



**YILDIZ TEKNİK ÜNİVERSİTESİ  
KİMYA-METALÜRJİ FAKÜLTESİ  
MATEMATİK MÜHENDİSLİĞİ BÖLÜMÜ**

**MATEMATİK MÜHENDİSLİĞİ TASARIM UYGULAMALARI**

**Zenginleştirilen Verinin Model Tahminlemedeki Etkisinin  
Yapay Zeka Modelleri ile İncelenmesi**

Danışman: Doç. Dr. Ülkü YEŞİL

20052100, Canberk Ustaoğlu

İstanbul, 2023

## SAYFA

İÇİNDEKİLER	1
KISALTMA LİSTESİ	2
ŞEKİL LİSTESİ	3
TABLO LİSTESİ	4
ÖNSÖZ	5
ÖZET	6
1.GİRİŞ	7
2.METODOLOJİ	10
2.1 Kullanılan Araçlar	11
2.2 Literatürdeki Model Örnekleri	11
2.3 Benzer Çalışmalar	14
3. Kullanılan Yöntemlerin Matematiksel Açıklaması	18
3.1 Logistic Regression	18
3.2 K-Nearest Neighbor	19
3.3 Support Vector Machine (SVM)	20
3.4 Decision Tree Classifier	22
3.5 Random Forest Classifier (RFC)	23
3.6 Gaussian Naive Bayes (Gaussian NB)	25
3.7 Gradient Boosting (GB)	26
4. Kullanılan Kavramların Açıklanması	28
4.1 StandardScaler ve RobustScaler	28
4.2 Boyut Azaltma Yöntemleri (t-SNE, PCA, Truncated SVD)	28
4.3 Oversampling ve SMOTE	29
4.4 Cross Validation ve k-fold Cross Validation	29
4.5 ROC – AUC Skorlamaları	30
4.6 L1, L2 ve ElasticNet Penalty Skorları	30
5. UYGULAMA	32
5.1 Kodun Açıklanması ve Yöntemlerin Uygulanması	33
6. Sonuçlar	48
Kaynaklar	49
Özgeçmiş	51

## **Kısaltma Listesi**

ML: Makine Öğrenimi (Machine Learning)

DL: Derin Öğrenme (Deep Learning)

RF: Rassal Ormanlar (Random Forests)

RNN: Düzeltilmiş Doğrusal Ünite (Rectified Linear Unit)

VAE: Varyasyonel Otomatik Kodlayıcı (Variational Autoencoder)

SMOTE : Sentetik Azınlık Aşırı Örneklem Tekniği (Synthetic Minority Oversampling Technique)

PCA: Temel Bileşen Analizi Principal Component Analysis

SVD: Kesik Tekil Değer Ayrıştırması (Truncated)

t-SNE: t-Dağıtılmış Stokastik Komşu Gömme

KNN: K-En Yakın Komşu (k-Nearest Neighbour),

DT: Karar Ağaçları (Decision Trees),

CV: Çapraz Doğrulama (Cross Validation),

ROC: Alıcı İşletim Karakteristikleri (Receiver Operating Characteristics),

AUC: Eğrinin Altındaki Alan (Area Under Curve)

Şekil Listesi	Sayfa No
Şekil 2.1 Makine Öğrenimi Uygulamaları tablosu [2].....	10
Şekil 2.2 Anaconda evreni ve alt programları [3].....	11
Şekil 2.3 XGBoost uygulamaları kod örneği.....	12
Şekil 2.4 Rassal Orman uygulamaları kod örneği.....	12
Şekil 2.5 Lojistik Regresyon uygulamaları kod örneği.....	12
Şekil 2.6 Multilayer Perceptron modeli uygulamaları kod örneği.....	12
Şekil 2.7 Autoencoder Mimarisi kurularak modelin uygulaması kod örneği.....	13
Şekil 2.8 LSTM Mimarisi kurularak modelin uygulaması kod örneği .....	13
Şekil 2.9 RNN Mimarisi kurularak modelin uygulaması kod örneği.....	13
Şekil 2.10 SMOTE Tekniğinin Grafiksel Gösterimi [4].....	14
Şekil 2.11 Data Augmentation yöntemlerinden biri olan VAE'nin şekil gösterimi [5].....	15
Şekil 3.1 Sigmoid fonksiyonun grafiksel gösterimi.....	18
Şekil 3.2 KNN algoritması grafik örnek [9].....	19
Şekil 3.3 SVM kernel tiplerine göre sınıflandırma grafikleri [10].....	20
Şekil 3.4 Ensemble metodlarının mantığının görsel karşılaştırmaları [25].....	24
Şekil 4.1 Boyut Azaltımının mantıksal gösterimi [36].....	28
Şekil 5.1 Kütüphanelerin yüklenmesi.....	33
Şekil 5.2 Veri setinin okunması.....	34
Şekil 5.3 Veri setinin incelenmesi:.....	34
Şekil 5.4 Veri setindeki sütunların etiket oranlarının grafikleştirilmesi.....	35
Şekil 5.5 Amount ve time kolonlarının dağılımının dağılımının grafikleştirilmesi.....	35
Şekil 5.6 Amount ve time kolonlarının scale edilmesi.....	36
Şekil 5.7 Veri setinin eğitim ve test setlerine bölünmesi.....	37
Şekil 5.8 Veri setindeki sütunların etiket oranlarının grafikleştirilmesi.....	38
Şekil 5.9 Veri setindeki özelliklerin birbiriyle korelasyonunun incelenmesi.....	38
Şekil 5.10 Veri setindeki bazı özelliklerin sınıf etiketi ile ilişkisinin görselleştirilmesi.....	39
Şekil 5.11 Tüm özelliklerin class=1 etiketli veriye göre dağılımının normal dağılımı .....	39
Şekil 5.12 Tüm özelliklerin aykırı değerlerinin bulunması.....	40
Şekil 5.13 Veri setinde boyut azaltma işlemi uygulanması.....	40
Şekil 5.14 Boyut azaltma uygulanmış verinin görselleştirilmesi.....	41
Şekil 5.15 YZ modellerinin default değerlerinin modele uygulanması ve doğruluk skorları.....	41
Şekil 5.16 YZ modelleri için en iyi parametrelerinin bulunması.....	42
Şekil 5.17 En iyi parametrelere sahip modellerin çapraz doğrulama skorları.....	43
Şekil 5.18 NearMiss çapraz doğrulama skorlarının yazdırılması.....	43
Şekil 5.19 En iyi parametrelere sahip YZ modellerinin auc skorlarının bulunması.....	44
Şekil 5.20 En iyi parametrelere sahip YZ modellerinin ROC grafiklerinin çizdirilmesi.....	45
Şekil 5.21 Lojistik regresyon modelinde en iyi parametrelere sahip modeli ile SMOTE uygulanması öncesi doğruluk tahminlemesi yaptırılması.....	46
Şekil 5.22 SMOTE öncesi modellerin precision-recall skor grafiğinin çizdirilmesi.....	46
Şekil 5.23 SMOTE uygulaması sonrası Lojistik Regresyon doğruluk skorlarının bulunması.....	47

