YILDIZ TEKNİK ÜNİVERSİTESİ
ENDÜSTRİ MÜHENDİSLİĞİ BÖLÜMÜ

ENDÜSTRİ MÜHENDİSLİĞİ TASARIM 2

FRAUD DETECTİON

21061701 Okan DEMİRKAYA
19061068 Galip ŞAHİN
20061044 Büşra TARHAN

Danışman
Doç. Dr. Selçuk ALP

İSTANBUL
2024

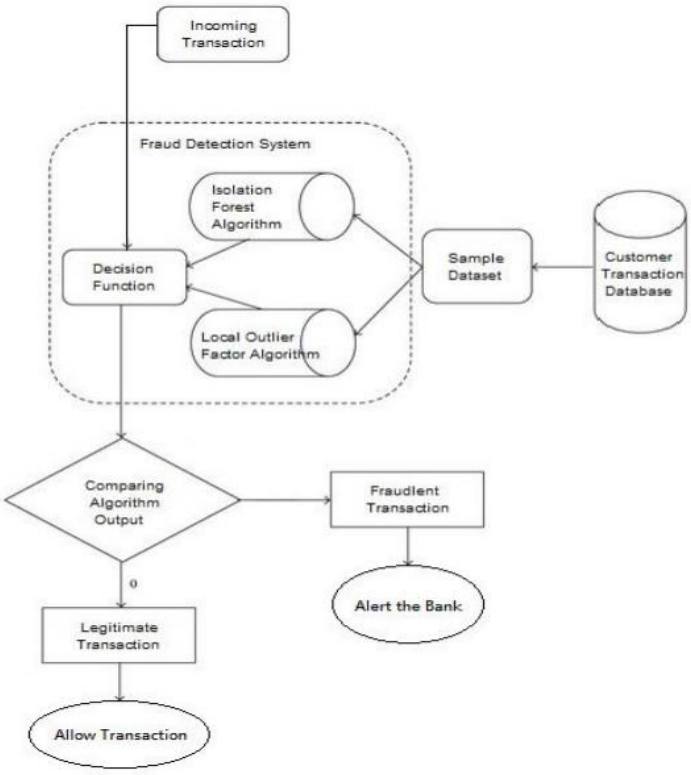
İçindekiler

1.Fraud Nedir

'Fraud' kredi kartı işlemlerinde, hesap sahibi dışındaki bir kişi tarafından izinsiz ve istenmeyen bir şekilde hesabın kullanılmasıdır. Bu tür kötüye kullanımları durdurmak ve benzer olaylara karşı korunmak için gerekli önlemler alınabilir. Başka bir deyişle, Kredi Kartı Dolandırıcılığı, bir kişinin başkasının kredi kartını kişisel nedenlerle kullanması durumudur, ancak kart sahibi ve kartı veren otoriteler bu durumun farkında değildir.

Dolandırıcılık tespiti, nüfusun faaliyetlerini izleyerek, dolandırıcılık, ihlal ve temerrüt gibi istenmeyen davranışları tahmin etmek, algılamak veya önlemekle ilgilenir. Bu, çözümünün otomatize edilebileceği makine öğrenimi ve veri bilimi gibi toplulukların dikkatini çeken çok önemli bir sorundur. Bu sorun, öğrenme perspektifinden özellikle zordur, çünkü sınıf dengesizliği gibi çeşitli faktörlere sahiptir. Geçerli işlemlerin sayısı sahte işlemleri belirgin bir şekilde aşar. Ayrıca, işlem desenleri genellikle zaman içinde istatistiksel özelliklerini değiştirir.

Ancak, bir gerçek dünya dolandırıcılık tespit sistemini uygulamanın karşılaştığı tek zorluklar bunlar değildir. Gerçek dünya örneklerinde, ödeme taleplerinin büyük akışı hızla otomatik araçlar tarafından taranır ve hangi işlemlerin yetkilendirileceğini belirleyen araçlar kullanılır. Makine öğrenimi algoritmaları, tüm yetkilendirilmiş işlemleri analiz etmek ve şüpheli olanları rapor etmek için kullanılır. Bu raporlar, işlemi gerçek veya sahte olduğunu doğrulamak için kart sahipleri ile iletişime geçen uzmanlar tarafından incelenir. Araştırmacılar, otomatik sistem için geri bildirim sağlarlar ve zaman içinde dolandırıcılık tespit performansını iyileştirmek için algoritmayı eğitmek ve güncellemek için kullanılır.



1.1. Dolandırıcılığın Tespit Edilmesinin Önemi

Nüfus talebi, ürün çeşitliliği ve kabul edilen ödeme yöntemlerindeki artışla birlikte, ödemeleri güvence altına almak ve müşteri hesaplarını korumak her geçen gün daha da önemli hale gelmektedir. Şirketler müşteri beklentilerini artırırken ve müşteri deneyimlerini iyileştirirken, güvenliği göz ardı etmeden bu beklentilere karşılık vermeleri gerekmektedir. Experian'ın 2022 Küresel Kimlik ve Dolandırıcılık Raporu'na göre, şirketlerin %70'i dolandırıcılık önlemine olan ilgilerinin geçen yıldan bu yana arttığını belirtmektedir. 2018'de, Tek Avro Ödeme Alanı'nda (SEPA) ihraç edilen ve dünya genelinde alınan kartlarla gerçekleştirilen sahtekârlık işlemlerinin toplam değeri 1,8 milyar Euro idi. Kartla yapılan ödemelerde dolandırıcılık toplamı, kart kullanılmadan yapılan ödemeler (internet, posta veya telefon aracılığıyla yapılan ödemeler) tarafından oluşturulan 1,43 milyar Euro'ya ulaştı. Bu tür işlemleri önleyememenin sonuçları, sadece finansal sonuçlar değil, aynı zamanda itibar, sadakat ve uzun süre devam edecek diğer marka ile ilgili sorunları da beraberinde getirebilir.

1.2. Fraud Detection Kullanım Alanları

Dolandırıcılık tespit yöntemleri, suçluların dolandırıcı stratejilerine adapte olmalarına karşı sürekli olarak geliştirilmektedir. Bu dolandırıcılıklar şu şekilde sınıflandırılabilir:

- Kredi Kartı Dolandırıcılıkları: Çevrimiçi ve Çevrimdışı
- Kart Hırsızlığı
- Hesap Batması
- Cihaz İhlali
- Uygulama Dolandırıcılığı
- Sahte Kart
- Telekomünikasyon Dolandırıcılığı

1.3. Fraud Tespitinde Kullanılan Yaklaşımlar

Bu tür dolandırıcılıkların tespitine yönelik şu anda kullanılan bazı yaklaşımlar şunlardır:

- Yapay Sinir Ağları
- Bulanık Mantık
- Genetik Algoritmalar

• Lojistik Regresyon

• Karar Ağaçları

• Destek Vektör Makineleri

• Bayesian Ağları

• Gizli Markov Modeli

• K-En Yakın Komşu

1.4.Fraud Tespit Sistemleri

Dolandırıcılık saldırılarının tam zamanlaması öngörülemez. Dolandırıcılar genellikle saldırılarını en yüksek başarı şansına sahip olduklarına inandıkları anlarda gerçekleştirirler. Potansiyel olarak sahtekârlık tehdidi taşıyan her işlem doğrulama sürecinden geçmelidir. Bu amaçla, dolandırıcılık tespit sistemleri kullanılarak bir işlemin onaylanıp onaylanmamasına veya ileri bir inceleme için belirli bir süre tutulmasına karar verilir. Şekil 1.1'de bir örnek dolandırıcılık olayı gösterilmiştir.

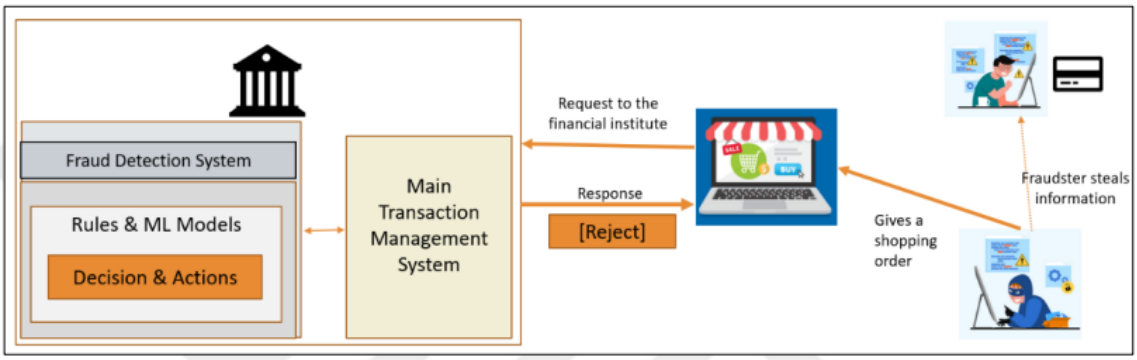


Figure 1.1: A sample fraud event and Fraud Detection System.

Bu sistemler, işlemler üzerinde gerçek zamanlı kontroller gerçekleştirir. İşlem onaylanmadan önce, işlem kaynak kanalı, işlem verilerini içeren bir giriş ile dolandırıcılık tespit sistemine bir kontrol talebi gönderir. Dolandırıcılık tespit sistemi, bu girişi işler, işleme ilişkin kart bilgileri, geçmiş veriler, müşteri ve konum bilgileri, cihaz özellikleri ve daha fazlası dahil olmak üzere işleme yönelik işlemleri zenginleştirir. Bu zenginleştirilmiş verilere dayanarak, işlemi sahte veya normal olarak sınıflandırmak için çeşitli kural setleri ve algoritmalar çalıştırılır. Elde edilen sonuç, işlem kaynak sistemine, işlemin devam edilip edilmemesini, reddedilip edilmemesini veya belirli özel senaryolarda işlemin belirli bir süre için geçici olarak bekletilip bekletilmemesini belirlemek için kritik bilgi olarak hizmet verir. "Bekleme" sonucu, bir işlemin detaylı bir inceleme için bekletilmesi gerektiğinde üretilir. Bu durumlarda,

dolandırıcılık analistleri, işlemin yasallığını doğrulamak ve sahte olmadığından emin olmak için inceleme yapar.

