GAZİ ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BM496 BİLGİSAYAR
PROJESİ

SOFTWARE DESIGN DOCUMENT (SDD)

Green AI ile Kredi Kartı Fraud Analizi

Öğr. Gör. Dr. MUHAMMET ÜNAL

181180005 - Emre Can Ant
181180014 - Büşra Bayındır
C181112034 - Erencan Tezel

2023

İÇİNDEKİLER

1. GİRİŞ

Projenin amacı, hedef kitlesi, proje sdd dokümanında kullandığımız ve bilinmesi gereken kısaltmalar ve tanımlar ve referanslar açıklanmıştır.

1.1. Amaç

Bu projenin amacı, Green AI'ın kredi kartı dolandırıcılık analizlerinde kullanımını incelemek ve güç tüketim analizini yapmaktır. Proje özellikle, kuruluşların kredi kartı sahtekarlığını tespit edip önlemek ve aynı zamanda karbon ayak izlerini ve çevresel etkilerini azaltmak için AI teknolojilerinden hangisinin seçilebileceği ve nasıl yararlanabileceğini keşfetmeyi amaçlamaktadır. Projemiz, yeşil yapay zekanın faydalarını vurgulayarak, dolandırıcılık tespiti ve önleme alanında çevresel olarak sürdürülebilir uygulamaların benimsenmesini teşvik etmeyi ve sonuçta daha sürdürülebilir bir geleceğe katkıda bulunmayı amaçlamaktadır.

1.2. Hedef Kitle

Bu projenin hedef kitlesi, kredi kartı dolandırıcılığını tespit etmek ve önlemek için yenilikçi çözümler keşfetmekle ve aynı zamanda operasyonlarına çevresel olarak sürdürülebilir uygulamaları dahil etmekle ilgilenen finans sektöründeki profesyoneller ve karar vericilerdir. Buna bankacılık, finansal hizmetler ve ödeme işleme sektörlerindeki yöneticiler, müdürler ve analistlerin yanı sıra yapay zeka ve çevresel sürdürülebilirliğin kesişimiyle ilgilenen araştırmacılar ve akademisyenleri dahil edebiliriz.

1.3. Kısaltmalar ve Tanımlar

Kısaltmalar	**Açıklamalar**
AI	Artificial Intelligence
CSV	Virgülle ayrılan değerler [1] (Dosya Formatı).
JSON	JavaScript Object Notation.
QT	Platform bağımsız arayüz tasarım çerçevesi

MB	Megabyte

Tanımlar	**Açıklamalar**
Kredi Kartı Dolandırıcılığı	Bir kredi kartının, genellikle kartın veya kart bilgilerinin çalınması yoluyla, hileli bir şekilde mal veya hizmet elde etmek için yetkisiz kullanımı.
Green AI	Karbon ayak izini ve çevresel etkisini en aza indirmek için tasarlanmış Yapay Zeka. Yeşil yapay zeka.
Karbon Ayak İzi	Bir kişi, kuruluş veya ürün tarafından salınan toplam sera gazı miktarı.
Framework	Çerçeve. Standart fonksiyonların hazır olarak sunulduğu ancak programcı tarafından bu fonksiyonlardan arzu edilen kısımların ek kodlarla istenildiği şekilde güncellenebildiği sistemlerdir.
Çevresel Sürdürülebilirlik	Gelecek nesillerin kendi ihtiyaçlarını karşılama yeteneğinden ödün vermeden bugünün ihtiyaçlarını karşılamak için doğal kaynakların sorumlu kullanımı.
Izgara Görünümü	Verilerin gösterilmesi, bir arada toplanması için kullanılan hazır araç.

1.4. Referanslar

[1] "Comma Separated Values (CSV) Standard File Format". Edoceo, Inc. Retrieved June 4, 2014.

[2] "Framework". *DocForge*. Archived from the original on 7 October 2018. Retrieved 15 December 2008.

[3] Blanchette, Jasmin; Summerfield, Mark (June 2006). "A Brief History of Qt". C++ GUI Programming with Qt 4 (1st ed.). Prentice-Hall. pp. xv–xvii. Archived from the original on 1 October 2020. Retrieved 5 August 2013.
[4] Khan, Mohammad & Masud, Mehedi & Aljahdali, Sultan & Kaur, Manjit & Singh, Parminder. (2021). A Comparative Analysis of Machine Learning Algorithms to Predict Alzheimer's Disease. Journal of Healthcare Engineering. 2021. 10.1155/2021/9917919.

1.5. Genel Bakış

Belge, projenin nasıl yapılacağı ile ilgili bilgiler sunar. Tasarım viewpointleri ile görsel içeriklere yer verilmiştir. UML, Use Case, Sekans, flowchart ile tasarım detaylı bir biçimde açıklanmıştır.

2. İLGİLİ DÖKÜMANLAR

a. Green AI ile Kredi Kartı Fraud Analizi, Literatür Taraması Dokümanı

b. Green AI ile Kredi Kartı Fraud Analizi, SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS) Dokümanı

3. KAVRAMSAL MODEL

Bu bölümde SDD ile ilgili temel terimler, kavramlar ve içerik verilecektir.

3.1. Bağlam Üzerinde Yazılım Tasarımı

Projenin amacı, kredi kartı fraud analizi için kullanılabilecek makine öğrenmesi modellerini bulmak, her model için; modelin eğitim ve analiz sırasında ne kadar güç tükettiğini hesaplayıp loglamak, bu verilere dayanarak en uygun modeli oluşturmaktır. En verimli ve doğru modeli bulduktan sonra kullanıcılar için bir arayüz sağlayarak geçmiş dönemdeki yapılmış işlemleri analiz ederek geçmişe yönelik kontrolleri sağlamaktır. Projede arayüz ve socket programlama için C++ ve Qt/QML kullanılacak olup model eğitimi ve işlem kontrolü gibi kısımlarda Python kullanılacaktır.

3.2. Yazılım Yaşam Döngüsü Ürünleri Üzerindeki Etkiler

3.2.1. SDD Hazırlamadaki Etkiler

Bir yazılım tasarımını yönlendiren temel yazılım yaşam döngüsü ürünü, tipik olarak yazılımın gereksinimleridir. Yapılan SRS dokümanındaki gereksinimler (işlevsel ve işlevsel olmayan gereksinimler ve arayüz gereksinimleri) ve ayrıca paydaşların talepleri, projenin tasarımı üzerinde etkilidir.

3.2.2. Yazılım Yaşam Döngüsü Ürünleri Üzerindeki Etkiler

SDD'nin hazırlık aşamasında ve/veya projenin uygulama aşamasında bazı gereksinimler değişebilir. Ayrıca SDD, Kredi Kartı Fraud analiz sisteminin test planlarını ve test belgelerini etkiler.

4. YAPISAL TASARIM

Kredi Kartı Dolandırıcılık Analiz Sistemi'nin Yapısal Tasarım Açıklaması, bu sistemin nasıl tasarlanıp uygulanacağını tanımlar. Belge tanımlaması boyunca diyagramlar, kullanıcı görünümleri ve kullanıcı viewpointleri kullanılmıştır.

