ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

MAKİNE ÖĞRENMESİ TABANLI TOPLULUK YÖNTEMLERİ İLE YAZILIM KALİTESİ TAHMİNİ

Abidin Ayberk CERAN

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2022

ÖZET

Yüksek Lisans Tezi

MAKİNE ÖĞRENMESİ TABANLI TOPLULUK YÖNTEMLERİ İLE YAZILIM KALİTESİ TAHMİNİ

Abidin Ayberk CERAN

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Ömer Özgür TANRIÖVER

Eş Danışman: Dr. Öğr. Üyesi Yılmaz AR

Yazılım kalite tahmini, yazılım projelerinin çeşitli aşamalarında kullanılmaktadır. Bu tezin amacı yapılan önceki çalışmalara göre daha yüksek doğruluk oranında yazılım kalitesi tahmini yapmaktır. Yapılan çalışma ile veri önişleme, özellik önemi ve çeşitli makine öğrenmesi algoritmalarının kullanılması işlemleri sonucunda yazılımın kalitesinin tahmininde daha doğru sonuçlar elde edildiğini göstermektedir. Makine öğrenmesi algoritmalarının uygulanabilmesi için gerekli olan kategorik sözel verilerin sayısallaştırılması üzerine iki farklı model tasarlanmıştır. Tasarlanan modeller çerçevesinde oluşturulan veri setleri üzerinde ilgili algoritmalar uygulanmış ve doğruluk yüzdeleri karşılaştırılmıştır. Farkı makine öğrenmesi algoritmalarının birlikte kullanılmasını sağlayan topluluk yöntemleri sayesinde de çalışmada yazılım kalitesi tahmininde daha doğru sonuçlar elde edildiği gösterilmiştir.

Ocak 2022, 53 Sayfa

Anahtar Kelimeler: Makine Öğrenmesi, Topluluk Yöntemleri, Yazılım Kalitesi Tahmini

ABSTRACT

Master Thesis

PREDICTION OF SOFTWARE QUALITY WITH MACHINE LEARNING-BASED ENSEMBLE METHODS

Abidin Ayberk CERAN

Ankara University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Danışman: Dr. Öğr. Üyesi Ömer Özgür TANRIÖVER

Eş Danışman: Dr. Öğr. Üyesi Yılmaz AR

Software quality estimation is used at various stages of software projects. The aim of this thesis is to predict software quality with higher accuracy than previous studies. The study shows that data pre-processing, feature importance and machine learning algorithms provide more accurate results in estimating the quality of the software. Two different models were designed on the digitization of categorical verbal data in data sets. Data sets were arranged according to the designed models. Algorithms were applied on these datasets and accuracy percentages were compared. Ensemble methods, which enable the use of different machine learning algorithms together, contributed to obtaining results with higher accuracy rates in software quality estimation.

January 2022, 53 Pages

Key Words: Machine Learning, Ensemble Methods, Software Quality Prediction

TEŞEKKÜR

Çalışmalarımı yönlendiren, araştırmalarımın her aşamasında bilgi, öneri ve yardımlarıyla destekleyen, akademik ortamda olduğu kadar insani ilişkilerde de engin fikirleriyle yetişme ve gelişmeme katkıda bulunan danışman hocam sayın Dr. Öğr. Üyesi Ömer Özgür TANRIÖVER'e (Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı) ve eş danışman hocam sayın Dr. Öğr. Üyesi Yılmaz AR'a (Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı), çalışmalarım süresince desteklerini esirgemeyen değerli Bölüm Başkanlarıma en derin duygularla teşekkür ederim.

Çalışmalarım süresince maddi, manevi pek çok fedakârlık göstererek beni destekleyen ve her anımda yanımda olan değerli eşime, aileme ve arkadaşlarıma tüm içtenliğimle minnettarlığımı belirterek, teşekkür ederim.

Abidin Ayberk CERAN
Ankara, Ocak 2022

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Tarih boyunca teknolojik büyüme hiçbir zaman şimdiki kadar hızlı olmamıştı. Her geçen gün geliştirilen yeniliklerle birlikte, yazılım endüstrisi de bu büyümeden kendine düşen payı almaktadır. İnsanların teknolojiye her alanda bağımlı olmasının getirdiği ortam sonucunda yazılım sektörünün hızlı gelişimi kaçınılmaz bir hal almıştır. Yazılımların da insan hayatını kolaylaştırmak adına akla gelebilecek tüm sektörlerde kullanılmasıyla birlikte bu alana ilgi ve talep artmıştır. Bunun yanında günümüzdeki rekabetçi ortam da şirketleri sektörden bağımsız olarak yazılım dünyasının içerisine çekmektedir. Yapılan yenilikçi yazılımlar ile diğer şirketler arasından sıyrılarak pazardaki paylarını büyütmek şirketlerin öncelikli hedeflerinden biri olabilmektedir. Bu da beraberinde kaliteli yazılımların gerekliliğini ortaya çıkarmıştır.

Yazılımlar üzerinde yapılan ölçümler, uygulanan tekniklerin yazılımı gerçekten iyileştirip iyileştirmediğini ve ürünün kalitesini nasıl etkilediğini gösterir (Kitchenham vd. 1996). Şirketlerin içerisinde bulunduğu ticari kaygılar, yazılımda yüksek kalitenin sağlanması sonucunda daha yüksek karlar veya daha büyük pazar payı elde etme konusunda katkı sağlayabileceği düşüncesini ortaya çıkararak bu ölçümlerin yapılmasında başlıca rol oynar. Ayrıca son kullanıcıya ulaşan ürünlerin kalite düzeyinin beklentileri karşılayamaması, maliyet artışına ve beraberinde şirketin gelecekteki başarısına yönelik doğrudan etki etmektedir (Kurtel vd. 2011).

Yazılım uygulamaları, gereksinim analizinden, özelleştirmelerden ve yazılım geliştirmede yürütülen diğer faaliyetlerden kaynaklanan kusurlar içerebilir. Yazılım kalitesi ise projenin başlangıcından sonlandırılmasına kadar her aşamada kontrol işlemlerinin gerçekleştirilmesi ve kontrol sonucunda bulunan sorunların giderilmesi sürecini içermektedir. Bu nedenle yazılım kalite tahmini, çeşitli aşamalarda ihtiyaç duyulan bir faaliyettir (Vijay vd. 2017). Ayrıca birim başına düşen kusur sayısı da yazılımın kalitesini gösteren en önemli faktörlerden biri olarak kabul edilmektedir (Bowes vd. 2018). Buradan yola çıkılarak yazılımda yapılan kusur sayılarından yazılım kalitesinin tahmin edilebileceği fikri ortaya çıkmıştır.

Kısaca ifade etmek gerekirse bu tez çalışmasında kullanılan üç veri setinden biri ile daha önce yapılmış olan yazılım kalitesi tahmini çalışmaları üzerinde bazı iyileştirmeler yapılarak, elde edilen sonuçlar geliştirilmeye çalışılmıştır. Önceki çalışmalarda çeşitli formüllerin içerisinde tek başına kullanılan kusur sayısı yerine, kusur sayısının yazılımın boyutuna oranını tanımlayan kusur yoğunluğu değeri kullanılarak yazılım kalitelerinin daha doğru bir şekilde sınıflandırılması amaçlanmıştır (Lokan vd. 2001). Yine önceki çalışmalarda yazılım kalitesi iki sınıfa ayrılırken bu tez çalışmasında dört sınıf kullanılmıştır.

Bu doğrultuda üç farklı veri kümesi kullanılarak sınıflandırma işlemi gerçekleştirilmiştir. Bu veri kümelerinin ikisi üzerinde makine öğrenmesi yöntemlerini uygulayabilmek için gerekli olan kategorik sözel değerlerin sayısal değerlerle değiştirilmesi işlemi uygulanmıştır. Bu işlem iki farklı yöntem kullanılarak gerçekleştirilmiştir.

- İlk yöntemde verilecek sayısal değer için yeni bir sütun oluşturulmuştur. Farklı olan her sözel değer için ilk değer 1 olacak şekilde yeni bir rakam verilmiştir. Yeni oluşturulan ve sayısal değerlerin bulunduğu sütun, tahmin işlemine dahil edilmiştir.
- İkinci yöntemde ise her bir farklı değer için yeni bir sütun oluşturulmuştur. İlgili değer için şart sağlanıyorsa 1, diğer sütunlara ise 0 değeri atanmıştır. Bu işlem sonucunda oluşturulan tüm yeni sütunlar tahmin işlemine dahil edilmiştir.

Elde edilen doğruluk değerlerinin farklılık göstermesinin nedenleri arasında aşağıda sıralanan ifadeler etkili olmaktadır.

- Veri kümelerinin içerdiği kayıt sayılarının farklılık göstermesi sebebiyle öğrenme ve test etme işlemlerinin farklılık gösteren sayılardaki kayıtlar ile gerçekleştirilmesi
- Veri kümelerinin tahmin için kullanılan sütunları içerisinde bulunan kategorik sözel verilerin sayısal verilere dönüştürülme işleminin farklı iki yöntem kullanılarak yapılmış olması
- Uygulanan farklı topluluk yöntemlerinin bulunması

Tez içeriğindeki bölümlerde hangi bilgilerin verildiği aşağıda belirtilmiştir.

- İkinci bölümde önceki çalışmalar hakkında bilgi verilmiştir.
- Üçüncü bölümde kullanılan veri kümesi özellikleri, yapılan veri önişleme işlemleri, kullanılan makine öğrenmesi ve topluluk yöntemleri, çalışmada uygulanacak yöntem detaylıca açıklanmıştır.
- Dördüncü bölümde korelasyon matrisleri ve özellik önemi tabloları elde edilerek çalışmada kullanılan makine öğrenmesi algoritmaları ve topluluk yöntemleriyle elde edilen doğruluk oranları gösterilmiştir.
- Beşinci bölüm çalışmanın sonucunu içermektedir.

2. KURAMSAL TEMELLER VE KAYNAK ÖZETLERİ

Kusur sayılarını kullanarak yazılım kalitesi tahmin etme konusunda International Software Benchmarking Standards Group (ISBSG) veri setinin kullanılması sebebiyle doğrudan karşılaştırılabilir iki çalışma bulunmaktadır.

İlk çalışmada, yazılım kalitesi yüksek ya da düşük olarak iki sınıfa ayrılmıştır. MCLP ve MCQP yöntemlerinin ISBSG veri tabanındaki performansını ölçmek için k-kat çapraz doğrulama tekniği kullanılmıştır (Wang vd. 2010). 4017 kayıt ve 106 sütun içeren 2007 yılının Ocak ayında yayınlanan ISBSG Sürüm 10 veri seti kullanılmıştır. Önişlemeden sonra, veri setinde 374 kayıt ve 11 sütun kalmıştır.

Bu çalışmada yazılım kalitesi, aşağıdaki belirtilen değerlerin birbirine eklenmesi ile elde edilen sonucun sınır değerinden büyük ya da küçük olmasına göre sınıflandırılmıştır.

- Küçük kusur sayısı
- 2 * Büyük kusur sayısı
- 4 * Aşırı kusur sayısı

Bu çalışma sonucunda elde edilen doğruluk oranları aşağıda gösterilmiştir (Çizelge 2.1).

Çizelge 2.1 Önceki çalışmalarda elde edilen doğruluk oranları

Yöntem	**Doğruluk Yüzdesi**
MCLP	61.70%
MCQP	66.90%

Yapılan diğer bir çalışmada ise aynı şekilde 4017 kayıt ve 106 sütun içeren 2007 yılının Ocak ayında yayınlanan ISBSG Sürüm 10 veri seti kullanılarak C5.0, SVM ve Neutral network ile sınıflandırma yapılmıştır (Wang vd. 2010). Önişlemeden sonra, veri setinde 746 proje ve 53 sütun kalmıştır.

Bu çalışmaya göre bir yazılımın düşük kaliteli olarak sınıflandırılması için aşağıdaki şartlardan birini sağlaması gereklidir.

- Çok Büyük kusurlarının sayısı 1'den fazla olmalıdır.
- Büyük kusurlarının sayısı 1'den fazla olmalıdır.
- Küçük kusurlarının sayısı 10'dan fazla olmalıdır.

Geri kalanların ise yüksek kalite sınıfına ait olduğu varsayılır. Elde edilen doğruluk oranları tablo ile gösterilmiştir (Çizelge 2.2).

Çizelge 2.2 Önceki çalışmalarda elde edilen doğruluk oranları

Yöntem	**Doğruluk Yüzdesi**
SVM	69.17%
Neutral Network	70.43%
C5.0	77.88%

Sinir ağı modeli ile nesne yönelimli yazılım kalitesi tahmin etmeyi amaçlayan bir çalışma bulunmaktadır (Thwin vd. 2005). Bu çalışmada kusurlar ve sınıf başına değişen satır sayısı tahmin edilerek bu işlem gerçekleştirilir. Tahmini iyileştirmek için Ward sinir ağı ve Genel Regresyon sinir ağı kullanılmıştır. Ward sinir ağı, farklı aktivasyon fonksiyonlarına sahip bir geri yayılım ağıdır. Daha iyi tahmin sonuçlarına ulaşabilmek için bir ağ üzerinden işlenen bir modeldeki farklı özellikleri tespit etmek için gizli katmanlara uygulanırlar. Verilerin orta aralığındaki özellikleri tespit etmek için bir gizli katmanda bir Gauss fonksiyonu uygulanırken verilerin üst ve alt uçlarındaki özellikleri tespit etmek için başka bir gizli katmada başka bir Gauss tamamlayıcısı kullanılmıştır. Böylece çıktı katmanında farklı veri görünümleri elde edilerek iki özellik setinin birleştirilmesiyle daha iyi bir tahmin amaçlanmıştır. Tahmin için farklı bir mimari olan Genel Regresyon Sinir Ağı da kullanılmıştır. Sürekli değişkenlerin tahminlerini sağlayan ve temeldeki regresyon yüzeyine yakınsayan bellek tabanlı bir ağ olduğunu belirtir. Bu, oldukça paralel bir yapıya sahip tek geçişli bir öğrenme algoritmasıdır. Çok boyutlu bir ölçüm uzayında seyrek verilerle bile; algoritma, gözlemlenen değerden diğer

bir değere yumuşak geçişler sağlar. Sonuç olarak GRNN ağ modelinin Ward ağ modelinden daha doğru tahmin yaptığı gözlemlenmiştir.

Destek vektör makinesi kullanılarak yazılım kalitesi tahmini yapılmıştır (Xing vd. 2005). Destek vektör makinesinin geçerliliği ve sağlamlığı, bu yöntemi yazılım kalitesi tahmininde gerçek dünya uygulamaları için anlamlı bir araç haline getirir. İlk olarak, diğer tekniklerle modellemesi zor olan doğrusal olmayan fonksiyonel ilişkileri modellemeye uyarlanabilir. İkinci olarak, küçük eğitim örnek koşulları altında yüksek boyutlu uzaylarda bile iyi genelleme yapar. Sonuç olarak, yazılım kalitesi tahmin modelleri, diğer geleneksel tekniklerden çok daha erken oluşturulabilir. Ayrıca, yeni bir yaklaşım olan dönüştürücü destek vektör makinesi sınıflandırıcılar oluşturulurken eğitim örneklerinin yanı sıra belirli test örneklerini de dikkate alır. Uygulanan hipotez testlerine göre, destek vektör makinesi sınıflandırma sonuçlarının, ikinci dereceden diskriminant analiz veya sınıflandırma ağacından daha iyi olduğu gözlemlenmiştir.