2. Literatür İncelemesi

Dolandırıcılık, finansal veya kişisel avantaj elde etmeyi amaçlayan yasa dışı veya suç teşkil eden bir aldatmaca olarak hareket eder. Bu, yasa, kural veya politikaya karşı bir kastı olan, yetkisiz finansal avantaj elde etmeyi amaçlayan kasıtlı bir eylemdir.

Bu alandaki anormallik veya dolandırıcılık tespitiyle ilgili birçok literatür zaten yayımlanmış ve kamuya açıktır. Clifton Phua ve ekibinin gerçekleştirdiği kapsamlı bir anket, bu alanda kullanılan tekniklerin veri madenciliği uygulamalarını, otomatik dolandırıcılık tespitini ve karşıt tespiti içerdiğini ortaya koymuştur. Başka bir makalede, Suman, Hisar HCE'de GJUS&T'de Araştırma Görevlisi, kredi kartı dolandırıcılığı için Denetimli ve Denetimsiz Öğrenme gibi teknikleri sunmuştur. Bu yöntemler ve algoritmalar bazı alanlarda beklenmeyen bir başarı elde etmiş olsa da, dolandırıcılık tespiti için kalıcı ve tutarlı bir çözüm sunamamıştır.

Wen-Fang YU ve Na Wang tarafından sunulan benzer bir araştırma alanında, kredi kartı işlem veri setinin emülasyon deneyinde sahte işlemleri doğru bir şekilde tahmin etmek için Aykırı Değer madenciliği, Aykırı Değer tespit madenciliği ve Mesafe toplamı algoritmalarını kullanmışlardır. Aykırı Değer madenciliği, temelde mali ve internet alanlarında kullanılan bir veri madenciliği alanıdır. Ana sistemden ayrılmış nesneleri tespit etme ile ilgilenir, yani gerçek olmayan işlemleri. Müşteri davranışlarına ait özellikleri almışlar ve bu özelliklerin değerine dayanarak gözlemlenen değer ile önceden belirlenmiş değer arasındaki mesafeyi hesaplamışlardır. Hibrit veri madenciliği/karmaşık ağ sınıflandırma algoritması gibi sıradışı teknikler, gerçek bir kart işlem veri setinde yasa dışı örnekleri algılamak için etkili olmuş ve genellikle orta büyüklükteki çevrimiçi işlemlerde kanıtlanmıştır.

Tamamen yeni bir bakış açısından ilerleme çabaları da olmuştur. Sahte bir işlem durumunda uyarı-geri bildirim etkileşimini geliştirmeye yönelik çabalar bulunmaktadır. Sahte bir işlem durumunda yetkilendirilmiş sistem uyarılır ve devam eden işlemi reddetmek için bir geri bildirim gönderilir.

Bu alanda yeni bir ışık tutan yaklaşımlardan biri olan Yapay Genetik Algoritma, dolandırıcılığı farklı bir yönden ele aldı. Sahte işlemleri tespit etme konusunda doğru olduğunu kanıtladı ve yanlış uyarı sayısını azaltmada etkili oldu. Ancak değişken yanlış sınıflandırma maliyeti sorunuyla birlikte geldi.

3.Fraud Detection Alanında Kullanılan Makine Öğrenimi Metotları

3.1.Lojistik Regresyon

Lojistik regresyon, yapay zeka ve makine öğrenimi (AI/ML) alanında önemli bir tekniktir. ML modelleri, insan müdahalesi olmadan karmaşık veri işleme görevlerini gerçekleştirebilen yazılım programlarıdır. Lojistik regresyon kullanılarak oluşturulan ML modelleri, kuruluşların iş verilerinden eyleme dönüştürülebilir öngörüler elde etmelerine yardımcı olur. Bu bilgiler, operasyonel maliyetleri azaltmak, verimliliği artırmak ve daha hızlı ölçeklendirmek amacıyla tahmine dayalı analiz için kullanılabilir. Örneğin, işletmeler, çalışanların elde tutulmasını artıran veya daha kârlı ürün tasarımına yol açan kalıpları ortaya çıkarabilir.

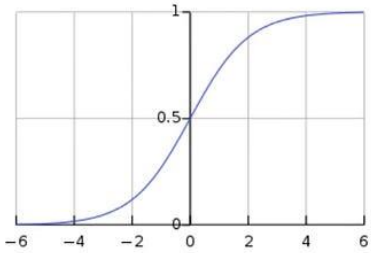
Bir ikili bağımlı değişkeni modellemek için lojistik fonksiyonu kullanan istatistiksel bir modeldir. Bu model, ikili sınıflandırma sorunu olasılığı olduğu durumlarda başlıca kullanılır. Doğrusal olarak ayrılabilir sınıflar üzerinde iyi çalışır. Odds Ratio, logit fonksiyonunu tanımlayabileceğimiz bir kavramdır. Bir olayın gerçekleşme olasılığıdır.

Odds Ratio = p / (1 —p)

Burada, p pozitif olayın olasılığını temsil eder. Logit fonksiyonu, Odds Ratio nun logaritmasıdır. Girişi [0,1] aralığında alır ve bunları gerçek sayı aralığındaki değerlere dönüştürür.

$$\hat{y}_i = P(Y = 1 \mid X = x)$$

$$\frac{1}{1 + e^{-(z)}}$$



$$z = b + w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_px_p$$

3.1.1.Lojistik Regresyon Ne Zaman Kullanılır?

Lojistik regresyon, girdinin doğrusal bir sınırla “iki bölgeye” ayrılması gerektiğinde kullanılır. Veri noktaları gösterildiği gibi doğrusal bir çizgi kullanılarak ayrılır.

Kategori sayısına bağlı olarak lojistik regresyon şu şekilde sınıflandırılabilir:

- **binom:** Hedef değişken sadece 2 olası tipe sahip olabilir: “0” veya “1”; bunlar “kazanma “ya karşı “kaybetme “yi, “geçme “ye karşı “başarısız olma “yı, “ölü “ye karşı “canlı “yı vb. temsil edebilir.
- **multinomial:** Hedef değişken, “hastalık A” vs “hastalık B” vs “hastalık C” gibi sıralanmamış (yani türlerin niceliksel önemi yoktur) 3 veya daha fazla olası türe sahip olabilir.
- **ordinal:** sıralı kategorilere sahip hedef değişkenlerle ilgilenir. Örneğin, bir test puanı “çok zayıf”, “zayıf”, “iyi”, “çok iyi” olarak kategorize edilebilir. Burada her kategoriye 0, 1, 2, 3 gibi bir puan verilebilir.

Lojistik regresyonun en basit şeklini, yani binom lojistik regresyonunu inceleyecek olursak, bir sınıflandırma problemini çözerken, yani y değişkeni yalnızca iki değer aldığında kullanılabilir. Böyle bir değişkenin “ikili” veya “dikotom” değişken olduğu söylenir. “İkili” temel olarak evet/hayır, kusurlu/kusursuz, başarılı/başarısız gibi iki kategori anlamına gelir. “İkili” ise 0’ları ve 1’leri ifade eder.

Farklı lojistik regresyonların doğru kullanımı, verilerle ilgili sorunları çözmek için bir zorunluluktur ve veri bilimi sertifikası alırken öğrendiğiniz istatistiksel beceriler, seçenekleri daraltmak için kullanışlıdır.

3.1.2.Makine Öğreniminde Lojistik Regresyon Uygulamaları

Lojistik regresyon, makine öğreniminde aşağıdakiler de dahil olmak üzere geniş bir uygulama alanına sahiptir:

1. Müşteri Kaybını Tahmin Etme

Müşteri kaybı birçok sektörde yaygın bir sorundur ve lojistik regresyon hangi müşterilerin ayrılma olasılığının yüksek olduğunu tahmin etmek için kullanılabilir. Demografik bilgiler, satın alma geçmişi ve müşteri hizmetleri etkileşimleri gibi müşteri verilerini analiz ederek lojistik regresyon, müşteri kaybıyla en güçlü şekilde ilişkili faktörleri belirleyebilir ve hangi müşterilerin ayrılma riskinin en yüksek olduğunu tahmin edebilir.

2. Kredi Puanlaması

Lojistik regresyon, bir kredinin temerrüde düşme olasılığını tahmin etmek için kredi puanlamasında da yaygın olarak kullanılır. Lojistik regresyon, kredi geçmişini, geliri ve diğer faktörleri analiz ederek, temerrütle en güçlü şekilde ilişkili olan faktörleri belirleyebilir ve hangi borçluların temerrüde düşme riskinin en yüksek olduğunu tahmin edebilir.