4.1. Tasarım Paydaşları

Sistemin kullanımı kolay, arayüzleri ilgi çekici ve modern olmalıdır. Ayrıca şahısların bankacılık işlemleri gibi özel bilgilerine içermesi sebebiyle çeşitli güvenlik önlemlerine sahip olmalıdır.

4.2. Tasarım Bakış Açıları

Bu belge bağlam, mantıksal, arayüz ve etkileşim viewpointlerini açıklar.

4.2.1. Bağlam Bakış Açısı

Kullanıcılar ve etkileşimde bulunan diğer paydaşlar gibi sistem ile çevresi arasındaki ilişkileri, bağımlılıkları ve etkileşimleri açıklar. Sistem sınırını gösteren bir kullanım durumu, bağlam ve blok diyagramı içerir.

4.2.2. Etkileşim Bakış Açısı

Eylemlerin sırasını ve sistemde eylemlerin nasıl, neden, nerede ve hangi düzeyde gerçekleştiğini açıklar. Bu proje için durum dinamiği görünümlerinin ayrıntılı olarak kullanılması tercih edilir.

4.2.3. Mantıksal Bakış Açısı

Sınıf yapılarını, aralarındaki etkileşimleri ve bunların nasıl tasarlanıp uygulandığını açıklar. Ayrıca mevcut mantıksal bileşenlerin geliştirilmesini ve yeniden kullanılmasını destekler. Nesneleri ve sınıfları ve aralarındaki ilişkileri tanımlayan sınıf diyagramı içerir.

4.2.4. Arayüz Bakış Açısı

İç ve dış arayüzlerin ayrıntılarını açıklar. Sistemin detaylı tasarımına geçmeden önce tasarımcılara, programcılara ve test uzmanlarına bilgi sağlar.

4.3. Tasarım Ögeleri

Tasarım görünümünde görünen herhangi bir öğe tasarım öğeleri olarak adlandırılır. Bu alt durumlardan biri veya birkaçı olabilir; tasarım varlığı, tasarım ilişkisi, tasarım niteliği ve tasarım kısıtlamaları. Tüm tasarım öğeleri, yazılım tasarım açıklamasının 5. bölümünde karşılık gelen bakış açıları altında alt durumlarla tanımlanır.

4.4. Tasarım Gerekçeleri

Tasarım sırasında nesne yönelimli yaklaşım seçilmiştir, çünkü bu sayede donanım kısmı ve yazılım kısmı kolayca birleştirilecektir. Yazılım bölümü, notların ayrıştırılmasını ve okunmasını ve verilerin iletilmesini içerir. Bu paketleri tasarlamak için çok sayıda paket kullanıldı. Bu paketler birbirine bağlıdır ve ayrı ayrı kontrol edilebilirler. Ayrıca, donanım parçası için bir paket kullanılmış ve yazılım ile donanım parçalarını birleştirmek için başka bir paket daha bulunmaktadır.

4.5. Tasarım Dilleri

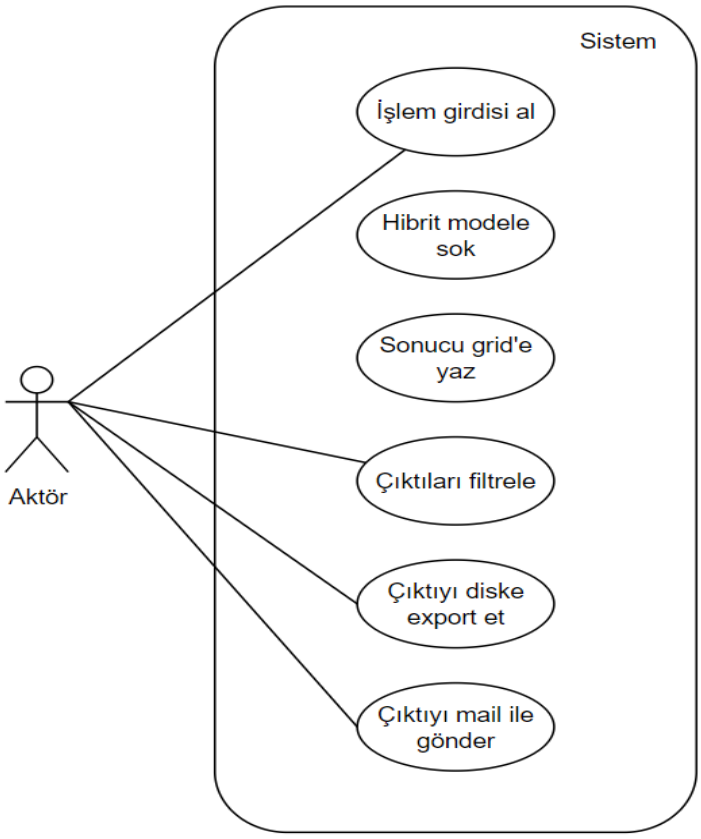
Bu belgede, diyagramlar için modelleme dili olarak Birleşik Modelleme Dili (UML) kullanılacaktır

5. TASARIM BAKIŞ AÇILARI

Bu bölüm, ilgili tasarım kaygıları ve uygun tasarım dilleri ile Kredi Kartı Fraud Analiz Sisteminin birkaç ana tasarım viewpointi sağlıyor. Sırasıyla bağlam, mantıksal, arayüz ve etkileşim bakış açıları aşağıdaki alt bölümlerde tanımlanmıştır. Her bakış açısı için minimum tasarım varlıkları kümesi, tasarım ilişkileri, tasarım varlığı özniteliği ile ilgili kısa açıklamalar sağlanmıştır.

5.1. Bağlam Bakış Açısı

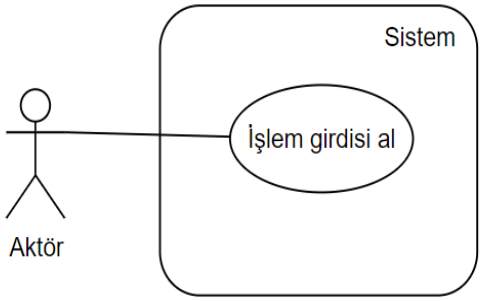
Sistemde sadece bir tür kullanıcı vardır. Sistem, güç tüketimi analizi sonuçlarından yararlanılarak seçilmiş en verilmi ve accuracy değeri o derece yüksek modele sokma, sonucu data grid'e yazma, çıktıları filtreleme, çıktıyı diske export etme, çıktıyı mail ile gönderme olmak üzere yedi hizmet sağlar. Kullanım durumları hakkında daha ayrıntılı bilgi aşağıda sunulmuştur. Şekil 5.1'de sistem use case diyagramı verilmiştir.