Bayes ağlarının kalite tahminlerinde umut verici sonuçlar sağladığı gösterilen bir çalışma yapılmıştır (Wagner 2010). Faaliyet tabanlı kalite modellerinin yapısını doğrudan yansıtabilen net yapılanmaları nedeniyle, bu modellerin Bayes ağlarına aktarılması için 4 aşamalı bir yaklaşım önerilmiştir. Belirli bir değerlendirme veya tahmin hedefi hakkında bilgi sağlamak için kalite modelinde kodlanmış bilgiyi kullanan bir Bayes ağının sistematik olarak oluşturulmasına izin vermektedir. Bir kalite tanımlama modeli olan ABQM kullanılmıştır. Aynı zamanda bu model, kalite değerlendirme ve kalite tahmin modeli olan Bayes ağı ile zenginleştirilmiştir. Çalışma gerçek NASA proje verileri kullanılarak yapılmıştır. NASA'daki veri kümelerinde farklı bir efor dağılımı veya mevcut verilerde bir takım eksiklikler olması ve dikkate alınmayan faktörlerin etkisi gibi çeşitli nedenlerle yüksek oranda tahmin sonuçları elde edilememiştir.

C ile yazılmış JM1 ve C++ ile yazılmış KC2 iki NASA yazılım projesidir. Bunlardan elde edilen yazılım metrikleri ve kalite verileri, yazılım kalite tahmini için beklenti maksimizasyonu tabanlı yarı denetimli öğrenme yapılmak üzere kullanılmıştır (Seliya vd 2004). JM1 projesi, görevler için belirli tahminler oluşturmak amacıyla

simülasyonları kullanan gerçek zamanlı bir yer sistemidir. 2105 kusur içeren modül ve 8778 kusur içermeyen modül olmak üzere toplam 10883 yazılım modülünden oluşmaktadır. Tutarsız ve eksik değerlere sahip modüller çıkarıldığında, veri kümesi 10883'ten 8850 modüle düşürülmüştür. Bu azaltılmış veri kümesi, 1687 kusur içeren ve 7163 kusur içermeyen modüllerden oluşmaktadır. KC2 projesi, görevler için yer verilerinin alınması ve işlenmesi amacıyla kullanılan bir depolama yönetim sisteminin bilim verilerini işleme birimidir. Orijinal KC2 veri seti 3000'den fazla yazılım modülünden oluşmasına rağmen sadece 520 modül dikkate alınmıştır. KC2 veri kümesinde tutarsız modüller gözlemlenmesine rağmen veri kümesinin boyutu nispeten küçük olduğundan, bu modüller kaldırılmamıştır. 520 modülden 106'sı kusur içeren ve 414'ü kusur içermeyen modüllerdir. JM1 ve KC2 projelerindeki her modül 21 yazılım ölçeği ile karakterize edilmiştir. Analizde ise yalnızca on üç temel yazılım metriği kullanılmıştır. Beklenti maksimizasyonu algoritması, eksik verilerin tahmini değerlerle değiştirilmesi, model parametrelerinin tahmin edilmesi ve eksik veri değerlerinin yeniden tahmin edilmesinden oluşan yinelemeli bir yaklaşım benimser. Beklenti maksimizasyonunun her yinelemesi, her biri doğrudan istatistiksel yoruma sahip olan bir beklenti adımı ve bir maksimizasyon adımından oluşur. Etiketli-etiketsiz veri problemini simüle etmek için, JM1 yazılımı ölçüm veri kümesinden rastgele küçük bir etiketli veri seçilir. JM1 veri kümesindeki kalan örnekler, etiketlenmemiş veriler olarak kabul edilir. Beklenti maksimizasyonu algoritmasıyla oluşturulan yarı denetimli sınıflandırma modellerinin performansı, bağımsız bir test veri seti olan KC2 projesi kullanılarak değerlendirilmiştir. Daha sonra etiketli veri seti için rastgele seçilen diğer iki örnekle çalışma tekrarlanmıştır. Elde edilen sonuçlar, beklenti maksimizasyonu tabanlı yarı denetimli yaklaşımın yazılım kalite modellerinin tahminini geliştirdiğini göstermektedir. Yarı denetimli sınıflandırma modelleri; aralarında lojistik regresyon, naive bayes, adaboost, torbalama gibi bilinen modellerin de bulunduğu 25 denetimli sınıflandırıcıya kıyasla KC2 veri kümesi için benzer veya daha iyi sınıflandırma sağlamıştır.

AdaBoost'un veri örneklemesini içerdiği topluluk öğrenme yaklaşımıyla birlikte bir özellik seçimi çalışması yapılmıştır (Gao vd. 2014). Topluluk artırma tekniğinin iki biçimini veren rastgele alt örnekleme ve sentetik azınlık aşırı örnekleme olmak üzere iki

farklı örnekleme yöntemi incelenmiştir. Bu iki farklı senaryo uygulanarak topluluk öğrenme sürecinden hemen önce gerçekleştirilen özellik seçimi ve topluluk öğrenme sürecinde gerçekleştirilen özellik seçiminden hangisinin daha iyi sonuç vereceği araştırılmıştır. Deneylerde önerilen yöntem; altı filtre tabanlı özellik sıralama tekniği ve beş temel sınıflandırıcı kullanılarak bir grup veri kümesine uygulanır. Sonuç olarak topluluk öğrenme yaklaşımı içinde gerçekleştirilen özellik seçiminin, topluluk öğrenme yaklaşımından önce uygulanmasından daha iyi sınıflandırma performansı ile sonuçlandığı görülrmektedir. Beş sınıflandırma yöntemi arasında, destek vektör makinesi ve lojistik regresyon diğer yöntemlere (çok katmanlı algılayıcı, naive bayes ve k en yakın komşu) göre çok daha iyi performans göstermektedir.

Yazılımın güvenilirliğinin ve sürdürülebilirliğinin bir ölçüsü, yazılım kalitesini ifade eder (Reddivari vd. 2019). Güvenilirlik, kusurlar açısından ölçülür. Sürdürülebilirlik, kodda yapılan değişiklikler açısından ölçülür. PROMISE veri kümesindeki açık kaynak projelerinden elde edilen metrikler ile sekiz farklı popüler makine öğrenmesi algoritması denenmiştir. Bu yöntemler PART, J48, rastgele orman, bayes ağı, naive bayes, k en yakın komşu, destek vektör makinesi, yapay sinir ağlarıdır. Genel olarak, rastgele orman gibi karar ağacı tabanlı tahmin teknikleri kalite tahmininde daha iyi performans göstermiştir.

Yapılan iki çalışmada makine öğrenmesi yöntemleri kullanılarak yazılım kalitesi tahmin edilmeye çalışılmıştır.

- İlkinde veri seti, 30 öznitellik ve 36928 kayıt içeren içerir (Kumbakonam 2021). Tahmin için kullanılan algoritmalar gradyan artırma, torbalama sınıflandırıcı, rastgele orman, karar ağacı ve evrişimsel sinir ağlarıdır.
- İkincisinde veri seti 30 öznitelik ve 76119 kayıttan oluşmaktadır (Kumbakonam vd. 2021). Tahmin için bernoulli naive bayes, karar ağacı, rastgele orman, lojistik regresyon, torbalama sınıflandırıcı, gradyan artırma ve evrişimli sinir ağları yöntemleri kullanılmıştır.

Elde edilen sonuçlara bakıldığında birden fazla algoritmanın beraber kullanılabilmesine olanak sağlayan topluluk yöntemlerinden gradyan artırma ve rastgele orman yöntemlerinin daha iyi sonuçlar verdiği görülmüştür.

3. MATERYAL VE YÖNTEM

Bu bölümde kullanılan veri kümeleri, bu veri kümeleri üzerinde yapılan önişleme adımları, sınıflandırma işleminde kullanılan makine öğrenmesi algoritmaları ve topluluk yöntemleri detaylıca açıklanmıştır.

3.1 Veri Kümeleri

Bir yazılım mühendisliği veri havuzu; kaynaklar, ürünler, süreçler, teknikler, yönetim vb. hakkında nicel ve açıklayıcı bilgileri barındıran, veri kümeleri adı verilen yazılım projelerini içerir (Cheikhi vd. 2013). Bu tür veriler, tanınmış kuruluşların yanı sıra bireysel yazılım kuruluşları ve araştırmacılar tarafından çeşitli amaçlarla toplanmaktadır. Genellikle çoğu bilim ve mühendislik disiplininde; kıyaslama ve deneysel çalışmalar yürütmek için kullanılır.

3.1.1 ISBSG veri kümesi

ISBSG, 1997 yılında bir grup ulusal yazılım ölçüm birliği tarafından Avustralya'da kuruldu. Kar amacı gütmeyen bir kuruluştur. Amaçları, yazılım süreçlerini ve ürünlerini iyileştirmek için Bilgi Teknolojileri endüstrisi verilerinin kullanımını teşvik etmekti. Bugün ISBSG'nin çeşitli ölçüm derneklerinden ve özel şirketlerden ortakları vardır. Dünyanın her yerinden müşteriler, yazılım projelerinin planlamasını iyileştirmek için proje verilerini, raporlarını ve tahmin araçlarını kullanmaktadır. Ayrıca üniversitelere ve akademisyenlere kaynaklar sağlamaktadır.

Yapılan çalışmada kullanılan ISBSG 11 Veri Sürümü Haziran 2009'da yayınlanmıştır. Veri kümesinde 5052 projenin bilgisi yer almaktadır. 20 adet birinci seviye özniteliğe ve 118 adet ikinci seviye özniteliğe sahiptir. ISBSG veri kümesinin bir kesiti ve yazılım kalitesi tahmininde kullanılan özelliklerin açıklamaları aşağıda verilmiştir (Çizelge 3.1).

- Veri Kalitesi: Bu alan, aşağıdakileri belirtmek için ISBSG kalite gözden geçirenleri tarafından proje verilerine uygulanan A, B, C veya D ISBSG derecelendirme kodunu içerir:

    - A: Gönderilen veriler sağlam olarak değerlendirildi ve bütünlüğünü etkileyebilecek hiçbir şey tanımlanmadı.
    - B: Gönderim temelde sağlam görünüyor ancak gönderilen verilerin bütünlüğünü etkileyebilecek bazı faktörler var.
    - C: Önemli verilerin sağlanmaması nedeniyle, gönderilen verilerin bütünlüğünü değerlendirmek mümkün olmadı.
    - D: Bir faktör veya faktörlerin bir kombinasyonu nedeniyle, sunulan verilere çok az güvenilirlik verilmelidir.
- Proje Yılı: Projeyle ilgili aşağıdaki tarihlerden birisi elde edilebiliyorsa bu tarihlerden biri kullanılır. Aksi durumda proje tarihi bilinmiyorsa, verinin ISBSG tarafından alındığı yıldır.
    - Proje bitiş tarihi
    - Proje başlangıç tarihi
    - Tahmini uygulama tarihi
    - Veri toplanma tarihi
- Sayma Yaklaşımı: Projenin hangi büyüklükte olduğunu belirtmek için kullanılan tekniğin ifade edilmesidir. ISBSG deposundaki çoğu proje için bu, IFPUG, MARK II, NESMA, FiSMA, COSMIC gibi işlevsel boyutu ölçmek için kullanılan işlevsel boyut ölçüm yöntemleriyle isimlendirilir.
- Fonksiyonel Boyut: Projede uygulanan fonksiyonel boyutlama metoduna bağlı olarak farklı ölçümler ve sonuçlar elde edilebilir.
    - IFPUG, NESMA, FiSMA ve MARK II sayıları için:
        - Tam boyutlu bir döküm sağlandığında, bu, herhangi bir ayarlama faktörü tarafından ayarlanmayan hesaplanmış işlevselliktir.
        - Boyut verisinin sağlanmadığı, ancak toplam ayarlanmış sayım ve ayar faktörünün verildiği durumlarda, verilerden hesaplanarak bulunan ayarlanmamış işlevsellik değerine eşittir.
        - Ayarlanmamış bir işlevsel boyut değeri mevcut ise bu değere eşittir.
        - Ayarlanmamış fonksiyonel sayının sağlanmadığı veya hesaplanamadığı durumlarda bu değer boştur.

    - COSMIC-FFP sayıları için: Bu, COSMIC İşlev Noktalarında (CFP) toplam işlevsel boyuttur.
    - DİĞER boyut ölçü sayıları için: Boyut verileri, Kod Satırları veya Diğer boyut birimlerindeki 'FSM Dışındaki Boyut' bölümündedir.
- Ayarlanmış İşlev Noktaları: Nihai sayımda projenin düzeltilmiş işlevsel boyutunu ifade eder. Bu sayı, kullanılan sayım yaklaşımına bağlıdır.
    - IFPUG, NESMA, FiSMA ve MARK II sayıları için: Ayarlanan fonksiyon puan sayısı, (Kullanıldığı yerde Değer Ayarlama Faktörü ile ayarlanır).
    - COSMIC-FFP sayıları için: COSMIC İşlev Noktaları (CFP) cinsinden toplam işlevsel boyut.
    - DİĞER boyut ölçü sayıları için: Boyut verileri, Kod Satırları veya Diğer boyut birimlerindeki 'FSM Dışındaki Boyut' bölümündedir.
- Normalleştirilmiş İş Çabası: Bildirilen tüm ekipler için tam yaşam döngüsü çalışmasını ifade eder.
    - Tam bir geliştirme yaşam döngüsünden daha azını kapsayan projeler için bu değer, rapor edilen tüm ekipler için tam yaşam döngüsü çabasının bir tahminidir.
    - Geliştirme yaşam döngüsünün tamamını kapsayan projeler ve yaşam döngüsü kapsamının bilinmediği projeler için bu değer, özet iş çabası ile aynıdır.
    - Özet iş çabasının bilinmediği projeler için bu değer boştur.
- Özet İş Çabası: Projeye göre kaydedilen saat cinsinden toplam harcanan çaba değerini ifade eder.
    - Tam bir geliştirme yaşam döngüsünden daha azını kapsayan projeler için bu değer yalnızca rapor edilen aşamalar için çabayı kapsar. Raporlanan tüm ekipler için çabayı içerir.
    - Geliştirme yaşam döngüsünün tamamını kapsayan projeler ve yaşam döngüsü kapsamının bilinmediği projeler için bu değer, rapor edilen tüm ekipler için toplam çabadır.
    - Toplam çabanın bilinmediği projeler için bu değer boştur.

- Normalleştirilmiş Verimlilik Teslimat Oranı: 2002 yılından bu yana ISBSG tarafından kullanılan ve raporlanan proje teslim oranıdır. Normalleştirilmiş çaba ve ayarlanmamış sayım kullanımı, normalize edilmemiş çaba ve ayarlanmış sayımdan daha karşılaştırılabilir oranlar vermelidir. İşlevsel büyüklük birimi başına saat cinsinden Normalleştirilmiş İş Çabası / Fonksiyonel Boyut formülü yardımıyla hesaplanır.
- 2002 Öncesi Verimlilik Teslimat Oranı: 2002 yılı öncesinde ISBSG tarafından kullanılan ve raporlanan projenin teslim oranıdır. O zamandan beri, analiz ve raporlama için Normalleştirilmiş PDR kullanılmaktadır. İşlevsel büyüklük birimi başına saat cinsinden Özet İş Çabası / Düzeltilmiş İşlev Noktaları formülü yardımıyla hesaplanır.
- Toplam Kusur Sayısı: Yazılımın kullanımının ilk ayında rapor edilen toplam kusur sayısını ifade eder. Bu sütun, rapor edilen kusurların küçük, büyük ya da çok büyük olma durumlarını önemsemeksizin sadece sayılarının toplamını gösterir. Kusurların büyüklükleri konusunda herhangi bir bilgi bulunmadığı durumlarda, kusur sayısı mevcut ise eldeki tek değer yine burada gösterilir.
- Geliştirme Türü: Bu alan, yazılımın hangi aşamadan itibaren geliştirildiğini belirtmek için aşağıdaki ifadeleri kullanır.
    - Yeni bir geliştirme
    - İyileştirme
    - Yeniden geliştirme
- Kaynak Seviyesi: Raporlanan iş çabası verilerine dahil olan kişiler hakkında veriler toplanır. ISBSG veri havuzunda dört seviyede tanımlanmıştır. Bu alandaki sayı, bu ve önceki seviyelerdeki tüm çaba alanlarına dahil edildiğini gösterir. Örneğin, bir proje için bu alanda "3" olması, geliştirme ekibi, geliştirme ekibi desteği ve bilgisayar operasyonları için iş çabası numarasına dahil olduğu anlamına gelir.
    - 1 = geliştirme ekibi çalışması (proje ekibi, proje yönetimi, proje yönetimi)
    - 2 = geliştirme ekibi desteği (veritabanı yönetimi, veri yönetimi, kalite güvencesi, veri güvenliği, standartlar desteği, denetim ve kontrol, teknik destek)

- 3 = bilgisayar operasyonlarının katılımı (yazılım desteği, donanım desteği, bilgi merkezi desteği, bilgisayar operatörleri, ağ yönetimi)
- 4 = son kullanıcılar veya müşteriler (kullanıcı irtibatları, kullanıcı eğitim süresi, uygulama kullanıcıları ve/veya müşteriler)

Çizelge 3.1 ISBSG veri kümesi kesiti

Proje Numarası	Veri Kalitesi	Proje Yılı	Sayma Yaklaşımı	Fonksiyonel Boyut
10001	D	1998	NESMA	237
10011	B	1996	IFPUG	443
10012	B	2002	IFPUG	76
10014	B	2004	IFPUG	3
10015	B	2000	IFPUG	382

3.1.2 EBSPM veri kümesi

Evidence-Based Software Portfolio Management kısa adıyla EBSPM, yazılım şirketlerinin yazılım dağıtım portföylerini aktif olarak geliştirmelerini desteklemek için kanıta dayalı, pratik bir model olarak geliştirilmiş bir araştırma havuzudur (Huijgens 2016).