<b>Tablo Listesi</b>	<b>Sayfa No</b>
Tablo 2.1 Hangi Data Augmentation metodlarının hangi modellerle kullanılabilceğinin tablosu [6] ..16	
Tablo 2.2 SMOTE metodu ile ML modellerinin doğruluk artışları üzerine yapılmış bir çalışmanın grafiği [7].....17	
Tablo 6.1: Eğitilmiş bütün modellerin hata ve doğruluk skorlarının tablolaştırılması.....48	

## **ÖNSÖZ**

Bu tasarım projesini yazmam noktasında yardımcıları için sevgili Tasarım Projesi Danışman Hocam Ülkü Yeşil'e,

AIBOSS Laboratuvarı'nda yapay zekâ uygulamaları üzerine mentorluğumu yürüten sevgili Hocam Alev Taşkın'a,

Proje konusundaki tüm yardımları için ayrı ayrı teşekkürler.

## ÖZET

Yapay Zeka ile Banka işlemleri hareketlerinin zenginleştirilmiş verisine ikili tahminleme yöntemlerinin uygulanması ve normal haline göre karşılaştırılarak yorumlanması işlemi gerçekleştirildi.

Bu Tasarım Projesi'nin oluşturulmasında Credit Fraud Data olarak isimlendirilen ve evrensel olarak tanınan, üzerine pek çok model ve makaleler yazılabilen şekilde temizlenmiş ve düzenlenmiş, bilinen bir veri seti kullanıldı. Bu veri setinde, 30 kolon ve yaklaşık 30.000 satır veri var. Kolonlar şu şekilde, Amount ve time olarak total miktar ve hangi zamanda yapıldığının bilgisi, 28 adet yapılan işlemin nümerik değeri ve Class olarak ikili sınıflandırmaya işlemenin şüpheli mi yoksa şüphesiz olarak mı tespit edildiğinin bilgisi. Burada 30 bin kişinin işlem bilgileri yapay zeka modelleri ile incelenerek en iyi hangi modelin hangi parametreler ile tespit edilebileceğini seçme işlemi gerçekleştirildi.

Fakat özellikle şüpheli hareket örneğinin az olması ve gerçek dünya durumlarını düşünürsek, şüpheli banka hareketlerinin sayısının normalde de az olduğunu düşünürsek modellerin şüpheli olmayan hareket olarak tespit noktasında bir yakınsama durumu şüphesiz karşılaşan ve hem kişilere hem de firmalara problem yaratan bir durum olmasına sebep olmakta. Bunun için de veri zenginleştirirken minör etiketli verilere sanal örnekler üretmeyi sağlayan bilinen bir metod mevcut. Bu metodun ismi SMOTE. Bu metodu kullanarak gerçekten de böyle bir problemin çözümü noktasında veri zenginleştirme işlemi, aynı modellerin aynı parametreleri mevcutken uygulansa ne olurdu görebilmek adına kullanıldı ve karşılaşmaları yapıldı. Bu noktada, verideki accuracy ve hata skorları, çapraz doğrulama skorları ve roc-auc skorları baz alındı ve bir karşılaştırma tablosu oluşturuldu.

Hazırlanan karşılaştırma tablosu, doğruluk ve diğer metriklerin baz alındığı tablolar sonucu iki sonuca varıldı: SMOTE uygulaması bu tip bir veride daha iyi sonuçlar almamızı katkı sağladı. İkinci sonuç ise, Rassal Ormanlar kullanılarak en iyi tahminleme yürüten model üretildi.

## ABSTRACT

Artificial Intelligence has been used to analyze and interpret the enriched data of banking transactions by applying binary prediction methods to it and comparing it with its normal behavior.

In the construction of this Design Project, we used a globally known dataset called Credit Fraud Data, which has been cleaned and organized in such a way that many models and articles can be written on it. This dataset has 30 columns and about 30,000 rows of data. The columns are as follows: Amount and time as the total amount and the time of the transaction, 28 numeric values of the transaction, and Class as the binary classification of whether the transaction is suspicious or unsuspicious. Here, the transaction information of 30 thousand people was analyzed with Artificial Intelligence models and the process of deciding which model could be best detected with which parameters was performed.

However, especially if we consider the scarcity of suspicious transactions and real-world conditions, the convergence of the models at the point of detection of non-suspicious transactions is certainly a problem for both people and companies, given that the number of suspicious bank transactions is relatively low. For this issue, there is a well-known method for enriching the data by generating virtual samples for minor labeled data. This method is named SMOTE. By using this method, it has been used and then compared to see what would happen if the data enrichment process was applied with the same parameters of identical models in order to solve such a problem. At this point, a comparative table was created using the accuracy and the error scores, the cross-validation scores and the ROC-AUC scores from the data as a baseline.

As a result of the comparison table and the tables based on accuracy and other metrics, two conclusions were reached: SMOTE helped us to get better results on this type of data. The second conclusion is that the model with the best prediction performance was produced by using Random Forests.

