BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİMDALI
BİLGİSAYAR MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS PROGRAMI

MAKİNE ÖĞRENMESİ TEKNİKLERİYLE YAZILIM UYUM METRİKLERİNİN TAHMİNİ

HAZIRLAYAN

ELİF NUR HANER KIRĞIL

YÜKSEK LİSANS TEZİ

ANKARA - 2022

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİMDALI
BİLGİSAYAR MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS PROGRAMI

MAKİNE ÖĞRENMESİ TEKNİKLERİYLE YAZILIM UYUM METRİKLERİNİN TAHMİNİ

HAZIRLAYAN

ELİF NUR HANER KIRĞIL

YÜKSEK LİSANS TEZİ

TEZ DANIŞMANI

DR. ÖĞR. ÜYESİ TÜLİN ERÇELEBİ AYYILDIZ

ANKARA - 2022

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı çerçevesinde Elif Nur HANER KIRĞIL tarafından hazırlanan bu çalışma, aşağıdaki jüri tarafından Yüksek Lisans Tezi olarak kabul edilmiştir.

TEZ SAVUNMA TARİHİ: 11/ 08/2022

Tez Adı: Makine Öğrenmesi Teknikleriyle Yazılım Uyum Metriklerinin Tahmini

Tez Jüri Üyeleri **İmza**

Dr. Öğr. Üyesi, Emre SÜMER, Başkent Üniversitesi

Dr. Öğr. Üyesi, Tülin ERÇELEBİ AYYILDIZ, Başkent Üniversitesi

Dr. Öğr. Üyesi, Damla TOPALLI, Atılım Üniversitesi

ONAY

Prof. Dr. Faruk ELALDI
Fen Bilimleri Enstitüsü Müdürü

Tarih: … / … / 2022

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih …/08/2022

Öğrencinin Adı, Soyadı: Elif Nur Haner Kırğıl
Öğrencinin Numarası: 21920301
Anabilim Dalı: Bilgisayar Mühendisliği
Programı: Bilgisayar Mühendisliği
Danışmanın Unvanı/Adı, Soyadı: Dr. Öğr. Üyesi Tülin Erçelebi Ayyıldız
Tez Başlığı: Makine Öğrenmesi Teknikleriyle Yazılım Uyum Metriklerinin Tahmini

Yukarıda başlığı belirtilen Yüksek Lisans tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam ….. sayfalık kısmına ilişkin, …./08/2022 tarihinde tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % ?'dür.

Uygulanan filtrelemeler:
1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:

ONAY
…. / 08 /2022
Dr. Öğr. Üyesi Tülin ERÇELEBİ AYYILDIZ

TEŞEKKÜR

Çalışmanın başından sonuna kadar değerli fikirlerini benimle paylaşan, her sorunumda sabırla ve anlayışla çözüm bulan, karşılaştığım tüm güçlüklerde bana yol gösteren tez danışmanım değerli hocam Dr. Öğretim Üyesi Tülin ERÇELEBİ AYYILDIZ'a,

Çalışmam boyunca beni her zaman destekleyen, yanımda olan çok değerli eşim Mustafa Erdem KIRĞIL'a

Hayatım boyunca desteklerini üzerimde hissettiğim, yaptığım her işte arkamda olan babaannem ve aileme,

Destek ve yardımlarından dolayı Dr. Çağatay Berke ERDAŞ, Dr. Didem ÖLÇER ve Araş. Gör. Begüm ERKAL'a

Bu süreçte beni yalnız bırakmadıkları için teşekkürlerimi sunuyorum.

ÖZET

Elif Nur HANER KIRĞIL

MAKİNE ÖĞRENMESİ TEKNİKLERİYLE YAZILIM UYUM METRİKLERİNİN TAHMİNİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

2022

Günümüzde yazılıma olan ihtiyaç arttıkça, oluşturulan yazılımların kalitesini ölçmek de gerekli hale gelmiştir. Yazılım kalitesini ölçmek için literatürde çeşitli metrikler bulunmaktadır. Bu metriklerin bazıları sınıf bazında ölçüm yapmaktadır. Sınıflar nesne yönelimli programlamanın temel birimleridir. Sınıflarda bulunan metotların ve özniteliklerin birbiri ile uyumunu ölçen uyum metriği yazılım kalitesine doğrudan etki etmektedir. Uyumun temel amacı her sınıfın tek bir amaca hizmet etmesi gerektiği kuralını sağlamaktır. Sınıflar gerçekleştirim aşamasında uyuma dikkat edilerek oluşturulduğunda, bakım onarım aşamasında her amaç için tek bir sınıfa bakılacağından zaman ve maliyet açısından fayda sağlanacaktır. Literatürde yazılımın uyumunu ölçen birçok metrik ve araç bulunmaktadır. Her metrik farklı yöntemlerle ve farklı konuları dikkate alarak uyumu hesaplamaktadır. Uyumu hesaplamak için her ne kadar araç kullanımı yaygın olsa da araç kullanmak zaman ve maliyet gerektiren bir durumdur. Ayrıca araçların deneme versiyonları ile istenilen tüm metriklere her durumda erişilemeyebilir. Bildiğimiz kadarıyla şu ana kadar, yazılım uyum metriklerini makine öğrenmesi teknikleri ile tahmin etmeye çalışan herhangi bir çalışma bulunmamaktadır. Bu çalışmada Lack of Cohesion in Methods (LCOM2), Tight Class Cohesion (TCC), Loose Class Cohesion (LCC) ve Low Level Design Class Cohesion (LSCC) uyum metrikleri Rastgele Orman (Random Forest), REPTree, K En Yakın Komşu (K Nearest Neighbor – KNN), Doğrusal Regresyon (Linear Regression), Çok Katmanlı Algılayıcı (Multilayer Perceptron – MLP) ve Destek Vektör Makinesi (Support Vector Machine – SVM) yöntemleri kullanılarak tahmin edilmeye çalışılmıştır. Bu sayede uyum metriği daha hızlı ve kolay bir şekilde elde edilebilmektedir. LCOM2 metriği 0 ve 1 arasında normalize edilmediği için LCOM2 metriğine ait veri kümesine normalizasyon işlemi ve aykırı/uç değer analizi yapılmıştır. Bu işlemlerden sonra elde edilen analiz sonuçlarının tamamı verilmiştir. Alınan sonuçlara göre en iyi performanslar, LCOM2 metriği için uç değer analizinden sonra 5,080

hata değeri, normalizasyon işleminden sonra 0,079 hata değeri KNN algoritması kullanılarak, TCC için 0,231 hata değeri KNN algoritması kullanılarak, LCC için 0,259 hata değeri REPTree algoritması kullanılarak ve LSCC için 0,149 hata değeri REPTree algoritması ile elde edilmiştir.

ANAHTAR KELİMELER: Yazılım Kalitesi, Yazılım Uyumu, Makine Öğrenmesi, LCOM, Bakım Yapılabilirlik

ABSTRACT

Elif Nur HANER KIRĞIL

PREDICTING SOFTWARE COHESON METRICS WITH MACHINE LEARNING TECHNIQUES

Başkent University Institute of Science and Engineering

Department of Computer Engineering

2022

Today, as the need for software increases, it has become necessary to measure the quality of these software. There are various metrics in the literature to measure software quality. Some of these metrics measures on a class basis. Classes are the basic units of object oriented programming. The cohesion metrics, which measures the compatibility of the methods and attributes in the classes with each other, directly affects the software quality. The main purpose of cohesion is to provide the rule that every class should serve a single purpose. When classes are created by paying attenion to cohesion, a single class will be traced for each purpose during the maintenance phase, so it is profitable in terms of time and cost. There are many metrics and tools in the literature that measure cohesion. Each metric calculates cohesion with different methods and considering different issues. Although it is common to use tools to calculate the cohesion, using a tool requires time and cost . In addition, with the trial versions of the tools, all desired metrics may not be accessible in all cases. As far as we know, there is no study that tries to predict cohesion metrics with machine lerning techniques so far. In this study, Lack of Cohesion in Methods (LCOM2), Tight Class Cohesion (TCC), Loose Class Cohesion (LCC) and Low Level Design Class Cohesion (LSCC) metrics were tried to be estimated using Random Forest, REPTree, K Nearest Neighbor – KNN, Linear Regression, Multilayer Perceptron (MLP) and Support Vector Machine (SVM). In this way, the cohesion metric is obtained more quickly and easily. Since the LCOM2 metric was not normalized between 0 and 1, normalization and outlier analysis were performed on the dataset of the LCOM2 metric. All of the analysis results obtained after these processes. According to the results obtained, the best performances were obtained for the LCOM2 metric with an error value of 5.080 after outlier analysis, 0.079 error value after normalization using KNN algorithm, using the KNN algorithm with an error value 0.231 for TCC, using the REPTree

algorithm with an error value 0.259 for LCC and using the REPTree algorithm with an error value 0.149 for LSCC

KEYWORDS: Software Quality, Software Cohesion, Machine Learning, LCOM, Maintainability

İÇİNDEKİLER

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

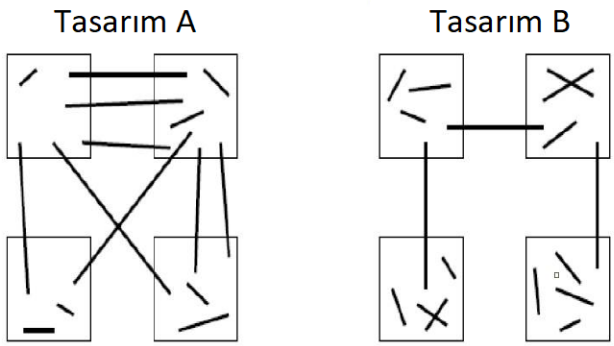
SİMGELER VE KISALTMALAR LİSTESİ

AOP	Arithmetic Operator Change: Aritmetik Operatör Değişimi
C&K	Chidamber and Kemerer
CBO	Coupling Between Objects: Sınıflar Arası Bağımlılık
CCC	Client-Based Cohesion Metric: İstemci Tabanlı Uyum Metriği
CSV	Comma Seperated Values: Virgülle Ayrılmış Değerler
DIT	Depth of Inheritance Tree: Kalıtım Ağacının Derinliği
DLCOM	Dynamic Lack of Cohesion in Methods: Metotlarda Dinamik Uyum Eksikliği
FUP	Frequent Usage Patterns: Sık Kullanım Örüntüleri
IMMC	Impact of Mutant Methods Cohesion: Mutant Metot Uyumlarının Etkisi
ISO	International Organization for Standardization: Uluslararası Standartlar Organizasyonu
JSON	JavaScript Object Notation: JavaScript Nesnesi Gösterimi
LCC	Loose Class Cohesion: Gevşek Sınıf Uyumu
LCC	Logical Connector Change: Mantıksal Bağlantı Değişimi
LCCM	Lack of Conceptual Cohesion Methods: Metotlarda Kavramsal Uyum Eksikliği
LCOM	Lack of Cohesion of Methods: Uyum Eksikliği
LSCC	Low Level Design Class Cohesion Metric: Düşük Seviyeli Tasarım Sınıf Uyum Metriği
NOC	Number of Children: Alt Sınıf Sayısı
RAAR	Runtime Attribute Access Rate: Çalıştırma Zamanında Öznitelik Erişim Oranı
RFC	Response for a Class: Sınıfın Tetiklediği Metot Sayısı
RLCOM	Runtime Lack of Cohesion in Methods: Çalıştırma Zamanında Metotların Uyum Eksikliği
RLOC	Revised Line of Code: Gözden Geçirilen Satır Sayısı
RMMC	Runtime Method Method Call: Çalıştırma Zamanında Metot-Metot Çağırma
ROC	Relational Operator Change: İlişkisel Operatör Değişimi
SCCM	Scoped Class Cohesion Metric: Kapsamlı Sınıf Uyum Metriği
TCC	Tight Class Cohesion: Sıkı Sınıf Uyumu
TXT	Text: Yazı
UPBC	Usage Pattern Based Cohesion: Kullanım Modeli Tabanlı Uyum
WMC	Weighted Methods per Class: Sınıfın Ağırlıklı Metot Sayısı
XML	Extensible Markup Language: Genişletilebilir İşaretleme Dili

1. GİRİŞ

Yazılım kalitesi yazılımın maliyetini, anlaşılabilirliğini, test edilebilirliğini, güvenilirliğini ve bakım yapılabilirliğini etkileyen temel etkendir. Yazılımın kalitesi, yazılıma olan ihtiyacın artmasıyla daha da önem kazanmıştır. Literatürde yazılımın kalitesini ölçmek için birçok metrik ve metrik seti vardır. ISO (International Organization for Standardization) 9126 standardı yazılım ürünlerinin değerlendirilmesi için kullanılan kalite modellerinden biridir. Bu kalite özellikleri içsel kalite, dışsal kalite ve kullanımdaki kaliteye bağlı olarak gruplandırılmıştır [1]. Dışsal kalite yazılımı kullanacak olanları ilgilendirmektedir. İçsel kalite ise yazılımı geliştirenlerin dikkat etmesi gerekenlerden oluşmaktadır. ISO 9126 standardına göre bir yazılımın kaliteli olabilmesi için içsel ve dışsal kalite özellikleri olarak işlevsellik, güvenilirlik, kullanılabilirlik, verimlilik, bakım yapılabilirlik ve taşınabilirlik özelliklerini sağlaması gerekmektedir [2].