Şekil 5.1. Sistem Use Case Diyagramı

5.1.1. İşlem girdisi al

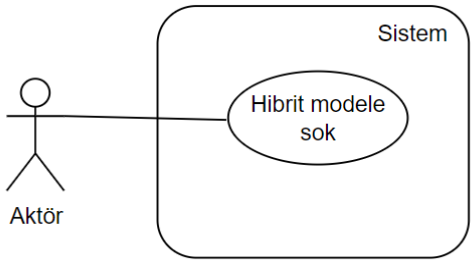
Sistemin işlemleri sınıflandırılabilmesi için sisteme kredi kartı işlem verisi/verileri verilmelidir. Bu fonksiyonda csv formatında dosya okuması işlemi yapılır. Şekil 5.2'de verilmiştir.



Şekil 5.2. İşlem girdisi al Use Case

5.1.2. Modele sok

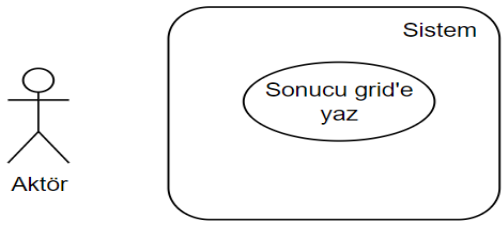
Aktör ile gelen işlem verileri, güç tüketimi analiziyle belirlenecek modele sonuçları üretmek amacıyla sokulur. Kullanıcının butona basması beklenir. Modele 6. bölümde değinilmiştir. Şekil 5.3'te gösterilmiştir.



Şekil 5.3. Modele sok Use Case

5.1.3. Sonucu gride yaz

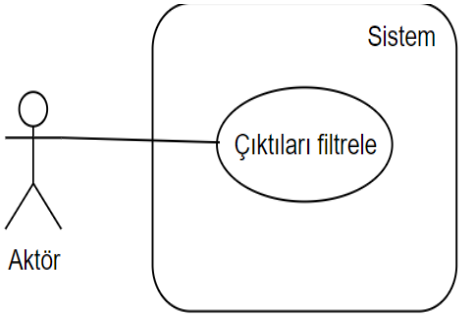
Modele sokulduktan sonra elde edilen sonuçlar kullanıcının görebilmesi için ağaç yapısındaki data gride aktarılır. Şekil 5.4'te gösterilmiştir.



Şekil 5.4. Sonucu data grid'e yaz Use Case

5.1.4. Çıktıları filtrele

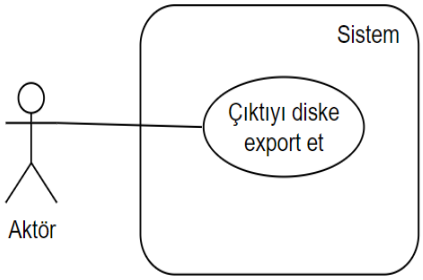
Aktörün arayüzden seçebileceği filtreler (sadece fraudulent işlemleri göster, sadece normal işlemleri göster, tüm işlemleri göster) ile data grid'de gösterilenler güncellenir. 'Filtreyi Uygula' butonuna basılması beklenir. Şekil 5.5'te gösterilmiştir.



Şekil 5.5. Çıktıları filtrele Use Case

5.1.5. Çıktıyı diske export et

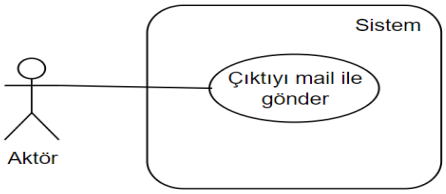
Filtrelenen veya tüm sonuçlar aktör tarafından belirlenen konuma export edilir. "Dışa Aktar" butonuna basılması beklenir. Şekil 5.6'da gösterilmiştir.



Şekil 5.6. Çıktıyı diske export et Use Case

5.1.6. Çıktıyı mail ile gönder

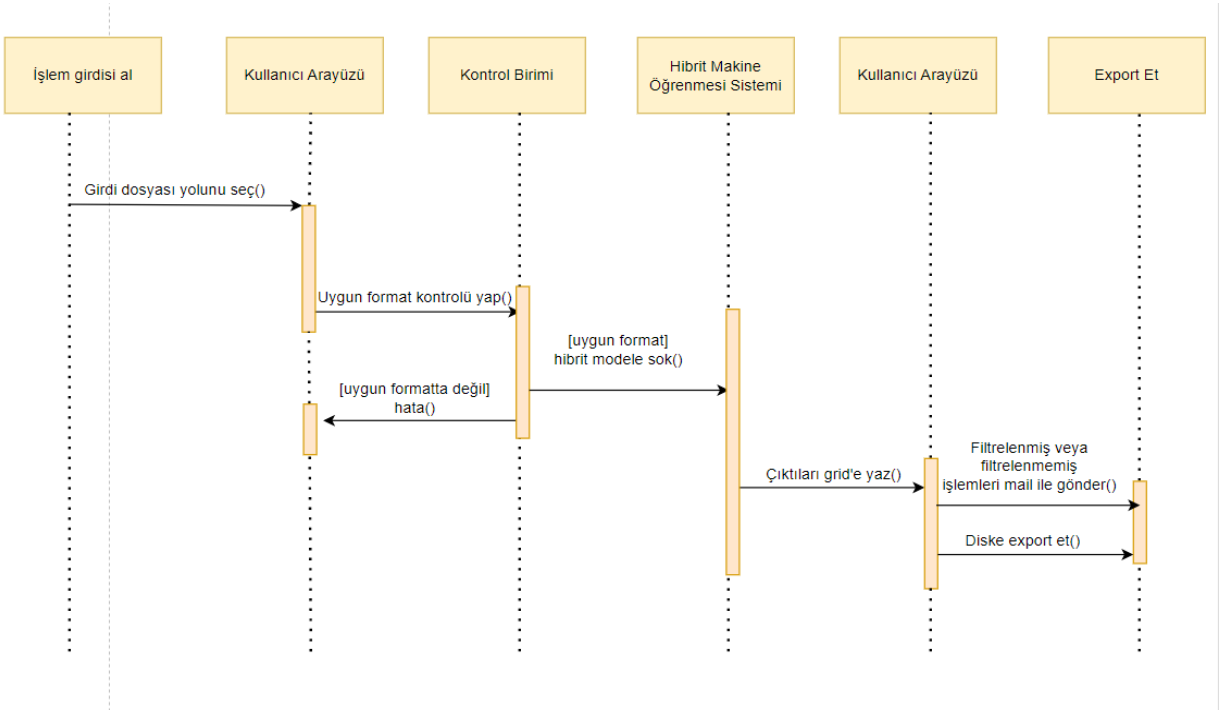
Filtrelenen veya tüm sonuçlar aktör tarafından belirlenen mail adresine gönderilir. "Mail ile Gönder" butonuna basılması beklenir. Şekil 5.7'de verilmiştir.



Şekil 5.7. Çıktıyı mail ile gönder Use Case

5.2. Etkileşim Bakış Açısı

Bu bölümde, sistemin temel işlevleri sekans diyagramı yardımı ile verilmektedir. Ayrıca varlıklar arasındaki etkileşim için stratejiler tanımlar. Şekil 5.8'de sistemin sekans diyagramı verilmiştir.