## 1. GİRİŞ

Bir Yapay Zeka tahminlemesi yapılmadan önce, öncelikli olarak bir veri incelemesi ve bu verinin isterlere giden yolunun haritası çizilmeye çalışılır. İstenen çıktı göz önünde tutularak modelin bir kümeleme mi, sınıflandırma mı yoksa bir regresyon modeli mi olduğu anlaşılmaya çalışılır. Bu üç ana başlıkta toplanan ve evrensel kabul edebileceğimiz üç ayrı tipteki tahminleme sistemlerinin incelenmesi, bu verilerin veri bilimi uygulamaları çerçevesinde düzenlenmeleri, istatistikci açıdan incelemeleri ve gerekirse yeni özelliklerin elimizdeki verilerin kullanılarak üretilmesi, daha sonrasında da uygun ML ve DL modelleri ile tahminlenmesi işlemleri gerçekleştirilerek bir yol izlenir. Bu modellerin doğrulukları, tahmin başarıları, geleceğe yönelik tahminlerinin başarısının ölçülmesi ve modelin aşırı uyum(overfitting) veya yetersiz uyum (underfitting) durumları göstermesine dair kontrollerin çapraz doğrulama (cross-validation) metodları ile kontrollerinin gerçekleştirilmesi ve gerekirse geçmişe dönük parametre güncellemleri ile süren bir sürecin tamamını bir YZ Tahminlemesi süreci olarak adlandırabiliriz.

Bu Tasarım Projesi’nde, sınıflandırma tahminlemesine uygun isterleri olan bir veri ile çalışıldı. Bu veride, öncelikli olarak özellik mühendisliği yaptı ve bilinmeyen satır değerleri kontrolleri gerçekleştirildi. Sonrasında, bu değerlerin ikili sınıflandırılmış etiketlerinin nümerik dağılımının incelenmesi işlemleri gerçekleştirildi. Dağılım olarak ciddi bir yüzdelik fark olduğu görüldüğü noktada, modelin doğruluğunu artıtabileceğimize inandığımız bir Rastgele Aşırı Örneklem (Random Oversampling) yöntemi olarak kullanılan Sentetik Azınlık Aşırı Örneklem Tekniği (Synthetic Minority Oversampling Technique -SMOTE) metodunu kullanarak elimizde özellikle 1 etiketli verinin sentetik üretimini gerçekleştirmeye planları yapıldı. Bu noktaya gelmeden öncesinde elbette bir modelin doğru şekilde tahminleme yapabilmesine fayda sağlayabilecek ve aykırı değerlerin bariz ezberleme sebebi olmamasını sağlama adına sınırlandırma metodlarından StandardScale ve Robust Scale metodları Amount ve Time kolonlarına uygulandı bu sayede ezberleme ihtiyimali en yüksek ihtimali kısımlar (-1,1) aralığında sınırlandırıldı. Bu metodlardan daha sonrasında detaylı biçimde bahsedilecek.

30 bin banka işlemleri verisinin %99.83’ü temiz, %0.17’si kirli veri olduğundan ötürü, modelimizi train ve test setlerine ayırirken de daha dengeli bir biçimde ayrılmasını sağlamamız öncelikli bir problemdir. Bu sebeple StratifiedKFold fonksiyonundan yardım alınarak train/test dağılımı yapıldı. Verinin 5’tे 4’ü train, 5’tे 1’i test seti olarak seçildi. Rastgele dağıtılan bu setler ile ML ve DL modelleri eğitimleri gerçekleştirilecek.

Scale ve train/test dağılımları yapılan bu veride model testleri gerçekleştirilmeden önceki son aşamada, görsel olarak çıkarımlar yaparak hangi modellerin daha iyi olabileceği tahmini çıkarımlar kullanarak daha emin olabilmek amacıyla istatistikci bazı tablolar oluşturulup incelemeleri gerçekleştirildi. İlk önce, korelasyon matrisi oluşturularak her bir kolonun, kendi harici diğer kolonların her biriyle düz-ters korelasyon oranları incelendi. Bu sayede gerekirse bir kolonu dataset üzerinden silmemiz gerekip gerekmemi incelemelerini sağlayabiliyoruz. İncelemelerimizin sonucu, herhangi bir kolon silme işlemeye gerek olmadığını gördük. Sonrasında, İstatistik’ten de aşina olduğumuz sağa çarpıklık-sola çarpıklık durumlarının incelemeleri gerçekleştirildi. Daha sonra, elimizdeki kolonların değer dağılımının normal dağılıma ne kadar benzettiğini karşılaştırmaları scipy.stats kütüphanesinden norm fonksiyonu kullanılarak kontrol edildi. Z-skor benzerliği durumları kontrolleri sonrasında bir istatistikci çıkarımı daha görebilmek açısından V1 ve V28 arasındaki kolonlar için aykırı değerlerin tanımlanması, bu değerlerin kartil aralıkları ve her bir kolonun alt-üst sınırları tespitlenmesinin gerçekleştirilmesi, bu değerlerin dışında kalan aykırı değerlerin sayısının ve bu değerlerinin kendisinin ayrı ayrı belirtilmesi işlemleri kolonlar için ayrı ayrı gerçekleştirilmişdir.

Sonrasında, boyut azaltma teknikleri kullanarak, veri kümesini iki boyutlu bir uzayda görselleştirmeyi ve dolandırılıcılık sınıfını vurgulamak amacıyla, t-SNE, PCA ve Truncated SVD metodları kullanılarak veri setlerinin görselleştirilmesi sağlanmıştır. Verinin tipi ve veride yapılan görselleştirmeler sonucunda şu modellerin kullanılmasına karar verilmiştir:

- Lojistik Regresyon (Logistic Regression)
- K-En Yakın Komşu (K-Nearest Neighbour)

- Destek Vektör Sınıflandırıcısı (Support Vector Classifier)
- Karar Ağacı Sınıflandırıcısı (Decision Tree Classifier)
- Rassal Orman Sınıflandırıcısı (Random Forest Classifier)
- Gaussian Naive Bayes Sınıflandırıcısı
- Gradyan Güçlendirme Sınıflandırıcısı (Gradient Boosting Classifier)

Bu yapay zeka metodları, öncelikli olarak SMOTE ile artırılmış normal veride uygulanmıştır. Uygulama esnasında, öncelikli olarak bu modellerin tamamı, Python sklearn kütüphanesinde tanımlanan default parametreler üzerinde herhangi bir değişiklik yapılmadan, sonrasında ise GridSearchCV kütüphanesi kullanılarak modeller üzerinde çok fazla deneme yapıp en uygun parametre dağılımlarını bulabileceği biçimde ayarlama yapılacak bir sistem oluşturulup modellerin doğruluk yüzdeleri daha da yukarıya çekilmiştir. Ayrıca bu parametre bilgileri tekrar kullanılmak üzere kaydedilmiştir. Bu parametreler, SMOTE metodu uygulanmış veriye de bir daha uygulanarak karşılaştırmaları yapılacak.

