

**TÜRKİYE CUMHURİYETİ**  
**YILDIZ TEKNİK ÜNİVERSİTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**YAZILIM HATA KAYITLARINDAN YENİDEN İŞLEME**  
**EMEK/ZAMAN KESTİRİMİ**

17011078 – Serkan YILDIRIM  
18011058 – Sedat Tuna AKIN

**BİLGİSAYAR PROJESİ**

Danışman  
Dr. Öğr. Üyesi Göksel BİRİCİK

Şubat, 2022



## TEŞEKKÜR

---

Çalışmanın gerçekleştirilmesinde her aşamada bizlere yardımcı olan, kıymetli zamanını bizimle paylaşan ve değerli bilgilerini bize aktaran saygı değer danışmanımız Dr. Öğr. Üyesi Göksel BİRİCİK'e teşekkürlerimizi sunuyoruz

Serkan YILDIRIM

Sedat Tuna AKIN

# İÇİNDEKİLER

---

<b>KISALTMA LİSTESİ</b>	<b>v</b>
<b>ŞEKİL LİSTESİ</b>	<b>vi</b>
<b>ÖZET</b>	<b>vii</b>
<b>ABSTRACT</b>	<b>ix</b>
<b>1 Giriş</b>	<b>1</b>
<b>2 Ön İnceleme</b>	<b>3</b>
<b>3 Fizibilite</b>	<b>5</b>
3.1 Teknik Fizibilite . . . . .	5
3.1.1 Yazılım Fizibilitesi . . . . .	5
3.1.2 Donanım Fizibilitesi . . . . .	5
3.2 Yasal Fizibilite . . . . .	5
3.3 Ekonomik Fizibilite . . . . .	5
<b>4 Sistem Analizi</b>	<b>6</b>
4.1 Veri Akış Diyagramı . . . . .	9
<b>5 Sistem Tasarımı</b>	<b>10</b>
5.1 Yazılım Tasarımı . . . . .	10
5.2 Veritabanı Tasarımı . . . . .	12
5.3 Girdi-Çıktı Tasarımı . . . . .	12
<b>6 Uygulama</b>	<b>14</b>
6.1 Korelasyon Matrisi . . . . .	14
6.2 Feature Selection . . . . .	16
6.3 Regresyon . . . . .	17
<b>7 DeneySEL Sonuçlar</b>	<b>18</b>
<b>8 Performans Analizi</b>	<b>20</b>

<b>9 Sonuç</b>	<b>22</b>
<b>Referanslar</b>	<b>23</b>
<b>Özgeçmiş</b>	<b>24</b>

## KISALTMA LİSTESİ

---

ACT	Annual Change Traffic
NNL	Number Of New Lines
NML	Number Of Modified Lines
NOL	Number Of Original Lines
FP	Function Point
CAF	Complexity Adjustment Factor
UFP	Unadjusted Function Point
EI	Number Of External Input
EO	Number Of External Outputs
EQ	Number Of External Inquiries
ILF	Number Of Internal Files
EIF	Number Of External Interfaces

## ŞEKİL LİSTESİ

---

Şekil 2.1	İncelenen yöntemler . . . . .	4
Şekil 4.1	Veritabanının ER Diyagramı . . . . .	6
Şekil 4.2	Veri Akış Diyagramı . . . . .	9
Şekil 5.1	Import ve Önışleme Aşaması . . . . .	10
Şekil 5.2	Korelasyon Matrisi ve veri setlerini oluşturma aşaması . . . . .	11
Şekil 5.3	Regresyon Yöntemleri İle Elde Edilen En İyi Sonuçlar . . . . .	12
Şekil 5.4	Elde Edilen Efor Formülü . . . . .	12
Şekil 5.5	Arayüz . . . . .	13
Şekil 6.1	Veri kümesinin korelasyon matrisi . . . . .	15
Şekil 6.2	Feature Seleciton Algoritmasından Sonra Her Özellik İçin Oluşan Değerler . . . . .	17
Şekil 7.1	Efor Formülü'nün İlk Adımı . . . . .	18
Şekil 7.2	Efor Formülünün İlk Halinin Gerçek Efora Oranı . . . . .	19
Şekil 7.3	Regresyon Analizinden Elde Edilen Katsayılar . . . . .	19
Şekil 8.1	Efor Formülünün Son Halinin Gerçek Efora Oranı . . . . .	20
Şekil 8.2	Elde Edilen Efor Formülünün COCOMO ile Karşılaştırılması . . . . .	21

# YAZILIM HATA KAYITLARINDAN YENİDEN İŞLEME EMEK/ZAMAN KESTİRİMİ

Serkan YILDIRIM

Sedat Tuna AKIN

Bilgisayar Mühendisliği Bölümü

Bilgisayar Projesi

Danışman: Dr. Öğr. Üyesi Göksel BİRİCİK

Yazılımlar için yapılan emek/zaman hesaplaması çalışmaları incelendiğinde yazılım geliştirmesi öncesi yapılacak hesaplamalar için geliştirilen uygulamalar bu çalışmalar arasında çok büyük bir yer kaplamakta. Bunun yanında mevcut yazılımlar için yapılan bakım ve geliştirme çalışmalarında emek/zaman kestirimi için geliştirilen uygulamalar ise gelişen ve canlanan yazılım ekosistemi için yetersiz kalmaktadır. Bizler de bu nedenle bu alan üzerinde çalışma yapmanın doğru olacağını düşünerekten danışmanımızın da onayı ile proje üzerinde çalışmalarımıza başladık.

Başlangıç olarak yaptığımız araştırmalar sonucunda COCOMO, KLOC ve FP gibi yöntemlerin mevcut çalışmalar arasında bulunduğunu gözlemledik ve çalışmalarımızda kullanabilmek adına detaylı incelemelerde bulunduk. Bu yöntemlerin yanında mevcut yazılımlar üzerinde yapılmış olan geliştirme kayıtlarının bulunduğu veri tabanı ihtiyacımızı gidermek adına arayışlarda bulunduk. Danışmanımızın yardımı aracılığı ile ulaştığımız veri tabanı üzerinde yapılan incelemeler sonrasında KLOC yönteminin çıkarılan veri seti üzerinde daha isabetli sonuç vereceği çıkarımında bulunup algoritmamızın çıktıları ile karşılaştırma kararı aldık.

Kullanılan veri tabanı üzerinde işlemler yapılarak algoritmalar için kullanılacak farklı veri setleri oluşturulmuştur. İlk olarak kopyalanmış olan veriler temizlenmiştir ve bir hata kaydı için açılan farklı commitler bir araya getirilmiştir. Veri kümesi sadeleştirilip kullanılabilir bir hale geldikten sonra kullanılacak özelliklerin tespiti için



korelasyon matrisi yöntemi ve özellik seçimi algoritması kullanılarak farklı veri setleri oluşturulmuştur.

Algoritmanın eğitimi için hangi yöntemin ve hangi veri setinin kullanılacağına karar vermek için KNN (K-Nearest Neighbors) algoritması ve Linear Regression algoritması çıkarılan tüm veri setleri üzerinde denenerek en isabetli sonucu çıktı olarak veren algoritma ve veri seti ikilisi belirlenmiştir.

Yapılan çalışmalar sonrasında kullanılan farklı veri setleri ve regresyon modelleri ile veri seti yetersizliği sebebi ile istenilen yüksek  $R^2$  skoruna ulaşamamıştır. Bu nedenle çalışmanın devamında farklı bir yönde ilerlenmiş ve elimizde bulunan özelliklerin çalışma içerisinde regresyon modellerinde ve korelasyon matrisinde yer alan puanlarına göre uygun görülenler kullanılarak yazılım yeniden işleme emek/zaman kestirimi yapabildiğimiz formül oluşturulmuştur.

**Anahtar Kelimeler:** Bakım, Efor, COCOMO, KLOC, FP, Özellik Seçimi, Korelasyon Matrisi, KNN (K-En yakın komşu), Doğrusal Regresyon Regression

## ABSTRACT

---

### SOFTWARE MAINTENANCE EFFORT ESTIMATION

Serkan YILDIRIM

Sedat Tuna AKIN

Department of Computer Engineering

Computer Project

Advisor: Assist. Prof. Dr. Göksel BİRİCİK

When the effort/time calculation studies for software are examined, the applications developed for the calculations to be made before the software development occupy a very large place among these studies. In addition, applications developed for effort/time estimation in maintenance and development studies for existing software are insufficient for the developing and living software ecosystem. For this reason, we started to work on the project with the approval of our consultant, thinking that it would be right to work on this area.

As a result of our initial research, we observed that methods such as COCOMO, KLOC and FP are among the existing studies, and we made detailed examinations in order to use them in our studies. In addition to these methods, we searched in order to meet our need for a database containing development records on existing software. After the examinations on the database we reached through the help of our consultant, we concluded that the KLOC method would give more accurate results on the extracted data set, and we decided to compare it with the outputs of our algorithm.

Different data sets to be used for algorithms were created by performing operations on the database. Firstly, the data that was copied was cleaned and different commits for a job were combined. After the data set was simplified and made usable, different data sets were created by using the Correlation Matrix method and Feature Selection algorithm to determine the features to be used.

In order to decide which method and which data set to use for the training of

the algorithm, we tested the KNN (K-Nearest Neighbors) algorithm and the Linear Regression algorithm on all the data sets we extracted and determined the pair of algorithm and data set that gave the most accurate result as output.

After the studies, the desired high R2 score could not be reached due to the lack of data. For this reason, in the continuation of the study, we proceeded in a different direction and a formula was created by which we can make software rework effort/time estimation by using the ones found appropriate according to the scores of the features we have in the regression models and correlation matrix within the study.

**Keywords:** Maintenance, Effort, COCOMO, KLOC, FP, Future Selection, Correlation Matrix, K-Nearest Neighbors, Linear Regression

# 1

## Giriş

---

Bu proje kapsamında yazılım bakımı maliyetinin emek/zaman olarak bakım öncesi kestirimi yapılabilmesi için mevcut yazılım hata ve geliştirme kayıtları üzerinde çalışma yapılmıştır.

Yazılımlar için emek/zaman kestirimi konusu ile ilgili çalışmalara bakıldığında çoğunlukla yazılıma başlanmadan önce projenin emek/zaman kestirimi ve bu konu ile ilgili yöntemler ele alınmıştır fakat şirketlerin yazılımlarının bakımları ile ilgili ciddi giderlerinin olduğunu da ele alırsak yazılım bakımı da oldukça önemli bir konudur.

Projenin amacı elde bulunan veriler içerisinde korelasyon matrisi ve özellik seçimi algoritmaları kullanılarak alınan çıktıya göre uygun veriler ile regresyon ve KNN algoritmaları kullanılarak emek/zaman kestiriminin yapılmasıdır.

Proje bakımının emek/zaman kestirimi için ilk adım eldeki yazılımın büyüklüğünü ve karmaşıklığının tespitidir. Bu tespit için kullanılan yöntemler; KLOC ve FP olarak öne çıkar. KLOC yöntemini kullanarak aldığımız sonucu algoritmamızın çıktısı ile karşılaştırarak çalışmamızın doğruluğunu bu yöntemle de kontrol etmiş oluyoruz.

Projede kullanılacak olan modelin eğitimi ve farklı metrikleri ile yeni yöntemler oluşturulabilecek, gerçek projelerin hata kayıtlarının yeniden işlemlerinden elde edilmiş metriklerin bulunduğu bir veri seti kullanılması gerekmektedir. Yapılan araştırmalardan sonra 33 farklı projeden toplanan metriklerin bulunduğu veri kümesi kullanılmaya karar verilmiştir.[1]

Kullanılan veri kümesi üzerinde işlemler yapılarak algoritmalar için kullanılacak farklı veri setleri oluşturulmuştur. Öncelikle veri kümesi içerisinde kopyalanmış olan satırların temizlenmesi ve açılmış olan iş kaydı için birden fazla olan commitlerin birleştirilmesi işlemleri yapılmıştır. Sonrasında kullanılacak özelliklerin tespiti için özellik seçimi algoritması ve korelasyon matrisi uygulanarak farklı veri setleri oluşturulmuştur.

Sonrasında algoritmanın eğitimi için bu veri setleri üzerinde KNN (K-Nearest Neighbors) algoritması ve Linear Regression algoritması kullanılarak alınan çıktılar birbiri arasında karşılaştırılarak kullanılan veri setleri içerisinde en tutarlı çıktıyı veren veri seti son adımda algoritmanın eğitim seti olarak kullanılmıştır.

Yapılan çalışmalar sonrasında kullanılan farklı veri setleri ve regresyon modelleri ile veri seti yetersizliği sebebi ile sonuca ulaşamamıştır. Bu nedenle çalışmamızın devamında regresyon modellerinden ve korelasyon matrisinden uygun gördüğümüz değerler alınarak yazılım yeniden işleme emek/zaman kestirimi yapabildiğimiz formül oluşturuldu.

2. Bölüm olan Ön İnceleme bölümünde bu zamana kadar yapılan çalışmalar ve kullanılan yöntemler açıklanmıştır. 3.Bölümde projenin fizibilitesi yapılmıştır. 4. Bölümde ise oluşturulan sistemin analizi yapılmıştır. 5.Bölümde oluşturduğumuz sistemin tasarımı anlatılmıştır. 6.Bölümde prototip uygulamanın kodları yer almaktadır. 7.Bölümde performans analizi, 8.Bölümde sonuç yer almaktadır.

## 2 Ön İnceleme

---

Yazılım projesinden önce kullanılabilecek emek/zaman kestirimi ile ilgili bir çok çalışma yapıldığı görülmektedir fakat yazılım hatalarının yeniden işlemesiyle ilgili pek çalışma olmadığı görülmektedir. Yazılımdan önce kullanılabilecek çalışmalardan en bilinenleri COCOMO modelidir. COCOMO modeli yazılım büyüklüğünü KLOC yöntemi esas alarak hesaplar ve formülündeki katsayı organik, yarı bağımsız ve gömülü projeler için değişiklik gösterir. COCOMO modeli bir çok model için bir şablon olarak kullanılmıştır. Esasında bir yazılım bakım emek/zaman kestirimi yöntemi değildir fakat uygun parametreler ve formüller ile bu işlem için kullanılabilmektedir.[2]

Yapılan incelemede görülmektedir ki üretilen neredeyse bütün modeller COCOMO modelinden türetilmiştir. Bu şekilde ortaya çıkan COCOMO 2.0 modeli bir emek/zaman kestirimi modelinden daha çok proje yönetimi ile ilgilidir. Bu sebeple çok fazla girdi ihtiyacı olup emek/zaman kestirimi için uygun değildir.[2]

ACT modeli bir yazılımın yıl içerisinde ne kadar değiştiğini belirten bir hesaplama yapmaktadır. Toplam yeni satır ve değiştirilen satır sayılarının toplamalarının orijinal satır sayısına oranlanması ile bulunur. Proje büyüklüğünün hesaplanıp bu değerle çarpılması bakım emek/zaman değerinin kestiriminde kullanılmaktadır.[3]

Yapılan çalışmalarda yazılımlarının büyüklük ve karmaşıklığı için FP kullanmanın çok daha tutarlı bir yöntem olduğu bulunmuştur. Bu sebeple FP'lerin bir proje ölçeği olarak kullanıldığı bir çok çalışma bulunmaktadır.[4] Yapılan çalışmalar FP arttıkça eforun arttığını matematiksel formüller ile ortaya koymuştur.

Model	Maintenance Type	Metrics	Effort
<b>COCOMO (Regression Model)</b>	Regular Maintenance	SLOC(KLOC) added and modified, parameters for organic, semi-detached and embedded projects,	<b>Organic projects</b> = $2.4(KLOC)^{1.05}$ PM <b>Semi-detached</b> = $3.0(KLOC)^{1.12}$ PM <b>Embedded</b> = $3.6(KLOC)^{1.20}$ PM PM=Person-Month
<b>The ACT Model Annual Change Traffic (ACT)</b>	Annual Maintenance	Number of new lines (NNL), number of modified lines(NML), number of original lines(NOL)	$ACT = (NNL + NML) / (NOL)$ Effort= $ACT * 2.4(KLOC)^{1.05}$
<b>The Function Point (FP) Model (Albrecht's FP revision model)</b>	Regular Maintenance	Number of functions, VAF or basic CAF.	Effort = $a * FP^b$
<b>COCOMO 2.0 Reuse Model</b>	Regular Maintenance	ASLOC (adapted SLOC), AA (assessment and assimilation), SU(software understanding), DM(design modification), CM(code modification), IM(integration redone during reuse), SF(scaling factor), EM (effort multipliers).	$SLOC = (ASLOC * (AA + SU + 0.4 * DM + 0.3 * CM + 0.3 * IM)) / 100$ Effort= $A * [size]^{1.01 + \sum_{i=1}^{17} SF_i * I_i - 17} \prod EM_i$

**Şekil 2.1** İncelenen yöntemler

### 3.1 Teknik Fizibilite

#### 3.1.1 Yazılım Fizibilitesi

Projenin bu zamana kadar yazılmış kodları için python kullanılmıştır. Model için kullanılacak veri kümesi SQLite üzerinde çalıştırılmış ve gerekli veri kümesinin çıkarımı SQL ile gerçekleştirilmiştir.

#### 3.1.2 Donanım Fizibilitesi

Bilgisayarın belleğinin kullanılacak veri kümesinin büyüklüğünden fazla olması gerekmektedir. En az 10GB bellek olması yeterli olacaktır.

### 3.2 Yasal Fizibilite

Kullanılan veri kümeleri açık kaynak bir şekilde paylaşılan kümelerdir. Bir yasal yükümlülükleri bulunmamaktadır.

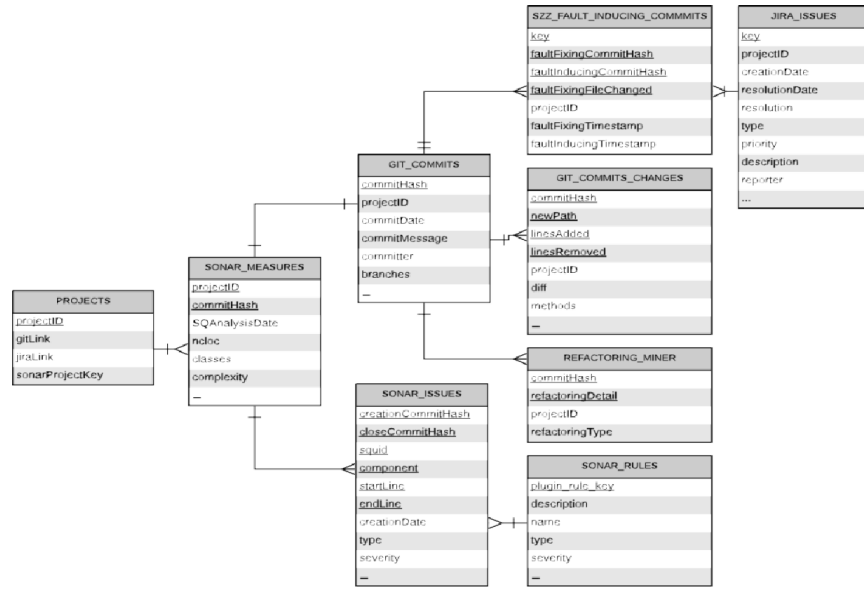
### 3.3 Ekonomik Fizibilite

Programın çalıştırılacağı bilgisayarın elektrik gideri dışında bir gider bulunmamaktadır.



# 4

## Sistem Analizi



Şekil 4.1 Veritabanının ER Diyagramı

Projemizde, yazılımlardaki hataların çözümlerindeki efor değerinin tahminlenmesi için proje ve hata ile ilgili hangi bilgilerin kullanılabileceğinin analizi yapılmıştır. Daha önce bu alanda yapılan çalışmalar incelenerek kullanılan yöntemlerde hangi özelliklerin önemli oldukları göz önünde bulundurularak farklı regresyon modelleri ile efor tahmini yapılmaya çalışılmıştır. Model oluşumunda gerekli olan veri kümesi arayışı literatürde, bu konu ile ilgili kullanılabilecek açık kaynaklı veri kümesi sayısının az olması ve bulunanların karşılaştırma yapılabilecek gerçek efor değerleri bulunmadığı için zorlu bir problem olarak karşımıza çıkmıştır.

Bizim projemiz için seçtiğimiz veri kümesi, JAVA'da yazılmış, 3 yıldan daha eski, 500 adet commit'ten ve 100 adet class'tan fazla olan projelerden, Pydriller, Ptidej, Refactoring Miner, SonarQube ve SZZ Algorithm araçları ile JIRA'da açılmış hata kayıtları için commit edilen kodların ve dosyaların özelliklerini barındıran bir veri kümesidir.[1] Veri setinin kendisi bir veritabanı olarak kullanılmaktadır. 9 adet tablodan oluşmaktadır. Tablolar arasındaki ilişkileri gösteren ER

Diyagramı Şekil 4.1’de verilmiştir. Veritabanından çekilecek bilgiler için SQL kullanılmıştır. SONARMEASURES tablosunda bulunan proje ile ilgili bilgileri JIRAISSUES tablosundaki kapatılmış olan kayıtlar ile commitHash değerleri ile eşleştirerek verilerin çekilmesi sağlanmıştır. Hem veritabanının büyüklüğünden hem de SQL sorgusu içerisinde fazla sayıda join bulunmasından dolayı verileri çekme işlemi uzun sürmüştür.

Veritabanından bizim için gerekli bilgileri çekmek için kullanılan SQL sorgusu aşağıdaki gibidir:

```
Select
JIRA_KEY, sum(linesAdded),sum(linesRemoved),sum(effort),
classes,files,functions,commentLines,commentLinesDensity,
fileComplexity,classComplexity,functionComplexity,duplicatedLines,
duplicatedBlocks,duplicatedFiles,duplicatedLinesDensity,
blockerViolations,criticalViolations,infoViolations,confirmedIssues,
codeSmells,bugs,cognitiveComplexity,lines,ncloc,linesToCover,
majorViolations,minorViolations,openIssues,sqaleRating,
missingPackageInfo,package,statements,uncoveredLines,sqaleIndex,
developmentCost,sqaleDebtRatio,newSqaleDebtRatio,vulnerabilities,
reliabilityRemediationEffort,reliabilityRating,
securityRemediationEffort,securityRating

from(Select DISTINCT
JIRA_ISSUES.key as JIRA_KEY,
SZZ_FAULT_INDUCING_COMMITS.key ,
SZZ_FAULT_INDUCING_COMMITS.faultFixingCommitHash,
GIT_COMMITS_CHANGES.linesAdded, GIT_COMMITS_CHANGES.linesRemoved,
GIT_COMMITS_CHANGES.ncloc, GIT_COMMITS_CHANGES.complexity,
SONAR_ISSUES.effort,classes,files,functions,commentLines,
commentLinesDensity,
fileComplexity,classComplexity,functionComplexity,duplicatedLines,
duplicatedBlocks,duplicatedFiles,duplicatedLinesDensity,
blockerViolations,
criticalViolations,infoViolations,confirmedIssues,codeSmells,bugs,
cognitiveComplexity,lines,ncloc,linesToCover,majorViolations,
minorViolations,
openIssues,sqaleRating,missingPackageInfo,package,statements,
uncoveredLines,
```

```

sqaleIndex,developmentCost,sqaleDebtRatio,newSqaleDebtRatio,
vulnerabilities,
reliabilityRemediationEffort,reliabilityRating,
securityRemediationEffort,
securityRating

from JIRA_ISSUES

INNER join SZZ_FAULT_INDUCING_COMMITS on
SZZ_FAULT_INDUCING_COMMITS.key = JIRA_ISSUES.key

INNER join GIT_COMMITS_CHANGES on
GIT_COMMITS_CHANGES.commitHash =
SZZ_FAULT_INDUCING_COMMITS.faultFixingCommitHash

INNER join SONAR_ISSUES on
SONAR_ISSUES.closeCommitHash =
SZZ_FAULT_INDUCING_COMMITS.faultFixingCommitHash

INNER join SONAR_MEASURES on
SONAR_ISSUES.closeCommitHash = SONAR_MEASURES.commitHash

where JIRA_ISSUES.resolution = 'Fixed' and
SZZ_FAULT_INDUCING_COMMITS.key is not NULL and
SONAR_ISSUES.effort is not NULL and
SONAR_ISSUES.resolution = 'FIXED' and SONAR_ISSUES.severity = 'MAJOR')

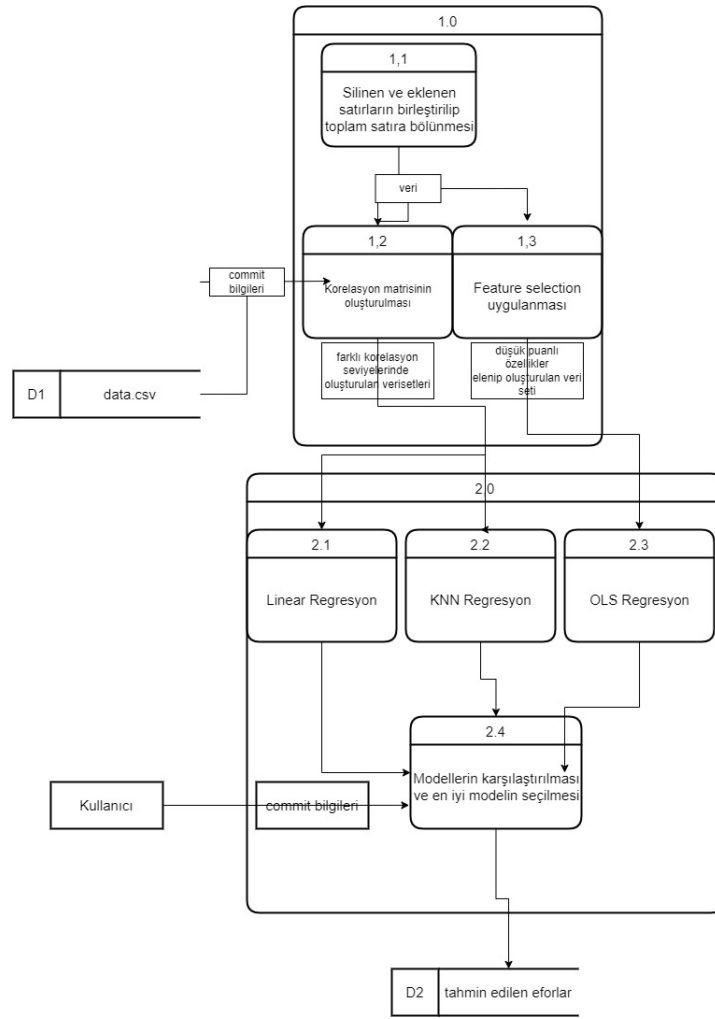
group by JIRA_KEY

```

Bu sorgu ile JIRA kayıt numaraları üzerinden gruplanmış, her bir kayıt için yapılan commitlerde toplam eklenen ve silinen satırlar, toplam efor ve commit yapılan dosyanın genel bilgileri çekilmiş ve bir .csv dosyasına aktarılarak projemizde kullanılmıştır.

Oluşturulan veri seti üzerinde korelasyon matrisi ve özellik seçimi uygulandıktan sonra farklı sayıda özellikler elenerek farklı verisetleri elde edilmiştir. Oluşturulan veri setleri farklı regresyon yöntemleri ile denenmiş ve R2 değeri en yüksek olan seçilmiştir. Sistemin başarısı regresyon modellerinde genellikle tercih edilen R2 değeri ile ölçülmüştür.

## 4.1 Veri Akış Diyagramı



Şekil 4.2 Veri Akış Diyagramı

# 5

## Sistem Tasarımı

### 5.1 Yazılım Tasarımı

Projemizde öncelikle kullanılacak olan veri seti işlendi. Veriyi işlerken öncelikle tekrar eden satırlar çıkartıldı çünkü oluşturulacak regresyon modelini kötü bir şekilde etkiler. Daha sonra her commit'ten elde edilen eklenen ve çıkartılan satırlar toplanarak yeni bir sütuna atılır. Bu aşamalar Şekil 5.5'ta gösterilmiştir.

Veri işlendikten sonraki adım olan özellik seçimi için korelasyon matrisi ve özellik seçimi yöntemleri kullanılmıştır. Korelasyon matrisi oluşturulduktan sonra farklı beş seviye belirlenmiş ve beş adet veri seti oluşturulmuştur. Bu aşamalar Şekil 5.2'de gösterilmiştir.

Daha sonraki adımda, çeşitli yöntemlerle oluşturulan veri setleri 0 ve 1 arasına normalize edilmektedir. Veri setinin sütunlarında bulunan değerler birbirleri ile uzak olduğu için verinin normalize edilmesi gerekmektedir.

Korelasyon matrisi ile oluşturulan ve daha sonra normalize edilen veri setleri üzerinde Linear Regresyon ve KNN Regresyon algoritmaları denenmiş ve efor değeri tahmininde hangisinin daha iyi çalıştığı denenmiştir. Analiz OLS Regresyon ile daha detaylı incelenmiştir.

```
# Importing Data
data = pd.read_csv(r"C:\Users\serka\Desktop\YTÜ\YTÜ 3-2\proje\data_deneme.csv")

# Removing Missing Values
data = data.dropna()

# Sum of Lines Added and Removed
sum_lines = data["sum(linesAdded)"] + KLOC_data_clean["sum(linesRemoved)"]
data["sum_lines"] = sum_lines
#sum_lines = data["sum_lines"] / data["nLoc"]
#data["sum_lines"] = sum_lines
data = data.drop(["faultFixingCommitHash", "linesAdded", "linesRemoved", "complexity.1"], axis=1)
x = data.drop("effort", axis = 1)
y = KLOC_data_clean["effort"]
```

Şekil 5.1 Import ve Önileme Aşaması

### # Corelation Matrix

```
plt.figure(figsize=(20,20))
cor = X_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
plt.show()
```

### # Creating Datasets of Different Correlation Levels

```
def correlation(dataset, threshold):
    col_corr = set() # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i] # getting the name of column
                col_corr.add(colname)
    return col_corr
```

```
corr_features = correlation(X_train, 0.3)
KLOC_data_clean_0_3_train = X_train.drop(corr_features,axis=1)
KLOC_data_clean_0_3_test = X_test.drop(corr_features,axis=1)
KLOC_data_clean_0_3_train
```

```
corr_features = correlation(X_train, 0.4)
KLOC_data_clean_0_4_train = X_train.drop(corr_features,axis=1)
KLOC_data_clean_0_4_test = X_test.drop(corr_features,axis=1)
KLOC_data_clean_0_4_train
```

```
corr_features = correlation(X_train, 0.5)
KLOC_data_clean_0_5_train = X_train.drop(corr_features,axis=1)
KLOC_data_clean_0_5_test = X_test.drop(corr_features,axis=1)
KLOC_data_clean_0_5_train
```

```
corr_features = correlation(X_train, 0.6)
KLOC_data_clean_0_6_train = X_train.drop(corr_features,axis=1)
KLOC_data_clean_0_6_test = X_test.drop(corr_features,axis=1)
KLOC_data_clean_0_6_train
```

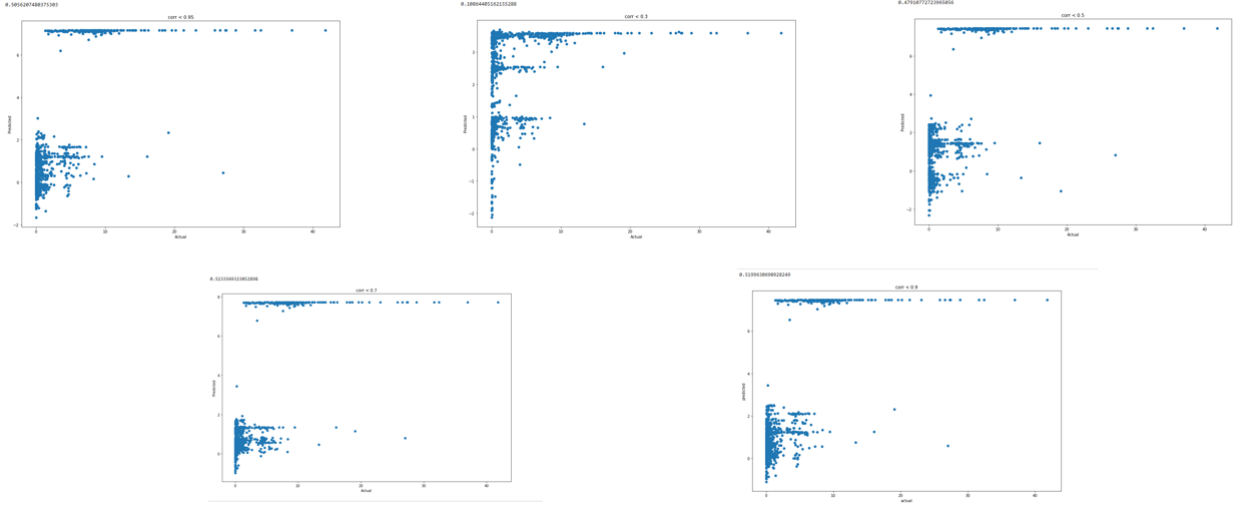
```
corr_features = correlation(X_train, 0.7)
KLOC_data_clean_0_7_train = X_train.drop(corr_features,axis=1)
KLOC_data_clean_0_7_test = X_test.drop(corr_features,axis=1)
KLOC_data_clean_0_7_train
```

Şekil 5.2 Korelasyon Matrisi ve veri setlerini oluşturma aşaması

Elde edilen Şekil 5.3'te görüldüğü üzere R2 değerleri 0.5'i geçmediği için projenin bu noktasında regresyon analizi yapmak yerine bir efor formülü üretilmeye karar verilmiştir.

Regresyon analizlerinde farklı bağımsız değişkenler, farklı korelasyon seviyelerinde oluşturulmuş veri setleri ve farklı regresyon modelleri ile yapılan denemeler sonucunda en yüksek R2 değerini bağımsız değişken, COCOMO modeli ile elde edilen efor değeri olarak kabul edildiğinde alınmıştır. Bu nedenle, veri setimizin GIT COMMITS CHANGES tablosundan alınan linesAdded, linesRemoved,nloc ve complexity değerleri ile formülümüzün ilk adımı oluşturulmuştur. COCOMO modelinden yola çıkılarak ilk adımda değiştirilen satırların döngüsel karmaşıklık değeri ile çarpılıp toplam satır sayısına bölünerek normalize edilmesi ile bir formül elde edilmiştir ve başarası COCOMO modeline göre ölçülmüştür.

Başarılı COCOMO seviyesine yükseltmek için formüle eklenecek diğer değişkeni bulmak için tekrar regresyon analizi uygulanmış ve kullanılan değişkenlerin katsayı değerleri incelenmiş, hangi özelliğin formüle katılacağı belirlenmiştir. İlk adımda oluşturduğumuz formüle ek olarak newSqaleDebtRatio özelliği ve regresyon analizinden elde edilen bir 7 katsayı çarpanı formüle eklenmiştir. Son adımda elde edilen formül Şekil 5.4'te gösterilmiştir.



**Şekil 5.3** Regresyon Yöntemleri İle Elde Edilen En İyi Sonuçlar

$$\frac{(linesAdded + linesRemoved) \times complexity \times newScaleDebtRatio \times 7}{nloc}$$

**Şekil 5.4** Elde Edilen Efor Formülü


## 5.2 Veritabanı Tasarımı

Proje için bir veri tabanı oluşturulmamıştır fakat model eğitimi için bir veri kümesi kullanılmaktadır. Projede, PyDriller, Ptidej, Refactoring Miner, SonarQube, SZZ Algorithm yöntemleri ile 33 adet JAVA ile yazılmış hata kayıtları için JIRA kullanan 3 yıldan daha eski projelerin metriklerinin toplandığı bir veri kümesi kullanılmıştır. [1] Bu metrikler kodun kalitesini ve karmaşıklığını belirler.

## 5.3 Girdi-Çıktı Tasarımı

Projemizde kullanılacak olan arayüz tasarımında kullanıcıdan oluşturulan formül için kullanılacak olan veriler alınmaktadır bu veriler; Eklenen Satır Sayısı, Silinen Satır Sayısı, Toplam Satır Sayısı, Fonksiyonel Karmaşıklık ve Yeni Ölçekli Borç Oranıdır. Alınan bilgiler bir buton ile kullanıcı tarafından onaylandıktan sonra program çalışmaktadır. Yapılan hesaplama sonucunda ekranın alt kısmında bulunan çıktı alanında hesaplanan efor tahmini yazdırılmaktadır.

Maintenance Effort Estimator



Enter the information of your issue:

Lines Added	Lines Deleted	Total Lines of Code	Complexity	New Scale Debt Ratio
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Estimated effort for this issue:

Şekil 5.5 Arayüz

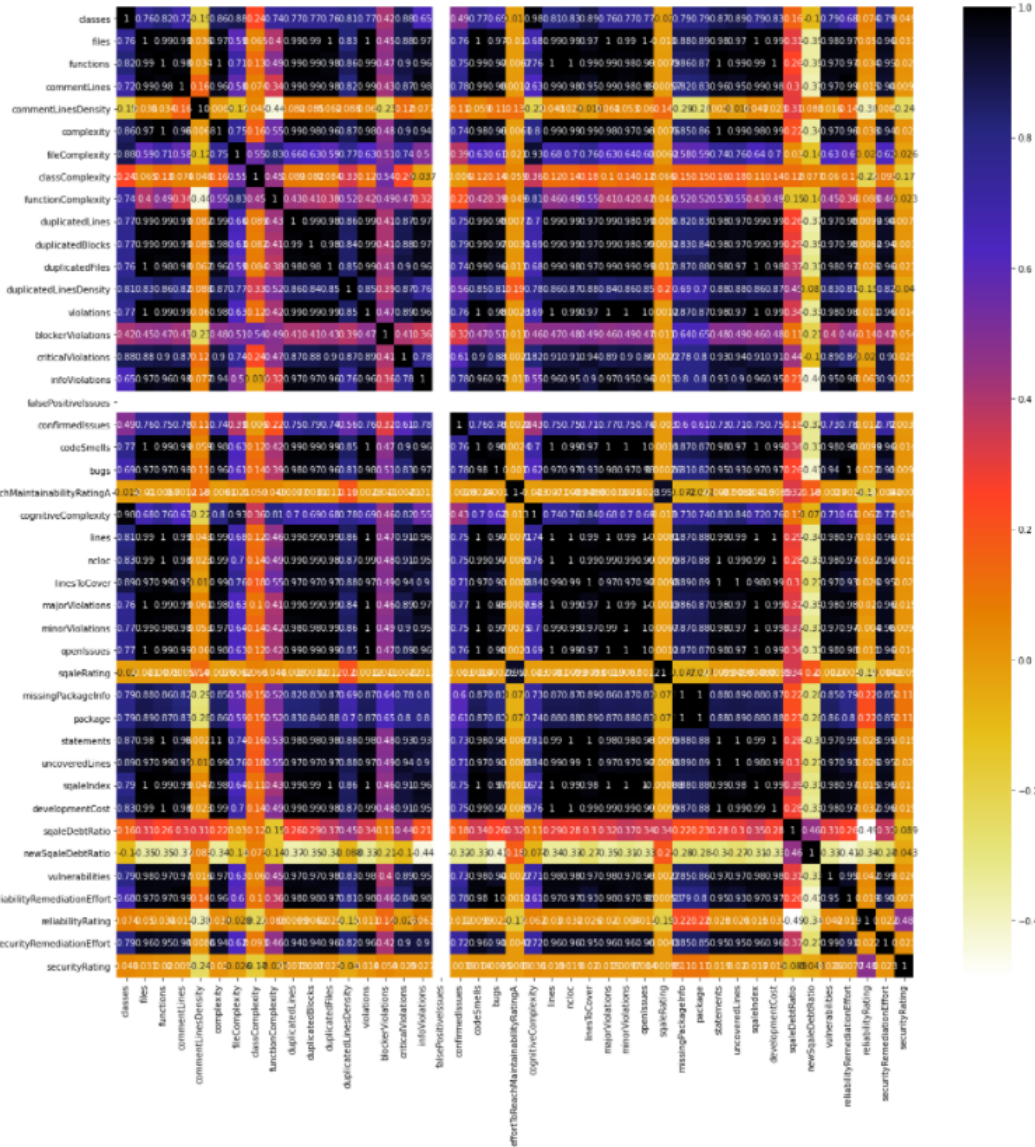


Projede bu zamana kadar uygulanan veri kümesi temizleme işlemlerinin kodu aşağıdadır.

### 6.1 Korelasyon Matrisi

Korelasyon matrisi oluşturulduktan sonra çeşitli eşik değerleri ile eleme yapılarak yeni veri kümeleri elde edilecektir. Korelasyon matrisini oluşturmak için kullanılan kod parçası aşağıdadır. Şekil 6.1 de bu kodun çıktısı görülmektedir.

```
import pandas as pd
import seaborn
import matplotlib.pyplot as plt
data = pd.read_csv(r"C:\Users\serka\Desktop\YTÜ 3-2\proje\sonuc.csv")
x = data.drop("efor", axis = 1)
y = data["efor"]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,
random_state=0)
x_train.shape, x_test.shape
import seaborn as sns
plt.figure(figsize=(20,20))
cor = x_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
plt.show()
```



Şekil 6.1 Veri kümesinin korelasyon matrisi

## 6.2 Feature Selection

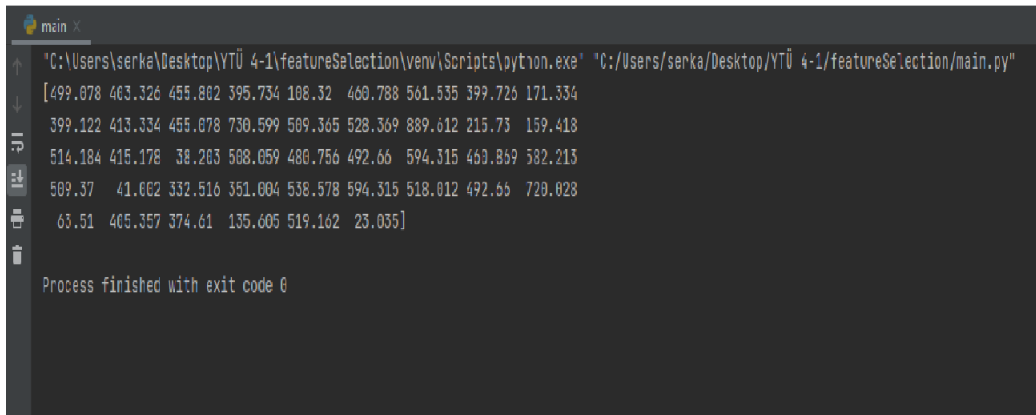
Veri kümesindeki özelliklerden hangilerinin kullanılması gerektiğinin tespiti için kullandığımız bir diğer yöntem ise feature selection yöntemidir. Feature selection için yazılan kod parçası aşağıdadır. Şekil 6.2 bu kod parçasının çıktısını göstermektedir.

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

filename = 'sonuc_yeni2.csv'
names = ['classes', 'files', 'functions', 'commentLines',
'commentLinesDensity',
'complexity', 'fileComplexity', 'classComplexity', 'functionComplexity',
'duplicatedLines', 'duplicatedBlocks', 'duplicatedFiles',
'duplicatedLinesDensity', 'violations', 'blockerViolations',
'criticalViolations',
'infoViolations', 'confirmedIssues', 'codeSmells', 'bugs',
'effortToReachMaintainabilityRatingA', 'cognitiveComplexity',
'lines', 'ncloc', 'linesToCover', 'majorViolations', 'minorViolations',
'openIssues', 'sqaleRating', 'missingPackageInfo', 'package', 'statements',
'uncoveredLines', 'sqaleIndex', 'developmentCost', 'sqaleDebtRatio',
'newSqaleDebtRatio', 'vulnerabilities', 'reliabilityRemediationEffort',
'reliabilityRating',
'securityRemediationEffort', 'securityRating', 'efor']
dataframe = read_csv(filename, names=names)

array = dataframe.values
X = array[:,0:42]
Y = array[:,42]
# feature extraction
test = SelectKBest(score_func=f_classif, k=4)
fit = test.fit(X, Y)

set_printoptions(precision=3)
print(fit.scores_)
```



```
"C:\Users\serka\Desktop\YTÜ 4-1\featureSelection\venv\Scripts\python.exe" "C:\Users\serka\Desktop\YTÜ 4-1\featureSelection\main.py"
[499.078 463.326 455.882 395.734 188.32 460.788 561.535 399.726 171.334
399.122 413.334 455.678 730.599 589.365 528.369 889.612 215.73 159.418
514.184 415.178 38.283 588.659 480.756 492.66 594.315 460.869 582.213
589.37 41.062 332.516 351.004 538.578 594.315 518.012 492.66 720.028
63.51 465.357 374.61 135.606 519.162 23.035]

Process finished with exit code 0
```

Şekil 6.2 Feature Seleciton Algoritmasından Sonra Her Özellik İçin Oluşan Değerler

### 6.3 Regresyon

Korelasyon matrisi ile oluşturulan veri setlerine ayrı ayrı regresyon algoritmaları uygulanmıştır.

```
reg = Lasso(alpha=1)
reg=LinearRegression()
model = BayesianRidge()
reg.fit(X_train,y_train)
y_pred=reg.predict(X_test)
plt.figure(figsize=(15,10))
plt.scatter(y_test,y_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('corr < 0.3')
r2_score(y_test,y_pred)
```

## 7 Deneysel Sonuçlar

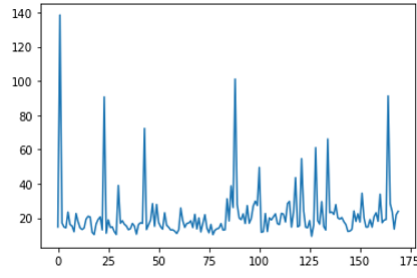
Projenin başında veri setine uygulanan regresyon analizleri incelenmiş ve örnek sayısının az olmasından kaynaklı istenilen R2 değerine ulaşamamıştır. Bu nedenle efor tahmin modeli yerine bir efor formülü elde edilmeye çalışılmıştır. İlk adımda Şekil 7.1’de görülen formül ile elde edilen değerler veri setine yeni bir özellik olarak eklenmiş ve formülü oluşturan özellikler veri setinden çıkartılmıştır.

Elde edilen yeni değerlerin gerçek efor olarak kabul ettiğimiz COCOMO formülünden elde edilen efor değerine yakın olup olmadığı Şekil 7.2’de yapılan bölüm ile incelenmiş ve sonuç olarak formülümüze yeni değerlerin eklenmesi gerektiği bulunmuştur.

Yeni özelliklerin çıkarımı için yeni oluşturduğumuz veri setine tekrar regresyon analizi uygulanmıştır. Bağımsız değişken COCOMO yönteminden elde edilen efor değeri olarak seçilirken geri kalan özellikler korelasyon matrisi ile elenerek daha sonra da katsayı değerleri incelenerek formüle eklenecek özellik seçilmiştir. Şekil 7.3’te görüldüğü üzere korelasyon matrisi uygulandıktan sonra elimizde kalan metriklerden formülümüz için en uygun olacağını düşündüğümüz Sqale Debt Ratio özelliği formülümüze eklenmiştir. Borç oranı, tüm kaynak kodunu sıfırdan geliştirmenin maliyetine kıyasla teknik borcun oranını ifade etmektedir. Bu zamana kadar oluşturulan formül ve seçtiğimiz yeni özelliğin katsayıları oranı 7 olduğu için bu sabit çarpan formülümüze eklenmiştir. Şekil 5.4’te verilen formül bütün metrikler eklendikten sonra bulunan formüldür.

$$\frac{(linesAdded + linesRemoved) \times complexity}{nloc}$$

**Şekil 7.1** Efor Formülü’nün İlk Adımı



**Şekil 7.2** Efor Formülünün İlk Halinin Gerçek Efora Oranı

	coef
newSqaleDebtRatio	113.335423
ESTIMATED_EFOR_x	16.558013
blockerViolations	4.897292
files	0.164582
classes	-0.167863
infoViolations	-1.239402
classComplexity	-75.969546
commentLinesDensity	-142.426830

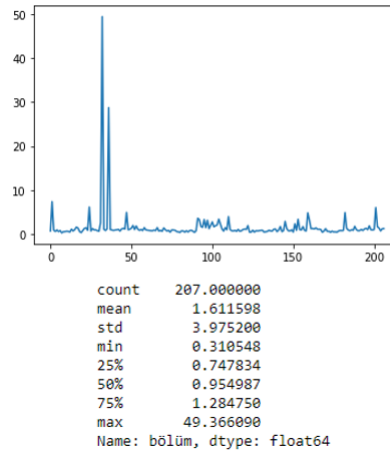
**Şekil 7.3** Regresyon Analizinden Elde Edilen Katsayılar

## 8 Performans Analizi

Elde ettiğimiz efor formülü ile bulunan efor değerlerinin doğruluğunu COCOMO ile hesaplanmış efor değerlerine bölerek 1'e ne kadar yakın oldukları incelenmiştir. Şekil 8.1'de görüldüğü üzere elde ettiğimiz son formül ile hesaplanan efor değerleri COCOMO modeli ile elde edilen efor değerlerine yaklaşmıştır. Bu iki efor değerinin birbirlerine bölümleri 1'e çok yakın değerler vermektedir.

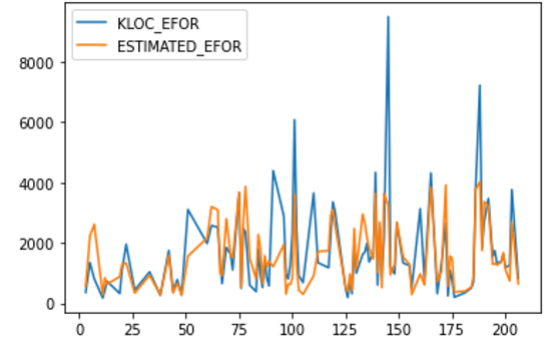
Efor değerlerinin sayısal olarak karşılaştırılması Şekil 8.2'de görüldüğü şekildedir. Formülden elde edilen efor değeri ve COCOMO modelinden elde edilen efor değeri için R2 değeri 0.92 olarak bulunmuştur.

Yapılan karşılaştırmalar sonucunda elde ettiğimiz formülün başarılı bir şekilde çalıştığı görülmektedir.



**Şekil 8.1** Efor Formülünün Son Halinin Gerçek Efora Oranı

	KLOC_EFOR	ESTIMATED_EFOR
0	7560.481845	10146.669814
1	343.789188	46.451637
2	5348.535427	5851.665950
3	361.265322	566.969159
4	163.050282	167.744119
...	...	...
202	1256.390030	740.537741
203	3764.322711	2681.279219
204	191.064932	252.308287
205	12827.834947	10365.128077
206	828.198421	647.659776



**Şekil 8.2** Elde Edilen Efor Formülünün COCOMO ile Karşılaştırılması



## 9 Sonuç

---

Bu projede yazılımların tekrar işlenmesi için efor/zaman kestirimi yapılmıştır. Efor tahmini yapılırken kullanılan veri kümesi korelasyon matrisi ile farklı özellikler içerecek şekilde bölünmüştür. Bölünen veri setleri üzerinde regresyon analizi uygulanmış ve hangi özelliklerin efor ile bağlantılı olduğu gözlemlenmeye çalışılmıştır.

Veri setimizde az sayıda örnek bulunmasından kaynaklı regresyon analizleri başarılı bir şekilde çalışmamıştır fakat bize efor hesaplaması için oluşturulacak formülün üretimi için fikir vermiştir.

Projede veri setinde bulunan her bir hata kaydı için eklenen ve çıkartılan satır sayısı toplam satır sayısına bölünerek normalize edilmiş, o dosyanın döngüsel karmaşıklığı ile çarpılarak formülün ilk aşaması oluşturulmuştur. Regresyon analizlerinden elde edilen tüm kaynak kodunu sıfırdan geliştirmenin maliyetine kıyasla teknik borcun oranını ifade eden newSqaleDebtRatio özelliğinin ve bu zamana kadar oluşturulan formülün katsayıları oranından elde edilen 7 sabit çarpanının formüle eklenmesi ile efor formülü elde edilmiştir.

Elde edilen efor formülü COCOMO modeli ile oranlanarak doğruluğu incelenmiştir. R2 değeri 0.92 olarak bulunmuştur. Elde edilen formül, efor değerini elde etmek için literatürde bulunan modellerden farklı metrikleri kullanarak ve COCOMO ile aynı sayılabilecek doğrulukta efor tahmini yapılmasını sağlar.

- [1] V. Lenarduzzi, N. Saarimäki, and D. Taibi, “The technical debt dataset,” *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, Sep. 2019. DOI: 10.1145/3345629.3345630. [Online]. Available: <http://dx.doi.org/10.1145/3345629.3345630>.
- [2] V. Nguyen, “Improved size and effort estimation models for software maintenance,” in *2010 IEEE International Conference on Software Maintenance*, 2010, pp. 1–2. DOI: 10.1109/ICSM.2010.5609554.
- [3] C. Syavasya, “Evaluation of changes on annual change traffic in calculating maintenance cost in man-months based on constructive cost model,” *www.ijcst.com*, vol. 4, Jan. 2013.
- [4] D. Tran-Cao and G. Lévesque, “1 maintenance effort and cost estimation using software functional sizes,” 2003.

### BİRİNCİ ÜYE

**İsim-Soyisim:** Serkan YILDIRIM  
**Doğum Tarihi ve Yeri:** 04.09.1999, İstanbul  
**E-mail:** 11117078@std.yildiz.edu.tr  
**Telefon:** 0531 925 01 25  
**Staj Tecrübeleri:** Softtech Şirketi RPA Developer

### İKİNCİ ÜYE

**İsim-Soyisim:** Sedat Tuna AKIN  
**Doğum Tarihi ve Yeri:** 05.01.1999, İstanbul  
**E-mail:** 11118058@std.yildiz.edu.tr  
**Telefon:** 0535 078 54 18  
**Staj Tecrübeleri:** Yapı Kredi Teknoloji Stajyer Yazılım Mühendisi

### Proje Sistem Bilgileri

**Sistem ve Yazılım:** Windows İşletim Sistemi, Python, SQL  
**Gerekli RAM:** 2GB  
**Gerekli Disk:** 10GB