3. Tıbbi Teşhis

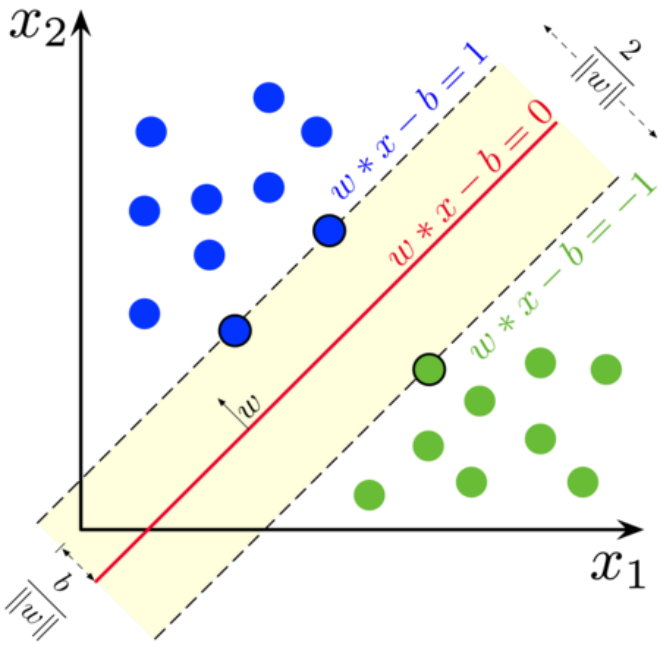
Lojistik regresyon, bir hastanın semptomlarına ve tıbbi geçmişine dayanarak belirli bir hastalığa sahip olma olasılığını tahmin etmek için tıbbi tanıda kullanılabilir. Lojistik regresyon, hasta verilerini analiz ederek hastalıkla en güçlü şekilde ilişkili olan faktörleri belirleyebilir ve hangi hastaların hastalığa yakalanma riskinin en yüksek olduğunu tahmin edebilir.

4. Dolandırıcılık Tespiti

Lojistik regresyon, işlem tutarı, konum ve zaman gibi çeşitli faktörlere dayalı olarak bir işlemin hileli olma olasılığını tahmin etmek için dolandırıcılık tespitinde kullanılabilir. Lojistik regresyon, işlem verilerini analiz ederek dolandırıcılıkla en güçlü şekilde ilişkili faktörleri belirleyebilir ve hangi işlemlerin dolandırıcılık riskinin en yüksek olduğunu tahmin edebilir.

3.2.Destek Vektör Makineleri

Support Vector Machines (SVM), bir makine öğrenimi algoritmasıdır ve hem sınıflandırma hem de regresyon problemleri için kullanılabilir. Temelde, veri noktalarını iki veya daha fazla sınıfa ayırmak veya bir çizgi üzerine sığdırmak için kullanılır. SVM'nin ana amacı, veri noktalarını sınıflandırmak için en iyi ayırt edici çizgiyi (veya düzlemi) bulmaktır.



SVM, özellikle yüksek boyutlu veri setlerinde etkili bir şekilde çalışabilir ve özellikle lineer ve non-lineer sınıflandırma problemlerine uygundur. Temel prensiplerinden biri, veri noktalarını sınıflandırmak için en geniş marjı sağlamaya çalışmaktır. Marj, sınıflar arasındaki boşluğu temsil eder ve bu boşluğun en geniş olması, genelleme yeteneğini artırabilir.

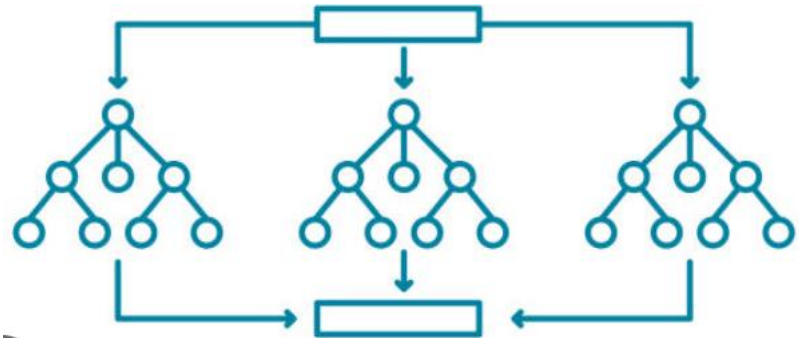
SVM'nin bir diğer önemli özelliği, çekirdek fonksiyonları kullanarak verileri yüksek boyutlu uzaylara taşıyarak ve orada sınıflandırmayı gerçekleştirerek non-lineer sınıflandırma problemleriyle başa çıkabilmesidir.

SVM'nin avantajlarından biri, modelin aşırı öğrenmeye karşı direnç göstermesi ve etkili bir şekilde genelleme yapabilmesidir. Ancak, büyük veri setleriyle çalışırken hesaplama gücü açısından talep edici olabilir.

3.3.Karar Ağacı

Bir karar ağacı, belirli bir problem için tüm potansiyel çözümleri haritalayan akış şeması benzeri bir diyagramdır. Genellikle kuruluşlar, bir dizi kararın tüm olası sonuçlarını karşılaştırarak en uygun hareket tarzını belirlemeye yardımcı olması amacıyla kullanılır. Örneğin, bir şirketin genel merkezini hangi şehre taşıyacağına veya bir uydu ofisi açıp açmamaya karar vermesine yardımcı olmak için bir karar ağacı kullanılabilir. Karar ağaçları, tahmine dayalı modeller oluşturmak için makine öğreniminde de yaygın olarak kullanılan araçlardır. Bu tür karar ağaçları, bir müşterinin önceki satın alma geçmişine dayanarak bir ürünü satın alıp almayacağı gibi tahminler yapmak için kullanılabilir.

Bu, en yaygın kullanılan öngörü modelleme yaklaşımlarından biridir. Modelin adından da anlaşılacağı gibi, bu model bir ağaç benzeri yapıda oluşturulur. Bu model, çok boyutlu bir analiz durumunda kullanılabilir, burada birden çok sınıf bulunmaktadır. Geçmiş veri, aynı zamanda geçmiş vektör olarak bilinen, sağlanan girişe dayalı olarak çıktının değerini tahmin etmek için kullanılır. Bir ağaçta birden çok düğüm bulunur ve her düğüm, bir vektöre karşılık gelir. Ağaç, her biri bir olası sonucu veya çıktıyı temsil eden bir yaprak düğümünde sona erer.



3.3.1.Karar Ağaçlarının Uygulamaları

İş Dünyası: Karar ağaçları iş dünyasında finans, pazarlama, operasyon ve strateji gibi farklı alanlarda kullanılmaktadır.

Sağlık Hizmetleri: Karar ağaçları hastalıkları tahmin etmek, tedavileri değerlendirmek ve hasta bakımını iyileştirmek için kullanılır.

Eğitim: Karar ağaçları, eğitimsel veri madenciliği, öğrenci verilerinin sınıflandırılması ve öğrenci performansının tahmin edilmesi için kullanılır.

Veri Madenciliği: Karar ağaçları, kümeleme ve sınıflandırma gibi veri madenciliği görevleri için kullanılır.

Robotik: Karar ağaçları robot navigasyonu ve kontrolünde kullanılır.

Bilgisayarlı Görme: Karar ağaçları nesne tanıma ve görüntü sınıflandırmada kullanılır.

Doğal Dil İşleme: Karar ağaçları metinleri sınıflandırmak ve metinlerdeki örüntüleri tanımlamak için kullanılır.

Üretim: Karar ağaçları, üretim süreçlerinde tahmine dayalı modelleme ve öngörüde kullanılır.

Oyun: Karar ağaçları bilgisayar oyunlarında ve yapay zekada da kullanılır.

3.2.2.Karar Ağacı Makine Öğrenimi Modelinin Özellikleri

Karar ağacının oluşturulma şekli oldukça basittir. İlk sorun (kök) ile başlanır. Temel kural, ağacınızı mümkün olduğunca küçük tutmanız gerektiğidir. Bu nedenle yalnızca geçerli sorular ve doğru yanıtı elde etmek için gerektiği kadar az soru sormanız şarttır. Ağacı oluşturmanın bu aşamasına tümevarım denir.

Soru size mümkün olduğunca çok bilgi/içgörü sağlıyorsa geçerlidir. Buna bilgi kazanımı denir. Ağacı oluştururken her adımda hangi özelliğe odaklanacağınıza karar vermek için kullanılır. Her zaman yalnızca daha iyi bir karar vermenize yardımcı olacak özellikleri/soruları seçmeniz gerekir.

Bu noktada basit ve tutarlı olmanız şarttır. Sırf daha büyük ve daha düzgün görünmesi için ağacı sağduyunun sınırlarına kadar uzatmanıza gerek yoktur.

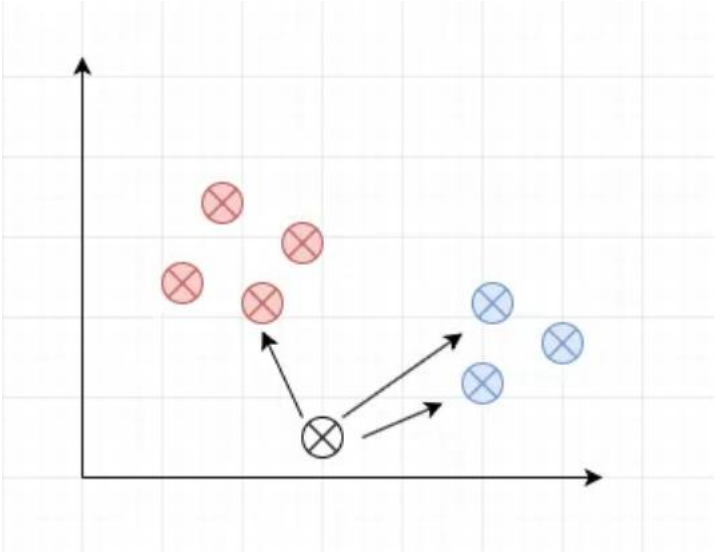
Bu konuyla ilgili olarak, ”budama” terimine de aşina olabilirsiniz. Bu, tahmin doğruluğunu azaltmadan bir makine öğrenimi ağacının boyutunu küçültmek için kullanılan bir tekniktir. Başka bir deyişle, düşük öneme sahip dalların ağaçtan çıkarılması anlamına gelir. Ağacın karmaşıklığını azaltırsanız, doğruluğu daha da yüksek olabilir. Bunu yapmak için, ağaçtaki her bir düğümü ve her bir yaprağı gözden geçirmeli ve çıkarmanın etkisini değerlendirmelisiniz. Sonuç üzerinde çok fazla değişiklik yapmıyorsa, tereddüt etmeden kaldırmanız şarttır.