Sonrasında karşılaştırma işlemleri olarak, modelin overfit veya underfit durumda olup olmadığını anlayabilmek açısından baktığımız ilk şey Çapraz Doğrulama (Cross-Validation) skorlarının kontrol edilmesi olmaktadır. Bu noktada bilgisayar kaynak gücü de hesaba katılarak en uygun ve en çok tercih edilen metodlardan biri olan 5-Fold Cross Validation metodu uygulanmıştır. Nasıl bir mantıkla çalıştığı matematiksel açıdan detaylandırarak anlatılacaktır.

Öğrenme eğrisinin nasıl bir şekilde ilerleyebildiğini görebilmek adına Öğrenme Eğrisi tablosunda ROC-AUC Skorlamalarının kontrolü, modelin hata skorları dışında ne kadar başarılı olduğunu görebilmek açısından önemli bir parametredir. Süreç içerisinde nasıl bir eğri şeklinde ve ne kadar başarılı öğrenebildiğini görmek açısından grafikleştirilerek kontrolleri işlemleri yapıldı. Matematiksel açıdan ROC-AUC skorlarının nasıl yapıldığı daha detaylı biçimde açıklanacak.

Veri üzerinde izlenen bütün bu yol haritası, gerçekçi bir karşılaştırma yapabilmek için aynı parametrelerle, aynı boyuttaki train/test veri setleri kullanılarak sadece dolandırıcılık etiketli veri seti üzerinde SMOTE metodu kullanılarak Random Oversampling işlemi uygulanmış biçimde tüm süreçler birebir uygulanarak skorlamalar yapılmış ve tablo biçiminde karşılaştırmaları yapılmıştır. Süreç içerisinde izlenen tüm yol bu sekildedir.

Banka işlemlerinde şüpheli işlem tespitlerinin bulunmasında genellikle kullanılan modeller, Google Cloud AutoML, H2O.ai, Auto-Sklearn, TPOT, Microsoft Azure AutoML gibi programlarla veriyi vermek ve özellik artırımı yaparak tahminlemeyi yükseltmesini beklemektir. Ancak bu noktada bu programlar çok cimri bir yaklaşım sergilerler ve doğruluğu artıran her şeye tamam olabilirler. Örnek vermek gerekirse, bir özelliğin kökünü alıp yeni bir özellik olarak eklemek, eğer tahminlemede iyileşme varsa bununla yola devam etmek çok normal bir durumdur. Günü kurtarmak için iyi gözükebilen bu tarz hareketler ise, uzun vadede bazı özelliklerin ezberlenmesi ve sonunda neredeyse sır o özelliğe bağlı olarak karar veren bir programa dönüşmesine sebep olabilir.

Bunlara manuel hazırlanan alternatifler ise Oversampling metodlarının kullanılması ile yapılan geliştirmelerin modele eklenmesi üzerine araştırma geliştirme çalışmalarında mevcut. Biz de bu noktada bir oversampling metodu olan SMOTE ile sektörde uyum sağlayıp, üzerine özellikle scale işlemlerinde ve gereksiz özelliklerin eklenmesi değil, aksine ağırlık değerlerinin azaltılması ve bazen mümkünse tamamen silinmesi işlemlerinin uygulandığı bir model oluşturma ve bunların karşılaştırılması işlemlerinin gerçekleştirilmesi üzerine çalışıldı.

Verinin çok boyutlu olduğu bilindiğinden ötürü öncelikli olarak bir boyut azaltma işlemi yapmamız gereğinin bilinciyle hareket edildi. Ayrıca outlier durumlarının belirli özelliklerde ciddi bir ezber durumu yaratmaması adına scaling işlemleri de model çalışmaları öncesinde uygulanmaya özen gösterildi.

Hangi modellerin seçilmesi iyi olabilir noktasında, optimal açıdan bakarsak tahmin gücü ve bilgisayar gücü arasında iyi bir denge yakalayabilecek, benzer çalışmalar incelendiğinde kullanıldığı görülen modeller kullanılmaya özen gösterildi. Bu noktada seçilen modeller: Lojistik Regresyon Sınıflandırıcısı, K-en Yakın Komşular, Destek Vektör Sınıflandırıcısı, Karar Ağaçları Sınıflandırıcısı, Rassal Ormanlar Sınıflandırıcı, Gaussian Naive Bayes Sınıflandırıcısı ve Gradyan Boosting Sınıflandırıcısı olarak seçildi. Sınıflandırma problemlerinde geniş bir yelpazede, ağaç tabanlı, öklididen