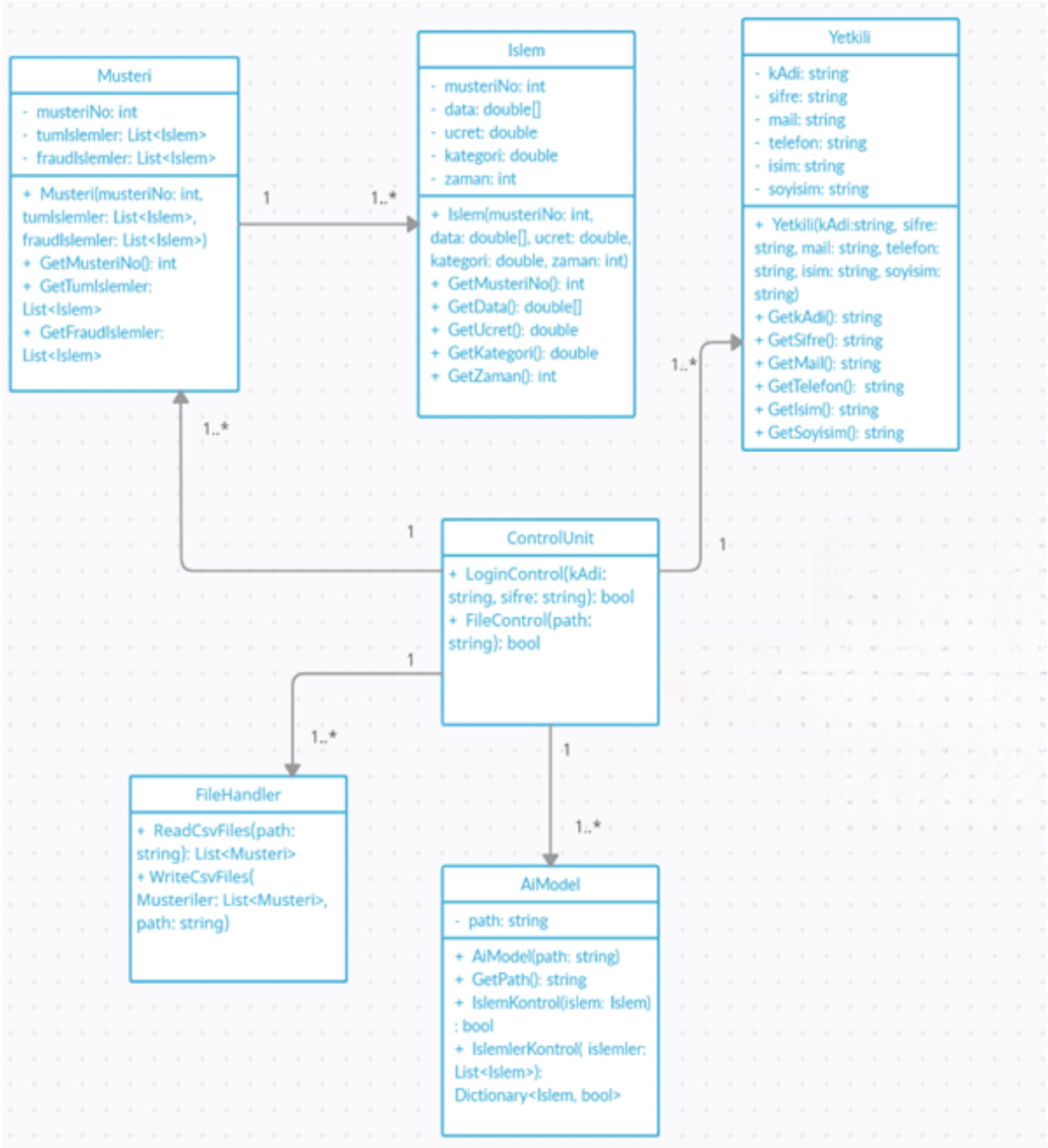


Şekil 5.8. Sekans diyagramı

Sistem sekansları yukarıdaki diyagram biçiminde tamamlar. Kullanıcı arayüzden girdiyi verir. Girdi istenen formatta mı kontrolü, kontrol birimi tarafından yapılır. Uygun formatta değil ise kullanıcı arayüzüne ilgili hata mesajı gönderilir. Uygun formatta gönderilen girdiler modele sokulur. Sonuçlar data grid'e yazılır. İsteğe bağlı olarak filtrelenebilir çıktılar diske veya kullanıcının girdiği bir e-mail'e export edilir.

5.3. Mantıksal Bakış Açısı

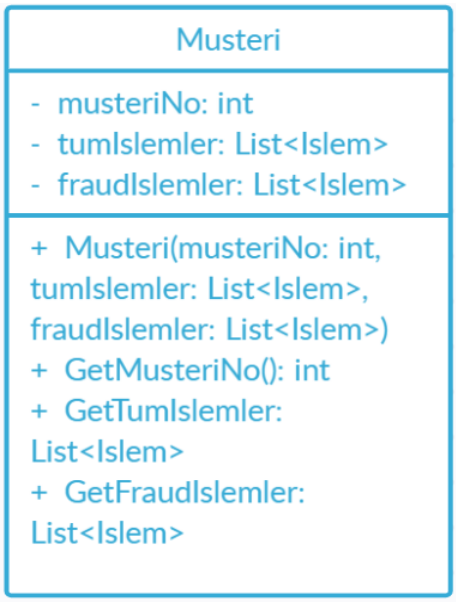
Mantıksal bakış açıyla UML diyagramı Şekil 5.9'da verilmiştir.



Şekil 5.9. Sistemin UML Class Diyagramı

5.3.1. Müşteri Sınıfı

Bu sınıf bankada hesabı bulunan kişilerin firma tarafından verilen bilgilerini ve işlemlerini içermektedir. Şekil 5.10'da verilmiştir.



Şekil 5.10. Musteri sınıfı

5.3.1.1. Değişkenler

5.3.1.1.1. musteriNo: Her bir müşteri için eşsiz bir integer değerdir.

5.3.1.1.2. tumIslemler: Kullanıcı tarafından yapılmış tüm işlemleri Islem nesnesinin listesi halinde tutan değişkendir.

5.3.1.1.3. fraudIslemler: Sadece fraud pozitif çıkan işlemleri Islem nesnesinin listesi halinde tutan değişkendir. Modele sokulduktan sonra çıkan sonuçlara göre oluşturulur.

