SOFTWARE DESIGN DOCUMENT

(YAZILIM TASARIM DOKÜMANI)

Yapay Zeka ile Kredi Kartı Fraud Analizi

181180005 - Emre Can Ant

181180014 - Büşra Bayındır

C181112034 - Erencan Tezel

BM495 BİTİRME PROJESİ

Kelime Sayısı: 3095

Öğr. Gör. Dr. MUHAMMET ÜNAL

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

1. GENEL BAKIŞ

1.1. Belgeye Genel Bakış

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

1.2. Proje Kapsamı

Bu sistem kredi kartı işlemlerinin sahte olup olmadığını otomatik olarak tespit etmesi için yerel bir banka olan KuveytTürk iş birliği ile geliştirilmektedir. Normalde kullanıcıların şikayetlerine binaen sahte işlemler tespit edilirken bu sistem sayesinde daha işlemler yapılırken gerçek zamanlı olarak işlem analiz edilecek ve böylece kullanıcıların maddi manevi zarar görmesi engellenecektir. Sadece Asya-Pasifik bölgesinde kredi kartı fraud oranının 22 milyar dolara ulaştığı bu yıllarda, kullanıcıların da zarar görmesinin engellenmesi sayesinde aslında dolaylı olarak KuveytTürk'ün de marka değerini ve güvenilirliğini artırarak KuveytTürk'e fayda sağlayacaktır. Bununla beraber bu sahte işlem şikayetlerini incelemesi ve gerekli aksiyonların alınması amacıyla bulunan personel ihtiyacını ve giderini de minimuma indirecektir.

Daha spesifik olarak anlatmak gerekirse kredi kartı sahtekarlık analizi için hibrit bir yapay zeka modeli geliştirilecektir. Bu model ile Qt ile geliştirilmiş olan arayüz birleştirilecektir. Arayüz sayesinde iki farklı kullanım senaryosu olacaktır. İlk senaryoda banka yetkilisi geçmiş işlem dökümanlarını programa yükleyerek geçmişteki işlemlerde sahtekarlık olup olmadığını kontrol edecek ve kart sahibine geçmişteki işlem sebebiyle bilgi verecektir. Diğer bir senaryo ise gerçek zamanlıdır. Bankanın ödeme onaylama sistemiyle entegre olan bu kullanım senaryosunda bankanın ödeme sistemi yazmış olduğumuz uygulamaya yapılmakta olan ödemeye dair bilgileri post ettikten sonra sistemimiz halihazırdaki yapay zeka modeline bu girdiyi sokarak banka sistemine sonucu gönderecektir. Modelden çıkan sonuca göre aksiyon alınması amaçlanmaktadır. Böylece hem geçmişe yönelik sahte işlemlerin tespiti sağlanabilecek hem de anlık olarak gerçekleştirilmeye çalışılan sahte işlemler önlenecektir.

1.3. Amaç

Projenin amacı, kredi kartı işlemlerinin sahte olup olmamasının tespit edilebilmesidir. Bu analiz hem gerçek zamanlı hem de gerçek zamanlı olmayan biçimdedir. Gerçek zamanlı analizin amacı kredi kartıyla çevrimiçi işlem yapıldığı sırada gelen makine öğrenmesi modeline sokularak, işlem gerçekleşmeden önce ilgili mekanizmaya bilgi sağlanmasıdır. Bunun sayesinde modelden çıkan sonuca göre işlem onaylanmadan önce müdahale edilebilme şansı sağlanmış olur. Aktif bir müdahale şart değildir. Fraudulent işlem sonucu döndüğünde ekstra bir otantikasyon sağlanabilir.
Bu belgenin amacı, yazılım tasarımının; yapılandırılacak tasarım algısıyla ilerlemesine ve tasarım sürecinin nasıl gelişeceğine olanak sağlamak için yazılım ürününün tasarımının ne şekilde gerçekleşeceğinin bir tanımını sağlamaktır.

1.4. Hedef Kitle

Hedef kitle, bu belgede açıklanan uygulamayı geliştirecek yazılım ekibi ve bu personellerin danışmanlarıdır.

2. TANIMLAR VE KISALTMALAR

Bu belgede kullanılan kısaltmalar ve tanımlar aşağıdaki tabloda açıklanmıştır.

Terim	**Açıklama**
CSV	Virgülle ayrılan değerler [1] (Dosya Formatı).
Export	Dışarı aktar.
Framework	Çerçeve. Standart fonksiyonların hazır olarak sunulduğu ancak programcı tarafından bu fonksiyonlardan arzu edilen kısımların ek kodlarla istenildiği şekilde güncellenebildiği sistemlerdir [2].
Fraud	Dolandırıcılık, sahtecilik.
Fraudulent	Sahte
Grid View	Verilerin gösterilmesi, bir arada toplanması için kullanılan hazır araç.
JSON	JavaScript Object Notation.
MB	Mega byte.
MS	Milisaniye.
Tree View Table	Ağaç veri tablosu.
QT	Platform bağımsız arayüz tasarım çerçevesidir [3].
Script	Programcık, betik.
Viewpoint	Bakış açısı.