EBSPM'nin amacı, bir şirketin yazılım sağlama yeteneğinin inovasyonunu desteklemek için birbirine bağlı yazılım projelerinin performansını; boyut, maliyet, süre ve kusur sayısı açısından ölçmek, analiz etmek ve kıyaslamaktır.

EBSPM veri tabanında Hollanda ve Belçika'daki 4 farklı şirket tarafından yapılmış 492 tamamlanmış yazılım projesi ve bu projelerin 37 öznitelik değeri bulunmaktadır. Veri kümesinde işlevsel boyut ve kusur sayısı değerleri yer almasına rağmen, kusur yoğunluğu değeri bulunmamaktadır. Bu nedenle kusur yoğunluğu değeri; Kusur * 1000/Fonksiyonel boyut formülü kullanılarak hesaplanır. Yazılım kalitesi tahmininde bu değer kullanılır.

Yapılan çalışmada kullanılan EBSPM veri kümesinin bir kesiti ve yazılım kalitesi tahmininde kullanılan özelliklerin açıklamaları aşağıda verilmiştir. (Çizelge 3.2). Bu veri kümesi nispeten yakın zamanda, 24 Temmuz 2017'de yayınlanmıştır.

- Firma İsmi: Bir projeyi hangi şirketin gerçekleştirdiğini ifade eder. Şirket isimleri anonim ifadeler olarak seçilmiştir.
    - Banka A
    - Banka B
    - Beltel
    - DutchCo
- Proje Tamamlanma Yılı: 2008'den 2016'ya kadar olan proje tamamlanma yılını ifade eder.
- Firma Sektörü: Projeyi geliştiren firmanın sektörlerinden hangisinde faaliyet gösterdiğini ifade eder.
    - Bankacılık
    - Telekomünikasyon
    - Faturalandırma
- Geliştirme Yöntemi: Projenin geliştirme yönteminin belirtilmesi için kullanılır.
    - Plan odaklı
    - RUP
    - Çevik
- Geliştirme Sınıfı: Projeye hangi aşamadan başlandığını ifade etmek için kullanılır.
    - Yeni geliştirme
    - Küçük geliştirme (%5-25 yeni)
    - Büyük geliştirme (%25-75 yeni)
    - Dönüştürme
- Fonksiyonel Boyut: Fonksiyon Noktalarında bir projenin fonksiyonel boyutunu ifade eder.
- Gerçekleştirme Süresi: Bir projenin fiili olarak Proje başlangıcından yayınlandığı ana kadar kaç ay süre geçtiği ifade eder.

- Toplam Maliyet: Euro cinsinden bir projenin başlangıcından yayınlandığı ana kadar maliyetini ifade eder. Lisans maliyetleri ve donanım yatırımları maliyete dahil edilmemiştir.
- Kusurlu İşlem Sayısı: Sistem kurulumundan proje yayınlanma anına kadar bir projede bulunan gerçek kusur, hata veya arıza sayısını ifade eder.

Çizelge 3.2 EBSPM veri kümesi kesiti

Proje Numarası	Firma İsmi	Proje Yılı	Firma Sektörü	Geliştirme Yöntemi
411	Beltel	2015	Telekomünikasyon	Plan Odaklı
429	Beltel	2015	Telekomünikasyon	Plan Odaklı
430	Beltel	2015	Telekomünikasyon	Plan Odaklı
403	Beltel	2015	Telekomünikasyon	Çevik
434	Beltel	2015	Telekomünikasyon	Plan Odaklı

3.1.3 PROMISE veri kümesi

Yazılım mühendisliğinde tahmin modelleri üzerine 15 Mayıs 2005 tarihinde Amerika'da kısa ismi PROMISE olan Yazılım Mühendisliğinde Tahmin Modelleri isimli uluslararası bir çalıştay düzenlenmiştir. Bu çalıştayın ana teması, yazılım modellerinde tahmin oluşturmaya ilişkin sorunlar ve zorluklardır. Çalıştayın amaçları şunlardır:

- Deneyim ve uzmanlık paylaşımı amacıyla tahmine dayalı modeller oluşturmaya ilgi duyan farklı geçmişlere sahip araştırmacıları ve uygulayıcıları bir araya getirmek.
- Tahmine dayalı yazılım modelleri oluşturmaya ilişkin çeşitli yönler ve konular hakkında tartışma ve tartışmayı yönlendirmek.
- Yazılım mühendisliği veri kümelerinin halka açık bir havuzunun oluşturulmasını başlatmak.
- Alandaki araştırmacılar tarafından gerekli görülen açık araştırma sorularının bir listesini oluşturmak

Bu çalıştayda araştırmacılara tahmine dayalı yazılım modelleri ve genel olarak yazılım mühendisliği topluluğu oluşturmaya hizmet etmek için halka açık veri kümeleri ile araçlardan oluşan bir koleksiyon oluşturulmuştur. Bu açık veri kümelerinden A. Güneş Koru tarafından depoya eklenmiş KC1 için sınıf düzeyinde veriler isimli kusur sayısı değeri içeren ve yazılım kusur tahmininde kullanılmış veri kümesi tez çalışmasında kullanılmıştır (Koru vd. 2005). Veri kümesi 145 kayıt ve 97 öznitelik değerinden oluşmaktadır. Veri kümesi kusur yoğunluğu değerini içermemektedir. Bu değer üzerinden yazılım kalitesi sınıfları belirlenmesi sebebiyle kusur yoğunluğu değeri; Kusur * 1000/ Satır sayısı formülü kullanılarak hesaplanır. Yazılım kalitesi tahmininde bu değer kullanılır. Veri kümesinin bir kesiti ve içerisindeki yazılım kalitesi tahmininde kullanılan özellikler açıklamalarıyla birlikte verilmiştir (Çizelge 3.3).

- Genel Veri Yüzdesi: Bir sınıftaki genel ve korunan verilerin yüzdesini ifade eder. Başka bir ifade ile kapsüllemenin ölçüsüdür. Genel olarak, daha düşük değerler daha büyük kapsüllemeyi gösterir.
- Nesneler Arasında Bağlantı: Bir sınıfın bağlı olduğu kalıtımla ilgili olmayan farklı sınıfların sayısını ifade eder. Hiyerarşisinin dışındaki birçok sınıfa büyük ölçüde bağımlı olan bir sınıf bir kütüphaneye tanıtılırsa, bağlı olduğu tüm sınıfların da tanıtılması gerekir.
- Derinlik: Bir sınıfın kendi sınıf hiyerarşisi içinde hangi düzeyde bulunduğunu gösterir. Genel olarak, derinlik arttıkça kalıtım artar. Bir sınıfın seviyesini belirten ifadedir. Örneğin, bir ebeveynin bir çocuğu varsa, çocuğun derinliği ikidir.
- Metotların Uyumluluk Eksikliği: Bir sınıftaki her veri alanı için, o veri alanını kullanan sınıftaki yöntemlerin yüzdesini ifade eder. Yüzdelerin ortalaması alınır ve ardından %100'den çıkarılır. Bu metrik, düşük veya yüksek uyum yüzdesini gösterir.
    - Yüzde düşükse, sınıf birbirine bağlıdır.
    - Yüzde yüksekse, sınıfın ayrı ayrı daha fazla uyum sağlayacak ayrı sınıflara bölünebileceğini gösterebilir.
- Sınıf İçin Yanıt: Bir sınıf içinde uygulanan metotların sayısı ile kalıtım nedeniyle bir nesne sınıfı tarafından erişilebilen metotların sayısının toplanması

sonucunda elde edilen sayıyı ifade eder. Genel olarak, daha düşük değerler daha büyük polimorfizmi gösterir.

- Sınıf Başına Ağırlıklı Metotlar: Sınıf hiyerarşisinde erişilebilir tüm metotlar yerine bir sınıf içinde uygulanan metotların sayısını ifade eder. Genel olarak, daha düşük değerler daha büyük polimorfizmi gösterir.
- Benzersiz Operatörlerin Sayısı: Operatörler, derleyiciye belirli matematiksel veya mantıksal işlemleri gerçekleştirmesini söyleyen sembollerdir. Kullanılmış olan farklı operatörlerin sayısını ifade eder.
- Toplam Kod Satırı: Bir koddaki açıklama veya boş satır olmayan herhangi bir metin satırıdır. Program başlık dosyalarını, herhangi bir değişkenin bildirimini ve çalıştırılabilir ya da çalıştırılamaz ifadeleri içeren tüm satırlardan oluşur. Kod satırları yalnızca kodun hacmini ifade etmemizi sağlar.
- Çocuk Sayısı: Belirli bir sınıftan türetilen sınıfların sayısını ifade eder.
- Çocuğa Bağlılık: Bir sınıfın bir alt sınıfa bağımlı olup olmadığını gösterir.
- Üst Sınıflara Bağlılık: Daha yüksek modüller tarafından yapılan çağrıların sayısıdır.
- Kusur Sayısı: Bir sınıf için kaydedilmiş kusur sayısını ifade eder.

Çizelge 3.3 PROMISE veri kümesi kesiti

Genel Veri Yüzdesi	**Nesneler Arasında Bağlantı**	**Derinlik**	**Metotların Uyumluluk Eksikliği**	**Çocuk Sayısı**
0	24	4	100	0
0	19	4	100	0
100	13	1	88	0
0	21	4	100	0
5	17	2	90	0

3.2 Yöntem

Bu tez çalışmasının amacı yazılım kalite sınıflarının makine öğrenmesi ve topluluk yöntemleri kullanılarak tahmin edilmesidir. ISBSG veri kümesi kullanılarak yapılan doğrudan karşılaştırılabilir önceki çalışmalara bakıldığında yazılım kalite sınıflarının sadece yüksek kalite ve düşük kalite olarak iki sınıfa ayrıldığı görülmektedir. Ayrıca yazılımın büyüklüğü hesaba katılmadan sadece kusur sayıları üzerinden yapılan hesaplamaların doğruluğu da ayrı bir tartışma konusu olabilir.

Buradan yola çıkılarak yazılım kalitesi hesaplamasında yazılım büyüklüğünün de formül içerisine dahil edildiği kusur yoğunluğu değeri kullanılarak daha doğru sonuçlar elde edilebileceği öngörülmüştür. Ek olarak kalite sınıflandırılması iki yerine dört sınıfa ayrılarak sınıflar arasındaki aralıkların azaltılması amaçlanmıştır.

Tahmin için kullanılacak veri kümelerinde boş değer bulunduran sütunlar bulunmaktadır. Bu nedenle, en az sayıda boş satır bulunduran sütunlar, tahmin işleminde kullanılacak sütunlar olarak seçilmiştir. Seçilen sütunlar üzerinde veri önişleme adımları uygulanarak tahmin aşaması için uygun hale getirilmiştir. Tahmin doğruluğu açısından önem arz etmesi sebebiyle, eksik ve tanımsız değerler barındıran satırlar veri kümelerinden silinmiştir. Ayrıca seçilen sütunlar içerisinde sözel veriler içeren kategorik değerler var ise sayısal değerlere dönüştürülmüştür. Bunun nedeni kullanılan kütüphane tarafından sözel değerlerin tahmin işleminde kullanılmasına izin verilmemesidir. Bu işlem iki farklı şekilde gerçekleştirilmiştir. İki farklı çalışmada yapılan işlemlerin detayları ilgili bölümlerde anlatılmıştır. Ayrıca veri kümelerine kusur yoğunluklarının büyüklüklerine göre 4 sınıfa ayrılmış olan yazılım kalitesi niteliği eklenmiştir. Sonrasında makine öğrenmesi algoritmaları ve topluluk yöntemleri uygulanarak yazılım kalitesi tahmini gerçekleştirilmiştir.

3.3 Veri Önişleme

Tahmin edilecek yazılım kalitesi sınıflarını belirleme konusunda kusur yoğunluğu değeri kullanılmıştır. Bu nedenle veri kümesi içerisinde kusur yoğunluğu değerinin kendisi ve hesaplanmasında kullanılan sütunlar içerisinde eksik ya da tanımsız değerler içeren satırlar tahmin doğruluğu konusunda önem arz etmesi sebebiyle silinmiştir. Kusur yoğunluklarının büyüklüklerine göre 4 sınıfa ayrılmış olan yazılım kalitesi niteliği tüm veri kümelerine eklenmiştir.

Toplamda çalışmada kullanılan üç veri kümesi içerisinden ikisinde kategorik sözel veriler bulunduğundan ISBSG ve EBSPM veri kümeleri için iki farklı veri sayısallaştırma yöntemi kullanılarak çalışmalar gerçekleştirilmiştir.

- İlk yöntemde kategorik sözel verilerin sayısal verilerle değiştirilmesi işlemi için yeni bir sütun oluşturulmuştur. İlgili sütun içerisindeki her bir farklı değer için 1'den başlanıp artan sayılar kullanılarak numaralandırılma işlemi gerçekleştirilmiştir. Oluşturulan yeni sütun aşağıda gösterildiği gibi veri kümesine eklenmiştir.

Çizelge 3.4 İlk yöntem ile sözel verilerin sayısal verilere dönüştürülmesi

Veri Kalitesi	**Veri Kalitesi**
A	1
B	2
C	3
D	4

- İkinci yöntemde ise tahmin için seçilmiş ve kategorik sözel veriler içeren sütunlardaki değerin sayısal değerlere dönüştürülmesi işleminde ilgili sütun içerisinde kaç farklı değer bulunuyor ise o kadar yeni sütun oluşturulmuştur. Her bir sütuna değer ataması yapılıp ilgili değer o sütun ile eşleşiyorsa 1 eşleşmiyorsa 0 olarak sayısallaştırma işlemi gerçekleştirilmiştir. Bu işlemin

yapılma amacı sütun içerisindeki değerler arasındaki aralıkların azaltılması ve tahmin aşamasında daha doğru sonuçlar elde edilmesidir. Oluşturulan yeni sütunlar aşağıda gösterildiği gibi veri kümesine eklenmiştir.

Çizelge 3.5 İkinci yöntem ile sözel verilerin sayısal verilere dönüştürülmesi

Veri Kalitesi	**A**	**B**	**C**	**D**
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

3.3.1 ISBSG ilk yöntem ile veri önişleme

En az sayıda boş değer bulunduran sütunların seçilmesi işleminden sonra ISBSG veri kümesi 11 tahmin sınıfı ve 1 hedef sınıfa indirgenmiştir. Bu sütunlar içerisinde boş değer bulunduran satırların silinmesi sonucunda elde edilen nihai veri kümelerinin kesitleri aşağıda verilmiştir.