Yazılım yaşam döngüsü sırasıyla planlama, analiz, tasarım, gerçekleştirim ve bakım onarım aşamalarından oluşmaktadır [3]. Bakım yapılabilirlik, yazılım yaşam döngüsünün en çok zaman ve maliyet harcandığı kısımdır [4]. Yazılım kalitesine özen gösterilerek oluşturulan yazılımlarda, bakım ve onarım aşaması daha az maliyet ve zaman harcanarak tamamlanabilir. Yazılım kalitesini ölçmek için literatürde en çok kullanılan metrik seti C&K (Chidamber and Kemerer) metrik setidir [5]. Bu metrik setinde WMC (Weighted Methods per Class), DIT (Depth of Tree), NOC (Number of Children), CBO (Coupling Between Objects), RFC (Response for a Class), LCOM (Lack of Cohesion in Methods) olmak üzere 6 adet kalite ölçütü bulunmaktadır [5]. Bu çalışmada hesaplanması diğer C&K metriklerine göre nispeten zaman alıcı ve kavram olarak daha soyut olduğu için uyum metriği üzerinde durulmuştur. LCOM sınıfın uyum değerini ölçen C&K metriğidir [6]. Sınıf uyumu nesne yönelimli bir yazılımda temel kalite özelliğidir [7]. Sınıf uyumunu ölçmek için genelde metotlar ve öznitelikler arasındaki yapısal ilişki baz alınmaktadır [8]. Metotların kendi içindeki uyumu veya metot-öznitelik arasındaki ilişkiye göre uyum değeri ölçülmektedir. Uyum metriği yazılımın bakım yapılabilirliğine doğrudan etki etmektedir [9]. Dolayısıyla uyum değerinin tespit edilmesi yazılımın kalitesi açısından oldukça önemlidir. Yazılım geliştiriciler kötü tasarlanmış sınıfların yeniden düzenlenmesinde yol göstermesi için uyum metriklerini kullanırlar [7].



Şekil 1.1. Farklı 2 Yazılım Tasarımı Örneği [10]

Bir yazılımın kaliteli olabilmesi için sınıflar arası bağımlılığın düşük, sınıf içi bağlılığın yüksek olması beklenmektedir. Şekil 1.1.'de 2 adet örnek sınıf tasarımı verilmiştir. Her tasarımda 4 adet sınıf bulunmaktadır. Bu tasarımlar incelendiğinde A tasarımında sınıfların birbirleri ile etkileşimi fazla, sınıf içi etkileşimin düşük olduğu görülmektedir. B tasarımında ise sınıfların birbiri ile etkileşimi düşük sınıf içi etkileşim fazladır. Kıyaslama yapıldığında B tasarımının A tasarımından daha kaliteli olduğu söylenebilir.

Literatürde sınıfın uyum değerini farklı açılardan ele alarak hesaplayan birçok metrik bulunmaktadır. Kaynak kodun gözle taranarak uyumun hesaplanması zor olmaktadır. O yüzden uyum metriklerini hesaplamak için birçok araç geliştirilmiştir. Bu araçlar girdi olarak kaynak kodu alıp, verilen kodun uyum değerini çıktı olarak vermektedir. Çıktıda elde edilen değere göre kodun tasarımı yeniden gözden geçirilmelidir. Uyumsuz yazılmış kodlar yazılım geliştirme yaşam döngüsünde bakım yapılabilirlik aşamasında daha fazla maliyet ve zaman gerektirecektir. Yapılan bir çalışmaya göre literatürde en çok kullanılan uyum metrikleri LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, TCC ve LCC olarak belirlenmiştir [11]. Bildiğimiz kadarıyla, literatürde uyum metriklerini makine öğrenmesi yöntemleri ile tahmin eden herhangi bir çalışma bulunmamaktadır. Bu çalışmada literatürde en çok kullanılan [11] metrik olan LCOM2, normalize edilmiş metrikler olmaları nedeniyle TCC ve LCC, metotlar arasında sadece ilişki olup olmamasını değil var olan ilişkinin düzeyini de dikkate aldığı için LSCC kullanılmıştır. Farklı tür hesaplama yöntemi ve dikkate alınan farklı durumları bulunan 4 metriğin makine öğrenmesi yöntemleri ile tahmini yapılmıştır. LCOM2 [12] metriği sınıfın uyum değerini ölçmek için kullanılan en eski yöntemlerden biridir. Sınıftaki metotların birbiri ile aynı özniteliği kullanıp kullanmama durumuna göre hesaplama yapmaktadır. LCOM2 metriği uyum ölçümünün çıkış noktası olarak kullanılmış ve hesaplama için başka parametreler dikkate alınarak günümüze kadar geliştirilen versiyonları tasarlanmıştır. TCC ve

LCC [13] metrikleri ise metotlar arası ve metot-öznitelik ilişkilerini dikkate almıştır. TCC metriği sadece doğrudan bağlantıları dikkate alırken LCC metriği sadece doğrudan bağlantıları değil dolaylı ilişkileri de göz önünde bulundurarak hesaplama gerçekleştirmiştir. TCC ve LCC metrikleri 0 ve 1 normalize edildiği için elde edilen uyum değerini yorumlaması daha kolay hale gelmektedir. LSCC [9] metriği ise sadece sınıf uyumunu değil metotlar arasındaki bağlantının derecesini de hesaplamaktadır. Tez kapsamında yapılan çalışmada bu farklı hesaplama özelliği bulunan 4 metriğin değerleri gerçek değere en yakın olacak şekilde tahmin edilmeye çalışılmıştır. Bunu gerçekleştirebilmek için farklı makine öğrenmesi algoritmaları kullanılmıştır. Elde edilen sonuçlar metrik bazında farklılaşmıştır.

TCC, LCC ve LSCC metriklerinden farklı olarak LCOM2 0 ve 1 arasında normalize edilebilen bir metrik değildir. Veri kümesinde LCOM2 metriğine ait gerçek değerler 0 ile 5749 arasındadır. Bu nedenle sadece bu veri kümesine uç değer ve aykırı değer analizi yapılmıştır. Analiz sonucunda 64 ile 120 arasında bulunan değerler aykırı, 120'den büyük olan değerler ise uç değer olarak belirlenmiştir. Veri kümesinden bu aralıktaki veriler çıkarılarak kalan 1052 veri ile deney tekrarlanmıştır. Sonrasında aynı veri kümesine normalizasyon işlemi yapılmıştır. Değerler 0 ve 1 arasında normalize edilmiştir. Bu işlemler sonucunda elde edilen deney sonuçları ayrı ayrı verilmiştir.

Bu çalışma ile yazılım uyum değerinin sınıfta bulunan metot sayısı ve öznitelik sayısına bakılarak daha hızlı ve kolay bir şekilde gerçek değerine en yakın değerin elde edilmesi amaçlanmıştır. Yazılım uyum değerini araç ile hesaplamanın getirmiş olduğu zaman ve maliyet kaybının en aza indirilmesi hedeflenmiştir. Yapılan çalışma ile ilgili detaylı bilgiler ilerleyen bölümlerde verilmektedir.

Giriş bölümünde; yazılım kalitesinin ölçülmesinin önemi, yazılım kalitesinin ölçülmesinde kullanılan C&K metrik seti ve çalışmada kullanılacak yazılım uyum metriklerinden ve neden bu metriklerin seçildiğinden kısaca bahsedilmiştir.

Literatür çalışması bölümünde; yazılım uyum metriğinin hesaplanması amacıyla kullanılan araçlardan ve literatürde sıklıkla kullanılan uyum metriklerden oluşan örnek çalışmalar incelenmiştir.

Yapılan çalışma bölümünde; seçilen makine öğrenmesi teknikleri, tahmin edilmek istenilen uyum metrikleri, veri kümesinin nasıl hazırlandığı, uyum metriklerini hesaplarken kullanılan araç ile ilgili detaylar, başarı ölçütü olarak hangi metriklerin kullanıldığı ve neden bu ölçütlerin seçildiği detaylı şekilde anlatılmıştır. Yapılan deneyin sonucunda elde edilen bilgiler ayrı ayrı tablolarda verilerek yorumlanmıştır.

Sonuç ve öneriler bölümünde; yapılan çalışmanın kısa bir özeti verilerek tüm metrikler için elde edilen sonuçlar ayrı ayrı yorumlanmıştır. Her bir uyum metriği için en iyi sonucun hangi makine öğrenmesi tekniği kullanılarak ve ne kadarlık bir hata değeri ile elde edildiğinden bahsedilmiştir. Çalışmada elde edilen sonuçların iyileştirilmesi için önerilere de bu bölümde yer verilmiştir.

2. LİTERATÜR ÇALIŞMASI

Yazılımın uyumu yazılımın kalitesini, anlaşılabilirliğini ve bakım yapılabilirliğini artırmak, karmaşıklığını azaltmak için önemli bir etkendir. Bu konuda hesaplama yapabilmek için önerilmiş birçok yöntem, metrik ve araç bulunmaktadır. Hesaplanan uyum değerine göre kodun kalitesi hakkında çıkarım yapılabilir. Bu yüzden uyum değerinin hesaplanması yazılımın kalitesi açısından oldukça önemlidir. Bu bölümde yazılımın uyumunu hesaplamak için öne çıkan çalışmalar incelenmiştir.

LCOM1 [14] Chidamber ve Kemerer tarafından 1991 yılında tanımlanmıştır. LCOM metriği uyumsuzluğu ölçtüğü için LCOM değerinin düşük olması uyumun yüksek olduğu anlamına gelmektedir. LCOM1 ölçütü arasında öznitelik paylaşmayan metot çifti sayısı olarak hesaplanır. LCOM1 metriği normalize edilmemiştir. Alabileceği minimum değer 0, maksimum değer ise metot çifti sayısıdır. Değer 0'a ne kadar yakın ise o kadar uyumlu bir sınıftır.

LCOM2 [12] metriği Chidamber ve Kemerer tarafından 1994 yılında tanımlanmıştır. Uyumsuzluğu ölçtüğü için metrik değerinin düşük olması beklenmektedir. P parametresi arasında öznitelik paylaşmayan metot çifti sayısı olarak, Q parametresi ise arasında öznitelik paylaşan metot çifti sayısı olarak tanımlanmıştır. P değeri Q değerinden büyük ise LCOM2 metriği P-Q olarak hesaplanmaktadır. Aksi halde LCOM2 değeri 0 olarak bulunur.

LCOM3 [15] metriği 1995 yılında Hitz ve Montazeri tarafından önerilmiştir. Hesaplama LCOM1 ve LCOM2'den farklı olarak grafik yapısı ile sağlanmaktadır. LCOM3 metriğinde yönsüz bir grafik oluşturulur. Bu grafiğin düğümleri metotları gösterir. Metotlar arasındaki bağlantılar ise grafikler arasındaki kenarları gösterir. Eğer 2 metot aynı özniteliğe erişiyor ise o 2 metot düğümü arasına bir çizgi çizilir. Tüm metot çiftleri için grafik tamamlandığında ise bağımsız bileşen sayısı LCOM3 değerini vermektedir. Bağlantılı olan düğümler tek olarak kabul edilir. LCOM3 metriğinin minimum değeri 1, maksimum değeri ise metot sayısıdır. Tüm metotlar birbiri ile bağlantılı olduğunda bağımsız bileşen sayısı 1 olur ve bu durum sınıfın tam uyuma sahip olduğunu gösterir. Eğer bileşen sayısı 2 ve 2'den fazla ise sınıfın 2 veya daha fazla sınıfa bölünmesi gerektiği anlaşılır.

LCOM4 [16] metriği LCOM3 gibi Hitz ve Montazeri tarafından 1995 yılında önerilmiştir. LCOM3'ün geliştirilmiş versiyonudur. LCOM4 metriğinin hesaplama yöntemi LCOM3 ile aynıdır fakat tek fark LCOM3'te sadece metotların ilişkisine, herhangi bir öznitelik paylaşıp paylaşmama durumuna göre bakılırken; LCOM4'te buna ek olarak metotların birbiri arasındaki ilişki de dikkate alınmaktadır. Eğer 2 metot birbirini çağırıyor ise bu 2 metot bağlantılıdır. LCOM4 bu ilişkiyi de dikkate almaktadır.

LCOM5 [17] uyum metriği Henderson ve Sellors tarafından 1996 yılında tanımlanmıştır. Metrik formülü Eş. 2.1'de verilmiştir.

$$\text{LCOM5} = \frac{a - kl}{l - kl} \tag{2.1}$$

l= öznitelik sayısı

k= metot sayısı

a= her bir metot tarafından erişilen öznitelik sayısı toplamı

LCOM5 için eğer sınıfta herhangi bir öznitelik yoksa veya sadece bir metot var ise payda 0 olacağından LCOM5 değeri sonsuz olarak hesaplanır. LCOM5 değeri 0 ve 1 arasında değer alabilen bir metriktir. LCOM5 değeri 0'a yaklaştıkça uyum artmaktadır.

TCC [13] ve LCC [13] metrikleri 1995 yılında Bieman ve Kang tarafından geliştirilmiştir. TCC ve LCC 0 ile 1 arasında normalize edilmiştir. Değer 0'a yaklaştıkça uyum düşer. A ve B ve C birer metot olsun. A metodu B metodunu çağırdığında A ve B doğrudan bağlantılı olur. Fakat A metodu B metodunu çağırıp, B metodu da C metodunu çağırdığında A ve C metodu dolaylı olarak bağlantılı olur. N metot sayısı olsun. Maksimum olası bağlantı sayısı ise (N*(N-1)) / 2 olarak hesaplanır. TCC sadece doğrudan bağlantılı metot sayısını dikkate alırken LCC hem doğrudan hem dolaylı bağlantıları göz önünde bulundurur. İlgili hesaplama formülleri Eş. 2.2 ve 2.3'te verilmiştir.

$$\text{TCC} = \frac{\textit{Doğrudan bağlantı sayısı}}{\textit{Maksimum olası bağlantı sayısı}} \tag{2.2}$$

$$\text{LCC} = \frac{\textit{Doğrudan bağlantı sayısı} + \textit{Dolaylı bağlantı sayısı}}{\textit{Maksimum olası bağlantı sayısı}} \tag{2.3}$$

LSCC [9] ise 2012 yılında Dallal ve Briand tarafından önerilmiştir. LSCC metriği sadece iki metot arasında uyum olup olmamasını değil ayrıca uyum var ise bunun derecesini ölçmeyi hedeflemektedir. LSCC 0 ve 1 arasında normalize edilmiştir. Değer 1'e ne kadar yaklaşırsa uyum o kadar artar. Hesaplanma aşamalarından ilki k*l boyutunda metot-öznitelik referans matrisi oluşturmaktır. Burada k metot sayısını, l ise öznitelik sayısını göstermektedir. Matriste her bir indis için, eğer i. metot j. niteliği kullanıyorsa kesişimleri olan indise 1, diğer durumda 0 yazılır. Oluşturulan matriste 2 satır arasındaki benzerlik metot çiftleri arasındaki uyumu gösterir. Bir metot çifti arasındaki benzerlik, diğer metot çiftleri arasında aynı ikilik değere sahip olması ile sayısallaştırılır. Bunu ölçmek için Eş. 2.4'teki formül kullanılır:

$$\text{ns(i,j)} = \frac{\sum_{x=1}^{Y}(m_{ix} \cap m_{jx})}{Y} \tag{2.4}$$

Formülde i ve j değişkenleri matristeki satırları temsil etmektedir. Y ise matrisin bir satırındaki eleman sayısını gösterir. LSCC formülü ise Eş. 2.5'teki gibidir.

$$\text{LSCC} = \begin{cases} 0 \; eğer \; l = 0 \; ve \; k > 1 \\ 1 \; eğer \; (l > 0 \; ve \; k = 0) \; veya \; k = 1 \\ \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} ns(i,j) \; diğer \; durumlar \end{cases} \tag{2.5}$$

Alzahrani et al., [18] tarafından 2019 yılında yapılan çalışmada; Client-Based Cohesion Metric (CCC) önerilmiş ve bu metriğin bakım yapılabilirlik tahmininde güçlü bir etken olduğu vurgulanmıştır. Bakım yapılabilirlik ölçütü olarak RLOC (Revised Line of Code) kullanılmıştır. Uyum metriklerinin gerçekleştirim aşamasından önce, tasarım aşamasında elde edilmesi gerektiği düşünülmüştür. CCC metriğinin hesaplanabilmesi için UML sınıf diyagramı ve bağlantı diyagramının yeterli olacağı belirtilmiştir fakat dizayn aşamasında metotlar arası ve metot öznitelik arası bağlantılar henüz belli olmadığı için tam olarak sınıf uyumunu yansıtamayacağı düşünülmektedir. Ayrıca büyük projeler için araç kullanmadan uyumu hesaplamak uzun süreceği için bu durumda da araç kullanarak aynı maliyet ve zaman kaybı yaşanacaktır.

Yusuf and Hammad [19] yapmış oldukları çalışmada uyumu ölçmek için yazılım mühendisleri tarafından çok zaman ve çaba harcanması gerektiği için uyumu otomatik ölçen bir yaklaşım önermişlerdir. Yaklaşım SrcML aracını kullanarak program kaynak kodunu bir XML (Extensible Markup Language) dosyasına ayrıştırır. Sonrasında XML dosyasındaki belirteçler ayrıştırılmaktadır. Sınıf kaynak kodu belirteçleri çıkarıldıktan sonra, sınıf ögeleri arasındaki uyum ilişkisi eşleştirmesi sağlanmaktadır. Eşleşme varsa bu değerler CSV (Comma

Seperated Values) ve JSON (JavaScript Object Notation) dosyalarında depolanmaktadır. Fakat bu çalışmada 3 ayrı araç kullanılması gerektiği belirtilmiştir. Bu durum oldukça zaman alıcı olmaktadır. Ayrıca kaynak kodun içerisinden direkt olarak metot ve öznitelik sayısı çıkarılabilmektedir. Kaynak kodu XML'e çevirerek belirteçlerin çıkarılması ve bu belirteçler üzerinden işlem yapmak durumu daha uzun hale getirmektedir.

Rathee et al., [20] tarafından yapılan çalışmada modül seviyesinde uyum ölçen UPBC (Usage Pattern Based Cohesion) metriği önerilmiştir. Sınıf başlangıçta bir modül olarak ele alınmaktadır. Sonrasında sınıflar grubu genel uyumu iyileştirme amacıyla modül olarak değerlendirilmektedir. Kümeleme modüller arasındaki FUP (Frequent Usage Patterns) etkileşimlerine dayalı olarak FUPClust adı verilen yeni önerilen bir kümeleme algoritması kullanarak gerçekleştirilir. Ancak sınıfları birleştirmek her sınıfın bir amaca hizmet etmesi gerektiği kuralını ihlal eder. Dolayısıyla uyum ölçütü bu kuralın sağlanıp sağlanmadığını tam olarak ölçemeyeceği düşünülmektedir.

Priyambadha et al., [21] tarafından yapılan çalışmada yöntemler ve öznitelikler arasındaki ilişkinin uyum derecesini artırabilecek bilgiler elde etmeye çalışılmış ve aralarındaki ilişki için isim özniteliği kullanılmıştır. Eğer metodun ve özniteliğin isimleri benziyor ise onlar arasında ilişki olduğunu belirtmişlerdir. Deneysel sonuçlara göre anlam benzerliğine bakılarak üretilen uyum değerinde bir artış olduğu görülmüştür. Metotlara veya özniteliklere verilen isimler yazılımcının fikirlerine, deneyimine, görüşüne göre değişebileceği için farklı durumlar yaşanması olasıdır. Her yazılımcı isim verirken farklı yorumlamalar yapabilir. Farklı isimlendirmeler yapıldığında ise uyumun düşmesi ihtimali göz önüne alındığında yazılımcıya göre uyum değeri değişiklik gösterebilir. Bu durumun güvenilir bir uyum değeri elde edilmesine engel olabileceği düşünülmektedir.

Tee [22] çalışmasında karmaşık hesaplama kullanmadan sınıf uyumunu ölçmek için bir teknik önermiştir. Önerilen yaklaşım IMMC (Impact of Mutant Methods Cohesion) olarak isimlendirilmiş sezgisel bir yaklaşımdır. Bu araştırmada mutant yöntemler ve mutant operatörler kullanılmıştır. AOP (Arithmetic Operator Change), LCC (Logical Connector Change) ve ROC (Relational Operator Change) mutant operatörleri kullanılmış ve mutant sayısı arttıkça sınıfın daha uyumlu olduğu belirtilmiştir. Mutant operatörleri kaynak kodda göz ile taramak ve mutant operatör sayılarını çıkarmak zaman alabilecek bir işlemdir. Sınıftaki metot sayısı arttıkça bu işlem daha da uzun bir hale gelecektir. Literatürde kullanılan diğer uyum metrikleri ile kıyaslandığında da sınıf uyumunu etkileyecek tüm durumların göz önüne

alınmadığı görülmektedir. Bu durum da yazılımcıyı kodun uyumu konusunda yanlış yönlendirebilir.

Altaie [23] çalışmasında uyum derecesini bir sınıfta kullanılan öznitelik sayısının o sınıf için toplam öznitelik sayısına oranı olarak hesaplamıştır. Bir araç geliştirmiştir. Bu araç oluşturulan sınıf diyagramından XMI (XML Metadata Interchange) dosyasını oluşturur ve bu XMI dosyasından da belirteçler çıkarılır. Bu belirteçler aracılığıyla uyum hesaplamak için gereken bilgiler elde edilir. Fakat sadece öznitelik sayısına ve bu özniteliklerin kullanım durumuna bakılarak uyumun hesaplanması tüm sınıfın uyum değerini yansıtmayacağı düşünülmektedir. Dolayısıyla geliştirilen bu aracın ve hesaplama yönteminin sınıf uyumunu hesaplama konusunda eksikleri bulunmaktadır.

Razafimahatratra et al., [24] tarafından yapılan çalışmada UML dizi diyagramındaki her bir nesnenin uyum tipini tespit etmek için bir algoritma önerilmiştir. Uyum tipleri tesadüfi, mantıksal, geçici, prosedürel, iletişimsel, sıralı, fonksiyonel olmak üzere öncelik sırasına göre düşükten yükseğe sıralanmıştır. Burada en zayıf olan uyum türü tespit edilerek bu uyum türü sınıfın yeniden tasarlanması ile o uyum türü üst seviyelere taşınmaktadır. Bu çalışmada belirli durumlara bakılarak uyum türü tespit edilmiştir fakat bu uyum türünün derecesi ile ilgili bilgi verilmemiştir. Hangi seviyede sınıfın yeniden düzenlenmesi gerektiği konusunda bilgi bulunmamaktadır. Sadece uyum türü tespit edilmiştir. Sınıfın uyum değeri hakkında herhangi bir bilgi bulunmamaktadır. Bu durumda sınıf tasarımını değiştirmek için yeterli bilginin olmadığı düşünülmektedir.

Singh and Khalsa [25] yaptıkları çalışmada program dilimleme kullanarak sıralama (sequence) diyagramından uyumu ölçmeyi hedeflemişlerdir. Bu çalışmada sıralama diyagramının durumlarından ve senaryolarından sıralama bağımlılık grafiği oluşturulmuştur. Bu grafik çeşitli yönler dikkate alınarak dinamik olarak dilimlenir. Bu dilimler uyum ölçmek için kullanılır. Her dilimdeki ortak düğüm sayısı, yapılan toplam dilim sayısı, en küçük dilimdeki düğüm sayısı, en büyük dilimdeki düğüm sayısı uyum hesaplama formüllerinde kullanılmaktadır. Sıkılık, kapsam, en küçük kapsam, en büyük kapsam ve örtüşme (overlap) durumları hesaplanmıştır. Bu çalışmada hesaplanan 4 formülden hangisinin tam olarak uyum değerini verdiği belirtilmemiştir. Ayrıca 4 formül için alınan değerler sonucu sınıfın uyumu hakkında ne tür bir yorum yapılacağı hakkında da bilgi bulunmamaktadır.

Wanjiku et al., [26] tarafından yapılan çalışmada SCCM (Scoped Class Cohesion Metric) metriği önerilmiştir ve bu metriği hesaplayan bir araç geliştirilmiştir. Metrik 0 ve 1 arasında normalize edilmiştir. Public, private ve protected yöntem ve öznitelik sayıları

kullanılarak hesaplama sağlanmıştır. Genel ve özel uyumun toplam değeri ile genel sınıf uyumu hesaplanmıştır. Fakat sınıf içerisindeki private veya protected öznitelik ve metotlar da genel sınıfa ait metot sayısı ve öznitelik sayısına doğrudan dahil edilmektedir. Özel ve korumalı metot ve öznitelik sayılarının ayrıca hesaplanması ve sonradan birleştirilmesinin daha çok formül ve uğraş gerektirdiği düşünülmektedir.

Jain and Gupta [27] yapmış oldukları çalışmada LCCM (Lack of Conceptual Cohesion Methods) isimli metrik önermiştir. Bu uyum metriği LCOM metriğinin kavramsal versiyonu olarak tanımlanmıştır. Koddaki yorumlara ve tanımlayıcıların analizine bağlı olarak hesaplanmaktadır. LDA (Latent Drichlet Allocation) tekniği kullanılarak kaynak koddaki ilişkili kelimeler çıkarılmaktadır. LCOM1-5 uyum metriklerine karşılık olarak LCCM1-5 tanımlanmıştır. Uyum belirleyicisi olarak tanımlayıcıların ismi ve yorum satırlarının dikkate alınmasının sübjektif olduğu düşünülmektedir. Verilen tanımlayıcı isimler, yazılan yorumlar ve buralarda kullanılan kelimeler yazılımcıya bağlı olarak değişmektedir.

Chen et al., [28] yaptıkları çalışmada OntRECoh (Online Ontological Cohesion Assessment for SRE Design Quality) aracını önermişlerdir. Bu yaklaşım, sistemlerin uyum değerlerini hesaplamak ve web tabanlı ara yüzü sayesinde iyileştirme önerilerini sunabilmek için tasarlanmıştır. Araç internet ortamında çalışmaktadır. Bu çalışmada tersine mühendislik ile kod oluşturulduktan sonra gereken tasarım, UML diyagramı ile oluşturulmuştur. Araç ile statik ve dinamik uyum değerleri hesaplanırken, uyumu iyileştirmek için tavsiyelerde bulunulmaktadır. Bu çalışmada oluşturulan kaynak kod bazlı uyum hesaplanabilecekken ayrıca tersine mühendislik ile oluşturulan kod üzerinden tekrar tasarım diyagramının oluşturulması ve sonrasında uyum hesaplanmasının ekstra süre ve çaba gerektireceği düşünülmektedir.

Manju et al., [29] yaptıkları araştırmada bir sınıfın uyumunu çalışma zamanında (run time) hesaplayabilmek için DLCOM (Dynamic Lack of Cohesion in Methods) metriğini önermişlerdir. Ayrıca bu metriği hesaplayabilmek için DynaDlcom adlı bir araç geliştirilmiştir. Bunu hesaplayabilmek için RLCOM (Runtime Lack of Cohesion in Methods), RAAR (Runtime Attribute Access Rate) ve RMMC (Runtime Method Method Call) metrikleri de hesaplanıp formüle dahil edilmiştir. Metrik 0 ile 1,5 arasında değer almaktadır. Değer 0 ise sınıf uyumu yüksektir. Araç AspectJ'de çalışmaktadır. Alınan değerlere göre derleme zamanı ile çalışma zamanı uyum değerleri arasında fark olduğu gözlemlenmiştir. Bu durumu sınıf özniteliklerine sınıf yöntemleri ile tam olarak erişilemediği ile açıklamaktadırlar. Erişilememe durumu kaynak koda bakılarak da hesaplanabilmektedir. Kod çalıştırılmadan

önce uyumun hesaplanması ve eğer gerekiyorsa sınıf tasarımının güncellenmesi gerektiği düşünülmektedir.

Oluşturulan yazılımın uyum değerinin hesaplanması için literatürde birçok çalışma yapılmıştır. Yapılan çalışmalar Bölüm 2'de detaylı olarak anlatılmıştır. Yapılan çalışmalar dikkate alındığında yazılım uyumun yeni formüller veya araçlarla ölçülmeye çalışıldığı görülmektedir. Fakat araçlarla ölçüm yapmak uzun ve zaman alıcı bir işlemdir. Bu çalışmada yazılımın uyum değeri makine öğrenmesi yöntemleri kullanılarak tahmin edilmeye çalışılmıştır. Bu sayede uyum hesaplama işlemi daha hızlı ve kolay bir şekilde gerçekleşmektedir.

3. YAPILAN ÇALIŞMA

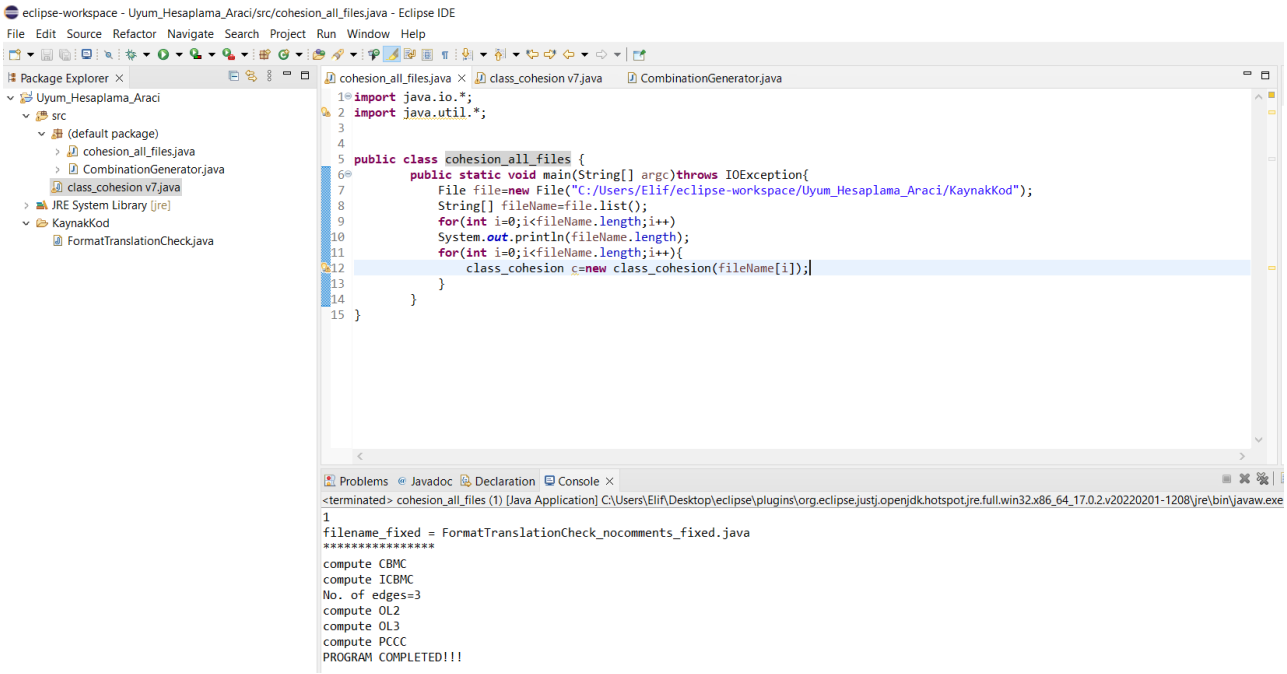
Tez kapsamında gerçekleştirilen çalışmada yazılım uyumunu ölçmek için kullanılan metriklerden LCOM2, TCC, LCC ve LSCC makine öğrenmesi yöntemleri kullanılarak tahmin edilmeye çalışılmıştır. Metrikler seçilirken farklı türde hesaplama ve özelliklerinin olmasına dikkat edilmiştir. En çok kullanılan metrik olması sebebi ile LCOM2, normalize edildikleri için TCC ve LCC, sadece sınıf bazında uyum olup olmamasını değil her metot ikilisi arasındaki uyumun derecesini de ölçtüğü için LSCC çalışmaya dahil edilmiştir. Çalışmanın amacı, sınıfta bulunan metot sayısı ve öznitelik sayısına bakılarak sınıfın uyum değerini tahmin etmektir. Bu sayede uyumu hesaplamak için araç kullanmaktan kaynaklanan zaman ve maliyet kaybı en aza indirilecektir.

3.1. Veri Kümesi

Yapılacak herhangi bir çalışmada, kullanılacak olan veri kümesini bulmak ve bulunan veriyi temizlemek oldukça zaman alıcı bir işlemdir. Herkese açık (public) olarak elde edilen veri kümelerinde tam olarak çalışmanın amacına uygun öznitelikler bulunmayabilir. Çalışmada kullanılması gereken fakat veri kümesinde eksik olan öznitelikler bulunabilir. Buna karşın, çalışma amacının dışında olmasına rağmen veri kümesinde bulunan farklı öznitelikler de bulunabilir. İhtiyaç olmayan özniteliklerin direkt olarak veri kümesinden çıkarılması da analizden alınan sonuçlarda farklılıklara sebep olabilmektedir. Bu da kurulan modelin gücünü düşürecektir. Hazır bulunan veri kümelerini yapılacak olan çalışmaya uygun hale getirmek de zaman alan bir işlemdir. Tam olarak çalışmaya uygun özniteliklerin olduğu veri kümesini bulmak da oldukça zordur. Ayrıca herkese açık olan veri kümelerinde bazı değerler boş olabilmektedir. Bunun gibi durumlarda bazı ön işlemler ile boş olan değer tahmin edilip veri kümesi doldurulabilmektedir fakat doldurulan değerin gerçek değere ne kadar yakın olduğu bilinmemektedir. Bu durum da veri kümesinin durumu tam olarak yansıtamamasına ve yapılacak analizlerin yanlış sonuçlar vermesine sebep olabilmektedir. Boş değeri olan kayıtların veri kümesinden çıkarılması durumunda ise, kalan kayıtların tüm durumu yansıtmayacağı ihtimali göz önüne alınmalıdır. Bunun gibi durumlar dikkate alınarak tez kapsamında yapılan çalışmada veri kümesi hazır olarak elde edilmemiş ve çalışma sürecinde üretilmiştir.

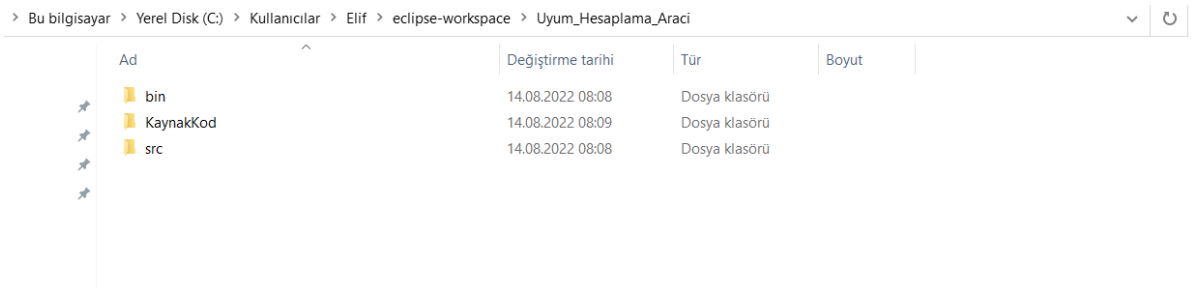
Veri seti oluşturmak için http://sourceforge.net sitesinden elde edilen, birçok nesne yönelimli programlama dili olmasına rağmen en çok kullanılan [11], Java ile oluşturulmuş

açık kaynak kodlu Freeplane uygulaması kullanılmıştır [30]. Uygulamada 1471 tane sınıf bulunmaktadır. 198,486 satır koddan oluşan orta ölçekli bir projedir. Literatürde bulunan uyum metrikleri ile genelde sınıf bazında hesaplama yapılmaktadır. 170 sınıfta sadece arayüz (interface) tanımı, 24 sınıfta sadece enum tanımı ve 6 sınıfın içeriğinin boş olması sebebiyle veya sadece öznitelik tanımı olduğu için bu sınıflar çalışmaya dahil edilmemiştir. Ayrıca 82 sınıf için metot sayısı 2'den küçük olduğu için uyum hesaplaması yapılmamıştır ve çalışmaya dahil edilmemiştir. Kalan 1189 sınıf için LCOM2, TCC, LCC ve LSCC metriklerinin gerçek değerleri Al Dallal [9] tarafından geliştirilen araç kullanılarak hesaplanmıştır. Veri kümesi sınıftaki metot sayısı, öznitelik sayısı, LCOM2, TCC, LCC ve LSCC değerleri ile oluşturulmuştur. Kullanılan araç Java dili ile oluşturulmuştur. Toplam 16 ayrı uyum metriğinin değerlerini hesaplayabilmektedir. Kullanılan araca ait ekran görüntüsü Şekil 3.1'de verilmiştir.

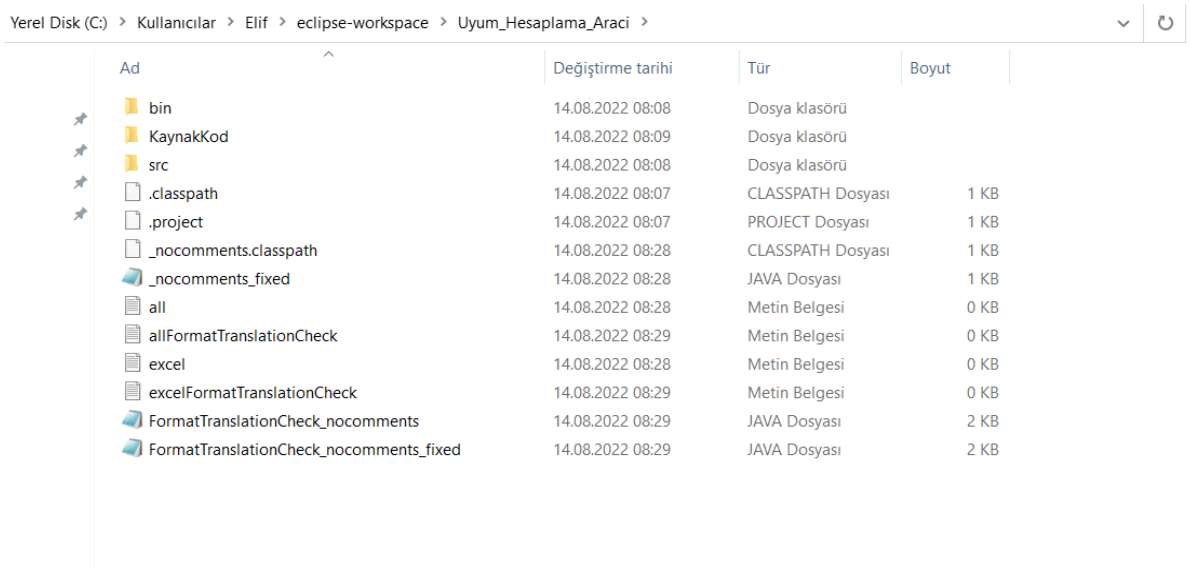


Şekil 3.1. Kullanılan Aracın Eclipse IDE üzerinde çalıştırılması

Aracı kullanabilmek için bir Java idesi üzerinde yeni proje oluşturularak aracın kaynak kodunda bulunan sınıfların bu projeye eklenmesi gerekmektedir. Toplamda 3 adet sınıftan oluşmaktadır. İlk olarak, "cohesion_all_files" sınıfı içerisine projenin ve uyum değerinin hesaplanması gereken kaynak kodun uzantısı verilmektedir. Program çalıştırıldığında, eğer herhangi bir hata ile karşılaşılmadı ise "PROGRAM COMPLETED" yazısı çıktı olarak ekrana verilmektedir.

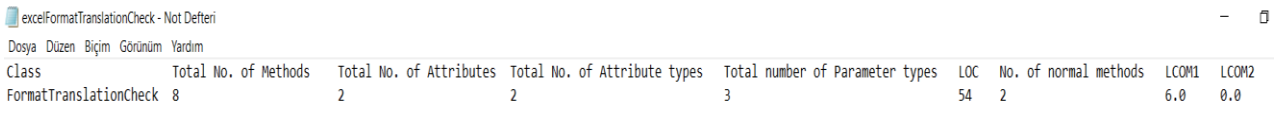


Şekil 3.2. Proje Çalıştırılmadan Önce Verilen Uzantıdaki Klasörler

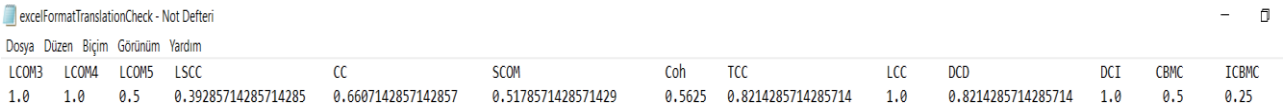


Şekil 3.3. Proje Çalıştırıldıktan Sonra Verilen Uzantıdaki Klasörler

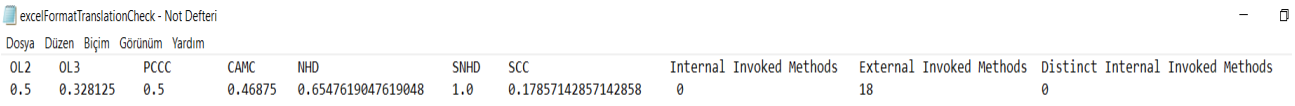
Ekran görüntüleri verilen örnekte “FormatTranslationCheck” sınıfını için uyum değerinin hesaplanması gösterilmiştir. Araç, uyumu hesaplayabilmek için hesaplanması istenilen kaynak kodun yorum satırlarını çıkararak yeni bir txt dosyası (FormatTranslationCheck_nocomments_fixed) üretmektedir ve hesaplama işlemlerini artık bu yeni txt dosyası üzerinden gerçekleştirmektedir. Araç, “cohesion_all_files” sınıfında belirtilen uzantıdaki klasöre yeni bir txt dosyası (excelFormatTranslationCheck) oluşturarak içerisine uyum değerlerini yazmaktadır. “FormatTranslationCheck” sınıfı için elde edilen uyum değerlerinin ekran görüntüleri Şekil 3.4.’de verilmiştir.



Şekil 3.4. Elde Edilen Uyum Değerleri



Şekil 3.4. devam ediyor



Şekil 3.4. devam ediyor

Oluşturulan veri kümesindeki en düşük metot sayısı 2, en yüksek metot sayısı 126, en düşük öznitelik sayısı 0 ve en yüksek öznitelik sayısı 33'tür. Veri kümesinde bulunan metot sayılarına ait ortalama değer 8, öznitelik değerlerine ait ortalama değer ise 3,24'tür.

3.2. Çalışmada Kullanılan Makine Öğrenmesi Teknikleri

Yapılan çalışmada koda ait uyum değerlerinin makine öğrenmesi teknikleri kullanılarak tahmin edilebilmesi için Weka aracı kullanılmıştır. Konu ile ilgili detaylı bilgi, Bölüm 3.3'te verilmiştir. Bu çalışmada makine öğrenmesi tekniklerinden KNN, REPTree, Rastgele Orman, MLP, SVM ve Doğrusal Regresyon kullanılmıştır.

- **KNN**

  K En Yakın Komşu algoritması sınıflandırma ve regresyon için kullanılan bir makine öğrenmesi tekniğidir. Yeni tahmin edilecek veri ile eğitim kümesinde bulunan her bir veri arasındaki mesafe ayrı ayrı hesaplanmaktadır. Mesafe hesaplamada genel olarak Öklid uzaklığı kullanılmaktadır. Verilen k değişkenine göre tahmin edilecek veriye en yakın k tane veri tespit edilmektedir. Bu k adet verinin ortalaması alınarak yeni verinin tahmini gerçekleştirilmektedir [31].

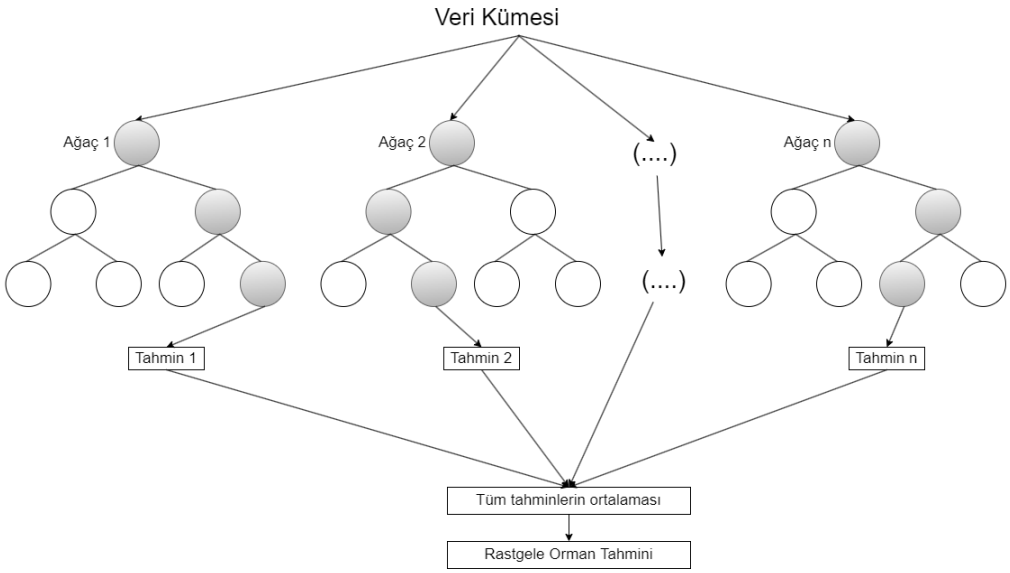
- **REPTree**

  REPTree regresyon/karar ağacı mantığı ile farklı yinelemeler sayesinde birden çok karar ağacı oluşturur. Regresyon ağacı verileri bölümlere veya dallara ayıran ve bu durumu ikili özyinelemeli bölümleme ile devam ettirerek çıktıları tahmin etmeye çalışan bir karar ağacıdır. REPTree algoritması varyans ve bilgi kazancına göre

oluşturulan ağaçlar arasından en iyisini seçer. Ayrıca bu algoritma azaltılmış hata budama ile ağacı budamaktadır [32].

- **Rastgele Orman**

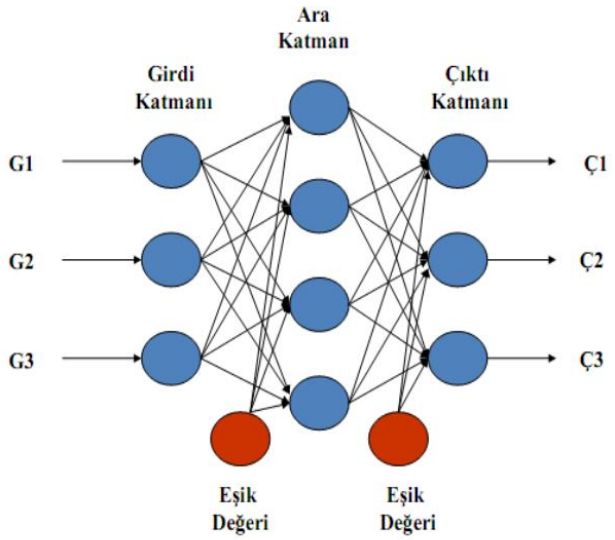
Rastgele Orman algoritması hem sınıflandırma hem regresyon için kullanılmaktadır. Veri kümesinden birden fazla birbirinden bağımsız karar ağacı oluşturularak her bir karar ağacı sonucu tahmin değerleri elde edilmektedir. Tüm karar ağaçlarından elde edilen değerlerin ortalaması alınarak regresyon problemi çözülmektedir [33].



Şekil 3.5. Rastgele Orman Regresyonu

- **MLP**

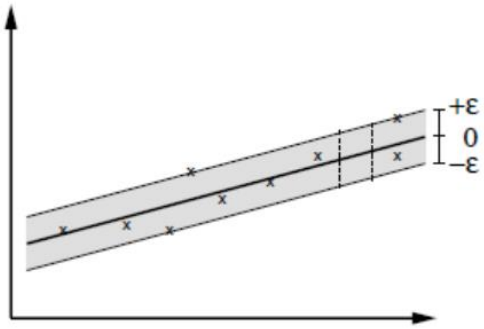
Çok katmanlı algılayıcı birden fazla katmanın birbirine bağlanması ile oluşturulan bir grafik olarak temsil edilmektedir. Girdi katmanı, ara katman ve çıktı katmanından oluşmaktadır. Girdi katmanı dışarıdan gelen verileri toplayarak ara katmana iletmek için kullanılmaktadır. Gelen bilgiler işlenmeden, direkt olarak ara katmana iletilir. Ara katmanda girdi katmanından gelen veriler işlenmektedir. Çok katmanlı bir ağda birden fazla ara katman bulunabilmektedir. Çıktı katmanında ise ara katmandan iletilen bilgiler için ağın ürettiği çıkış bilgilerini belirleyerek ağdan çıkarmaktadır [34].



Şekil 3.6. Çok Katmanlı Algılayıcı Modelinin Topolojik Yapısı [34]

- **SVM**

Veri kümesinde sahip olunan öznitelik sayısı kadar boyuta sahip bir düzlem üzerine kurgulanmış bir makine öğrenmesi algoritmasıdır. Eğitim kümesindeki her bir veri için bu düzlem üzerinde denk gelen noktalar belirlenir ve işaretlenir. Regresyon için bu düzlem üzerinde bir aralık belirlenmektedir ve bu aralık en çok noktayı içerisine alacak şekilde planlanmalıdır. Burada regresyon ile elde edilecek f(x) fonksiyonunun üreteceği değerin en fazla ε değeri kadar bir sapma ile bulunması hedeflenmektedir [35].



