**Kredi Risk Analizi**

**Özet**

Risklerin gerçekçi bir şekilde tespit edilmesi ve olası kayıpların görünür hale getirilmesi amacıyla kredi riskinin ölçülmesine ihtiyaç duyulmaktadır. Kredi riskinin ölçülmesine yönelik olarak gelişti rilen çeşitli modellerde temel amaç, temerrüde düşmesi olas borçlunun belirlenerek risk miktarının tahmin edilmesidir.

Bu çalışmada, dört farklı makine öğrenmesi algoritması kullanarak modeller oluşturacağız: **Logistic Regression**, **Decision Tree**, **Gradient Boosting ve Malatya Centrality Algoritması**.

Öncelikle, modelleri eğitmek için Kaggle sitesinden bir veri seti seçiyoruz. Seçtiğimiz veri seti, bir Alman bankasına ait gerçek verilerden oluşmaktadır. Daha sonra, sonuca en çok etki eden ve en çok ilişkili özellikleri seçiyoruz. Veriler işlenirken, bilgi kaybı fazla olan veriler temizlenmiştir ve bu süreçte Standardizasyon kullanılmıştır. Veriler işlenip eğitim için hazırlandıktan sonra, üç gruba ayrılır: eğitim, doğrulama ve test.

Veri hazırlama süreci tamamlandıktan ve modellere ana girdi olacak özellikler seçildikten sonra eğitim süreci başlar.

Eğitim işlemi tamamlandıktan sonra, modellerin performansını test grubuyla değerlendiriyoruz. Bu aşamada, doğruluk, hasasiyet gibi çeşitli performans ölçütlerini kullanarak üç modelin sonuçlarını karşılaştırıyoruz.

Sonuç olarak, genel doğruluk ve yüksek geri çağırma açısından **Gradient Boosting** algoritmasının üstün performans gösterdiğini gözlemliyoruz. **Decision Tree** iyi bir performans sergilemiştir ancak Gradient Boosting'e kıyasla daha düşük bir verim sunmuştur; makul bir doğruluğa sahip olsa da geri çağırma değeri biraz daha düşüktür. **Malatya Merkezlik** ise en düşük doğruluk ve geri çağırma oranları ile üç algoritma arasında en düşük performansı göstermiştir.

**Giriş**

Projemle ilgili birçok kaynağı ve bunlarda açıklanan çözümleri inceledikten sonra, kredi riski analizi sorununu çözmek için en popüler üç algoritmayı seçtim. Daha sonra, bu proje için uygun veri setine ve seçilen algoritmaların yürütülmesine seçtim.

veri setimiz çok gereksiz özellik içerdiği nedeniyle ,hedefle maksimum korelasyona sahip özellikleri seçmek gerekti. Bunu yapmak için RandomForestClassifier'ı kullandım. Bu, en iyilerini seçmek için her özelliği hedef üzerindeki etkileriyle birlikte kontrol eden bir makine öğrenme algoritmasıdır. Algoritmalar aracılığıyla ana girdiler olacak özellikleri seçtikten sonraki adım, modeli eğitmek için verileri ön işlemeye başladım.

Veri setinin modelde zayıflığa sebep olacak eksik değerler içermediğini kontrol ettim,bundan sonra veri modellerinin eğitim seviyesini yükseltmek amacıyla, verilerin ölçü birimlerini standartlaştırmak için Normalizasyon yöntemini kullandım. Bu adım, özellikler arasındaki ölçüm birimlerindeki farkı azaltarak modellerin öğrenme oranını ve performansını iyileştirmektir.

Daha sonra, veri setini üç parçaya böldüm (%85 eğitim ve %15 test) ve eğitim seti %70 eğitim ve %15 doğrulama olarak bölündü.

böylece ön işleme aşamasını bitirdim, ve veri setimiz modellerin eğitimi için hazırladım.

Projenin dayanacağı modelleri tanımladım: Gradient Boosting, Decision Tree ve Logistic Regression.