Çizelge 3.6 ISBSG ilk çalışma eksik verilerin silinmesi

Adım	**Öznitelik**	**Filtre**	**İşlem Öncesi Satır Sayısı**	**Silinen Satır Sayısı**	**Kalan Satır Sayısı**
1	Kusur Yoğunluğu	Boş Değer	5052	4292	760
2	Foksiyonel Nokta	Boş Değer	760	15	745

Kusur yoğunluklarının büyüklüklerine göre 4 sınıfa ayrılmış olan yazılım kalitesi niteliği aşağıda gösterildiği şekilde veri kümesine eklenmiştir.

Çizelge 3.7 ISBSG ilk çalışma kalite sınıfları

Yazılım Kalitesi Sınıfı	Kusur Yoğunluğu	Kayıt Sayısı
1	0	273
2	0.5 – 20	246
3	20 – 80	173
4	80 – 4237	52

3.3.2 ISBSG ikinci yöntem ile veri önişleme

En az boş değer bulunduran sütunların seçilmesi işleminden sonra ISBSG Veri Kümesi 12 tahmin sınıfı ve 1 hedef sınıfa indirgenmiştir. Bu sütunlar içerisinde boş değer bulunduran satırların silinmesi sonucunda elde edilen nihai veri kümelerinin kesitleri aşağıda verilmiştir.

Çizelge 3.8 ISBSG ikinci çalışma eksik verilerin silinmesi

Adım	Öznitelik	Filtre	İşlem Öncesi Satır Sayısı	Silinen Satır Sayısı	Kalan Satır Sayısı
1	Kusur Yoğunluğu	Boş Değer	5052	4292	760

Kusur yoğunluklarının büyüklüklerine göre 4 sınıfa ayrılmış olan yazılım kalitesi niteliği aşağıda gösterildiği şekilde veri kümesine eklenmiştir.

Çizelge 3.9 ISBSG ikinci çalışma kalite sınıfları

Yazılım Kalitesi Sınıfı	Kusur Yoğunluğu	Kayıt Sayısı
1	0	276
2	0.5 – 20	254
3	20 – 80	176
4	80 – 4237	54

Çizelge 3.10 ISBSG yeni oluşturulan sütun sayıları

Öznitelik	Yeni Oluşturulan Sütun Sayısı
Veri Kalitesi	4
Sayım Yaklaşımı	5
Geliştirme Türü	5
Kaynak Seviyesi	4

3.3.3 EBSPM ilk yöntem ile veri önişleme

En az sayıda boş değer bulunduran sütunların seçilmesi işleminden sonra EBSPM veri kümesi 10 tahmin sınıfı ve 1 hedef sınıfa düşürülmüştür. Bu sütunlar içerisinde boş değer bulunduran satırların silinmesi sonucunda elde edilen nihai veri kümelerinin kesitleri aşağıda verilmiştir.

Çizelge 3.11 EBSPM ilk çalışma eksik verilerin silinmesi

Adım	Öznitelik	Filtre	İşlem Öncesi Satır Sayısı	Silinen Satır Sayısı	Kalan Satır Sayısı
1	Kusur Sayısı	Boş Değer	492	222	270

Kusur yoğunluklarının büyüklüklerine göre 4 sınıfa ayrılmış olan yazılım kalitesi niteliği aşağıda gösterildiği şekilde veri kümesine eklenmiştir.

Çizelge 3.12 EBSPM ilk çalışma kalite sınıfları

Yazılım Kalitesi Sınıfı	Kusur Yoğunluğu	Kayıt Sayısı
1	0 – 80	56
2	80 – 160	74
3	160 – 320	54
4	320 – 4875	86

3.3.4 EBSPM ikinci yöntem ile veri önişleme

EBSPM Veri Kümesi 9 tahmin sınıfı ve 1 hedef sınıfa düşürülmüştür. Bu sütunlar içerisinde boş değer bulunduran satırların silinmesi sonucunda elde edilen nihai veri kümelerinin kesitleri aşağıda verilmiştir.

Çizelge 3.13 EBSPM ikinci çalışma eksik verilerin silinmesi

Adım	Öznitelik	Filtre	İşlem Öncesi Satır Sayısı	Silinen Satır Sayısı	Kalan Satır Sayısı
1	Kusur Sayısı	Boş Değer	492	222	270

Kusur yoğunluklarının büyüklüklerine göre 4 sınıfa ayrılmış olan yazılım kalitesi niteliği aşağıda gösterildiği şekilde veri kümesine eklenmiştir.

Çizelge 3.14 EBSPM ikinci çalışma kalite sınıfları

Yazılım Kalitesi Sınıfı	Kusur Yoğunluğu	Kayıt Sayısı
1	0 – 80	56
2	80 – 160	74
3	160 – 320	54
4	320 – 4875	86

Çizelge 3.15 EBSPM yeni oluşturulan sütun sayıları

Öznitelik	Yeni Oluşturulan Sütun Sayısı
Firma İsmi	4
Firma Sektörü	3
Geliştirme Yöntemi	3
Geliştirme Sınıfı	4

3.3.5 PROMISE veri önişleme

Diğer iki veri kümesinde olan kategorik sözel veriler ya da eksik veriler bulunmaması nedeniyle PROMISE veri kümesinde herhangi bir işlem yapılmamıştır. Sadece veri kümesindeki öznitelikler içerisinde seçim yapılmıştır. 12 tahmin niteliği ve 1 hedef sınıfa indirgenmiştir. Kusur yoğunluklarının büyüklüklerine göre 4 sınıfa ayrılmış olan yazılım kalitesi niteliği aşağıda gösterildiği şekilde veri kümesine eklenmiştir.

Çizelge 3.16 PROMISE kalite sınıfları

Yazılım Kalitesi Sınıfı	Kusur Yoğunluğu	Kayıt Sayısı
1	0	85
2	0.5 – 13.4	20
3	13.4 – 32	20
4	32 – 141	20

3.4 Sınıflandırma

Bu çalışmada makine öğrenmesi ve topluluk yöntemleri kullanılarak çok sınıflı sınıflandırma yapılmıştır. Tüm kavramların açıklamaları bu bölümde verilmiştir.

Sınıflandırma; veri örnekleri için grup üyeliğini tahmin etmekte kullanılan bir veri madenciliği ya da makine öğrenimi tekniğidir (Chaitra vd. 2018). Sınıflandırma, bilinen yapıları yeni verilere uygulamak için genelleştirme görevine verilen isimdir. Kümeleme ise verilerdeki bilinen yapıları kullanmadan, benzer olan grupları ve yapıları keşfetme görevine verilen isimdir.

Sınıflandırma; ikili ve çok sınıflı sınıflandırma olarak iki kategoriye ayrılır.

- İkili veya iki terimli sınıflandırma; verilen bir kümenin öğelerini bir sınıflandırma kuralı temelinde iki gruba ayırarak her birinin hangi gruba ait olduğunu tahmin etme görevidir.
- Çok sınıflı veya çok terimli sınıflandırma; örnekleri üç veya daha fazla sınıftan birine sınıflandırma sorunudur.

Sınıflandırma; denetimli, denetimsiz ve yarı-denetimli olarak üç şekilde gerçekleşir.

- Denetimsiz: Verilerin etiketsiz olduğu durumda kullanılır. Sadece girdi verileri bulunur. Karşılığında çıktı değişkeni bulunmaz. Algoritmalar, giriş verilerinden

ilişki kurmaya çalışarak sonucu üretir. Ayrıca denetimsiz öğrenme sorunları; kümeleme ve ilişkilendirme sorunları olarak da gruplandırılabilir.

- Kümeleme: Bir kümeleme sorunu, verilerdeki doğal gruplandırmaların keşfedilmesi için kullanılır.
- İlişkilendirme: Birliktelik kuralı öğrenme problemi, verilerin büyük bölümlerini tanımlayan kuralların bulunması için kullanılır.

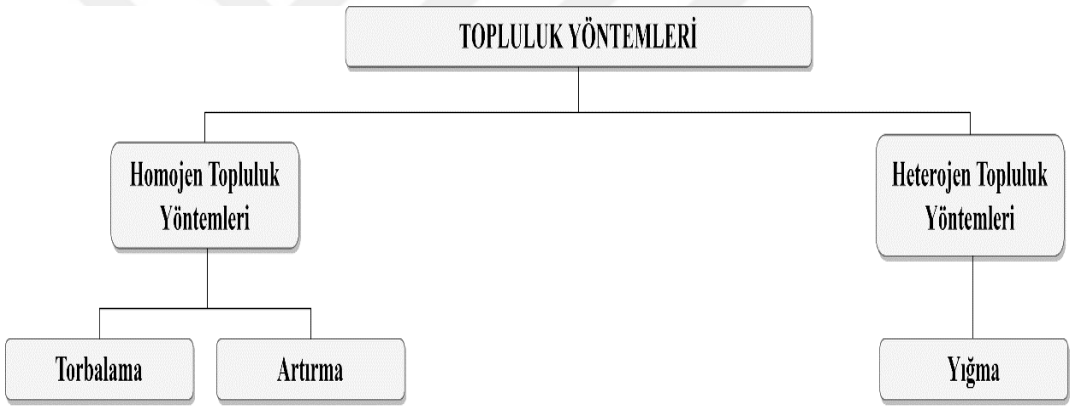
- Yarı denetimli: Eğitim için hem etiketli hem de etiketsiz verilerin kullanılmasını ifade eder. Etiketli örnekler, uzman görüşü ve çabasını gerektirdiğinden, elde edilmesi genellikle zor, pahalı veya zaman alıcıdır. Etiketlenmemiş verilerin toplanması nispeten kolay olmakla birlikte bu verileri kullanmanın birkaç yolu vardır. Yarı denetimli öğrenme, daha iyi sınıflandırıcılar oluşturmak için etiketlenmiş verilerle birlikte büyük miktarda etiketlenmemiş veri kullanarak bu sorunu çözer. Çünkü yarı denetimli öğrenme daha az insan çabası gerektirir ve daha yüksek doğruluk sağlar.
- Denetimli: Tüm veriler etiketlidir. Algoritmalar, girdi verilerinden çıktıyı tahmin etmeyi öğrenir. Denetimli öğrenme yapılırken girdi değişkenlerinden çıktıya ulaşma adımlarını öğrenmek için bir algoritma kullanılır. Denetimli öğrenme problemleri, iki grupta incelenebilir.
  - Sınıflandırma: Çıktıyı ifade eden değişkenin bir kategoriyi temsil etmesidir.
  - Regresyon: Çıktıyı ifade eden değişkenin gerçek bir değeri temsil etmesidir.

Makine öğrenimi; bilgiyi deneyimden ve diğer yollardan otomatik olarak öğrenme yeteneğine sahip bir sistemi ifade eder. Topluluk öğrenimi ise genellikle kalabalığın bilgeliği teorisinin makine öğrenmesi konusunda bir yorumu olarak kabul edilir (Sagi vd. 2018). Denetimli makine öğrenimi görevlerinde karar vermek için birden çok sınıflandırma algoritmasını birleştiren yöntemlerin ifade edilmesinde kullanılan bir kavramdır.

Temel öğrenen olarak da adlandırılan bir sınıflandırıcının, girdi olarak bir dizi etiketli veriyi örnek olarak alması ve bu örnekleri genelleştiren bir model üretmesi fikrini temel alır. Üretilen model kullanılarak henüz etiketlenmemiş yeni örnekler için tahminler

yapılabilir. Karar ağacı, sinir ağı, doğrusal regresyon modeli gibi herhangi bir türde makine öğrenme algoritması topluluk öğrenmesinde kullanılabilmektedir.

Topluluk öğrenmesinin temel hedefi, birden fazla modelin birleştirilmesi ve tek bir sınıflandırma algoritmasından kaynaklanabilecek hataların diğer sınıflandırma algoritmaları tarafından telafi edilmesini sağlamaktır. Bunun sonucunda, topluluğun genel tahmin performansının tek bir sınıflandırma algoritmasından daha iyi olması beklenmektedir. Bir dizi zayıf öğrenen, tek bir güçlü öğrenen yaratabilir fikrinden yola çıkmaktadır (Schapire 1990). Bir dizi sınıflandırıcının birleşimiyle yapılan tahminlerin en iyi tek sınıflandırıcı tarafından yapılan tahminlerden genellikle daha doğru olduğu bulunmuştur (Hansen vd. 1990).

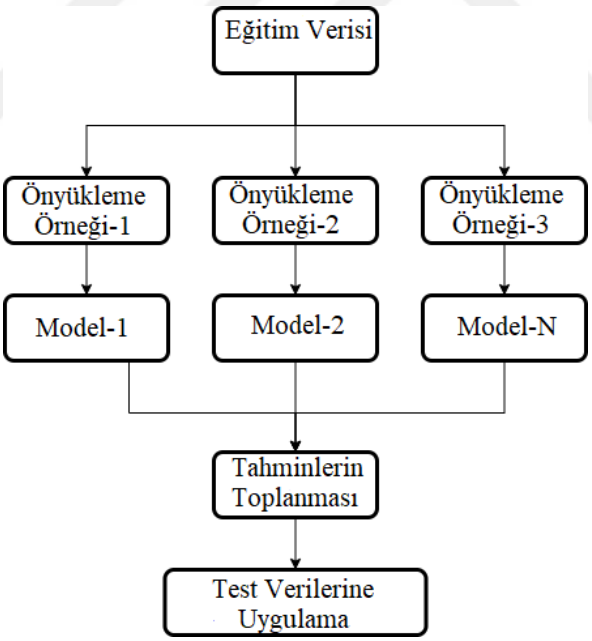


Şekil 3.1 Topluluk Yöntemleri şeması

Aynı eğitim veri kümesi kullanılarak her biri birbirinden farklı modeller oluşturulur. Mevcut eğitim verileri küçük veri kümelerine bölünüp oluşturulan modeller kullanılarak eğitilir. Bu modeller oluşturulma yöntemine göre ikiye ana başlıktan oluşur.

- Aynı makine öğrenmesi algoritması kullanılarak oluşturulduğunda homojen topluluk yöntemi olarak adlandırılır. Homojen topluluk yöntemi de modellerin çalışma yöntemine göre ikiye ayrılır.
  - Torbalama: Torbalama kelimesi önyükleme ve toplama kelimelerinin birleştirilmesiyle oluşmuştur (Breiman 1996). Farklı temel öğrenenler oluşturmak için önyükleme dağılımını benimser (Tibshirani vd. 1993). Yani,

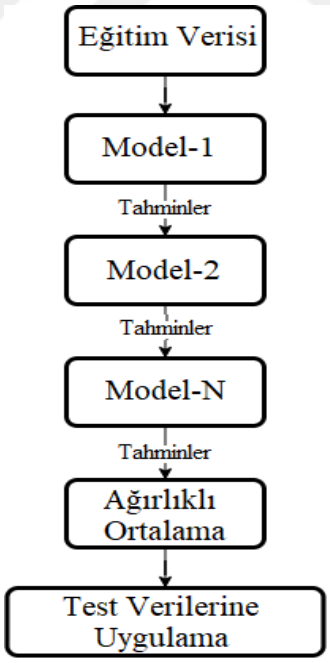
temel öğrenicileri eğitmekte kullanılacak veri alt kümelerini elde etmek için önyükleme örneklemesi uygular. Tüm bireysel modeller paralel çalışacak şekilde inşa edilir. Her bir bireysel model birbirinden farklıdır. Bu yöntemde, önyükleme örneğindeki tüm gözlemler eşit olarak ele alınacaktır. Başka bir deyişle, tüm gözlemler sıfır ağırlıkta eşit olacaktır. İlk adım olarak önyükleme yöntemi yardımıyla, veri kümesi n sayıda örneğe bölünür. Eğitim aşaması tamamlandığında, hedef sonucu tahmin etmek için elde edilen gözlemler tüm modellere iletilir. Her model bir hedef sonucu tahmin edecektir. Nihai tahmin, çoğunluk oylamasına göre seçilecektir. Torbalama yöntemleri hem sınıflandırma hem de regresyon problemleri için kullanılabilir. Çoğunluk oylaması yaklaşımı kullanılarak tüm model çıktılarının ortalaması alındığından aşırı öğrenme ya da ezberlemenin önüne geçebilir.