3.4.K-En Yakın Komşu (k-Nearest Neighbors, KNN) Algoritması

k-En Yakın Komşu (k-Nearest Neighbor) Modeli, en basit ancak en etkili modellerden biridir. Bu modelde, test veri setlerinin sınıf etiketi, komşu eğitim veri öğelerinin sınıf etiketine dayanarak belirlenir. İki öğe arasındaki benzerlik, Öklidyen Mesafe kullanılarak ölçülür.

Bu aynı zamanda bir Örnek öğrenme veya Tembel model olarak da bilinir. 'k' değeri hesaplanır ki bu aslında düşünülmesi gereken en yakın komşu sayısını temsil eder. Uygun bir 'k' değeri seçilmelidir. Ayrıca uygun bir mesafe metriği de gereklidir. Bazı durumlarda 'Minkowski' mesafesi kullanılabilir. Bu, Öklidyen ve Manhattan mesafesinin genelleştirilmiş bir formudur.

Temel "çoğunluk oylaması" sınıflandırmasının bir dezavantajı, sınıf dağılımının çarpık olduğu durumlarda ortaya çıkar. Yani, daha sık görülen bir sınıfın örnekleri, yeni örneğin tahminini domine etme eğilimindedir, çünkü büyük sayıda komşu arasında yaygın olma eğilimindedirler. Bu sorunu aşmanın bir yolu, sınıflandırmaya ağırlık vererek, her bir k en yakın komşusuna olan uzaklığı dikkate almaktır.



3.4.1.KNN Algoritmasının Kullanımları

KNN Algoritması, bankacılık sisteminde, bir kişinin temerrüde düşen bir kişiyle benzer özelliklere sahip olup olmadığını tahmin ederek, bir kişinin kredi onayına uygun olup olmadığını tahmin etmek için kullanılır. KNN, benzer özelliklere sahip kişilerle karşılaştırarak kişilerin kredi notlarının hesaplanmasına da yardımcı olur.

Basitliğine rağmen KNN, diğer güçlü sınıflandırıcılardan çok daha iyi çalışır ve ekonomik tahmin ve veri sıkıştırma, Video Tanıma, Görüntü Tanıma, El Yazısı Algılama ve Konuşma Tanıma gibi yerlerde kullanılır.

Amazon veya Netflix gibi E-ticaret ve eğlence şirketlerinin çoğu, satın alınacak ürünleri veya izlenecek filmleri/şovları önerirken KNN'yi kullanır.

Bu tavsiyeleri nasıl yapıyorlar? Peki, bu şirketler web sitelerinde daha önce satın aldığınız ürünler veya izlediğiniz filmler gibi kullanıcı davranışları hakkında veri toplar ve KNN uygular.Şirketler, mevcut müşteri verilerinizi girecek ve bunları benzer ürünleri satın almış veya benzer filmler izlemiş olan diğer müşterilerle karşılaştıracaktır.Algoritmanın bu veri noktasını nasıl sınıflandırdığına bağlı olarak ürünler ve filmler size önerilecektir.

4.Uygulama

correlation_matrix

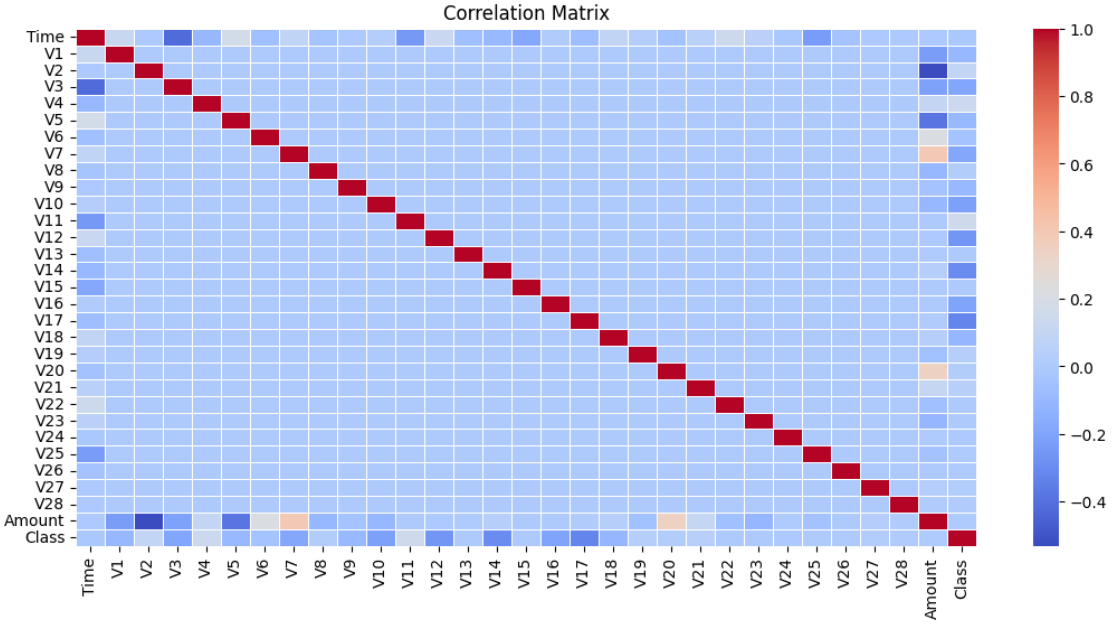
January 11, 2024

0.1 Korelasyon Analizi:

Korelasyon analizi, veri setindeki değişkenler arasındaki ilişkileri inceleyen önemli bir analiz yöntemidir. Yapılan korelasyon analizi, "Class" değişkeni ile diğer değişkenler arasındaki ilişkileri anlamamıza yardımcı oldu. Görsel olarak incelendiğinde, belirli özelliklerin sınıf değişkenini etkileyebileceği görülmekte, bu da kredi kartı dolandırıcılığı tespiti açısından önemli ipuçları sunabilir.

Ancak, korelasyon sadece iki değişken arasındaki ilişkiyi gösterir ve nedensellik hakkında bir şey söylemez. Bu nedenle, bu analiz sonuçlarını desteklemek ve daha kapsamlı bir anlayış elde etmek adına ileri seviye analizler yapmak faydalı olacaktır.

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Veriyi yukler
data = pd.read_csv('credit_card_data.csv')

# Korelasyon matrisini olusturur
correlation_matrix = data.corr()

# Korelasyon matrisini gosterir
plt.figure(figsize=(13, 6))
sns.heatmap(correlation_matrix, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

[ ]:

fraud_detection_machine_learning

January 11, 2024

[1]:
```
import pandas as pd

# Veriyi yukler
data = pd.read_csv('credit_card_data.csv')

# Veriyi gosterir
data.head()
```

[1]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	…	V21	V22	V23	V24	V25	\
0	0.098698	0.363787	…	-0.018307	0.277838	-0.110474	0.066928	0.128539	
1	0.085102	-0.255425	…	-0.225775	-0.638672	0.101288	-0.339846	0.167170	
2	0.247676	-1.514654	…	0.247998	0.771679	0.909412	-0.689281	-0.327642	
3	0.377436	-1.387024	…	-0.108300	0.005274	-0.190321	-1.175575	0.647376	
4	-0.270533	0.817739	…	-0.009431	0.798278	-0.137458	0.141267	-0.206010	

	V26	V27	V28	Amount	Class
0	-0.189115	0.133558	-0.021053	149.62	0
1	0.125895	-0.008983	0.014724	2.69	0
2	-0.139097	-0.055353	-0.059752	378.66	0
3	-0.221929	0.062723	0.061458	123.50	0
4	0.502292	0.219422	0.215153	69.99	0

[5 rows x 31 columns]

[2]:
```
# Veri setinin genel bilgilerini gösterir
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
```

```
---   ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

[3]:
```
# 'Class' sütunundaki değerlerin dağılımını gösterir
data['Class'].value_counts()
```

[3]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

[4]:
```
# Bagimsiz degiskenler ve hedef degiskeni belirler
X = data.drop('Class', axis=1)  # Hedef sutununu (Class) cikartarak yeni bir␣
 ↪dataframe olusturur
y = data['Class']
```

```
X.head()
```

[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	…	V20	V21	V22	V23	V24	\
0	0.098698	0.363787	…	0.251412	-0.018307	0.277838	-0.110474	0.066928	
1	0.085102	-0.255425	…	-0.069083	-0.225775	-0.638672	0.101288	-0.339846	
2	0.247676	-1.514654	…	0.524980	0.247998	0.771679	0.909412	-0.689281	
3	0.377436	-1.387024	…	-0.208038	-0.108300	0.005274	-0.190321	-1.175575	
4	-0.270533	0.817739	…	0.408542	-0.009431	0.798278	-0.137458	0.141267	

	V25	V26	V27	V28	Amount
0	0.128539	-0.189115	0.133558	-0.021053	149.62
1	0.167170	0.125895	-0.008983	0.014724	2.69
2	-0.327642	-0.139097	-0.055353	-0.059752	378.66
3	0.647376	-0.221929	0.062723	0.061458	123.50
4	-0.206010	0.502292	0.219422	0.215153	69.99

[5 rows x 30 columns]

[5]:
```
from sklearn.model_selection import train_test_split

# Veriyi eğitim ve test setlerine böler
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=123)
print(f'X_train: {X_train.shape}\nX_test: {X_test.shape}\ny_train: {y_train.
 ↪shape}\ny_test: {y_test.shape}')
```