1. **Gradient Boosting**: Gradyan artırma, zayıf modelleri (örneğin küçük karar ağaçları) güçlendiren ve güçlü bir model oluşturan bir makine öğrenimi tekniğidir. Bu algoritma, bir önceki modelin hatalarını (Residual Errors) azaltarak iteratif olarak çalışır. Kayıp fonksiyonu (Loss Function) kullanılarak modeller ardışık olarak eğitilir ve performans adım adım geliştirilir.
2. **Decision Tree**: Karar ağacı, verileri anlamak ve karar almak için kullanılan bir algoritmadır. Bu algoritma, verileri belirli kurallara göre alt gruplara ayırarak bir karar ağacı oluşturur. Model, kök düğümden (Root Node) başlar ve dallar (Branches) boyunca ilerleyerek yaprak düğümlerine (Leaf Nodes) ulaşır; bu düğümler kararları içerir.
3. **Logistic Regression**: Lojistik regresyon, sınıflandırma için kullanılan bir makine öğrenimi algoritmasıdır, özellikle ikili sınıflandırma (Binary Classification) problemlerinde yaygın olarak kullanılır. Örneğin, "Evet/Hayır" veya "Kabul/Red" gibi durumlar için idealdir. Bu algoritma, çıktıları 0 ile 1 arasında olasılık değerlerine dönüştürmek için lojistik fonksiyon (Sigmoid Fonksiyonu) kullanır. Ana hedef, bağımsız değişkenler (Features) ile bağımlı değişken (Target) arasında doğrusal bir ilişki bulmaktır.
4. **Malatya Centralty:** Malatya Merkezilik: Vertex cover problemini çözmek için polinomsal bir yaklaşım sunan bir algoritmadır. Düğümlerin merkezilik değerlerini, düğüm derecesinin bitişik düğümlerin derecelerine oranlarının toplamı ile hesaplar ve veri gruplarını sınıflandırmada kullanılır.

Eğitim seti, eğitimi başlatmak için kullanılır. Daha sonra, modelin kararlılığını değerlendirmek için ortalama doğruluk ve standart sapma hesaplanır; ayrıca doğruluk ve hassasiyet gibi metrikler de hesaplanır. Uygulanan modellerin sonuçları karşılaştırılarak en iyi performansı gösteren model belirlenir.

**Yöntem**

**Veri seti**

* **Kaynak:** Veri setini Kaggle'dan aldım. Bu, bir Alman bankasının gerçek verilerinden oluşan bir veri setidir.
* **Boyut:** Veri seti **1 milyon** gözlemdenoluşmaktadır.

modellerde girdi olarak kullanılacak özellikler belirlenmesi, model tahmin gücünü doğrudan etkileyen faktördür. Aynı zamanda girdi olarak kullanılacak gözlemler içinde bu durum geçerlidir.

RandomForestClassifier algoritması kullanılarak hedef değişkenle en yakından ilişkili en önemli on özellik belirlendi ve seçildi. (RandomForestClassifier, rastgele seçilmiş veri örnekleri ve özellik alt kümeleriyle birden fazla karar ağacı eğitip bunların tahminlerini birleştirerek sınıflandırma yapan ve yüksek doğruluk sağlayan bir makine öğrenimi algoritmasıdır.)

Seçilen özellikler bunlardır**:**

* **Kredi Puanı**
* **Yıllık Gelir**
* **Yıllık Giderler**
* **Kredi Tutarı**
* **Borç-Gelir Oranı**
* **İflas Geçmişi**
* **Önceki Kredi Temerrütleri**
* **Ödeme Geçmişi**
* **Net Varlık Değeri**
* **Faiz Oranı.**
* **Hedef Değişken:** **Kredi Onayı** (kabul/red)

A graph with blue squares

Description automatically generated

RandomForestClassifier algoritmasi veri setimiz üzerinde sonuçları

Veri seti hakkında bilgiler

|  |  |  |  |
| --- | --- | --- | --- |
| **özellik** | **Her sütundaki en sık tekrar eden değer (mode):** | **Her sütundaki ortalama değer (mean):** | **Her sütundaki benzersiz değer sayısı:** |
| Kredi Puanı | 508 | 574.559628 | 550 |
| Yıllık Gelir | 53002 | 59971.2273 | 82425 |
| Yıllık Giderler | 30128 | 29990.07072 | 57746 |
| Kredi Tutarı | 19143 | 20001.6435 | 31209 |
| Borç-Gelir Oranı | 0.000061 | 0.500026 | 999509 |
| İflas Geçmişi | 0 | 0.049872 | 2 |
| Önceki Kredi Temerrütleri | 0 | 0.100256 | 2 |
| Ödeme Geçmişi | 11 | 14.503627 | 30 |
| Net Değer | 45409 | 50035.89736 | 147941 |
| Faiz Oranı | 0.010084 | 0.155151 | 998240 |
| Kredi Onayı | 0 | 0.373169 | 2 |

**Veri Ön İşleme**

* **Eksik Değerlerin Kontrolü:** Veri setinde eksik değerlerin olup olmadığını kontrol ettim.
* **Veri Standardizasyonu:** Özellikleri standartlaştırmak için StandardScaler kullandım. Bu, modellerin daha iyi performans göstermesine yardımcı oldu.( Standardizasyon (Standartlaştırma), verilerin ölçeklerini birbiriyle uyumlu hale getirmek için kullanılan bir yöntemdir. Bu yöntemle, veriler ortalaması 0 ve standart sapması 1 olacak şekilde dönüştürülür. Formül şu şekildedir:

x: Veri değeri

μ: Verilerin ortalaması

σ: Verilerin standart sapması

Bu işlem, özellikle farklı ölçeklere sahip değişkenlerin bir arada kullanıldığı makine öğrenimi algoritmalarında modelin doğruluğunu artırmak için önemlidir.)

* **Veri Seti Bölme:** Veri seti, modelin eğitimi için %70 eğitim, modelin ayarlanması için %15 doğrulama ve modelin performansını değerlendirmek için %15 test olarak bölündü.

**Çalışmada Kullanılan Sınıflandırma Algoritmaları**

**1. Lojistik Regresyon:**

* **Genel Açıklama:**
  + Lojistik regresyon, **denetimli öğrenme** algoritmalarından biridir ve genellikle **ikili sınıflandırma** (örneğin, evet/hayır, onaylandı/onaylanmadı) problemlerini çözmek için kullanılır.
  + Bağımsız değişkenler (özellikler) ile bağımlı değişken (hedef) arasındaki ilişkiyi lojistik fonksiyon kullanarak modellemektedir.
  + Algoritma, **logistik fonksiyonunu** (**sigmoid fonksiyonu**) kullanarak sürekli değerleri 0 ile 1 arasında olasılıklara dönüştürür.
  + **Lojistik Fonksiyon (Sigmoid Fonksiyonu):** 
    - ho tahmin edilen olasılık
    - b katsayılar (model parametreleri)
    - x bağımsız değişkenler
* **Neden Kullandım:** Basitlik ve Yorumlanabilirlik, ve Hız ve Verimlilik

**2. Karar Ağacı (Decision Tree):**

* **Genel Açıklama:**
  + Karar ağacı, özelliklere dayalı olarak kararlar veren ve bu kararları ağaç yapısı şeklinde gösteren bir makine öğrenma algoritmasıdır.
  + Her düğümde en iyi bölünme noktasını belirlemek için belirli bir özelliği kullanır ve bu sayede veriyi sınıflandırır.
  + Verileri, özelliklerin belirli eşik değerlerine göre dallandırarak son karara (yapraklara) ulaşır.
  + Entropy förmülü :
    - Entropy =
    - p(x) bir sınıfın olasılığı.
* **Neden Kullandım:** Anlaşılabilirlik ve Görselleştirme, Çeşitli Veri Tipleriyle Uyumluluk.

**3. Gradient Boosting:**

* **Genel Açıklama:**
  + Gradient Boosting, **makine öğrenimi** algoritmalarından biridir ve genellikle **sınıflandırma** ve **regresyon** problemlerini çözmek için kullanılır. Bu yöntem, bir dizi **zayıf öğreniciyi** ardışık olarak birleştirerek güçlü bir model oluşturur. Gradient Boosting(Gradyan arrtırma)'deki zayıf öğreniciler genellikle **karar ağaçlarıdır yada linner regresiyon.**
  + Her bir yeni model, önceki modelin hatalarını düzeltmeye çalışır ve böylece modelin performansı kademeli olarak iyileşir.
  + **Nasıl Çalışır?**

Gradient Boosting'in temel çalışma prensibi şu şekildedir:

1. **Başlangıç Modeli Oluşturma:**

İlk olarak, basit bir model (örneğin, hedef değişkenin ortalaması) ile başlar.

1. **Hataların Hesaplanması:**

Mevcut modelin tahmin hataları (rezidüeller) hesaplanır.

1. **Yeni Model Eğitimi:**

Yeni bir model, bu hataları tahmin etmek için eğitilir.

1. **Modelin Güncellenmesi:**

Yeni model, önceki modele eklenir. Bu işlem genellikle bir öğrenme oranı (ν) ile ağırlıklandırılır.

1. **Tekrar:** Adımlar belirli bir iterasyon sayısı boyunca veya hata istenilen seviyeye düşene kadar tekrarlanır.

Her iterasyonda, model hataları azaltmak için gradyan inişi kullanır. Bu sayede, model sürekli olarak iyileşir ve daha doğru tahminler yapar.

* **Neden Kullandım:** Yüksek Tahmin Gücü, Esneklik ve Genel Başarı.

**4. Malaya Merkezlik algoritması.**

**Nasıl uygulandı**

1. **verileri yüklenir ve ön işlemler yapılır.**

* Veri seti yüklenir ve önemli sütunlar seçilir.
* Veriler temizlenir ve gruplara ayrılır.

1. **Grafik oluşturulur**
   * Her veri grubu için düğümler (örnekler) arasındaki benzerliklere dayalı grafikler oluşturulur.
   * Düğümler örnekleri, kenarlar ise benzerlikleri temsil eder.
2. **Düğümlerin Ağırlıklarının Hesaplanması**:

* Her düğüm için merkezilik (centrality) değeri, komşu düğümlerle olan ilişkisine göre hesaplanır.

1. **Yeni Bir Örneğin Sınıflandırılması**:

* Yeni bir örnek eklendiğinde, bu örnek dört grafiğe de dahil edilir.
* Her grafikteki merkezilik değeri hesaplanır.

1. **Grubun Belirlenmesi**:

* Yeni örneğin en yüksek merkezilik değerine sahip olduğu grafik seçilir ve bu grafik, örneğin grubunu belirler.

Malatya Algoritması düğüm (v) değerlerinin hesaplanmasında kullanılan algoritma aşağıda verilmiştir.  Bu denklemde, n sırasıyla grafikteki tüm düğümleri temsil eder. Bir düğüm vi için, bitişik düğümlerin kümesi N(vi) olarak temsil edilir.

A graph diagram of a graph

Description automatically generated with medium confidenceψ(𝑣𝑖)=∑ 𝑑(𝑣𝑖) 𝑑(𝑣𝑗)

Malatya Merkezlik algoritmasi çalışma makanizması

Bu modelleri seçmem nedeni, hem basit hem de karmaşık modellerin performansını karşılaştırmak ve en iyi sonuçları elde etmektir.

**Model Ayarlaması ve Değerlendirme**

* **Değerlendirme Metrikleri:** Modelleri değerlendirmek için aşağıdaki metrikleri kullandık:
  + A table with a list of different languages

    Description automatically generated with medium confidence**Karışıklık matrisi:** bir modelin performansını değerlendirmek için kullanılan ve görsel olarak bir tablo şeklinde temsil edilen bir araçtır.
  + **Doğruluk (Accuracy):** Tüm doğru tahminlerin toplam tahmin sayısına oranıdır.

* + **Duyarlılık (Recall):** Gerçek pozitiflerin doğru şekilde tahmin edilen oranıdır.

* + **Kesinlik (Precision):** Pozitif tahminlerin ne kadarının doğru olduğunu **gösterir.**

* + **F1 Skoru (F1 Score):** Kesinlik ve duyarlılığın harmonik ortalamasıdır.

**En İyi Performans Gösteren Model:** **Gradient Boosting** modeli, diğer modellere göre en iyi performansı gösterdi.

**Sonuçlar**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model Performans Karşılaştırması:** | |  |  |  |  | |
| **Model** | **Doğruluk** | **Duyarlılık / Tespit Oranı** | **Kesinlik** | **F1 Score** | | **Karmasiklik matris** |
| Gradient Boosting | 75.13 | 59.98 | 74.98 | 75.13 | [[37.85, 24.875], [0.0036, 37.3]] | |
| Decision Tree | 70.05 | 60.11 | 59.55 | 70.05 | [[48.075, 14.64], [15.24, 22.06]] | |
| Logistic Regression | 68.69 | 59.05 | 55.92 | 68.69 | [[97951, 27467], [34978, 39604]] | |
| Maltya merkezlik | 59.33 | 44.44 | 35.71 | 39.60 | [[47.98, 16.734], [24.497, 13.8]] | |

* **Sonuçlar:**

**1. Gradient Boosting**

**1. Gradient Boosting (Doğruluk: %75.13)**

* **Doğruluk (%75.13):** Gradient Boosting, genel doğruluk açısından en iyi modeldir.
* **Duyarlılık (%59.98):** Model, pozitif durumları tespit etmede makul bir performans göstermektedir.
* **Kesinlik (%74.98):** Yanlış alarmları azaltma konusunda oldukça başarılıdır.
* **F1-Skoru (%75.13):** Doğruluk ve hatırlama arasında iyi bir denge sağlamaktadır.
* **Karmaşıklık Matrisi:** Pozitif sınıfların doğru tespit oranı yüksek (74,575), yanlış pozitif ve negatiflerin sayısı ise genel performans göz önüne alındığında kabul edilebilir düzeydedir (49,750).
* Sonuç: Genel olarak, Gradient Boosting modeli, diğer modellere göre daha dengeli ve güçlü bir performans göstermiştir.

**2. Karar Ağacı (Decision Tree) (Doğruluk: %70.05)**

* **Doğruluk (%70.05):** Gradient Boosting'e kıyasla daha düşük olsa da makul bir doğruluk oranına sahiptir.
* **Recall (%60.11):** Gradient Boosting'den biraz daha iyi bir hatırlama oranı sunar, pozitif sınıfları tespit etme konusunda daha başarılıdır.
* **Precision (%59.55):** Gradient Boosting'e kıyasla daha düşük olup, daha fazla yanlış pozitif sınıf tespit etmektedir.
* **F1-Skoru (%70.05):** Göreceli olarak dengeli fakat Gradient Boosting'in gerisindedir.
* **Karmaşıklık Matrisi:** Yanlış pozitif sınıfların daha yüksek olması (30,575) doğruluk oranındaki düşüşü açıklamaktadır.

**Sonuç:** Bu model, doğruluk açısından iyi performans gösterse de, pozitif tahminlerde Gradient Boosting kadar güçlü değildir.

**3. Lojistik Regresyon (Doğruluk: %68.69)**

* **Doğruluk (%68.69):** Karar Ağacı ve Gradient Boosting'e kıyasla daha düşük doğruluğa sahiptir.
* **Recall (%59.05):** Pozitif sınıfları tespit etme oranı düşüktür ve daha fazla pozitif sınıfı kaçırma eğilimindedir.
* **Precision (%55.92):** Diğer modellere göre daha düşük, yanlış pozitif oranı daha yüksektir.
* **F1-Skoru (%68.69):** Gradient Boosting ve Karar Ağacı'nın gerisindedir.
* **Karmaşıklık Matrisi:** Yanlış pozitif sınıfların artması (34,978) ve doğru pozitif sınıfların azalması modelin performans düşüklüğünü açıklamaktadır.

**Sonuç:** Logistic Regression modeli, diğer modellere göre daha zayıf bir performans göstermiştir.

**4. Malatya Merkezlilik (Doğruluk: %59.30)**

* **Doğruluk (%59.33):** En düşük doğruluk oranına sahiptir.
* **Recall (%44.44):** Pozitif durumları tespit etme konusunda oldukça zayıf bir performans göstermektedir.
* **Precision (%35.71):** Yanlış alarmlar oldukça fazladır ve doğruluk oranı düşüktür.
* **F1-Skoru (%39.60):** Doğruluk ve hatırlama arasında dengesiz bir performans göstermiştir.
* **Karmaşıklık Matrisi:** Yanlış negatiflerin (57,733) ve yanlış pozitiflerin (15,624) yüksek sayısı modelin düşük performansını açıklamaktadır**.**

**Sonuç:** Bu modelin performansı, diğer modellerle kıyaslandığında düşüktür.

**Genel Karşılaştırma:**

Gradient Boosting: Genel performans (doğruluk, hatırlama, doğruluk, F1-Skoru) açısından en iyi modeldir ve en güvenilir sonuçları sunar.

Karar Ağacı (Decision Tree): Kabul edilebilir bir performans sunar ancak Gradient Boosting'in gerisindedir.

Lojistik Regresyon: Daha basit bir seçenek olarak makul bir performansa sahiptir, ancak Karar Ağacı ve Gradient Boosting'e kıyasla daha zayıf kalmaktadır.

Malatya Merkezlilik: Diğer modellere kıyasla oldukça zayıf bir performansa sahiptir ve geliştirilmesi gerekmektedir.

Sonuç olarak, Gradient Boosting, kredi risk analizi gibi projelerde tercih edilmesi gereken en iyi modeldir.

**Deney Sonuçları**

Yapılan çalışma kapsamında, ([**Financial Risk Data - Large**](https://www.kaggle.com/datasets/deboleenamukherjee/financial-risk-data-large)) kaggle’den ait kredi veri seti bilgilerini içeren veriler kullanılarak, veri kalitesi ve model ön işleme çalışmaları gerçekleştirilmiştir. Yeni başvuru sahiplerinin veya aktif kredilerin temerrüt risk oranını değerlendirmek amacıyla, ilgili istatistiksel ve makine öğrenmesi algoritmaları uygulanmış olup, algoritmaların iyi ve kötü müşterileri ayrıştırıcı gücünün belirlenmesi performans ölçüleriyle sağlanmıştır.

Öznitelik seçimi kapsamında istatistiksel hatalardan arınma ve sınıflandırma gücü yüksek olan değişkenlerin belirlenmesi için çeşitli değişken indirgeme tekniklerinden faydalanılmıştır.

İlgili algoritmalarda girdi değişkeni olarak kullanılacak nihai özniteliklerin belirlenmesi için RandomForestClassifier değişken indirgeme tekniği olarak kullanılmış,.

Ham veri kümesinin %70’ı eğitim, %15’si doğrulama ve %15’si test veri seti olacak şekilde ayrılarak, makine öğrenmesi teknikleri bu veri setleri üzerinden gerçekleştirilmiştir.

karmaşıklık matrisinden elde edilen doğruluk, hassasiyet, özgünlük, kesinlik ve F1 skor ölçüleri incelenerek, dört algoritmanın makine öğrenmesi yetenekleriyle gerçekleştirdiği performansları değerlendirilmiştir.

müşterilerinin temerrüt riskini değerlendirmek için en iyi sınıflandırma başarısını her bir veri setinde sağlayan modelin, Gradyan Artırma algoritması olduğu gözlemlenmiştir.

**Kaynaklar**

1. Bahçe Çizer, E. (2017, Aralık). *Makine Öğrenmesi Teknikleriyle Kredi Risk Analizi*.
2. Gursakal, S. (2023, Mart). *Kredi Temerrüt Riskini Tahmin Etmede Makine Öğrenme Algoritmalarının Karşılaştırılması*.
3. Karcı, A., Yakut, S., & Öztemiz, F. (2022, Kasım 27). *Minimum Tepe Örtüsü Probleminin Çözümünde Merkeziyet Değerine Dayalı Yeni Bir Yaklaşım: Malatya Merkeziyet Algoritması*.
4. Keskenler, M. F., Dal, D., & Aydın, T. (2021, Mayıs 2). *Yapay Zeka Destekli ÇOKS Yöntemi ile Kredi Kartı Sahtekarlığının Tespiti*.
5. Kurt, R. (2020, Eylül 18). *BASEL Düzenlemelerine Uygun Olarak Kredi Risk Analizi*.
6. Chow, J. C. K. (2017, April). ***Analysis of Financial Credit Risk Using Machine Learning.***
7. Sakal, M. (2020, Haziran 20). *Makine Öğrenmesi Algoritmaları Kısa Açıklamaları****.***
8. Northern Trust. (2023). *Machine Learning Algorithms for Credit Risk Assessment: An Economic and Financial Analysis*.
9. *Gradient boosting machines, a tutorial( Dec, 2013) (Alexey Natekin)*

**Ekler**

Ek1 – kullanılan ana kütüphaneler :

* from sklearn.linear\_model import LogisticRegression
* from sklearn.ensemble import GradientBoostingClassifier
* from sklearn. tree import DecisionTreeClassifier
* from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

Ek2 – Modellerin tanımı:

models = {  
 "Gradient Boosting": GradientBoostingClassifier(),  
 "Decision Tree": DecisionTreeClassifier(),  
 "Logistic Regression": LogisticRegression()  
}

Ek3– Model eğtim:

model.fit(X\_train, y\_train)

Ek4 – Performans değerlendirme:

accuracy = accuracy\_score(self.y\_test, y\_pred)  
precision = precision\_score(self.y\_test, y\_pred)  
recall = recall\_score(self.y\_test, y\_pred)  
f1 = f1\_score(self.y\_test, y\_pred)  
conf\_matrix = confusion\_matrix(self.y\_test, y\_pred)  
conf\_matrix\_percentage = (conf\_matrix / conf\_matrix.sum())

Ek5 – : Malatya Merkezlik Kabul ve red için benzerlik grafiği oluşturma

def create\_graph(data):  
 graph = nx.Graph()  
 for i in range(len(data)):  
 graph.add\_node(i, features=data.iloc[i].values, similarity\_count=0)  
 for i in range(len(data)):  
 for j in range(i + 1, len(data)):  
 similarity = sum(data.iloc[i].values == data.iloc[j].values)  
 if similarity > 0:  
 graph.add\_edge(i, j, weight=similarity)  
 return graph

Ek6 – : Malatya Merkeziliği Hesaplama

def calculate\_malatya\_centrality(graph):  
 centrality = {}  
 for node in graph.nodes:  
 neighbors = list(graph.neighbors(node))  
 total\_weight = sum(graph[node][neighbor]["weight"] for neighbor in neighbors)  
 centrality[node] = total\_weight / len(neighbors) if neighbors else 0  
 return centrality

Ek7 – :Merkeziyet ve benzerlik kullanarak sınıflandırma

def classify\_with\_centrality(sample, acceptance\_graph, rejection\_graph,  
 acceptance\_centrality, rejection\_centrality):  
 acceptance\_scores = [  
 sum(sample == data['features']) \* acceptance\_centrality[node]  
 for node, data in acceptance\_graph.nodes(data=True)  
 ]  
 rejection\_scores = [  
 sum(sample == data['features']) \* rejection\_centrality[node]  
 for node, data in rejection\_graph.nodes(data=True)  
 ]  
 acceptance\_score = sum(acceptance\_scores) / sum(acceptance\_centrality.values())  
 rejection\_score = sum(rejection\_scores) / sum(rejection\_centrality.values())  
 return 'Accepted' if acceptance\_score > rejection\_score else 'Rejected'

Ek8 – : Grafa yeni düğüm ekleme

def add\_sample\_to\_graph(sample, graph):  
 new\_node\_index = len(graph.nodes)  
 graph.add\_node(new\_node\_index, features=sample, similarity\_count=0)  
 for node in graph.nodes:  
 if node == new\_node\_index:  
 continue  
 similarity = sum([1 for s, g in zip(sample, graph.nodes[node]['features']) if s == g])  
 if similarity > 0

graph.add\_edge(new\_node\_index, node, weight=similarity)  
 graph.nodes[new\_node\_index]['similarity\_count'] += similarity  
 graph.nodes[node]['similarity\_count'] += similarity  
 return new\_node\_index

Ek9 – projenin kodu :

[malek721/KrediRiskML](https://github.com/malek721/KrediRiskML)

Ek10 – kullanılan veri seti :

[Financial Risk Data - Large](https://www.kaggle.com/datasets/deboleenamukherjee/financial-risk-data-large)