Şekil 3.2 Torbalama Yöntemi gösterimi

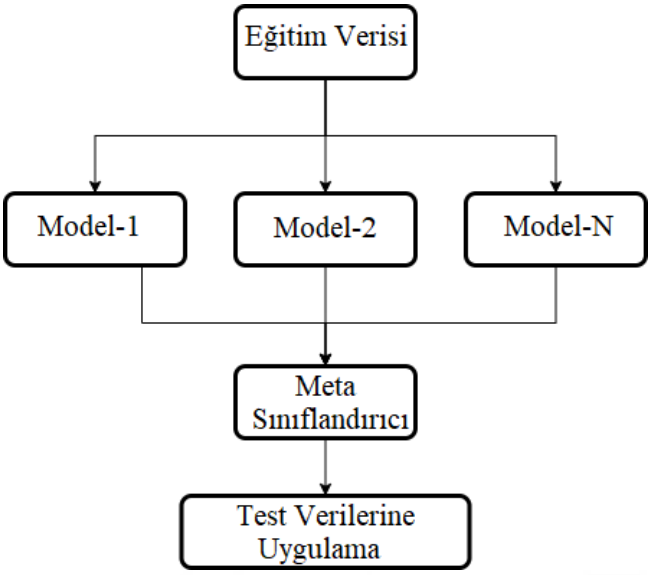
- Artırma: Tüm bireysel modeller sırayla oluşturulur. Bu, modellerin sonuçlarının bir sonraki modele geçtiği anlamına gelir. Torbalamada modeller paralel inşa edilir. Bu nedenle her bir modelin hatasının ne olduğunu bilinmemektedir. Oysa artırma yönteminde ilk model bir kez

çalıştırıldıktan sonra o modelin hataları bilinmektedir. Dolayısıyla, modellerin sonuçları bir sonraki modele geçirilerek hatanın daha da azaltılması amaçlanmaktır. Torbalamadan farklı olarak, önyükleme örneğindeki tüm modeller eşit olarak ele alınmaz. Nihai hedef için tüm modellerden gelen tahminler ağırlıklandırılacaktır. Bu nedenle, nihai tahmin ağırlıklı ortalama olacaktır. Her model çıktısı, önyükleme örnek verileriyle birlikte bir sonraki modele girdi olarak geçirilecektir. Her model sıralı zincirde bir önceki modelin hatalarını azaltmaya çalışır. Çoğu güçlendirme algoritması, daha sonra son bir güçlü sınıflandırıcıya eklenen tekrarlayan öğrenme zayıf sınıflandırıcılardan oluşur. Yanlış sınıflandırılan girdi verileri daha fazla ağırlık kazanırken, doğru sınıflandırılan veriler ağırlık kaybeder. Bu ortam, gelecekteki zayıf öğrencilerin, yanlış sınıflandırılmış önceki zayıf öğrencilere daha kapsamlı bir şekilde odaklanmasını sağlar. Çeşitli artırma algoritmaları arasındaki temel fark, eğitim veri noktalarının ağırlıklandırılmasında kullanılan tekniktir.



Şekil 3.3 Artırma Yöntemi gösterimi

- Farklı makine öğrenmesi algoritmaları kullanılarak oluşturulduğunda heterojen topluluk yöntemi olarak adlandırılır.
    - Yığın: Yeni bir model oluşturmak için çoklu tahmin modellerinden gelen bilgileri birleştirmek için kullanılan bir model birleştirme tekniğidir (Wolpert 1992). Her bir temel modelin en iyi performans gösterdiği yerlerin önemini artırıp kötü performans gösterdiği yerlerin önemini azaltacağı için her bir modelden daha iyi performans göstereceği varsayılır. Bu nedenle, temel modeller önemli ölçüde birbirinden farklı özellikler gösterdiğinde etkili bir yöntem olabilmektedir. Torbalamadan farklı olarak, modeller tipik olarak farklıdır ve aynı veri kümesine uygulanır. Artırmadan farklı olarak, katkıda bulunan modellerden gelen tahminlerin en iyi şekilde nasıl birleştirileceğini öğrenmek için önceki modellerin tahminlerini düzelten bir dizi model yerine tek bir model kullanılır. Bir modelin mimarisi, iki veya daha fazla temel model ve temel modellerin tahminlerini birleştiren bir meta modeli içerir. Temel modeller, eğitim verilerine uygulanan ve tahminleri derlenen modellerdir. Meta model, temel modellerin tahminlerini en iyi şekilde nasıl birleştireceğini öğrenen modeldir. Meta model, örnek dışı veriler üzerinde temel modeller tarafından yapılan tahminler üzerinde eğitilir. Yani, temel modelleri eğitmek için kullanılmayan veriler, temel modellerden beslenir ve tahminler yapılır. Bu tahminler, beklenen çıktılarla birlikte, meta-modele uyması için kullanılan eğitim veri setinin girdi ve çıktı çiftlerini sağlar. Eğitim veri kümesini meta model için hazırlamaya yönelik en yaygın yaklaşım, kat dışı tahminlerin meta model için eğitim veri kümesinin temeli olarak kullanıldığı temel modellerin k kat çapraz doğrulamasıdır.



Şekil 3.4 Yığın Yöntemi gösterimi

Topluluk yöntemleri kullanıldığında daha iyi sonuçların elde edilmesi birçok farklı durumun beraberinde getireceği sonuçlara bağlıdır. Performansta bir gelişme elde etmek, problemin zorluğuna, tahminleri birleştirerek öğrenilecek yeni durumların bulunmasını gerektirecek kadar karmaşık olup olmadığına ve eğitim verileriyle yeterince iyi temsil edilip edilmediğine bağlıdır. Ayrıca, temel modellerin seçimine, tahminlerinde veya hatalarında yeterince doğru sonuçlar vermesine ve modellerin birbiriyle ilişkisinin derecesine de bağlıdır.

Çalışmalarda Python dili ile Scikit-learn kütüphanesindeki çoklu sınıflandırmayı destekleyen makine öğrenmesi algoritmaları ve topluluk yöntemleri kullanılmıştır. Scikit-learn, Python ile yazılmış açık kaynaklı bir makine öğrenimi kütüphanesidir (Kramer 2016). Makine öğrenimi yöntemlerinin Python koduna kolay ve hızlı entegrasyonunu sağlar. Sınıflandırma, regresyon, kovaryans matrisi tahmini, boyut azaltma, veri önişleme gibi geniş bir içerik sunar.

Scikit-learn kütüphanesinde bulunan makine öğrenmesi algoritmalarının veri kümesi üzerinde uygulanabilmesi için yazılım kalite tahmininde kullanılmak üzere seçilen sınıfların sayısal değerler içermesi gerekliliği söz konusudur. Bu nedenle sözel değerler

içeren sütunların sayısallaştırılması için iki farklı yöntem kullanılmıştır. Yapılan tüm işlemler detaylı görseller ile birlikte açıklanmıştır.

Yukarıda belirtilen işlemler ile veri kümeleri, makine öğrenmesi algoritmalarının uygulanması için uygun duruma getirildikten sonra uygulama bölümüne geçilmiştir. Sınıflandırma işleminde kullanılan makine öğrenmesi algoritmaları ve bu algoritmaların birlikte kullanılabilmesine olanak sağlayan topluluk yöntemleri aşağıda özetlenmiştir.

- Lojistik regresyon: Regresyondan ziyade sınıflandırma için kullanılan doğrusal bir modeldir. Özellikler ve sonucun olasılığı arasında bir ilişki bulma amacıyla uygulananır. Sınıf olasılığını tahmin etmek için doğrusal regresyon tarafından üretilen düz çizgiyi kullanmak yerine, lojistik regresyon, sınıf olasılığını tahmin etmek için sigmoidal bir eğri kullanır. Bu yöntemin avantajı, olasılıkların 0 ile 1 arasında sınırlandırılmasıdır. Yalnızca iki olası sonucun olduğu durumda binom veya ikiden fazla olası sonucun olabileceği durumda çok terimli olabilir.
- K-en yakın komşu: Komşu tabanlı sınıflandırma, örneğe dayalı öğrenmenin veya genelleştirmeyen öğrenmenin bir türüdür. Genel bir dahili model oluşturmaya çalışmaz, yalnızca eğitim verilerinin örneklerini depolar. Sınıflandırma, her noktanın en yakın komşularının basit çoğunluk oyu ile hesaplanır. Bir sorgu noktasına, noktanın en yakın komşuları içinde en fazla temsilciye sahip olan veri sınıfı atanır. İki komşunun aynı mesafelere sahip ancak farklı etiketlere sahip olduğu durumda, sonuçlar eğitim verilerinin sıralamasına bağlı olacaktır.
- Karar Ağaçları: Sınıflandırma ve regresyon için kullanılan denetimli bir öğrenme yöntemidir (Breiman vd. 1984). Amaç, veri özelliklerinden çıkarılan basit karar kurallarını öğrenerek hedef değişkenin değerini tahmin eden bir model oluşturmaktır. Aynı ve en yüksek olasılığa sahip birden fazla sınıf olması durumunda, sınıflandırıcı bu sınıflar arasında en düşük indekse sahip sınıfı tahmin edecektir. Böl ve yönet yöntemiyle çalışan bir dizi ağaç yapılı karar testinden oluşur. Yaprak olmayan her düğüm, özellik testindeki farklı değerlerine göre verileri farklı alt kümelere ayıran, bölme olarak da adlandırılan bir özellik testi ile ilişkilendirilir. Her yaprak düğüm, bu düğüme düşen örneklere atanacak bir etiketle ilişkilendirilir.

- Rastgele Orman: Rastgele ormanlarda, topluluktaki her ağaç, eğitim kümesinden değiştirilerek çizilen bir örnekten üretilir (Ho 1995). Ayrıca, bir ağacın inşası sırasında her bir düğümü bölerken, en iyi bölme ya tüm girdi özelliklerinden ya da rastgele bir boyut alt kümesinden bulunur. Bu iki rastgelelik kaynağının amacı, tahminin varyansını azaltmaktır. Bireysel karar ağaçları tipik olarak yüksek varyans sergiler ve fazla uyum gösterme eğilimindedir. Ormanların yapısına eklenen rastgelelik, bir şekilde ayrılmış tahmin hatalarına sahip karar ağaçları verir. Bu tahminlerin bir ortalamasını alarak bazı hatalar ortadan kaldırılabilir. Rastgele ormanlar, çeşitli ağaçları birleştirerek, bazen sapmada hafif bir artış pahasına, azaltılmış bir varyans elde eder. Uygulamada, varyans azalması genellikle önemlidir. Dolayısıyla genel olarak daha iyi bir model verir. Scikit-learn, her sınıflandırıcının tek bir sınıf için oy vermesine izin vermek yerine, olasılık tahminlerinin ortalamasını alarak sınıflandırıcıları birleştirir.
- Torbalama Sınıflandırıcı: Torbalama sınıflandırıcısı, temel sınıflandırıcıların her birini orijinal veri kümesinin rastgele alt kümelerine sığdıran ve daha sonra nihai bir tahmin oluşturmak için bireysel tahminlerin oylama veya ortalama alma yoluyla toplanması sonucuyla oluşan topluluk tahmincisidir
- Çoğunluk-Sert Oylama: Veri seti üzerinde tahmin algoritmaları çalıştırıldıktan sonra en fazla tahmine sahip sınıf etiketi seçilir. Tahmin sayısı eşitse, son tahmin edilen sınıf seçilir.
- Yumuşak Oylama: Çoğunluk oylaması ya da diğer adıyla sert oylamanın aksine, yumuşak oylama, sınıf etiketini tahmin edilen olasılıkların toplamının ortalaması olarak döndürür. Ağırlık parametresi aracılığıyla her sınıflandırıcıya belirli ağırlıklar atanabilir. Ağırlıklar sağlandıktan sonra, her sınıflandırıcı için tahmini sınıf olasılıkları toplanır. Sınıflandırıcı ağırlığı ile çarpılır ve ortalaması alınır. Nihai sınıf etiketi daha sonra en yüksek ortalama olasılığa sahip sınıf etiketinden türetilir.
- AdaBoost: AdaBoost'un temel ilkesi, küçük karar ağaçları gibi rastgele tahminden yalnızca biraz daha iyi olan bir dizi zayıf öğreniciyi verilerin tekrar tekrar değiştirilmiş sürümlerine sığdırmaktır (Freund vd. 1997). Hepsinden elde edilen tahminler daha sonra nihai tahmini üretmek için ağırlıklı çoğunluk oyu veya toplamı ile birleştirilir. Başlangıçta, bu ağırlıkların tümü $w_i = 1/n$ 'dir. Bu

nedenle ilk adım zayıf bir öğrenciyi orijinal veriler üzerinde eğitir. Her ardışık yineleme için, numune ağırlıkları ayrı ayrı değiştirilir. Daha sonra öğrenme algoritması, yeniden ağırlıklandırılmış verilere tekrar uygulanır. Sınıflandırılması zor örneklere daha fazla ağırlık verilirken; doğru sınıflandırılmış örneklere daha az ağırlık verilmeye çalışır. Yinelemeler ilerledikçe, tahmin edilmesi zor örnekler artan bir etki gösterir. Eğitimlerini yanlış tahmin edilen örneklere odaklayan yeni zayıf öğrenciler sırayla eklenir.

- Gradyan Güçlendirme: Bu algoritma sınıfı, aşamalı bir artırma modeli olarak tanımlanmıştır (Friedman 2001). Bunun nedeni, her seferinde yeni bir zayıf öğrenicinin eklenmesi ve modeldeki mevcut zayıf öğrenicilerin dondurulup değişmeden bırakılmasıdır. Regresyon, çok sınıflı sınıflandırma ve daha fazlasını desteklemek için tekniği ikili sınıflandırma problemlerinin ötesine genişleterek keyfi türevlenebilir kayıp fonksiyonlarının kullanılmasına izin vermektedir.
- XGBoost: Son derece verimli, esnek ve taşınabilir olacak şekilde tasarlanmış optimize edilmiş bir dağıtılmış gradyan artırma yöntemidir (Chen vd. 2016). Gradyan artırma çerçevesi altında makine öğrenmesi algoritmaları uygular. Birçok veri bilimi problemini hızlı ve doğru bir şekilde çözen paralel bir ağaç güçlendirme sağlar. Aynı kod, büyük dağıtılmış ortamlarda çalışır ve sorunları milyarlarca örneğin ötesinde çözebilir.
- Yığılmış Genelleme: Yığılmış genelleme, yanlış tahminleri azaltmak için tahmin edicileri birleştirme fikrinden yola çıkan yöntemdir. Her bir tahmin edicinin tahminleri bir araya toplanır ve tahmini hesaplamak için nihai bir tahmin ediciye girdi olarak velirir. Son tahmin edici de çapraz doğrulama yoluyla eğitilerek tahmin işlemi gerçekleştirilir.

4. BULGULAR VE TARTIŞMA

Bu bölümde veri kümelerindeki özniteliklerin birbirleriyle ilişkileri korelasyon matrisi ve özellik önemi kullanılarak gösterilmiştir. Bu doğrultuda daha önce açıklamaları yapılmış olan makine öğrenmesi algoritmaları ve topluluk yöntemleri uygulanıp elde edilen sonuçlarla birleştirilerek ortaya çıkan bulgular ifade edilmiştir.

4.1 Korelasyon Matrisi

İki değişken arasındaki ilişki korelasyon olarak kabul edilir. Korelasyon matrisi, özniteliklerin kendi aralarındaki korelasyon değerlerini verir. Korelasyon matrisi, bir değişkenin başka bir değişkeni ne kadar etkilediğini göstermek için kullanılır. Python'da, bir kod satırı ile korelasyon matrisi hesaplanabilir. Ardından Seaborn kitaplığı kullanılarak ısı haritası olarak görüntülenebilecek bir şekil oluşturmak için Matplotlib kitaplığına iletilebilir (Sial vd. 2021).