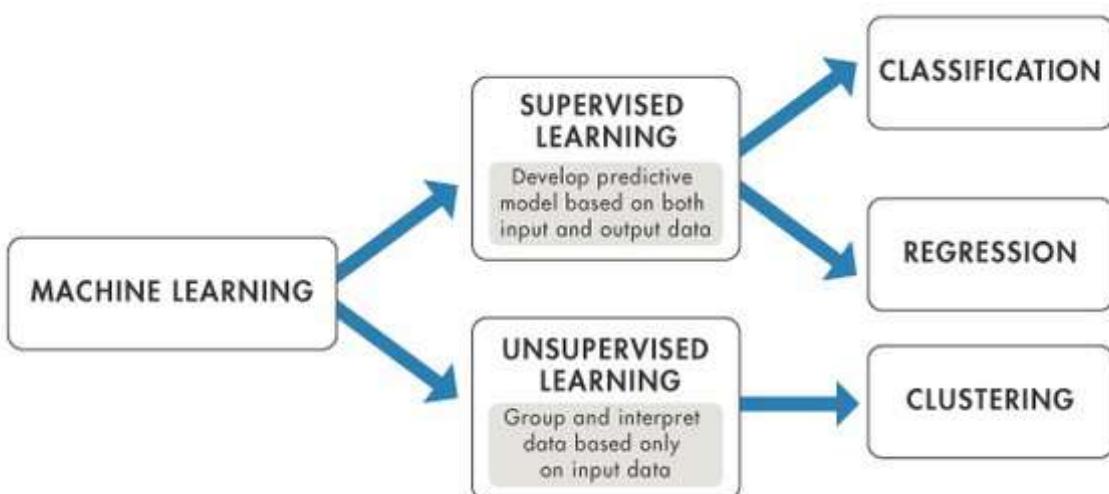
tabanlı veya olasılıksal tabanlı yaklaşabilen bu modeller ile hangi yaklaşımın da aslında bu tarz bir probleme daha uygun olabileceğini de görebilmek ve inceleyebilmek istedim. SMOTE seçim sebebi ise, etiketli verilerin incelendiğinde özellikle 1 yani şüpheli etiketi mevcut verinin oranının 0 etiketli veriye oran olarak çok az olması ve bu etiketli verinin zenginleştirilerek alarm durumuna geçebilmesi için görmesi gereken veriden daha fazla görebilmesini sağlayabilmekti.

Buradan sonraki bölümlerde metodoloji ve kullanılan araçlardan kısaca bahsedilecek, literatürdeki model örneklerinin görsel anlatımı yapılacak, sonrasında anlaşılabilirliği artırmak adına projede kullanılan istatistikî ve yapay zeka ile alakalı terimlerin matematiksel açıklamaları yapılacak, daha sonrasında kod bloğu gösterimi açıklamalı olarak yapılarak hangi işlemlerin yapıldığı ve ne amaçla yapıldığı, işlemler sonucu hangi çıktıların alındığı detaylandırılacak ve son kısımda Kaynaklar kısmı ile alıntı yapılan veya anlatılan kısımların nereden alındığı belirtilecek [1].

## 2. METODOLOJİ

Buradaki çalışma, özellikle sınıflandırma problemlerinde görülen en büyük problemlerden birine eğiliyor: sınıflandırma problemlerinde etiketli verilerinbazısında oransal olarak inanılmaz düşük oranlar var ise, daha doğru tahminlemeler oluşturabilmek için yapılan veri zenginleştirme işlemleri bekleneni verebilecek mi görebilmek açısından bir çalışma yürütüldü. GridSearchCV metodu ile en başarılı makine öğrenimi metodlarının en başarılı parametreleri seçimi sağlandıktan sonra, bu doğruluğu artırabilir bir ekleme olabilir mi görebilmek açısından Random Oversampling metodlarından bu veri için en uygun olanını uygulayıp değişimin ölçülmesi amaçlanmaktadır. Bu değişimin ölçülmesi, ROC-AUC Skorları, Cross-Validation skorları, Accuracy skorları, MSE-RMSE-MAPE-MAE gibi error skorlarının karşılaştırılmaları ile gözlemlenmeye çalışılacak.

Eğitim işlemi sonrasında karşılaştırma olarak görebileceğimiz doğruluk skorları, hata skorları ve verinin tipine uygun belirteçler kullanılarak karşılaştırmalar yapılacak, zenginleştirme iyi bir sonuç veriyor mu, gerçekten doğru bir uygulama metodu mu gibi sorulara cevaplar aranacak.



Şekil 2.1 Makine Öğrenimi Uygulamaları tablosu [2]

Makine Öğrenimi kavramını detaylandırmak gerekirse, böyle bir ağaç yapısı ile belirtilmesi üzerinde çalışan, verinin belirli bir kısmının öğretildip yeni gelecek verileri tahminleyebileceği, bu tahminlemenin ne kadar doğru olacağının ölçümünü ise hem train hem test setinin üzerindeki kontrolleri ve yeni gelecek veri üzerindeki tahminleme işlemlerini, İstatistikci açıdan ne yapmasını öğrettiğimiz bir bilgisayar sistemidir. Bu noktadaki kavramları tanımlamamız gereklidir.

- **Supervised Learning:** Verileri sınıflandırmak veya sonuçları doğru bir şekilde tahmin etmek için algoritmaları eğitmek üzere etiketli veri kümelerinin kullanılmasıyla tanımlanır. Denetimli öğrenmede, modelleri istenen çıktıyı üretecek şekilde eğitmek için bir eğitim seti kullanılır. Bu eğitim veri kümesi hem doğru hem de yanlış çıktıları içerir ve modelin zaman içinde gelişmesine olanak tanır. Kayıp fonksiyonu, algoritmanın doğruluğunu değerlendirmek için kullanılır ve hata uygun şekilde en aza indirilene kadar ayarlanır.

Denetimli öğrenme, veri madenciliğinde sınıflandırma ve regresyon olmak üzere iki tür probleme ayrılabilir:

- Sınıflandırma
  - Test verilerini sınıflandırmak ve belirli gruplara ayırmak için bir algoritma kullanılır. Veri kümesindeki belirli öğeleri tespit eder ve bunların nasıl etiketlenmesi veya tanımlanması gerektiği konusunda bazı yargılara varmaya çalışır. Destek vektör makineleri (SVM), karar ağaçları, K-en Yakın Komşu(K-Nearest Neighbor) ve Rassal Orman(Random Forest) en çok kullanılan sınıflandırma tekniklerinden bazlılardır.