```
X_train: (227845, 30)
X_test: (56962, 30)
y_train: (227845,)
y_test: (56962,)
```

[6]:
```
from sklearn.preprocessing import StandardScaler

# Veriler üzerinde normalize islemi
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

0.1 LOGISTIC REGRESSION

[7]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix


# Logistic Regresyon modelini oluştur ve eğit
lr = LogisticRegression()
lr.fit(X_train, y_train)

# Eğitim seti için classification report
y_train_pred_lr = lr.predict(X_train)
y_train_cl_report_lr = classification_report(y_train, y_train_pred_lr,␣
 ↪target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TRAIN MODEL CLASSIFICATION REPORT")
print("_" * 100)
print(y_train_cl_report_lr)


# Test seti için classification report
y_test_pred_lr = lr.predict(X_test)
y_test_cl_report_lr = classification_report(y_test, y_test_pred_lr,␣
 ↪target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TEST MODEL CLASSIFICATION REPORT")
print("_" * 100)
print(y_test_cl_report_lr)
print("_" * 100)
```

```
----------------------------------------------------------------------------------
--------------------
TRAIN MODEL CLASSIFICATION REPORT
----------------------------------------------------------------------------------
--------------------
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    227468
       Fraud       0.89      0.63      0.74       377

    accuracy                           1.00    227845
   macro avg       0.94      0.81      0.87    227845
weighted avg       1.00      1.00      1.00    227845


----------------------------------------------------------------------------------
--------------------
TEST MODEL CLASSIFICATION REPORT
```

```
--------------------------------------------------------------------------------
--------------------
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00     56847
       Fraud       0.83      0.61      0.70       115

    accuracy                           1.00     56962
   macro avg       0.92      0.80      0.85     56962
weighted avg       1.00      1.00      1.00     56962

--------------------------------------------------------------------------------
--------------------
```

0.2 Classification Report Açıklamaları

- **Precision (Hassasiyet):** Precision, pozitif olarak tahmin edilen örneklerin ne kadarının gerçekten pozitif olduğunu gösterir. Precision, yanlış pozitif örnekleri (modelin gerçekte negatif olan bir örneği pozitif olarak tahmin etmesi) azaltmayı hedefler. Precision, aşağıdaki formül ile hesaplanır:

  Precision = True Positives / (True Positives + False Positives)

- **Recall (Duyarlılık veya Recall):** Recall, gerçek pozitif örneklerin ne kadarının doğru bir şekilde saptandığını gösterir. Recall, pozitif sınıftaki tüm gerçek örnekleri yakalamayı hedefler. Recall, aşağıdaki formül ile hesaplanır:

  Recall = True Positives / (True Positives + False Negatives)

- **F1-Score:** F1-score, precision ve recall'ın harmonik ortalamasıdır. F1-score, precision ve recall arasında denge sağlar ve aşağıdaki formül ile hesaplanır:

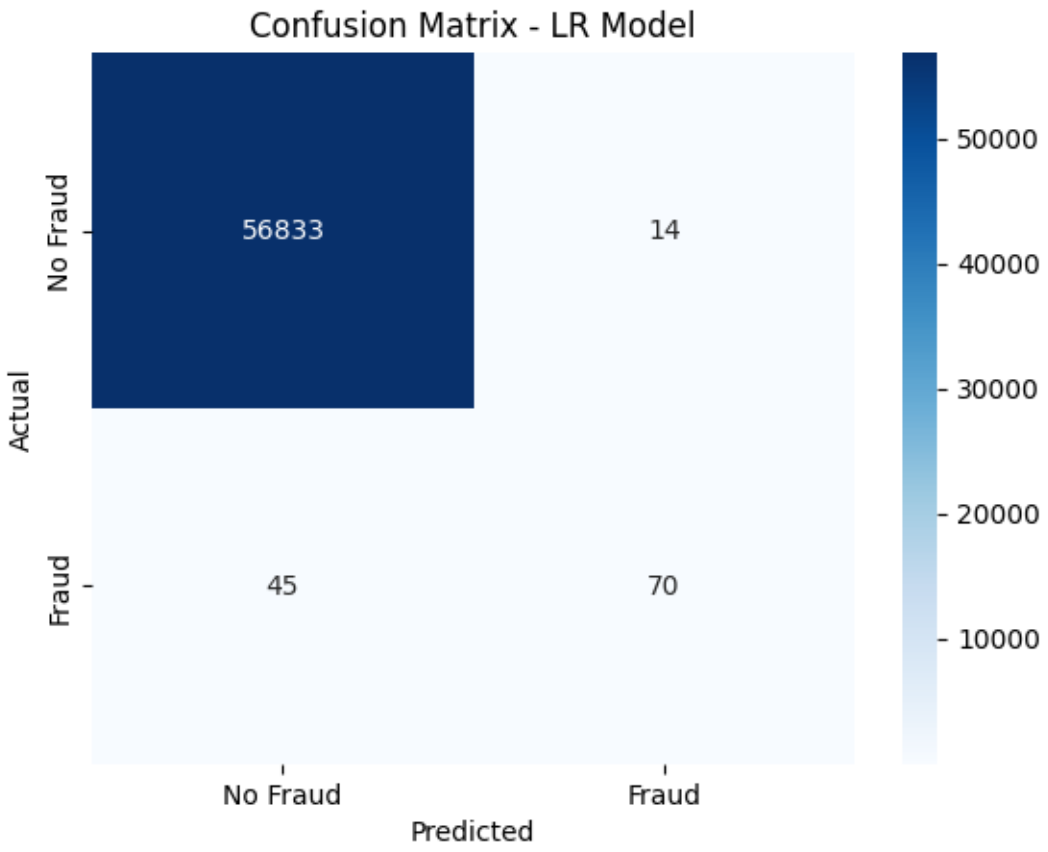
  F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

- **Support:** Her bir sınıf için gerçekte kaç adet örnek olduğunu gösterir. Support, modelin değerlendirildiği veri setindeki örnek sayılarını temsil eder.

Bu metrikler, bir sınıflandırma modelinin performansını değerlendirmek için kullanılır. Precision, recall ve f1-score, özellikle dengesiz sınıflı veri setlerinde modelin performansını anlamak için önemlidir. Support, her sınıfın veri setindeki gerçek örnek sayısını belirtir ve bu da değerlendirmenin güvenilirliğini sağlar.

1. **Precision:**
   - Precision is the ratio of correctly predicted positive observations to the total predicted positives.
   - It measures the accuracy of the positive predictions.
   - Precision is calculated as: $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$
   - Precision is high when the false positive rate is low.
2. **Recall (Sensitivity or True Positive Rate):**
   - Recall is the ratio of correctly predicted positive observations to all actual positives.
   - It measures the ability of the model to capture all the relevant cases.
   - Recall is calculated as: $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$
   - Recall is high when the false negative rate is low.
3. **F1-score:**
   - The F1-score is the harmonic mean of precision and recall.
   - It provides a balance between precision and recall, making it useful when you want to consider both false positives and false negatives.
   - F1-score is calculated as: $\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
   - F1-score ranges from 0 to 1, where 1 indicates perfect precision and recall.

[52]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Confusion matrix
con_mat = confusion_matrix(y_test, y_test_pred_lr)

# Sınıf etiketleri
labels = ['No Fraud', 'Fraud']

# Isı haritasını oluştur
sns.heatmap(con_mat, annot=True, fmt='d', xticklabels=labels,␣
 ↪yticklabels=labels, cmap='Blues')

# Eksen etiketleri
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - LR Model')

# Görseli göster
plt.show()
```



0.3 Confusion Matrix Açıklaması

Confusion matrix (karmaşıklık matrisi), bir sınıflandırma modelinin performansını daha ayrıntılı bir şekilde değerlendirmek için kullanılır. 2 sınıflı bir problemde confusion matrix, aşağıdaki dört öğeyi içerir:

- **True Positives (TP):** Gerçek pozitif örneklerin sayısı.
- **True Negatives (TN):** Gerçek negatif örneklerin sayısı.
- **False Positives (FP):** Gerçek negatif örnekleri pozitif olarak yanlış sınıflandırılan örneklerin sayısı.
- **False Negatives (FN):** Gerçek pozitif örnekleri negatif olarak yanlış sınıflandırılan örneklerin sayısı.

Confusion matrix, sınıflandırma modelinin hangi tür hatalar yaptığını ve hangi sınıfları daha iyi veya daha kötü tahmin ettiğini görmek için kullanılır.