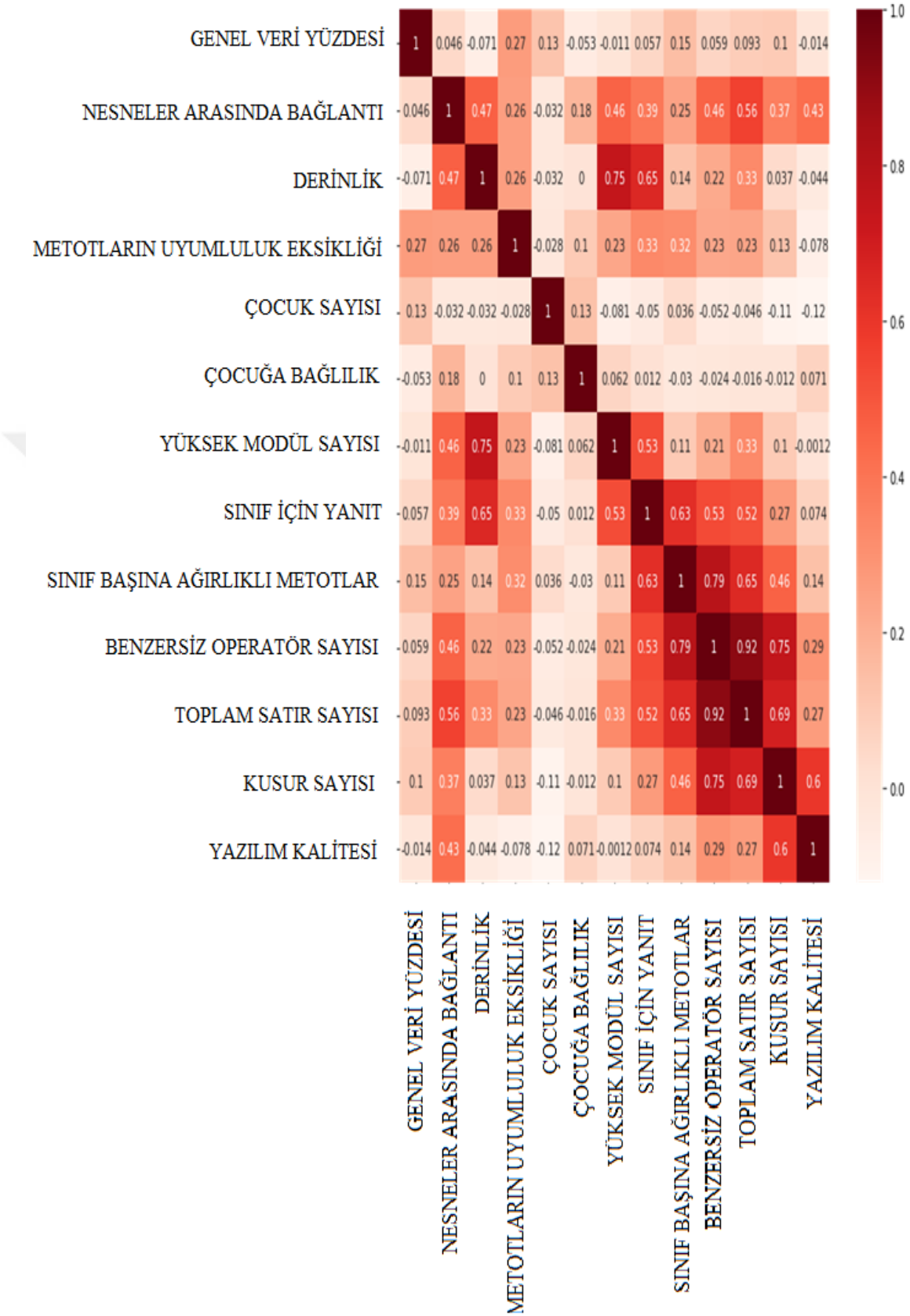
Gösterimi yapılan matrisler içerisinde hesaplanmış matematiksel değerler, ilişkinin kuvvetli ya da zayıf olduğunu belirten ifadelerdir. Ayrıca ısı haritasında renk koyulaştıkça iki özelliğin arasındaki ilişkinin kuvvetli olduğu, renk soluklaştıkça ilişkinin zayıfladığı ifade edilebilmektedir.

Tüm veri kümeleri için ortak olarak kusur sayısının yazılım kalitesi ile çok yüksek bir korelasyon değerine sahip olduğu gözlemlenmiştir.

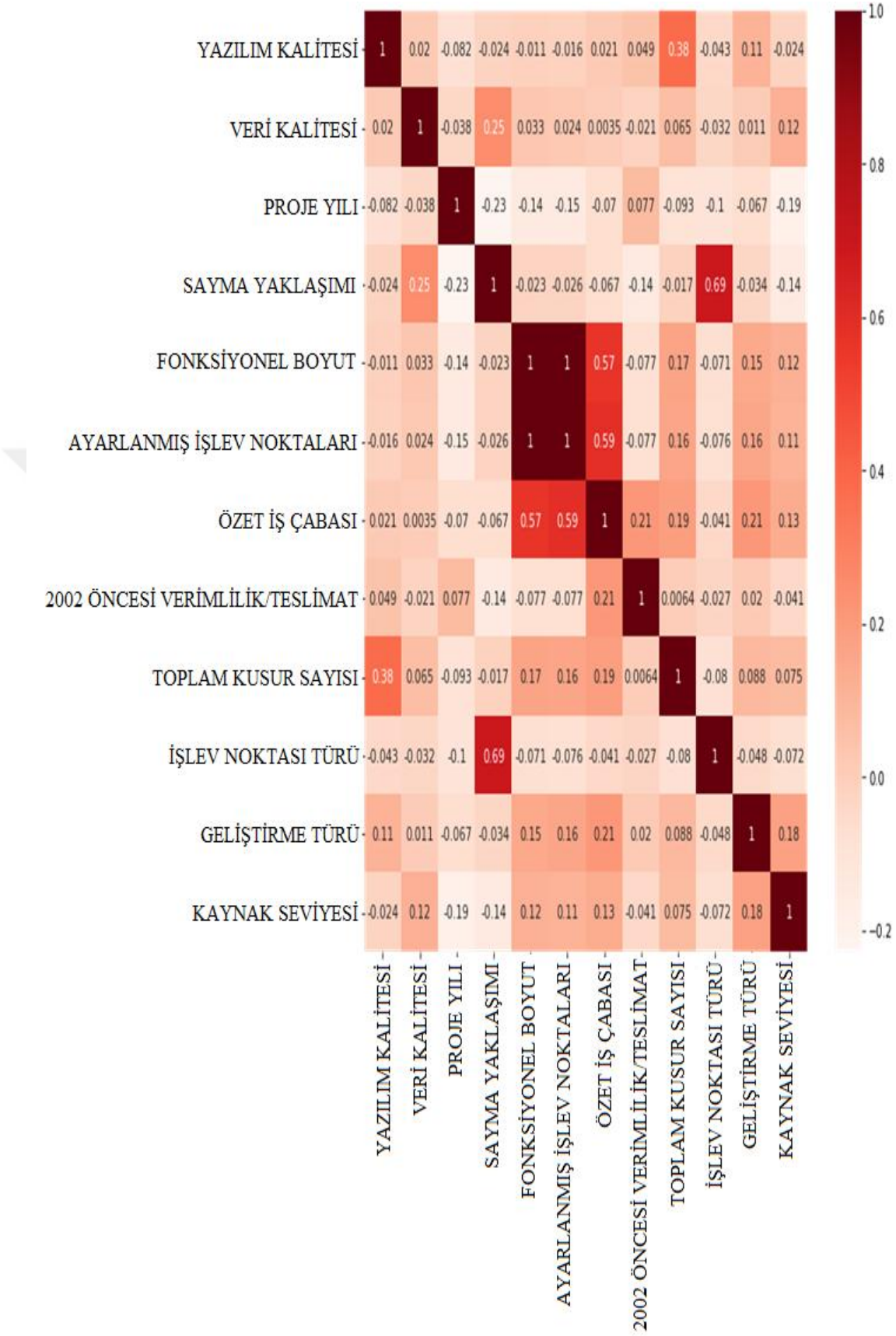
- PROMISE veri kümesi için kusur sayısından sonra en yüksek korelasyon değerlerine sahip olan öznitelikler nesneler arasındaki bağlantı, benzersiz operatörlerin sayısı ve toplam kod satırı olmuştur.
- EBSPM ve ISBSG veri kümeleri için ilk yöntem kullanılarak uygulanan sözel verilerin sayısal verilere dönüştürülmesi işlemi sonucunda:
    - Yazılımın geliştirme süresi ve maliyetinin yazılım kalitesini önemli oranda etkilemesi beklenir. Nitekim EBSPM veri kümesi için kusur sayısından sonra ikinci ve üçüncü en yüksek korelasyon değerlerine sahip olan özniteliklerin süre ve maliyet olduğu korelasyon matrisinde görülmektedir.

- ISBSG veri kümesi için kusur sayısından sonra en yüksek korelasyon değerlerine sahip olan öznitelik ise yazılım geliştirme türü olmuştur.

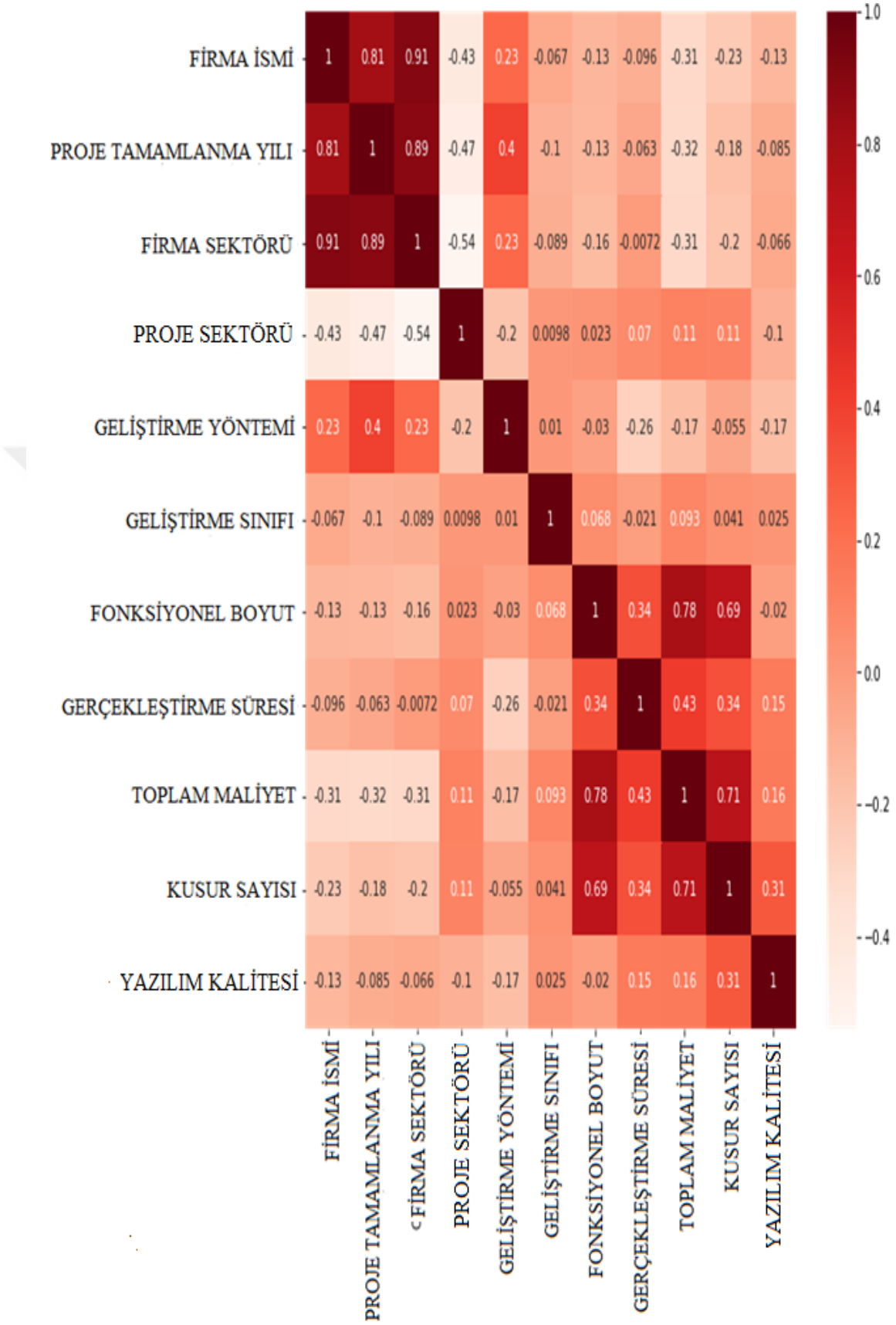
- EBSPM ve ISBSG veri kümeleri için ikinci yöntem kullanılarak uygulanan sözel verilerin sayısal verilere dönüştürülmesi işlemi sonucunda çok sayıda ek sütun oluşturulması korelasyon matrislerinde ve özellik önemi tablolarında değişikliklere yol açmıştır. Ancak her iki veri kümesi için kusur sayısı ile yazılım kalitesi arasında en yüksek korelasyon değerine sahip olma durumu devam etmektedir.
  - EBSPM veri kümesi için kusur sayısından sonra en yüksek korelasyon değerlerine sahip olan öznitelikler ise yazılım geliştirme sürecindeki zaman, maliyet, şirket sektörü ve yazılım geliştirme türü değerleri olmuştur.
  - ISBSG veri kümesi için kusur sayısından sonra en yüksek korelasyon değerlerine sahip olan öznitelikler yazılım geliştirme türü, veri kalitesi ve çaba değerleri olmuştur.