- Regresyon
  - Bağımlı ve bağımsız değişkenler arasındaki ilişkiyi keşfetmek için regresyon kullanılır. Bir şirketin satış geliri gibi tahminler üretmek için yaygın olarak kullanılır. Popüler regresyon teknikleri arasında Lineer Regresyon ve Polinomal Regresyon yer alır.
- 1.
- **Unsupervised Learning**, makine öğrenimi tekniklerini kullanarak etiketlenmemiş bilgileri analiz eder ve kümeler. İnsan etkileşimine gerek kalmadan bu algoritmalar gizli kalıpları veya veri gruplamalarını ortaya çıkarır. Bilgideki benzerlikleri ve zıtlıkları tespit etme kapasitesi nedeniyle keşifsel veri analizi, çapraz satış teknikleri, tüketici segmentasyonu ve resim tanımlama için mükemmelidir.
  - **Clustering:** aynı gruptaki veri noktalarının diğer gruptardaki veri noktalarından daha benzer olması için bir popülasyonu veya veri noktaları kümesini birçok gruba bölmeye işlemi. Başka bir deyişle, amaç benzer özelliklere sahip grupları ayırmak ve bunları kümelere atamaktır.
  - Hierarchical Clustering: Hiyerarşik Küme Analizi (HCA) olarak da bilinir, iki türe ayrılabilen denetimsiz bir kümeleme tekniğidir: Yiğilmalı(Agglomerative) ve Bölünmeli(Divisive) kümeleme.

## 2.1 Kullanılan Araçlar

Bu çalışmada kullanılan kodlama dili Makine Öğrenmesini en rahat ve kapsamlı şekilde uygulayabileceğimiz Python'dur. Python, web siteleri ve uygulamalar oluşturmak, işlemleri otomatikleştirmek ve veri analizi yapmak yaygın olarak kullanılan bir programlama dilidir. Python genel amaçlı bir programlama dilidir, bu da çok çeşitli uygulamalar geliştirmek için kullanılabileceği ve belirli bir konuya özel olmadığı anlamına gelir.

Çalışmanın sürdürdüğü program Spyder ve Jupyter Notebook'tur. Bu programlar Anaconda Navigator programından temin edilebilir



**Şekil 2.2** Anaconda evreni ve alt programları [3]

## 2.2 Literatürdeki Model Örnekleri

Yapılacak çalışmada, veri zenginleştirme ve veri artırma işlemleri için Data Augmentation tekniklerinden biri olan SMOTE teknigi, ikili sınıflandırma verisi üzerinde kullanılacak. Elimizdeki tablo tipi verilerde, bu hem ML tabanlı hem DL tabanlı yöntemler olabilir.

SMOTE teknigi ile, sınıflandırma problemlerinin en büyük problemlerinden biri olan eşit olmayan etiketsiz dağılım problemlerine daha makul bir yaklaşım gerçekleştirmek istiyoruz. Örneğin, bir kredi kartı dolandırıcılığı verisini ele almamız gerekirse, yaklaşık %99.80'i şüpheli olmayan hareketler olarak etiketli olabilir. %0.20'lik kısım ile şüpheli olarak etiketli veri olsun. Böyle bir durumda ise, verinin ciddi bir biçimde şüpheli hareketten ziyade normal hareket olarak tanımlamaya yakınsaması durumu mevcut. Olaylara daha doğru bir bakış açısı getirebilmesi açısından ise biz de az etiketli olan kısımda sentetik bir zenginleştirme yapmanın etkilerini, normal durum ile karşılaştıracağız.

Veri artırımı sonrasında, model inşası için kullanılacak önemli modeller Logistic Regression, K-Nearest Neighbor, Support Vector Classifier, Random Forest Classifier, Decision Tree

Classifier, Gradient Boosting Classifier gibi makine öğrenimi modelleri ve Multi-Layer Perceptron, Autoencoder, LSTM, RNN gibi Derin Öğrenme modelleri olarak ilerleyecek. Burada kullanılacak dil Python3, kullanılacak kütüphaneler de Sklearn ve Tensorflow ağırlıklı olacak. Temel Model mimarilerinden nasıl kullanıldıklarına dair bazı örnekler vermek gereklidir:

```
xgb = XGBClassifier(objective='binary:logistic', reg_lambda = 10, scale_pos_weight = 3,
                     subsample = 0.9, colsample_bytree= 0.5, n_estimators = 200, learning_rate = 0.1, seed = 42)
xgb.fit(X_train, y_train, verbose=True, early_stopping_rounds = 40, eval_metric = 'aucpr', eval_set=[(X_test, y_test)])
y_pred = xgb.predict(X_test)
```

**Şekil 2.3** XGBoost uygulamaları kod örneği

```
rfc = RandomForestClassifier(n_estimators = 30, criterion='entropy', max_depth=10, bootstrap=True, random_state=42)
# n_estimators decision tree ağaç sayısını belirtiyor
# max_depth en fazla ne kadar derin ağaç oluşturabilir belirtiyor
# bootstrap true demek tek seferde bütün veriyi almayıorum demek, bu da decision tree ve rfc arasındaki en önemli fark zaten
rfc.fit(X_train, y_train)

y_pred = rfc.predict(X_test)
```

**Şekil 2.4** Rassal Orman uygulamaları kod örneği

```
logistic_reg = LogisticRegression(penalty='l2',random_state = 42)

logistic_reg.fit(X_train, y_train)

y_pred = logistic_reg.predict(X_test)
```

**Şekil 2.5** Lojistik Regresyon uygulamaları kod örneği

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(units=6, kernel_initializer='uniform', activation = 'relu', input_dim = 11))
model.add(Dense(units=6, kernel_initializer='uniform', activation = 'relu'))
model.add(Dense(units=1, kernel_initializer='uniform', activation = 'sigmoid'))

model.compile(optimizer = 'adam', loss = 'binary_crossentropy' , metrics = ['accuracy'])

model.fit(X_train, y_train, epochs=50)

y_pred = model.predict(X_test)

y_pred = (y_pred > 0.5)

conf_matrix = confusion_matrix(y_pred, y_test)

print(conf_matrix)
print(classification_report(y_pred, y_test))
```

**Şekil 2.6** Multilayer Perceptron modeli uygulamaları kod örneği