5.3.1.2. Metodlar

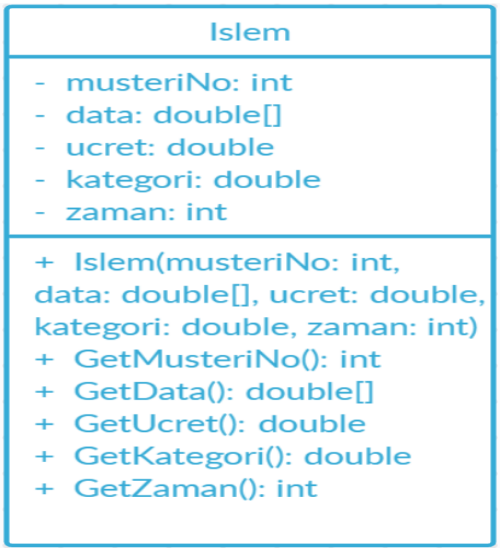
5.3.1.2.1. Musteri(): Musteri sınıfının oluşturucusudur. Gerekli kontrolleri ve ilk değer. atamalarını gerçekleştirmektedir.

5.3.1.2.2. GetTumIslemler: tumIslemler değişkenini geri dönen fonksiyondur.

5.3.1.2.3. GetFraudIslemler: fraudIslemler değişkenini geri dönen fonksiyondur.

5.3.2. İşlem Sınıfı

Bu sınıf KuveytTürk'ün sağlamış olduğu her bir işlem verisini tutmaktadır. Şekil 5.11'de verilmiştir.



Şekil 5.11. Islem sınıfı

5.3.2.1. Değişkenler

5.4.2.1.1. musteriNo: Her bir müşteri için eşsiz bir integer değerdir.

5.4.2.1.2. data: KuveytTürk tarafından sağlanan normalizasyondan geçirilmiş içeriği gizli double türünde verilerdir.

5.4.2.1.3. ucret: Yapılan işlem tutarlarının normalizasyondan geçirilmiş halini double türünde tutan değişkendir.

5.4.2.1.4. kategori: Yapılan işlem kategorisini double veya integer türünde tutacak olan değişkendir.

5.4.2.1.5. zaman: Önceki işlem ile şuanki işlem arasında geçen zamanı belirten integer türündeki değişkendir.

5.3.2.2. Metodlar

5.3.2.2.1. Islem(): Islem sınıfının oluşturucusudur. Gerekli kontrolleri ve ilk değer atamalarını gerçekleştirmektedir.

5.3.2.2.2. GetMusteriNo(): musteriNo değişkenini geri dönen fonksiyondur.

5.3.2.2.3. GetData(): data değişkenini geri dönen fonksiyondur.

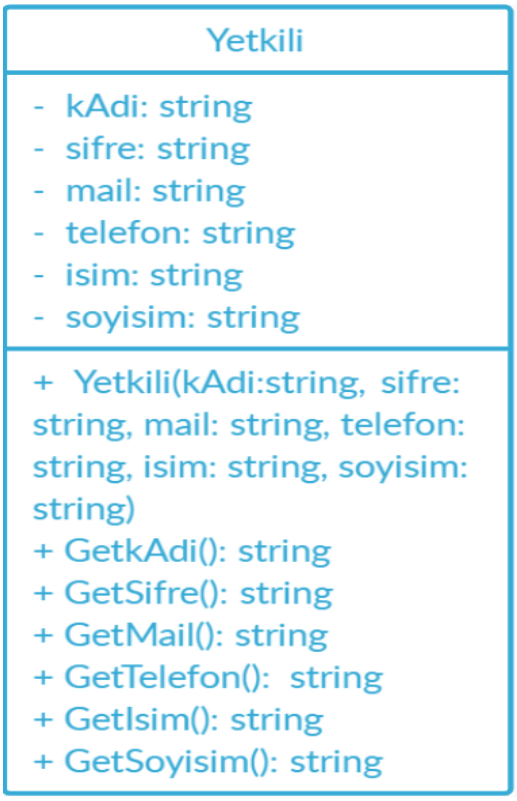
5.3.2.2.4. GetUcret(): ucret değişkenini geri dönen fonksiyondur.

5.3.2.2.5. GetKategori(): kategori değişkenini geri dönen fonksiyondur.

5.3.2.2.6. GetZaman(): zaman değişkenini geri dönen fonksiyondur.

5.3.3. Yetkili Sınıfı

Bu sınıf yönetici yetkisine sahip kişilerin bilgilerini tutmaktadır. Şekil 5.12'de verilmiştir.



Şekil 5.12. Yetkili sınıfı

5.3.3.1. Değişkenler

5.3.3.1.1. kAdi: Her bir yetkili için eşsiz bir string değerdir.

5.3.3.1.2. sifre: En az 8 karakterden oluşan string bir değerdir.

5.3.3.1.3. mail: Her bir yetkili için eşsiz bir string değerdir.

5.3.3.1.4. telefon: Her bir yetkili için eşsiz bir string değerdir.

5.3.3.1.5. isim: Yetkilinin log işlemlerinde tutulan ismini ifade eden string değerdir.

5.3.3.1.6. soyisim: Yetkilinin log işlemlerinde tutulan soyismini ifade eden string değerdir.

5.3.3.2. Metodlar

5.3.3.2.1. Yetkili(): Yetkili sınıfının oluşturucusudur. Gerekli kontrolleri ve ilk değer atamalarını gerçekleştirmektedir.

5.3.3.2.2. GetkAdi(): kAdi değişkenini geri dönen fonksiyondur.

5.3.3.2.3. GetSifre(): sifre değişkenini geri dönen fonksiyondur.

5.3.3.2.4. GetMail(): mail değişkenini geri dönen fonksiyondur.

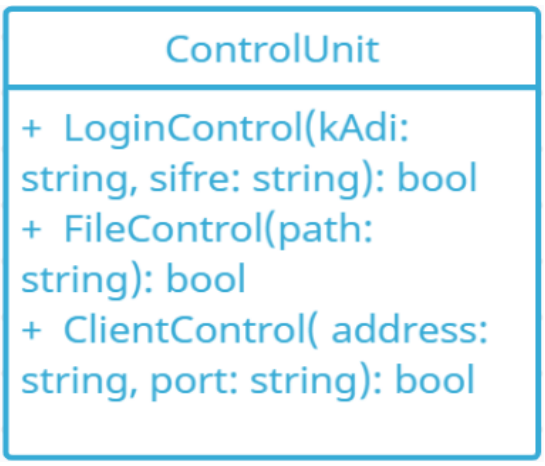
5.3.3.2.5. GetTelefon(): telefon değişkenini geri dönen fonksiyondur.

5.3.3.2.6. GetIsim(): isim değişkenini geri dönen fonksiyondur.

5.3.3.2.7. GetSoyisim(): soyisim değişkenini geri dönen fonksiyondur.

5.3.4. ControlUnit Sınıfı

Bu sınıf, sınıflar arası bağlantılar ve metod çağırmaları yaparak yazılımın genel kontrol ve görev akış işlemlerini yapmaktadır. Şekil 5.13'te verilmiştir.