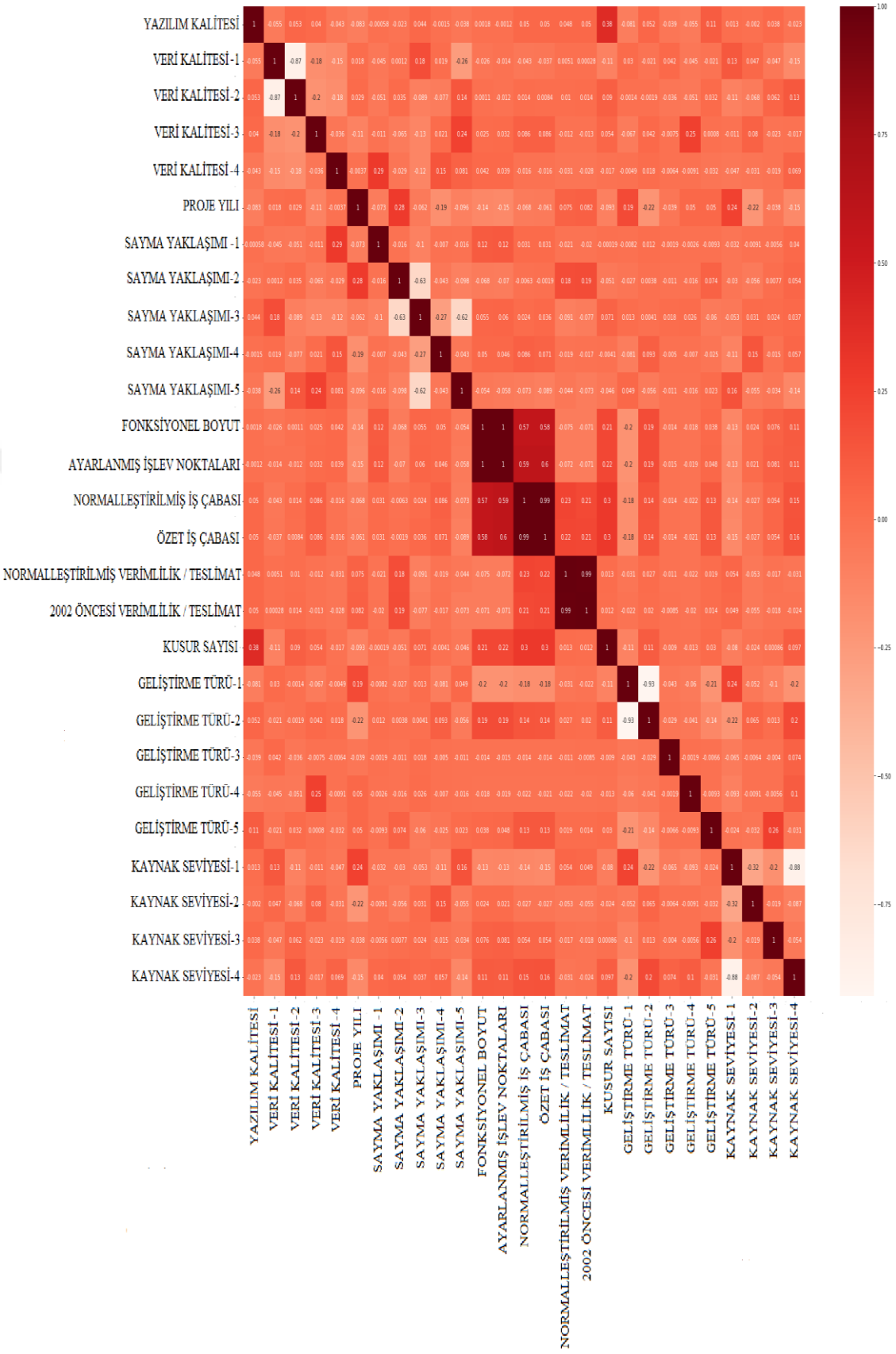
Şekil 4.1 PROMISE seçilen özniteliklerin korelasyon matrisi



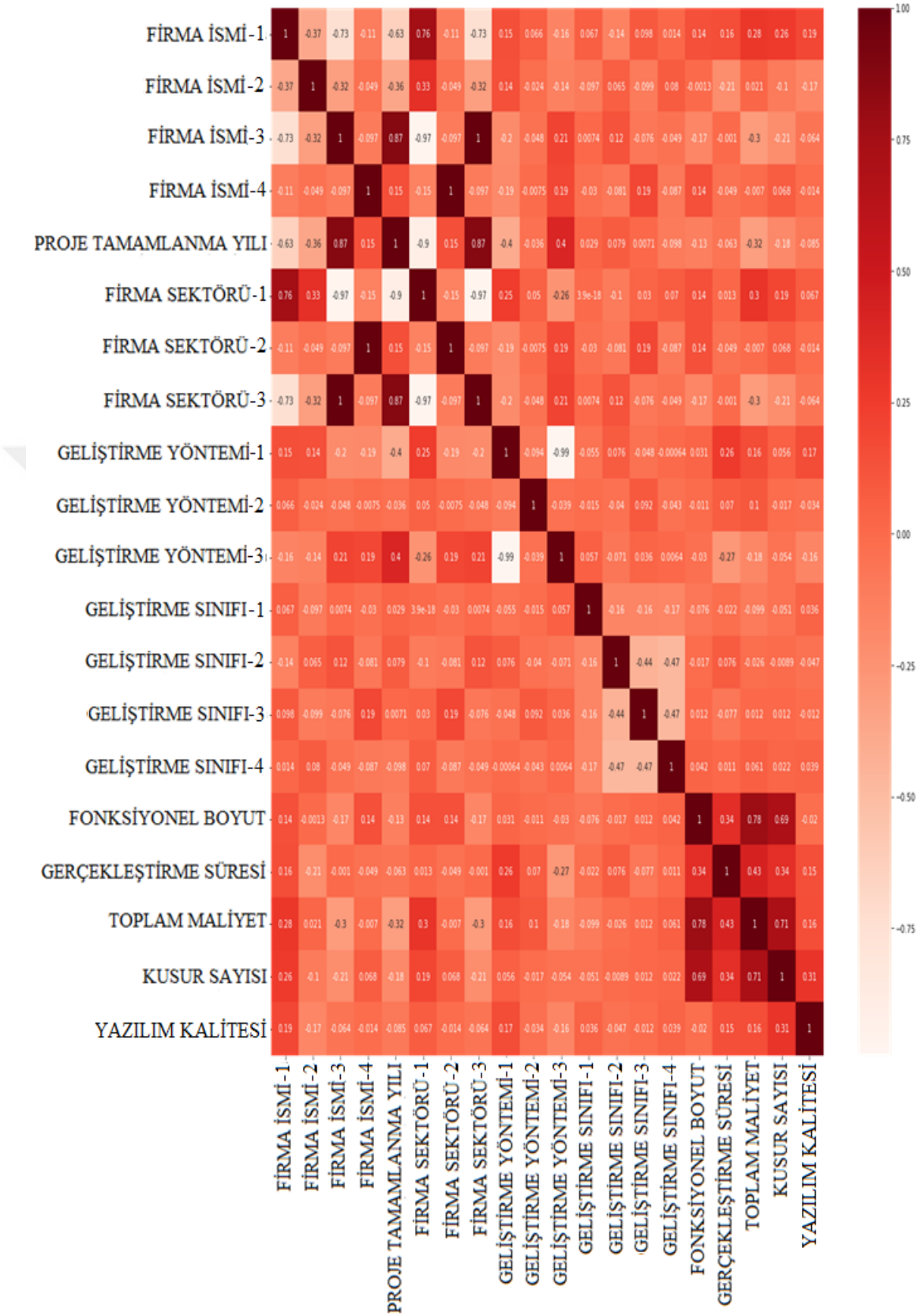
Şekil 4.2 ISBSG ilk çalışmada seçilen özniteliklerin korelasyon matrisi



Şekil 4.3 EBSPM ilk çalışmada seçilen özniteliklerin korelasyon matrisi



Şekil 4.4 ISBSG ikinci çalışmada seçilen özniteliklerin korelasyon matrisi



Şekil 4.5 EBSPM ikinci çalışmada seçilen özniteliklerin korelasyon matrisi

4.2 Özellik Önemi

Özellik önemlerini oluşturmada kullanılan yöntemden bağımsız olarak aşağıdaki ifadeleri bu kavramı tanımlamak için kullanabiliriz (Hartshorn 2016).

- Bir özelliğin diğer özelliklere göre ne kadar önemli olduğunu ifade etmek için kullanılır.
- Genellikle normalize edilmiş değerler olarak rapor edilir.
- Ne kadar çok özellik kullanılırsa, herhangi bir özellik o kadar az önemli olacaktır.
- Özelliklerin birbiriyle ilişkisi ne kadar yüksek düzeyde ise önem dereceleri o kadar fazla bölünecektir.

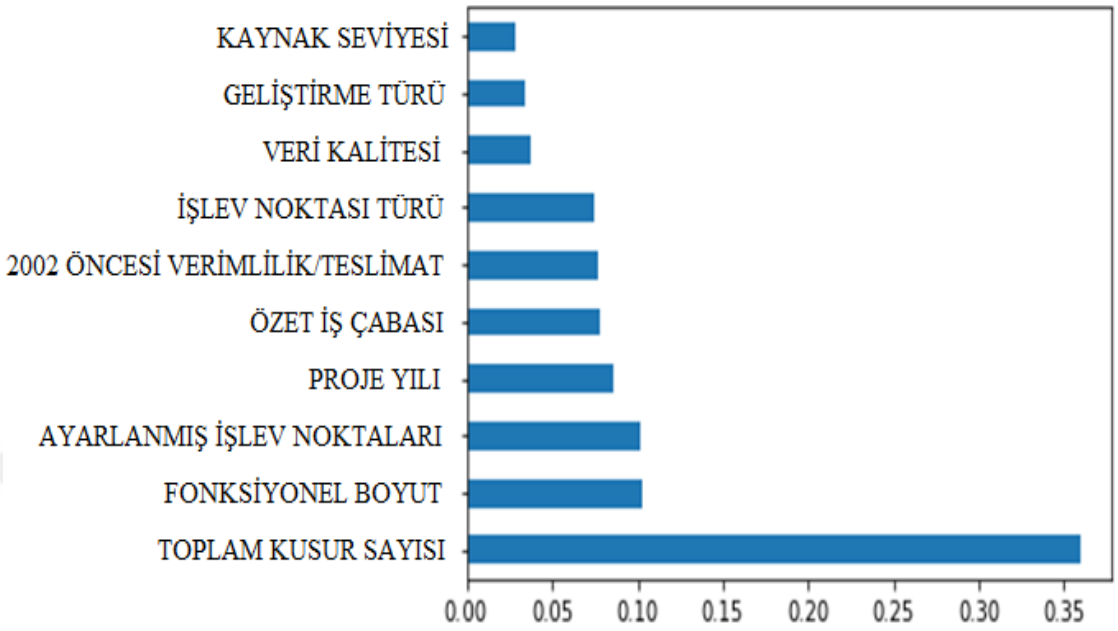
Diğer sınıfların hedef sınıf ile ilişkisini gösterebilmek için bir topluluk yöntemi ile özellik önemi değerleri hesaplanıp Matplotlib kütüphanesi yardımıyla görselleri oluşturulmuştur.

PROMISE veri kümesi için benzersiz operatörlerin sayısı en çok öneme sahip olan özellik iken onu sırasıyla toplam satır sayısı, derinlik ve sınıf başına ağırlıklı metotlar takip etmektedir.

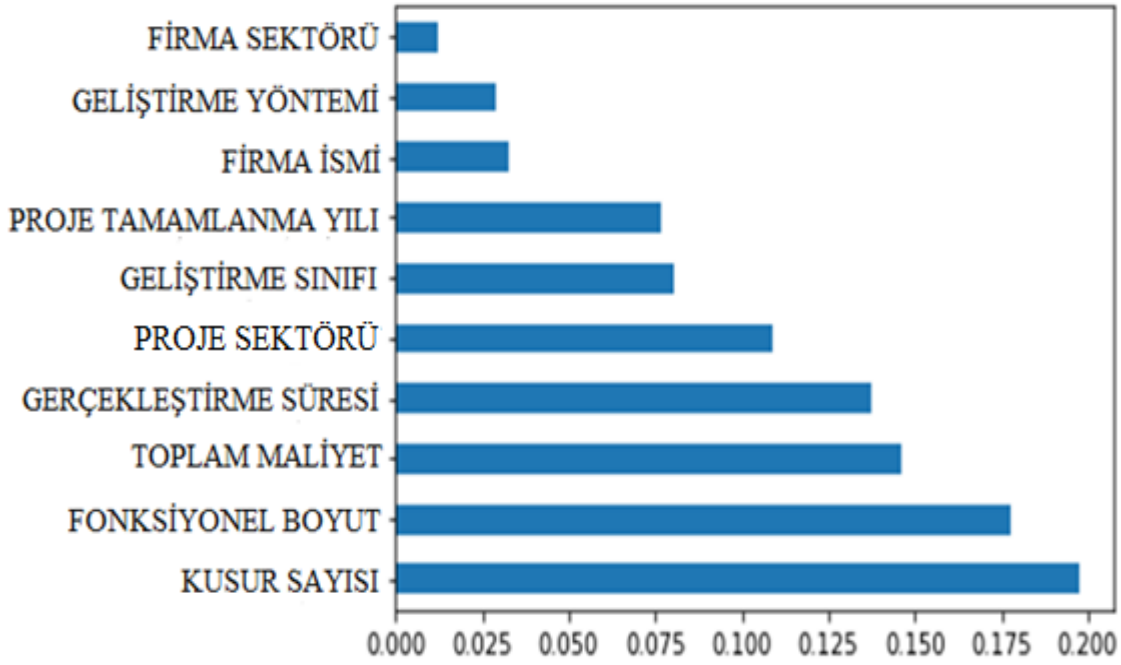
EBSPM ve ISBSG veri kümeleri için uygulanan ilk yöntem sonucunda kalite tahmininde kullanılan özelliklerden en çok etkili olan faktörün kusur sayısı olduğu açıkça görülmektedir. Sonrasında gelen fonksiyonel boyut, yazılım maliyeti ve yazılımın üretildiği sürenin ay cinsinden değeri de yazılım kalitesi tahmininde önem arz eden özellikler olarak ön plana çıkmaktadır.

İkinci yöntem sonucunda elde edilen özellik önemi tablosunda, korelasyon matrisinde olduğu gibi daha farklı özelliklerin önem kazandığı görülmektedir. ISBSG veri kümesi için fonksiyonel boyut en yüksek öneme sahipken sonrasında gelenler ise yazılım geliştirme yöntemleri ve kusur sayısı olmuştur. EBSPM veri kümesi için ise yazılım yılı

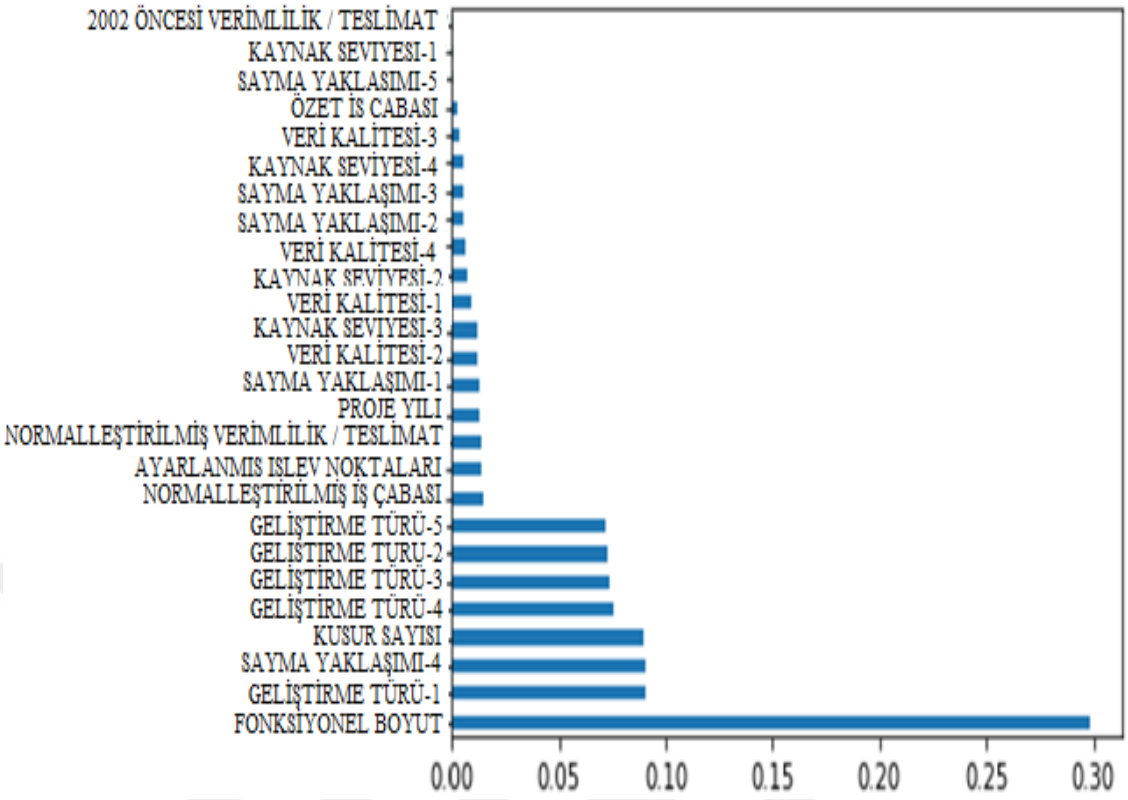
en yüksek öneme sahipken sonrasında gelenler ise şirket ismi, şirket sektörü ve yazılım geliştirme yöntemi olmuştur.