0.4 KNN (K nearest neighborhood, En Yakın K Komşu)

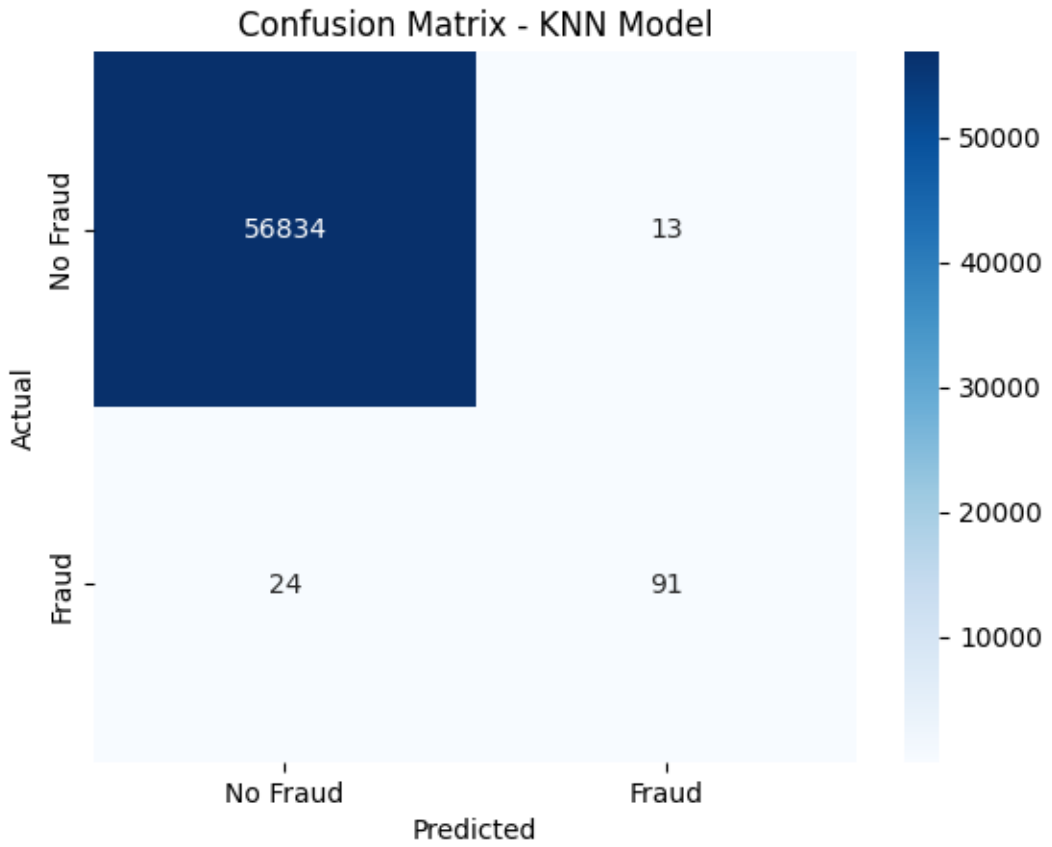
[9]:
```
from sklearn.neighbors import KNeighborsClassifier

# KNN modelini oluşturalım ve eğitelim
knn_model = KNeighborsClassifier(n_neighbors=5)  # Örnek olarak 5 komşu␣
 ↪alınmıştır, siz bu değeri ayarlayabilirsiniz
knn_model.fit(X_train, y_train)


# Eğitim seti için classification report
y_train_pred_knn = knn_model.predict(X_train)
y_train_cl_report_knn = classification_report(y_train, y_train_pred_knn,␣
 ↪target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TRAIN MODEL CLASSIFICATION REPORT - KNN")
print("_" * 100)
print(y_train_cl_report_knn)

# Test seti için classification report
y_test_pred_knn = knn_model.predict(X_test)
y_test_cl_report_knn = classification_report(y_test, y_test_pred_knn,␣
 ↪target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TEST MODEL CLASSIFICATION REPORT - KNN")
print("_" * 100)
print(y_test_cl_report_knn)
print("_" * 100)
```

```
____________________________________________________________________________________
____________________
TRAIN MODEL CLASSIFICATION REPORT - KNN
____________________________________________________________________________________
____________________
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00    227468
       Fraud       0.95      0.80      0.87       377

    accuracy                           1.00    227845
   macro avg       0.97      0.90      0.93    227845
weighted avg       1.00      1.00      1.00    227845

____________________________________________________________________________________
____________________
TEST MODEL CLASSIFICATION REPORT - KNN
____________________________________________________________________________________
____________________
```

```
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00     56847
       Fraud       0.88      0.79      0.83       115

    accuracy                           1.00     56962
   macro avg       0.94      0.90      0.92     56962
weighted avg       1.00      1.00      1.00     56962
```

---

[51]:
```
# Confusion matrix'i görselleştirelim
con_mat_knn = confusion_matrix(y_test, y_test_pred_knn)
labels_knn = ['No Fraud', 'Fraud']
sns.heatmap(con_mat_knn, annot=True, fmt='d', xticklabels=labels_knn,␣
 ↪yticklabels=labels_knn, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - KNN Model')
plt.show()
```



0.4.1 Resampling (Yeniden Örnekleme) Nedir?

Veri setinin sınıfları arasındaki dengesizliği gidermek için kullanılan bir yöntemdir. Dengesiz sınıflara sahip bir veri setinde, model, çoğunluktaki sınıfı daha iyi öğrenme eğilimindedir. Resampling, özellikle sınıflar arasındaki örnek sayılarını dengelemek için kullanılır.

Bu yöntem, genellikle iki şekilde uygulanır:

1. **Upsampling (Artırma):** Azınlık sınıfındaki örnek sayısını artırmak için kullanılır. Örneğin, azınlık sınıfındaki örnekleri çoğaltabilir veya benzer örnekleri ekleyebilirsiniz.
2. **Downsampling (Azaltma):** Çoğunluk sınıfındaki örnek sayısını azaltmak için kullanılır. Örneğin, çoğunluk sınıfındaki rastgele örnekleri çıkarabilirsiniz.

Bu yöntemler, veri setinin dengesini sağlayarak, modelin dengesiz sınıfları daha adil bir şekilde öğrenmesine yardımcı olabilir.

[11]:
```
from sklearn.utils import resample

# Fraud ve fraud olmayan sınıfları ayıralım
fraud_data = data[data['Class'] == 1]
non_fraud_data = data[data['Class'] == 0]

# Fraud olmayan sınıftan 1000 örnek alalım (upsampling)
non_fraud_resampled = resample(non_fraud_data, replace=True, n_samples=1000,␣
 ↪random_state=42)

# Yeni dengelenmiş veri setini oluşturalım
balanced_data = pd.concat([fraud_data, non_fraud_resampled])

# Veriyi karıştıralım
balanced_data = balanced_data.sample(frac=1, random_state=42)
```

[12]:
```
# 'Class' sütunundaki değerlerin dağılımını gösterir
balanced_data['Class'].value_counts()
```

[12]:
```
Class
0    1000
1     492
Name: count, dtype: int64
```

[13]:
```
# Bağımsız değişkenler ve hedef değişkeni belirleyelim
X_balanced = balanced_data.drop('Class', axis=1)
y_balanced = balanced_data['Class']
```

[14]:
```
# Veriyi eğitim ve test setlerine bölelim
X_train_balanced, X_test_balanced, y_train_balanced, y_test_balanced =␣
 ↪train_test_split(X_balanced, y_balanced, test_size=0.2, random_state=42)
```

```
print(f'X_train_balanced: {X_train_balanced.shape}\nX_test_balanced:␣
 ↪{X_test_balanced.shape}\ny_train: {y_train_balanced.shape}\ny_test:␣
 ↪{y_test_balanced.shape}')
```

```
X_train_balanced: (1193, 30)
X_test_balanced: (299, 30)
y_train: (1193,)
y_test: (299,)
```

[15]:
```
# Veriyi normalize edelim
scaler = StandardScaler()
X_train_balanced = scaler.fit_transform(X_train_balanced)
X_test_balanced = scaler.transform(X_test_balanced)
```

0.5 LOGISTIC REGRESSION WITH RESAMPLING

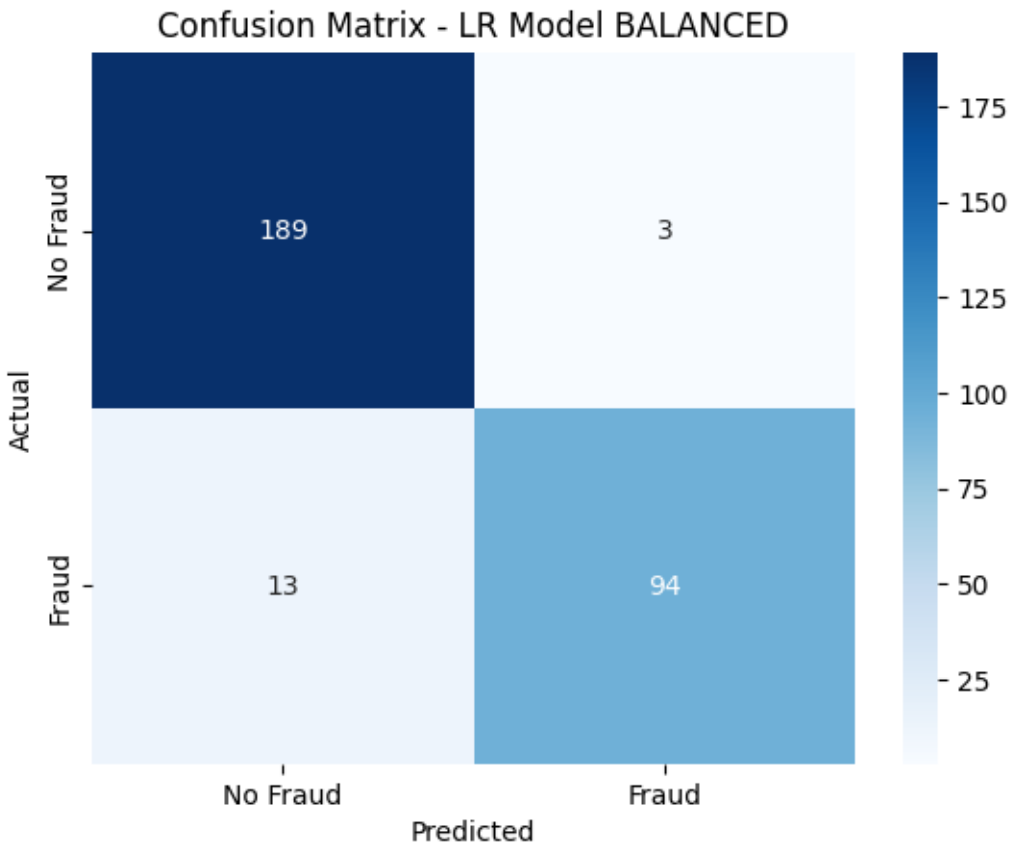
[16]:
```
# Logistic Regresyon modelini oluşturur ve eğit
lr_balanced = LogisticRegression()
lr_balanced.fit(X_train_balanced, y_train_balanced)

# Eğitim seti için classification report
y_train_pred_balanced_lr = lr_balanced.predict(X_train_balanced)
y_train_cl_report_balanced_lr = classification_report(y_train_balanced,␣
 ↪y_train_pred_balanced_lr, target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TRAIN MODEL CLASSIFICATION REPORT BALANCED")
print("_" * 100)
print(y_train_cl_report_balanced_lr)

# Test seti için classification report
y_test_pred_balanced_lr = lr_balanced.predict(X_test_balanced)
y_test_cl_report_balanced_lr = classification_report(y_test_balanced,␣
 ↪y_test_pred_balanced_lr, target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TEST MODEL CLASSIFICATION REPORT BALANCED")
print("_" * 100)
print(y_test_cl_report_balanced_lr)
print("_" * 100)
```

```
____________________________________________________________________________
____________________
TRAIN MODEL CLASSIFICATION REPORT BALANCED
____________________________________________________________________________
____________________
              precision    recall  f1-score   support

    No Fraud       0.95      0.99      0.97       808
```

```
       Fraud       0.99      0.90      0.94       385

    accuracy                           0.96      1193
   macro avg       0.97      0.95      0.96      1193
weighted avg       0.96      0.96      0.96      1193

--------------------------------------------------------------------------------
--------------------
TEST MODEL CLASSIFICATION REPORT BALANCED
--------------------------------------------------------------------------------
--------------------
              precision    recall  f1-score   support

    No Fraud       0.94      0.98      0.96       192
       Fraud       0.97      0.88      0.92       107

    accuracy                           0.95       299
   macro avg       0.95      0.93      0.94       299
weighted avg       0.95      0.95      0.95       299

--------------------------------------------------------------------------------
--------------------
```