Şekil 5.13. ControlUniti sınıfı

5.3.4.1. Metodlar

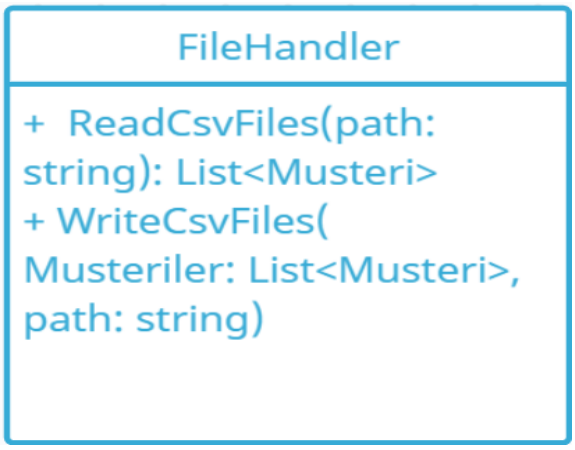
5.3.4.1.1 LoginControl(): Gönderilen kullanıcı adı ve şifreleri sistemde kayıtlı kullanıcı adı şifrelerle karşılaştırarak giriş onayını bool cinsinden dönen fonksiyondur.

5.3.4.1.2. FileControl(): Verilen dosya yolunun programa uygun olmadığını ve dosyanın gerçekte varolup olmadığı gibi çeşitli kontrolleri yapan ve sonucu bool türünden dönen fonksiyondur.

5.3.4.1.3. ClientControl(): Client'in başarılı bir şekilde çalışıp çalışmadığını çeşitli şekillerde deneyerek sonucu bool türünden dönen fonksiyondur.

5.3.6. FileHandler Sınıfı

Dosya okuma yazma gibi işlemlerin gerçekleştirildiği sınıftır. Şekil 5.14'te verilmiştir.



Şekil 5.14. FileHandler sınıfı

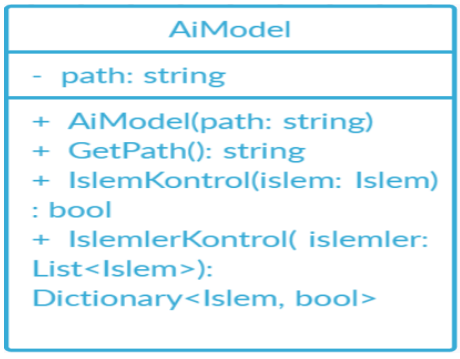
5.3.6.1. Metodlar

*5.4.6.1.1. ReadCsvFiles():*Verilen klasördeki uygun formattaki tüm dosyaları okuyarak elde ettiği verilerden müşteri ve işlem nesleleri oluşturan ve bunları liste halinde döndüren fonksiyondur.

5.4.6.1.2. WriteCsvFiles(): Argüman olarak yollanan müşteri listesindeki işlem verilerini önceden belirlenmiş formatta dosyalara yazan fonksiyondur.

5.3.7. AiModel Sınıfı

Eğitilmiş yapay zeka modelini kullanarak dolandırıcılık tespiti yapan sınıftır. Şekil 5.15'de verilmiştir.



Şekil 5.15. AiModel sınıfı

5.3.7.1. Değişkenler

5.3.7.1.1. path: Eğitilmiş yapay zeka modelinin dosya konumunu belirten string değerdir.

5.3.7.2. Metodlar

5.3.7.2.1. AiModel(): AiModel sınıfının oluşturucusudur. Gerekli kontrolleri ve ilk değer atamalarını gerçekleştirmektedir.

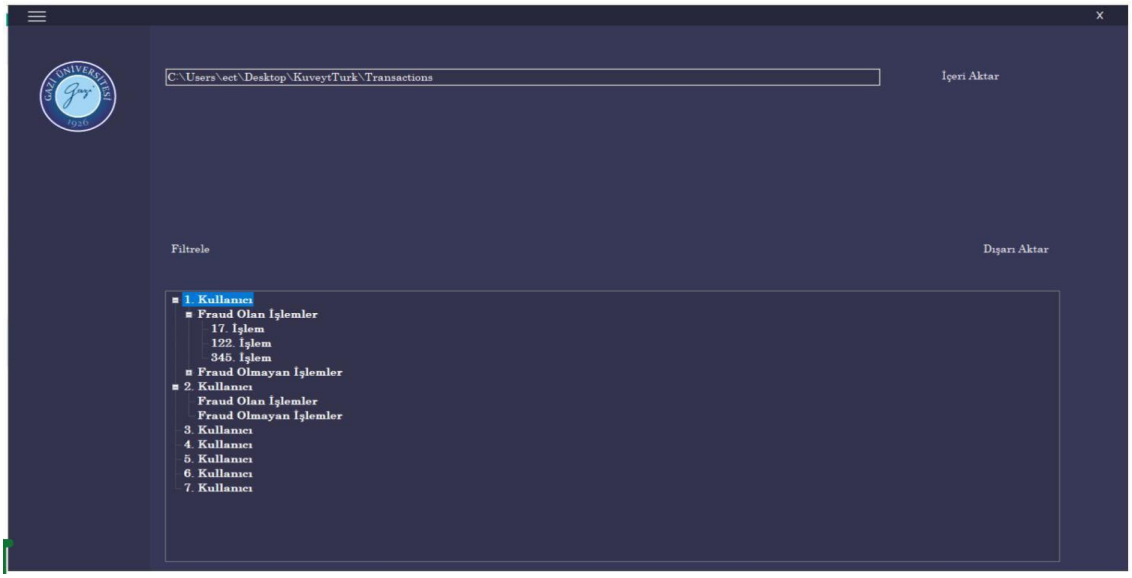
5.3.7.2.2. GetPath(): path değişkenini dönen fonksiyondur..

5.3.7.2.3. IslemKontrol: Gönderilen Islem nesnesindeki değerlere bakarak dolandırıcılık analizi yapan ve çıkan sonucu bool türünde geri döndüren fonksiyondur.

5.3.7.2.1.4. IslemlerKontrol(): Gönderilen Islem nesnelerindeki değerlere bakarak dolandırıcılık analizi yapan ve çıkan sonuçları Islem nesnesi ve bool türündeki sonuç şeklinde bir dictionary türündeki değişkene kaydederek geri döndüren fonksiyondur.

5.4. Arayüz Viewpoint

Dokümanın bu bölümünde, harici ve dahili arayüzlerin detayları tanımlanacaktır. Şekil 5.16'da düşünülen arayüz görseli verilmiştir.



Şekil 5.16. Sistem Dosya Modu arayüzü

Kullanıcı uygulamayı çalıştırdığında karşılaşacağı dosya modunda, işlem verilerinin tutulduğu dosyayı browse edebilmek için bir dosya konum girdisi alanı ve o dosya üzerinde çalışmak istediğimizi belirtmek için içeri aktar butonu bulunmaktadır. Filtreleme seçenekleri bu ekranda bulunmaktadır. Dışa aktar seçeneği seçildiğinde dışarı aktarma arayüzü gelmesi planlanmaktadır. Seçeneğe göre mail yoluyla veya belirlenen dizine, filtrelenen işlemler export edilmesi için dışarı aktar butonu yerleştirilmiştir.Fraudulent işlemler filtrelenebilir. Filtrelenenler export edilebilir. Arayüz QT ile tasarlanmıştır.

6. SDD DETAYLI PLANI

Bu bölümde projenin nasıl yapılacağı ve nelerin yapıldığı noktalarına değinilmiştir.

6.1. VERİ SETİ ÖN İŞLEMİ, MAKİNE ÖĞRENMESİ MODELLERİ VE GÜÇ TÜKETİMİ

Bu ksım sırasıyla veri seti üzerinde gerçekleştirilen ön işlemler, bu ön işlemlerden geçmiş verilerin modeller üzerinde eğitilmesi ve her modelin bu verisetleriyle çalışırken ne kadar güç harcadığı kısımlarına değinmektedir.

6.1.1 Yapılan Ön İşlemler

Makine öğrenmesi sürecinden önce modellerin eğitimi için veri seti ön işlemlere tabii tutulur. Eğitim için kullanılan veri seti fazlaca dengesizdir.

```
0    284315
1       492
Name: Class, dtype: int64
```

Şekil 6.1. European Kredi Kartı İşlem Dataset içerikleri

Kullanılan veri setinde toplam 284807 işlem kaydı bulunmaktadır. Labelı 0 olan normal işlemleri, 1 olan ise fraudulent işlemleri temsil etmektedir.

Dengesiz dağılımdan bir nebze kurtulmak için normal işlemlerde under-sampling yapılmıştır. "legit", orijinal veri setinin sadece normal olan işlemlerini içerir.

```
legit_sample= legit.sample(n=180000)
```

Şekil 6.2. Under-sampling

Under-sampling yapıldıktan sonra fraudulent işlemlerle under-sampling yapılmış set birleştirilir Şekil 6.3'te gösterilmiştir. "fraud", orijinal veri setinin sadece fraudulent işlemlerini içerir.

```
new_dataset=pd.concat([legit_sample,fraud],axis=0)
```

Şekil 6.3. Yeni veri setinin elde edilmesi

Veri setinde Time ve Amount attributeları da modellerde daha iyi sonuçlar verebilmesi için -1,1 arasına normalize edilmiştir.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaling = scaler.fit_transform(np.array(new_dataset['Amount']).reshape(-1, 1))
scaling2 = scaler.fit_transform(np.array(new_dataset['Time']).reshape(-1, 1))
new_dataset['Amount']=scaling
new_dataset['Time']=scaling2
X = new_dataset.drop('Class', axis=1)
y = new_dataset['Class']
```

Şekil 6.4. Time ve Amount normalizasyonu

new_dataset oluşturulduktan sonra dengesiz olan verisetini dengeli hale getirebilmek için SMOTE tekniği kullanılmıştır. Yapay zekayla sentetik fraudulent işlemler üretir. Model eğitimlerinde kullanılacak olan veriseti df_smote oluşturulmuştur. 180.000 normal, 180.000 fraudulent işlem içerir. Şekil 6.5'te bu işlemler gösterilmiştir.

```
from collections import Counter
from imblearn.combine import SMOTETomek

#Implementing the technique
smk = SMOTETomek(random_state=42)

# fit and apply the transform
X_smk, y_smk = smk.fit_resample(X, y)

#Make a train set dataframe for SMOTE
df_smote = X_smk
df_smote['Class']=y_smk
```

Şekil 6.5. Smote ile balanced veriseti elde etme

Kullanacağımız veri setini oluşturduktan sonra %70'ü eğitim %30'u ise test olmak üzere parçaladık.Şekil 6.6'da gösterilmiştir.

```
X=df_smote.drop(columns='Class', axis=1)
Y=df_smote['Class']

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3, stratify=Y, random_state=42)
```

Şekil 6.6. Train ve Test datasetlerin oluştulması

6.1.2. Kullanılan Modeller

GreenAI'ı bulmak adına kullanacağımız, testini gerçekleştireceğimiz makine öğrenmesi algoritmaları Logistic Regression, Decision Tree, Random Forest, kNN ve kMEANS algoritmalarıdır. Süreç boyunca daha fazla makine örğenmesi modeli eklenmeye müsaittir. Şekil 6.7, Şekil 6.8'de bu modellerin kodları mevcuttur.

```
def fonksiyon1():
    # Logistic Regression

    model= LogisticRegression()
    # training  the logistic regression model with traning data (X_train)

    model.fit(X_train,Y_train) # X_train tüm featureları tutuyo. Y_train onlara karşılık gelen Label'ları tutuyor.
    #Accuracy Score
    #accuracy on training data
    X_train_prediction = model.predict(X_train)
    training_data_accuracy= accuracy_score(X_train_prediction, Y_train)

    log("Accuracy on Training data; "+ str(training_data_accuracy))

    #accuracy on test data
    X_test_prediction = model.predict(X_test)
    test_data_accuracy= accuracy_score(X_test_prediction, Y_test)

    log("Accuracy on Test data; "+str(test_data_accuracy))

def fonksiyon2():
    #Decision Tree Classification

    classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 42)
    classifier.fit(X_train, Y_train)
    y_pred=classifier.predict(X_test)
    cm = confusion_matrix(Y_test, y_pred)
    log(str(cm))
    accuracy_score(Y_test, y_pred)
    Accuracy_Decison = ((cm[0][0] + cm[1][1]) / cm.sum()) *100
    log("dec tree Accuracy_Decison    : "+ str(Accuracy_Decison))
    Error_rate = ((cm[0][1] + cm[1][0]) / cm.sum()) *100
    log("Error_rate  : "+str(Error_rate))
```

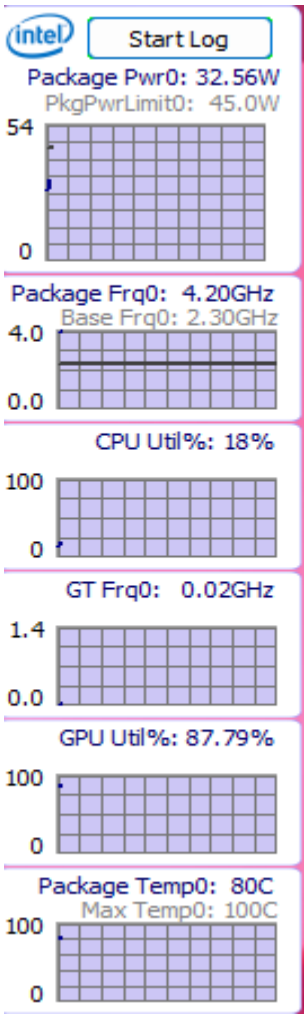
Şekil 6.7. Logistic Regression ve Decision Tree algoritmaları uygulaması

