BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
SAVUNMA TEKNOLOJİLERİ VE SİSTEMLERİ ANABİLİM DALI
SAVUNMA ELEKTRONİĞİ VE YAZILIMI TEZLİ
YÜKSEK LİSANS PROGRAMI

SAVUNMA SİSTEMLERİNDE TEST EFOR TAHMİNLENMESİ

HAZIRLAYAN

ESRA CIBIR

YÜKSEK LİSANS TEZİ

ANKARA - 2021

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
SAVUNMA TEKNOLOJİLERİ VE SİSTEMLERİ ANABİLİM DALI
SAVUNMA ELEKTRONİĞİ VE YAZILIMI TEZLİ YÜKSEK LİSANS PROGRAMI

SAVUNMA SİSTEMLERİNDE TEST EFOR TAHMİNLENMESİ

HAZIRLAYAN

ESRA CIBIR

YÜKSEK LİSANS TEZİ

TEZ DANIŞMANI

DR. ÖĞR. ÜYESİ TÜLİN ERÇELEBİ AYYILDIZ

ANKARA – 2021

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Savunma Teknolojileri ve Sistemleri Anabilim Dalı Savunma Elektroniği ve Yazılımı Tezli Yüksek Lisans Programı çerçevesinde Esra Cıbır tarafından hazırlanan bu çalışma, aşağıdaki jüri tarafından Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 12 /08/ 2021

Tez Adı: Savunma Sistemlerinde Test Efor Tahminlenmesi

Tez Jüri Üyeleri **İmza**

Dr. Öğr. Üyesi, Tunç AŞUROĞLU, Başkent Üniversitesi

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

Dr. Öğr. Üyesi, Damla TOPALLI, Atılım Üniversitesi

ONAY

Prof. Dr. Ömer Faruk ELALDI

Fen Bilimleri Enstitüsü Müdürü

Tarih: … / … / ……..

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: 23 / 08 / 2021

Öğrencinin Adı, Soyadı: Esra Cıbır

Öğrencinin Numarası: 21820298

Anabilim Dalı: Savunma Teknolojileri ve Sistemleri

Programı: Savunma Elektroniği ve Yazılımı Tezli Yüksek Lisans

Danışmanın Unvanı/Adı, Soyadı: Dr. Öğr. Üyesi Tülin ERÇELEBİ AYYILDIZ

Tez Başlığı: Savunma Sistemlerinde Test Efor Tahminlenmesi

Yukarıda başlığı belirtilen Yüksek Lisans tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 38 sayfalık kısmına ilişkin, 23/ 08 / 2021 tarihinde tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %4'dür. Uygulanan filtrelemeler:

1. Kaynakça hariç

2. Alıntılar hariç

3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

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

Öğrenci İmzası:

ONAY

Tarih: … / … / ……..

Dr. Öğr. Üyesi Tülin ERÇELEBİ AYYILDIZ

Bu tezi bana her zaman destek veren aileme ithaf ediyorum.

ESRA CIBIR

Ankara – 2021

TEŞEKKÜR

Başkent Üniversitesi Fen Bilimleri Enstitüsü'ne yüksek lisans tezi olarak sunmuş olduğum bu çalışma Dr. Öğr. Üyesi Tülin ERÇELEBİ AYYILDIZ danışmanlığında yürütülmüştür. Çalışma sürem boyunca, çalışmalarımı yönlendiren, araştırmalarımın bütün aşamalarında bilgi ve deneyimlerini esirgemeyerek verdiği her türlü destek ve katkılarından dolayı kendilerine teşekkürü borç biliyorum.

Hayatımın her döneminde hiçbir fedakarlıktan kaçınmayan, beni cesaretlendiren, destekleyen sevgili aileme teşekkür etmek isterim.

ÖZET

Esra CIBIR

SAVUNMA SİSTEMLERİNDE TEST EFOR TAHMİNLENMESİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Savunma Teknolojileri ve Sistemleri Anabilim Dalı

2021

Savunma projelerinde en temel sorunlardan biri projelerin büyüklüğünün ölçülmesi ve efor, maliyet tahminidir. Efor tahmini, yalnızca geliştirme faaliyetlerinin eforunu kapsamamaktadır, aynı zamanda test faaliyetlerini de kapsamaktadır. Yazılım geliştirme döngüsü için belirlenmiş efor tahmin yöntemleri olmasına rağmen, test faaliyetleri için efor tahmini genellikle kabaca tahmin edilmektedir. Bu nedenle, gerçek proje eforu, yanlış test eforu tahmini nedeniyle beklenenden daha uzun sürmektedir. Bu çalışmanın amacı, seçilen yazılım test ölçütlerini kullanarak test eforunu tahmin etmek için yeni bir yöntem önermektir. Bu seçilen yazılım test ölçütleri, önceki yazılım test efor tahmini yapan yöntemlerde göz ardı edilen ölçütlerdir. Bu amaçla, CMMI Seviye-3 sertifikalı bir savunma sanayi firmasının tamamlanmış 15 yazılım projesi incelenmiştir. Gerekli ölçütlerin seçimi için istatistiksel p-değeri kullanılmıştır. Yazılım test eforu ve yazılım test ölçütleri arasındaki ilişkiyi bulabilmek için Adımsal Doğrusal Regresyon Analizi (Stepwise Linear Regression Analysis) uygulanmıştır.Bu amaçla Minitab istatistik aracı kullanılmış ve bir yöntem önermek için Birisi Dışarıda Çapraz Doğrulama tekniği (Leave One Out Cross Validation-LOOCV) uygulanmıştır.

Deneysel çalışmanın sonuçları, önerilen metodun Pred(0,25) ve Pred(0,30) değerlerinin 0,867' ye eşit olduğunu göstermektedir. Bu değerler, önerilen yöntemin test efor tahminlemesi için kabul edilebilir sonuçlar sağladığını göstermektedir.

ANAHTAR KELİMELER: Yazılım Test Eforu, Yazılım Test Ölçütleri, Yazılım Testi, Efor Tahminleme, Çapraz Doğrulama.

ABSTRACT

Esra CIBIR

TEST EFFORT ESTIMATION FOR DEFENSE SYSTEMS

Başkent University Institute of Science and Engineering

Department of Defense Technologies and Systems

2021

One of the most fundamental problems in defense projects is measuring the size of the projects and cost and effort estimation. The effort estimation does not only covers the implementation effort but also testing activities. Although there are established effort estimation methods for software implementation cycle, effort for test phase is generally roughly estimated. Therefore the actual project effort gets longer than expected due to inaccurate test effort estimation.The aim of this study is to propose a new method for estimating testing effort using selected software testing metrics. These selected software testing metrics are the metrics which have been ignored in the previous software test effort estimation methods. For this purpose, 15 completed software projects of a CMMI Level-3 certified defense industry company have been analyzed. Statistical p-value was used for the selection of required metrics. Stepwise linear regression analysis is used to select necessary software testing metrics. For this purpose Minitab statistical tool was used and leave one out cross validation technique was applied to propose a method.

The results of the empirical study show that the proposed method provides Pred (0,25) and Pred (0,30) are equal to 0,867. We obtained plausible results and showed that these metrics are also important in software test effort estimation of the proposed method.

KEYWORDS: Software Testing Effort, Software Testing Metrics, Software Testing, Effort Estimation, Leave One Out Cross Validation.

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

Sayfa

SİMGELER VE KISALTMALAR LİSTESİ

COCOMO	The Constructive Cost Model: Yapıcı Maliyet Modeli
KLOC	Kilo Line of Code: Bin Adet Kod Satırı
LOC	Line of Code: Kod Satırı
LOOCV	Leave One Out Cross Validation: Birisi Dışarıda Çapraz Doğrulama
MARE	Mean Absolute Relative Error: Ortalama Mutlak Hata
MdMRE	Median Magnitude of Relative Error: Medyan Bağıl Hata Büyüklüğü
MMRE	Mean Magnitude of Relative Error: Ortalama Bağıl Hata Büyüklüğü
MRE	Magnitude of Relative Error: Bağıl Hata Büyüklüğü
MSE	Mean Squared Error: Ortalama Karekök Hatası
SLIM	Software Life Cycle Management: Yazılım Yaşam Döngüsü Yönetimi

1.GİRİŞ

Yazılım projelerinde önemli olan yazılımın kaliteli, düşük maliyetli ve minimum zaman içerisinde üretilmesidir [1;2]. Yazılımın kalitesinde yaşanan düşüşler müşteri memnuniyetsizliğine ve gecikmelere sebep olacaktır [3]. Yazılım süreçlerindeki gibi yazılım test süreçleri de zaman ve maliyetle ilgili problemlerle karşı karşıya kalmaktadır. Önemli olan doğru test süreçlerini tanımlamak ve proje boyunca kontrollü şekilde sürekliliği sağlayabilmektir [4]. Yazılım testi, yazılımdaki gereksinimleri yerine getirdiği için savunma şirketleri yazılım test aktiviteleri için çalışanlar arasından ekipler kurmaktadır. Yazılım test aktivitelerinde emülatörler, simülatörler, otomatik test araçları, manuel testler, test dokümanları ve bağımsız test takımları nihai ürüne ulaşılmasında katkı sağlamaktadır. Projeden sorumlu test yöneticileri de kaynaklarını planlayabilmektedir.Yazılım süreçlerine test aktivitelerini dahil edebilmek amacıyla zaman tahmini gereklidir.Hedeflere ulaşılmasına engel olan en etkili olan faktör ise yeteri kadar tahmin yapılmamış olması, yeterli verinin ve proje personelinin olmamasıdır.Savunma projelerinde test efor tahminlemesi, iç ve dış faktörlerden dolayı kolay, basite indirgenecek bir olgu olmadığı aşikardır. Tecrübeye dayalı yapılan tahminlerde yanıltıcı olma ihtimali yüksektir. Yazılım test eforunun, literatürde doğru ve etkili kullanımı ile ilgili tahmin eden teknikler olmasına rağmen tam anlamıyla keşfedilmemiş ve üzerine yeteri kadar çalışmalar yapılmamıştır. Bu nedenle yazılım test sürecindeki eforların belirlenmesi için alternatif yöntemlere ihtiyaç duyulmaktadır.

Bu çalışmanın konusu olan bu araştırmada, yazılım test sürecindeki eforun hesaplanması için yazılım test ölçütlerini kullanan metod önerilmesi amaçlanmaktadır.Yapılan çalışma ilerleyen bölümlerde anlatılacaktır.

1. Bölümde; yazılım test eforunun önemi özet olarak verilmektedir. Bölüm 2'de ; yazılım test efor tahminlemesi konusunda yapılan çalışmalar incelenmiştir.

Yazılım test eforunu tahmin etmek için birçok yöntem bulunmaktadır. Daha önce geliştirilen yöntemler 3.Bölüm'de detaylı olarak verilmiştir. Yapılan çalışmalara bakıldığında ölçütleri kullanan herhangi bir yöntem olmadığı görülmüştür.

Bölüm 4' te yapılan çalışma ve detayları yer almaktadır. Yazılım test eforu için önerilen metodda seçilen ölçütler ve bu ölçütlerin seçilme sebepleri açıklanmaktadır. Çalışmada kullanılan veri setinin detayları, kullanılan araç, yöntem ve çalışmanın bulguları yer almaktadır.

5.ve son bölümde; çalışma sonucunda çıkan bulguların neler olduğuna ve literatürdeki çalışmalar ile kıyaslanmasına değinilmektedir.

2. LİTERATÜR ÇALIŞMASI

Literatürde yazılım test eforu tahmini ile ilgili birçok çalışma bulunmaktadır. Test eforunu tahmin etmeye yarayan yöntemler genellikle geliştirme eforunu tahmin etmeye yarayan yöntemlerden türetilmiştir. Bu bölümde ağırlıklı olarak öne çıkanlar incelenmiştir.

Souza et al., [5] tarafından 2010 yılında yapılan bir çalışmada; Test Noktası Analizi Tekniğinin karmaşık, uygulanması ve yorumlanması zor olduğu belirtilmektedir. Test Noktası Analizi tekniğine bazı uyarlamalar önerilmiştir. Bu çalışmada incelenen önceki çalışmalarda, gerçek efor ile tahmin edilen efor arasında %350'ye varan farklılıklar bulunmaktadır. Önerilen teknikte gerçek efor ile tahmini efor arasındaki fark %9 ile %35 arasında değişmektedir.

Kafle [6] , tarafından 2014 yılında yaptığı çalışma Nepal'de 5 farklı şirket ve 150 makale incelenerek test efor tahmini üzerine yapmıştır. Firmaların deneysel kanıtlara ve uzman görüşlerine dayalı olarak test eforu tahmini yaptıkları görülmüştür. Test eforu tahminindeki hataların, projenin toplam efor hatası ile yakından ilişkili olduğu fark edilmiştir. Ancak bu çalışmada önerilen bir test eforu tahmin yöntemi bulunmamakta, sadece bu alanda daha fazla çalışmaya ihtiyaç olduğunu belirtmektedir.

Felipe et al., [7] 2014 yılında yaptıkları araştırmada üç farklı test efor tahmin yöntemini birbiriyle karşılaştırımıştır. Bu yöntemlerden ikisi test noktası analizi ve kullanım durumu noktalarıdır. Test noktası analizi %19 hatalı iken, kullanım senaryosu noktasında hata %142 olduğunu bildirmişlerdir. Üçüncü yöntem, test noktası analizine dayalı yapay sinir ağıdır. Sinir ağları kullanılarak iyi sonuçlar elde edilmesine rağmen, her kullanım durumu için hatanın büyük olduğu vurgulanmaktadır. Tek bir kullanım durumu için efor tahmini yapılırken yeterli olmadığı vurgulanmaktadır. Sinir ağlarının en iyi sonuçları verdiği gözlenlenmiştir, ardından test noktası analizi ve test nokta analizinden sonra kullanım noktaları geldiği görülmüştür.

Chawla et al., [8] 2017 yılında Bulanık mantık kavramını, Cocomo'nun (Yapıcı Maliyet Modeli) genişletilmesiyle tanıtmaktadır. Çalışmada 4 farklı bulanık mantık tekniği karşılaştırılmıştır. Guass Bell Membership Function (GBellMF), MMRE'nin en düşük değerine sahiptir. Bu nedenle dört farklı bulanık mantık tekniği arasında en yüksek doğruluğa sahiptir.

Clemmons [9] , 2014 yılında Kullanım durumu yöntemini üzerine çalışmıştır. Çalıştığı yöntemin sonuçlarını incelediğinde gerçek test eforunun %20'si içinde sonuç verdiği bildirmiştir.

Sharma et al., [10] 2017 yılında Neuro Fuzzy Inference System (NFIS) yöntemi ile yazılım test eforunu tahmin etme üzerine çalışmışlardır. Bulanık mantık ve yapay sinir ağlarını birleştiren bir HBRID yöntemin daha iyi olduğu vurgulanmıştır.

Mittal et al., [11] 2010 yılında Bulanık mantık yöntemi ile Cocomo modelini birleştiren bir yöntem önermişlerdir.Bulanık boyut(fuzzy size), MATLAB üzerinden ayarlamışlardır.Bulanık sayı olarak Cocomo modelinin parametrelerinden kod satır sayısı(kloc) kullanılmıştır**.** Çalışmada 5 farklı modelin Ortalama Mutlak Bağıl Hata(The Mean Absolute Relative Error) ve Pred(40) değerleri karşılaştırılmıştır. Önerilen yöntem MARE(the Mean Absolute Relative Error) 39,295 ve Pred(40) değeri 69,230 olarak en iyi sonuca sahip olduğu belirtilmiştir.

Ghafory et al., [12] 2020 yılında Slim , Slim Control, Slim Metrics, Slim Database konularını incelemişler, avantaj ve dezavantajları ifade etmişlerdir. Yazılım projelerinde maliyet eforunu tahmin etmedeki ana faktörün kod satır sayısı olduğunu belirtmişler. Maliyet tahmini alanındaki çalışmaların çoğu algoritmik tabanlı olduğunu vurgulamışlar. Slim tabanlı efor tahmininin, büyük projeler için uygun olduğu , tahmin edebilmek için bazı parametrelere ihtiyaç duyulduğu çalışmada görülmüştür.

Nageswaran [13], tarafından 2001 yılında yaptığı çalışmada Kullanım durumu tekniğini açıklamaktadır. Bu tekniğin geliştirilmesi gerektiğinden bahsedilmiştir. Kullanım durumu noktası(Use Case Point), işlev noktasından(Function Point) daha güvenilir olduğu belirtmiştir.Kullanım durumu tekniğini incelediğinde dayanıklı bir teknik gibi gözlemlenmiştir. Kullanım durumu tekniği titiz olmasa da, ad hoc tekniğe göre önemli avantajlara sahip olduğunu bildirmiştir.

Srivastava et al., [14] 2012 yılında Cuckoo arama modeli üzerinde çalışılmışlardır. Bu modelde geçmiş sonuçlar dikkate alınarak çeşitli faktörlere ağırlıklar atanmaktadır. Kullanım durumu noktası analizi kullanılmıştır. Diğer metasezgisel tekniklerden daha iyi sonuçlar elde edildiği görülmüştür.

Bu çalışmada, Bölüm 4.4. ‘de detaylı olarak anlatılan kriterlere göre seçilmiş yazılım test ölçütlerini dikkate alarak yukarıda belirtilen çalışmalardan farklı olan yeni bir test efor tahmin etme yöntemi önerilmektedir. Bu seçilen yazılım test ölçütleri, daha önce yapılan çalışmalardaki yazılım test eforu tahmin etme yöntemlerinde göz ardı edilen ölçütlerdir. Kabul

edilebilir sonuçlar elde ederek seçilen ölçütlerin yazılım test efor tahmininde de önemli olduğu görülmektedir.

3. YAZILIM TEST EFORU TAHMİN ETME YÖNTEMLERİ

Test eforunu tahmin etme teknikleri, farklı felsefelere göre bölünmüştür.

1. Uzman Görüşüne Dayalı Yöntemler,
2. Analoji ve İş Kırılım Yöntemleri,
3. Faktör ve Ağırlık Tabanlı Yöntemler
4. Yazılım Büyüklüğüne Dayalı Yöntemler
5. Bulanık Mantık ve Diğer Modellere Dayalı Yöntemler

Tahminleme yaparken aşağıdaki kriterler önem taşımaktadır.

1. Gereksinimlere müşteri bakış açısı (Customer view of requirement)
2. Fonksiyonel büyüklük (Functional size)
3. Matematiksel geçerlilik (Mathematical validity)
4. Doğrulanabilirlik (Verifiability)
5. Kıyaslama (Benchmarking)

3.1 Uzman Görüşüne Dayalı Yöntemler

Uzman görüşüne dayalı yöntemler test eforunu tahmin etmede hızlı olmasına rağmen, tahminler doğrulanamamaktadır. Uzman görüşüne dayalı yöntemler üçe ayrılır.

3.1.1 Delfi Yöntemi (Delphi)

Bireysel tahminlerin belirlenmesi uzmanların yer aldığı klasik bir tahmin etme yöntemidir. Tahminler gereksinim setlerine bağlı olarak uzmanlar tarafından yapılır. Tahmin değerleri yakın olana kadar 3-5 kez tekrarlanır. Proje yöneticisinin uzmanlar tarafından oluşturulan tahminleri incelediği ve son haliyle proje tahminlerine yansıttığı yöntemdir. Deneme yanılma yöntemi ile tahmin edildiği için başarısı yüksek değildir [15].