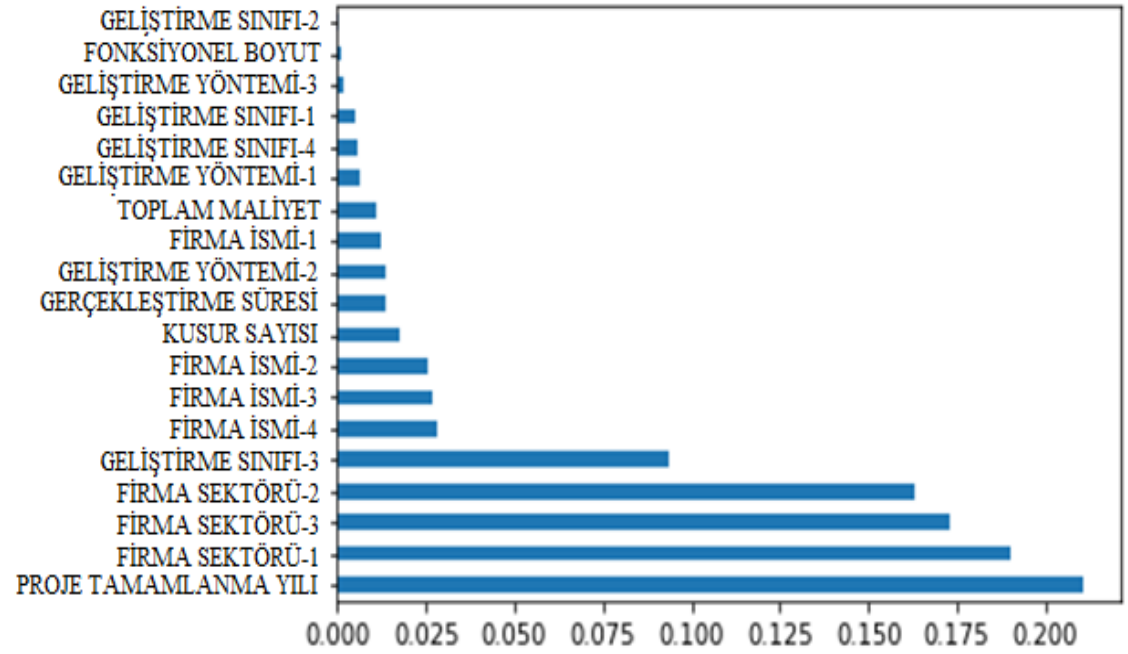
Şekil 4.6 ISBSG ilk çalışma özellik önemi



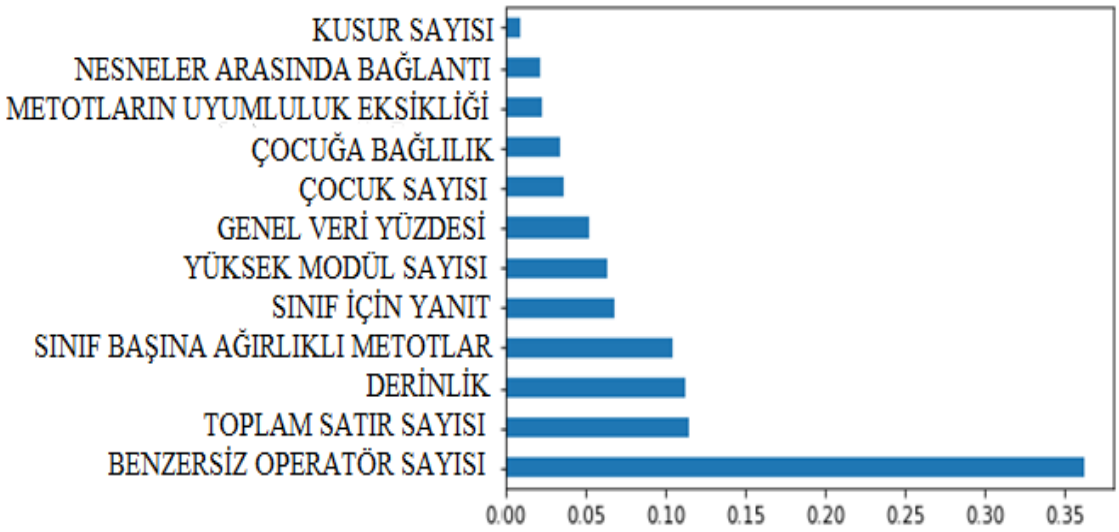
Şekil 4.7 EBSPM ilk çalışma özellik önemi



Şekil 4.8 ISBSG ikinci çalışma özellik önemi



Şekil 4.9 EBSPM ikinci çalışma özellik önemi



Şekil 4.10 PROMISE özellik önemi

4.3 Uygulama

Veri önişleme adımından sonra elde edilen veri kümeleri fonksiyon yardımıyla okunmuştur. Sonraki adımda ise veri kümelerinin içerikleri bir değişkene aktarılmıştır. Daha sonra hedef sınıf ile tahmin sınıfını ayırmak için iki değişken oluşturulup hedef sınıf olan yazılım kalitesi bir değişkene, seçilen diğer öznitelikler de başka bir değişkene atanmıştır.

Modelleri uygulamak için Python dili ile scikit-learn kütüphanesindeki çok sınıflı sınıflandırma algoritmaları ve bunların birlikte çalıştırılabildiği topluluk yöntemleri kullanılmıştır. Eğitim ve test verileri %33-%67 oranlarında ayrılmıştır.

Makine öğrenmesi algoritmaları ve topluluk yöntemlerinin uygulanması sonucunda elde edilen doğruluk oranları her veri kümesi için ilgili çizelge üzerinde gösterilmiştir. Veri kümelerindeki tahmin için seçilmiş sütunların korelasyon değerleri arasındaki fark doğruluk oranlarının farklılık göstermesine neden olmuştur. Ayrıca veri kümeleri arasındaki proje sayılarındaki farklılık da doğrulukları etkileyen önemli bir faktördür.

Sonuçlar çizelgeler üzerine aktarılırken topluluk yöntemleri normal yazılırken; tek başına kullanılan makine öğrenmesi yöntemleri eğik yazıyla gösterilmiştir. PROMISE ve ISBSG veri kümeleri için topluluk yöntemleri daha iyi sonuç verirken EBSPM veri kümesi için ise tek başına kullanılmış olan bir makine öğrenmesi yöntemi olan Lojistik Regresyon daha iyi sonuç vermiştir.

Çizelge 4.1 PROMISE doğruluk yüzdeleri

Yöntem	**Doğruluk Yüzdesi**
AdaBoost	72.97%
Stacking	78.38%
Logistic Regression	83.78%
Random Forest	86.49%
Voting (Soft)	89.19%
XGBoost	89.19%
Decision Tree	91.89%
Voting (Hard)	91.89%
Gradient Boosting	94.59%
Bagging	94.59%

Çizelge 4.2 ISBSG doğruluk yüzdeleri

Yöntem	**Veri Önişleme İlk Yöntem**	**Veri Önişleme İkinci Yöntem**
BernoulliNB	69.92%	75.26%
AdaBoost	—	78.95%
Random Forest	89.43%	91.05%
Decision Tree	89.43%	94.74%
Gradient Boosting	—	94.74%
XGBoost	92.28%	94.74%
Bagging	92.28%	95.26%
Voting (Hard)	—	95.78%
Stacking	—	96.31%
Voting (Soft)	—	96.31%

Çizelge 4.3 EBSPM doğruluk yüzdeleri

Yöntem	**Veri Önişleme İlk Yöntem**	**Veri Önişleme İkinci Yöntem**
Random Forest	66.67%	56.67%
AdaBoost	—	57.77%
XGB	65.56%	64.44%
Gradient Boosting	—	70.00%
Bagging	67.78%	71.11%
Decision Tree	67.78%	71.11%
Stacking	—	74.44%
Voting (Soft)	—	74.44%
Voting (Hard)	—	85.55%
Logistic Regression	92.22%	96.67%

5. SONUÇ

Bu tez çalışmasının amacı makine öğrenmesi algoritmaları kullanılarak yazılımın kalitesinin tahmin edilmesidir. Yazılım kalitesi hesaplanırken önceki çalışmalarda tek başına kullanılan kusur sayısı yerine yazılım büyüklüğünün de hesaplamaya katıldığı, kusur yoğunluğu değeri kullanılmıştır. Bununla bitlikte yazılım kalite sınıflandırılmasında iki sınıf yerine dört sınıf kullanılarak problem çoklu sınıf sınıflandırma problemine dönüştürülmüştür. Bu işlem tahmin işlemini zorlaştırsa da sınıflar arasındaki aralıkları daha anlamlı hale getirerek doğruluk oranlarının artmasına katkı sağlamıştır.

Toplamda üç veri kümesi üzerinde çalışılmıştır. Bunlardan iki tanesi sözel kategorik veriler içermektedir. Dolayısıyla bu iki veri kümesine de makine öğrenmesi yöntemlerini uygulayabilmek için yapılması gereken sözel değerlerin sayısal değerlerle değiştirilmesi işlemi iki farklı şekilde gerçekleştirilmiştir. Bu işlemler neticesinde doğruluk oranları da farklı şekilde etkilenmiştir.

- İlk çalışmada yalnızca bir yeni sütun oluşturulup 1 ile başlanarak her görülen farklı değer için yeni bir rakam verilmiştir. Yeni oluşturulan ve sayısal değerlerin bulunduğu sütun tahmin işlemine dahil edilmiştir.
- İkinci çalışmada ise ilgili sütunda bulunan her bir farklı değer için yeni bir sütun oluşturularak eşleşme sağlanan sütuna 1 diğer sütunlara 0 değeri atanmıştır. Bu işlem sonucunda oluşturulan tüm yeni sütunlar tahmin işlemine dahil edilmiştir.

Sınıflandırma işlemleri Scikit-learn kütüphanesi kullanılarak gerçekleştirilmiştir. Kütüphane içerisinde bulunan çok sınıflı sınıflandırmayı destekleyen algoritmalarla çalışmalar yapılmıştır. Doğruluk oranlarının arttırılabilmesi için birden fazla algoritmanın beraber kullanılabilmesine olanak sağlayan topluluk yöntemleri kullanılmıştır.

- Yapılan ilk çalışma sonucunda EBSPM veri kümesinde Lojistik Regresyon algoritmasıyla %92.22 ve ISBSG veri kümesinde XGBoost ve Torbalama algoritmaları ile %92.28 doğruluk oranları elde edilmiştir. Daha önceki

doğrudan karşılaştırılabilir çalışmalara kıyasla, kabul edilebilir düzeyde çok sınıflı kalite tahmini elde edilmiştir.

- Yapılan ikinci çalışma sonucunda ise EBSPM veri kümesinde Lojistik Regresyon algoritmasıyla %96.67 ve ISBSG veri kümesinde Yumuşak Oylama algoritmasıyla %96.31 doğruluk oranları elde edilmiştir. Verinin farklı şekilde sayısal değerlere dönüştürülmesi ve kullanılan farklı topluluk yöntemleri ilk çalışmaya göre doğruluk oranlarının yükselmesine katkı sağlamıştır.
- PROMISE veri kümesiyle yapılan çalışmada ise Torbalama ve Gradyan Güçlendirme algoritmaları %94.59 doğruluk oranı elde etmiştir.

Elde edilen sonuçlar doğrultusunda, yazılım kalite tahmininde topluluk yöntemlerinin kullanılmasının daha doğru tahminler ürettiği ortaya çıkmıştır. Yürütülen bu çalışma ile makine öğrenmesi tabanlı topluluk yöntemleri algoritmaları kullanılarak yapılan yazılım kalite tahmini çalışmasının, literatürde yapılmış olan diğer çalışmalara göre daha yüksek doğruluk oranları verdiği ispatlanmıştır.

KAYNAKLAR

Bowes, D. Hall, T. ve Petrić, J. 2018. Software defect prediction: do different classifiers find the same defects?. Software Quality Journal, 26(2), 525-552.

Breiman, L. 1996. Bagging predictors. Machine learning, 24(2), 123-140.

Breiman, L. Friedman, J. Stone, C. J. ve Olshen, R. A. 1984. Classification and regression trees. CRC press.

Ceran, A. A. ve Tanrıöver, Ö. Ö. 2020. An experimental study for software quality prediction with machine learning methods. In 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 1-4, IEEE.

Chaitra, P. ve Kumar, D. R. S. 2018. A review of multi-class classification algorithms. Int. J. Pure Appl. Math, 118(14), 17-26.

Cheikhi, L. ve Abran, A. 2013. PROMISE and ISBSG Software Engineering data repositories: A survey. In 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, 17-24, IEEE.

Chen, T. ve Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 785-794.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

Gao, K., Khoshgoftaar, T. M., ve Wald, R. 2014. Combining feature selection and ensemble learning for software quality estimation. In The Twenty-Seventh International Flairs Conference.

Hartshorn, S. 2016. Machine learning with random forests and decision trees. Kindle edition.

Hansen, L. K. ve Salamon, P. 1990. Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence, 12(10), 993-1001.

Ho, T. K. 1995. Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). IEEE.

Huijgens, H. 2016. Evidence-based software portfolio management: a tool description and evaluation. In Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, 1-5.

Kitchenham, B. ve Pfleeger, S. L. 1996. Software quality: the elusive target [special issues section]. IEEE software, 13(1), 12-21.

Koru, A. G. ve Liu, H. 2005. An investigation of the effect of module size on defect prediction using static measures. In Proceedings of the 2005 workshop on Predictor models in software engineering, 1-5.

Kramer, O. 2016. Scikit-learn. In Machine learning for evolution strategies, 45-53, Springer, Cham.

Kumbakonam, M. ve Shafi, R. M. 2021. Application Measurement and Software Quality Testing Using Machine Learning Performance Techniques. In 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT), 372-376, IEEE.

Kumbakonam, M. 2021. A Comparison Experiment On Software Quality Testing With Machine Learning Using Different Algorithms. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(12), 48-61.

Kurtel, K. ve Şaban, E. R. E. N. 2011. Yazılım Mimarisinin Kalite Gereksinimleri: Yazılım Güvenilirliği. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 4(1).

Lokan, C. Wright, T. Hill, P. ve Stringer, M. 2001. Organizational benchmarking using the ISBSG data repository. IEEE Software, 18(5), 26-32.

Sagi, O. ve Rokach, L. 2018. Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), e1249.

Schapire, R. E. 1990. The strength of weak learnability. Machine learning, 5(2), 197-227.

Freund, Y. ve Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1), 119-139.

Seliya, N. Khoshgoftaar, T. M. ve Zhong, S. 2004. Semi-supervised learning for software quality estimation. In 16th IEEE International Conference on Tools with Artificial Intelligence, 183-190, IEEE.

Sial, A. H. Rashdi, S. Y. S. ve Khan, A. H. 2021. Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python. International Journal, 10(1).

Reddivari, S. ve Raman, J. 2019. Software quality prediction: an investigation based on machine learning. In 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), 115-122, IEEE.

Thwin, M. M. T. ve Quah, T. S. 2005. Application of neural networks for software quality prediction using object-oriented metrics. Journal of systems and software, 76(2), 147-156.

Tibshirani, R. J. ve Efron, B. 1993. An introduction to the bootstrap. Monographs on statistics and applied probability, 57, 1-436.

Vijay, T. J. Chand, M. G., ve Done, H. 2017. Software quality metrics in quality assurance to study the impact of external factors related to time. International Journal of Advanced Research in Computer Science and Software Engineering Research Paper, 7(1).

Wagner, S. 2010. A Bayesian network approach to assess and predict software quality using activity-based quality models. Information and Software Technology, 52(11), 1230-1241.

Wang, X. Zhang, Y. Zhang, L. ve Shi, Y. 2010. A knowledge discovery case study of software quality prediction: Isbsg database. In 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Vol. 3, 219-222, IEEE.

Wang, X. Zhang, Y. Zhang, L. ve Shi, Y. 2010. A knowledge discovery case study of software quality prediction: Isbsg database. In 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Vol. 3, 219-222, IEEE.

Wolpert, D. H. 1992. Stacked generalization. Neural networks, 5(2), 241-259.

Xing, F. Guo, P. ve Lyu, M. R. 2005. A novel method for early software quality prediction based on support vector machine. In 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05), IEEE.