3. SDD İÇİN KAVRAMSAL MODEL TANIMI

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

3.1. Bağlam İçerisinde Yazılım Tasarımı

Projenin amacı KuveytTürk bankasına bir arayüz ve server sistemi sağlayarak anlık yapılmak istenen bankacılık işlemlerinin dolandırıcılık analizini yapmak veya 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. Yapay zeka modelinde kullanılacak algoritma ise tek bir algoritma yerine kNN, random forest gibi çeşitli makine öğrenmesi sınıflandırma algoritmaların hibritlenmesiyle oluşturulacaktır.

3.2. Yaşam Döngüsü İçerisinde Yazılım Tasarım Açıklamaları

3.2.1. SDD hazırlığı üzerindeki etkileri

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 etkileri

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.

3.2.3. Tasarım doğrulaması ve onaylamada tasarımının rolü

Doğrulama (verification) ve geçerli kılma (validation), test senaryolarının hazırlanmasından sonra test edilecektir. Bu caselere karşı gerekliliklerin sağlanıp sağlanmadığı kontrol edilecektir.

4. TASARIM TANIMI BİLGİ İÇERİĞİ

4.1. Giriş Yazılım

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

4.2. SDD Kimliği

Doğrulama ve geçerli kılma için yapılan testlerden sonra, Kredi kartı analiz sistemi gerçek dünya ortamına dağıtılacaktır. Sözlük "TANIMLAR VE KISALTMALAR" bölümünde bulunabilir.

4.3. Tasarım Paydaşları ve Endişeleri

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

4.4. Tasarım Görünümleri

Tasarım görünümleri, tasarım paydaşlarına tasarım ayrıntılarına belirli bir perspektiften odaklanma ve ilgili gereksinimleri karşılama konusunda yardımcı olur.Bu belgede bağlam, mantıksal, arayüz ve etkileşim bunlara karşılık gelen bakış açıları olarak açıklanacaktır. Bazı görünümler için, açıklama amacıyla ilgili UML diyagramları gösterilecektir.

4.5. Tasarım Viewpointleri

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

4.5.1. Bağlam Viewpoint

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.

4.5.2. Mantıksal Viewpoint

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.

4.5.3. Arayüz Viewpoint

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

4.5.4. Etkileşim Viewpoint

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

4.6. Tasarım Elementleri

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.

4.7. Tasarım Mantığı

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.

4.8. Tasarım Dili

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

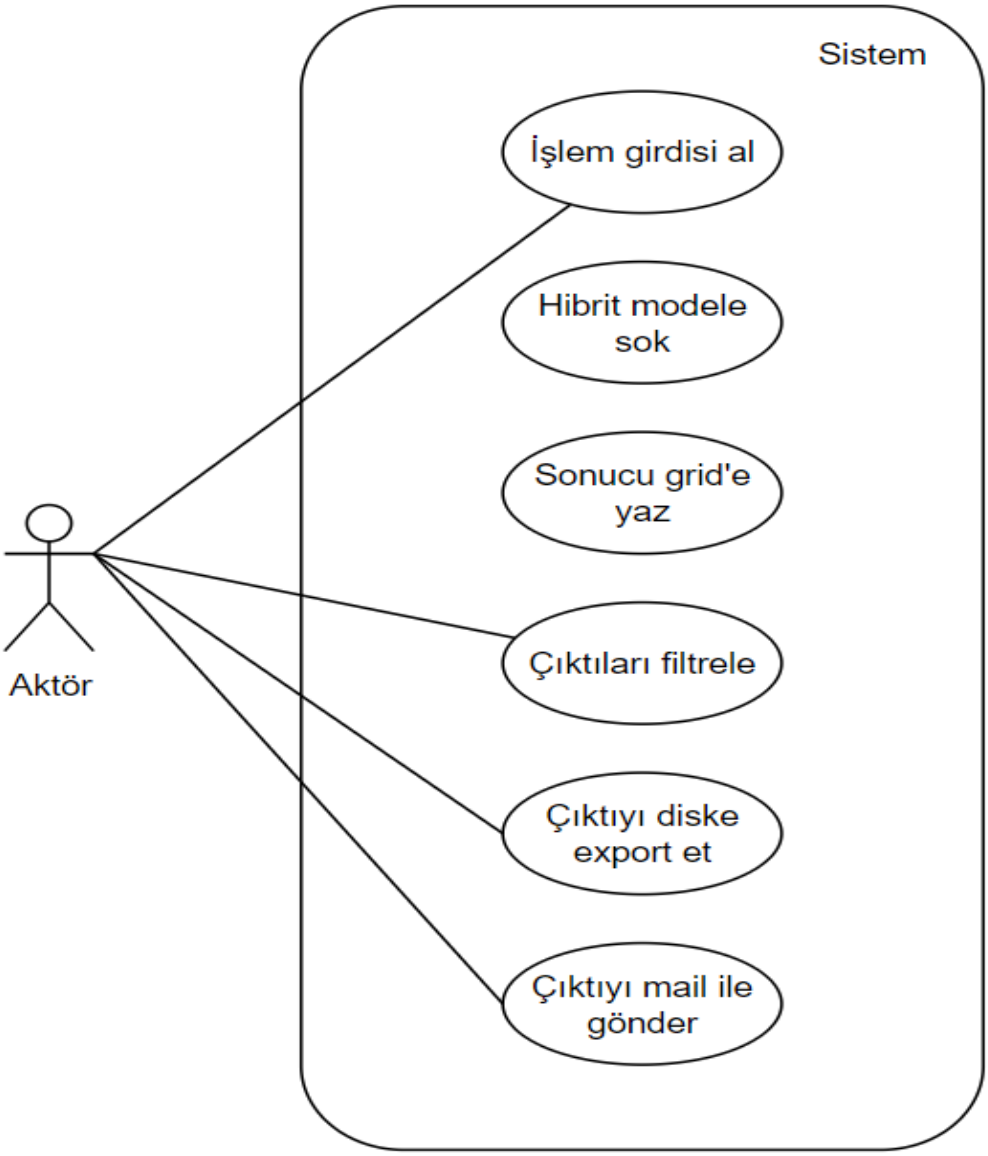
5. TASARIM BAKIŞ AÇILARI

5.1. Giriş

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.

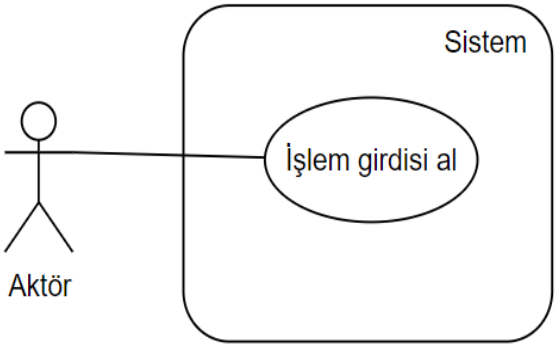
5.2. Bağlam Viewpoint

Sistemde sadece bir tür kullanıcı vardır. Sistem, hibrit 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. Sistem Use Case Diyagramı

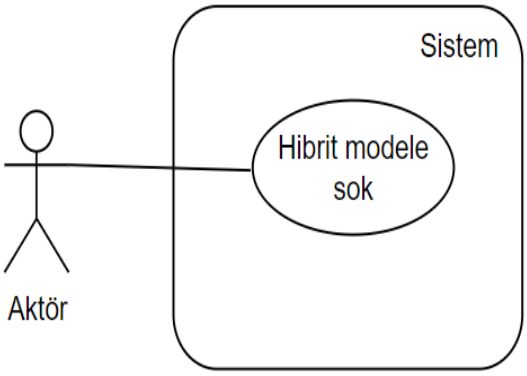
5.2.1. İşlem girdisi al



Şekil 5.2. İşlem girdisi al Use Case

Sistemin işlemleri sınıflandırabilmesi için sisteme kredi kartı işlem verisi/verileri verilmelidir. SRS Dokümanında belirtilen Mod 1 için csv formatında dosya okuması işlemi yapılır. Mod 2, çevrimiçi istemcili gerçek zamanlı tespit için JSON formatında veri alınır.

5.2.2. Hibrit modele sok



Şekil 5.3. Hibrit modele sok Use Case

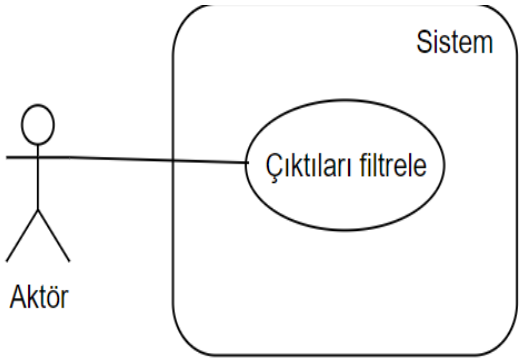
Aktör ile gelen işlem verileri hibrit sınıflandırma modeline sonuçları üretmek amacıyla sokulur. Mod 1 için kullanıcının butona basması beklenir. Hibrit modele 6. bölümde değinilmiştir, 7.4. bölümde tanımı yapılmıştır.

5.2.3. Sonucu gride yaz

Şekil 5.4. Sonucu data grid'e yaz Use Case

Hibrit modele sokulduktan sonra elde edilen sonuçlar kullanıcının görebilmesi için ağaç yapısındaki data gride aktarılır.

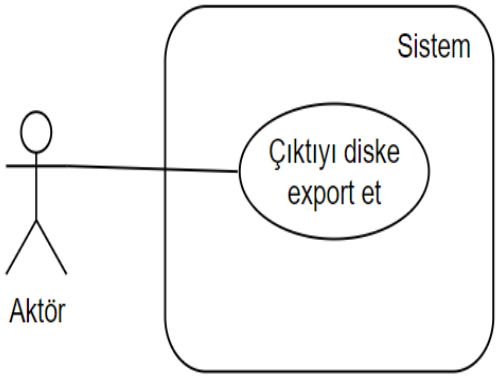
5.2.4. Çıktıları filtrele



Şekil 5.5. Çıktıları filtrele Use Case

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.

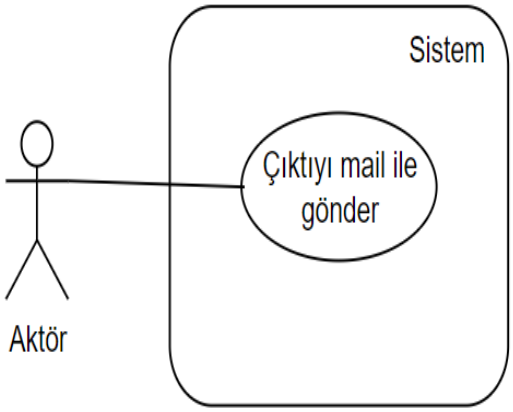
5.2.5. Çıktıyı diske export et



Şekil 5.6. Çıktıyı diske export et Use Case

Filtrelenen veya tüm sonuçlar aktör tarafından belirlenen konuma export edilir. "Dışa Aktar" butonuna basılması beklenir.