3.1.2 Geniş-Band Delfi Yöntemi (Wide Band Delphi)

Geniş-Band Delfi yöntemi bir karara varmak için konu uzmanlarının katıldığı yöntemdir. Koordinatör tarafından yönetilir. Konu uzmanları tarafından hızlı tahminler yapılır. Sonuçlar yakın olmalıdır ve dikkatlice uygulanmalıdır. Aşağıdaki adımlardan oluşmaktadır.

1. Koordinatör bir toplantı daveti düzenler. Konu ile ilgili uzmanlara tahmin edilecek konu hakkında bilgilendirme yapar ve tahminlerini yapabilmeleri için form dağıtır.

2. Uzmanlar dağıtılan formu tahminlerine göre doldurup, koordinatöre teslim ederler.
3. Koordinatör çıkan tahmin sonuçlarını uzmanlarla paylaşır ve yeniden toplanmayı sağlar.
4. Toplantıda tahminlerdeki farklılıklar tartışılır.
5. Tahminler yakın olana kadar, yeni tahminler uzmanlardan form dağıtılarak alınır.
6. Koordinatör sonuçları gözden geçirerek tahmin sonuçlarını nihai haline ulaştırır [16].

Bu yöntemde tahmin yapılacak ekibin kalabalık olmaması verim ve uzlaşma açısından önem taşımaktadır. Projeye benzer alan bilgisi olan uzmanların tahmin etme olasılığı daha yüksek olduğundan katılımları önemlidir. Daha hızlı ve yakın tahmin etmeyi sağlarlar. Bu yöntemde yapılan tahminler doğrulanamaz. Bu teknikte aslında efor tahmini yapmak için önemli olan gereksinimler ve fonksiyonel büyüklük ihmal edildiği için başarısı yüksek değildir.

3.1.3 Baş Parmak Kuralı (Rule of Thumb)

Uzmanlar tarafından yapılan tasarlanmış kurallara ve oranlara dayalı tahmin etme yöntemidir. İyi belgelenmiş tarihsel verilerin analizine dayanmazlar. Bağımsız olarak doğrulanması pek mümkün değildir. Test eforunu tahmin etmek için kullanılan kriterlerden olan fonksiyonel büyüklük ve kıyaslama dikkate alınmadığı için uygulanabilir değildir.

3.2 Analoji ve İş Kırılım Yöntemleri

Analoji ve İş Kırılım yöntemleri, Delfi yöntemi gibi hem yazılım eforunu tahmin etmede hem test eforunu tahmin etmede kullanılan tekniklerdir. Bu yöntem organizasyon içerisinde daha önce yapılmış benzer projeleri dikkate almaktadır. Anoloji ve İş Kırılım yöntemleri, test sürecine ve teknolojilere ince ayarlamalar yapıldığı zaman daha etkili olur.

3.2.1 Analoji Tabanlı Kestirim (Analogy-Based)

Bu yöntemin etkili kullanabilmesi için organizasyon daha önce yapmış olduğu benzer proje sayısı çok olmalıdır ve projelerdeki detayları kayıt altına almış olmalıdır. Bu bilgi kayıtları sayesinde gelecekteki projelerin başarısı ve performansı ölçülmektedir. Projeler doğrulandıktan sonra elde edilen değerler gelecekteki projelere referans olacak şekilde saklanır. Geliştirme yapılırken programlama dili, geliştirme platformu gibi faktörler dikkate alınır. Analoji tabanlı kestirim yöntemi, fonksiyonel büyüklüğü ve gereksinimleri dikkate almazken matematiksel

geçerlilik ve doğrulanabilirlik gibi test eforunu etkileyen kriterler bu yöntemde etkili olduğu görülür. Yöntemde net olmayan tahminler söz konusu olduğunda parametrelere ağırlık ataması yapılır. Ağırlık 0 olduğunda eski proje ile o parametrenin benzerlik taşımadığını, 1 olduğunda benzerlik taşıdığını 0 ile 1 arasında olduğunda ise parametrenin uygulanabilir olduğu görülür. Parametrelerin ağırlıkların ortalaması bulunarak geçmiş projelerin ağırlık faktörü ile çarpılır. Elde edilen değerler düzenlenerek, onay alınır [15].

3.2.2 Görev Tabanlı Kestirim (Task-Based)

Görev Tabanlı Kestirim yönteminde yapılacak tüm görevler listelenir. Bu yöntemde görevler Yukarıdan Aşağı veya Aşağıdan Yukarıya listelenmektedir. Her görev için en iyi, en kötü ve olası durumdaki değerleri hesaplanmaktadır. Tüm görevler için yapılan test efor tahminlerinin toplandığı yöntemdir. Tahmin edilecek efor aşağıdaki gibi Beta dağılım formülü ile hesaplanmaktadır.

$$\text{Efor} = \frac{\text{En iyi durumdaki değeri+En iyi kötü durumdaki değeri+[4 x olası durumdaki değeri]}}{6} \quad (3.1)$$

Bu yöntem benzer küçük projelere uygulanabilir. Kıyaslama yapmak bu yöntemde pek mümkün değildir [16].

3.2.3 Test Senaryosu Sayma Tabanlı Kestirim (Test Case Enumeration-Based)

Bu yöntemde, çalıştırılacak tüm test senaryoları listelenir. Hassas tahminler yapılmaya başlanır. Bu yöntemde çalıştırılacak test senaryo sayısı ve eforu toplam test eforu hakkında bilgi sağlar. Her test senaryosuna adam-saat veya adam gün olarak efor tahmini yapılmaktadır. Çalıştırılacak olan her test senaryosuna Beta formülüne göre en iyi, en kötü ve normal durumu için değer atanmaktadır.

$$\text{Beta Dağılımı} = \frac{\text{En iyi durumdaki değeri+En iyi kötü durumdaki değeri+[4 x olası durumdaki değeri]}}{6} \quad (3.2)$$

Test eforu için tahminde bulunmadan önce test senaryoları hazırlanarak belirli eforun harcanması bu yöntemin dezavantajıdır.Test senaryolarının sayılması çok zaman alıp tahmin

süresini artırmasına rağmen kapsamlı ve kesin tahminler yapılabilmektedir. Test eforunu tahmin etmeye yarayan kriterlerden kıyaslama bu yöntemde uygulanabilir değildir [16].

3.3 Faktör ve Ağırlık Tabanlı Yöntemler

Faktör ve ağırlık temelli yöntemler; Test Noktası Analizi, Kullanım Noktaları ve Test Çalıştırma Noktaları olarak üç gruba ayrılır.

3.3.1 Test Noktası Analiz (Test Point Analysis)

Bu yöntemde yalnızca kara kutu test tekniği ile gerçekleşen testlerin eforu tahmin edilir. Dinamik ve statik test noktalarından test efor hesaplaması yapılmaktadır.
Test edilecek sistem boyutu, test stratejisi ve verimlilik olarak test eforunu tahmin etmede üç temel unsurdan oluşmaktadır. Test edilecek sistem boyutu işlev noktalarınının sayısını dikkate alır. İşlev noktalarının sisteme etkisi farklı olabilmektedir [5].
Dinamik test noktaları işlev noktaları, işlevselliğe bağlı faktörler ve kalite özelliklerine bağlı olarak hesaplanır. İşlevselliğe bağlı faktörler, kullanıcı, kullanım yoğunluğu, arayüz gereksinimleri, karmaşıklık gibi daha önceden belirlenen faktörlerdir.Kalite özellikleri ise uygunluk, kullanılabilirlik, verimliliktir. Kalite özellikleri 0-6 arasında derecelendirilmektedir. Test edilecek sistem boyutu ve test stratejisi çarpılarak test hacmini vermektedir.Test hacmi de verimlilik ile çarpılarak toplam test eforu bulunur.Adam-saat olarak ifade edilir.
Test Noktası Analizi yönteminde toplama ve çarpmanın ilkeleri unutulur. Bu nedenle karmaşık bir “iyi hissetme” yöntemidir.

3.3.2 Kullanım Noktaları (Use Case Test Points)

Kullanım Noktaları yöntemi, Test Noktası Analiz yöntemine alternatif olarak türetilmiştir. Kullanım Noktaları yöntemi ile test eforu hesaplaması, test senaryoları ve işlemlere dayalı olarak kullanım noktalarının ağırlıklandırılmasına dayanmaktadır.

Test eforu hesaplama adımları aşağıdaki gibidir.

1. Sistemdeki ayarlanmamış aktör ağırlığını, aktör sayısı (Unadjusted Actor Weight-UAW) vermektedir.
2. Sistemdeki kullanım durumlarının ağırlığı (Unadjuested Use Case Weights-UUCW) ile ifade edilir.

3. Ayarlanmamış kullanım durum sayısı (Unadjuested UCP) ayarlanmamış aktör ağırlığı ve kullanım durum ağırlığının toplanması ile elde edilmektedir.

$$UCP = UAW+UUCW \quad (3.3)$$

4. Ayarlanmış kullanım durumu sayısının hesaplanması(Adjuested UCP) için teknik ve çevresel faktörler dikkate alınır. Teknik karmaşıklık faktörü TEF (Technical Complexity Factor) ile gösterilmektedir.

$$AUCP = UUCP*[0,65+(0,01*TEF)] \quad (3.4)$$

5. Ayarlanmış kullanım durumu sayısı ve dönüşüm faktörü çarpılarak tahmin edilen test eforu hesaplanır. Bu dönüşüm faktörü gerekli adam-saat sayısını göstermektedir [13].

Bu yöntem, kullanıcı gereksinimlerini dikkate alsa da kıyaslama açısından uygun değildir. Bu yönteme zayıf matematiksel işlemler uygulanması sebebiyle, süreç içerisinde bilimsel güvenirliği kaybolmuştur.

3.3.3 Test Çalıştırma Noktası (Test Execution Points)

Bu yöntem sistem boyutunu dikkate alarak test çalıştırma eforunu tahmin eden bir yöntemdir. İlk adım olarak test çalıştırma noktalarını ölçer ve ardından ölçülen noktaları kullanan bir tekniktir.Test Çalıştırma Noktası yöntemi ile test eforu hesaplama adımları aşağıdaki gibidir.

1. Test senaryolarında her test adımının karekteristik özellikleri bireysel olarak analiz edilir.
2. Düşük, orta, yüksek gibi ölçek kullanılarak derecelendirilir.
3. Her karekteristik için test çalıştırma noktalarının etki seviyesine göre nicel bir değer atanır.
4. Her karekteristik için atanan puanlar toplanır.
5. Her test adımının test çalıştırma noktaları toplanır.Toplanan değer test eforunu ve karmaşıklığını gösterir.

Test eforu ürün inşa edilmeden çok önce tahmin edildiğinden dolayı, Test Çalıştırma Noktası tekniğini yaşam döngüsünün başlarında kullanmak mümkündür [16]. Bu teknik kullanıcı gereksinimlerini dikkate alsa da kıyaslama yapmak bu teknikte pek mümkün değildir.

3.4 Yazılım Büyüklüğüne Dayalı Yöntemler

Yazılımın boyutunu dikkate alan bir yöntemlerdir. Regresyon modeli tarihsel temellere dayalı olarak inşa edilmiştir. Bu yöntemde boyut test eforunu tahmin etmeye yarayan önemli bir girdi parametresidir. Bazı teknikler test eforunu tahmin etmek için boyutu dönüştürme faktörü olarak kullanılırlar. Yazılım büyüklüğüne dayalı yöntemler kıyaslama, fonksiyonel büyüklük, matematiksel geçerlilik, doğrulanabilirlik, gereksinimlere müşteri bakış açıcı ve performans bakımından uygundur. Bu yöntemlerde daha doğru tahminler üretmeye eğilim gösterilir.

3.4.1 Test Büyüklüğü Tabanlı Kestirim (Test Size Based)

Proje başlanıldığında mevcutta olan yazılımın boyutu bilinmektedir. Test Büyüklüğü Tabanlı Kestirim yönteminde, yazılım boyutunu test boyutu olarak kabul edilir.

3.4.2 COCOMO ve SLIM

COCOMO (The Constructive Cost Model), yazılım geliştirme eforunu ölçmek için SLOC (Software lines of code) kullanır. Bu modelinin test sürecine yönelik ayarlamaları geliştirilmiştir. Basit, orta, detaylı olarak üç modeli bulunmaktadır. COCOMO geçmiş ve mevcut projeleri dikkate alan bir yöntemdir. COCOMO modelin hesaplaması kolaydır fakat yazılımın geliştirildiği ortam ve geliştiren ekibin özelliklerini dikkate almadığı için büyük projeler için önerilmez [8].

COCOMO yönteminin avantajları:

- COCOMO yöntemi şeffaftır. SLIM yöntemi gibi diğer modellerden farklı olarak nasıl çalıştığı görülür.
- Projenin toplam maliyetini hesaplamak kolaydır.
- Geçmiş veriler üzerinde çalışır ve daha doğru ayrıntılar sağlar.

COCOMO yönteminin dezavantajları:

- Projenin başlarında efor tahmini gerektiği zaman KDSI ‘yi tahmin etmek zordur.
- KDSI aslında bir büyüklük ölçüsü değil, uzunluk ölçüsüdür.

- Başarısı organizasyonun geçmiş verileri kullanarak ihtiyaçlara göre ayarlamasına bağlıdır.
- Zaman faktörüne bağlıdır.
- Müşteri becerilerini, iş birliğini ve bilgi birikimini ihmal eder.
- Tüm dokümanları ve gereksinimleri ihmal eder.

SLIM tekniğinde kod satır sayısı kullanılır. SLIM yöntemi ile belirli boyuttaki yazılımın tamamlanması için gerekli zaman ve eforun tanımlanması sağlanır. Aşağıdaki algoritmayı kullanılır.

$$K = \left(\frac{LOC}{C\ x\ t^{\frac{4}{3}}}\right) x3 \qquad (3.5)$$

K= Toplam yaşam döngüsü için gerekli efor (adam-yıl)

C= Teknoloji sabiti

t= geliştirme süresi

SLIM yönteminin avantajları:

- Hem maliyet hem de efor üzerindeki geliştirme kısıtlamalarını dikkate almak için doğrusal programlama kullanılır.
- COCOMO modeline göre daha az parametreye ihtiyaç duyar.

SLIM yönteminin dezavantajları:

- Küçük projeler için uygun değildir.
- Tahminler teknoloji faktörüne karşı aşırı hassastır.

3.4.3 Kod Satırı (Line Of Code)

Adından da anlaşılacağı gibi, bu teknikte bir projedeki toplam kod satır sayısı hesaplanmaktadır. Bilgisayar programlamada boyutu tahmin etmek için kullanılan en popüler yöntemdir. Yorumlar ve başlıklar göz ardı edilmektedir. Bu teknikte yazılım ve geliştirilmekte olan süreç ölçülmektedir. Doğrudan bir yaklaşım yöntemidir. Yazılım daha kolay tahmin yapılabilmesi için fonksiyonlarına ayrılır ve kod satırı her bir fonksiyon için hesaplanmaktadır. Daha detaylı ayrıştırma ve bölümlemeye ihtiyaç duyulur. Hem ücretsiz, hem ticari kod satır sayısını hesaplayabilen birçok araç bulunmaktadır. Projenin boyutu hakkında çeşitli ölçümler vardır. Bunlar, bin satır kod, yorumsuz kod satırları, binlerce teslim edilen kaynak komutlarıdır.

Avantajları:

- Evrensel olarak kabul edilir ve COCOMO gibi birçok modelde kullanılır.
- Tahminler yazılım geliştiricinin bakış açısına daha yakındır.
- Yorumlaması, hesaplaması, kullanımı kolaydır.

Dezavantajları:

- Farklı programlama dilleri farklı sayıda kod satırı içermektedir.
- Bu teknik için uygun bir endüstri standardı mevcut değildir.
- Projenin erken aşamalarında bu tekniği kullanarak proje boyutunu tahmin etmek pek mümkün değildir.

3.4.4 İşlev Puanı (Function Point)

Bu yöntem veri işleyen sistemler için kullanılmaktadır. Yazılım tarafından tutulan fonksiyonların sayısı ve türü işlev puanını bulmak için kullanılmaktadır. Dolaylı bir yaklaşım yöntemidir. Kod satırını genişletilebilen bir yöntemdir. İşlev puanı, kullanıcı odaklıdır ve şartnameye bağlıdır. Proje zamanını göstermek için de kullanılabilmektedir. Yapı maliyet modelinde kullanılmaktadır. Yazılım kullanıcıları için anlamlı yazılım içerisinde yer alan fonksiyonları sayısallaştıran bir yöntemdir. Hesaplama yapabilmek için beş farklı parametreye ihtiyaç duymaktadır. Bunlar dış girdi türleri, dış çıktı türleri, mantıksal dahili dosya türleri, dış sorgulama türü ve dış ara yüz dosya türleridir. Bir uygulamada son kullanıcı gereksinimlerini ölçmede kullanılmaktadır. İşlev puanı ölçütü, kod satırı yönteminin kullanılması pek mümkün olmayan ekonomi çalışmalarında da kullanılmaktadır. Kaynak kod satırlarını tahmin etmeye yarayan alternatif bir yöntemdir.

Temel adımları aşağıdaki gibidir.
- İşlev puanı sayısının türü belirlenmelidir. (İyileştirme, geliştirme, uygulama)
- Uygulamanın sınırları tanımlanmalıdır.
-Ayarlanmamış işlev puanı belirlenmelidir.
-Ayarlama faktör değerine karar verilmelidir.
-Ayarlanan işlev puanı hesaplanmalıdır.

Ayarlanan işlev puanı, ayarlanmamış işlev puanı ile ayarlama faktörünün çarpılması ile elde edilmektedir. Ayarlanmamış işlev puanı ise; veri ve işlem fonksiyonlarının toplanması ile hesaplanmaktadır.

Avantajları:

- Proje planlamanın erken aşamalarında kullanımı kolaydır.
- Programlama dilinden bağımsızdır.
- Farklı teknoloji kullanan projeleri bile karşılaştırmak mümkündür.

Dezavantajları:

- Gerçek zamanlı ve gömülü sistemler için iyi değildir.
- Birçok maliyet tahmin modeli COCOMO gibi kod satırını kullanmaktadır bu nedenle işlev puanının kod satırına dönüştürülmesi gerekmektedir.

3.5 Bulanık Mantık ve Diğer Modellere Dayalı Yöntemler

Bulanık Mantık ve diğer modellere dayalı yöntemlerin kullanılarak yazılım test eforunu tahmin etmeye yönelik çalışmaların olduğu literatürde bilinmektedir.

Bulanık mantık yaklaşımlarından biri temel olarak COCOMO modelini kullanır.Mod ve büyüklük parametrelerini girdi olarak alır. Bu model yazılım geliştirmeyi tahmin etmek için önerilmiş olsa da test efor tahmini etmek için de kabul edilir.Makine öğrenmesi tabanlı olan yapay sinir ağları, basit problemler için modellenir.Genelde yapay sinir ağları COCOMO ile beraber kullanılır. En çok tercih edilen yapay sinir ağları ileri beslemeli geri yayımlı yapay sinir ağlarıdır.Yapay sinir ağları kapalı bir kutu gibi düşünülürse üzerinde ayarlamalar yapmak gerektirir [11].

Bulanık Mantık, Yapay Sinir Ağları ve Vaka Tabanlı Akıl Yürütme gibi yenilikçi yaklaşımlar test eforunu tahmin etmek için endüstride henüz benimsenmemiştir.

4. YAPILAN ÇALIŞMA

Test eforu tahmin etmede kullanılan yöntemler incelendiğinde Test noktası analizi, yazılım test eforu tahmin teknikleri arasında en tutarlı teknik gibi görünse de, bu teknikleri geliştirmek için ölçütler kullanılarak yeni ölçümler yapılmalıdır.

4.1 Yazılım Test Ölçütleri

Yazılım ölçütleri, yapılan ölçümlerin sonuçlarına göre ölçülebilen veya hesaplanabilen yazılım değerleridir [17] . Ölçütler, ölçümlerin göstergesi ve ölçülebilen çalışmaların kontrolü ile belirlenmektedir [18]. Ölçütler süreçlerin kalitesini izlemektedir ve yönetmektedir.

Bu çalışmada yer alan ölçütler aşağıdaki gibidir.

- Senaryo Sayısı
- Test Planı Oluşturma Süresi
- Test Raporu Oluşturma Süresi
- Test Tasarlanma Süresi
- Test ve Gereksinimleri Gözden Geçirme Süresi
- Hataların İşlenme Süresi (Hata Kayıt Aracına)
- Test Ortamı Oluşturma Süresi
- Hata Çözme Süresi
- Toplantı Sayısı
- Toplantı Süresi
- Test Çalıştırma Süresi
- Hatalı Test Senaryo Sayısı

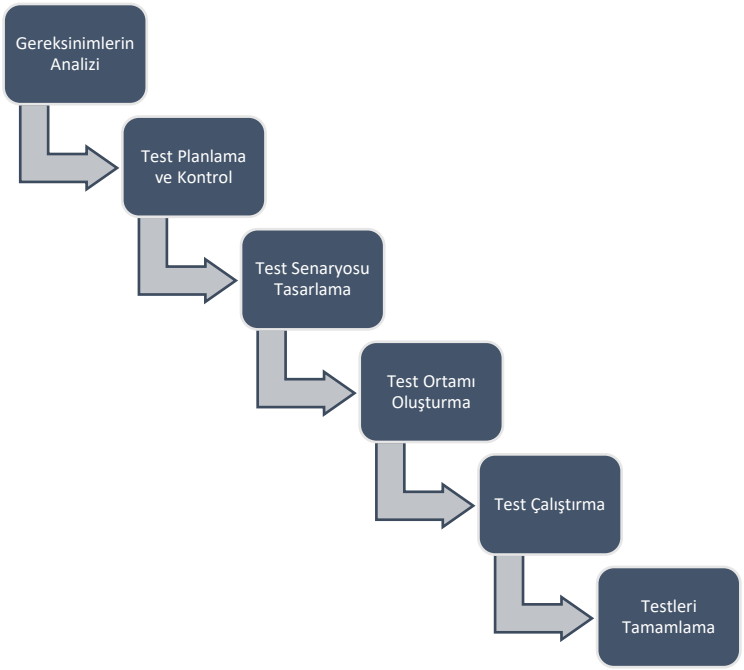
Ölçütlerin belirlenmesinde projelerdeki test süreci dikkate alınır.

Bu çalışmada incelenen projelerdeki testler kara kutu testi olarak geliştirilen yazılımın testleridir. Kara kutu testi yazılımın iç yapısı hakkında bilgi sahibi olmadan, fonksiyonelliğin kontrol edildiği testlerdir. Kara kutu testleri tamamen yazılımdaki gereksinimlere ve tanımlamalara bağlıdır. Kara kutu testlerinde kodun detayı hakkında bilgi sahibi olmaya gerek olmayan testlerdir.

Şekil 4.1. Kara Kutu Test Tekniği [19]

Kara Kutu Test Tekniğinde gereksinimler analiz edilir. Girdiler, bu girdilere karşılık gelen çıktılar belirlenir.Gereksinimlerin doğrulamasını sağlayacak testler oluşturulur. Testler çalıştırılarak girdilere karşılık gelen çıktılar kontrol edilir. Bu şekilde yazılımın doğru çalışıp çalışmadığı test edilmiş olur.

Şekil 4.2'de görüldüğü üzere önerilen test süreci 6 aşamadan oluşmaktadır.



Şekil 4.2. Yazılım Test Yaşam Döngüsü [20]

1. Gereksinimlerin Analizi

   Bu aşamada test edilecek gereksinimler belirlenir. Gereksinimlerin ne istediği anlaşılmaya çalışılır.

2. Test Planlama ve Kontrol

   Test Planlama ve Kontrol tüm testlerin tanımlandığı ve yazılım test yaşam döngünün en önemli ve verimli aşamasıdır.Bu aşamada testleri gerçekleştirmek için gerekli olan test efor tahmini test yöneticisi tarafından yapılır. Roller, görevler, girdiler ve çıktılar belirlenir.Risk analizi yapılarak kritik ve öncelikli alanlar belirlenir.

3. Test Senaryosu Tasarlama

   Test planlama aşaması tamamlandıktan sonra test senaryosu tasarlama aşaması başlar. Test ekibi tüm gereksinimleri karşılamak için gerekli tüm test senaryolarını yazdığı aşamadır. Test ekibi ayrıca gerekli test verilerini oluşturur. Test senaryoları oluşturulduktan sonra kalite ekibi tarafından gözden geçirilir.

4. Test Ortamı Oluşturma

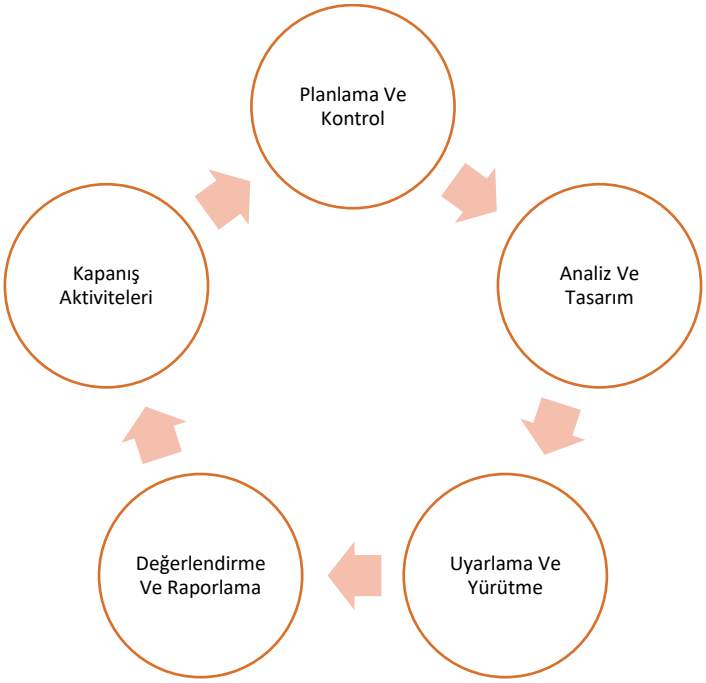
   Test ortamı oluşturma yazılım test yaşam döngüsünün önemli bir aşamasıdır.
   Yazılımın test edileceği koşullara karar verilir.Test senaryosu oluşturma ile beraber de yapılır. Bağımsız bir aktivitedir.

5. Test Çalıştırma

   Test senaryoları tasarlanıp ve test ortamı kurulduktan sonra testlerin çalıştırıldığı aşamaya geçilir. Test ekibi hazırlanan test senaryolarını adım adım çalıştırır.

6. Testleri Tamamlama(Raporlama ve Kapanış)

   Test süreci analiz edilir. Test sonuçlarının raporlandığı, kapanışın yapıldığı aşamadır.



Şekil 4.3. Projelerdeki Test Süreçleri [21]

Organizasyon içerisindeki projelerde test süreçleri Şekil 4.3'te verilmektedir. Her projede, projenin başlamasıyla birlikte gelen teknik isteklere göre belirlenir. Her teknik istek için bir veya birden fazla gereksinim yazılır. Gereksinimler, gereksinim yönetimi aracına işlenmektedir. Her gereksinimi doğrulamak için bir veya birden çok test senaryosu oluşturulur. Gereksinimlerin sayısı ve içeriği, teknik isteklerin içeriğine bağlı olarak değişmektedir.

Test eforunu hesaplamak için öncelikli olarak gereksinimlerden test senaryoları tasarlanmaktadır. Test senaryo sayısı ile her bir testin tasarlanma süresi kadar efor harcanmaktadır.

Proje başlangıcından itibaren haftalık test toplantıları yapılmaktadır.Test toplantılarında, tasarlanacak test senayoları, test ortamı oluşturma ihtiyaçları, test senaryolarını çalıştırma sonuçları, test raporu gibi konular konuşulmaktadır.

Testlere başlanmadan önce test planı oluşturulması gerekmektedir. Test planında çalıştırılacak test senaryoları , testlerin çalıştırılmasında katılacak kişiler, test senaryolarının çalıştırılmasının ne kadar süreceği, gerekli donanım ve yazılım ön koşulları, risk yönetimi, hatada izlenecek yol yönetimi gibi konular yer almaktadır.

Testlere başlamadan önce testlerin çalışmasını sağlayacak donanımsal ve yazılımsal test ortamı kurulumu oluşturulmalıdır. Gerçek sistemde kullanılacak her bir yazılım ve donanım kurulumu ayrı cihazlar üzerinde simülasyonu sağlanarak testler gerçekleştirilmelidir.Kurulumda gerekli donanım ve yazılım ayarları yapılmalıdır.

Gerekli test ortamı oluşturulduktan sonra tasarlanan test senaryolarındaki test adımları tek tek çalıştırılır. Test senaryoları sonucunda başarılı, başarısız, koşulmadı, koşulamadı şeklinde test senaryo durumları belirlenmektedir.Test senaryoları çalıştırıldığı zaman beklenen her test senaryosunun başarılı bir şekilde geçtiğinin gözlemlenmesidir.Test senaryoları her zaman başarılı şekilde çalışmaz. Test senaryosunun başarısız olmasının çeşitli nedenleri olabilir.Yazılım ve donanımsal hatalar, test senaryosunun yanlış oluşturulmuş olması, gereksinimlerin düzenlenmesi gibi birçok sebebi olabilir.

Test senaryoları koşulduktan sonra her bir senaryo için çıkan başarılı,başarısız, koşulmadı, koşulamadı durumları yazılım test raporuna yazılarak kaydedilmelidir.Hatalı test senaryoları için hata yönetim aracına bir kayıt açılarak nedeni incelenmelidir. Hatanın kök sebebi test senaryosu, yazılım hatası, donanımsal hata, gereksinim düzenlenmesi olabilir.

Hataların kök nedeni belirlendikten sonra ilgili hata çözülmeye çalışılır.Tüm hatalar çözüldükten sonra yeni bir sürüm ile tekrardan test ortamı oluşturulur. Bu işlemler hatalı ve koşulamayan test senayosu kalmayıncaya kadar tekrarlanmalıdır. Tüm gereksinimler test senaryoları çalıştırılarak doğrulandıktan sonra ürün için kabul faliyetleri gerçekleşmiş olur.

Hatasız yazılım olmadığı için hataların çözülmesi ve testlerin yapılması ürünün kalitesini dolayısıyla müşteri memnuniyetini artırır.

4.2 Projelerin Seçimi

Bu çalışmada, yazılım test eforu tahmini ile ilgili daha önce yapılmış çalışmalar incelenmiş ve veri seti olarak savunma sanayi sektöründe hizmet veren firma bünyesinde geliştirilen farklı büyüklükteki 15 projede çalışan yazılım test mühendisleri tarafından gerçekleştirilen test çalışmaları seçilmiştir.

Her projenin farklı sayıda konfigürasyon birimi vardır. Veriler Deniz, Hava ve Kara Savunma Platformları için yapılan çalışmalara aittir. Projelerde harcanan test eforları, çalışanın organizasyon içinde proje bazında yaptığı işi, aktivite bazlı olarak kaydettiği bir araç olan "Timesheet" uygulamasının veri tabanından alınmıştır.Timesheet veritabanında, eforlar adam-saat olarak kaydedilmektedir.

"Timesheet" test eforlarını kaydeden aracın veri tabanı analiz edilmiş ve seçilen projelerin gerçek test efor değerleri elde edilmiştir.

4.3 P Değeri ve Regresyon Analizi

Regresyon analizi; bağımlı bir değişken ile bağımsız değişkenler arasındaki ilişkilerin tahmininde kullanılan istatistiksel bir yöntemdir.Regresyon analizinde değişkenler arasındaki ilişki analiz edilir. Regresyon analizi doğrusal, çoklu doğrusal ve doğrusal olmayan gibi üçe ayrılır. En yaygın modeller basit doğrusal ve çoklu doğrusaldır.

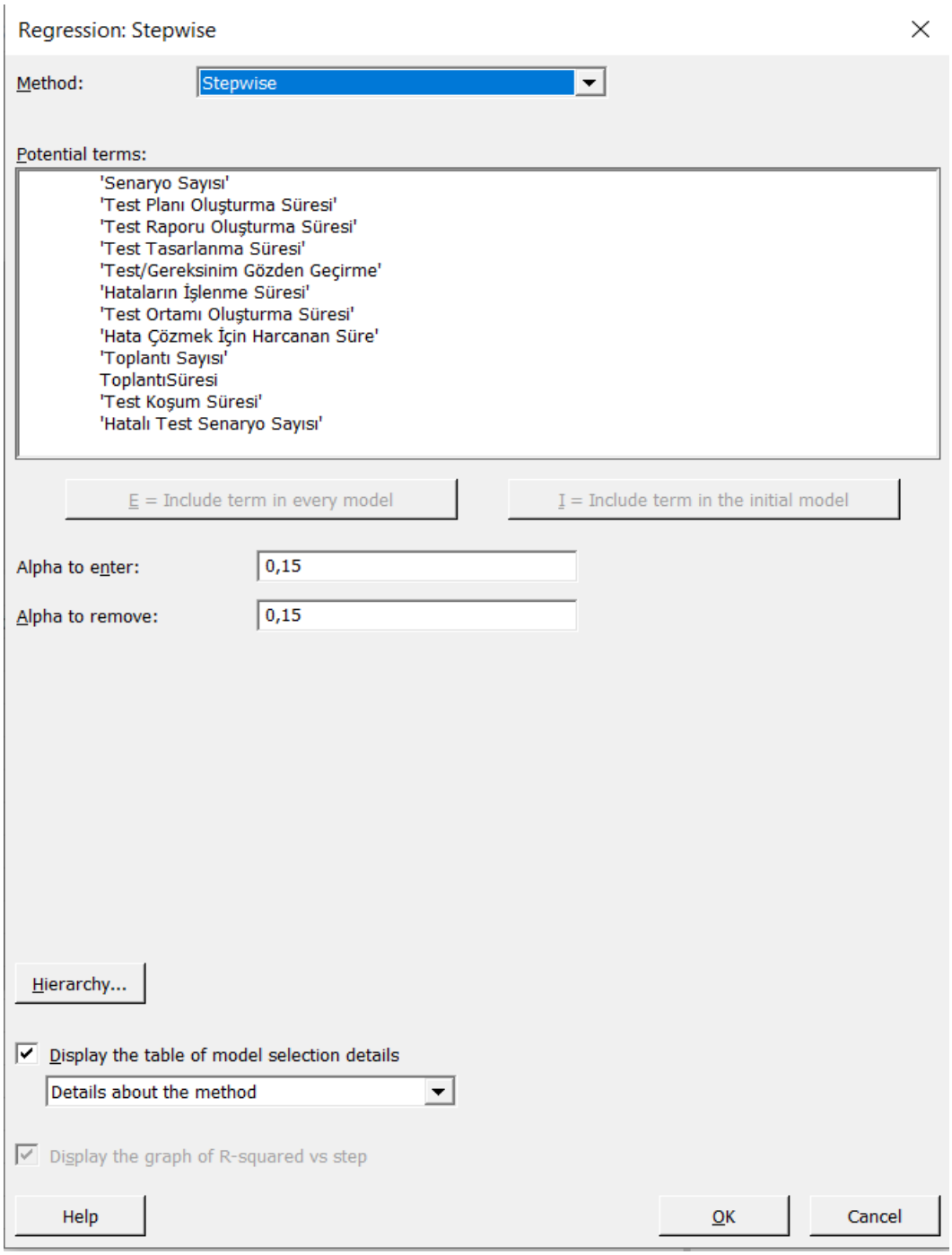
Adımsal (Stepwise) Doğrusal Regresyon Analizi, en yaygın kullanılan ölçüt seçme tekniğidir. Potansiyel ölçütlerin eklenmesi veya çıkarılmasını ve her yinelemeden sonra istatistiksel anlamlılığın test edilmesini sağlamaktadır.

Adımsal Doğrusal Regresyon Analizi, p değeri eşiğini dikkate almaktadır. Modele girmek için p değeri 0,15 değerinden küçük veya eşit olmalıdır[22]. P değeri 0,15'ten büyük olduğu durumda ilgili ölçüt modelden çıkarılmaktadır. Adımsal Doğrusal Regresyon Analizinde p değeri modele giriş ve çıkış değeri olarak kullanılmaktadır [22].

Minitab aracında, Adımsal Doğrusal Regresyon sonuçlar (Responses) bölümünde gerçek test eforu girilmektedir. Kalan ölçütler Sürekli kestirici alanına (Continuous Predictor) girilmektedir. P değeri olarak 0,15 değeri yazılır. Ölçütlerden, P değeri 0,15'e eşit veya daha küçük olanlar seçilmektedir. Her bir ölçüt eklendiğinde, ölçüt seçimine ölçütlerin p değerine bakılarak karar verilmektedir.

4.4 Seçilen Ölçütler

Adımsal Doğrusal Regresyon Analizine ilgili tüm ölçütler girilir. Analiz sonucunda, senaryo sayısı, test planı oluşturma süresi, test/gereksinim gözden geçirme süresi, test ortamı oluşturma süresi, toplantı sayısı ve hatalı test senaryosu sayısı kullanılan ölçütler olarak seçildiği görülür.



Şekil 4.4. Adımsal Doğrusal Regresyon Analiziyle Ölçüt Seçme

Minitab aracında Adımsal Doğrusal Regresyon Analizi sonucunda Tablo 4.1'de verilen ölçütlerin daha etkili olduğu belirlenmiştir.

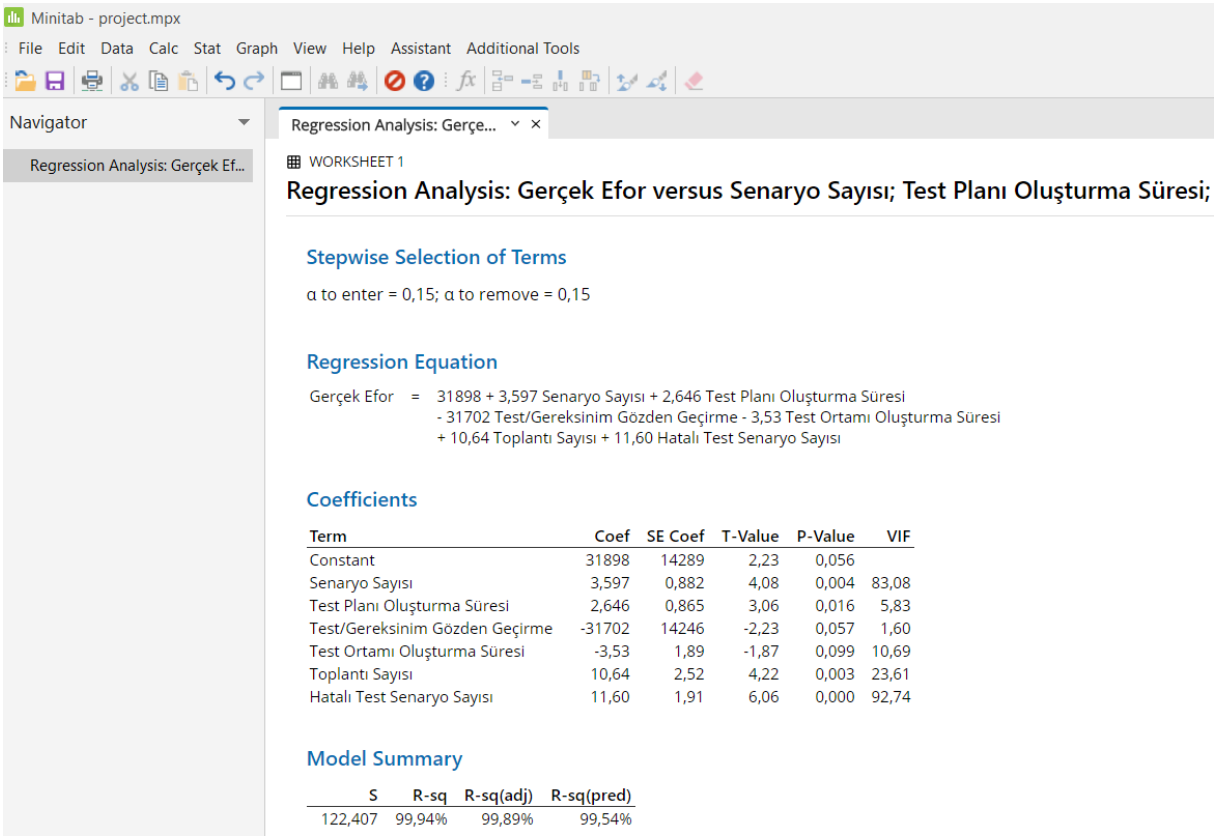
Tablo 4.1. P-Değeri <=0,15 Olan Ölçütler ve Katsayıları

Ölçütler	Katsayı	P Değeri
Senaryo Sayısı	3,597	0,004
Test Planı Oluşturma Süresi	2,646	0,016
Test/Gereksinim Gözden Geçirme Süresi	-31702	0,057
Test Ortamı Oluşturma Süresi	-3,530	0,099
Toplantı Sayısı	10,640	0,003
Hatalı Senaryo Sayısı	11,600	0,000
Sabit	31898	0,056

Adımsal Doğrusal Regresyon Analizi uygulandığında, gerçek efor için ölçütlerden bir regresyon denklemi oluşturulur. Regresyon denklemini elde etmek için seçilen ölçütlerin değerleri ve katsayıları ile çarpılır.

Elde edilen regresyon denklemi Şekil 4.5.'de de görüldüğü gibi aşağıdaki gibidir.

$$\begin{aligned}&\text{Gerçek Efor}\\&= 31898 + 3{,}597\ \text{Senaryo Sayısı} + 2{,}646\ \text{Test Planı Oluşturma Süresi}\\&- 31702\ \text{Test Gereksinim GözdenGeçirme Süresi}\\&- 3{,}53\ \text{Test Ortamı Oluşturma Süresi} + 10{,}64\ \text{Toplantı Sayısı}\\&+ 11{,}60\ \text{Hatalı Test Senaryo Sayısı}\end{aligned} \qquad (4.1)$$



Şekil 4.5. Seçilen Ölçütlerden Oluşan Regresyon Denklemi

4.5 Gerçek Test Eforları

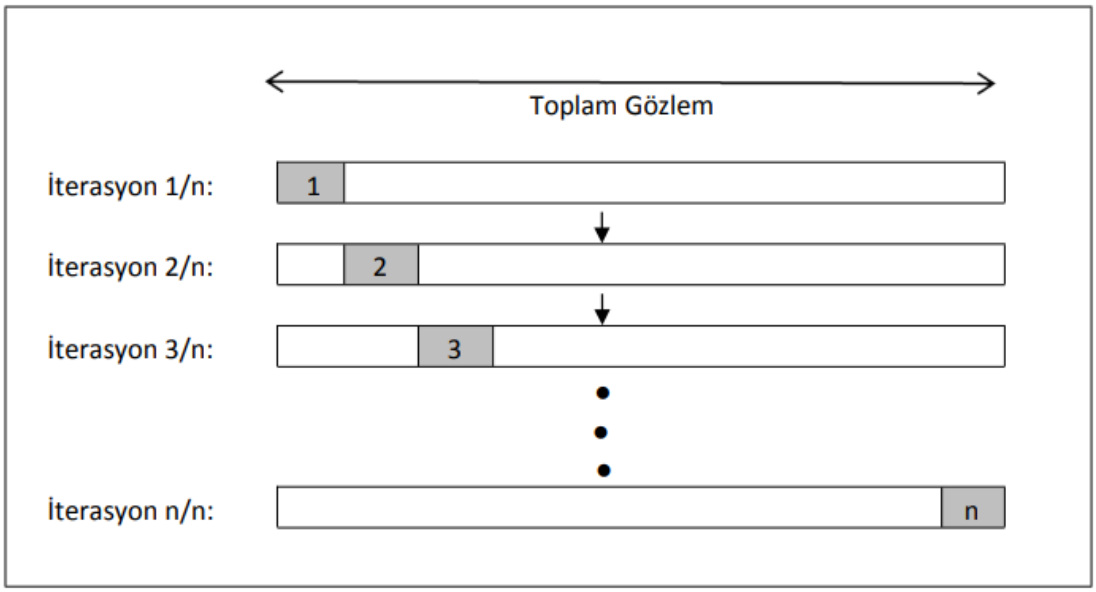
Projedeki gerçek test eforları organizasyon bünyesindeki “Timesheet” adındaki test eforlarını kaydeden araçtan alınmıştır. Projelerdeki gerçek test eforları Tablo 4.2. ‘de yer verilmiştir.

Tablo 4.2. Projelerdeki Gerçek Test Eforları

Proje İsmi	Gerçek Efor (adam-saat)
P1	2850,000
P2	250,000
P3	1966,000
P4	3750,000
P5	3854,000
P6	903,000
P7	2143,000
P8	1988,000
P9	13246,000
P10	4012,000
P11	5017,000
P12	3994,000
P13	3914,000
P14	435,000
P15	11555,000

4.6 Birisi Dışarıda Çapraz Doğrulama (Leave One Out Cross Validation-LOOCV)

Birisi Dışarıda Çapraz Doğrulama yönteminde, makine öğrenme algoritmalarının performans ölçümünde kullanılmaktadır. Modeli eğitmek için kullanılmayan veriler üzerinde tahminler yapmayı sağlamaktadır. Veri seti kaç gözlemden oluşuyor ise o kadar parçaya ayrılır ve her iterasyonda veri setinden bir gözlem çıkarılarak geriye kalan gözlemlerle eğitilmesi sağlanır [23]. İkinci gözleme geçildiği zaman ilk veri seti eklenir ikinci gözlem seti çıkarılır. Bu adımlar tüm gözlemler için tekrarlanır. Gözlem sayısı fazla olduğu zaman hesaplaması zaman alan bir yöntemdir. Veri seti büyük olduğunda maliyetli bir yöntemdir.



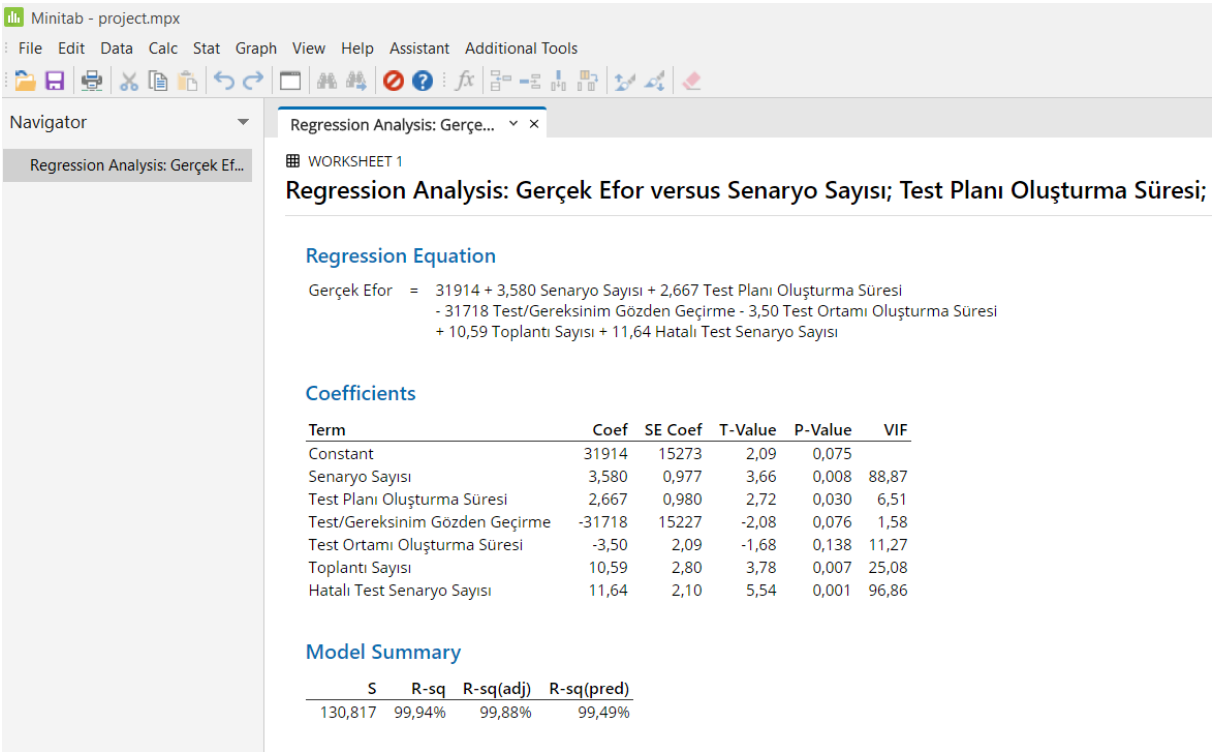
Şekil 4.6 LOOCV Görsel Örneği [24]

Birisi dışarıda çapraz doğrulama tekniğinde model tüm veri seti üzerinde eğitildiğinden dolayı test hata oranı düşüktür. Güvenilir ve tarafsız tahminler elde edilmektedir.

Tahmin edilen test eforu, "Birisi Dışarıda Çapraz Doğrulama tekniği" kullanılarak hesaplanmıştır. Adımsal Doğrusal Regresyon Analizi ile kullanılacak ölçütler seçilmiştir. Birisi Dışarıda Çapraz Doğrulama tekniğinde projelerin seçilen ölçütleri ve gerçek test eforları Minitab uygulamasına girildikten sonra ilk projenin tüm değerleri kaldırılır ve kalan projeler üzerinden regresyon modeli oluşturulur.

Oluşan regresyon modeli Şekil 4.7. Proje 1 İçin Oluşan Regresyon Denkleminde ve aşağıdaki denklemde yer verilmiştir. Regresyon modeli sabit ve seçilen ölçütlerden oluşmaktadır.

$$\begin{aligned}&\text{Gerçek Efor}\\&= 31914 + 3{,}580\ \text{Senaryo Sayısı} + 2{,}667\ \text{Test Planı Oluşturma Süresi}\\&- 31718\ \text{Test Gereksinim GözdenGeçirme Süresi}\\&- 3{,}50\ \text{Test Ortamı Oluşturma Süresi} + 10{,}59\ \text{Toplantı Sayısı}\\&+ 11{,}64\ \text{Hatalı Test Senaryo Sayısı}\end{aligned} \tag{4.2}$$

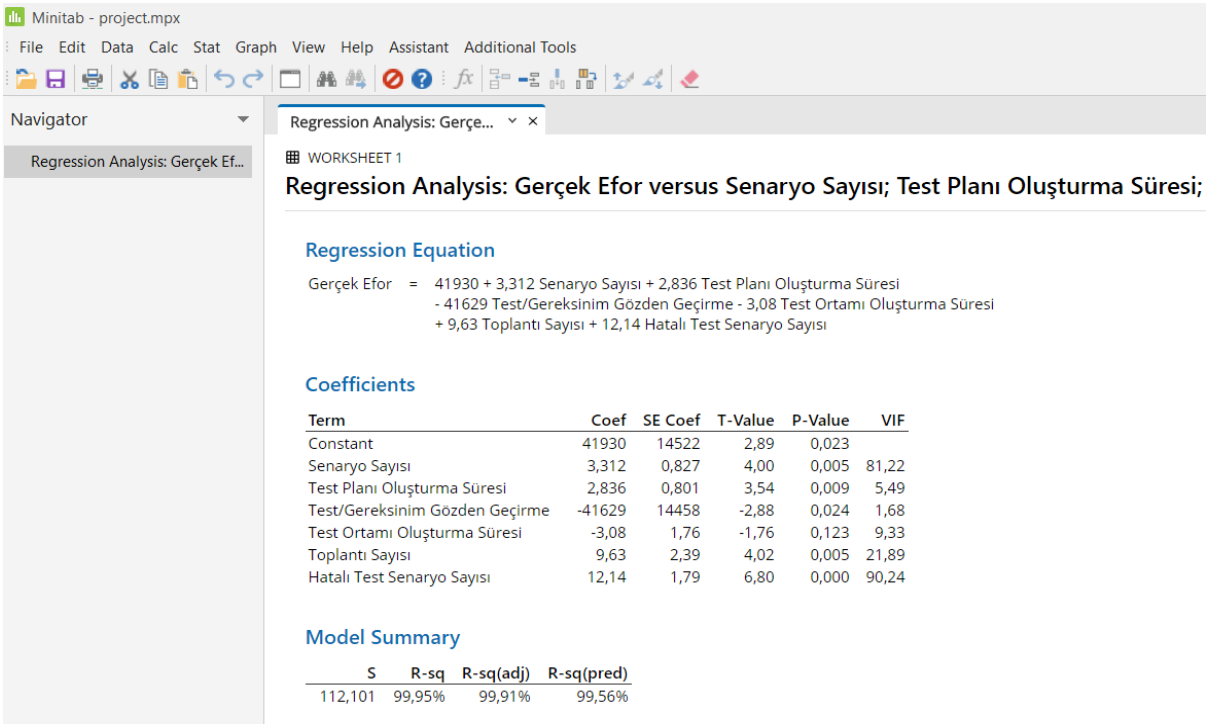


Şekil 4.7. Proje 1 İçin Oluşan Regresyon Denklemi

Oluşturulan regresyon modeline ilk projenin ölçüt değerleri girilerek tahmin edilen test eforu hesaplanır. İlk proje için tahmin edilen test eforu hesaplandıktan sonra ilk projenin tüm ölçüt değerleri Minitab aracına eklenir ve ikinci projenin tüm ölçüt değerleri çıkarılır. Yeniden regresyon modeli oluşturulur.

İkinci proje için oluşan regresyon modeli Şekil 4.8. Proje 2 İçin Oluşan Regresyon Denklemi ve aşağıdaki denklemde verilmiştir.

$$\begin{aligned}&\text{Gerçek Efor}\\&= 41930 + 3{,}312\ \text{Senaryo Sayısı} + 2{,}836\ \text{Test Planı Oluşturma Süresi}\\&- 41629\ \text{Test Gereksinim GözdenGeçirme Süresi}\\&- 3{,}08\ \text{Test Ortamı Oluşturma Süresi} + 9{,}63\ \ \text{Toplantı Sayısı}\\&+ 12{,}14\ \text{Hatalı Test Senaryo Sayısı}\end{aligned} \qquad (4.3)$$

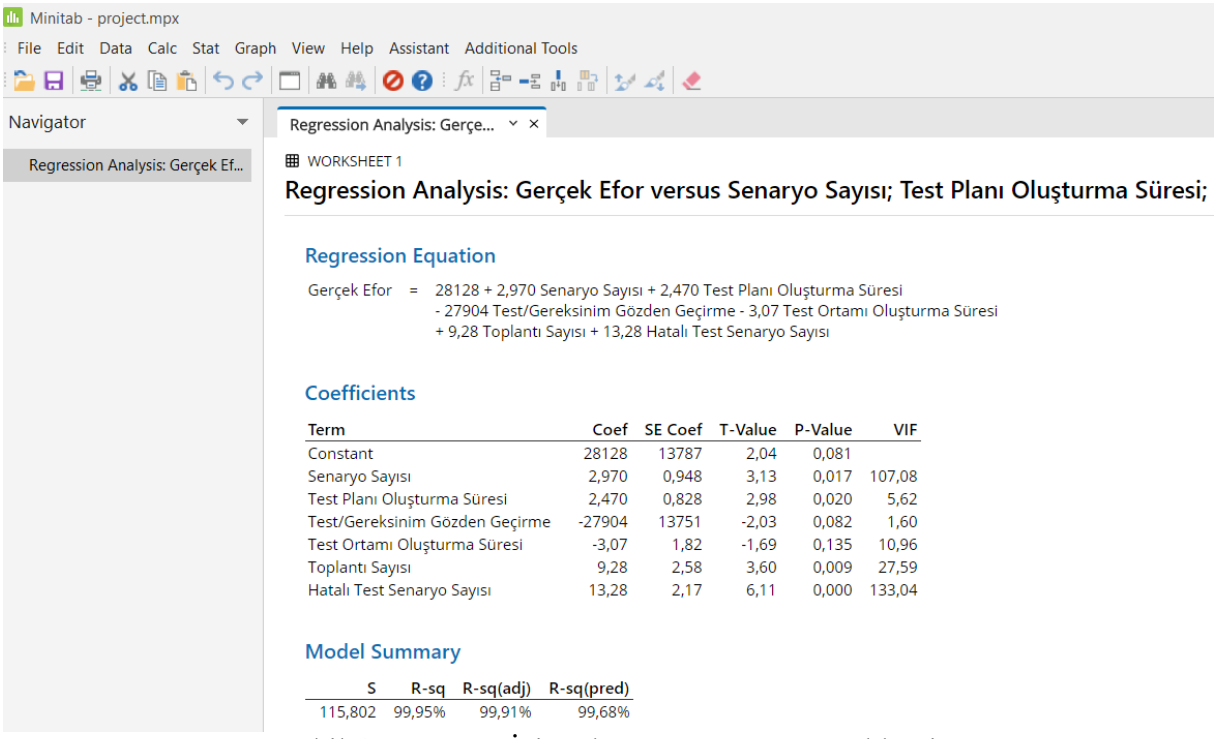


Şekil 4.8. Proje 2 İçin Oluşan Regresyon Denklemi

Oluşturulan bu yeni regresyon modeline ikinci projenin ölçüt değerleri girilerek ikinci proje için tahmin edilen test eforu hesaplanır. İkinci proje için tahmin edilen test eforu hesaplandıktan sonra ikinci projenin tüm ölçüt değerleri Minitab aracına eklenir.

Rasgele seçtiğimiz beşinci projenin tüm ölçüt değerleri silinir. Yeniden regresyon modeli oluşturulur. Beşinci proje için oluşan regresyon modeli Şekil 4.9. Proje 5 İçin Oluşan Regresyon Denklemi ve aşağıdaki denklemde verilmiştir.

$$\begin{aligned} &\text{Gerçek Efor} \\ &= 28128 + 2{,}970\ \text{Senaryo Sayısı} + 2{,}470\ \text{Test Planı Oluşturma Süresi} \\ &- 27904\ \text{Test Gereksinim GözdenGeçirme Süresi} \\ &- 3{,}07\ \text{Test Ortamı Oluşturma Süresi} + 9{,}28\ \ \text{Toplantı Sayısı} \\ &+ 13{,}28\ \text{Hatalı Test Senaryo Sayısı} \end{aligned} \qquad (4.4)$$

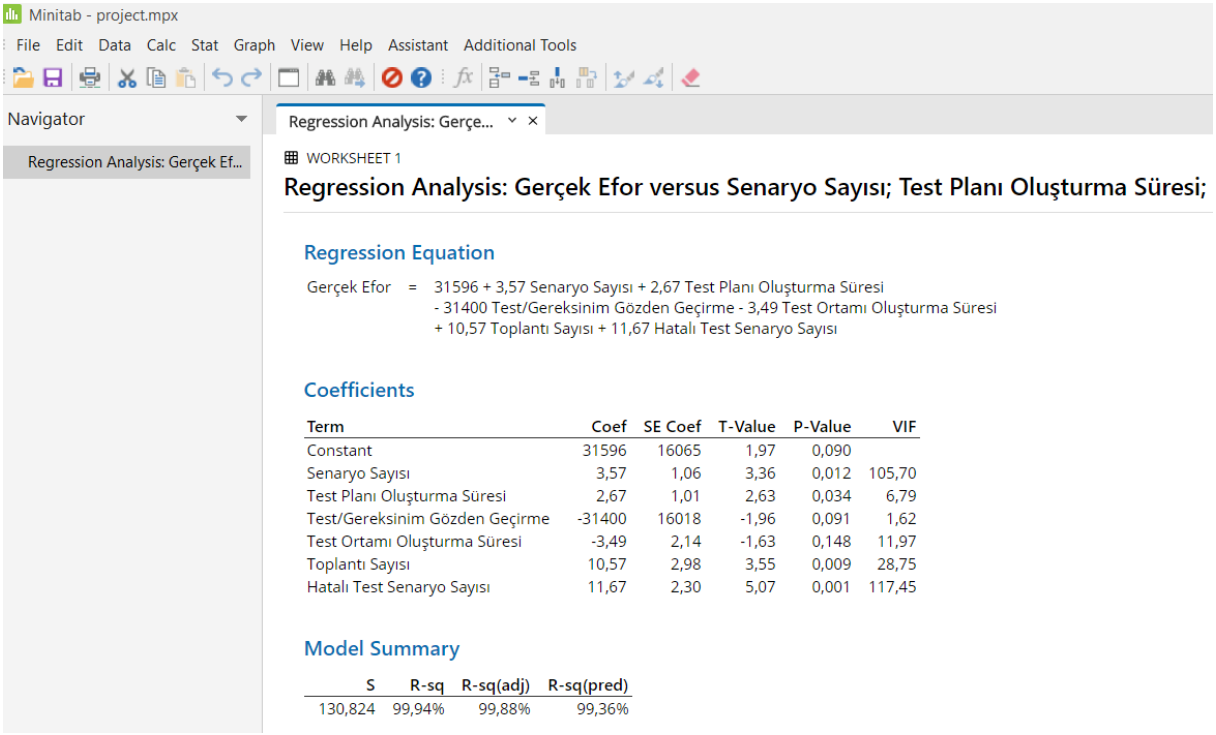


Şekil 4.9. Proje 5 İçin Oluşan Regresyon Denklemi

Oluşturulan bu yeni regresyon modeline beşinci projenin ölçüt değerleri girilerek beşinci proje için tahmin edilen test eforu hesaplanır. Beşinci proje için tahmin edilen test eforu hesaplandıktan sonra beşinci projenin tüm ölçüt değerleri Minitab aracına eklenir.

Rasgele seçtiğimiz onuncu projenin tüm ölçüt değerleri silinir. Yeniden regresyon modeli oluşturulur. Onuncu proje için oluşan regresyon modeli Şekil 4.10. Proje 10 İçin Oluşan Regresyon Denklemi ve aşağıdaki denklemde verilmiştir.

$$\begin{aligned}&\text{Gerçek Efor}\\&= 31596 + 3{,}57\ \text{Senaryo Sayısı} + 2{,}67\ \text{Test Planı Oluşturma Süresi}\\&- 31400\ \text{Test Gereksinim GözdenGeçirme Süresi}\\&- 3{,}49\ \text{Test Ortamı Oluşturma Süresi} + 10{,}57\ \ \text{Toplantı Sayısı}\\&+ 11{,}67\ \text{Hatalı Test Senaryo Sayısı}\end{aligned} \qquad (4.5)$$

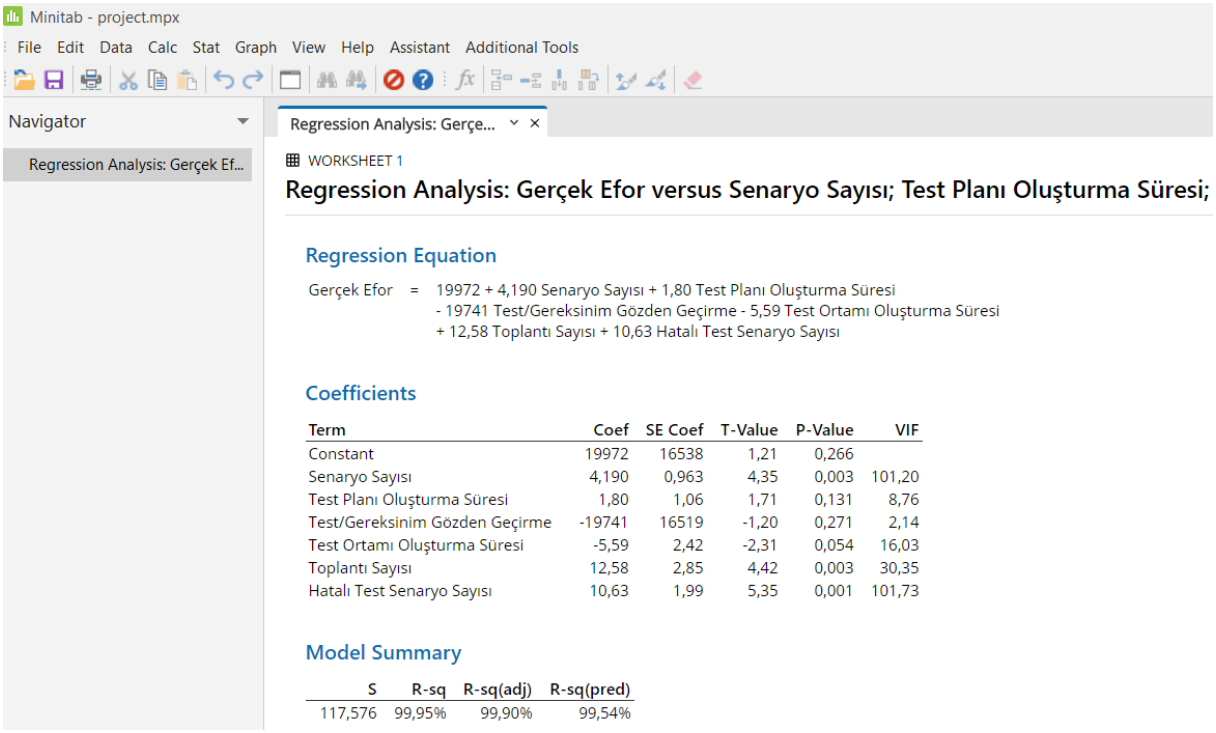


Şekil 4.10. Proje 10 İçin Oluşan Regresyon Denklemi

Oluşturulan bu yeni regresyon modeline onuncu projenin ölçüt değerleri girilerek onuncu proje için tahmin edilen test eforu hesaplanır. Onuncu proje için tahmin edilen test eforu hesaplandıktan sonra onuncu projenin tüm ölçüt değerleri Minitab aracına eklenir.

Rasgele seçtiğimiz on dördüncü projenin tüm ölçüt değerleri silinir. Yeniden regresyon modeli oluşturulur. On dördüncü proje için oluşan regresyon modeli Şekil 4.11. Proje 14 İçin Oluşan Regresyon Denklemi ve aşağıdaki denklemde verilmiştir.

$$\begin{aligned} &\text{Gerçek Efor} \\ &= 19972 + 4{,}190\ \text{Senaryo Sayısı} + 1{,}80\ \text{Test Planı Oluşturma Süresi} \\ &- 19741\ \text{Test Gereksinim GözdenGeçirme Süresi} \\ &- 5{,}59\ \text{Test Ortamı Oluşturma Süresi} + 12{,}58\ \ \text{Toplantı Sayısı} \\ &+ 10{,}63\ \text{Hatalı Test Senaryo Sayısı} \end{aligned} \qquad (4.6)$$



Şekil 4.11. Proje 14 İçin Oluşan Regresyon Denklemi

Oluşturulan bu yeni regresyon modeline on dördüncü projenin ölçüt değerleri girilerek on dördüncü proje için tahmin edilen test eforu hesaplanır. On dördüncü proje için tahmin edilen test eforu hesaplandıktan sonra onuncu projenin tüm ölçüt değerleri Minitab aracına eklenir.

Uygulanan süreçler sonucunda 15 adet gerçek savunma projesi için tahmin edilen test efor sonuçları hesaplanmış olur. Projelerdeki tahmin edilen test eforları Tablo 4.3 de yer verilmiştir.

Tablo 4.3. Projelerdeki Tahmin Edilen Test Eforları

Proje İsmi	Tahmin Edilen Test Eforu (adam-saat)
P1	2859,574
P2	484,849
P3	1796,658
P4	3498,012
P5	3625,982
P6	887,586
P7	2115,796
P8	2262,523
P9	13736,410
P10	4024,100
P11	5284,048
P12	3971,659
P13	3773,201
P14	670,912
P15	11076,230

4.7 Belirtme Katsayısı (Coefficient of Determination)

Lineer regresyon denklemlerindeki R-kare (Belirtme Katsayısı) değeri, araştırmacıları, seçtiğimiz bağımsız değişkenlerin, bağımlı değişkenin varyans değerini ne kadar açıkladığı konusunda bilgilendirir. R kare istatistiksel olarak ölçüm sağlamaktadır. Modelin performansını ölçmekte de kullanılır. R kare değeri [0, 1] aralığında yer alır [25].

R kare değerinin sıfır olması bağımsız değişkenlerin bağımlı değişkenleri açıklayamadığını gösterirken bir olması modelin tam olarak açıklandığını göstermektedir. R kare değerinin yüksek olması modelin açıklayıcı olduğunu ifade eder. Lineer regresyon modelinin yorumlanmasını sağlar. R kare katsayısı bazı özel durumlarda negatif çıkmaktadır. R kare değeri model karşılaştırmada da kullanılır.

Projelerdeki R kare değerleri Tablo 4.4. ‘de yer verilmiştir.

Tablo 4.4. Projelerdeki R Kare Değerleri

Proje İsmi	R Kare (%)
P1	99,940
P2	99,950
P3	99,950
P4	99,960
P5	99,950
P6	99,930
P7	99,950
P8	99,950
P9	99,930
P10	99,940
P11	99,950
P12	99,940
P13	99,940
P14	99,950
P15	99,930

R kare değeri 0,5 değerine eşit veya büyük olduğu durumlar kabul edilir [26]. Projelerin "R kare" değerleri %99 'un üzerinde olduğu için başarılı oldukları görülür.

4.8 Bağıl Hata ve Kestirim Doğruluğu (Magnitude of Relative Error- MRE- Prediction Accuracy)

Yapılan çalışmada 15 proje için Adımsal Doğrusal Regresyon Analizinde çıkan denklem eşitliğinde ölçüt değerleri yerine konulduğunda tüm projelerden tahmini test eforları bulunmuştur. Gerçek test eforları ise “Timesheet” aracından alınmıştır.

Bağıl Hatanın Büyüklüğü (MRE), tahmin modellerini değerlendirmek için en yaygın kullanılan ölçüdür. Pred(l) düzeyi tahmin, literatürde sıklıkla kullanılan ve belirli bir doğruluk düzeyi için yapılan gözlemlerin oranıdır.

Ortalama Bağıl Hata Büyüklüğü (MMRE), Bağıl hata büyüklüklerin toplamının proje sayısına bölünmesiyle elde edilir.

Medyan Bağıl Hata Büyüklüğü, bağıl hata büyüklükleri sıralandığında ortadaki değeri baz alır. Bağıl hata (MRE), MMRE, MdMRE, ve Pred(x) ve eşitlikleri Eş. 4.7, Eş.4.8, Eş. 4.9 ve Eş. 4.10. formülleriyle birlikte verilmektedir.

MRE: Magnitude of Relative Error : Bağıl Hata Büyüklüğü

MMRE: Mean Magnitude of Relative Error: Ortalama Bağıl Hata Büyüklüğü

MdMRE Median Magnitude of Relative Error : Medyan Bağıl Hata Büyüklüğü

$$\text{MRE} = |\text{Egerçek} - \text{Etahmin}| \,/\, \text{Egerçek} \tag{4.7}$$

$$MMRE = \frac{1}{n} + \sum_{i=1}^{n}(MREi) \tag{4.8}$$

$$MdMRE = \text{Median}\ (MREi) \tag{4.9}$$

$$\text{Pred(l)} = k/N \tag{4.10}$$

Bağıl hata değerleri (MRE); gerçek test eforundan tahmin edilen test eforunun çıkarılması ve tekrar gerçek test eforuna bölünmesiyle elde edilir.

"k", MRE'si "l"den küçük olan projelerin sayısını belirtir. Eş. 4.10'da "N" sayısı, veri setindeki toplam proje sayısını temsil etmektedir. Pred değerleri hesaplanırken "l" değeri 0,25 ve 0,30 olarak alınmıştır çünkü bu değerler tahmin modelleri oluşturulurken en sık kullanılan değerlerdir [27].

Kabul edilen Bağıl Hata değerinin (MRE) Conte et al.,[27] tarafından (0,25) için 0,75'e değerine eşit veya daha büyük olması gerektiğini belirtilmektedir. Genel olarak bir tahmin modelinin doğruluğunun Pred(0,25) ile orantılı olduğundan bahsedilmektedir.

Pred (0,30) değerinin ise Tate ve Verner [28], tarafından 0,70' e eşit veya daha büyük olması gerektiği belirtilmiştir.

Hesaplanan Bağıl Hata Değerleri Tablo 4.5. 'de verilmiştir.

Tablo 4.5. Projelerin Hesaplanan Bağıl Hata Değerleri

Proje İsmi	MRE
P1	0,003
P2	0,939
P3	0,086
P4	0,067
P5	0,059
P6	0,017
P7	0,013
P8	0,138
P9	0,037
P10	0,003
P11	0,053
P12	0,006
P13	0,036
P14	0,542
P15	0,041
Pred (0,25) =0,867	
Pred(0,30) =0,867	
MMRE = 0,136	
MdMRE =0,041	

Conte et al., [27]' nin de belirttiği gibi tahmin başarısının iyi olduğu; Pred(0,25) değeri için elde edilen değerin 0,867 geldiği ve bu değerin 0,75'den büyük olduğu, Tate ve Verner, [28]'in belirttiği gibi elde edilen değerin Pred(0,30) için 0,867 geldiği ve bu değerin de 0,70'den büyük olduğu görülmektedir. Verilen referanslardan yola çıkılarak efor tahmininin başarılı olduğu görülür.

Hastings ve Sajeev [29] tarafından MMRE ve MdMRE değerlerinin 0,20 ile 0,50 arasında olması kabul edilebilir olarak değerlendirilir. Bu nedenle Tablo 4.5 'e göre kabul edilebilir MMRE ve MdMRE sonuçları elde edildiği görülmektedir.

4.9 Ortalama Karekök Hatası (Mean Square Error)

Ortalama karekök hatası regresyon eğrisinin bir dizi noktaya ne kadar yakın olduğunu belirtmektedir [25] . Ortalama karesi alınmış hata, tahmini test eforu değeri ile gerçek test eforu değeri arasındaki ortalama kare kök farkını ölçer. Gerçek efor ile tahmini efor arasındaki fark küçük olduğunda ortalama karekök hatası değeri daha düşüktür. Karesi alındığından dolayı her zaman pozitif değer almaktadır. Ortalama Karekök Hatası denklemi Eş. 4.11 ‘de verilmektedir

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Yi - \widehat{Yı})^2 \quad (4.11)$$

MSE: Mean Squared Error (Ortalama Karekök Hatası)

n: Proje Sayısı

Yi: Tahmin Edilen Test Eforu

$\widehat{Yı}$: Gerçek Test eforu

Tablo 4.6. Projelerde Hesaplanan Ortalama Karekök Hataları

Proje İsmi	MSE
P1	91,652
P2	55154,053
P3	28676,713
P4	63497,952
P5	51992,208
P6	237,591
P7	740,058
P8	75362,880
P9	240501,000
P10	146,410
P11	71314,634
P12	499,120
P13	19824,358
P14	55654,472
P15	229222,628
MSE Toplamı=892915,715	
MSE Hatası=59527,714	

Ortalama karekök hata değerleri sıfıra yakın olduğunda yapılan tahminlerin daha iyi performanslı olduğu söylenmektedir.Ortalama karekök hata değerleri küçük olduğunda daha iyi olduğu için P1 ‘deki ortalama karekök hatası en iyi iken P9 ‘daki ortalama karekök hatası en kötüsüdür.

4.10 Tartışma

İlgili çalışmalara bakıldığında uzman görüşüne dayalı yöntemlerin yanılma payının yüksek ve yetersiz olduğu görülmektedir**.**

Souza et al., [5] tarafından 2010 yılında yapılan bir çalışmada; Test Noktası Analizi Tekniğinin gerçek efor ile tahmin edilen efor arasında %350'ye varan farklılıklar bulunmaktadır. Önerdiği teknikte gerçek efor ile tahmini efor arasındaki fark %9 ile %35 arasında değişmektedir.

Kafle [6], tarafından 2014 yılında yaptığı çalışmada test eforu tahminindeki hataların, projenin toplam efor hatası ile yakından ilişkili olduğu fark edilmiştir. Ancak bu çalışmada önerilen bir test eforu tahmin yöntemi bulunmamakta, sadece bu alanda daha fazla çalışmaya ihtiyaç olduğunu belirtilmiştir.

Felipe et al., [7] 2014 yılında yaptıkları araştırmada üç farklı test efor tahmin yöntemini birbiriyle karşılaştırımıştır. Yaptıkları çalışmada test noktası analizi %19 hatalı iken, kullanım senaryosu noktasında hata %142 olduğunu bildirmişlerdir.Üçüncü yöntem olan yapay sinir ağlarının daha iyi olduğunu vurgulamalarına rağmen yapay sinir ağları literatürde tam benimsenmiş bir teknik değildir.

Chawla et al., [8] 2017 yılında yaptığı çalışmada 4 farklı bulanık mantık tekniği karşılaştırılmıştır. Guass Bell Membership Function (GBellMF), MMRE'nin en düşük değerine sahiptir. Bu nedenle dört farklı bulanık mantık tekniği arasında en yüksek doğruluğa sahiptir. Bulanık mantık tekniği yapı itibariyle savunma projelerine uygun değildir.

Clemmons [9], 2014 yılında Kullanım durumu yöntemini üzerine çalışmıştır. Çalıştığı yöntemin sonuçlarını incelediğinde gerçek test eforunun %20'si içinde sonuç verdiği bildirmiştir.

Sharma et al., [10] 2017 yılında Neuro Fuzzy Inference System (NFIS) yöntemi ile yazılım test eforunu tahmin etme üzerine çalışmışlardır. Bulanık mantık ve yapay sinir ağlarını birleştiren bir HBRID yöntemin daha iyi olduğu vurgulanmıştır.

Mittal et al., [11] 2010 yılında Bulanık mantık yöntemi ile cocomo modelini birleştiren bir yöntem önermişlerdir**.** Çalışmada 5 farklı modelin Ortalama Mutlak Bağıl Hata(The Mean Absolute Relative Error) ve Pred(40) değerleri karşılaştırılmıştır. Önerilen yöntem MARE(the Mean Absolute Relative Error) 39,295 ve Pred(40) değeri 69,230 olarak en iyi sonuca sahip olduğu belirtilmiştir.

Ghafory et al., [12] 2020 yılında Slim , Slim Control, Slim Metrics, Slim Database konularını incelemişler, avantaj ve dezavantajları ifade etmişlerdir. Yazılım projelerinde maliyet eforunu tahmin etmedeki ana faktörün kod satırları olduğu belirtilmiştir. Kod satır sayısı yazılım geliştirildikten sonra belli olduğundan dolayı efor tahminlemesi için yeterli ve doğru değildir. Yazılım test efor tahmininin proje başlandığında zaman yapılması gereklidir.

Nageswaran [13], tarafından 2001 yılında yaptığı çalışmada Kullanım durumu tekniğini açıklamaktadır. Bu tekniğin geliştirilmesi gerektiğinden bahsedilmiştir.

Srivastava et al., [14] 2012 yılında Cuckoo arama modeli üzerinde çalışılmışlardır. Bu modelde geçmiş sonuçlar dikkate alınarak çeşitli faktörlere ağırlıklar atanmaktadır. Kullanım durumu noktası analizi kullanılmıştır. Diğer metasezgisel tekniklerden daha iyi sonuçlar elde edilmesine rağmen sonuçlar yeterli değildir.

Yapılan çalışmalar genellikle geliştirme efor tahminlemesi üzerinedir. Yazılım test ölçütleri kullanılmamıştır. Yapılan bu çalışmada ise 15 proje için yazılım test ölçütleri kullanarak test efor tahmininin önemli olduğu vurgulanmıştır. Elde edilen sonuçlar literatürde yapılan çalışmaların sonuçlarından daha yüksektir.

5. SONUÇ VE ÖNERİLER

Bu çalışmada, seçilen yazılım test ölçütleri kullanarak test eforunu tahmin etmek için CMMI-3 sertifikalı bir savunma sanayi şirketinin 15 tamamlanmış savunma projesi incelenmiştir. Projelerdeki gerçek test eforları şirket içerisindeki "Timesheet" aracından alınmıştır. Tahmin edilen test eforlarını hesaplamak için Minitab aracından yararlanılmıştır. Gerekli yazılım test ölçütlerini seçmek için Adımsal Doğrusal Regresyon Analizi kullanılmıştır. P-değeri 0.15'ten küçük veya buna eşit olan ölçütler Adımsal Doğrusal Regresyon Analizi ile seçilmiştir.

Senaryo sayısı, test planı oluşturma süresi, test/gereksinim inceleme süresi, test ortamı oluşturma süresi, toplantı sayısı ve hatalı test senaryosu sayısı Adımsal Doğrusal Regresyon Analizindeki p değerine bağlı olarak seçilmiştir. Birisi Dışarıda Çapraz Doğrulama tekniği uygulanmıştır.

Elde edilen tahmin denkleminin Pred (0,25) = 0,867 ve Pred(0,30) = 0,867 oranında doğru tahminler yaptığı görülmüştür. Elde edilen tahmin kalite değerleri literatürde belirtilen değerleri desteklemektedir.

Sonuç olarak, çalışma literatürdeki pred(0.25) ve pred(0.30) eşik değerlerini karşılamış ve önerilen yöntemin doğruluğu kabul edilebilir düzeyde kalmıştır.

Ortalama karekök hatası değerlerine bakıldığında karşılaştırma yapmak daha anlamlı sonuçlar elde etmemizi sağlamaktadır. Ortalama karekök hatası değeri ne kadar küçük olursa, hata miktarı o kadar küçük olur. Her bir aykırı değer modelden çıkarıldığında ortalama karekök hatası değerleri azalır. P1'deki ortalama karekök hatası en iyisi iken P9'daki ortalama karekök hatası ise en kötüsüdür.

Sonuç olarak, bu çalışma seçilmiş bazı yazılım test ölçütlerini dikkate alan yeni bir test efor tahmin yöntemi önermektedir. Bu seçilen yazılım test ölçütleri, literatürde geçen önceki yazılım test efor tahmin yöntemlerinde göz ardı edilen ölçütlerdir. Literatürde geçen yöntemler genellikle geliştirme efor tahminlemesinde kullanılan yöntemlerdir. Bu yöntemler test efor tahminlemesi için yeterli değildir. Literatürde geçen yöntemlerin hata oranları yüksektir. Bu çalışmada ise seçilen ölçütlerle yazılım test efor tahminlemesinde kabul edilebilir sonuçlar elde ederek test efor tahminlemesinde ölçüt kullanmasının önemli olduğu gösterilmiştir.

Bu çalışma kapsamında elde edilen test eforu tahmin denkleminin 15 proje ile sınırlıdır. Bu nedenle ileride farklı proje veya çalışmalarda daha fazla proje eklenerek modelin tahmin doğruluğunun daha çok proje için arttırılması hedeflenebilir.

KAYNAKLAR

[1] Malcolm J. Morgan , "Controlling Software Development Costs", Industrial Management & Data Systems, 1994, Vol. 94 No. 1, 1994, pp. 13-18,1994, 0263-5577.

[2] Debasisha Mishra and Biswajit Mahanty, "A study of software development project cost, schedule and quality by outsourcing to low cost destination", RG Indian Institute of Management, Shillong, India, and Department of Industrial and Systems Engineering, IIT Kharagpur, Kharagpur, India, Journal of Enterprise Information Management 29(3):454-478,2014, DOI:10.1108/JEIM-08-2014-0080.

[3] Jurka Rahikkala Vaadin Ltd, Turku, Finland, and Ville Leppänen, Jukka Ruohonen and Johannes Holvitie, "Top management support in software cost estimation A study of attitudes and practice in Finland", Department of Information Technology, University of Turku, Turku, Finland, International Journal of Managing Projects in Business 8(3):513-532 ,2015, DOI:10.1108/IJMPB-11-2014-0076.

[4] N. Ahmad, M.G.M. Khan and L.S. Raf, "A study of testing-effort dependent inflection S-shaped software reliability growth models with imperfect debugging", School of Computing Information and Mathematical Sciences, The University of the South Pacific, Suva, Fiji Islands,2009, DOI:10.1063/1.3516395.

[5] Priscila Pereira de Souza and Marcelo Werneck Barbosa, "Tailoring the Test Point Analysis Estimation Technique in a Software Testing Process", Pontificia Universidade Católica de Minas Gerais Belo Horizonte, Brazil, IV Encontro Brasileiro de Testes (EBTS) At: Recife,2010.

[6] Lava Kafle, "AN EMPIRICAL STUDY ON SOFTWARE TEST EFFORT ESTIMATION", International Journal of Soft Computing and Artificial Intelligence, 2014,Volume 2, Issue 2, Nepal.

[7] Natália França Felipe, Raphael Pena Cavalcanti, Eduardo Habib Bechelane Maia, Weber Porto Amaral, Augusto Campos Farnese, Leonardo Daniel Tavares, Eustáquio São José de Faria , Clarindo Isaias Pereira da Silva e Padua and Wilson de Pádua Paula Filho.(2014), "A Comparative Study of Three Test Effort Estimation Methods", Revista Cubana De Ciencias Informaticas, Brazil, RNPS:2301 SSN-e:2227.1899 vol.8.

[8] Rshma Chawla and Deepak Ahlawat," Software Development Effort Estimation using Fuzzy Logic Framework- An Implementation", International Journal On Advanced Computer Theory And Engineering (IJACTE),2017, Volume 4, Issue 4.

[9] Roy K. Clemmons, "Project Estimation With Use Case Points", Cross Talk, The journal of Defence Software Engineering, Diversified Technical Services, Inc. ,2014, Volume 19, Number 2.

[10] Aditi Sharma and Ravi Ranjan., "Software Effort Estimation using Neuro Fuzzy Inference System: Past and Present", International Journal on Recent and Innovation Trends in Computing and Communication, Delhi Technological University Delhi, India, 2017, ISSN: 2321-8169 2017, 5(8):78-83.

[11] Anish Mittal, Kamal Parkash and Harish Mittal, "Software Cost Estimation Using Fuzzy Logic", MMEC, ACM SIGSOFT Software Engineering Notes, Mullana Amballa, India, 2010, volume 35, Issue 1.

[12] Hamayoon Ghafory and Faqeed Ahmad Sahnosh, "The review of software cost estimation model: SLIM", International Journal of Advance Academic Studies, Rbbani Education University, Kabul, Afghanistan,2020, volume 2, Issue 4, Part H.

[13] Suresh Nageswaran, "Test Effort Estimation Using Use Case Points", Cognizant Technology Solutions, Quality Week 2001, San Francisco, California.

[14] Praveen Ranjan Srivastava, Abhishek Varshney and Priyanka Nama, "Software test effort estimation: a model based on cuckoo search", International Journal of Bio-Inspired

Computation, Department of Computer Science and Information Systems, Birla Institute of Technology and Science (BITS), Pilani, Vidya Vihar Campus, Pilani-333031, Rajasthan, India, 2012, pp 278–285, volume 4 Issue 5.

[15] Kamala Ramasubramani Jayakumar and Alain Abran, “A Survey of Software Test Estimation Techniques”, University of Quebec, Montreal, Canada, 2013, DOI: 10.4236/jsea.2013.610A006 , Volume 6, Issue 10.

[16] Gizem Kahveci, “Bankacılık Alanında Kişisel Yazılım Test Eforunu Tahmin Etmek İçin Proxy Tabanlı Bir Metot ve Vaka Çalışması”, Hacettepe Üniversitesi, 2017.

[17] Umar Farooq, S. M. K. Quadri and Nesar Ahmad, “SOFTWARE MEASUREMENTS AND METRICS: ROLE IN EFFECTIVE SOFTWARE TESTING”, Sheikh Umar Farooq et al. International Journal of Engineering Science and Technology (IJEST),2014, ISSN:0975-5462, Volume 3 Issue 1.

[18] Mr. Premal B. Nirpal and Dr. K. V. Kale, “A Brief Overview Of Software Testing Metrics”, International Journal of Advanced Research in Computer and Communication Engineering Department of CS & IT, Dr. B. A. M. University, Aurangabad, India, 2014,ISSN (Print) : 2319-5940, Volume 2, Issue 12.

[19] “Black box testing” https://en.wikipedia.org/wiki/Black-box_testing (Accessed : July 2021).

[20] “What Is Software Testing Life Cycle (STLC)?” https://www.softwaretestinghelp.com/what-is-software-testing-life-cycle-stlc/ (Accessed : July 2021).

[21] “Project test life cycle” https://www.softwaretestingmaterial.com/stlc-software-testing-life-cycle/ (Accessed : July 2021).

[22] “Minitab Stepwise Regression” https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/regression/how-to/fit-regression-model/perform-the-analysis/perform-stepwise-regression/ (Accessed : July 2021).

[23] Tülin Erçelebi Ayyıldız ve Begüm Erkal, ”Yazılım Hata Kestiriminde Nesne Yönelimli Ölçütlerin Etkisi ”, International Journal of Systems and Management Research ,2019, Vol. 1, No. 1 pp 1-8.

[24] "K fold and other cross-validation techniques", https://medium.com/datadriveninvestor /kfold-and-other-cross-validation-techniques-6c03a2563f1e (Accessed: July 2021).

[25] Tülin Erçelebi Ayyıldız ve Hasan Can Terzi, ” Case Study on Software Effort Estimation”, International Journal of Information and Electronics Engineering,2017, Vol. 7, No. 3.

[26] C. DeSanto, M. Totoro, and R. Moscartelli, “Introduction to statistics 9th edition,” Pearson, 2010, Volume 32, Issue 3.

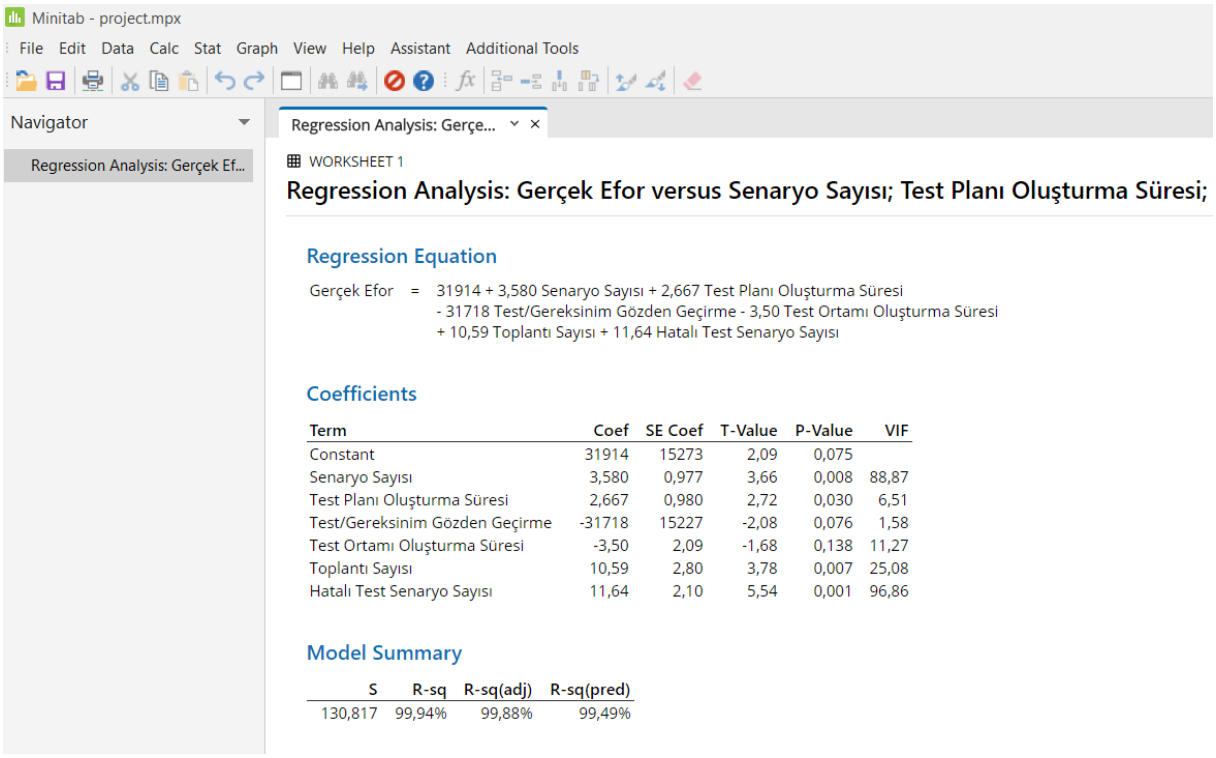
[27] S. D. Conte, H. E. Dunsmore and V.Y. Shen, “Software Engineering Metrics and Models”, Redwood City, CA, USA: The Benjamin/Cummings, 1986 - 396 pages, ISBN:978-0-8053-2162-3.

[28] G. Tate and J. Verner. “ Software Costing in Practice”, The Economics of Information Systems and Software, Oxford, Butterworth-Heinemann,1991, pp. 101-126.

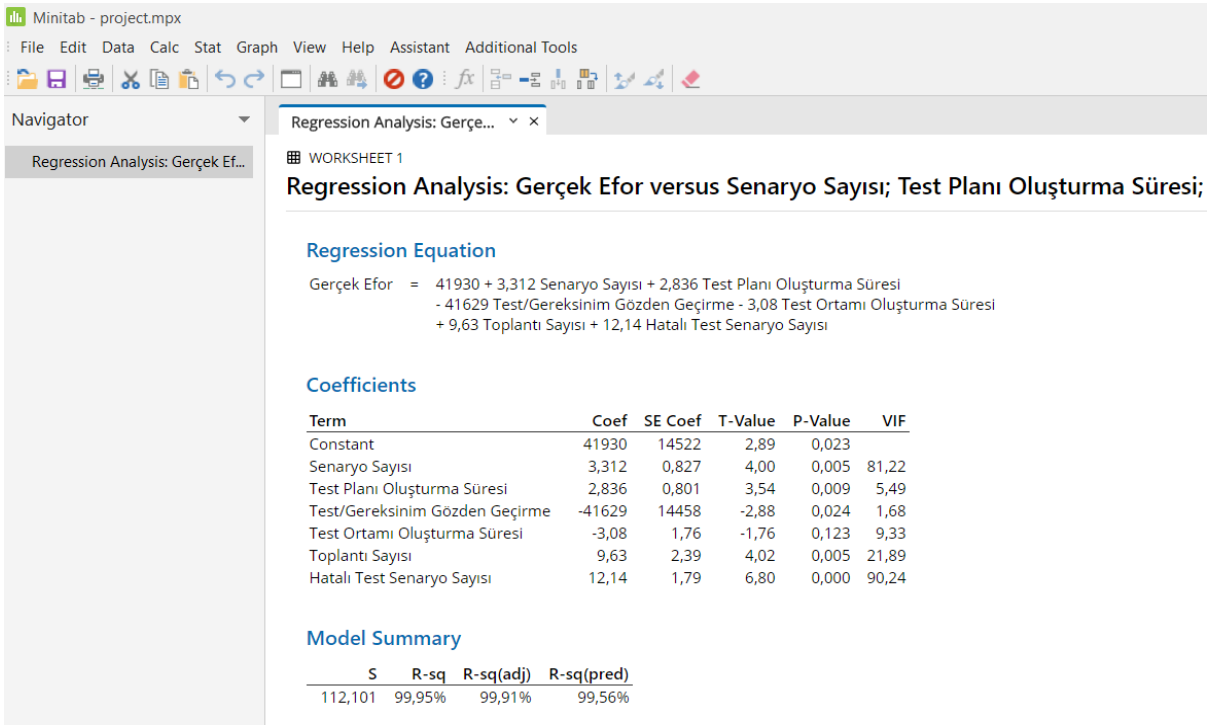
[29] T. E. Hastings and A. S. M. Sajeev, “A vector based approach to software size measurement and effort estimation,” IEEE Transactions on Software Engineering,2001, vol. 24, no. 4, pp. 337-350.

EKLER

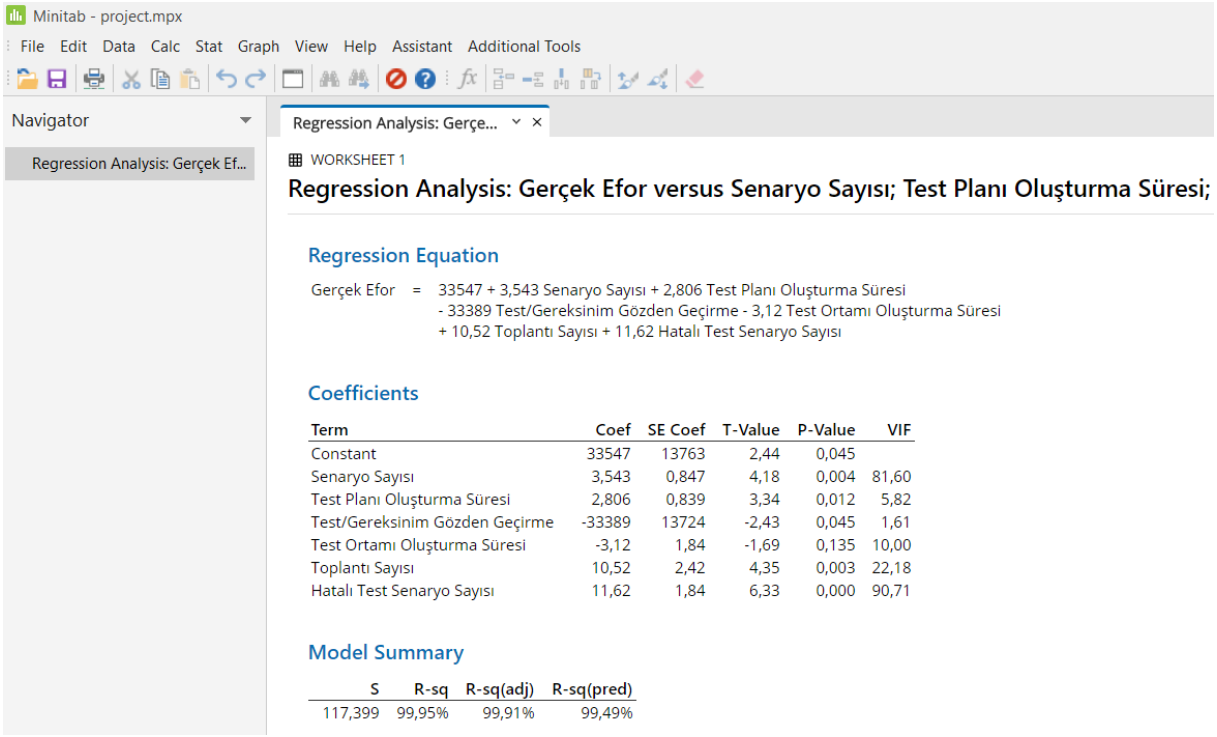
EK 1: Projelerin Regresyon Denklemleri



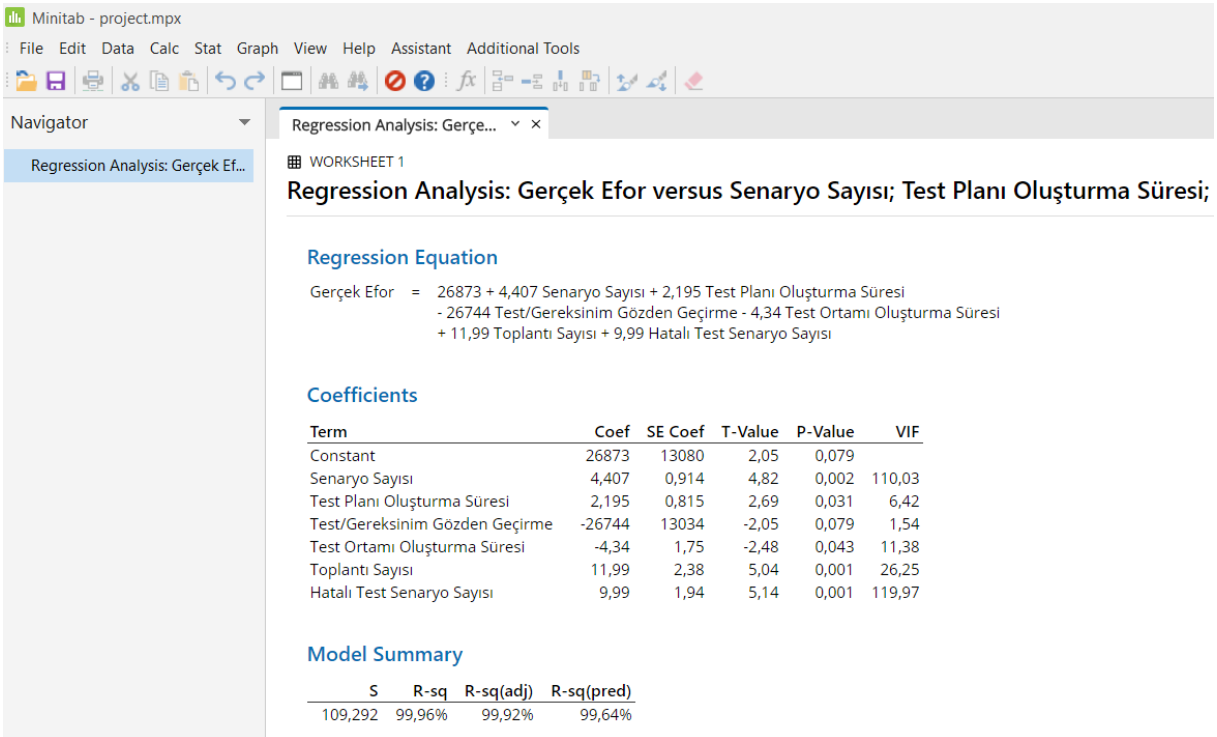
Proje 1 İçin Oluşan Regresyon Denklemi



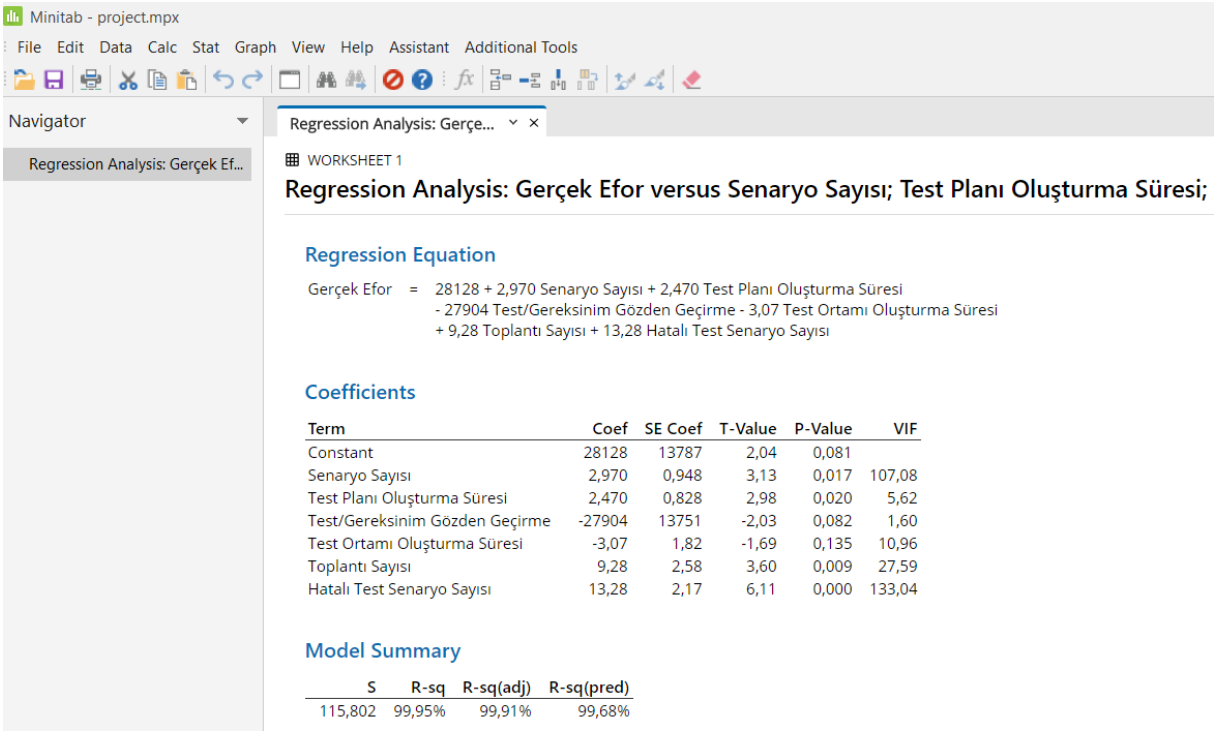
Proje 2 İçin Oluşan Regresyon Denklemi



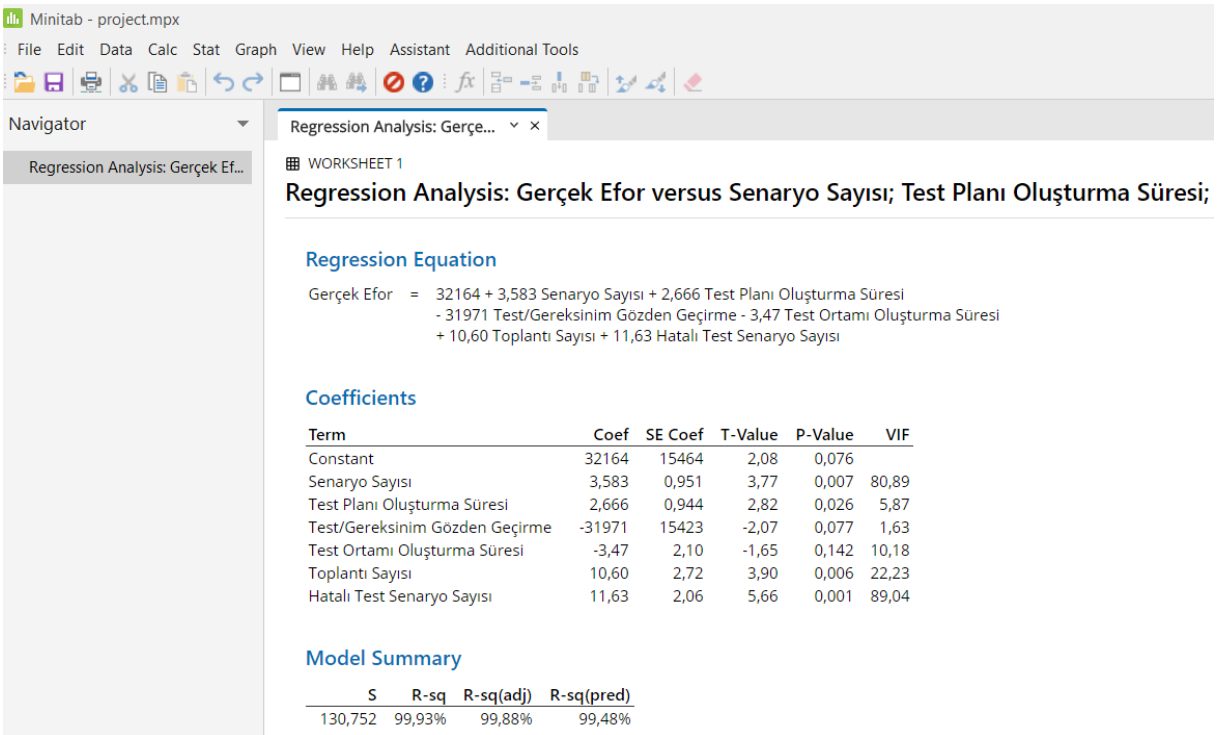
Proje 3 İçin Oluşan Regresyon Denklemi



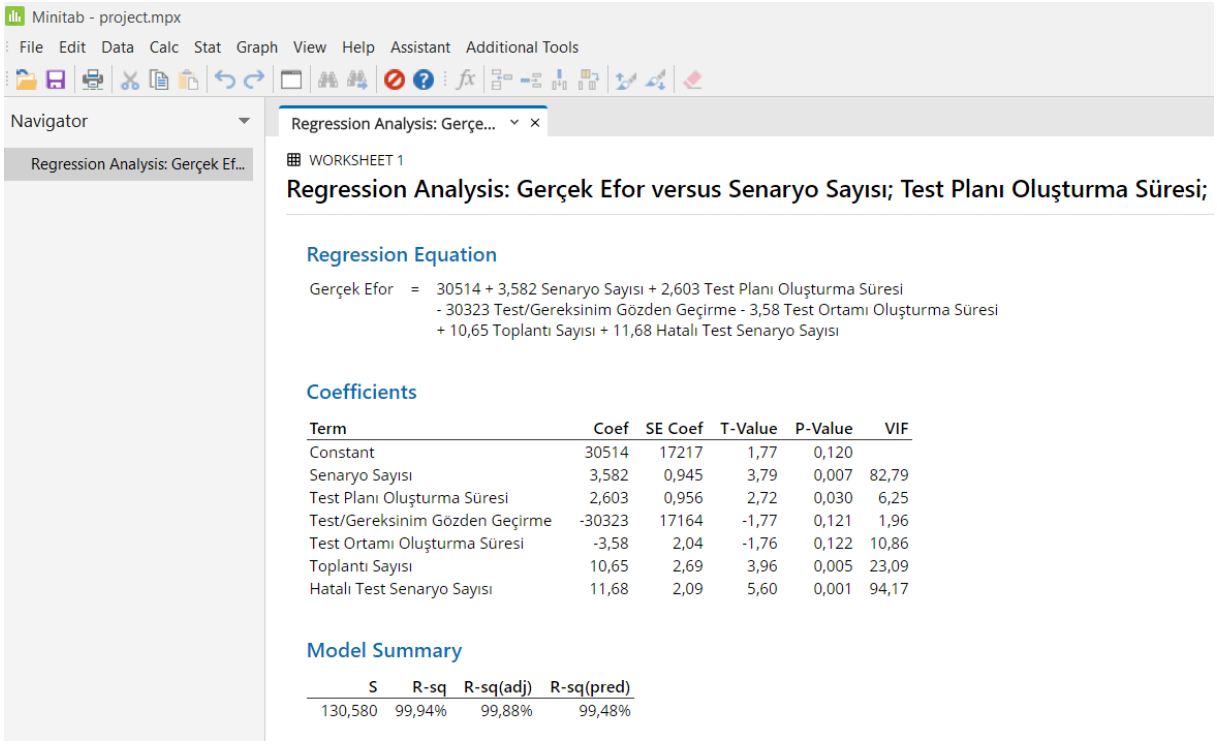
Proje 4 İçin Oluşan Regresyon Denklemi



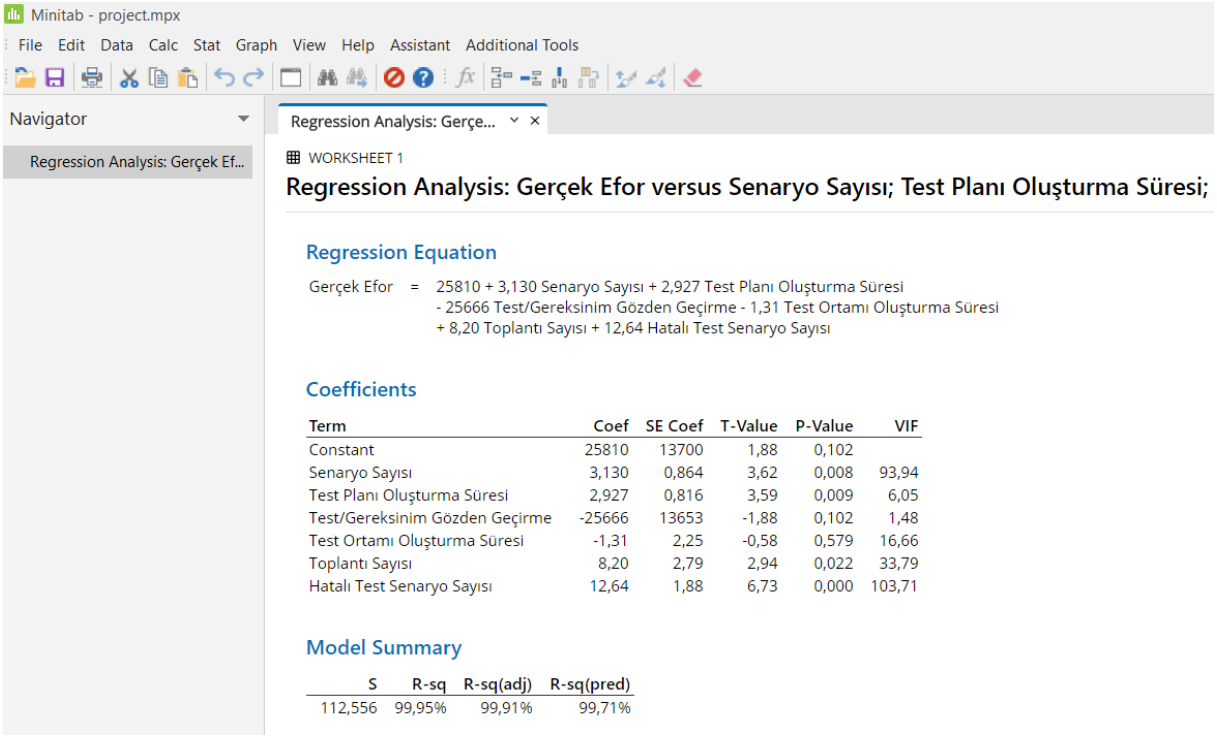
Proje 5 İçin Oluşan Regresyon Denklemi



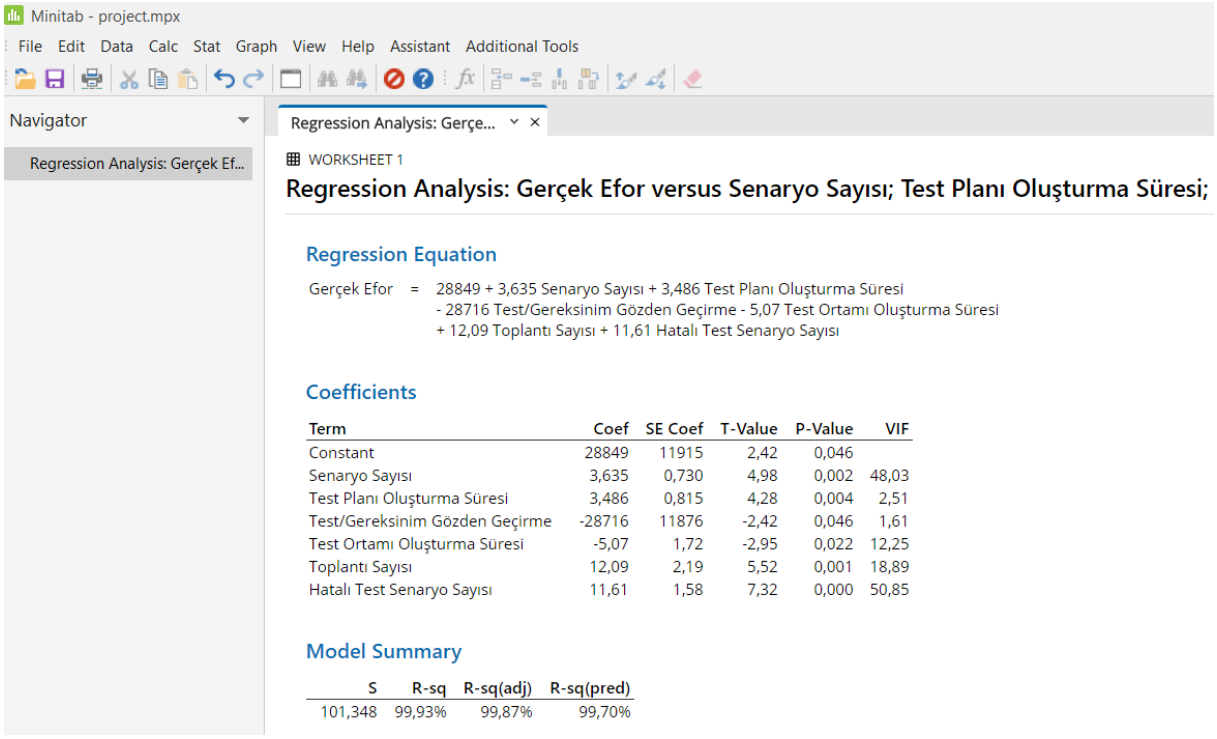
Proje 6 İçin Oluşan Regresyon Denklemi



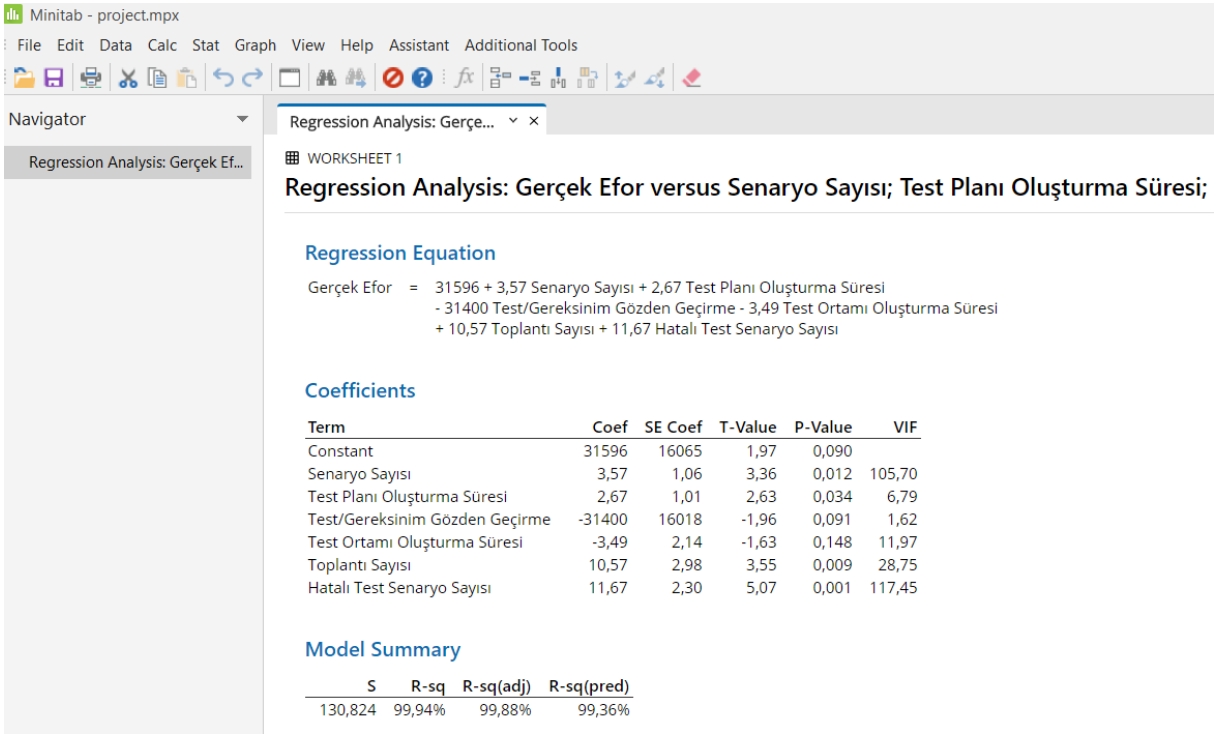
Proje 7 İçin Oluşan Regresyon Denklemi



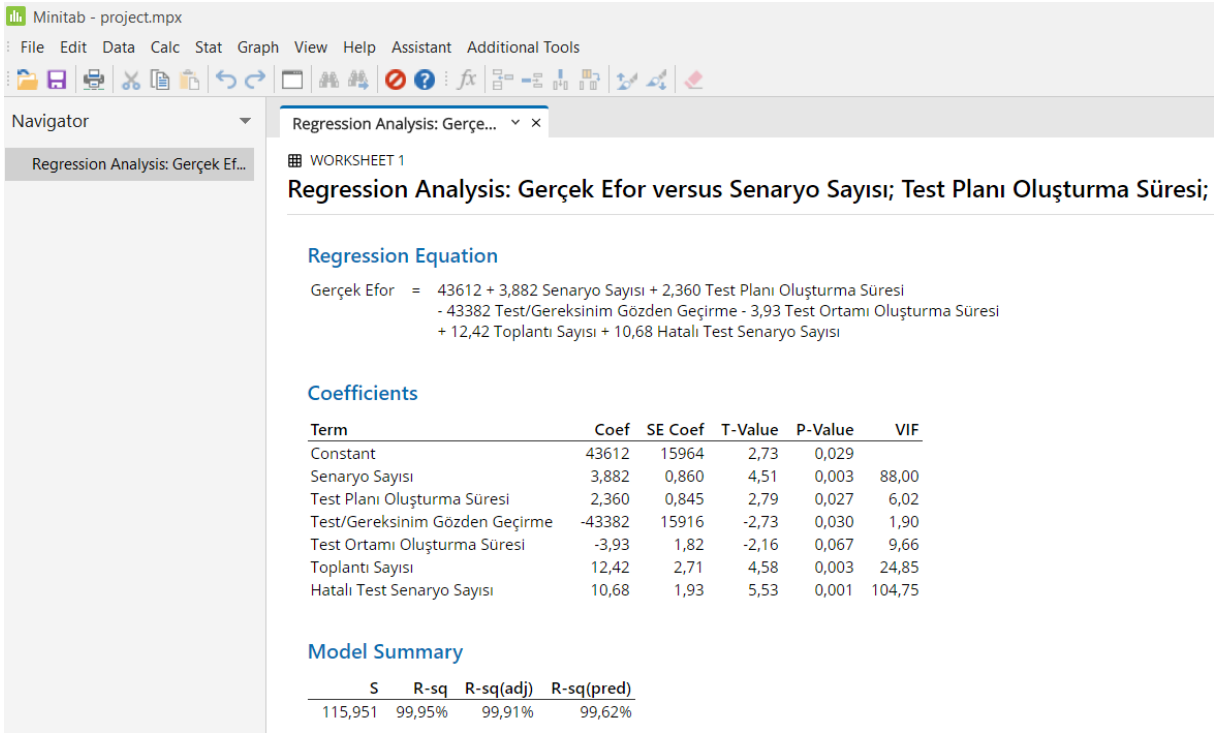
Proje 8 İçin Oluşan Regresyon Denklemi



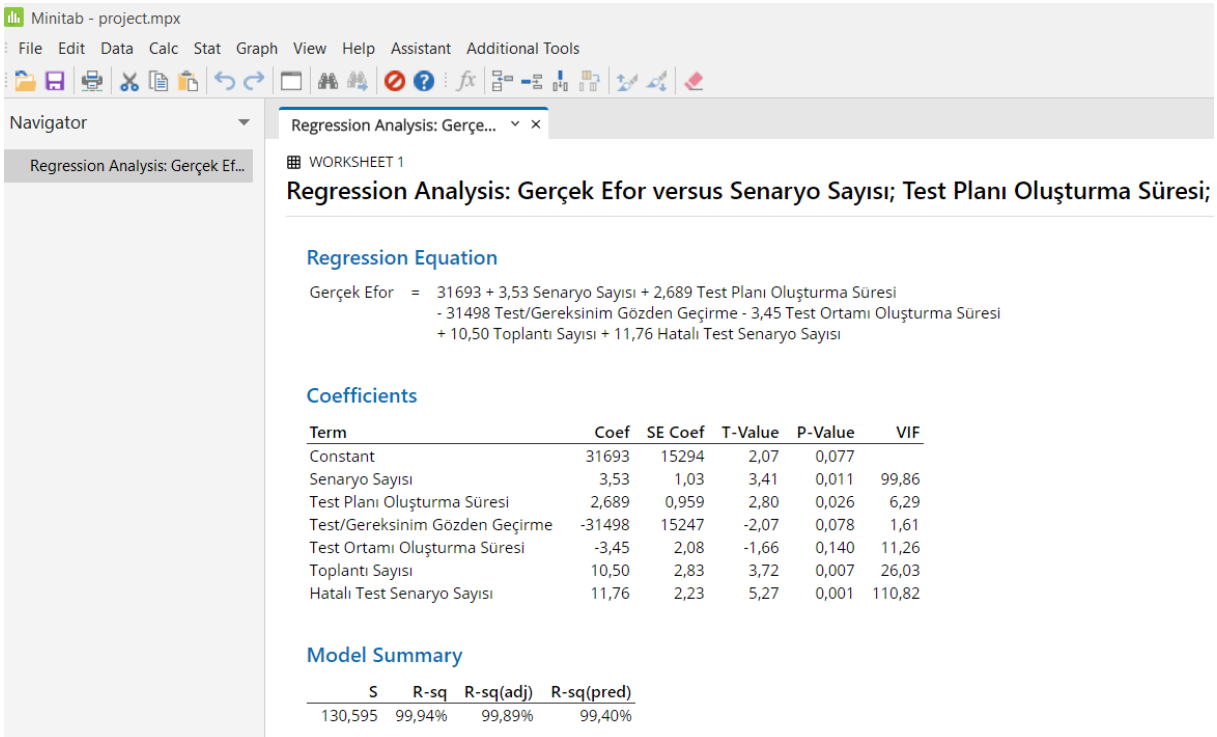
Proje 9 İçin Oluşan Regresyon Denklemi



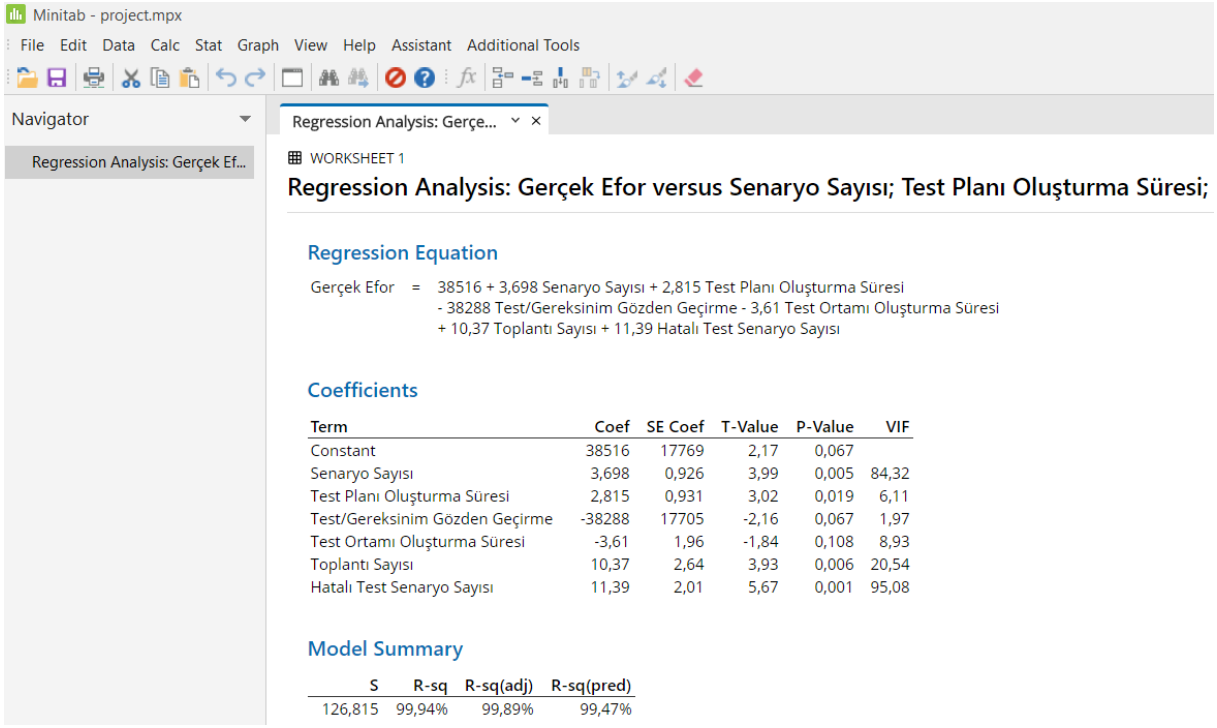
Proje 10 İçin Oluşan Regresyon Denklemi



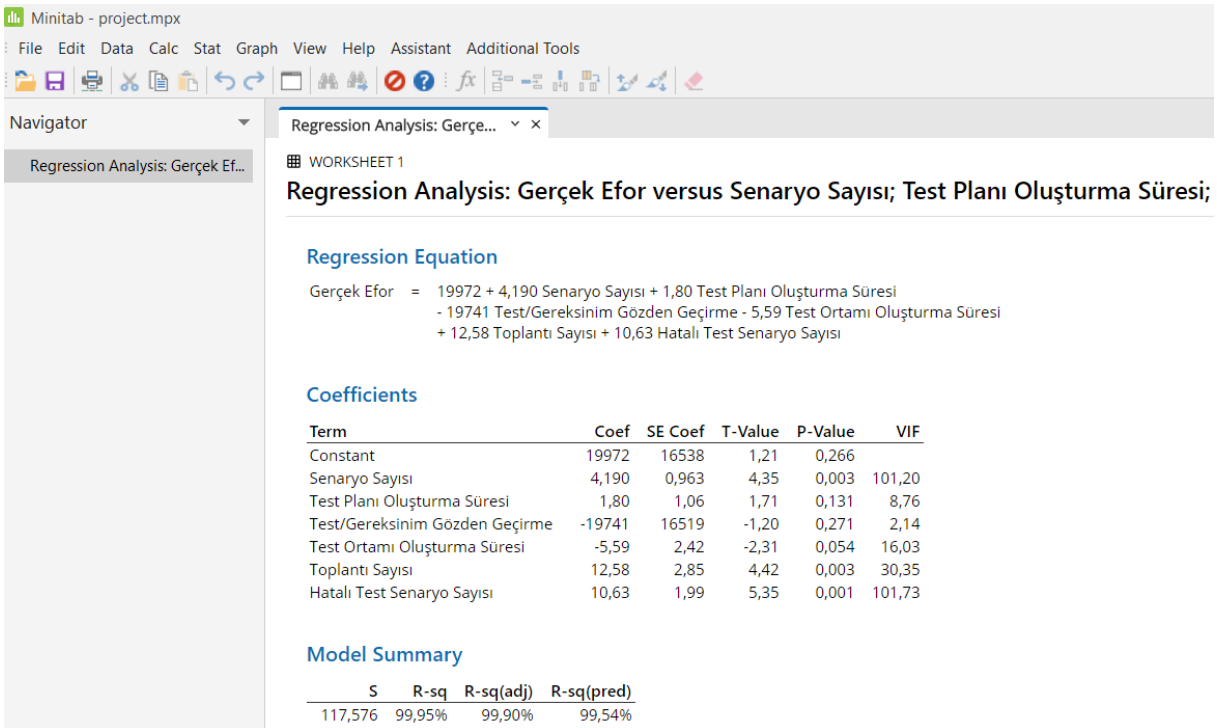
Proje 11 İçin Oluşan Regresyon Denklemi



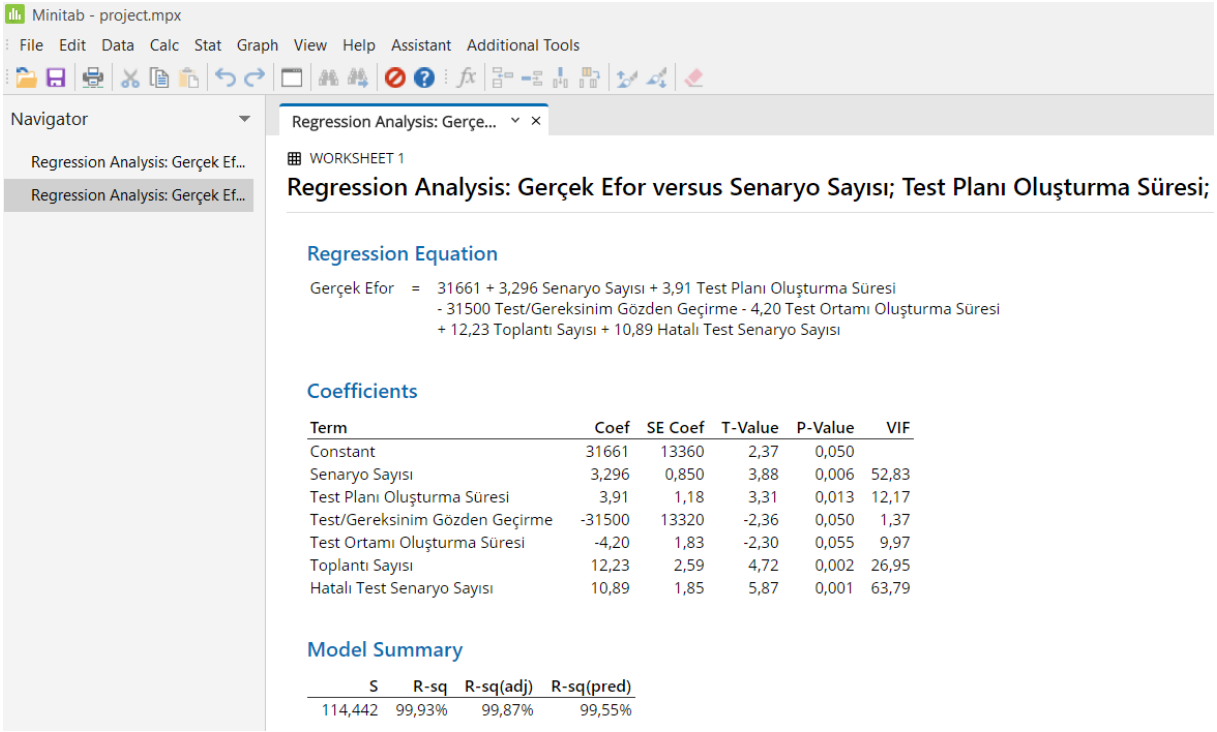
Proje 12 İçin Oluşan Regresyon Denklemi



Proje 13 İçin Oluşan Regresyon Denklemi



Proje 14 İçin Oluşan Regresyon Denklemi



Proje 15 İçin Oluşan Regresyon Denklemi