```
def fonksiyon3():
    #Random Tree Classification

    classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 42)
    classifier.fit(X_train, Y_train)
    y_pred=classifier.predict(X_test)
    cm = confusion_matrix(Y_test, y_pred)
    log(str(cm))
    accuracy_score(Y_test, y_pred)
    Accuracy_Decison = ((cm[0][0] + cm[1][1]) / cm.sum()) *100
    log("ranDOM foresT Accuracy_Decison     : "+ str(Accuracy_Decison))
    Error_rate = ((cm[0][1] + cm[1][0]) / cm.sum()) *100
    log("Error_rate  : "+str(Error_rate))

def fonksiyon4():
    # KNN Classifier

    classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
    classifier.fit(X_train, Y_train)
    y_pred = classifier.predict(X_test)
    cm = confusion_matrix(Y_test, y_pred)
    log(str(cm))
    accuracy_score(Y_test, y_pred)
    Accuracy_Decison = ((cm[0][0] + cm[1][1]) / cm.sum()) *100
    log(" knn Accuracy_Decison     : "+str( Accuracy_Decison))
    Error_rate = ((cm[0][1] + cm[1][0]) / cm.sum()) *100
    log("Error_rate  : "+str( Error_rate))

def fonksiyon5():
    #KMEANS
    kmeans = KMeans(init='k-means++', n_clusters=2, n_init=10)
    kmeans.fit(X_train)

    predictions = kmeans.predict(X_test)

    pred_fraud = np.where(predictions == 1)[0]
    real_fraud = np.where(Y_test == 1)[0]
    false_pos = len(np.setdiff1d(pred_fraud, real_fraud))

    pred_good = np.where(predictions == 0)[0]
    real_good = np.where(Y_test == 0)[0]
    false_neg = len(np.setdiff1d(pred_good, real_good))

    false_neg_rate = false_neg/(false_pos+false_neg)

    accuracy = (len(X_test) - (false_neg + false_pos)) / len(X_test)

    log("Accuracy:" + str(accuracy))
    log("False negative rate (with respect to misclassifications): "+ str(false_neg_rate))
    log("False negative rate (with respect to all the data): "+ str((false_neg / len(predictions))))
    log("False negatives, false positives, mispredictions:"+str (false_neg)+" "+str(false_pos)+" " +str(false_neg + false_pos))
    log("Total test data points:"+ str(len(X_test)))
```

Şekil 6.8. Random Forest, kNN ve Kmeans algoritmaları uygulaması

6.1.3. Güç Tüketimi Analizi

Güç tüketimi analizi için kontrollü bir deney yapılması planlanmaktadır. Bu analizle literatüre katkı sağlanması amaçlanmaktadır. Güç tüketimi 3. parti bir yazılım olan Intel Power Gadget ile tespit edilecektir. Yazılım 100ms aralıklarıyla güç tüketimini ölçecektir. Yazılım içinde kendi loglama sistemi mevcuttur. Şekil 6.9'da yazılımın arayüzü sunulmuştur.



Şekil 6.9. Intel Power Gadget Arayüzü

Analizin yapılabilmesi için oluşturduğumuz makine öğrenmesi modellerini aynı dataset ön işlemleriyle öğrenme sürecine girmesi ve öğrenme aşamasından sonra test edilmesi gerekmektedir. Sistem aynı şartlar altındayken hem öğrenme sürecinde ortalama, min ve max güç tüketimi verileri hesaplanacak, hem de test edilme sürecindeki güç tüketim sonuçları hesaplanacaktır. Log dosyası csv formatında oluşturulur. 26 feature içermektedir. Bu featurelar aşağıda sıralanmıştır:

a. System Time

b. RDTSC

c. Elapsed Time (sec)

d. CPU Utilization(%)

e. CPU Frequency_0(MHz)

f. Processor Power_0(Watt)

g. Cumulative Processor Energy_0(Joules)

h. Cumulative Processor Energy_0(mWh)

i. IA Power_0(Watt)

j. Cumulative IA Energy_0(Joules)

k. Cumulative IA Energy_0(mWh)

l. Package Temperature_0(C)

m. Package Hot_0

n. DRAM Power_0(Watt)

o. Cumulative DRAM Energy_0(Joules)

p. Cumulative DRAM Energy_0(mWh)

q. GT Power_0(Watt)

r. Cumulative GT Energy_0(Joules)

s. Cumulative GT Energy_0(mWh)

t. Package PL1_0(Watt)

u. Package PL2_0(Watt)

v. Package PL4_0(Watt)

w. Platform PsysPL1_0(Watt)

x. Platform PsysPL2_0(Watt)

y. GT Frequency(MHz)

z. GT Utilization(%)

Geçen süre, toplam işlemci enerjisi, ortalama işlemci gücü, program tarafından log işleminin sonunda otomatik olarak oluşturulur. Örnek bir log işlemi sonucu Şekil 6.10'da verilmiştir.

A	B	C	D	E	F	G	H	I	J	K	L	M
00:41:23:654, 451941649126357,	23.293,	4.000, 4500,	9.880, 299.255,	83.126,	7.796, 244.961,	68.045, 69,	0,	0.000,	0.000,	0.000,	0.	
00:41:23:760, 451941892713165,	23.399,	4.000, 4500,	10.446, 300.359,	83.433,	8.764, 245.888,	68.302, 67,	0,	0.000,	0.000,	0.000,	0	
00:41:23:871, 451942150673623,	23.511,	3.000, 4500,	9.491, 301.422,	83.728,	7.518, 246.729,	68.536, 66,	0,	0.000,	0.000,	0.000,	0.	
00:41:23:988, 451942419724813,	23.627,	9.000, 4500,	15.667, 303.251,	84.236,	14.020, 248.366,	68.991, 77,	0,	0.000,	0.000,	0.000,	(	
00:41:24:020, 451942492617479,	23.659,	12.000, 4500,	24.972, 304.041,	84.456,	22.944, 249.092,	69.192, 77,	0,	0.000,	0.000,	0.000,		
Total Elapsed Time (sec) = 23.659024												
Measured RDTSC Frequency (GHz) = 2.304												
Cumulative Processor Energy_0 (Joules) = 304.041077												
Cumulative Processor Energy_0 (mWh) = 84.455855												
Average Processor Power_0 (Watt) = 12.850956												
Cumulative IA Energy_0 (Joules) = 249.092285												
Cumulative IA Energy_0 (mWh) = 69.192301												
Average IA Power_0 (Watt) = 10.528426												
Cumulative DRAM Energy_0 (Joules) = 0.000000												
Cumulative DRAM Energy_0 (mWh) = 0.000000												
Average DRAM Power_0 (Watt) = 0.000000												
Cumulative GT Energy_0 (Joules) = 12.008179												
Cumulative GT Energy_0 (mWh) = 3.335605												
Average GT Power_0 (Watt) = 0.507552												

Şekil 6.10. Log dosyası son satırları

Yapılacak analizde her bir model 12 saat boyunca aynı şartlar altında ilk önce eğitilecektir. Her bir model için tekrarlanacak bu süreç eğitim sonrasında test veri setiyle çalıştırılacaktır. Her bir model için eğitim ve test sırasında güç tüketim hesaplamaları ayrı ayrı hesaplanacaktır. Modeller, modellerin eğitim sırasında harcadığı güç, test sırasında harcadığı güç ve bu modellerin doğruluk oranları loglanacak, en verimli ve doğruluğu yüksek olan optimum modelin bulunması amacıyla karşılaştırılacaktır.