0.5.1 Confusion Matrix

[50]:
```
# Confusion matrix'i görselleştirelim
con_mat_knn = confusion_matrix(y_test_balanced, y_test_pred_balanced_lr)
labels_knn = ['No Fraud', 'Fraud']
sns.heatmap(con_mat_knn, annot=True, fmt='d', xticklabels=labels_knn,␣
 ↪yticklabels=labels_knn, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - LR Model BALANCED')
plt.show()
```



0.6 KNN (K-nearest neighborhood) RESAMPLING

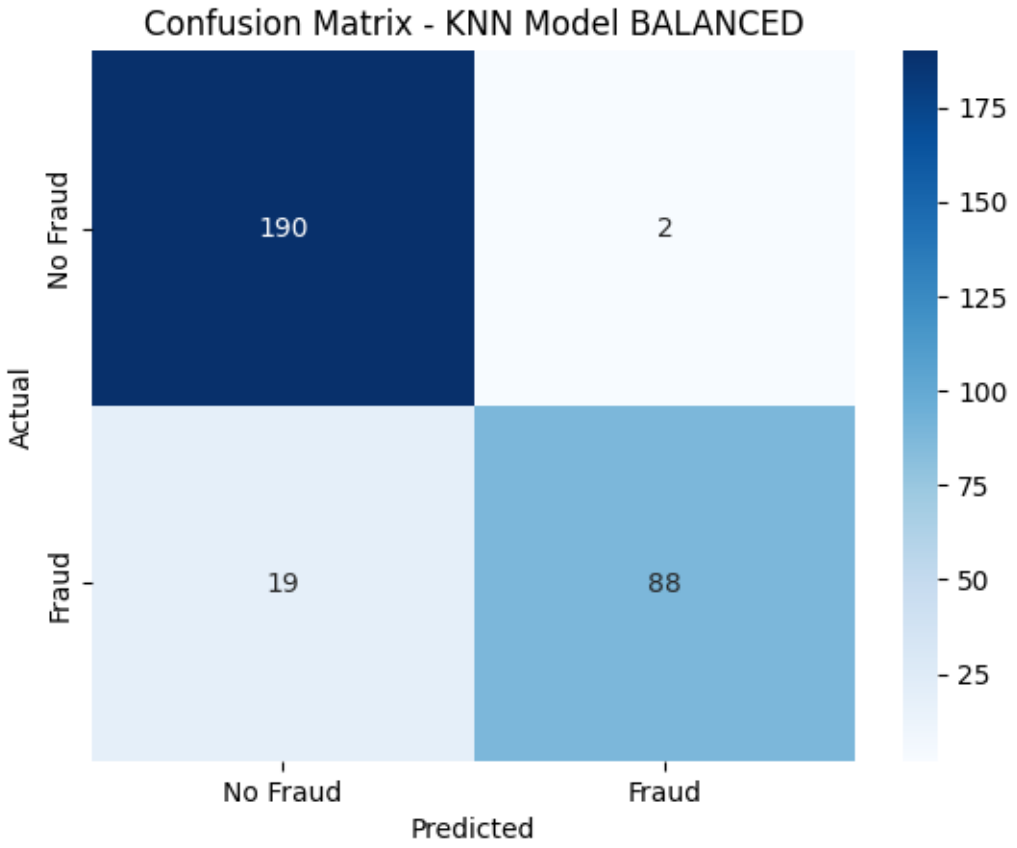
[18]:
```
# KNN modelini oluşturalım ve eğitelim
knn_model_balanced = KNeighborsClassifier(n_neighbors=7)  # Örnek olarak 5␣
 ↪komşu alınmıştır, siz bu değeri ayarlayabilirsiniz
knn_model_balanced.fit(X_train_balanced, y_train_balanced)


# Eğitim seti için classification report
y_train_pred_balanced_knn = knn_model_balanced.predict(X_train_balanced)
y_train_cl_report_balanced_knn = classification_report(y_train_balanced,␣
 ↪y_train_pred_balanced_knn, target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TRAIN MODEL CLASSIFICATION REPORT - KNN BALANCED")
print("_" * 100)
print(y_train_cl_report_balanced_knn)

# Test seti için classification report
y_test_pred_balanced_knn = knn_model_balanced.predict(X_test_balanced)
```

```
y_test_cl_report_balanced_knn = classification_report(y_test_balanced,␣
 ↪y_test_pred_balanced_knn, target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TEST MODEL CLASSIFICATION REPORT - KNN")
print("_" * 100)
print(y_test_cl_report_balanced_knn)
print("_" * 100)
```

```
----------------------------------------------------------------------------------
--------------------
TRAIN MODEL CLASSIFICATION REPORT - KNN BALANCED
----------------------------------------------------------------------------------
--------------------
              precision    recall  f1-score   support

    No Fraud       0.94      1.00      0.97       808
       Fraud       0.99      0.86      0.92       385

    accuracy                           0.95      1193
   macro avg       0.96      0.93      0.94      1193
weighted avg       0.95      0.95      0.95      1193

----------------------------------------------------------------------------------
--------------------
TEST MODEL CLASSIFICATION REPORT - KNN
----------------------------------------------------------------------------------
--------------------
              precision    recall  f1-score   support

    No Fraud       0.91      0.99      0.95       192
       Fraud       0.98      0.82      0.89       107

    accuracy                           0.93       299
   macro avg       0.94      0.91      0.92       299
weighted avg       0.93      0.93      0.93       299

----------------------------------------------------------------------------------
--------------------
```

[49]:
```
# Confusion matrix'i görselleştirelim
con_mat_knn = confusion_matrix(y_test_balanced, y_test_pred_balanced_knn)
labels_knn = ['No Fraud', 'Fraud']
sns.heatmap(con_mat_knn, annot=True, fmt='d', xticklabels=labels_knn,␣
 ↪yticklabels=labels_knn, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - KNN Model BALANCED')
```

```
plt.show()
```



0.7 Support Vector Machine (SVM) BALANCED:

Support Vector Machine (SVM), özellikle sınıflar arasındaki en iyi ayrım çizgisini (veya düzlemi) bulmaya çalışan bir sınıflandırma ve regresyon algoritmasıdır.

SVM, veri noktalarını iki sınıf arasında optimal bir hiper-düzlemle ayırmaya çalışır. Bu düzlem, sınıflar arasındaki en büyük marjı (mesafe) maksimize eder ve bu nedenle model, veri noktalarına karşı daha dirençli ve genelleme yeteneği daha yüksek olabilir.

