilmiştir. Sonuçlara göre STC algoritması 11 küme oluştururken 1200 hata kaydından 95 tanesini kümeleyememiştir. K-means algoritması ise 12 farklı küme oluşturmuştur. Bulunan kümelerdeki hata kayıtları kategorilerine göre işaretlenip doğruluk metriği hesaplandığında elde edilen sonuçlara göre STC algoritması %78 başarı gösterirken, K-means algoritması %61 başarı gösterebilmiştir. (Tablo 2) Daha önceden de belirtildiği üzere tüm veri kümeleri üzerinde başarılı olarak çalışabilen bir algoritma varlığından söz etmek zordur. İlgili veri seti üzerinde hangi algoritmanın daha başarılı olacağı ancak deneysel olarak bulunabilmektedir.

**Tablo 2**. Doğruluk Ölçümleri

| Algoritma | Doğruluk |
|---|---|
| STC | 78 % |
| K-Means | 61 % |

Şekil 3'te değerlendirme aşamasında STC algoritması için yazılımdan alınan bir ekran görüntüsü verilmiştir. Sonuçlar incelendiğinde 11 adet küme bulunduğu görülebilir. Bulunmuş olan ilk kümenin etiketi "Sistem Silahlı Silah Aktif" olarak belirlenmiş, kümenin silah ile ilgili yazılım bileşenleri ile ilgili hataları içerdiği sonucu çıkarılmıştır.




**Şekil 3.** STC Algoritması için kümeleme (Yazılımdan ekran görüntüsü)

## 5 Sonuç

Yaptığımız bu çalışmada, hataya sebep olan bileşenlerin tespiti amacıyla, hata kayıtlarının denetimsiz öğrenme teknikleri ile kümelenmesi anlatılmıştır. Çalışmanın değerlendirilmesinde, ASELSAN Savunma Sistem Teknolojileri Sektör Başkanlığı Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü bünyesinde geliştirilen iki farklı atış kontrol yazılımı için girilen yaklaşık 1200 hata kaydı veri kümesi olarak kullanılmıştır. Kümeleme için ise K-means ve Sonek Ağacı Clustering(STC) denetimsiz öğrenme algoritmaları kullanılarak performansları değerlendirilmiş, deneysel çalışmalar sonucu elde edilen sonuçlar oldukça umut verici olmuştur. Öyle ki, hata analizi sonucu hesaplanan doğruluk metriğine göre, hataların %78'inin başarılı bir şekilde hataya sebep olan bileşene göre sınıflandırıldığı görülmüştür. Böylece yazılım geliştirme sürecinde oldukça önemli bir yere sahip olan yeterlilik testlerinde bulunan hata kayıtları, daha hızlı bir şekilde çözülmesi hedeflenmektedir. Ayrıca, yazılım geliştiricilere geliştirdikleri yazılım bileşenlerine ait hata yoğunluğu ile ilgili geri bildirim de verilebilecektir. Çalışmanın ilerleyen aşamalarında, bulunan küme etiketleri ile bileşenler arası ilişkilerin, kullanıcı geri bildirimleri ile öğrenilmesine yönelik yapılabilecek bir eklenti, bulunan kümelerin bileşenler ile doğrudan eşleşebilmelerini sağlayacaktır.

## Kaynaklar


1. Kahraman, E., İpek, T., İyidir, B., Bazlamaçcı, C.F., Bilgen, S.: Bileşen Tabanlı Yazılım Ürün Hattı Geliştirmeye Yönelik Alan Mühendisliği Çalışmaları. In: UYMS'09, pp. 283–287 (2009)
2. Zamir, O., Oren E.: Web document clustering: A feasibility demonstration. In: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, (1998)
3. Hartigan, John A., Manchek A.: Algorithm AS 136: A k-means clustering algorithm. In: Journal of the Royal Statistical Society. Series C (Applied Statistics) 28.1, pp.100-108 (1979)
4. Carrot Framework, https://project.carrot2.org/, erişim tarihi 2017/06/14
5. Chillarege, R., Kao, W.-L, Condit, R.G.: Defect type and its impact on the growth curve, Proc. 13th Int. Conf. Software Engineering, (1991)
6. Chillarege, R., Bhandari, I.S., Chaar, J.K., Halliday, M.J., Moebus, D.S., Ray, B.K. and Wong, M.-Y.: Orthogonal defect classification–a concept for in-process measurements, IEEE Trans. Softw. Eng., vol. 18, pp. 943–956, November (1992)
7. Kumaresh, S. and Baskaran, R.: Article:defect analysis and prevention for software process quality improvement, International Journal of Computer Applications, vol.8, no.7, pp.42–47, October (2010)
8. Falessi, D. and Cantone, G.: Exploring feasibility of software defects orthogonal classification, in International Conference on Software and Data Tecnologies (ICSOFT2006), (2006)
9. Söylemez, M., Tarhan, A., Dikici, A.: Dikey Hata Sınıflandırması(DHS) ile Yazılım Hatalarının Kök Sebeplerinin İncelenmesi, in UYMS'12, pp. 77-84 (2012)
10. Stop Words List, http://countwordsfree.com/stopwords/turkish, erişim tarihi 2017/06/14.
11. Zemberek NLP, https://github.com/ahmetaa/zemberek-nlp, erişim tarihi 2017/06/14.