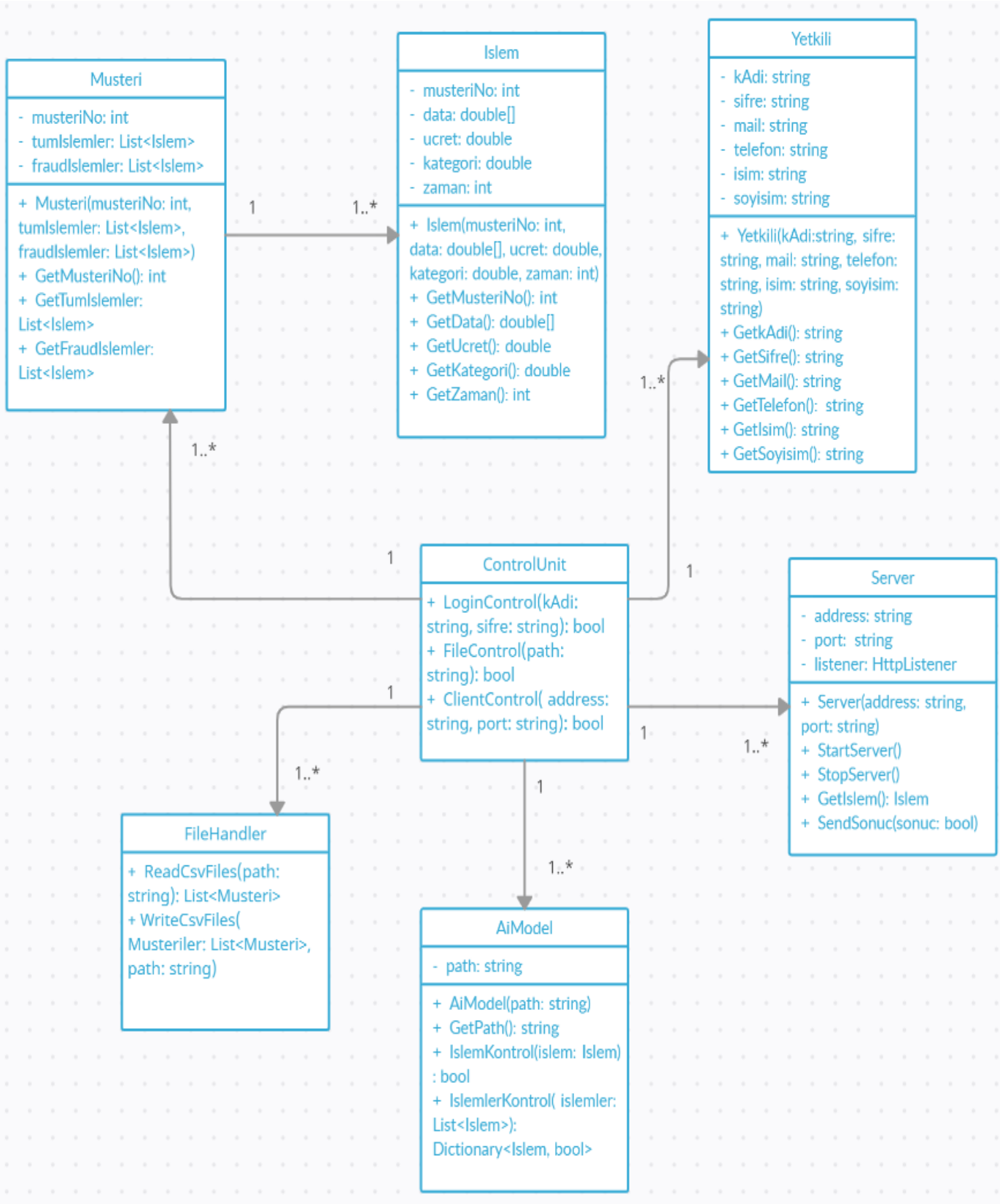
5.2.6. Çıktıyı mail ile gönder



Şekil 5.7. Çıktıyı e-mail ile gönder Use Case

Filtrelenen veya tüm sonuçlar aktör tarafından belirlenen mail adresine gönderilir. "Mail ile Gönder" butonuna basılması beklenir.

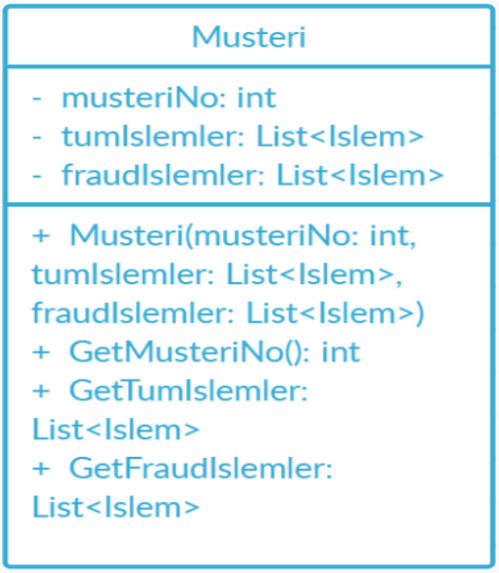
5.3. Mantıksal Viewpoint



Şekil 5.8. Sistemin UML Class Diyagramı

5.3.1. Müşteri Sınıfı

Bu sınıf KuveytTürk'de hesabı bulunan kişilerin firma tarafından verilen bilgilerini ve işlemlerini içermektedir.



Şekil 5.9. 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 5.10. 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 5.11. 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.

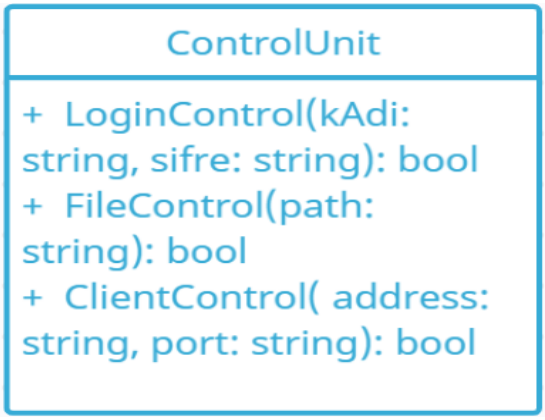
5.3.3.1.6. soyisim: Yetkilinin log işlemlerinde tutulan soyismini ifade eden string değerdir.

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.



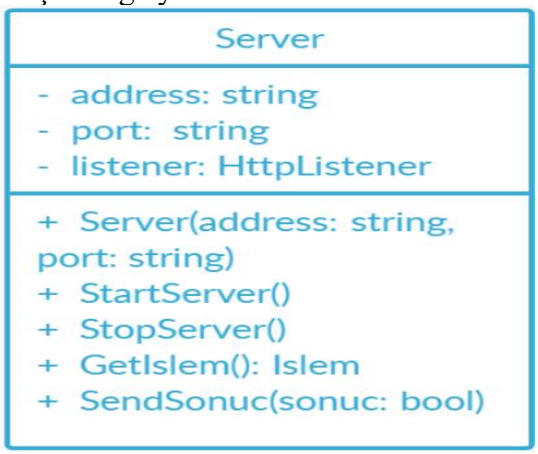
Şekil 5.12. ControlUniti sınıfı

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.

5.3.5. Server Sınıfı

Http üzerinden işlem verilerinin akışını sağlayan sınıftır.



Şekil 5.13. Server sınıfı

5.3.5.1. Değişkenler

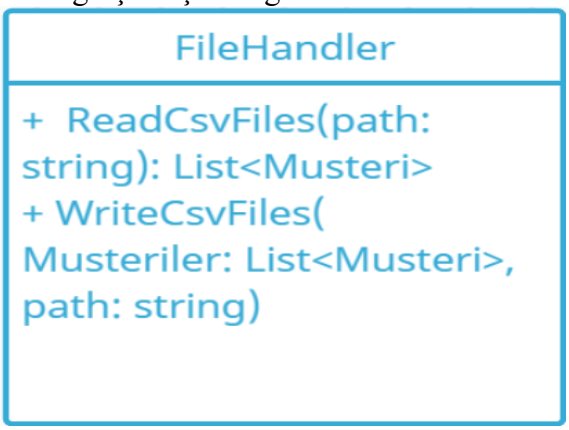
5.3.5.1.1. address: Veri akışının hangi adres üzerinden sağlanacağını belirten string değerdir.
5.3.5.1.2. port: Veri akışının hangi port üzerinden sağlanacağını belirten string değerdir.
5.3.5.1.3. listener: Veri akışının sağlandığı HTTP protokolü dinleyicisidir.

5.3.5.2. Metodlar

5.3.5.2.1. Server(): Server sınıfının oluşturucusudur. Gerekli kontrolleri ve ilk değer atamalarını gerçekleştirmektedir.
5.3.5.2.2. StartServer(): Http protokolü üzerinden veri akış işlemini başlatan fonksiyondur.
5.3.5.2.3. StopServer(): Http protokolü üzerinden veri akış işlemini durduran fonksiyondur.
5.3.5.2.4. GetIslem(): Veri geldiğinde tetiklenen fonksiyondur.
5.3.5.2.5. SendSonuc(): Gelen işlem verisinin sonucunu http protokolü aracılığıyla ileten fonksiyondur.

5.3.6. FileHandler Sınıfı

Dosya okuma yazma gibi işlemlerin gerçekleştirildiği sınıftır.



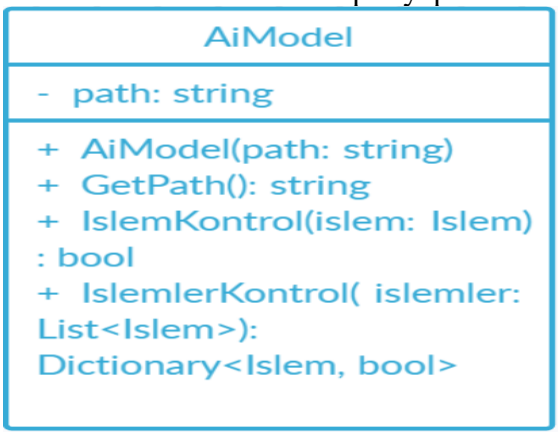
Şekil 5.14. FileHandler sınıfı

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. 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.

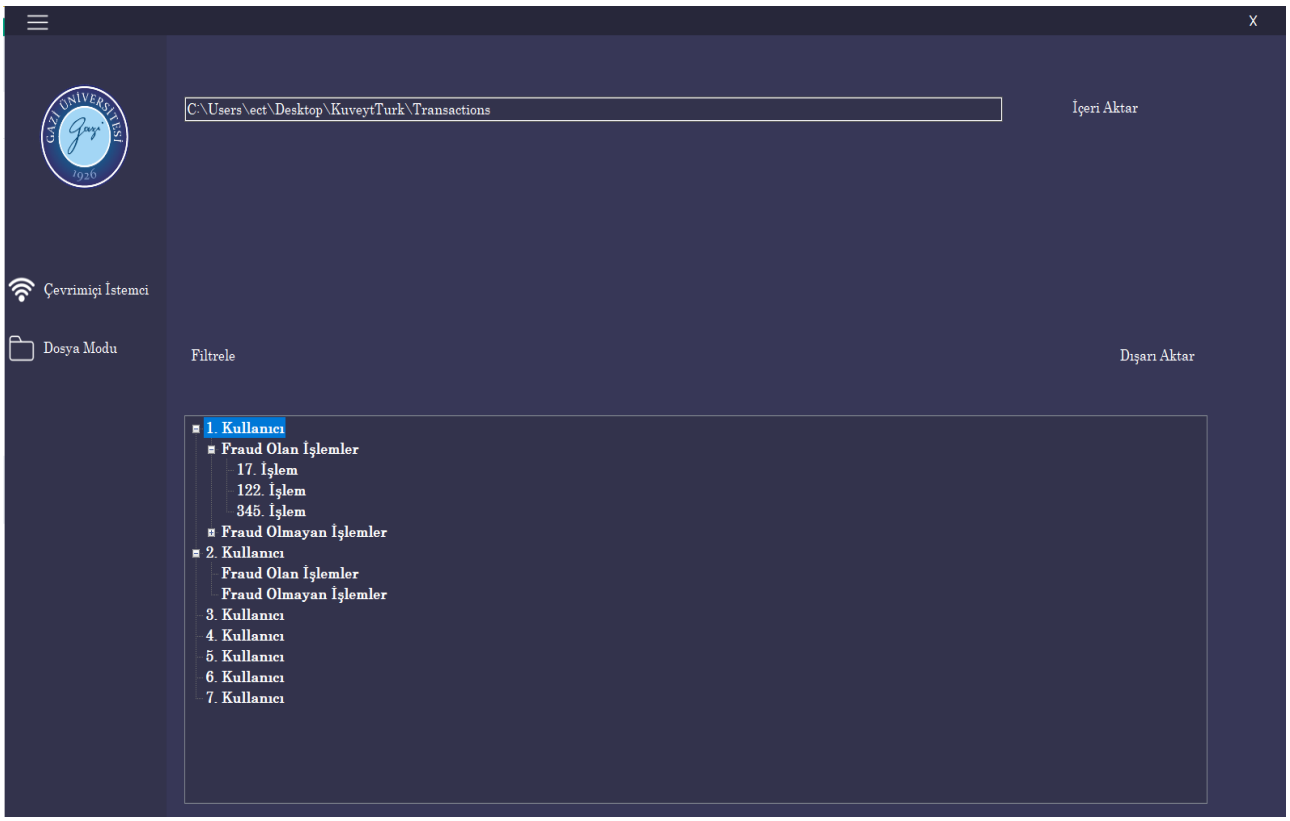
5.3.7.2.2. GetPath(): path değişkenini dönen fonksiyondur..

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.



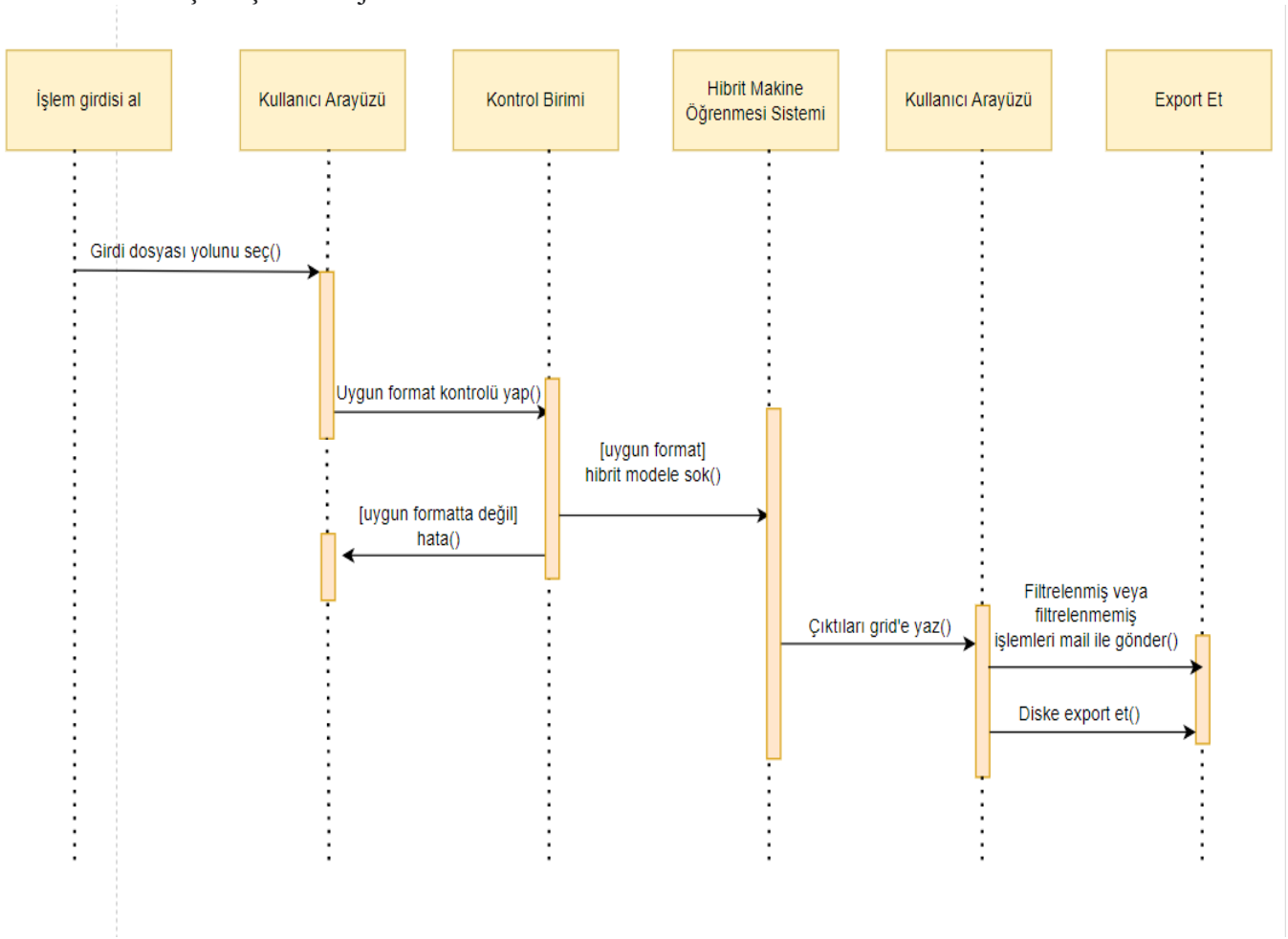
Şekil 5.16. Sistem Dosya Modu arayüzü

Kullanıcı uygulamayı çalıştırdığında çevrimiçi istemci modu ve dosya modu olmak üzere 2 farklı seçenekle karşılaşacaktır. 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, filtrrelenen işlemler export edilmesi için dışarı aktar butonu yerleştirilmiştir.

Çevrimiçi istemci modunun tasarımı da dosya modu ile benzerdir. Alınan girdi json formatında, get post metodları ile alınır. Fraudulent işlemler filtrelenebilir. Filtrelenenler export edilebilir. Arayüz QT ile tasarlanmıştır.

5.5. Etkileşim Viewpoint

Bu bölümde, sistemin temel işlevleri sekans diyagramı yardımı ile verilmektedir. Ayrıca varlıklar arasındaki etkileşim için stratejiler tanımlar.



Şekil 5.17. 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 hibrit modele sokulur. Sonuçlar data grid'e yazılır. İsteğe bağlı olarak filtrelenebilir çıktılar diske veya kullanıcının girdiği bir e-mail'e export edilir.

6. MAKİNE ÖĞRENMESİ MODELİ VE VERİ SETİ ÖN İŞLEMİ

6.1. Yapılan Ön İşlemler

Makine öğrenmesi sürecinden önce model eğitimi için veri seti ön işlemlere tabii tutulur. Eğitim için kullanılan veri setinin 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=30000)
```

Şekil 6.2. Under-sampling

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

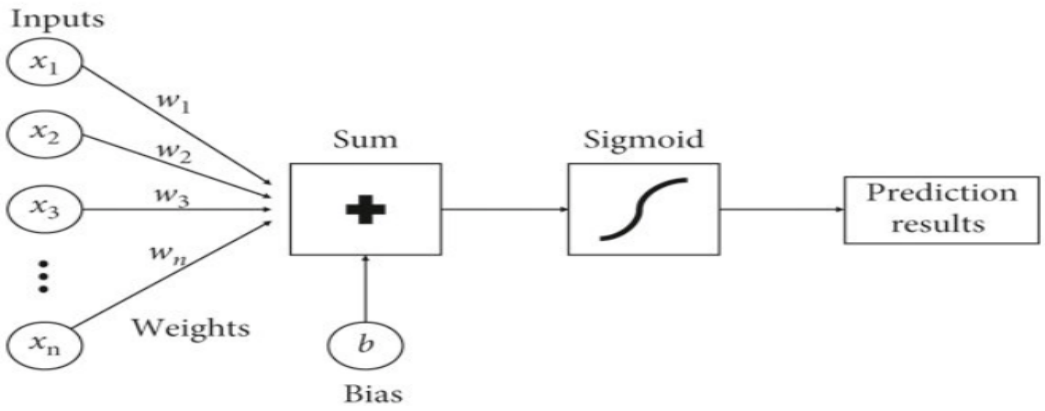
```
new_dataset=pd.concat([legit_sample,fraud],axis=0)
```

Şekil 6.3. Yeni veri setinin elde edilmesi

Yeni veri setinin %70'i train için geri kalan %30 ise test set olacak şekilde bölünmüştür.

6.2. Kullanılan Model

Projenin şu anki halinde hibrit sisteme geçilmemiştir. Logistic Regression sınıflandırma algoritması kullanılmıştır. Yeniden ürettiğimiz veri setiyle eğitilen logistic regression makine öğrenmesi modeli bizim şu anki hibrit modelimizdir.
Aşağıdaki görselde kullanılan logistic regression algoritmasının flowchartı sunulmuştur.



Şekil 6.4. Logistic regression flowchart [4]

7. NOTLAR

Bu bölüm bu dokümanın anlaşılmasına yardımcı olan her türlü genel bilgiyi içerir.

7.1. Mod 1

SRS dokümanında açıklanmıştır. Sistemin dosya okumalı modudur.

7.2. Mod 2

SRS dokümanında açıklanmıştır. Sistemin çevrimiçi istemcili modudur.

7.3. Validation ve Verification

İngilizce kavramların tam olarak Türkçe karşılığı bulunamamış olup, verification doğrulama, validation ise onaylama, geçerli kılma olarak çevrilmiştir.

7.4. Hibrit Model Tanımı

Normal veya fraudulent işlem sonucunu veren makine öğrenmesi modelimizin adı hibrit modeldir.

8. KAYNAKÇA

[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.