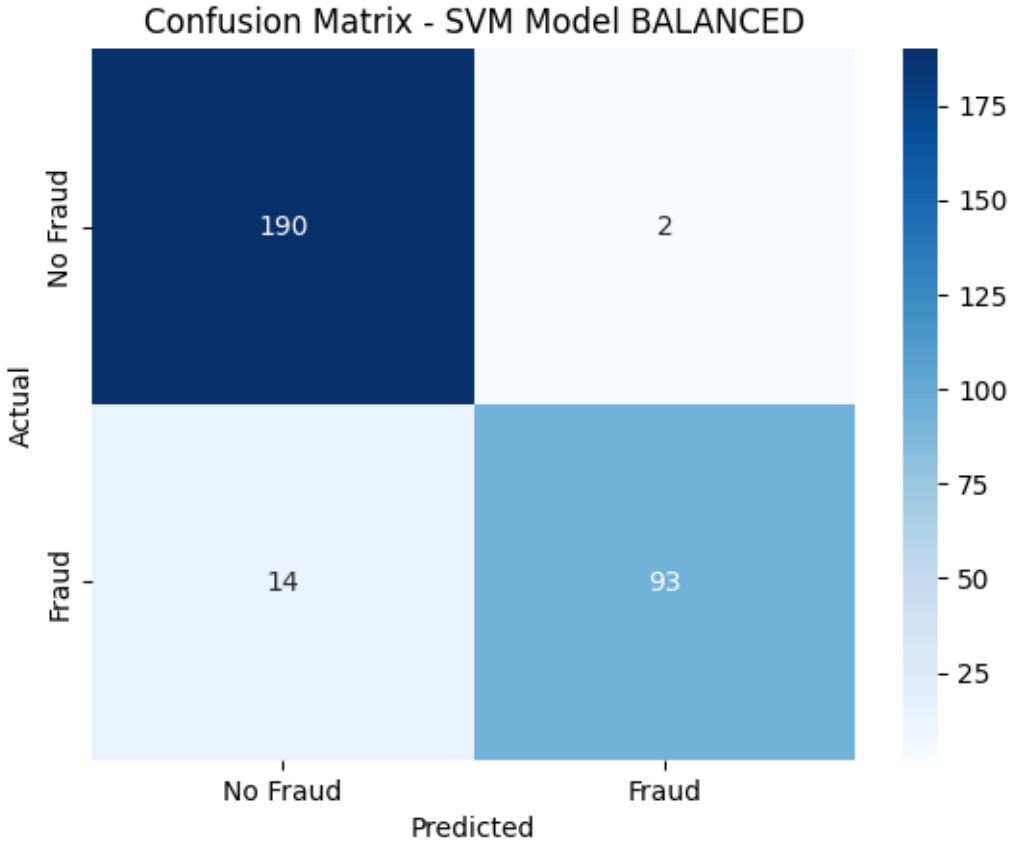
[32]:
```
from sklearn.svm import SVC

# SVM modelini oluşturalım ve eğitelim
svm_model_balanced = SVC(kernel='linear', C=0.5)  # Lineer kernel ve C␣
 ↪parametresi örnek olarak belirlenmiştir
svm_model_balanced.fit(X_train_balanced, y_train_balanced)


# Eğitim seti için classification report
```

```
y_train_pred_balanced_svm = svm_model_balanced.predict(X_train_balanced)
y_train_cl_report_balanced_svm = classification_report(y_train_balanced,␣
 ↪y_train_pred_balanced_svm, target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TRAIN MODEL CLASSIFICATION REPORT - SVM BALANCED")
print("_" * 100)
print(y_train_cl_report_balanced_svm)

# Test seti için classification report
y_test_pred_balanced_svm = svm_model_balanced.predict(X_test_balanced)
y_test_cl_report_balanced_svm = classification_report(y_test_balanced,␣
 ↪y_test_pred_balanced_svm, target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TEST MODEL CLASSIFICATION REPORT - SVM BALANCED")
print("_" * 100)
print(y_test_cl_report_balanced_svm)
print("_" * 100)
```

```
----------------------------------------------------------------------------------------------------
TRAIN MODEL CLASSIFICATION REPORT - SVM BALANCED
----------------------------------------------------------------------------------------------------
              precision    recall  f1-score   support

    No Fraud       0.95      1.00      0.97       808
       Fraud       0.99      0.89      0.94       385

    accuracy                           0.96      1193
   macro avg       0.97      0.94      0.95      1193
weighted avg       0.96      0.96      0.96      1193


----------------------------------------------------------------------------------------------------
TEST MODEL CLASSIFICATION REPORT - SVM BALANCED
----------------------------------------------------------------------------------------------------
              precision    recall  f1-score   support

    No Fraud       0.93      0.99      0.96       192
       Fraud       0.98      0.87      0.92       107

    accuracy                           0.95       299
   macro avg       0.96      0.93      0.94       299
weighted avg       0.95      0.95      0.95       299


----------------------------------------------------------------------------------------------------
```

---

[39]:
```
# Confusion matrix'i görselleştirelim
con_mat_svm = confusion_matrix(y_test_balanced, y_test_pred_balanced_svm)
labels_svm = ['No Fraud', 'Fraud']
sns.heatmap(con_mat_svm, annot=True, fmt='d', xticklabels=labels_svm,␣
 ↪yticklabels=labels_svm, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - SVM Model BALANCED')
plt.show()
```



0.8 Decision Tree - BALANCED

Bu bölümde, dengelenmiş bir veri seti üzerinde eğitilmiş bir Decision Tree modelinin performansını değerlendiriyoruz. Bu model, dengesiz sınıflar arasındaki performans farkını azaltmaya yönelik olarak özellikle tercih edilmiştir.

[42]:
```
from sklearn.tree import DecisionTreeClassifier
```

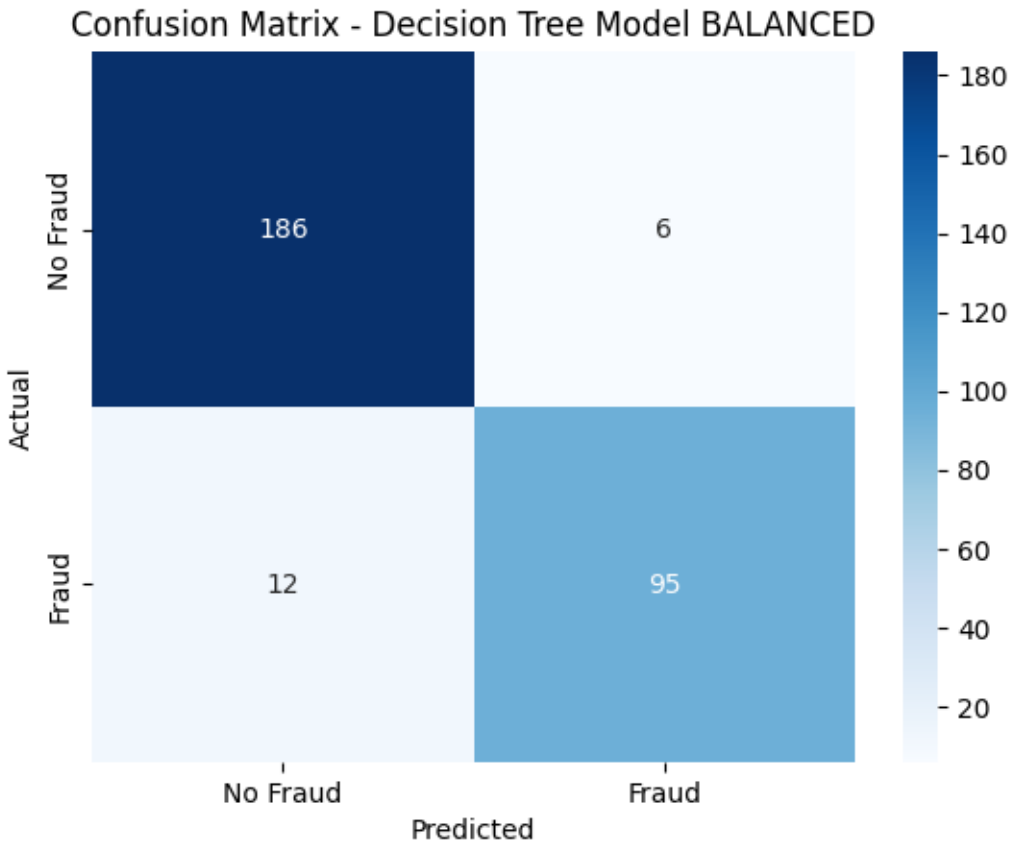
```
# Decision Tree modelini oluşturalım ve eğitelim
dt_model_balanced = DecisionTreeClassifier(criterion='gini', max_depth=50)  #␣
 ↪Örnek olarak belirlenen kriter ve maksimum derinlik
dt_model_balanced.fit(X_train_balanced, y_train_balanced)

# Eğitim seti için classification report
y_train_pred_balanced_dt = dt_model_balanced.predict(X_train_balanced)
y_train_cl_report_balanced_dt = classification_report(y_train_balanced,␣
 ↪y_train_pred_balanced_dt, target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TRAIN MODEL CLASSIFICATION REPORT - DECISION TREE BALANCED")
print("_" * 100)
print(y_train_cl_report_balanced_dt)

# Test seti için classification report
y_test_pred_balanced_dt = dt_model_balanced.predict(X_test_balanced)
y_test_cl_report_balanced_dt = classification_report(y_test_balanced,␣
 ↪y_test_pred_balanced_dt, target_names=['No Fraud', 'Fraud'])
print("_" * 100)
print("TEST MODEL CLASSIFICATION REPORT - DECISION TREE BALANCED")
print("_" * 100)
print(y_test_cl_report_balanced_dt)
print("_" * 100)
```

```
____________________________________________________________________________________
____________________
TRAIN MODEL CLASSIFICATION REPORT - DECISION TREE BALANCED
____________________________________________________________________________________
____________________
              precision    recall  f1-score   support

    No Fraud       1.00      1.00      1.00       808
       Fraud       1.00      1.00      1.00       385

    accuracy                           1.00      1193
   macro avg       1.00      1.00      1.00      1193
weighted avg       1.00      1.00      1.00      1193


____________________________________________________________________________________
____________________
TEST MODEL CLASSIFICATION REPORT - DECISION TREE BALANCED
____________________________________________________________________________________
____________________
              precision    recall  f1-score   support

    No Fraud       0.94      0.97      0.95       192
       Fraud       0.94      0.89      0.91       107
```

```
    accuracy                           0.94       299
   macro avg       0.94      0.93      0.93       299
weighted avg       0.94      0.94      0.94       299
```

---

[48]:
```
# Confusion matrix
con_mat_dt = confusion_matrix(y_test_balanced, y_test_pred_balanced_dt)
# Sınıf etiketleri
labels_dt = ['No Fraud', 'Fraud']
# Isı haritasını oluştur
sns.heatmap(con_mat_dt, annot=True, fmt='d', xticklabels=labels_dt,␣
 ↪yticklabels=labels_dt, cmap='Blues')
# Eksen etiketleri
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Decision Tree Model BALANCED')

# Görseli göster
plt.show()
```



[ ]: