T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

NESNE YÖNELİMLİ YAZILIM ÖLÇÜTLERİNİN UZUN ÖMÜRLÜ YAZILIMLARDA HATA ORANLARINI GÖSTERME DÜZEYİNİN BELİRLENMESİ

Özge Esen ÇOLAK

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Danışman

Dr.Öğr.Üyesi Yunus Emre SELÇUK

Şubat, 2024

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

NESNE YÖNELİMLİ YAZILIM ÖLÇÜTLERİNİN UZUN ÖMÜRLÜ YAZILIMLARDA HATA ORANLARINI GÖSTERME DÜZEYİNİN BELİRLENMESİ

Özge Esen ÇOLAK tarafından hazırlanan tez çalışması 22.02.2024 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Dr.Öğr.Üyesi Yunus Emre SELÇUK
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Dr.Öğr.Üyesi Yunus Emre SELÇUK, Danışman
Yıldız Teknik Üniversitesi _____

Prof. Dr. Mehmet Sıddık AKTAŞ, Üye
Yıldız Teknik Üniversitesi _____

Prof. Dr. Selim AKYOKUŞ, Üye
İstanbul Medipol Üniversitesi _____

Danışmanım Dr.Öğr.Üyesi Yunus Emre SELÇUK sorumluluğunda tarafımca hazırlanan “Nesne Yönelimli Yazılım Ölçütlerinin Uzun Ömürlü Yazılımlarda Hata Oranlarını Gösterme Düzeyinin Belirlenmesi” başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Özge Esen ÇOLAK

İmza

Aileme
ve
öğretmenlerime

TEŞEKKÜR

Tez çalışmamı tamamlamamın ardında, birçok kişinin katkısı ve destekleri bulunmaktadır. Bu süreçte endişelerimin üstesinden gelmemi sağlayan, cesaretlendiren herkese teşekkürlerimi sunarım. En başta tez danışmanım Yunus Emre SELÇUK, dikkatimin dağıldığı, pes etmeye yaklaştığım ve zorlandığım her anda bana rehberlik etti. Güçlü iletişimi, fikir, bilgi ve deneyimi sayesinde bu çalışmaya ulaşmamızı sağladı. Ayrıca ders döneminde ve tez çalışmamın başında bana destek veren Ferkan YILMAZ'a da değerli katkıları ve emeği için teşekkür borçluyum. Tez sürecinde benimle manevi olarak ilgilenen ailem ve sevdiklerime en başta annem Ayfer UÇKAN'a bana bu dünyadaki varlığın mesleğine duyduğun saygı ve insanlığa sağladığın katkıdan geldiğini gösterdiği için teşekkür ederim.

Eğitim hayatım boyunca bilgi ve birikimiyle bana ışık tutan tüm öğretmenlerime, son olarak tüm meslektaşlarıma değerli görüşleri ve eleştirileri için teşekkür ederim.

Özge Esen ÇOLAK

İÇİNDEKİLER

SİMGE LİSTESİ

U_{all}	Birileşim
ϕ	Boş Küme
$\geq$	Büyük Eşittir
$>$	Büyüktür
$Depth(C_i)$	Derinlik
$\neq$	Eşit Değildir
$=$	Eşittir
$\cap$	Kesişim
KLOC	Kilo Lines of Code
$\leq$	Küçük Eşittir
$<$	Küçüktür
max	Maksimum
$\|\|$	Mutlak Değer
R^2	R squared
$\sum_{i=1}^{n} Wi$	Toplam Sembolü

KISALTMA LİSTESİ

Adj	Adjusted
AIC	Akaike Information Criterion
API	Application Programming Interface
BIC	Bayesian Information Criterion
CBO	Coupling Between Objects
CBO*	Coupling Between Objects – Modified
CK	Chidamber and Kemerer
Coef	Coefficient
Cond	Condition
Df	Dataframe
DD	Defect Density
DIT	Depth Inheritance Tree
Err	Error
HDFS	Hadoop Distributed File System
IDE	Integrated Development Environment
JB	Jarque-Bera
KPI	Key Performance Indicator
LCC	Loose Class Cohesion
LCOM	Lack of Cohesion of Methods
LOC	Lines Of Code
mae	Mean Squared Error
NOC	Number of Children
NOSI	Number of Static Invocations
OLS	Ordinary Least Squares
RFC	Response for a Class
Std	Standard Deviation
SWT	Standard Widget Toolkit
TCC	Tight Class Cohesion
WMC	Weight Method Class
YARN	Yet Another Resource Negotiator

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Nesne Yönelimli Yazılım Ölçütlerinin Uzun Ömürlü Yazılımlarda Hata Oranlarını Gösterme Düzeyinin Belirlenmesi

Özge Esen ÇOLAK

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Yüksek Lisans Tezi

Danışman: Dr.Öğr.Üyesi Yunus Emre SELÇUK

Nesne Tabanlı Programlama ilkeleri, yazılım dünyası tarafından uzun süredir benimsenen ve uygulanan prensiplerdir. Bu prensipler, kalite, sürdürülebilirlik ve karmaşıklık gibi önemli ölçümlerle değerlendirilmiş ve uzun vadeli yazılım projelerinde bakım ve standartlaşma planlamasında kritik bir rol üstlenmiştir. Chidamber ve Kemerer (CK) metrikleri, Nesne Tabanlı Programlamadaki karmaşıklığı değerlendirmek için temel metrikler sunar. Bu metrikler, yalnızca değerli bilgi kaynakları sağlamakla kalmaz, aynı zamanda proje planlamasına yeni yaklaşımlar getirir. Benzer şekilde sürümler boyunca karşılaşılan hata sayıları, geliştirme ve eklenen yeni özellikler de proje hakkında önemli bilgi kaynaklarıdır. Öncelikle elde edilen kaynaklar birleştirilerek bir veri seti oluşturulmuştur. Veri seti üzerinde lineer regresyon modeli ve korelasyon analizi çalışması gerçekleştirilmiştir. Lineer regresyon modeli metot düzeyinde yapılan modellerin

diğer yaklaşımlara oranla dağa uygun modelleme sağlandığı ve Eclipse SWT projesinin metot düzeyinde yapılan analizde %60'ın üzerinde bir uygunluk oranı sağladığı gözlemlenmiştir. Korelasyon analizi sonucunda elde edilen sonuçlar ve sürüm takip sisteminden elde edilen veriler üç farklı yaklaşım ile incelenmiştir. İlk yaklaşım, yıllar içindeki CK metrik değerlerinin evrimine odaklanarak, CBO, WMC ve LCOM metriklerinde ortak bir artış oranı ortaya koymuştur. Başka bir yaklaşımda, metrikler ile hata sayısı arasındaki ilişki Spearman korelasyon katsayısı kullanılarak değerlendirilmiş ve LCOM parametresinin hata sayısı üzerinde en etkili parametrelerden biri olduğu belirlenmiştir. Son olarak, DD ve WMC başına hata sayısı metriklerinin benzer dağılımlar sergilediği gözlemlenmiştir.

Anahtar Kelimeler: CK metrikleri, açık-kaynak kodlu yazılım, spearman korelasyonu, lineer regresyon

ABSTRACT

Determining the Level of Indicating Bug Rates in Long-Lived Software Based on Object-Oriented Software Metrics

Özge Esen ÇOLAK

Department of Computer Engineering

Master of Science Thesis

Supervisor: Dr. Yunus Emre SELÇUK

Object-Oriented Programming (OOP) principles are long-standing and widely adopted principles in the software world. These principles have been evaluated based on important metrics such as quality, sustainability and complexity, playing a critical role in maintenance and standardization planning for long-term software projects. Chidamber and Kemerer (CK) metrics provide fundamental metrics to assess complexity in Object-Oriented Programming. These metrics not only provide valuable sources of information but also introduce new approaches to project planning. Similarly, bug counts encountered throughout versions and newly added features are significant sources of information about the project. Initially, the obtained resources were combined to create a dataset. A linear regression model and correlation analysis were conducted on the dataset. The linear regression model showed that models made at the method level provide more suitable modeling compared to other approaches, and Eclipse SWT project achieved an over 60% fit

rate in the analysis conducted at the method level. The results of the correlation analysis and data obtained from the version tracking system were examined using three different approaches. In the first approach, focusing on the evolution of CK metric values over the years revealed a common increase rate in CBO, WMC, and LCOM metrics. Another approach evaluated the relationship between metrics and bug counts using the Spearman correlation coefficient, determining that LCOM is one of the most effective parameters on bug counts. Lastly, it was observed that bug counts per DD and WMC metrics exhibited similar distributions.

Keywords: CK metrics, open-source software, spearman correlation, linear regression

1
GİRİŞ

1.1 Amaç ve Kapsam

Açık kaynak projeleri, günlük hayatımızda sıkça kullandığımız yazılım ürünlerinde, araştırma merkezlerinde ve ticari yazılımlarda kullanılan vazgeçilmez projeler haline geldi. Açık kaynak projelerinin felsefesi 1990'lara dayanmasına rağmen, büyük veri işleme, bulut teknolojileri ve yapay zeka araçlarındaki gelişmelerle 2010 yıllarında artış gösterdi ve özellikle Apache Vakfı gibi kuruluşların da desteğiyle büyüdü. Açık kaynak kodun en büyük avantajları, çoğu yazılım geliştiricisinin gönüllü katkılarda bulunarak yüksek motivasyonla çalışması, geliştirme ekibinin yeni çözümler sunmak için yeterli zamanının olması ve farklı kullanım amaçları için özelleştirilebilmesidir.

Ancak, açık kaynak ürünleri aynı zamanda belirli bir proje planının olmaması ve çoğu projede net bir ilerleme vizyonunun bulunmaması gibi dezavantajları da beraberinde getirir. Bu problemlerin en belirginlerinden biri, farklı yazılım geliştiricilerinin farklı geliştirme stillerine sahip olmaları ve genel standart ve yapıya uyum yeteneklerinin farklılık göstermesidir. Özellikle geniş bir kullanım ağına sahip uzun vadeli projeler söz konusu olduğunda, projenin gelişimi ve değişiminde her parametre büyük bir fark yaratır. Dahası, yazılım büyüdükçe, yeniden yapılandırma süreci daha karmaşık hale gelir.

Bu çalışmada, uzun yıllardır açık kaynak olarak geliştirilen Hadoop (2006), Eclipse SWT (2001) ve Kafka (2011) gibi nesne yönelimli programlama tabanlı projelerin metrikleri incelendi. Sürümler boyunca CK metrikleri ve hata oranları açısından kalite ve güvenilirlik kriterlerindeki değişimler izlendi. Bu metriklere sürüm hata sayılarını dahil ederek bir veri seti oluşturuldu. Veri seti, istatistiksel yöntemler kullanılarak analiz edildi ve bu metriklerdeki yeni özelliklerin, iyileştirmelerin ve hata düzeltmelerinin etkisi analiz edildi.

Uzun süredir açık kaynak olarak geliştirilen ve hala güncel çalışmalarda olan üç ürün üzerine bir inceleme yapıldı. Bunlardan ilki, büyük miktarda veriyi depolamak, işlemek ve dağıtmak için kullanılan bir çerçeve olan Hadoop projesidir. Bu proje, Java dilinde yazılmış olup, 2017'de Hadoop 3.0'da önemli değişiklikler ve performans iyileştirmeleri içeriyor. Hadoop, Hadoop MapReduce, Hadoop HDFS, YARN ve Hadoop Common gibi modüllerden oluşur. Eclipse SWT, Eclipse IDE'nin kullanıcı arayüzü bileşenlerini oluşturmak için kullanılan bir araç setidir. SWT, Java dilinde yazılmıştır. Diğer bir örnek ise gerçek zamanlı veri akışlarını işlemek ve entegre etmek için kullanılan Kafka'dır. Büyük ölçüde Scala ve Java dillerinde yazılmıştır.

Güncel zamana göre değerlendirme yapıldığında Apache Hadoop'un 2014-2023 yılları arasındaki 49 versiyonu, Eclipse SWT'nin 2003-2018 yılları arasındaki 18 versiyonu ve Kafka'nın 2013-2023 yılları arasındaki 61 versiyonundan elde edilen veriler incelenmiştir. Bu sebeple, elde edilen veri seti yıllar boyunca devam eden bir birikimin ürünü olup projeler hakkında benzer eğilimlerin var olup olmadığını gözlemleme şansı verir.

1.2 Literatür İncelemesi

Hedef proje örnekleri uzun zaman içerisinde çok fazla versiyon içeren projeleri ve hala oldukça aktif kullanılan nesne tabanlı yazılımları incelemek olunca Java dilinde yazılmış olması tercih edildi. Analiz edilecek projeler belirlendikten sonra hangi metriklerin nesne tabanlı yazılım projelerinde kullanıldığını anlamak için çeşitli çalışmalar incelendi. CK metrikleri bilinen halini alana kadar genellikle yazılım karmaşıklığını, bağımlılığını ve modüler yapısını değerlendirmek için farklı ölçümler mevcuttu. Ancak, S. R. Chidamber ve C. F. Kemerer (1994)'e göre metriklerin bazıları eksik teorik temele veya belirli ölçüm prensiplerine dayanıyordu. CK metrikleri, 1994 yılında yazılım geliştirme alanındaki süreç iyileştirmesine odaklanan nesne yönelimli programlama için bir dizi yeni yaklaşımın ortaya çıkmasıyla başladı. Önceki metriklerin eksikliklerini gidermek ve özellikle nesne yönelimli tasarımın ölçülmesi için daha kapsamlı teorik temele dayanan bir yaklaşım sunmak için geliştirilmiştir. Yazılım tasarımının kalitesini ölçmek ve iyileştirmek için kullanılır. Bunge'nin ontolojisin referans alır ve tasarımdaki altı farklı boyutu ölçmek için geliştirilmiştir. Bunlar bağlantı,

bağdaşıklık, soyutlama, miras, metot sayısı ve veri sayısıdır (Chidamber ve Kemerer, 1994).

CK metrikleri hem bağımlılık hem de metot düzeyinde metrik çıkarımı sağlarken Abreu ve Melo (1996) sistem düzeyinde geniş bir metrik yelpazesi önermiştir.

"Nesne Yönelimli Metriklerin Açık Kaynak Yazılımlarda Hata Tahmini Üzerindeki Etkisinin Doğrulanması" adlı 2005 yılında Tibor Gyimo´thy, Rudolf Ferenc, ve Istva´n Siket'in yaptığı bir çalışma mevcuttur (Gyimo´thy, Ferenc ve Siket, 2005). Bu çalışma C, C++ dillerinde yazılmış açık kaynak kodlu bir yazılım olan Mozilla'nın yedi versiyonu üzerinde benzer bir çalışma yapmayı hedeflemişlerdir. İstatistiksel yöntemleri, makine öğrenmesi ve yapay sinir ağlarını kullanmışlar ve çoklu regresyon modelinde %43 lük bir başarı sağlayabilmişler. Başarısız tahminlerin ise bu çalışmada da en kritik süreçlerden biri olan verilerin doğru etiketlenememesi yani sınıflar ile hataların doğru ilişkilendirilememiş olmasından kaynaklandığını öngörmektedirler. Bizim çalışmamıza benzer olarak metrikler arası korelasyon analizi gerçekleştirmişler detaylı kıyaslamaya korelasyon analizi bölümünde yer verilmiştir. Lojistik regresyon yöntemini kullanarak sınıf düzeyinde hatalı ve hatasız sınıfları ayırarak hatalı sınıf tahmini yapmaya çalışmışlar ve %72.57'ye varan bir başarı oranına yaklaşmışlardır. Benzer şekilde Lineer regresyonda da %43'lük bir başarı oranı yakalayabilmişlerdir. Karar ağaçları ve yapay sinir ağlarıyla da ilerlettikleri çalışmada yöntemlerin benzer sonuçlar verdiğini, CBO metriğinin hata eğilimini tahmin etmede etkili olduğunu, LOC metriğinin hızlı tahminlerde başarılı olduğunu ancak detaylı analizlerde çok değişkenli modellerin daha başarılı sonuçlar ürettiğini ve DIT ve NOC'nin hata tahmininde uygun olmadığını öner sürmüşlerdir (Gyimo´thy vd., 2005).

Mohammed Alqmase, Mohammad Alshayeb ve Lahouari Ghouti'nin literatüre kattığı "Yazılım Metrikleri için Sınır Değerlerini Çıkarma Çerçevesi" adlı çalışma yazılım metriklerinin sınır değerlerini çıkartmak amacı ile bir çerçeve sunmaktadır (Alqmase, Alshayeb ve Ghouti, 2019). Bu çalışma şu açıdan önemlidir ki topladığımız metriklerin değerlerinin yanı sıra metrik değeri belirli bir değerin üzerinde olan sınıf ve metotlar da hata oranı ile olan ilişkisini belirlemede yardımcı olacak bilgilerden birine dönüşebilir. Sözgelimi CBO değerinin sınıf içerisinde 22 değerinden yüksek olması toplamdaki CBO değeri yüksek sınıf sayısına bir

eklerken genel CBO ortalamasını da yükselten bir katkı sağlar. Bu iki yaklaşım da hata değerlerini tahmin etmede farklı bir hesaplama yöntemi sunacaktır.

"Yazılım Geliştirmede Hata Sayısını Azaltmak İçin Tasarım Metriğinin Değerlendirilmesi" adlı çalışmada ise biri tek proje diğeri ise 12 modülden oluşan 3 proje içeren 2 farklı şirketin toplam hata sayıları ile CK metrikleri arasındaki ilişki analiz edilmiştir (Qureshi, M. R. J., ve Qureshi, W. A., 2012). Bizim çalışmamıza en benzeyen tarafı Java dili kullanılarak yazılmış olmasıdır. Analiz sonuçlarına göre CBO, DIT, NOC, WMC ve RFC metriklerinin hata sayısı üzerinde anlamlı bir etkisinin olmadığını yalnızca LCOM metriğinin hata sayısı üzerinde anlamlı bir etkiye sahip olduğunu öne sürmüşlerdir (Qureshi, M. R. J., ve Qureshi, W. A., 2012).

Hata oranlarından bağımsız uzun soluklu projelerde metrik değerlerinin belirlenmesine yönelik "Açık Kaynak Uygulamalarda Yazılım Kalite Metriklerinin Uzunlamasına Değerlendirmesi" adlı bir çalışma da mevcuttur (Molnar, Neam ţu, Motogna, 2019). Bu çalışma 18 yıla aşkın bir geliştirme sürecinde 3 farklı açık kaynak kodlu yazılımı incelemiştir. Yazılım kalite modellerini metrik değerlere dayandırmak amacıyla bu ilişkilerin varlığını gözlemlemeye çalışmışlar ve güçlü metrik çiftlerini tanımlamaya çalışmışlardır (Molnar vd., 2019).

"Açık Kaynak Sistemlerde Nesne Yönelimli Metriklerin Kabul Edilebilir Risk Seviyelerine Yönelik Bir Nicel Araştırma" ise CK metriklerini Eclipse projesi için inceleyerek sınıf bazında hata tespitine bağlı olarak metriklerin sınır değerlerinin belirlenmesine dair bir çalışma yürütmüştür (Shatnawi, 2010). Çalışma sonucunda CBO=9, RFC=40 ve WMC=20 sınır değerleri olarak önerilmiştir ve sonuçların yazılım geliştiricilerinin önerilerinden daha başarılı sonuçlar verdiğini Eclipse 2.1 sürümüyle test ederek doğrulamaya çalışmıştır (Shatnawi, 2010).

"CK Metriklerinin Nesne Yönelimli Tasarım Ölçümü İçin Doğrulanması" adlı çalışmada ise 6 Java projesi CK metrik takımını kullanarak sistem kalitesini ve yeniden kullanılabilirlik, anlaşılabilirlik, test edilebilirlik, bakım yapılabilirlik gibi farklı kalite parametrelerini tersine etkileyebilecek olası tasarım hatalarını belirlemek için analiz edilmiştir (Kulkarni, Kalshetty ve Arde, 2010). Analiz sonuçlarına göre WMC, LCOM ve CBO gibi sınıf düzeyindeki metriklerin birbirleri ile güçlü bir ilişki içinde olduğu, yüksek WMC değerine sahip sınıfların

yüksek LCOM ve RFC değerlerine sahip olduğu gözlemlenmiştir. Bu sebeple sistem karmaşıklığını kontrol etmek için WMC, RFC ve LCOM'a odaklanmamız gerektiğini önermiştir. 6 sistemde de DIT ve NOC değerlerinin düşük olduğu dolayısıyla miras özelliğinin düzgün kullanılmadığı öne sürülmüştür (Kulkarni vd., 2010). Bu çalışma bize bazı CK metriklerinin hata oranlarıyla negatif ilişki sergileyebileceği fikrini sunmuştur. Metriklerin ideal değerin üstünde olması veya altında olması kod karmaşıklığını benzer şekilde arttırabilir.

Son olarak "Yazılım hata tahmin metrikleri: Bir sistemli literatür incelemesi" adlı çalışma hata tespitinde kullanılan geleneksel ve nesne tabanlı metrikleri inceleyerek geçmişe yönelik çalışmaların bir değerlendirmesini yapmıştır (Radjenović vd., 2013). Değerlendirme sonucunda "Nesne yönelimli metrikler hata tahmini için kullanışlı mıdır?" sorusuna karşı sunduğu bazı tespitler dikkat çeker. Bunlar;

1. CK metrikleri OO metrikleri arasında sık kullanılan ve başarılı olanlardır, ancak tüm CK metrikleri aynı başarıyı göstermemiştir.
2. Yapılan çalışmalara göre en başarılı metrikler CBO, WMC ve RFC iken LCOM etkili bulunamamıştır. LCOM hataları tespit etmede yeterince başarılı değildir.
3. DIT ve NOC güvenilmez olarak rapor edilmiştir ve NOC pozitif veya negatif anlam göstermiştir. (Radjenović vd., 2013)

Nesne tabanlı programlama ortaya çıktığından ve CK metrikleri oluşturulduğundan beri farklı diller, projeler ve yöntemler kullanılarak pek çok analiz çalışması yapılmıştır. En temelde kod kalitesi ölçütü olarak söz edilse de asıl amaç hata sayısını azaltmak veya hata çözümünde harcanan eforu azaltmaktır. Yıllarca farklı dillerle ve projelerle karşımıza çıkan bu çalışmalardaki önermeleri bu kez Apache Hadoop ve Eclipse SWT projeleri ile kıyaslayarak ele almış olacağız.

1.3 Hipotez

Çalışma dahilinde elde edilen veri seti iki kategoriden oluşmaktadır. İlki Java sınıflarından elde edilen CK ve farklı ölçüm koşulları ile elde edilen toplam 13 statik metrik, ikincisi ise metot düzeyinde elde edilen 8 metriktir. Çalışma boyunca farklı yöntemlerle yapılan analizler bu 21 metriğin sürümlerden elde edilen hata oranlarıyla ilişkisini aşağıdaki hipotezlere dayandırmaktadır.

N1→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen CBO metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen CBO metriği hata sayısını etkilemektedir.

N2→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen CBO değiştirilmiş metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen CBO değiştirilmiş metriği hata sayısını etkilemektedir.

N3→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen FANIN metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen FANIN metriği hata sayısını etkilemektedir.

N4→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen FANOUT metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen FANOUT metriği hata sayısını etkilemektedir.

N5→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen WMC metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen WMC metriği hata sayısını etkilemektedir.

N6→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen DIT metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen DIT metriği hata sayısını etkilemektedir.

N7→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen NOC metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen NOC metriği hata sayısını etkilemektedir.

N8→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen LCOM metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen LCOM metriği hata sayısını etkilemektedir.

N9→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen LCOM* metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen LCOM* metriği hata sayısını etkilemektedir.

N10→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen TCC metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen TCC metriği hata sayısını etkilemektedir.

N11→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen LCC metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen LCC metriği hata sayısını etkilemektedir.

N12→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen LOC metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen LOC metriği hata sayısını etkilemektedir.

N13→ **H_0 (Null Hipotezi):** Sınıf verilerinden elde edilen sayısal versiyon metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Sınıf verilerinden elde edilen sayısal versiyon metriği hata sayısını etkilemektedir.

N14→ **H_0 (Null Hipotezi):** Metot verilerinden elde edilen CBO metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Metot verilerinden elde edilen CBO metriği hata sayısını etkilemektedir.

N15→ **H_0 (Null Hipotezi):** Metot verilerinden elde edilen CBO* metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Metot verilerinden elde edilen CBO* metriği hata sayısını etkilemektedir.

N16→ **H_0 (Null Hipotezi):** Metot verilerinden elde edilen FANIN metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Metot verilerinden elde edilen FANIN metriği hata sayısını etkilemektedir.

N17→ **H_0 (Null Hipotezi):** Metot verilerinden elde edilen FANOUT metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Metot verilerinden elde edilen FANOUT metriği hata sayısını etkilemektedir.

N18→ **H_0 (Null Hipotezi):** Metot verilerinden elde edilen WMC metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Metot verilerinden elde edilen WMC metriği hata sayısını etkilemektedir.

N19→ **H_0 (Null Hipotezi):** Metot verilerinden elde edilen RFC metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Metot verilerinden elde edilen RFC metriği hata sayısını etkilemektedir.

N20→ **H_0 (Null Hipotezi):** Metot verilerinden elde edilen LOC metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Metot verilerinden elde edilen LOC metriği hata sayısını etkilemektedir.

N21→ **H_0 (Null Hipotezi):** Metot verilerinden elde edilen sayısal versiyon metriği hata sayısını etkilememektedir.

H_1(Alternatif Hipotez): Metot verilerinden elde edilen sayısal versiyon metriği hata sayısını etkilemektedir.

Hipotezler en temelde H_0 hipotezi olarak nitelendirilen ilgili metrik değeriyle hata oranı arasında bir ilişki olmadığına dayanmaktadır. Hata sayısıyla pozitif veya negatif ne yönde bir ilişki sergileyeceği tamamen tahminidir. Bu metriklerden herhangi biri tek başına bir anlam ifade etmezken belirli kombinasyonları hata tespitinde daha sağlıklı sonuçlar verebilir.

2
ANALİZ ÇALIŞMALARI

2.1 Metriklerin Belirlenmesi

CK metrikleri hem sınıf hem de metot düzeyinde çıktı sağladığı için metrikler bu iki açıdan da ele alınmıştır. CK metrik elde etme aracı, statik analiz yapılarak kullanılmıştır.

2.1.1 Chidamber ve Kemerer Metrikleri

CK metrikleri temelde CBO, DIT, NOC, WMC ve RFC metriklerinde oluşur bu metriklere geliştirilmiş başka metrikler de eklenerek metrikler genişletilmiştir.

Tablo 2.1 Toplanan metrikler

Metrik	Açıklama
CBO	Bir sınıfın diğer hangi sınıflara, veri türlerine veya bileşenlere bağlı olduğunu belirler. Java'nın kendi temel veri türleri veya sınıfları dışındaki diğer bağımlılıkları da sayar. CBO değerinin yüksek olması nesnelerin diğer nesnelere bağlılığının fazla olduğu dolayısıyla daha karmaşık ve hatalara müsait bir tasarıma işaret eder.
CBO Değiştirilmiş Veya CBO*	Ölçümler bir sınıftan bağımlılığı, hem türün başkalarına yaptığı referansları hem de diğer türlerden aldığı referansları değerlendirir. CBO'nun özelleştirilmiş bir türüdür ve benzer bir pozitif ilişkiye işaret eder.
FAN-IN	Bir sınıfa veya modüle gelen giriş bağımlılıklarının sayısını temsil eder. Girdi bağlantılarının sayısını ifade eder, yüksek olması hata sayısını pozitif yönde etkileyebilir.

Tablo 2.1 Toplanan metrikler (devamı)

FAN-OUT	Bir sınıfın veya modülün sahip olduğu çıkış bağımlılıklarının sayısını temsil eder. Çıktı bağlantılarının sayısını ifade eder, yüksek olması hata sayısını pozitif yönde etkileyebilir.
WMC	Programda yer alan dallanma yapılarının karmaşıklığını ifade eder. Metot ağırlıklarını ölçen bu metrik değerinin yüksek olması daha karmaşık metotlara dolayısıyla hataya daha açık sınıfların oluşmasına işaret edebilir. $WMC = \sum_{i=1}^{n} Wi$ *(1.1)* n = Sınıf içindeki metotların sayısını ifade eder. Wi = Metodun ağırlığını ifade eder.
DIT	Proje içindeki doğrudan veya dolaylı olarak bağlı olduğu ana sınıfların sayısını temsil eder. Tüm sınıflar en az Java.lang.Object'ten oluşturulduğundan minimum değeri 1'dir. Yüksek DIT değeri karmaşık bir miras hiyerarşisine işaret edebilir bu nedenle hata sayısı ile pozitif bir ilişki beklenmektedir. $DIT = max_i\, Depth(C_i)$ *(1.1)* C_i = Miras aldığı sınıfları temsil eder. Depth(C_i) = Sınıfın soy ağacındaki derinliği ifade eder.
NOC	Belirli bir sınıfın kaç tane doğrudan alt sınıfa sahip olduğunu sayar. Çocuk sayısı ne kadar fazlaysa, yeniden kullanım o kadar fazla olur. Dolayısıyla kalıtım tam anlamıyla uygulanmış olur. Bir sınıfın çok sayıda alt sınıfa sahip olması, sınıftaki yöntemlerin daha fazla test gerektirebilecek potansiyel etkisini belirtir. Yüksek NOC değeri sınıfın alt sınıflar tarafından daha fazla oranda kullanıldığı dolayısıyla daha karmaşık bir mimariye sahip olabileceği tahmin edilmektedir.

Tablo 2.1 Toplanan metrikler (devamı)

NOSI	Yapılan statik çağrı sayısı.
RFC	Sınıf içerisindeki benzersiz metot çağrılarının sayısıdır. Bir sınıftan çağrılabilen yöntemlerin sayısı ne kadar fazlaysa, sınıfın karmaşıklığı o kadar artar. $$RFC = RS = \{M\} \cup_{all\ i} \{R_i\} \quad (1.1)$$ {M} = Sınıftaki tüm metotların kümesidir. $\{R_i\}$ = Metot i tarafından çağırılan metotların kümesidir.
LOC	Kod satır sayısı. Kod satır sayısını ifade eder yüksek ve düşük olması yazılım kalitesi hakkında direkt bir anlam ifade etmese de yüksek olması bakım maliyetlerini negatif yönde etkilediği için hata sayısı ile pozitif bir ilişkiye sahip olabilir.
LCOM - LCOM*	0 ile 1 arasında bir değer verir, bu da sınıf uyumunun olmadığı anlamına gelir. Bir sınıftaki yöntemler veya özellikler güçlü bir şekilde ilişkili değilse veya birbirine bağlı değilse, o sınıf içindeki uyum düşük olacaktır. Yüksek LCOM değeri metotların birlikte çalışma eksikliğine ve ayrışmış bir tasarıma işaret eder. Hata sayısı ile negatif bir orantıya sahip olabileceği düşünülmektedir. C_1 = n tane metot içeren bir sınıfı ifade eder. (M_1, M_2 … M_n) $\{I_J\}$ = M_i metodunda kullanılan nesneleri ifade eder. ($\{I_1\}$, $\{I_2\}$ … $\{I_n\}$) $$P = \{(I_i, I_J) \mid I_i \cap I_J = \emptyset\} \quad (1.1)$$ $$Q = \{(I_i, I_J) \mid I_i \cap I_J \neq \emptyset\} \quad (1.2)$$ $$LCOM = |P| - \lceil Q \rceil,\ Eğer\ |P| > \lceil Q \rceil \quad (1.3)$$ $$= 0\ , Eğer\ |P| \leq \lceil Q \rceil$$

Tablo 2.1 Toplanan metrikler (devamı)

TCC	0 ile 1 arasındadır. Eğer yöntemler veya çağrı ağaçları aynı sınıf değişkenine erişiyorsa, TCC bir sınıfın uyumunu aralarındaki doğrudan bağlantılar yoluyla ölçer. Sınıf içerisindeki metotların ilişki metriği olarak nitelendirilebilen bu metriğin yüksek olması tutarlı, düzgün bir tasarıma işaret edebilir dolayısıyla hata sayısı ile negatif yönde bir ilişki beklenebilir.
LCC	TCC'ye benzer ancak görünür sınıflar arasındaki bağlantıları yalnızca doğrudan değil dolaylı olarak da dikkate alarak ölçüm yapar. Yüksek LCC, dolaylı ve bağımsız sınıf bağlantılarının arttığını gösterir ve bu durum genellikle yazılımın bakımını ve anlaşılabilirliğini zorlaştırır. Hata sayısı ile pozitif yönde bir ilişkiye işaret edebilir.

2.1.2 Hata Takip Sisteminden Elde Edilen Metrikler

Uzun vadeli projelerden toplanan hata kayıtları, iyileştirmeler ve yeni özellikler metriklere dönüştürüldü. Bu metriklerin netliği, detaylandırılması ve doğru etiketlenmesi, yazılım geliştiricilerin daha fazla katkı sağlamasını sağlar. Bu metrikleri elde etmek için Jira, Bugzilla ve sürüm notları kullanılmıştır.

Tablo 2.2 Hata takip sisteminden elde edilen metrikler

Metrik	Açıklama
Etkilenen Sürüm	Hatanın ilk kez ortaya çıktığı sürüm etkilenen sürüm olarak etiketlenir. Hata birden fazla sürümü etkilemiş olabilir, ancak bu hatanın ilk kez ortaya çıktığı sürüm dikkate alınır.
Düzeltilen Sürüm	Uygulama, yeni özellik veya düzeltilen hata sürümü düzeltilmiş sürüm olarak etiketlenir.
Seviye	Hatanın önemlilik düzeyi.

Yeni özellik ve iyileştirmeler için sadece etkilenen sürüm de denebilecek tek bir sürüm bilgisi tutulmaktadır.

2.1.3 Türetilen Metrikler

Son kategori, yukarıda belirtilen metriklerin belirli kombinasyonlarını içerir. Toplanan metriklerden türetilen metrikleri ifade eder.

Tablo 2.3 Türetilen metrikler

Metrik	Açıklama
Hata Yoğunluğu (DD)	Hata sayısı / KLOC oranını temsil eder.
Ağırlıklı Metotlar Sınıfı Başına Hata	Hata sayısı / WMC oranını temsil eder.

2.2 Verilerin Elde Edilmesi

Hadoop için 2.3.0 ile 3.0.3 arasındaki tüm sürümler hesaplamaya dahil edildi. Bu sürümler, CK metrik aracı kullanılarak kaynak kodlarıyla birlikte çıkartıldı ve tüm metot ve sınıf bazlı ölçümler toplandı. Eclipse SWT için 2.1-4.9 ve Kafka için 0.8.0 beta sürümünden 3.6.0 sürümüne kadar ölçüm sonuçları toplandı. Bu verileri elde etmek için Bugzilla ve Jira platformları kullanıldı. Toplama yöntemleri değişmiş olsa da Hadoop ve Kafka için platformdaki sürüm bilgileri tüm yeni özellikler, iyileştirmeler ve hata düzeltmeleriyle doğru bir şekilde platformda etiketlendiği varsayıldı. Eclipse SWT için sadece hata numaraları elde edilebildi. x.y.z olarak toplanan sürüm verileri, takip etmek için sayısal bir yaklaşıma dönüştürüldü; bu şekilde, sayısal değerler değişmemiştir. x.y.z.t formatı x.yzt formatına dönüştürülmüştür.

2.2.1 Verilerin Temizlenmesi

Toplanan tüm veriden sürüm bilgisi etiketleri olmayan veriler ve test dosyaları çıkarılmıştır. Elde edilen veri seti ve boyutları Tablo 2.4 'de sunulmuştur.

Tablo 2.4 Toplam veri seti

Metrik	Apache Hadoop	Eclipse SWT	Kafka (Java)
Toplam Sınıf Sayısı	445474	16266	142536
Toplam Metot Sayısı	1017284	179559	859470
Hata Sayısı	21640	8444	4162
İyileştirme Sayısı	258	-	1473
Yeni Geliştirme Sayısı	3139	-	143

2.3 Analiz Yöntemlerinin Belirlenmesi

Veri seti oluşturulduktan sonra veri seti üzerinde yapılacak analizlerde hangi yöntemlerin kullanılması gerektiği belirlenmiştir. Referans olarak önceki çalışmalarda kullanılan yöntemler dikkate alınmıştır. Veri setinin yapısı gereği sınıf veya metotların hata sayıları ile birebir eşlenmiş hali değil, versiyon başına kümülatif hata sayısı bilgisi vardır. Bu sebeple sınıf veya metot düzeyinde yapılan incelemeler ortalama, standart sapma ve maksimum değerler alınarak tekrar edilmiştir. Gruplamadan yapılan analizler için de hata sayısı her bir grup için çoğaltılarak ele alınmıştır.

İlk olarak korelasyon analizi ile metriklerin hata sayısı ile olan ilişki katsayıları incelenmiştir. Daha sonrasında çoklu lineer regresyon, OLS tabloları kullanılarak analiz edilmiştir. Son olarak Apache Hadoop ve Eclipse SWT veri seti birleştirilerek yapay sinir ağı modeli üzerinde doğruluk çalışması yapılmıştır.

2.3.1 Korelasyon Analizi

CK metriklerini bağımlı değişkenler olarak değerlendirebiliriz; bunun yanı sıra hata sayısı ve yazılım geliştirmelerini ise bağımsız değişkenler olarak ele alabiliriz. Özellikle odaklandığımız bağımlı değişkenlerin bağımsız değişkenleri ne ölçüde etkilediği ve ileriye dönük tahminlerimizde hangi parametrelerden ne ölçüde destek alabileceğimiz önemli hale gelir. İlk aşamada bu parametreler arasındaki doğrusal

ilişkiyi anlamak önemlidir. Örneğin, bu üç proje için hangi metrikler hata sayısını en çok oranda etkilemiştir? Hangi metrikler hata sayısı ile genellikle doğrusal veya doğrusal olmayan bir ilişki içindedir? Pearson ve Spearman korelasyon katsayıları, bu ilişkinin gücü ve yönü hakkında bize önemli bilgiler verir. Projeler içindeki dağılıma bağlı olarak, Pearson korelasyon katsayısı normal dağılımlı veri setlerinde incelenirken, Spearman korelasyon katsayısı normal dağılıma sahip olmayan veri setlerinde tercih edilir. İlk aşama bu iki dağılımdan hangisinin tercih edileceğini belirlemek olacaktır. Verilerin normal dağılıma sahip olup olmadığını belirlemek için en çok kullanılan testlerden bir tanesi Shapiro-Wilk testidir. H_0 hipotezi veri setinin dağılımını normal dağılıma sahip olmadığını, H_1 hipotezi iste normal dağılıma sahip olduğunu belirtir.

2.3.1.1 Normallik Testi

Normallik testi python SciPy kütüphanesi ile gerçekleştirilmiş olup Hadoop, Eclipse ve Kafka'nın sınıf ve metot düzeyindeki p değerleri ve 0.05 anlamlılık düzeyine göre normallik sonuçları aşağıdaki tablodaki gibidir.

Tablo 2.5 Normallik testi sonuçları

	İstatistik Değeri	**p değeri**	**Normal Dağılıma Sahip Mi? (p>0.05)**
Apache Hadoop	0.6031	0.0000	Hayır
Eclipse SWT	0.8715	0.0188	Hayır
Kafka (Java)	0.8693	0.0000	Hayır

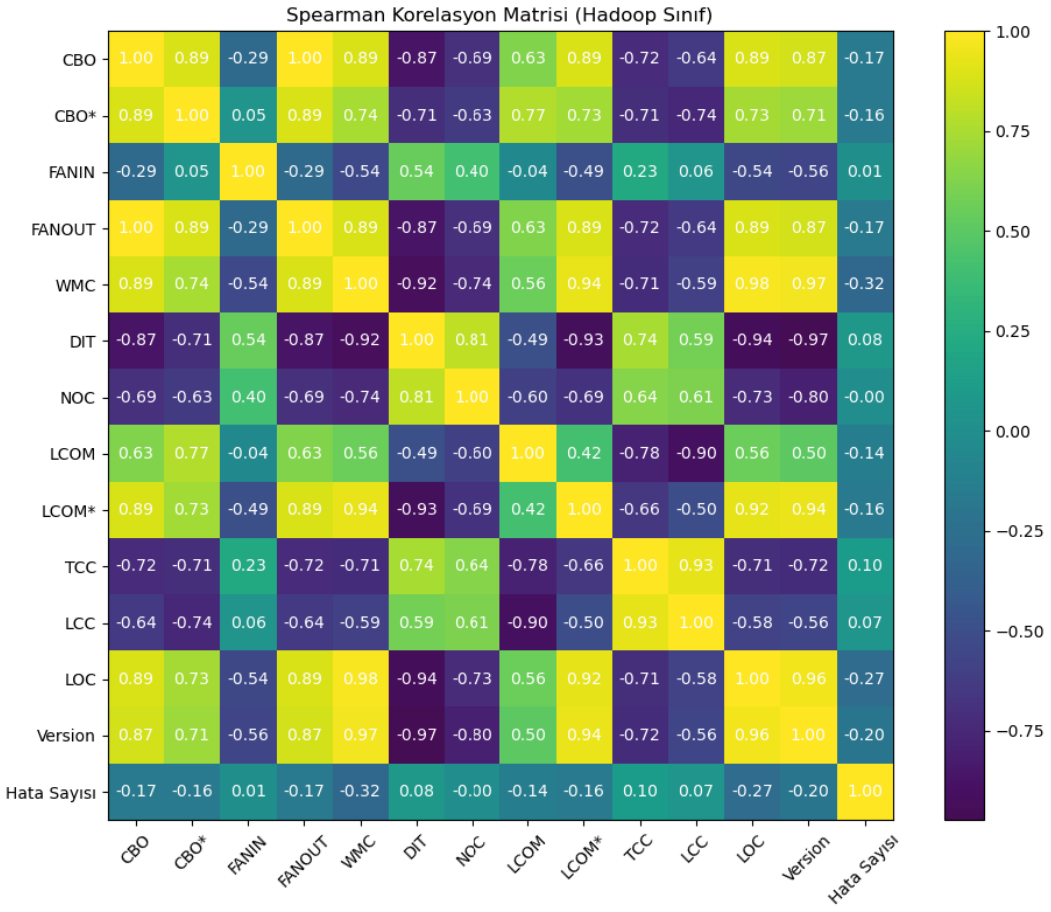
Normallik testi sonucuna göre 3 projedeki hata sayıları normal dağılım göstermemektedir. H_0 hipotezi kabul edilmiştir. Çoğunlukla, korelasyon terimi, iki sürekli değişken arasındaki lineer ilişki bağlamında kullanılır. Pearson korelasyon katsayısı genellikle birlikte normal olarak dağılmış veriler için kullanılır. Normal dağılıma sahip olmayan sürekli veriler, ordinal veriler veya ilgili aykırı değerlere sahip veriler için, bir düzenli ilişkinin ölçüsü olarak Spearman korelasyonu kullanılabilir (Schober vd., 2018b). Pearson korelasyonu iki parametre arasındaki

lineer ilişkiyi ölçerken Spearman korelasyonu iki parametre arasındaki sıralı ilişkiyi ölçer. Bu çalışmada Spearman korelasyon ölçüm sonuçlarına da yer verilecektir.

2.3.1.2 Apache Hadoop Sınıf Metrikleri ve Hata Sayılarının Spearman Korelasyon Değerleri

Aşağıdaki Şekil 2.1'de Hadoop'un sınıf metriklerinin diğer metrikler ile olan Spearman korelasyon değerlerine yer verilmiştir. Hata sayısı ve versiyon bu çalışma kapsamında ilgilenilen bağımsız değişkenler olduğu için diğer parametreler ile olan ilişkilerinin tespit edilmesi amaçlanmıştır. Versiyon değeri için projenin sürümler boyunca geçirdiği değişim olarak soyut bir nitelendirme yapabiliriz.

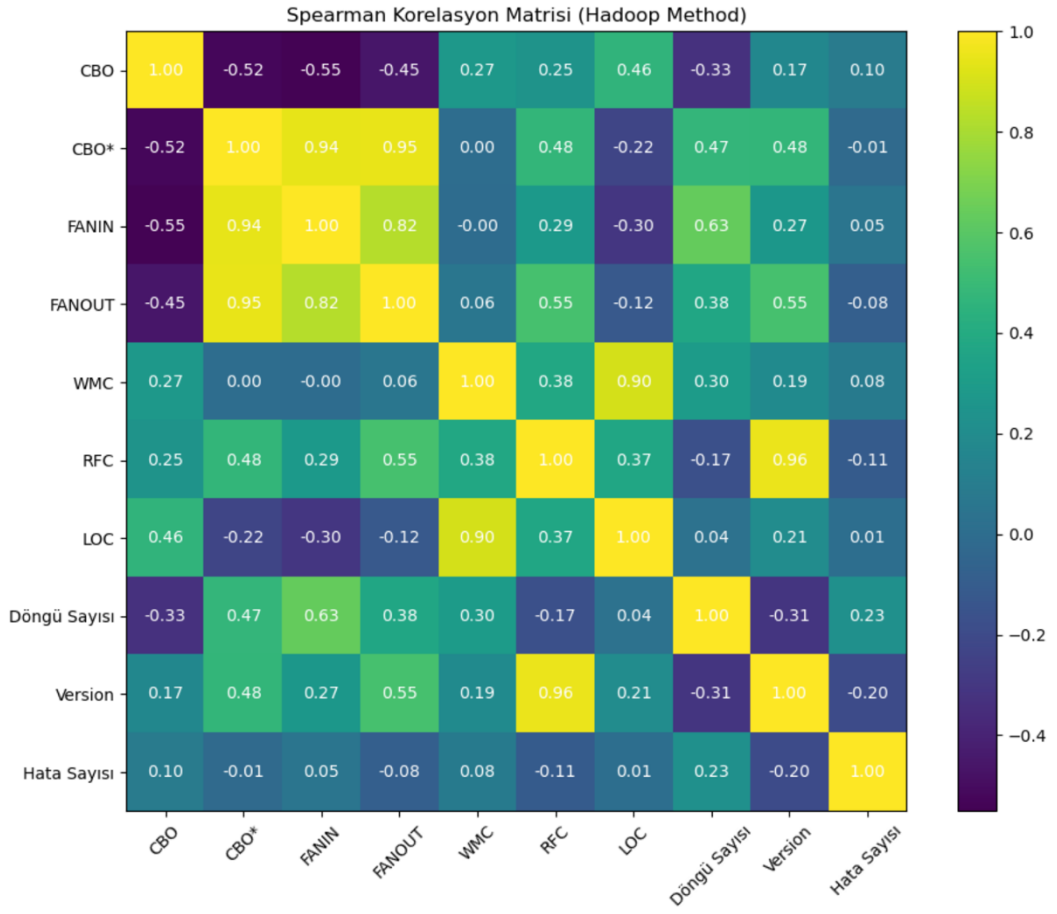
Hata sayısı parametresi için en yüksek değerler negatif ilişki için WMC, LOC ve versiyon gelmektedir. Pozitif ilişki için ise TCC, DIT ve LCC dikkat çekmektedir. Versiyon parametresi için ise pozitif sıralamada WMC, LOC, LCOM*. Negatif olarak DIT, NOC ve TCC'dir.



Şekil 2.1 Spearman korelasyon matrisi (Hadoop Sınıf)

2.3.1.3 Apache Hadoop Metot Metrikleri ve Hata Sayılarının Spearman Korelasyon Değerleri

Aynı analizi metot düzeyinde gerçekleştirdiğimizde hata sayısı için pozitif çerçevede döngü sayısı, CBO ve WMC. Negatif çerçevede versiyon, RFC ve FANOUT dikkat çekmiştir ancak çok yüksek oranlar gözlemlenmemiştir. Versiyon için ise yüksek oranda RFC, FANOUT ve CBO* dikkat çekmektedir.



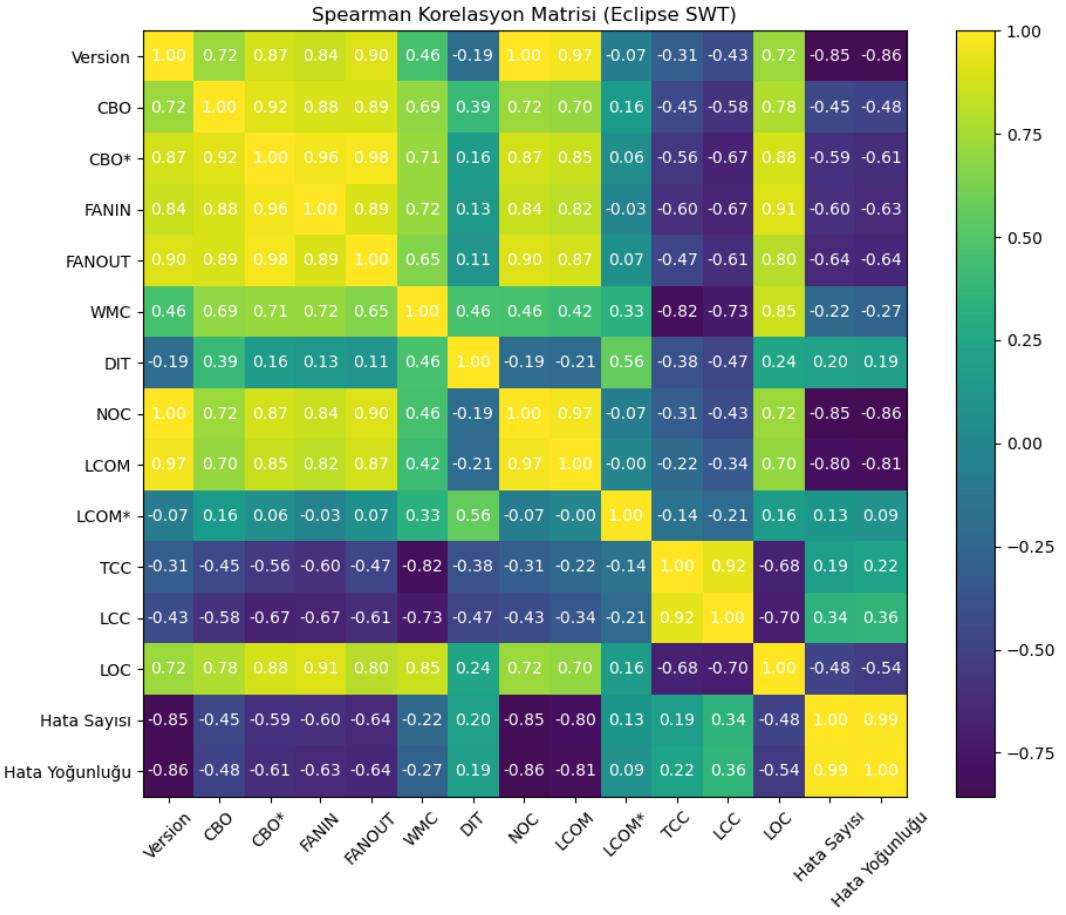
Şekil 2.2 Spearman korelasyon matrisi (Hadoop Metot)

2.3.1.4 Eclipse SWT Sınıf Metrikleri ve Hata Sayılarının Spearman Korelasyon Değerleri

Eclipse SWT projesi için de sınıf metriklerinin diğer metrikler ile olan Spearman korelasyon değerleri incelenmiştir. Hata sayısı için en yüksek oranlar negatif sonuçlar için versiyon, NOC ve LCOM değerlerinde pozitif olarak da LCC değerinde gözlemlenmiştir. Versiyon için en iyi pozitif oranlar LCOM, FANOUT ve CBO'da negatif oranlar ise hata sayısı haricinde LCC metriğinde gözlemlenmiştir. Apache Hadoop projesine kıyasla hata sayısı ile çok güçlü ilişkiler içeren metrikler oldukça fazladır. Daha güçlü ilişki içeren metrikler olsa da WMC,

LOC ve versiyon geçişi Apache Hadoop sınıf korelasyon çıktısına benzer şekilde hata sayısı ile negatif ilişki içerisindedir. TCC, DIT ve LCC ile de aynı Hadoop projesinde olduğu gibi pozitif yönlü bir ilişkiye işaret etmektedir.

Dikkat çeken bir nokta NOC ve LCOM değerlerinin sürüm ile güçlü pozitif bir ilişki göstermesidir. Benzer bir ilişki Apache Hadoop'da gözlemlenmemiştir. DIT ve TCC değerleri Hadoop'a benzer şekilde versiyon ile negatif bir ilişki, WMC ve LOC metriği ile de pozitif bir ilişki göstermiştir.

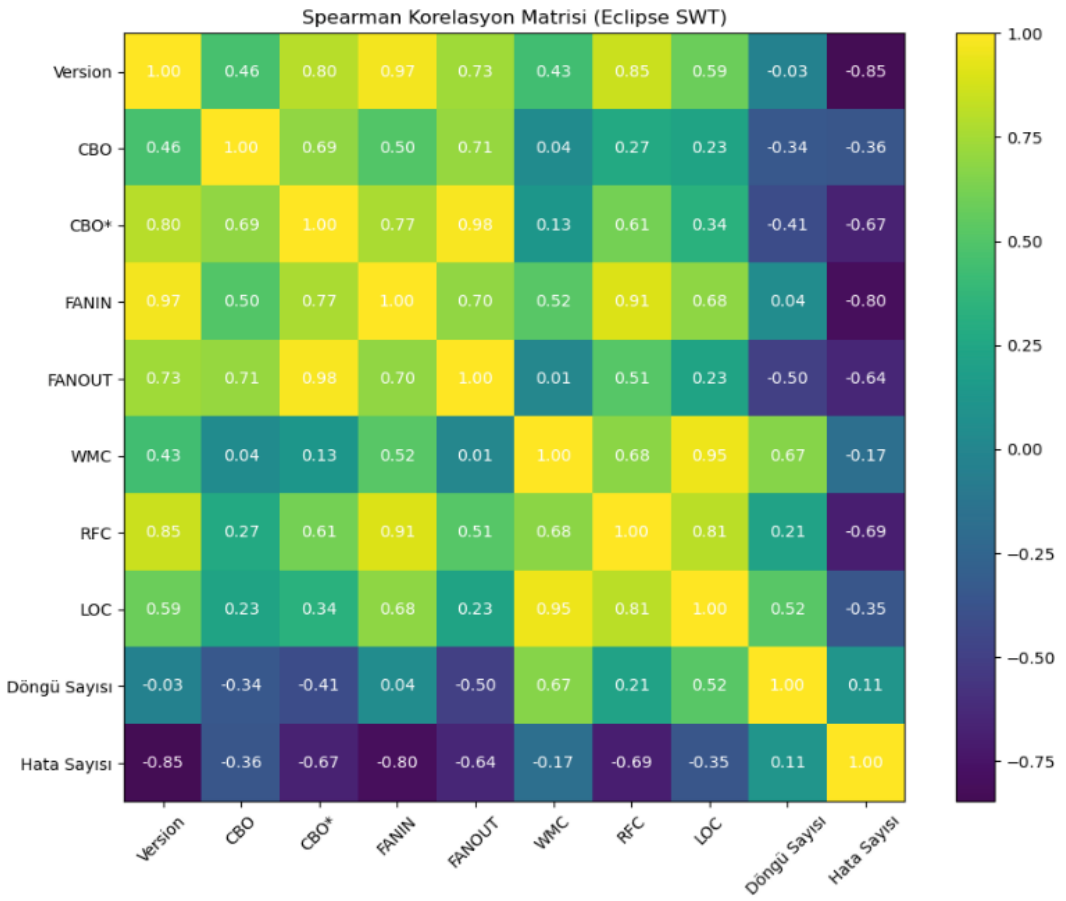


Şekil 2.3 Spearman korelasyon matrisi (Eclipse SWT Sınıf)

2.3.1.5 Eclipse SWT Metot Metrikleri ve Hata Sayılarının Spearman Korelasyon Değerleri

Son olarak Eclipse SWT projesi için de metot metriklerinin diğer metrikler ile olan Spearman korelasyon değerleri incelenmiştir. Dikkat çeken sonuçlar hata sayısı için FANIN, RFC ve versiyon ile negatif ilişki içerisinde olması versiyon metriği ile de FANIN, RFC, CBO* ile pozitif ilişki oranları göstermesidir. Hadoop projesinin

metot çıktılarına benzer şekilde RFC metriği iki projede de hata sayısı ile negatif ilişki eğilimi göstermiştir.



Şekil 2.4 Spearman korelasyon matrisi (Eclipse SWT Metot)

Tüm korelasyon oranları gruplanarak incelendiğinde iki projede de ortak olan bazı metrikler gözlemlenmiştir. Bulgular:

1. Sınıf düzeyinde hata sayısı ile LCC arasında pozitif yönlü bir ilişki mevcuttur. Sınıf düzeyinde hata sayısı ile WMC ile LOC arasında negatif yönlü bir ilişki mevcuttur.
2. Sınıf düzeyinde versiyon geçişi ile WMC, LOC ve LCOM arasında pozitif yönlü bir ilişki mevcuttur. Sınıf düzeyinde versiyon geçişi ile DIT ve TCC arasında negatif yönlü bir ilişki mevcuttur.
3. Metot düzeyinde hata sayısı ile RFC arasında negatif yönlü bir ilişki mevcuttur.
4. Metot düzeyinde versiyon geçişi ile RFC ve CBO* arasında negatif yönlü bir ilişki mevcuttur.

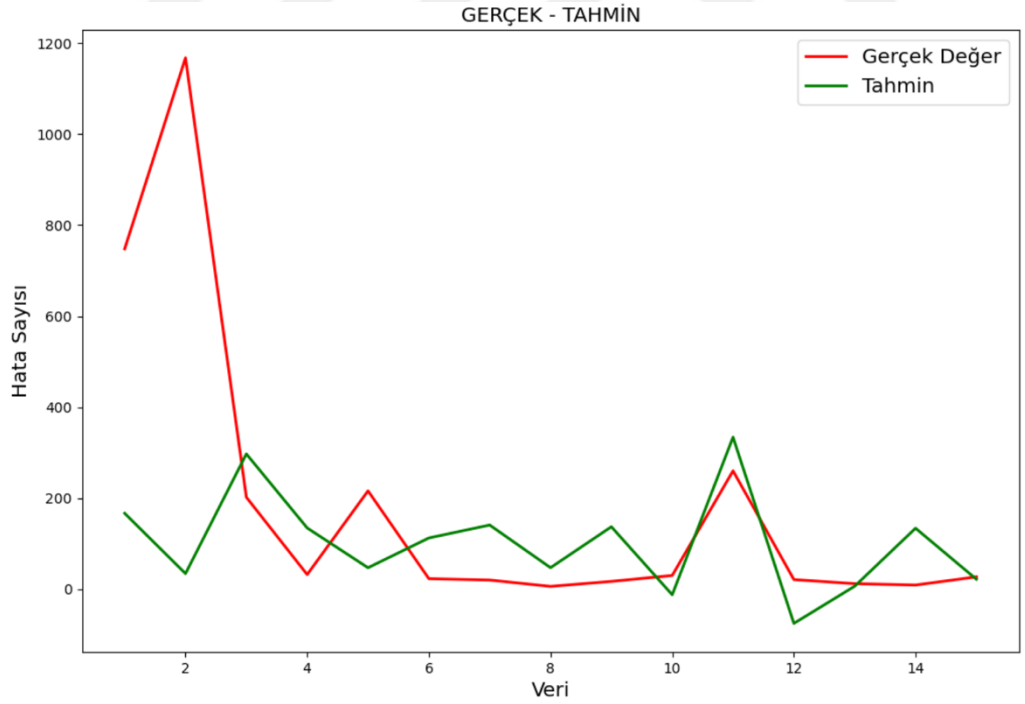
2.3.2 Veriler Üzerinde Çoklu Lineer Regresyon Analizi

Elde edilen veri seti maksimum, ortalama ve minimum değerlerine göre versiyon bilgisi üzerinden gruplanmıştır. Versiyon bilgisi değerlendirmeye alınabilmesi için rakamsal hale dönüştürülmüştür.

2.3.2.1 Apache Hadoop Sınıf Düzeyinde Çoklu Lineer Regresyon Analizi

Sınıf düzeyinde Apache Hadoop veri seti 445474 sınıf sayısı, 21640 hata sayısı, 3139 yeni özellik sayısı ve 258 tane geliştirme içermektedir.

Veri setinin 48 tane Hadoop versiyonuna göre aşağıdaki 14 sınıf metriği maksimum, ortalama ve standart sapma değerlerine göre analiz edilmiş ve üç model oluşturulmuştur. CBO, CBOMODIFIED, FANIN, FANOUT, WMC, DIT, NOC, LCOM, LCOM*, TCC, LCC, LOC, versiyon ve hata sayısı metriklerinden oluşmaktadır. COUNT hata sayısını ifade etmektedir. Veri setinin %80'i eğitim için ayrıldı ve 33 eğitim verisi, 15 test verisi olacak şekilde ayrıldı. 15 test verisinin gerçek ve tahmin değerleri aşağıdaki tabloda çizilmiştir. y_test ve y_pred arasındaki hata oranı aşağıdaki şekilde elde edilmiştir.



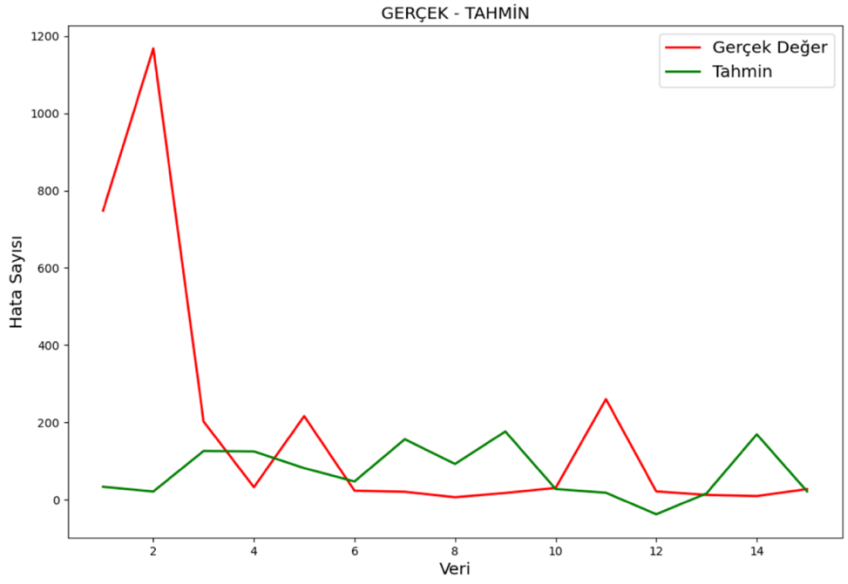
Şekil 2.5 Hadoop sınıf ortalama değerlerinin gerçek ve test çıktıları

Gruplanmış veriler üzerinde OLS tablosuna bakarak katsayılar ve varyans değerleri incelenmiştir. Sonuçlara göre Hadoop projesi sınıf ortalaması ile hata sayısı tespiti

%57.3 uygunluk oranı ile başarılı sayılabilecek bir oranı elde edilememiştir. Aynı tahminleri verilerin ortalaması yerine standart sapma için gerçekleştirelim.

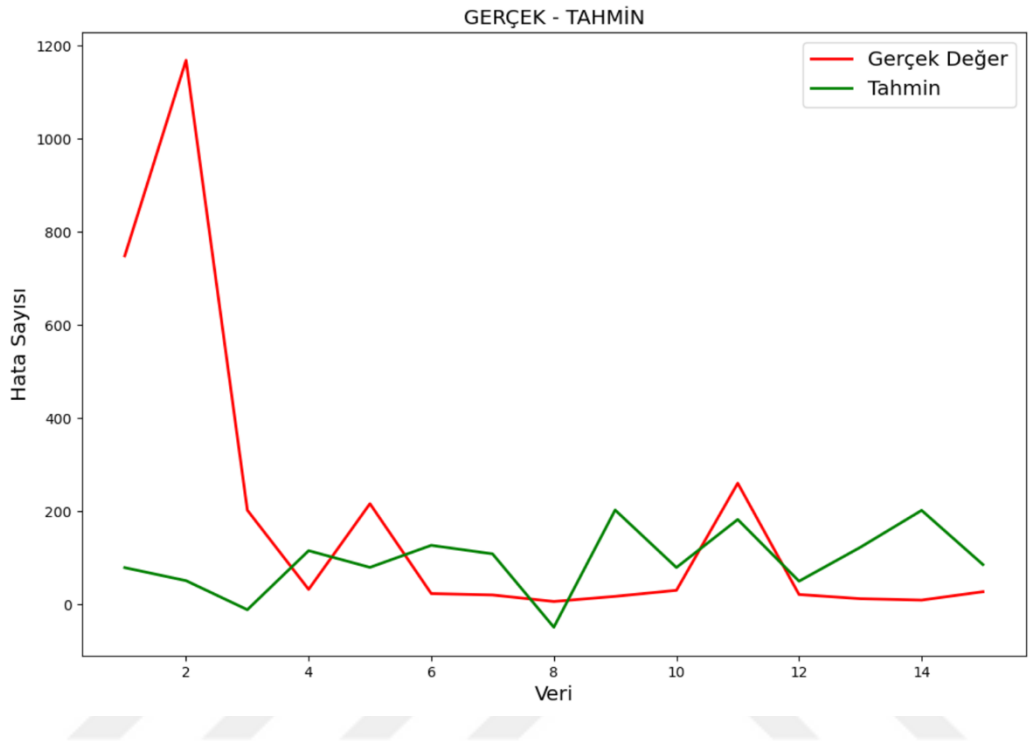
Tablo 2.6 Hadoop sınıf gruplanmış (ortalama) OLS regresyon tablosu

R^2	0.573	**Omnibus**	27.091			
Adj. R^2	0.349	**Prob (Omnibus)**	0.000			
F-statistic	2.557	**Skew**	1.784			
Prob (F-statistic)	0.0309	**Kurtosis**	8.801			
Log-Likelihood	-189.81	**Durbin-Watson**	2.287			
AIC	403.6	**Jarque-Bera (JB)**	63.776			
BIC	421.6	**Prob(JB)**	1.42×10^{-14}			
Df Residuals	21	**Cond. No.**	3.68×10^{18}			
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.402×10^4	1.37×10^4	1.025	0.317	-1.44×10^4	4.25×10^4
CBO	922.6901	715.164	1.290	0.211	-564.576	2409.956
CBO*	-571.9460	337.696	-1.694	0.105	-1274.223	130.331
FANIN	-1494.6361	768.459	-1.945	0.065	-3092.735	103.463
FANOUT	922.6901	715.164	1.290	0.211	-564.576	2409.956
WMC	-2497.0915	1829.541	-1.365	0.187	-6301.831	1307.648
DIT	-5418.3476	5798.839	-0.934	0.361	-1.75×10^4	6640.998
NOC	1.489×10^4	2.7×10^4	0.552	0.587	-4.12×10^4	7.1×10^4
LCOM	-10.6489	23.468	-0.454	0.655	-59.452	38.155
LCOM*	4.401×10^4	3.19×10^4	1.379	0.182	-2.23×10^4	11×10^4
TCC	12.85×10^4	11.9×10^4	1.082	0.291	-11.8×10^4	37.5×10^4
LCC	-13.16×10^4	8.98×10^4	-1.465	0.158	-31.8×10^4	5.52×10^4
LOC	311.1722	341.368	0.912	0.372	-398.742	1021.087
Versiyon	5.3361	14.088	0.379	0.709	-23.962	34.634



Şekil 2.6 Hadoop sınıf standart sapma değerlerinin gerçek ve test çıktıları

Aynı veri setinin versiyonlara göre gruplanmış değerlerinin standart sapması üzerinden yapılan hata tahmini bize **%48.2** uygunluk oranı sunmuştur. Son olarak maksimum değerleri üzerinden uygunluk tahmini çalışması yapılmıştır ve **%36.8** ile başarılı bir uygunluk sunmamıştır.



Şekil 2.7 Hadoop sınıf maksimum metrik değerlerinin gerçek ve test çıktıları

Başarısız tahminlerin en büyük nedenlerinden birinin versiyon gruplamasının parametreler üzerinde etki kaybına neden olabileceği tahmin edilmektedir. Örneğin düşük bir CBO değerinin yüksek bir CBO değerini nötralize ederek önemli verilerin kaybolmasına sebep olabileceği tahmin edilmektedir. Bu durumun analizini yapabilmek için gruplama işlemi aradan çıkartılarak bir model oluşturulmuştur. Bu model için ilgili versiyon değeri ile eşleşen hata sayısı işaretlenmiştir. 400926 eğitim verisi ve 44548 test verisi ile oluşturulan model **%3.23** ortalama kare hatası ile çok başarılı sonuçlar vermemiştir. Aşağıda OLS tablosu gözlemlenmektedir. P değeri 0.05'den küçük olan bir parametre gözlenememiştir.

Denemeler sonucunda hata tahmininde Apache Hadoop projesi için en başarılı sonuçları sınıf bazında ortalamaya göre yaptığımız gruplama ile elde ettik. Gruplama yapmadan elde ettiğimiz regresyon tablosu Tablo 2.7 de verilmiştir.

Tablo 2.7 Hadoop gruplanmamış sınıf OLS regresyon tablosu

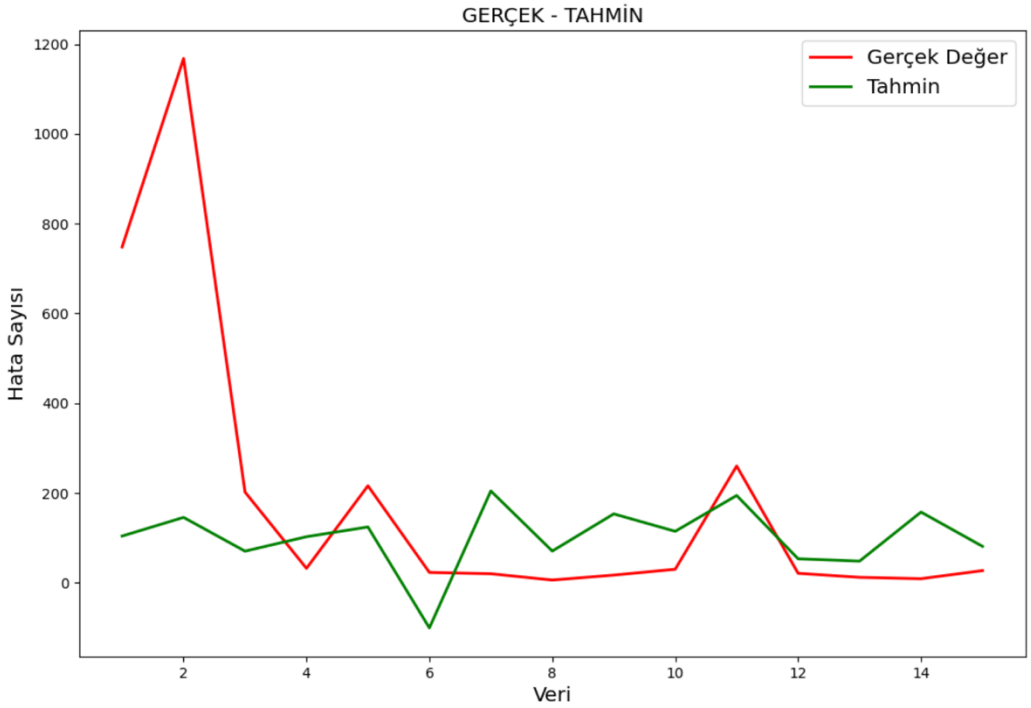
R^2		0.032	**Omnibus**		173905.009	
Adj. R^2		0.032	**Prob(Omnibus)**		0.000	
F-statistic		719.4	**Skew**		3.515	
Prob (F-statistic)		0.00	**Kurtosis**		17.613	
Log-Likelihood		-1.59×10^6	**Durbin-Watson**		1.995	
AIC		3.191×10^6	**Jarque-Bera (JB)**		2601153.128	
BIC		3.191×10^6	**Prob(JB)**		0.00	
Df Residuals		237390	**Cond. No.**		6.29×10^{14}	
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	506.8944	4.346	116.647	0.000	498.377	515.41
CBO	11.95×10^{10}	48.3×10^{10}	0.247	0.805	-82.8×10^{10}	107×10^{1}
CBO*	4.99×10^{10}	9.94×10^{10}	0.502	0.615	-14.5×10^{10}	24.5×10^{10}
FANIN	-4.99×10^{10}	9.94×10^{10}	-0.502	0.615	-24.5×10^{10}	14.5×10^{10}
FANOUT	-16.95×10^{10}	49.1×10^{10}	-0.345	0.730	-113×10^{10}	79.3×10^{10}
WMC	0.0039	0.020	0.199	0.842	-0.035	0.042
DIT	-0.3984	0.293	-1.361	0.173	-0.972	0.175
NOC	0.2988	0.226	1.325	0.185	-0.143	0.741
LCOM*	-1.0331	1.404	-0.736	0.462	-3.786	1.720
TCC	-2.2157	3.162	-0.701	0.483	-8.413	3.981
LCC	0.7995	2.930	0.273	0.785	-4.943	6.542
LOC	0.0030	0.002	1.614	0.107	-0.001	0.007
Versiyon	-1.2770	0.014	-88.903	0.000	-1.305	-1.249

2.3.2.2 Apache Hadoop Metot Düzeyinde Çoklu Lineer Regresyon Analizi

Sınıf verileri ile elde edilen CK metrikleri sınıf üzerinde en yüksek %57.3'e yakın bir uygunluk sunmuştur. Aynı çalışmayı metot verileri üzerinde gözlemlemek bize metot düzeyinde verilerin başarılı bir çoklu regresyon modeli sağlayıp sağlayamayacağını gösterecektir. Benzer şekilde önce ortalama, standart sapma ve maksimum değerlerine göre gruplanmış veriler üzerinden ölçüm yapılmıştır.

Veri setinin 48 tane Hadoop versiyonuna göre aşağıdaki 14 sınıf metriği maksimum, ortalama ve standart sapma değerlerine göre analiz edilmiş ve üç model oluşturulmuştur. Filtrelenmiş metot sayısı 1017284, hata sayısı 21640, yeni özellik sayısı 3139 ve geliştirme sayısı 258'dir. Versiyon sayısı 48 üzerinden gruplama gerçekleştirilmiştir. CBO, CBOMODIFIED, FANIN, FANOUT, WMC, RFC, LOC, versiyon ve hata sayısı metrikleri kullanılmıştır. Sınıf metriklerine benzer şekilde bazı aralıklarda benzer eğilimlere sahip olduğu ve bazı aralıklarda da büyük

uyuşmazlıklara sahip olduğu gözlenmiştir. y_test ve y_pred arasındaki hata oranı aşağıdaki şekilde elde edilmiştir.



Şekil 2.8 Hadoop metot metrik değerlerinin gerçek ve test çıktıları

Sınıf tahminlerinde incelendiği gibi metot tahminlerinde de gruplama işlemi kullanılarak bir analiz gerçekleştirilmiştir. Analiz sonucunda elde edilen OLS tablosu maksimum metrik değerleri için Tablo 2.8'deki gibidir.

Tablo 2.8 Hadoop gruplanmış(max) metot ortalaması OLS regresyon tablosu

R^2		0.489	**Omnibus**		11.679	
Adj. R^2		0.346	**Prob(Omnibus)**		0.003	
F-statistic		3.421	**Skew**		1.146	
Prob (F-statistic)		0.0105	**Kurtosis**		4.705	
Log-Likelihood		-192.75	**Durbin-Watson**		2.577	
AIC		401.5	**Jarque-Bera (JB)**		11.216	
BIC		413.5	**Prob(JB)**		0.00367	
Df Residuals		25	**Cond. No.**		6.60×10^{16}	
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	3653.5824	1641.046	2.226	0.035	273.785	7033.380
CBO	-17.1289	6.733	-2.544	**0.018**	-30.996	-3.262
CBO*	5.4620	2.306	2.369	**0.026**	0.714	10.210
FANIN	-6.6817	3.169	-2.108	**0.045**	-13.209	-0.154
FANOUT	5.4620	2.306	2.369	**0.026**	0.714	10.210
WMC	-2799.8724	788.676	-3.550	**0.002**	-4424.182	-1175.56
RFC	-6.3872	4.331	-1.475	0.153	-15.308	2.534
LOC	584.3222	162.760	3.590	**0.001**	249.112	919.533
Versiyon	-10.8962	3.900	-2.794	**0.010**	-18.929	-2.868

Sonuçlar gözlemlendiğinde versiyon ortalamaları dikkate alındığında **%22.6**, standart sapma dikkate alındığında **%21.9** ve maksimum değerler dikkate alındığında **%48.9** bir uygunluk oranı sunmaktadır. Elde edilen sonuçlara göre metot düzeyinde yapılan hata tahminlerinden en uygun sonuçlar maksimum değerleri ile elde edildi.

2.3.2.3 Eclipse SWT Sınıf Düzeyinde Çoklu Lineer Regresyon Analizi

Bu aşamada lineer regresyon modelimizi Eclipse SWT sınıf metrikleri üzerinde gerçekleştirerek bu çalışmanın çıktılarını değerlendirdik. Toplamda 16266 sınıf ve 8444 hata sayısı incelendi. Lineer modelden oluşturulmuş OLS tablosu aşağıdaki gibidir.

Tablo 2.9 Eclipse SWT gruplanmamış sınıf OLS regresyon tablosu

R^2		0.607	**Omnibus**		246.897	
Adj. R^2		0.606	**Prob(Omnibus)**		0.000	
F-statistic		880.6	**Skew**		0.410	
Prob (F-statistic)		0.00	**Kurtosis**		3.635	
Log-Likelihood		-44180.	**Durbin-Watson**		1.996	
AIC		8.839 x10^4	**Jarque-Bera (JB)**		307.147	
BIC		8.818 x10^4	**Prob(JB)**		2.01 x10^{-67}	
Df Residuals		6843	**Cond. No.**		1.82 x10^{17}	
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1450.3155	12.175	119.122	0.000	1426.449	1474.182
CBO	-0.9621	0.946	-1.017	0.309	-2.816	0.892
CBO*	0.3837	0.254	1.513	0.130	-0.113	0.881
FANIN	-0.3386	0.284	-1.190	0.234	-0.896	0.219
FANOUT	0.7223	0.525	1.375	0.169	-0.307	1.752
WMC	0.01210	0.043	0.487	0.626	-0.063	0.105
DIT	-1.5100	2.089	-0.723	0.470	-5.604	2.584
NOC	-0.0435	0.067	-0.646	0.518	-0.175	0.088
LCOM*	-7.0604	5.813	-1.215	0.225	-18.456	4.335
TCC	-24.8682	23.468	-1.060	0.289	-70.874	21.137
LCC	42.6590	20.888	2.042	**0.041**	1.712	83.606
LOC	-0.0112	0.014	-0.825	0.409	-0.038	0.015
Versiyon	-290.8868	2.870	-101.339	0.000	-296.514	-285.260

Sonuçları değerlendirdiğimizde gruplamaksızın yapılan sonuçlarda **%60.7** uygunluk oranı sunmaktadır. LCC değeri 0.041 p-değeri ile sonuç üzerinde en etkili parametrelerden biri olarak dikkat çekmiştir.

2.3.2.2 Eclipse SWT Metot Düzeyinde Çoklu Lineer Regresyon Analizi

Eclipse SWT metot düzeyinde çoklu lineer regresyon analizi gerçekleştirildi. Bu amaç doğrultusunda 179559 metot metriği ve 8444 hata verisi kullanıldı. Analiz sonucunda elde edilen OLS çıktısı aşağıdaki gibidir.

Tablo 2.10 Eclipse SWT gruplanmamış metot OLS regresyon tablosu

R^2		0.616	**Omnibus**		4417.810	
Adj. R^2		0.616	**Prob(Omnibus)**		0.000	
F-statistic		2.515×10^4	**Skew**		0.403	
Prob (F-statistic)		0.00	**Kurtosis**		3.642	
Log-Likelihood		-0.810×10^6	**Durbin-Watson**		2.003	
AIC		1.620×10^6	**Jarque-Bera (JB)**		5554.260	
BIC		1.620×10^6	**Prob(JB)**		0.00	
Df Residuals		125682	**Cond. No.**		4.74×10^{13}	
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1457.8043	2.647	550.83	0.000	1452.617	1462.991
CBO	-2.6949	0.329	-8.184	0.000	-3.340	-2.049
CBO*	-12.61×10^{10}	32×10^{10}	-0.394	0.694	-75.4×10^{10}	50.2×10^{10}
FANIN	12.61×10^{10}	32×10^{10}	0.394	0.694	-50.2×10^{10}	75.4×10^{10}
FANOUT	12.61×10^{10}	32×10^{10}	0.394	0.694	-50.2×10^{10}	75.4×10^{10}
WMC	0.1255	0.109	1.147	0.251	-0.089	0.340
RFC	0.9785	0.191	5.114	**0.000**	0.604	1.353
LOC	0.0208	0.037	0.565	0.572	-0.051	0.093
Versiyon	-291.0788	0.652	-446.63	**0.000**	-292.356	-289.801

Elde edilen sonuçlara göre Eclipse SWT'nin metot düzeyinde yapılan hata tahmininde **%61.6** oranda uyumlu olduğu gözlemlenmiştir. Özellikle CBO ve RFC metriklerinin sonuç üzerinde büyük oranda etkili olduğu gözlemlenmiştir.

2.3.3 Yapay Sinir Ağı Modeli Kullanılarak Hata Sayısı Tahmini

Şimdiye kadar korelasyon analizi, çoklu lineer regresyon gibi farklı yaklaşımlar kullanarak metriklerin hata sayısı üzerine etkilerini analiz etmeye çalıştık. Şimdi ise Apache Hadoop, Eclipse SWT ve Kafka projeleri birleştirilerek elde edilen veri seti üzerinden bir yapay sinir ağı modeli oluşturulacaktır. Oluşturulan bu model üzerinde yapılan hata tahmininin başarı oranları incelenecek ve genel anlamda başarılı bir tahmin modeli oluşturulup oluşturulamayacağı tespit edilecektir.

Verilerin hazırlanması aşamasında 3 veri seti birleştirildikten sonra metrikler nitelik bazında normalize edilmiştir. Her bir sütun için o sütunun ortalamasını çıkarıp standart sapmasına bölme işlemi gerçekleştirilmiştir.

Gruplama verileri bize versiyon düzeyinde bir veri seti sunduğu için gün sonunda 3 projenin toplam versiyon sayısı kadar bir veri seti elimizde kalmaktadır. Veri seti az olduğu için K-fold çapraz doğrulama yaklaşımı kullanılmıştır. Bu yöntemde veri K parçaya ayrılır ve K tane aynı model oluşturulur ve her bir model K-1 parça veri ile eğitilir. Kalan veri de değerlendirmede kullanılır. Doğrulama değeri K tane doğrulama değerinin ortalaması alınarak elde edilir. Çalışma boyunca Keras kütüphanesi kullanılmıştır.

Giriş katmanı, gizli katman ve çıkış katmanından oluşan basit birçok katmanlı bir yapay sinir ağı oluşturuldu. Regresyon problemleri için tasarlandı, çıkış katmanında aktivasyon fonksiyonu kullanılmadı ve "mse" kayıp fonksiyonu kullanılmıştır. Modelin performansı "mae" (ortalama mutlak hata) metriği ile değerlendirilecektir. Bu model, giriş verisini alır, katmanlar arasında işlem yapar ve çıkış olarak bir sayı üretir.

1.) *layers.Dense(64, activation='relu', input_shape=(X_train.shape[1*],)):

İlk katman, 64 nörona sahip bir tam bağlı (dense) katmandır. "relu" aktivasyon fonksiyonu kullanılmıştır.

2.) *layers.Dense(64, activation='relu'):*

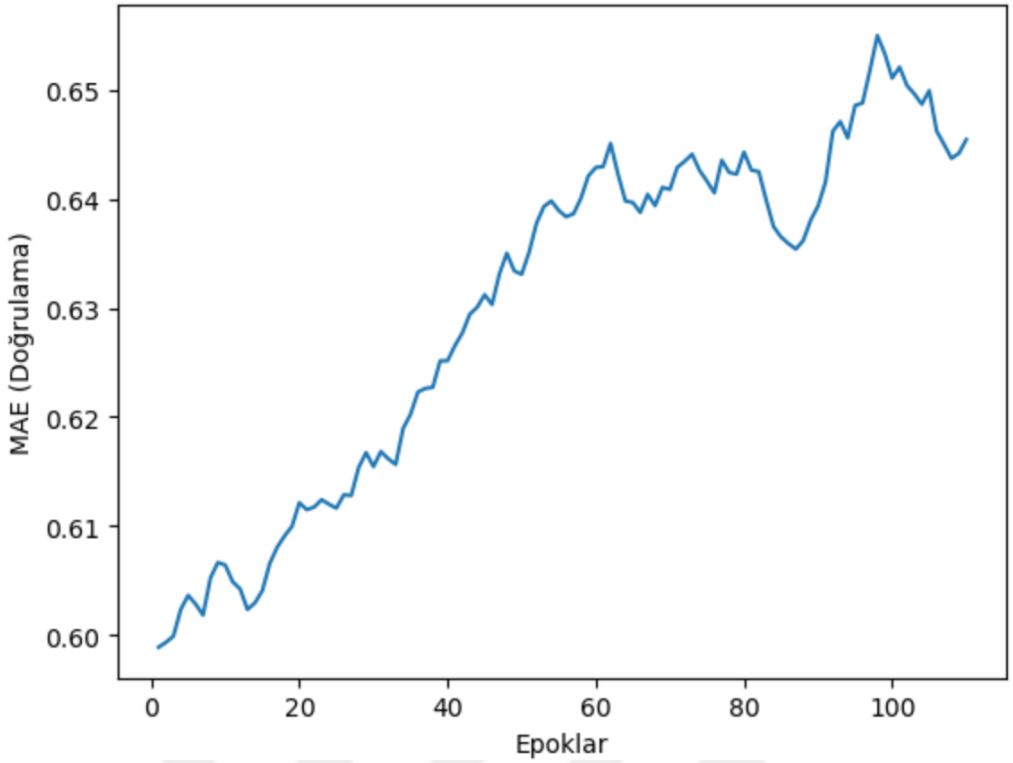
İlk katman, 64 nörona sahip bir tam bağlı (dense) katmandır. "relu" aktivasyon fonksiyonu kullanılmıştır. Bu katmanın giriş boyutunu önceki katmandan alınır.

3.) *layers.Dense(1):*

Son katman, tek bir nörona sahip bir çıkış katmanıdır. Bu tip bir çıkış, regresyon problemi olduğu için kullanılmıştır. Aktivasyon fonksiyonu belirtilmemiştir, çünkü çıkış değeri doğrudan modelin tahminidir.

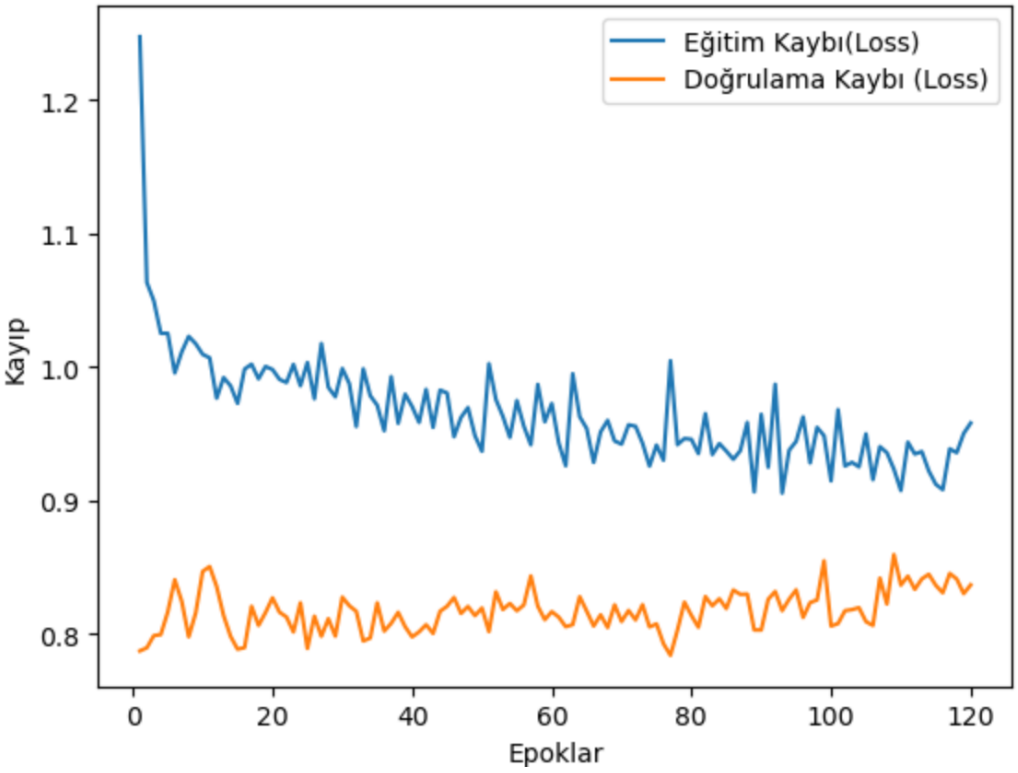
2.3.3.1 Sınıf Düzeyinde Sinir Ağı Modeli Hata Sayısı Tahmini

İlk aşamada 3 proje için ortalama gruplanmış veriler üzerinden bir model oluşturuldu. İlk aşamada 120 epok sayısı ile model eğitimi gerçekleştirildi. Şekil 2.9'dan anlaşılacağı üzere 80. epoktan sonra model aşırı uydurmaya başlıyor.



Şekil 2.9 Yapay sinir ağı modeli için epok sayısı ve doğrulama değerleri (Sınıf)

K-fold döngüsü sonucunda elde edilen doğrulama ortalama hata değeri **0.64**'dür. Gerçek değerden mutlak 0.64 uzaklıktadır. Şekil 2.10 da bu modelin kayıp fonksiyonuna eğitim ve doğrulama değerleri için yer verilmiştir.



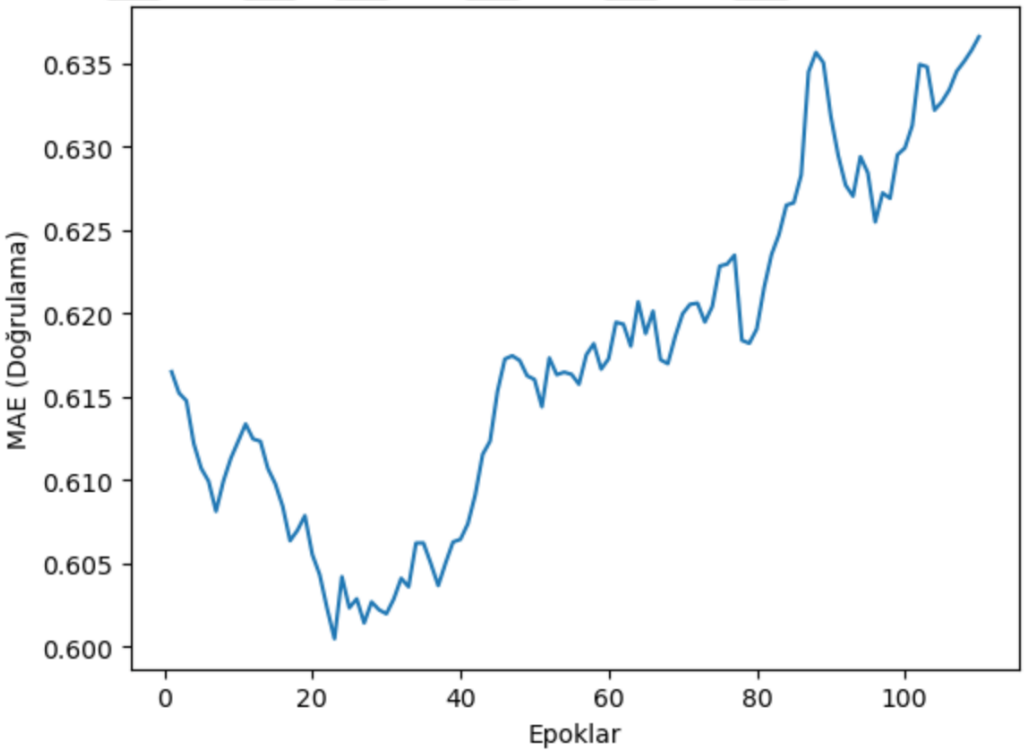
Şekil 2.10 Yapay sinir ağı modeli için epok sayısı ve kayıp fonksiyonu (Sınıf)

Kayıp fonksiyonu bize aşırı uydurmaya neden olacak noktayı belirlemekte fayda sağladı. 100. epoktan sonra doğrulama kaybı artmaya eğitim kaybı ise azalmaya başlamaktadır.

Benzer çalışma standart sapma ve maksimum değerler üzerinden de gerçekleşti. Standart sapma için doğrulama ortalama hata değeri 0.65 olarak elde edilirken maksimum değer için 695.277 olarak elde edildi. Ortalama hata değerinden de anlaşılacağı üzere maksimum değerler ile oluşturulan modeller başarılı doğruluk oranlarından çok uzaktır.

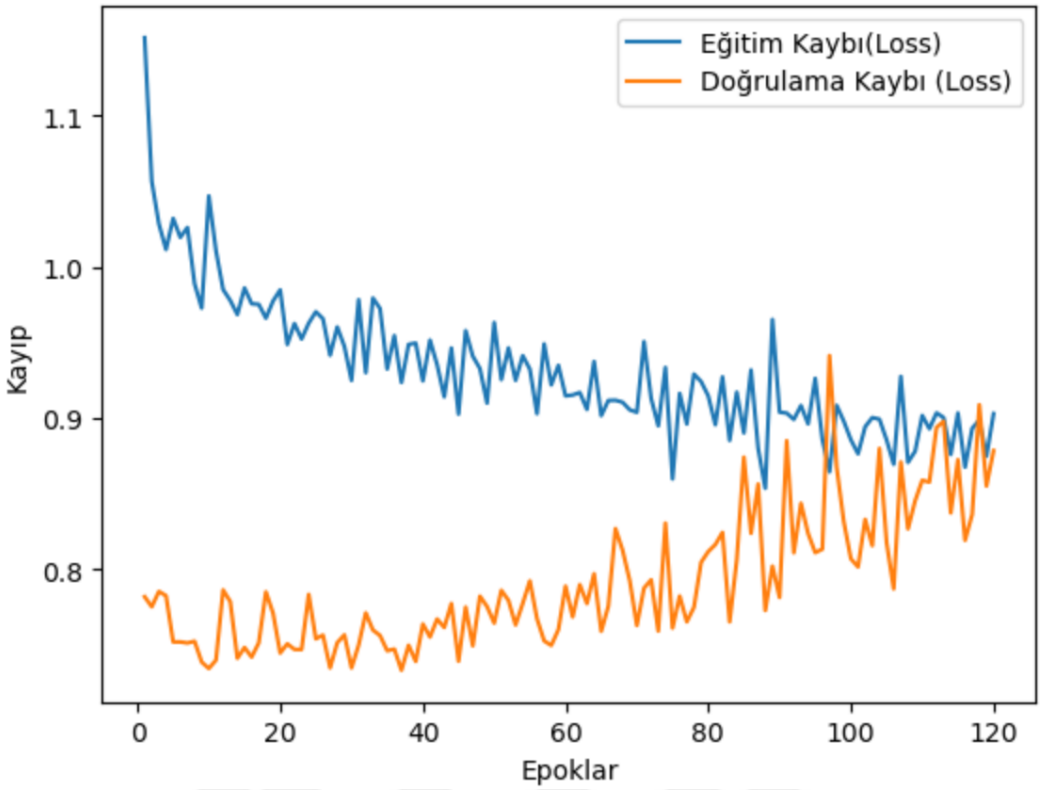
2.3.3.2 Metot Düzeyinde Sinir Ağı Modeli Hata Sayısı Tahmini

Benzer bir çalışmayı metot metrikleri ile elde ettiğimiz model üzerinde gerçekleştirilmiştir. Şekil 2.11 de metot metrikleri için yapay sinir ağı modeli için epok sayısı ve doğrulama değerleri gösterilmektedir.



Şekil 2.11 Yapay sinir ağı modeli için epok sayısı ve doğrulama değerleri (Metot)

K-fold döngüsü sonucunda elde edilen doğrulama ortalama hata değeri **0.62**'dür. Şekil 2.12 de bu modelin kayıp fonksiyonuna eğitim ve doğrulama değerleri için yer verilmiştir. Bu model için ideal epok 50 değeri olabilir çünkü model devamında eğitim verisi ile aşırı uydurmaya gitmeye başlamıştır.



Şekil 2.12 Yapay sinir ağı modeli için epok sayısı ve kayıp fonksiyonu (Metot)

Metot metriklerinde standart sapma için doğrulama ortalama hata değeri 0.68 olarak elde edilirken maksimum değer için yaklaşık 0.72 olarak elde edildi.

Bu çalışma sonucunda diğer çalışmalardan farklı olarak projelerdeki hata sayılarının metrikler ile ilişkisini incelemek yerine 3 projenin veri setleri birleştirilerek bir model oluşturulmaya çalışılmıştır. Böylelikle modeli hata tahmini konusunda farklı dinamiklerdeki projelerle besleyerek benzer projelerin hata tahminlerinde başarılı çıkarımlar üretmesi sağlanabilir. Sözgelimi bir firmada görev alan ve kalite operasyonlarını yöneten bir ekip yıllık KPI hedeflerini belirlerken projenin gidişatına ve önceki değerlerine dayanan bir hedef belirlemek yerine kurumsal yapay sinir ağı modelini kullanırsa, ekibin performansını ve verimliliğini doğru bir şekilde ölçmüş olur. Örneğin yeni başlayan bir projedeki hata sayısının yıllardır devam eden bir projeden daha fazla olmasını bekleyebiliriz. Diğer bir örnek hızlıca yapılmış ve bazı CK metriklerini kötü anlamda etkilemiş bir versiyonun sonraki sürümünün bu karmaşadan etkilenebilir ve hataya açık hale gelebilir.

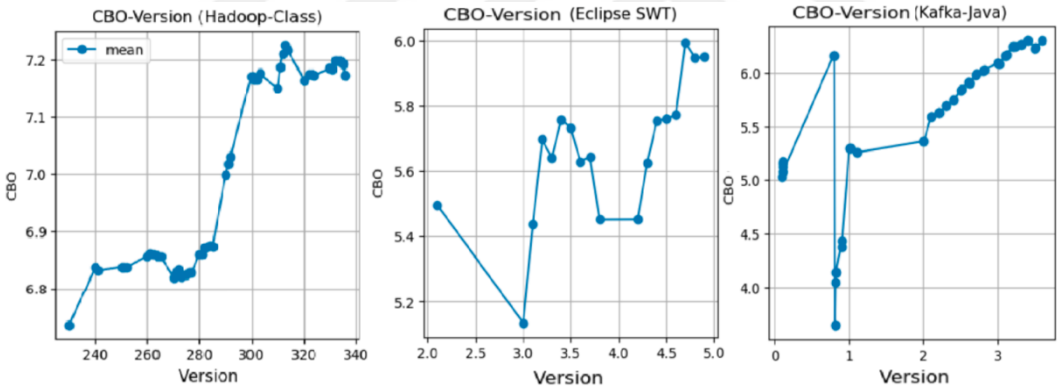
3
SONUÇ

3.1 Korelasyon Analiz Sonuçlarının Değerlendirilmesi

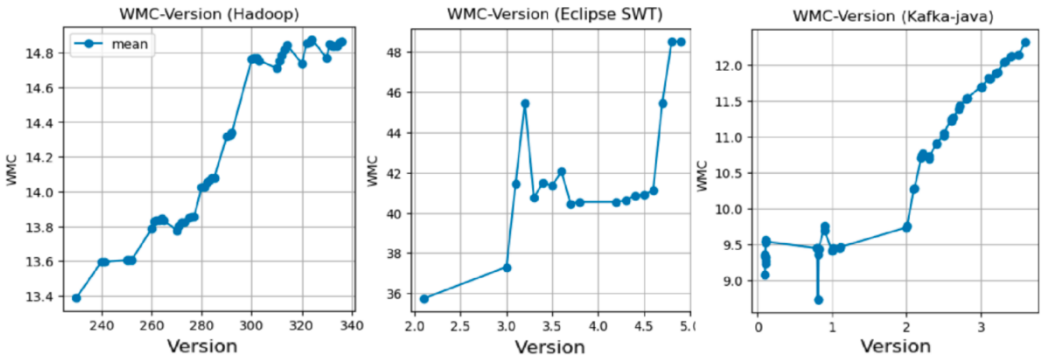
Analiz sonuçları üç farklı perspektiften ele alındı. İlk olarak, üç farklı proje için CK metriklerinin sürümler üzerindeki değişimleri ve dolaylı olarak zaman içindeki eğilimleri incelendi. İkinci olarak metrikler ile hata sayısı arasındaki ilişki incelendi, son perspektifte ise yazılım geliştirme süreçlerinde karşılaşılan iki türetilen metriğin analiz sonuçları değerlendirildi.

3.1.1 Metriklerin Zaman Bağlı Değişimi

Apache Hadoop, Eclipse SWT ve Kafka projeleri, CBO, CBO Modified, FAN-IN, FAN-OUT, WMC, DIT, NOC, LCOM, LCOM*, TCC, LCC ve RFC gibi metrikler temelinde ortalama, standart sapma ve maksimum değerler açısından analiz edildi. Bu çalışma belirli metriklere odaklanıldı ve grafiklerle bu metrikler incelendi.



Şekil 3.1 Hadoop, Eclipse SWT ve Kafka'nın sürümleri üzerindeki ortalama CBO dağılımları



Şekil 3.2 Hadoop, Eclipse SWT ve Kafka'nın sürümleri üzerindeki ortalama WMC dağılımları

Şekil 3.1'de, CBO metriğinin versiyon ilerlemesi boyunca ortalama değerler üzerinden gözlemlenen değişimi görülmektedir. Bu üç grafikte dikkat çeken durum, özellikle en son sürümlere doğru CBO parametresindeki artış eğilimidir. Şekil 3.2'de gösterildiği gibi, WMC parametresi, sürüm arttıkça değerinin arttığı şeklinde üç projede de benzer bir eğilim gösterdi. LCOM ve LCOM* metrikleri de benzer bir eğilim sergilemiştir. Burada dikkat edilmesi gereken önemli noktalardan biri, tüm üç projenin metriklerini elde etmek için aynı aracın kullanıldığıdır; özellikle Eclipse SWT projesinin CBO ve LCOM metrik ortalamalarının diğer iki projeden önemli ölçüde daha yüksek olduğu gözlemlenmiştir.

3.1.2 Korelasyon Analizi

Bu projelerde hata sayısının normal dağılıma uygunluğu başlangıçta incelendi. Yapılan Shapiro-Wilk testi, hata sayısının normal bir dağılıma uymadığını ortaya koydu. Sonuç olarak, Spearman korelasyon katsayısı incelendi. Bu ilişkinin analizi, Kafka'yı içermedi çünkü bu projenin hata verilerinin tam kaynağı hem Scala hem de Java dillerinde yazılmış olmasından dolayıdır.

Tablo 3.1'de, hata sayısı ile benzer yönlü ilişki sergileyen parametrelerin korelasyon katsayılarını sunmaktadır. Bu inceleme, Hadoop ve Eclipse SWT projelerinde aynı yönde lineer bir eğilim gösteren ortak metrikleri belirlemeyi kolaylaştırdı, bu da büyüklüğe dayalı bir değerlendirme yapılmasını sağladı. Sadece sıralama amacıyla sonuçları gözden geçirirken, LCOM, FAN-OUT ve CBO gibi metriklerin bu iki projede benzer şekilde hata sayısını önemli ölçüde etkileyebileceği gözlemlendi.

Tablo 3.1 Metriklerin korelasyon katsayıları

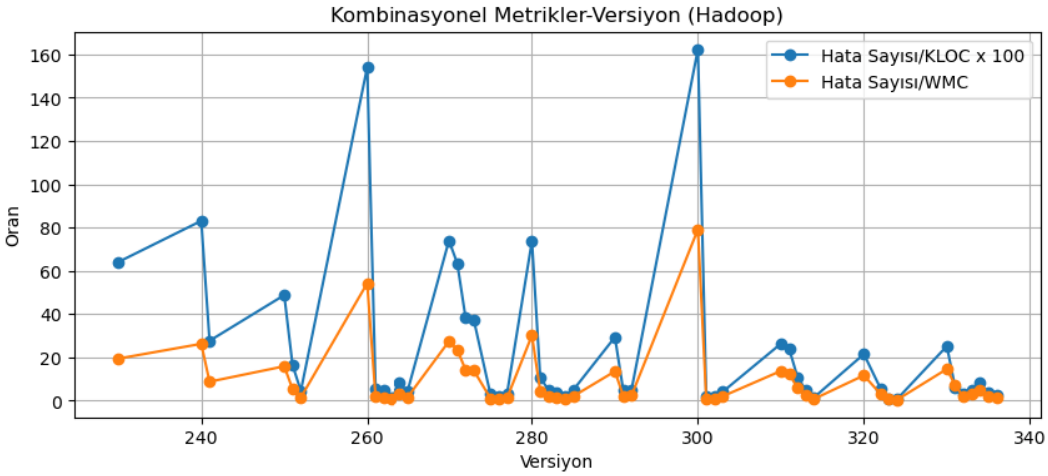
Project	CBO	CBO Modified	WMC	FAN-OUT	DIT	LCOM	TCC	LCC	LOC
Apache Hadoop	-0.17	-0.16	-0.32	-0.17	0.08	-0.14	0.10	0.07	-0.27
Eclipse SWT	-0.45	-0.59	-0.22	-0.64	0.20	-0.80	0.19	0.34	-0.48
Effect Percentage	%31	%37.5	%27	%38	%14	%47	%13	%20.5	%37.5

3.1.3 Türetilen Metriklerin Analizi

Proje yöneticileri ve araştırmacılar, yazılım projelerinden elde edilen özet hata yoğunluğu rakamlarından önemli ölçüde fayda sağlarlar. S. M. A. Shah, M. Morisio ve M. Torchiano'nun (2012) yaptığı çalışma, istatistiksel olarak, Java dilinde yazılmış bir projede ortalama hata yoğunluğu oranının 5.9 olduğunu

göstermektedir. Bu metriğin açık kaynaklı Hadoop projesindeki versiyonlar üzerindeki değişimi Şekil 3.3'te gözlemlenmektedir. Aynı şekilde, WMC başına hata oranı da sunulmuştur. Her iki dağılım birlikte gözlemlendiğinde, DD ile benzer bir dağılım sergilediği belirtilmiştir. Bu da parametrenin yazılım kalitesini incelemek için DD kadar etkili olduğunu önermektedir. Bu süreç boyunca DD değeri, diğer metriklerle uyumlu olması için 100 ile çarpılmıştır. Bu süreç boyunca gözlemlenen maksimum DD değeri 1.62 olmuştur ve sadece iki önemli çıkış değeri yaşamıştır. İlk çıkış değeri, 2.6.0 sürümünde gözlemlendi ve beta durumundaki birçok yeni özellik eklediği belirtilmiştir. Bunlar arasında farklı depolama türlerini destekleyen uygulama API'lerinin eklenmesi ve konteyner teknolojisinin Hadoop ekosistemiyle uyumlu hale getirilmesi bulunmaktadır. Diğer tepe ise Hadoop 3.0.0 ile gelmiştir. Bu sürümde YARN ve HDFS gibi bileşenler yeniden yapılandırılmış, GPU desteği eklenmiş ve Erasure Coding Desteği ile depolama sistemi iyileştirmeleri yapılmıştır.

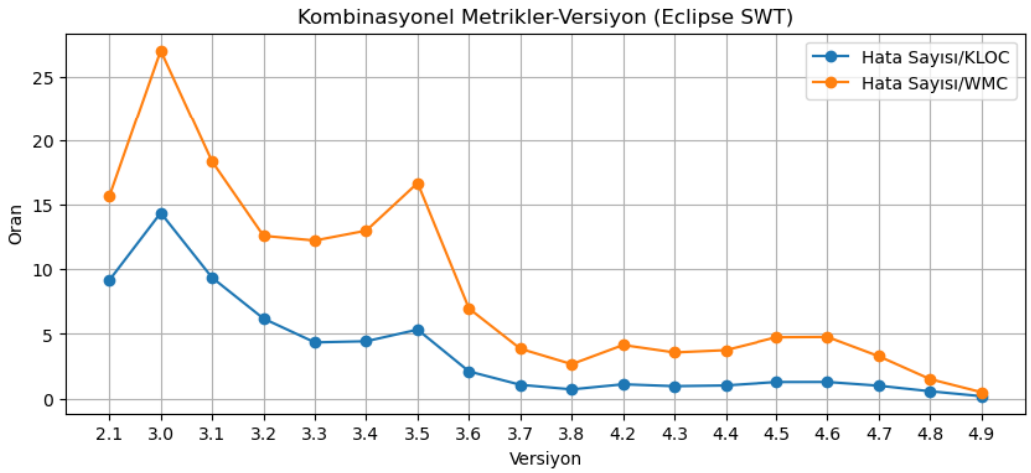
Aşağıdaki iki önerme çıkarılabilir: Bu özelliklerin yazılıma eklenmesi, onu hatalara daha duyarlı hale getirebilir veya bu yeni özellikler yazılımı kullanıcılar için daha faydalı hale getirmiş olabilir, bu nedenle daha fazla kullanılmış ve hataların ortaya çıkma olasılığı artmış olabilir.



Şekil 3.3 Eclipse SWT sınıf OLS regresyon tablosu

Şekil 3.4'te Eclipse SWT projesini incelediğimizde, proje 3.6 sürümüne kadar ortalama üstü bir DD oranına sahip olduğu söylenebilir. Benzer şekilde, bu dağılımda WMC başına hata değerleri neredeyse DD ile paralel seyretmektedir. Projenin başlangıç aşamalarında, düşük güvenilirlikle birlikte yüksek bir Hata

Yoğunluğu (DD) oranı sergilemiştir. Ancak, 3.1 güncellemesiyle önemli iyileştirmeler yapılmış ve ilk çıkış noktasını işaretlemiştir; daha sonra 3.5 sürümüyle daha fazla gelişme görülmüştür. Projenin istikrar kazandığı nokta ise 3.7 sürümüne kadar gelmemiştir. Özellikle Hadoop'ta belirgin olan, en son sürümlerde hata oranlarının düşük görünmesidir, bu muhtemelen bu sürümlerin henüz geniş çapta kullanılmamış olmasından kaynaklanabilir. Bu nedenle, en son sürümleri analiz ederken sonuç çıkarmak bazı zorluklar doğurabilir.



Şekil 3.4 Eclipse SWT sınıf OLS regresyon tablosu

Bu çalışma, başlangıçta, nesne yönelimli yazılım metrikleri ile hata sayıları arasında bir ilişki olup olmadığını araştırdı. Eğer bir ilişki varsa, hangi metriklerin özellikle ilişkili olduğuna ve bu bilgilerin proje yönetiminde güvenli ve yüksek kaliteli yazılım belirlemede nasıl faydalı olabileceğine odaklandı. Bu hedefe ulaşmak için yıllardır açık kaynak olarak geliştirilmiş farklı misyonlara sahip farklı projeler (Hadoop, Eclipse SWT, Kafka (Java)) incelendi. İlk yaklaşım, CK metriklerinin tüm bu yıllar boyunca ve çeşitli sürümlerdeki geçişini analiz etmeyi içeriyordu. Sonuçlar, bu üç rastgele seçilmiş projede CBO, WMC, LCOM ve LCOM* metriklerinin benzer dağılımlar sergilediğini ve sürüm ilerledikçe artma eğiliminde olduğunu gösterdi. Standart sapma değerleri de incelendi, ancak bunlar ortalama grafiklere benzer çıktılar verdiği için ayrıntılar sağlanmadı.

İkinci yaklaşım, bu parametreler arasındaki doğrusal ilişkiyi incelemeyi ve her iki projede hata sayıları ile en fazla ilişkili parametreleri belirlemeyi amaçlıyordu. Bunun için, normalite testini geçemeyen veriler için Spearman korelasyon katsayısı

değerlendirildi. Sonuçlar doğrultusunda, LCOM, FAN-OUT ve CBO'nun hata oranlarıyla daha fazla ilişkili olduğu gözlendi. Qureshi, M. R. J., ve Qureshi, W. A. (2012) tarafından yapılan çalışmada, LCOM metriğinin toplam hatalar üzerinde önemli bir etkisi olduğu belirtilmişti. Bu çalışmanın bulguları da LCOM metriği ile ilgili bu sonucu doğrulamaktadır. Hadoop'u incelediğimizde, DD ve CBO-WMC grafiklerinin yerel tepe noktalarının 2.6.0 ve 3.0.0 sürümleri için aynı olduğu gözlemlenmiştir. Ancak Eclipse SWT için benzer bir ilişki gözlenmemiştir.

Son olarak, birleşik parametre değerleri üzerinden veri seti incelendi. DD ve WMC başına hata gibi parametreler, bu iki projede benzer eğilimler gösterdi. Bu parametreler, geçmiş sürümler boyunca hem güvenilirlik hem de sürdürülebilirlik kriterleri için öngörü sağladı ve gelecekteki iyileştirmeler için belirli bilgiler sunabilir.

3.2 Çoklu Lineer Regresyon Sonuçlarının Değerlendirilmesi

Analiz çalışmaları aşamasında Apache Hadoop ve Eclipse SWT projelerinin sınıf ve metrik düzeylerinde OLS tabloları incelendi. OLS tabloları bize modelin çoklu regresyona uyduğunu ve hangi parametrelerin en çok katkı sağladığını anlamamız konusunda fikir verdi. Genel olarak sınıf veya metot düzeyinde iyi bir uygunluk oranına sahip olan regresyon modellerinde gruplanmamış metot düzeyinde CBO ve RFC metriğinin Eclipse SWT için bir etki sahibi olduğu ancak diğer metriklere bakıldığında iki projede de benzer kabul edilebilirlik katsayıları olmadığı gözlemlenmiştir.

Ortalama, standart sapma ve maksimum değerlerine bakıldığında iki proje de farklı karakteristikler göstererek farklı metrikleri önemli olarak belirlemişlerdir. 0.05 uygunluk oranlarımızın altında kalan yani model üzerinde etki sahibi metrikler Eclipse SWT sınıf için maksimum grup ortalaması ile WMC, LCOM ve CBO’dur. Metot için ise ortalama değer için CBO*, FANIN’dir. Standart sapma için ise WMC ve LOC’dur. Hadoop için ise sadece metot düzeyinde WMC, LOC, VERSION, FANOUT, CBO, CBO* ve FANIN’dir. Ayrıca bu çalışma göstermiştir ki gruplamaksızın elde edilen modellerde uygunluk oranı çok düşük ve standart hata çok yüksektir.

3.3 Hipotezlerin Değerlendirilmesi

Hipotez aşamasında ortaya atılan 13 sınıf ve 8 metot metriği mevcuttur. Bu hipotezler iki proje için korelasyon ve çoklu lineer regresyon yöntemleriyle incelenmiştir. İki projenin de geliştirime stratejileri veya tasarımından kaynaklı farklılıklar olacağı göz önüne alındığında belirlenen hedef değerin üstünde kalan bir katsayıya sahip herhangi bir proje olması H_0 hipotezinin yanlış olduğunu ve H_1 hipotezinin kabul edildiğini gösterir. Doğruluk değerimiz | 0.30 | olarak kabul edildi. Bu değerin üzerindeki katsayılar H_0 hipotezini reddederken H_1 ini kabul etmektedir. Çoklu lineer regresyon için ise P>|t| değeri 0.05'den küçük olan değerlerin model üzerinde güçlü bir etkisinin olduğu kabul edildi dolayısıyla bu değerler için H_0 hipotezi reddedildi.

Tablo 3.2 Hipotez değerlendirme özeti

Metrik		Korelasyon Analizi		Çoklu Lineer Regresyon		Kabul Edilen Hipotez Sonucu
		Hadoop	SWT	Hadoop	SWT	
N1	CBO		✓		✓ (Maksimum)	✓
N2	CBO*		✓			✓
N3	FANIN		✓			✓
N4	FANOUT		✓			✓
N5	WMC	✓			✓ (Maksimum)	✓
N6	DIT					
N7	NOC		✓			✓
N8	LCOM		✓		✓ (Maksimum)	✓
N9	LCOM*					
N10	TCC					
N11	LCC		✓		✓	✓
N12	LOC		✓			✓
N13	Versiyon		✓			✓

Tablo 3.3 Hipotez değerlendirme özeti (devamı)

N14	CBO		✓	✓ (Maksimum)	✓	✓
N15	CBO*		✓	✓ (Maksimum)	✓	✓
N16	FANIN		✓	✓ (Maksimum)	✓	✓
N17	FANOUT		✓	✓ (Maksimum)		✓
N18	WMC		✓	✓ (Maksimum)	✓ (σ)	✓
N19	RFC		✓		✓	✓
N20	LOC		✓	✓ (Maksimum)	✓ (σ)	✓
N21	Versiyon		✓	✓ (Maksimum)	✓	✓

Tablo 3.1'deki hipotez değerlendirme özeti tablosunda iki proje için hata oranları üzerinde etkili olan metrik değerleri işaretlenmiştir. En az bir yöntem ve projede kabul edilebilir etkiye sahip metotlar için H_0 hipotezi reddedilir, dolayısıyla H_1 hipotezi kabul edilir. Sonuç olarak N1, N2, N3, N4, N5, N7, N8, N11, N12, N13, N14, N15, N16, N17, M18, N19, N20, N21 toplamda 17 metrik için H_1 hipotezi kabul edilmiştir. Sınıf düzeyinde LOC, TCC, LCOM* ve DIT için de H_0 hipotezi kabul edilmiştir. Bu da "Nesne Yönelimli Metriklerin Açık Kaynak Yazılımlarda Hata Tahmini Üzerindeki Etkisinin Doğrulanması", "Yazılım Geliştirmede Hata Sayısını Azaltmak İçin Tasarım Metriğinin Değerlendirilmesi" ve "Yazılım hata tahmin metrikleri: Bir sistemli literatür incelemesi" adlı çalışmalardaki DIT parametresinin güvensiz olduğu bulgusunu doğrular niteliktedir (Gyimo´thy, Ferenc ve Siket, 2005) (Qureshi, M. R. J., ve Qureshi, W. A., 2012) (Radjenović vd., 2013).

Kullanılan veri seti sınırlı olduğundan, elde edilen sonuçların sadece Apache Hadoop ve Eclipse SWT projeleriyle sınırlı olabileceğini ve bu nedenle genellenebilirliğin daha dar olabileceğini unutmamak önemlidir. Daha kapsamlı bir veri seti, çeşitli projelerin ve versiyonların dahil edilmesiyle daha genelgeçer sonuçlara ulaşmada yardımcı olabilir.

KAYNAKÇA

Aggarwal, K. K., Singh, Y., Kaur, A., & Malhotra, R. (2006). Empirical Study of Object-Oriented Metrics. *The Journal of Object Technology*, *5*(8), 149. https://doi.org/10.5381/jot.2006.5.8.a5

Alqmase, M., Alshayeb, M., & Ghouti, L. (2019). Threshold Extraction Framework for Software Metrics. *Journal of Computer Science and Technology*, *34*(5), 1063–1078. https://doi.org/10.1007/s11390-019-1960-6

Bajwa, M. S., Singh, P. K., & Agarwal, A. P. (2016). Analyzing and interpreting the fault localized using PCA with CK metrics. In *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (s. 575-580). Waknaghat, India. https://doi.org/10.1109/PDGC.2016.7913189

Binanto, I., Warnars, H. L. H. S., Gaol, F. L., Abdurachman, E., & Soewito, B. (2018). Measuring the quality of various version an object-oriented software utilizing CK metrics. *2018 International Conference on Information and Communications Technology (ICOIACT)*. https://doi.org/10.1109/icoiact.2018.8350760

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, *20*(6), 476–493. https://doi.org/10.1109/32.295895

Dagpinar, M., & Jahnke, J. H. (2003). Predicting maintainability with object-oriented metrics -an empirical comparison. In *10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings.* (s. 155-164). Victoria, BC, Canada. https://doi.org/10.1109/WCRE.2003.1287246

Đurković, J., Vuković, V., & Raković, L. (2008). Open source approach in software development—Advantages and disadvantages. Manag Inforation Syst, 3, 029-33.

Gyimóthy, T., Ferenć, R., & Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering*, *31*(10), 897–910. https://doi.org/10.1109/tse.2005.112

Hadoop - Apache Hadoop 2.6.0. (2014). https://hadoop.apache.org/docs/r2.6.0/index.html

Hadoop – Apache Hadoop 3.0.0. (2020). https://hadoop.apache.org/docs/r3.0.0/index.html

HadoopEclipseSWT-Kafka-CK-JiraBugzilla metrics. (2023, November 23). Kaggle. https://www.kaggle.com/datasets/ozgesen/hadoopeclipseswt-kafka-ck-jirabugzilla-metrics

Kulkarni, U. L., Kalshetty, Y. R., & Arde, V. G. (2010). Validation of CK Metrics for Object Oriented Design Measurement. In 2010 *3rd International Conference on Emerging Trends in Engineering and Technology* (pp. 646-651). Goa, India. doi: 10.1109/ICETET.2010.159.

Mauricioaniche. (2015). *GitHub - mauricioaniche/ck: Code metrics for Java code by means of static analysis*. GitHub. https://github.com/mauricioaniche/ck/

Molnar, A., Neamţu, A., & Motogna, S. (2019). Longitudinal Evaluation of Software Quality Metrics in Open-Source Applications. *International Conference on Evaluation of Novel Approaches to Software Engineering*.

Qureshi, M. R. J., & Qureshi, W. A. (2012). Evaluation of the design metric to reduce the number of defects in software development. *International Journal of Information Technology and Computer Science*, *4*(4), 9–17. https://doi.org/10.5815/ijitcs.2012.04.02

Ozgesen. (2023). *ozgesen/HadoopSWTJupyterAnalyses*. GitHub. https://github.com/ozgesen/HadoopSWTJupyterAnalyses.git

Radjenović, D., Heričko, M., Torkar, R., & Živkovič, A. (2013). Software fault prediction metrics: A systematic literature review. *Information and Software Technology*, *55*(8), 1397-1418. https://doi.org/10.1016/j.infsof.2013.02.009

Tang, M.-H., Kao, M.-H., & Chen, M.-H. (1999). An empirical study on object-oriented metrics. In *Proceedings Sixth International Software Metrics Symposium (Cat. No.PR00403)* (s. 242-249). Boca Raton, FL, USA. https://doi.org/10.1109/METRIC.1999.809745

Schober, P., Boer, C., & Schwarte, L. A. (2018b). Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia & Analgesia*, *126*(5), 1763–1768. https://doi.org/10.1213/ane.0000000000002864

Shah, S. M. A., Morisio, M., & Torchiano, M. (2012). An Overview of Software Defect Density: A Scoping Study. In 2012 *19th Asia-Pacific Software Engineering Conference* (pp. 406-415). Hong Kong, China. doi: 10.1109/APSEC.2012.93.

Sharma, R. K., & Gandhi, P. (2019). Study of Reliability of Object-Oriented Structure Consuming CK Metrics. In *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)* (s. 828-831). New Delhi, India.

Shatnawi, R., & Li, W. (2006). An Investigation of Bad Smells in Object-Oriented Design. *Third International Conference on Information Technology: New Generations (ITNG'06),* 161-165. https://doi.org/10.1109/itng.2006.31

Shatnawi, R. (2010). A quantitative investigation of the acceptable risk levels of Object-Oriented metrics in Open-Source systems. *IEEE Transactions on Software Engineering*, *36*(2), 216–225. https://doi.org/10.1109/tse.2010.9

Strigini, L., Bosio, D., Littlewood, B., & Newby, M. (2002). Advantages of open source processes for reliability: clarifying the issues. In Proceedings of the Open Source Software Development Workshop (pp. 30-46).

Yuniasri, D., Rochimah, S., & Raharjo, A. B. (2020). A Correlation Analysis Between ISO 25010 based Modularity and CK Metrics in Object-Oriented Software. In 2020 *International Conference on Advanced Science and Engineering (ICOASE)*, (s. 1-6). doi: 10.1109/ICOASE51841.2020.9436617.

A
OLS TABLOSUNDAKİ ÖLÇÜM DEĞERLERİ

OLS Tablosundaki Ölçüm Değerleri

OLS tabloları Lineer regresyon analizinde yardımcı pek çok istatistik ölçümü sunar. Aşağıdaki Tablo Ek A. de bu istatistik değerlerinin açıklamalarına yer verilmiştir.

Tablo A.1 OLS tablosu istatistik değerleri ve açıklamaları

Değer	**Açıklama**
R^2	Modelin bağımsız değişkenler tarafından açıklanan varyansın oranını ölçer. Bağımsız değişkenlerin bağımlı değişkenleri ne kadar iyi açıkladığının bir ölçütüdür.
Adj. R^2	Model içerisinde çok fazla bağımsız değişken olduğunda R^2 yüksek uyum indeksine sahip olabiliyor. Adj. R2 gereksiz bağımsız değişkenlerin modele dahil edilmediği bir şekilde model uyumluluğunun oranını verir.
F-statistic	Modelin anlamlılığının bir ölçüsüdür. Yüksek bir değere sahip olması en az bir değişkenin modele etkisi olduğunu belirtir.
Prob (F-statistic)	F-statistic değerinin olasılık değerini ifade eder. Genellikle belirlenen anlamlılık düzeyinin altında kalması modeli kabul edilebilir kılar.
Log-Likelihood	Log-Olasılık şeklinde çevrilebilen değerlendirme sonucu Modelin veriyi ne kadar iyi açıkladığını belirten bir ölçüttür. Normal dağılıma sahip hata dağılımında anlamlı sonuçlar vermesi beklenir.

Tablo A.1 OLS tablosu istatistik değerleri ve açıklamaları (devamı)

AIC	Akaike ölçütü olarak tanımlanabilen ölçü, istatistiksel modelleri kıyaslamayı sağlar. Oluşturulabilecek diğer modelleri hesaplar ve hangi modelin verileri daha uyumlu olduğunu ve aşırı uydurmaya dirençli olduğunun tespitinde kullanılır. Düşük AIC değeri modelin daha iyi uyum sağladığını gösterir.
BIC	Bayesian Bilgi Kriteri, AIC gibi modelin uygunluğunu değerlendirir. Farkı parametre sayısının artmasından AIC'den daha fazla etkilenir. Düşük olması model seçiminde kullanılan önemli bir kriterdir.
Df Residuals	Derece serbestlik hatalarını ifade eder. Model uygunluğunu değerlendiren bir başka ölçüttür. Gerçek ve tahmin değerleri arasındaki farkı hesaplar.
Omnibus	Omnibus testi, regresyon modelinin tüm bağımsız değişkenlerin bir arada model üzerinde anlamlı bir etkisi olup olmadığını değerlendiren başka bir testtir. F-statistic'in aksine modelin tamamının anlamlılığını ölçer.
Prob(Omnibus)	Omnibus değerinin olasılık değerini ifade eder. Genellikle belirlenen anlamlılık düzeyinin altında kalması modeli kabul edilebilir kılar.
Skew	Modelin hata terimlerinin dağılımının çarpıklığını belirler. Eğer Skew değeri sıfıra yakınsa, bu hata terimlerinin dağılımının simetrik olduğunu gösterir. Dağılım sola çekik olduğunda negatif, sağa çekik olduğunda pozitif değer alır.
Kurtosis	Modelin hata terimlerinin basıklığını ifade eder. Normal dağılım için değeri 0dır. 0'ın üzeri sivri bir dağılımı, altı ise düze yakın bir dağılımı işaret eder.

Tablo A.1 OLS tablosu istatistik değerleri ve açıklamaları (devamı)

Durbin-Watson	Durbin-Watson istatistiği hata terimlerinin birbirleri arasındaki ilişkiyi ölçer ve otokorelasyonun varlığını kontrol eder. Değeri 2 ye yakınsa otokorelasyonun olmadığını, 2'den küçükse pozitif, büyükse negatif otokorelasyonu işaret eder.
Jarque-Bera (JB)	Hata terimlerinin normal dağılıma sahip olup olmadığını test etmek için kullanılır.
Prob(JB)	Prob(JB) değerinin olasılık değerini ifade eder. Genellikle belirlenen anlamlılık düzeyinin altında kalması modeli kabul edilebilir kılar.
Cond. No.	Regresyon modeldeki bağımsız değişkenler arasındaki korelasyon veya ilişkinin derecesini ölçer. Bu değerin yüksek olması bağımsız değişkenler arasında da ilişkinin olduğuna dair bir işarettir.
coef	Çok parametreli lineer regresyondaki sabit değeridir.
std err	Standart hata, bir regresyon katsayısının tahmini değerinin gerçek değerden ne kadar sapma gösterebileceğini ölçer.
t	t değeri katsayının anlamlılığını kontrol eder. Regresyon katsayısının standart hataya bölümü ile elde edilir.
P>|t|	Regresyon katsayısının anlamlılığını ifade eder ifade eder. Belirlenen anlamlılık düzeyinin altında kalması regresyon katsayısının anlamlı olduğuna işaret eder.
[0.025 - 0.975]	Regresyon katsayısının güven aralığını ifade eder. Regresyon katsayılarının tahminlerinin belirsizliğini ve güven düzeyini değerlendirmek için kullanılır.

TEZDEN ÜRETİLMİŞ YAYINLAR

Konferans Bildirileri

1. Çolak, Ö. E., & Selçuk, Y. E. (2023). Examining the relationship of object-oriented metrics with bug rates in long-term open source systems. *2nd International Symposium Series On Graduate Research (DEUISGR23).*