Şekil 3.7. Düzlemde Doğrusal Vektör Regresyonu [35]

- **Doğrusal Regresyon**

Doğrusal regresyon analizi bir değişkenin değerini başka bir değişkenin değerine göre tahmin için kullanılmaktadır. Tahmini yapılacak değişken bağımlı değişken; bağımlı değişkeni tahmin edebilmek için kullanılan değişkenlere ise bağımsız değişken denmektedir. Doğrusal regresyon bağımlı değişken ile bağımsız değişken arasında

ilişki kurmaktadır [36]. Doğrusal Regresyon tekniği ile oluşturulan denklem Eş. 3.1'de verilmiştir.

$$\boldsymbol{y' = b + w_1 + x_1} \tag{3.1}$$

y' = tahmin edilecek değer

b= y ekseni kesişim noktası

w_1 = öznitelik1 için verilen ağırlık değeri

x_1= öznitelik değeri

3.3. Çalışmanın Yöntemi

Çalışmada yazılım kalite ölçütlerinden biri olan sınıf uyum değeri gerçek değerine en yakın şekilde tahmin edilmeye çalışılmıştır. Bunu yapabilmek için sınıfta bulunan metot sayısı ve öznitelik sayısına bakılarak sınıfın uyum değerini ölçmek için kullanılan LCOM2, TCC, LCC ve LSCC metrikleri makine öğrenmesi yöntemleri kullanılarak tahmin edilmeye çalışılmıştır. Sayısal tahmin yapabilmek için regresyon analizi kullanılmaktadır. Regresyon analizinde bağımlı değişkenin değişimi bağımsız değişkene göre tahmin edilmektedir [37]. Bu çalışmada sınıfta bulunan metot sayısı ve öznitelik sayısına bakılarak LCOM2, TCC, LCC ve LSCC metrik değerleri tahmin edilmeye çalışıldığı için burada metot sayısı ve öznitelik sayısı bağımsız değişken; LCOM2, TCC, LCC ve LSCC bağımlı değişkendir.

Çalışmada regresyon analizi için Weka 3.8.6 kullanılmıştır. Weka Waikato Üniversitesi tarafından geliştirilmiş açık kaynak kodlu bir yazılımdır [38]. Veri madenciliği ve istatistik alanlarında sıklıkla kullanılmaktadır [39]. Weka veri madenciliği ve makine öğrenmesi tekniklerinin kullanımında bir dönüm noktası olarak kabul edilmektedir. Veri madenciliği konusunda akademik ve iş alanlarında yaygın olarak kabul görmektedir. Açık kaynak kodlu olması sayesinde Weka'nın gelişmesi de hızlanmıştır. Regresyon, sınıflandırma, ön işleme, veri görselleştirme, kümeleme için çeşitli algoritmalar içermektedir. Tasarlanan arayüzü de menülere kolay erişim sağlamaktadır [40]. Dolayısıyla çalışmada Weka aracı kullanılması tercih edilmiştir. Weka dosya türü olarak .arff uzantısı kabul etmektedir [41]. LCOM2, Normalize edilmiş LCOM2, TCC, LCC ve LSCC için ayrı arff dosyaları oluşturulmuştur. Her bir arff dosyasında metot sayısı, öznitelik sayısı ve metrik değeri bulunmaktadır. Weka 'ya bu arff dosyaları sırası ile yüklenmiş ve her biri için sırası ile makine öğrenmesi teknikleri uygulanmıştır. Arff uzantılı dosyalar Weka'ya yüklendikten sonra eğitim veri kümesindeki toplam eleman sayısı, arff dosyasında bulunan toplam öznitelik sayısı, seçilen özniteliğin türü,

ismi, ortalaması, standart sapması ve histogram grafiği bulunmaktadır.

3.4. Başarı Ölçütleri

Makine öğrenmesi teknikleri kullanılarak yapılan regresyon analizinin sonuçlarını ve kurulan modeli değerlendirmek için en sık kullanılan metriklerden Korelasyon Katsayısı (R), Ortalama Mutlak Hata (MAE) ve Kök Ortalama Kare Hata (RMSE) kullanılmıştır [42][43].

3.4.1. Korelasyon Katsayısı (Correlation Coefficient – R)

Korelasyon Katsayısı gerçek ve tahmin edilen değer arasındaki ilişkinin derecesini ve yönünü belirten bir ölçüttür. Regresyon analizinde gerçek ve tahmin edilen değer arasındaki ilişkiye bakıldığında ikisi de artma eğiliminde ise aralarında korelasyon olduğu söylenmektedir. Korelasyon Katsayısı -1 ile 1 arasında değer alabilmektedir. Değer 1 olduğunda biri artarken diğeri de artma eğiliminde olduğu anlamına gelmektedir. Bu durum aralarında mükemmel bir korelasyona sahip olduklarını belirtmektedir. Buna karşılık, korelasyon katsayısı -1 olduğunda biri artarken diğeri azalma eğiliminde olduğunu göstermektedir. R değeri 0,8'den büyük olduğunda güçlü korelasyon, 0,5'ten küçük olduğunda ise düşük korelasyon olduğu kabul edilmektedir. Değer 0 olduğunda ise aralarında korelasyon yoktur olarak yorumlanmaktadır [44].

3.4.2. Ortalama Mutlak Hata (Mean Absolute Error-MAE)

Ortalama Mutlak Hata bir makine öğrenmesi tekniği kullanılarak tahmin edilen değer ile gerçek değer arasındaki mutlak fark olarak hesaplanmaktadır. Regresyon modelini değerlendirmek için sıklıkla kullanılmaktadır. MAE değerinin düşük olması beklenen durumdur. MAE değeri 0'a yaklaştıkça modelin performansı daha iyidir şeklinde yorumlanmaktadır. [45] Ortalama Mutlak Hata formülü Eş. 3.2'de verilmiştir.

$$\text{MAE} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\mathbf{a}_{\boldsymbol{i}} - \boldsymbol{p}_{\boldsymbol{i}}| \tag{3.2}$$

a_i = Gerçek değer

p_i = Tahmin edilen değer

N = veri kümesindeki eleman sayısı

3.4.3. Kök Ortalama Kare Hata (Root Mean Squared Error -RMSE)

Kök Ortalama Kare Hata regresyon analizinde tahmin hatalarının standart sapması olarak tanımlanmaktadır. Tahmin hataları oluşturulan regresyon doğrusunun veri noktalarından ne kadar uzakta yayılım gösterdiğinin bir ölçüsüdür. Büyük hataların istenmediği durumlarda kullanılması daha avantajlıdır. Çünkü hata oranının karesi alınarak hesaplama yapıldığı için büyük hata oranlarını daha fazla cezalandırma eğilimindedir [46]. Kök Ortalama Kare Hata formülü Eş. 3.3'te verilmiştir.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(a_i - p_i)^2}{N}} \tag{3.3}$$

a_i= Gerçek değer

p_i = Tahmin edilen değer

N = veri kümesindeki eleman sayısı

3.5. Analiz Sonuçları

Tez kapsamında yapılan çalışmada kullanılan makine öğrenmesi algoritmaları KNN, Rastgele Orman, REPTree, Doğrusal Regresyon, MLP ve SVM'dir. Yapılan regresyon analizi sonuçları bu algoritmaların verdiği sonuçlar üzerinden değerlendirilecektir. Yapılan çalışmada kurulan modellerin hepsinde 10 kat çapraz doğrulama (10-folds cross validation) tekniği kullanılmıştır. Bu teknik ile verilen eğitim kümesi 10 eşit parçaya bölünmektedir. Her bir döngüde bir tanesi test için kalan 9 tanesi için eğitim için kullanılmaktadır. Diğer döngüye geçildiğinde başka bir bölüm test ve kalan diğer 9 bölüm eğitim içi kullanılmaktadır. Bu sayede aynı yöntem 10 kere 10 farklı eğitim ve test kümeleri kullanılarak çalıştırılmış olacaktır. Bu sayede her bölünen parça hem eğitim hem de test aşamasında kullanılacağından bölünmelerin sebep olduğu hatalar da bu sayede en aza indirgenmiş olacaktır [47].

Tablo 3.1. LCOM2 Metriği Regresyon Analizi Sonucu

	R	MAE	RMSE
RepTree	0,937	23,067	105,718
Random Forest	0,973	18,938	71,253
KNN	0,956	21,120	106,143
Linear Regression	0,811	90,992	172,806
MLP	0,982	23,813	56,555
SVM	0,823	40,737	240,276

Tablo 3.1. incelendiğinde LCOM2 metriği için en yüksek korelasyon katsayısı ve en düşük RMSE değeri MLP algoritması ile elde edilirken en düşük MAE değerinin Rastgele Orman algoritması ile elde edildiği görülmektedir. Bu durumda karar kullanıcıya bırakılmaktadır. Kurulan modelde eğer aralarındaki korelasyon katsayısı hata değerinden daha önemli ise MLP algoritmasının kullanılması gerekirken, daha az hata oranı ile sonuç elde etmek istendiğinde Rastgele Orman algoritmasının kullanılması gerekmektedir. KNN algoritması için 1'den başlanarak farklı k değerleri denenmiştir. K değeri değiştikçe korelasyon katsayısının artışı ve hata oranının azalışı devam ettiği sürece k değeri tek sayılar üzerinden artırılmaya devam edilmiştir. Azalmanın başladığı yerde ise deneme tamamlanmıştır. LCOM2 metriği için KNN algoritması ile en yüksek korelasyon katsayısı ve en düşük hata değeri k=3 durumunda elde edilmiştir. LCOM2 metriği oluşturulan veri kümesinde 0 ile 5749 arasında değer almaktadır. LCOM2 değerleri üzerinden aykırı değer ve uç değer analizi yapılmıştır. Aykırı değer analizi için kullanılan Eş. 3.4'te, uç değer analizi için kullanılan Eş. 3.5'te verilmiştir [48].

$$Q3+OF*IQR < x <= Q3+EVF*IQR \quad (3.4)$$

$$x>Q3+EVF*IQR \quad \text{ve} \quad x<Q1-EVF*IQR \quad (3.5)$$

Q3 : Üçüncü çeyrek değerini göstermektedir. Üçüncü çeyrek veri kümesi küçükten büyüğe doğru sıralandığında elde edilen ortanca değerden büyük olan verilerin ortancasına üçüncü çeyrek denmektedir [49].

IQR: Çeyrekler açıklığını göstermektedir. İlk çeyrek, veri kümesi küçükten büyüğe doğru sıralandığında ortancadan küçük olan değerlerin ortancası olarak tanımlanmaktadır. Üçüncü çeyrek ile ilk çeyreğin farkı çeyrekler açıklığını vermektedir [49].

EVF (Extreme Values Factor) : Kullanıcı tarafından belirlenen ve uç değerler için eşikleri belirleme faktörüdür. Genelde varsayılan 6 olarak kullanılmaktadır [50].

OF (Outlier Factor) : Kullanıcı tarafından belirlenen ve aykırı değerler için eşikleri belirleme faktörüdür. Genelde varsayılan 3 olarak kullanılmaktadır [50].

Yapılan uç değer ve aykırı değer analizi sonucunda 65<x<120 aralığındaki değerler aykırı değer, x>120 değerler ise uç değer olarak belirlenmiştir. Veri kümesinden LCOM2 gerçek değeri bu aralıkta bulunan veriler çıkarılmıştır. Bu aralıkta bulunan toplam 137 değer çıkarılarak, kalan 1052 veri ile aynı makine öğrenmesi teknikleri kullanılarak çalışma tekrarlanmıştır. Aykırı ve uç değerler çıkarıldıktan sonra elde edilen analiz sonuçları Tablo 3.2.'de verilmiştir.

Tablo 3.2. Aykırı Değerler Çıkarıldıktan Sonra LCOM2 Metriği için Regresyon Analizi Sonucu

	R	**MAE**	**RMSE**
RepTree	0,681	5,245	9,677
Random Forest	0,665	5,272	10,057
KNN	0,709	5,080	9,304
Linear Regression	0,607	5,989	10,470
MLP	0,603	6,841	10,653
SVM	0,601	5,886	10,542

Tablo 3.2. incelendiğinde en yüksek korelasyon ve en düşük hata oranlarının KNN algoritması ile elde edildiği görülmektedir. KNN için en iyi sonuçlar k=9 durumunda elde edilmiştir. Gerçek değer ile tahmin edilen değer arasındaki korelasyon 0,709 olarak ölçülmüştür. Bu durum da orta ölçekli korelasyon olduğunu göstermektedir. Uç ve aykırı değerler çıkarıldıktan sonra hata oranlarında azalma meydana gelmiştir. KNN algoritması LCOM2 değerini ortalama 5,080 fark ile tahmin edebilmektedir. Uç değerler çıkarıldıktan sonra LCOM2 metriği 0 ve 1 arasında normalize edilerek aynı deney tekrarlanmıştır ve deney sonucu Tablo 3.3.'te gösterilmiştir.

Tablo 3.3. Aykırı Değerler Çıkarıldıktan Sonra Normalize Edilmiş LCOM2 Metriği için Regresyon Analizi Sonucu

	R	**MAE**	**RMSE**
RepTree	0,681	0,082	0,151
Random Forest	0,665	0,082	0,157
KNN	0,709	0,079	0,145
Linear Regression	0,607	0,094	0,164
MLP	0,603	0,107	0,167
SVM	0,601	0,092	0,165

Tablo 3.3. incelendiğinde en yüksek korelasyon ve en düşük hata değerleri KNN algoritması ile elde edilmiştir. KNN algoritması için en iyi değerler k=9 durumunda elde edilmiştir. Gerçek değer ile tahmin edilen değer arasındaki korelasyon 0,709 olarak hesaplanmıştır. KNN algoritması ile ortalama 0,079 hata ile LCOM2 metriği tahmin edilebilmektedir. Tablo 3.1. ve Tablo 3.2. ile kıyaslandığında hata oranlarındaki büyük azalmanın sebebi LCOM2 değerinin 0 ve 1 arasında normalize edilmiş olmasıdır. Rastgele seçilmiş 40 sınıftan elde edilen gerçek ve tahmin değerleri Tablo 3.4.'te verilmiştir.

Tablo 3.4. Rastgele Seçilmiş 40 Sınıf için KNN Algoritması ile Elde Edilen LCOM2 Metriğine ait Gerçek ve Tahmin Değerleri

Örnek Sayısı	Gerçek Değer	Tahmin Edilen Değer	Hata Değeri
1	0,063	0,197	0,135
2	0,047	0,118	0,071
3	0,141	0,168	0,027
4	0,047	0,139	0,092
5	0	0,095	0,095
6	0,063	0,074	0,012
7	0,125	0,229	0,104
8	0,203	0,366	0,163
9	0,031	0,038	0,007
10	0	0,105	0,105
11	0,047	0,084	0,038
12	0,125	0,212	0,087
13	0	0,113	0,113
14	0,031	0,035	0,004
15	0,313	0,373	0,061
16	0,438	0,443	0,005
17	0	0,037	0,037
18	0	0,060	0,060
19	0,156	0,190	0,034
20	0,156	0,316	0,160
21	0,047	0,090	0,044
22	0,016	0,017	0,002
23	0,063	0,109	0,046
24	0,047	0,137	0,090
25	0	0,118	0,118
26	0	0,016	0,016
27	0,438	0,443	0,005
28	0,031	0,039	0,008
29	0,250	0,273	0,023
30	0	0,136	0,136
31	0	0,109	0,109
32	0,016	0,136	0,120
33	0,125	0,173	0,048
34	0	0,001	0,001
35	0,313	0,424	0,112
36	0,016	0,118	0,102
37	0,391	0,444	0,054
38	0	0,152	0,152
39	0,141	0,255	0,114
40	0	0,082	0,082

Tablo 3.5. TCC Metriği Regresyon Analizi Sonucu

	R	**MAE**	**RMSE**
RepTree	0,632	0,234	0,319
Random Forest	0,601	0,238	0,332
KNN	0,640	0,231	0,317
Linear Regression	0,217	0,362	0,401
MLP	0,279	0,354	0,409
SVM	0,208	0,361	0,409

TCC metriği için elde edilen regresyon analizi sonuçları Tablo 3.5.'te belirtilmiştir. En yüksek korelasyon katsayısı ve en düşük hata değerleri KNN algoritması ile k=13 durumunda elde edilmiştir. Gerçek değer ile tahmin edilen değer arasındaki en yüksek korelasyon 0,640 olarak, en düşük ortalama hata ise 0,231 olarak hesaplanmıştır. Gerçek değer ile tahmin edilen değer arasında orta ölçekli korelasyon vardır. Rastgele seçilen 40 örnek için elde edilen gerçek ve tahmin değerleri Tablo 3.6.'da bulunmaktadır.

Tablo 3.6. Rastgele Seçilmiş 40 Sınıf için KNN Algoritması ile Elde Edilen TCC Metriğine ait Gerçek ve Tahmin Değerleri

Örnek Sayısı	Gerçek Değer	Tahmin Edilen Değer	Hata Değeri
1	0,416	0,531	0,115
2	0,333	0,412	0,079
3	0,318	0,500	0,182
4	0,330	0,352	0,022
5	0,444	0,468	0,024
6	0,333	0,362	0,029
7	0	0,167	0,167
8	0,300	0,310	0,010
9	0,500	0,520	0,020
10	0,261	0,277	0,016
11	0,333	0,512	0,179
12	0,205	0,318	0,113
13	0,353	0,365	0,012
14	0,333	0,372	0,039
15	0,300	0,390	0,090
16	0,333	0,388	0,055
17	0,345	0,410	0,065
18	0,500	0,545	0,045
19	0,385	0,456	0,071
20	0,500	0,515	0,015
21	0,436	0,498	0,062
22	0	0,092	0,092
23	0,238	0,271	0,033
24	0,321	0,467	0,147
25	0,422	0,479	0,057
26	0,242	0,410	0,168
27	0,500	0,515	0,015
28	0,230	0,370	0,140
29	0	0,095	0,095
30	0,285	0,463	0,178
31	0,409	0,453	0,044
32	0,170	0,271	0,101
33	0,428	0,498	0,070
34	0,213	0,393	0,180
35	0,428	0,586	0,158
36	0,333	0,387	0,054
37	0,253	0,300	0,047
38	0,533	0,593	0,060
39	0,393	0,399	0,006
40	0,345	0,471	0,126

Tablo 3.7. LCC Metriği Regresyon Analizi Sonucu

	R	MAE	RMSE
RepTree	0,603	0,259	0,342
Random Forest	0,562	0,266	0,359
KNN	0,584	0,262	0,350
Linear Regression	0,239	0,383	0,417
MLP	0,247	0,374	0,427
SVM	0,234	0,381	0,421

LCC metriği için elde edilen regresyon analizi sonuçlarına göre en yüksek korelasyon ve en düşük hata değerleri REPTree algoritması ile elde edilmiştir. KNN algoritması için en yüksek korelasyon ve en düşük hata değerleri k=5 durumunda elde edilmiştir. LCC metriği için elde edilen sonuçlara göre en yüksek korelasyon 0,603 olarak belirlenmiştir. Gerçek değer ile tahmin edilen değer arasında orta ölçekli bir korelasyon mevcuttur. LCC metriği için rastgele 40 sınıftan elde edilen gerçek ve tahmin değerleri Tablo 3.8.'de verilmiştir.

Tablo 3.8. Rastgele Seçilmiş 40 Sınıf için REPTree Algoritması ile Elde Edilen LCC Metriğine ait Gerçek ve Tahmin Değerleri

Örnek Sayısı	Gerçek Değer	Tahmin Edilen Değer	Hata Değeri
1	0,622	0,637	0,015
2	0,400	0,515	0,115
3	0,333	0,526	0,193
4	0,166	0,379	0,213
5	0,333	0,563	0,230
6	0,535	0,596	0,061
7	0,494	0,515	0,021
8	0,476	0,526	0,050
9	0,400	0,576	0,176
10	0,458	0,496	0,038
11	0,439	0,515	0,076
12	0,333	0,395	0,062
13	0,333	0,569	0,236
14	0,714	0,777	0,063
15	0,400	0,456	0,056
16	0,545	0,561	0,016
17	0,400	0,614	0,214
18	0,012	0,102	0,090
19	0,300	0,382	0,082
20	0,428	0,576	0,148
21	0,416	0,596	0,180
22	0,434	0,563	0,129
23	0,500	0,510	0,010
24	0	0,164	0,164
25	0,424	0,561	0,137
26	0	0,172	0,172
27	0,333	0,373	0,040
28	0	0,178	0,178
29	0,500	0,626	0,126
30	0,458	0,563	0,105
31	0,307	0,421	0,114
32	0,509	0,559	0,050
33	0	0,196	0,196
34	0,500	0,526	0,026
35	0,333	0,376	0,043
36	0,166	0,376	0,210
37	0,300	0,506	0,206
38	0,188	0,299	0,111
39	0,524	0,553	0,030
40	0,358	0,563	0,205

Tablo 3.9. LSCC Metriği Regresyon Analizi Sonucu

	R	MAE	RMSE
RepTree	0,762	0,149	0,232
Random Forest	0,757	0,151	0,235
KNN	0,753	0,152	0,236
Linear Regression	0,382	0,270	0,331
MLP	0,631	0,223	0,294
SVM	0,358	0,248	0,369

LSCC metriği için Tablo 3.9.'da bulunan veriler incelendiğinde en yüksek korelasyon ve en düşük hata değerleri REPTree tekniği ile elde edilmiştir. Korelasyon değeri 0,762 olarak hesaplanmıştır. Güçlü korelasyon aralığına oldukça yakın bir değerdir. Kurulan model sayesinde ortalama 0,149 hata değeri ile LSCC değeri tahmin edilebilmektedir. KNN algoritması için en yüksek korelasyon ve en düşük hata değerleri k=5 durumunda elde edilmiştir. Rastgele seçilmiş 40 sınıfa ait gerçek ve tahmin değerleri Tablo 3.10'da verilmiştir.

Tablo 3.10. Rastgele Seçilmiş 40 Sınıf için REPTree Algoritması ile Elde Edilen LSCC Metriğine ait Gerçek ve Tahmin Değerleri

Örnek Sayısı	Gerçek Değer	Tahmin Edilen Değer	Hata Değeri
1	0,150	0,207	0,057
2	0,222	0,227	0,005
3	0,100	0,183	0,083
4	0,007	0,062	0,055
5	0,100	0,213	0,113
6	0,018	0,065	0,047
7	0,060	0,062	0,002
8	0	0,030	0,030
9	0,025	0,130	0,105
10	0,250	0,284	0,034
11	0,036	0,130	0,095
12	0,083	0,148	0,065
13	0,143	0,175	0,032
14	0,056	0,160	0,105
15	0,038	0,116	0,078
16	0,166	0,227	0,061
17	0,038	0,130	0,092
18	0,200	0,275	0,075
19	0,090	0,207	0,117
20	0,003	0,012	0,009
21	0,166	0,208	0,042
22	0,002	0,054	0,052
23	0,039	0,054	0,015
24	0,021	0,022	0,001
25	0,200	0,296	0,096
26	0,200	0,262	0,062
27	0,020	0,056	0,037
28	0,100	0,190	0,090
29	0,166	0,263	0,097
30	0,055	0,174	0,120
31	0,006	0,085	0,079
32	0,149	0,160	0,011
33	0,080	0,098	0,018
34	0,125	0,160	0,035
35	0,028	0,032	0,004
36	0,100	0,166	0,066
37	0,030	0,056	0,026
38	0,167	0,227	0,061
39	0,333	0,381	0,048
40	0,179	0,207	0,028

4. SONUÇ VE ÖNERİLER

Yapılan çalışmada yazılım kalitesini değerlendirme ölçütlerinden biri olan uyum değerinin makine öğrenmesi yöntemlerinden Rastgele Orman, KNN, REPTree, SVM, MLP ve Doğrusal Regresyon ile tahmin edilmesi sağlanmıştır. Veri kümesi hazır olarak elde edilmemiş, çalışma sürecinde araç kullanılarak üretilmiştir. Açık kaynak kodlu bir projede bulunan 1189 sınıfa ait metot sayısı, öznitelik sayısı ve çeşitli metriklere ait uyum değerleri kullanılarak oluşturulmuştur. Yazılım uyum değerinin elde edilmesi için literatürde bulunan ve kullanılan birçok araç vardır. Bu araçlar uyum değerini farklı şekillerde ölçmektedir. Elde edilen uyum değerine göre sınıf tasarımının değişikliği sağlanmalıdır. Genelde ölçüm yöntemi olarak sınıftaki metotlar ve özniteliklerin birbirleri ile olan yapısal ilişkileri dikkate alınmaktadır. Araçların bazıları ücretli olup, deneme sürümlerinde de istenilen her metrik elde edilememektedir. Dolayısıyla araçları kullanmak zaman alıcı ve daha maliyetli olmaktadır. Bu çalışmada sınıf uyum değerinin daha hızlı, kolay, pratik ve daha az maliyetle gerçek değerine en yakın olacak şekilde tahmin edilmesi sağlanmıştır. Ölçüm için sınıftaki metot ve öznitelik sayısı esas alınmıştır. Bu iki değere göre sınıfın uyum değeri tahmin edilmiştir. Uyum değeri olarak çalışma kapsamında farklı şekillerde ölçüm yapan ve farklı özelliklere sahip LCOM2, TCC, LCC ve LSCC metrikleri seçilmiştir.

Elde edilen sonuçlar kullanılan uyum ölçütüne göre farklılaşmıştır. LCOM2 metriği normalize edilmiş bir metrik olmadığı için alabileceği değerler 0 ile toplam metot çifti sayısı arasındadır. Tez kapsamında oluşturulan veri kümesinde LCOM2 metriği 0 ile 5749 arasında değer almıştır. LCOM2 metriğine ait veri kümesine uç ve aykırı değer analizi yapılmıştır. Elde edilen analiz sonucuna göre 137 değer veri kümesinden çıkarılmıştır. Uç ve aykırı değer analizinden sonra aynı veri kümesine normalizasyon işlemi yapılmıştır. Yapılan tüm analizler sonucu deney aynı makine öğrenmesi algoritmaları ile tekrarlanmıştır. Elde edilen tüm sonuçlar tez çalışmasında ayrı ayrı verilmiştir. LCOM2 metriği için en iyi sonuç ortalama 0,079 hata ile KNN algoritması kullanılarak elde edilmiştir. TCC metriği için de KNN algoritması kullanılarak ortalama 0,231 hata ile elde edilirken, LCC metriğinde ortalama 0,259 hata değeri ile REPTree algoritması kullanılarak elde edilmiştir. Son olarak, LSCC metriğinde en iyi sonuç REPTree algoritması ile ortalama 0,149 hata ile gerçek değere yakın bir tahmin sağlanmıştır. Sonuçlara bakıldığında kurulan modelde en iyi performansın elde edildiği uyum metrikleri LCOM2 ve LSCC'dir. Özellikle LCOM2 için gerçeğe en yakın tahmin sağlanmıştır. Tüm metrikler için gerçek değer ve tahmin edilen değer arasında orta ölçekli korelasyon bulunmaktadır.

Bu çalışmada kullanılan veri kümesi TCC, LCC ve LSCC metrikleri için 1189 sınıf ile, LCOM2 metriğinde ise aykırı ve uç değer analizi sonrası kalan 1052 sınıf ile sınırlıdır. Bu sayı gelecek çalışmada artırılıp, farklı alanlara(domain) ait sınıflar da dahil edilerek elde edilen tahmin değerinin gerçeğe olan yakınlığının artırılması hedeflenmektedir.

KAYNAKLAR

[1] S. A. Koçak, G. I. Alptekin, and A. B. Bener, "Bir Yazılımın Çevre ve Kalite Açısından Değerlendirilmesi için Önerilmiş Karar Alma Modeli,", Ulusal Yazılım Mühendisliği Sempozyumu, 2015, pp. 418–429.

[2] A. Obisat, F. M. Alhalhouli, A. Z. T., and T. I., & Alshabatat, T. E., "Review of literature on software quality," *World of Computer Science and Information Technology Journal*, vol. 8, no. 5, pp. 32–42, 2018.

[3] B. Erkal, "Ölçüt tabanlı yazılım hata kestirim yaklaşımlarının incelenmesi ve yeni bir yazılım hata kestirim önerisi," Başkent Üniversitesi, Ankara, Türkiye, 2020.

[4] S. Tarwani and A. Chug, "Assessment of optimum refactoring sequence to improve the software quality of object-oriented software," *J. Inf. Optimiz. Sci.*, vol. 41, no. 6, pp. 1433–1442, 2020.

[5] S. Tiwari and S. S. Rathore, "Coupling and cohesion metrics for object-oriented software: A systematic mapping study," in *Proceedings of the 11th Innovations in Software Engineering Conference*, 2018.

[6] B. MohanGoel and P. Kumar Bhatia, "Analysis of reusability of object-oriented system using CK metrics," *Int. J. Comput. Appl.*, vol. 60, no. 10, pp. 32–36, 2012.

[7] J. A. Dallal, "Improving Object-Oriented Lack-ofCohesion Metric by Excluding Special Methods,", *In Proceedings of the 10th WSEAS International Conference on Software Engineering Parallel and Distributed Systems*, 2011, pp. 124–129.

[8] H. Izadkhah and M. Hooshyar, Eds., *Class Cohesion Metrics for Software Engineering: A Critical Review* *, vol. 25, no. 1(73). Computer Science Journal of Moldova, 2017.

[9] J. Al Dallal and L. C. Briand, "A precise method-method interaction-based cohesion metric for object-oriented classes," *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 2, pp. 1–34, 2012.

[10] O. Vogel, I. Arnold, A. Chughtai, and T. Kehrer, *Software architecture: A comprehensive framework and guide for practitioners*, 2011th ed. Berlin, Germany: Springer, 2011.

[11] S. A. M. Domingos, “STUDY ON THE RELATIONSHIPS BETWEEN COHESION AND COUPLING METRICS ON FAULT PREDICTION IN OBJECT ORIENTED SYSTEMS,” Federal University of Santa Catarina , Florianópolis, Brazil, 2018.

[12] A. Marcus and D. Poshyvanyk, “The conceptual cohesion of classes,” in *21st IEEE International Conference on Software Maintenance (ICSM’05)*, 2005.

[13] J. M. Bieman and B.-K. Kang, “Cohesion and reuse in an object-oriented system,” *Softw. Eng. Notes*, vol. 20, no. SI, pp. 259–262, 1995.

[14] S. R. Chidamber and C. F. Kemerer, “Towards a metrics suite for object oriented design,” *SIGPLAN not.*, vol. 26, no. 11, pp. 197–211, 1991.

[15] W. Li and S. Henry, “Maintenance metrics for the object oriented paradigm,” in *[1993] Proceedings First International Software Metrics Symposium*, 2002.

[16] M. Saadati and H. Motameni, “Measuring cohesion and coupling of object-oriented systems,” *J. Math. Comput. Sci.*, vol. 09, no. 02, pp. 149–156, 2014.

[17] B. Henderson-Sellers, L. L. Constantine, and A. I. M. Graham, “Coupling and cohesion (towards a valid metrics suite for objectoriented analysis and design),” *Object Oriented Systems*, vol. 3, no. 3, pp. 143–158, 1996.

[18] M. Alzahrani and A. Melton, “Defining and validating a client-based cohesion metric for object-oriented classes,” in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, 2017.

[19] A. Yusuf and M. Hammad, “An approach to automatically measure and visualize class cohesion in object-oriented systems,” in *2020 International Conference on Decision Aid Sciences and Application (DASA)*, 2020.

[20] A. Rathee and J. K. Chhabra, “Improving cohesion of a software system by performing usage pattern based clustering,” *Procedia Comput. Sci.*, vol. 125, pp. 740–746, 2018.

[21] B. Priyambadha, T. Katayama, Y. Kita, H. Yamaba, K. Aburada, and N. Okazaki, “Utilizing the similarity meaning of label in class cohesion calculation,” *J. Robot. Netw. Artif. Life*, vol. 7, no. 4, p. 270, 2020.

[22] S.-H. Tee, “Measuring class cohesion using mutant methods,” *Softw. Eng. Notes*, vol. 35, no. 6, pp. 1–4, 2010.

[23] A. M. Altaie, “Designing and implementing a tool for measuring cohesion and coupling of Object-Oriented Systems,” *Turkish Journal of Computer and Mathematics Education*, vol. 13, no. 2, pp. 368–375, 2022.

[24] H. Razafimahatratra, T. Mahatody, J. P. Razafimandimby, and T. M. B. Andriamahazomandimby, “Automatic detection of the cohesion type in the UML sequence diagram,” in *2018 19th International Carpathian Control Conference (ICCC)*, 2018.

[25] D. Singh and S. K. Khalsa, “Measuring cohesion and coupling of sequence diagram using program slicing,” in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 84–88.

[26] R. Wanjiku and G. Okeyo, “Scoped class cohesion metric for software process assessment,” *Int. J. Comput. Sci. Issu.*, vol. 13, no. 2, pp. 12–18, 2016.

[27] V. Jain and A. Gupta, “Lack of Conceptual Cohesion of Methods: A new alternative to Lack Of Cohesion of Methods,” in *Proceedings of the 8th India Software Engineering Conference*, 2015.

[28] C.-Y. Chen, K.-Y. Tai, and S.-S. Chong, “Quality evaluation of structural design in software reverse engineering: A focus on cohesion,” *IEEE Access*, vol. 9, pp. 109569–109583, 2021.

[29] Manju and P. K. Bhatia, “Measurement of Dynamic Cohesion using Aspect Oriented Approach,” *INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS*, vol. 6, no. 2, pp. 438–442, 2019.

[30] “Freeplane,” *SourceForge*, 06-Mar-2022. [Online]. Available: sourceforge.net/projects/freeplane/. [Accessed: 29-Jul-2022].

[31] M. M. Islam, H. Iqbal, M. R. Haque, and M. K. Hasan, “Prediction of breast cancer using support vector machine and K-Nearest neighbors,” in *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, 2017.

[32] D. C. Lakshmi, “Comparative Analysis of Random Forest, REP Tree and J48 Classifiers for Credit Risk Prediction,”, *International Conference on Communication, Computing and Information Technology* , 2015, pp. 30–36.

[33] H. Yılmaz, “RANDOM FORESTS YÖNTEMİNDE KAYIP VERİ PROBLEMİNİN İNCELENMESİ VE SAĞLIK ALANINDA BİR UYGULAMA,” ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ, ESKİŞEHİR, 2014.

[34] İ. Gör, “Çok katmanlı algılayıcı yapay sinir ağı ile lineer diferansiyel denklem sisteminin çözümü,” 2016, pp. 738–745.

[35] M. T. Bilişik, “Destek vektör makinesi, çoklu regresyon ve doğrusal olmayan programlama ile perakendecilik sektöründe gelir yönetimi için dinamik fiyatlandırma,” ,*11. Üretim Araştırmaları Sempozyumu* 2011, pp. 785–799.

[36] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 6th ed. Standards Information Network, 2021.

[37] S. Chatterjee and A. S. Hadi, *Regression Analysis by Example*, 5th ed. Hoboken, NJ: Wiley-Blackwell, 2015.

[38] E. G. Kulkarni and R. B. Kulkarni, “WEKA Powerful Tool in Data Mining,” *International Journal of Computer Applications*, pp. 10–15, 2016.

[39] Z. Markov and I. Russell, “An introduction to the WEKA data mining system,” *SIGCSE bull.*, vol. 38, no. 3, pp. 367–368, 2006.

[40] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: An update,” *SIGKDD Explor.*, vol. 11, no. 1, pp. 10–18, 2009.

[41] S. Srivastava, “Weka: A tool for data preprocessing, classification, ensemble, clustering and association rule mining,” *Int. J. Comput. Appl.*, vol. 88, no. 10, pp. 26–29, 2014.

[42] A. Botchkarev, “A new typology design of performance metrics to measure errors in machine learning regression algorithms,” *Interdiscip. J. Inf. Knowl. Manag.*, vol. 14, pp. 045–076, 2019.

[43] V. Bewick, L. Cheek, and J. Ball, “Statistics review 7: Correlation and regression,” *Crit. Care*, vol. 7, no. 6, pp. 451–459, 2003.

[44] D. Edelmann, T. F. Móri, and G. J. Székely, “On relationships between the Pearson and the distance correlation coefficients,” *Stat. Probab. Lett.*, vol. 169, no. 108960, p. 108960, 2021.

[45] C. J. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Clim. Res.*, vol. 30, pp. 79–82, 2005.

[46] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature,” Geosci. Model Dev., vol. 7, no. 3, pp. 1247–1250, 2014.

[47] C. H. Feng et al., “Voxel-level classification of prostate cancer on magnetic resonance imaging: Improving accuracy using four-compartment restriction spectrum imaging,” J. Magn. Reson. Imaging, no. jmri.27623, 2021.

[48] X. Li, S. M. Mousavi, B. Dadashova, D. Lord, and B. Wolshon, “Toward a crowdsourcing solution to identify high-risk highway segments through mining driving jerks,” *Accid. Anal. Prev.*, vol. 155, no. 106101, p. 106101, 2021.

[49] A. D. Azcel and J. Sounderpandian, *Complete Business Statistics*, 7th ed. Maidenhead, England: McGraw Hill Higher Education, 2008.

[50] “InterquartileRange,” *Sourceforge.io*, 28-Jan-2022. [Online]. Available: weka.sourceforge.io/doc.dev/weka/filters/unsupervised/attribute/InterquartileRange.html. [Accessed: 04-Aug-2022].