```

from tensorflow.keras import layers, Model

class AnomalyDetector(Model):
    def __init__(self):
        super(AnomalyDetector, self).__init__()
        self.encoder = tf.keras.Sequential([
            layers.Dense(32, activation="relu"),
            layers.Dense(16, activation="relu"),
            layers.Dense(11, activation="relu")
        ])

        self.decoder = tf.keras.Sequential([
            layers.Dense(16, activation="relu"),
            layers.Dense(32, activation="relu"),
            layers.Dense(11, activation="sigmoid")]
        )

    def call(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded

autoencoder = AnomalyDetector()

autoencoder.compile(optimizer='adam', loss=tf.keras.losses.CosineSimilarity(axis=1))
#autoencoder.compile(optimizer='adam', loss='binary_crossentropy', metrics = ['accuracy'])

early_stopping = tf.keras.callbacks.EarlyStopping(patience=30)

```

```

history = autoencoder.fit(
    X_train, X_train, # Use X_train as both input and target (since it's an autoencoder)
    epochs=100,
    batch_size=64,
    validation_data=(X_test, X_test),
    callbacks=[early_stopping],
    shuffle=True)

```

**Şekil 2.7 Autoencoder Mimarisi kurularak modelin uygulaması kod örneği**

```

regressor = Sequential()
regressor.add(LSTM(units=8,
                   return_sequences=True,
                   input_shape=(x_train.shape[1],1)))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=8,
                   return_sequences=True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=8))
regressor.add(Dropout(0.2))
regressor.add(Dense(units=1))
regressor.compile(optimizer= "RMSprop",
                  loss= "mean_squared_error")
regressor.fit(x_train,
              y_train,
              epochs=100,
              batch_size=32)

```

**Şekil 2.8 LSTM Mimarisi kurularak modelin uygulaması kod örneği**

```

# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import SimpleRNN
from keras.layers import Dropout

# Initialising the RNN
regressor = Sequential()

# Adding the first RNN Layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second RNN Layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third RNN Layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth RNN Layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 250, batch_size = 32)

```

**Şekil 2.9 RNN Mimarisi kurularak modelin uygulaması kod örneği**

Bu tarzda python3 ile yazılmış yapay zeka modellerinin gelişmiş mimari çözümleri ile elimizdeki problemlere çözüm arama işlemleri gerçekleştirilebilir. Bu işlemler hem zenginleştirilmiş veride hem de zenginleştirilmemiş veride ayrı ayrı gerçekleştirilecek böylece gerçek bir karşılaştırma yapma imkanı bulabileceğiz.

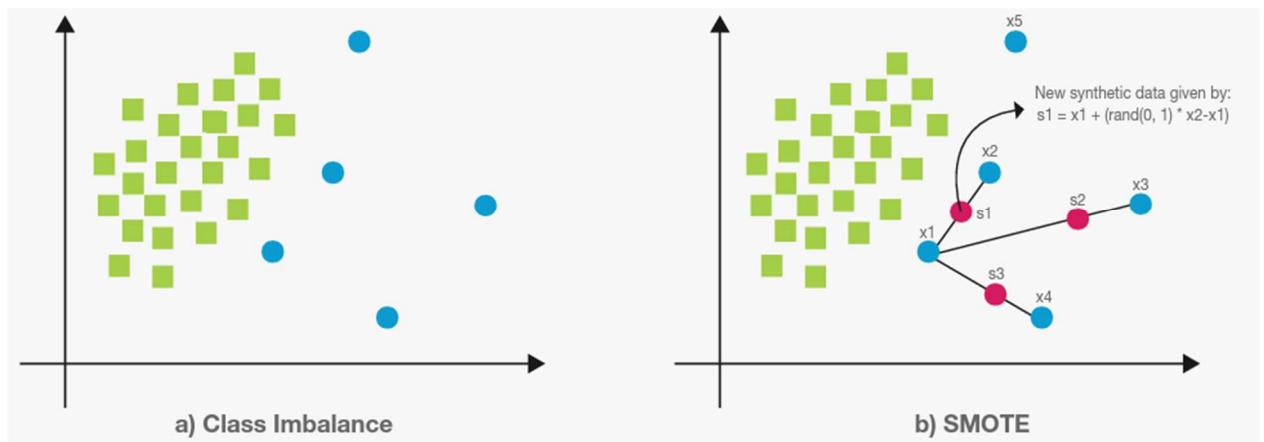
## 2.3 BENZER ÇALIŞMALAR

### SMOTE

Random oversampling is prone to overfitting as the minority class samples are replicated. Overfitting can be avoided by SMOTE. SMOTE stands for Synthetic Minority Oversampling Technique. It creates new synthetic samples to balance the dataset.

SMOTE works by utilizing a k-Nearest Neighbor algorithm to create synthetic data. Samples are created through the following steps:

- Identify the feature vector and its nearest neighbor
- Compute the distance between the two sample points
- Multiply the distance with a random number between 0 and 1
- Identify a new point on the line segment at the computed distance
- Repeat the process for identified feature vectors

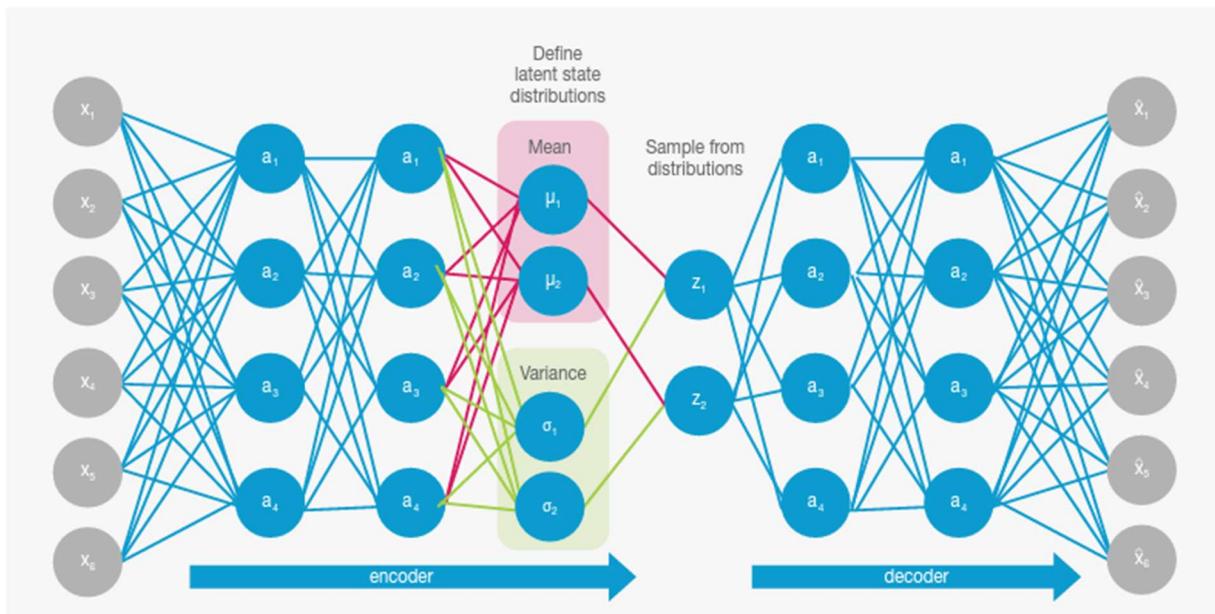


**Şekil 2.10** SMOTE Tekniğinin Grafiksel Gösterimi [4]

Benzer bir literatür çalışması olarak SMOTE ile nasıl veri üretebileceği makalesinde daha detaylı olarak bahsedilmekte. Elimizdeki veride eğer özellikle sınıflandırılan veride dengesiz dağılım mevcutsa çok iyi iş yapabilen bir metod. Sınıflandırma problemlerinde veri artırımı işlemleri için yardımcı olabilen bir durum [5].

A Variational Autoencoder (VAE), with a slight variation in the architecture, provides a probabilistic method for describing an observation in latent space. Thus, rather than building an encoder that outputs a single value to describe each latent state attribute, VAE provides a probability distribution for each latent attribute. By constructing the encoder model to output a range of possible values (a statistical distribution), sampling can be randomly done from the distribution so that it can be fed into the decoder model.

The latent space is continuous in nature. VAE performs sampling by outputting a 2-dimensional vector (mean and variance) from a random variable. The vector is then used to get a sampled encoding which is passed to the decoder. As encodings are generated from a distribution with the same mean and variance as those of the inputs, the decoder learns from all nearby points referred to as the same latent space.



**Sekil 2.11** Data Augmentation yöntemlerinden biri olan VAE'nin şkil gösterimi [5]

Variational Autoencoder metodu, Derin Öğrenme metodu olarak veri artırımı yapabilme noktasında çok başarılı bir yöntem. Bu kısımdan bahsedilen kaynak da [5] kısmında belirtildi. VAE'nin SMOTE karşısında dezavantajı, tüm verilerde sanal eklemeler yapması ve oransal olarak az oranlı etiket verilerinin oran artışı açısından ek bir faydası olmaması.

**Tablo 2.1** Hangi Data Augmentation metodlarının hangi modellerle kullanılabileceğinin tablosu [6]

method	noise removal	dimension reduction	uses classifier	componentwise sampling	ordinary sampling	memetic	density estimation	density based	changes majority	uses clustering	borderline	application	method	noise removal	dimension reduction	uses classifier	componentwise sampling	ordinary sampling	memetic	density estimation	density based	changes majority	uses clustering	borderline	application
1 SMOTE (2002)[20]													44 SMOTE-OUT (2014)[56]												
2 SMOTE-TomekLinks (2004)[8]	×		×										45 SMOTE-Cosine (2014)[56]												
3 SMOTE-ENN (2004)[8]	×		×										46 Selected-SMOTE (2014)[56]												
4 Borderline-SMOTE1 (2005)[47]			×										47 MWMOTE (2014)[7]												
5 Borderline-SMOTE2 (2005)[47]			×										48 PDFOS (2014)[41]												
6 AHC (2006)[25]													49 IPADE-ID (2014)[73]												
7 LLE-SMOTE (2006)[98]	×												50 RWO-sampling (2014)[106]												
8 cluster-SMOTE (2006)[24]													51 NEATER (2014)[4]												
9 distance-SMOTE (2007)[14]			×										52 SDSMOTE (2014)[67]												
10 ADASYN (2008)[48]			×										53 DSMOTE (2014)[76]												
11 SSMIO (2008)[15]	×		×										54 G-SMOTE (2014)[91]												
12 polynom-fit-SMOTE (2008)[43]													55 NT-SMOTE (2014)[102]												
13 Stefanowski (2008)[94]	×		×										56 SSO (2014)[88]												
14 ADOMS (2008)[96]			×										57 Supervised-SMOTE (2014)[51]												
15 Safe-Level-SMOTE (2009)[11]			×										58 DEAGO (2015)[9]												
16 MSMOTE (2009)[52]	×												59 Gazzah (2015)[44]												
17 ISOMAP-Hybrid (2009)[46]	×												60 MCT (2015)[54]												
18 DE-overSampling (2010)[22]													61 ADG (2015)[80]												
19 CE-SMOTE (2010)[23]													62 SMOTE-IPF (2015)[89]												
20 Edge-Det-SMOTE (2010)[55]													63 KernelADASYN (2015)[95]												
21 SMOBD (2011)[17]	×												64 MOT2LD (2015)[101]												
22 SUNDO (2011)[18]													65 V-SYNTH (2015)[103]												
23 MSYN (2011)[36]													66 Lee (2015)[62]												
24 LN-SMOTE (2011)[75]													67 SPY (2015)[26]												
25 CBSO (2011)[5]													68 SMOTE-PSOBAT (2015)[66]												
26 E-SMOTE (2011)[28]	×		×										69 OUPS (2016)[87]												
27 Random-SMOTE (2011)[30]			×										70 SMOTE-D (2016)[97]												
28 NDO-sampling (2011)[108]													71 MDO (2016)[1]												
29 DSRBF (2011)[40]	×		×										72 VIS-RST (2016)[10]												
30 SVM-balance (2012)[37]			×										73 GASMOTE (2016)[53]												
31 TRIM-SMOTE (2012)[81]													74 A-SUWO (2016)[79]												
32 SMOTE-RSB (2012)[84]													75 SMOTE-FIRST-2T (2016)[85]												
33 DBSMOTE (2012)[12]	×												76 AND-SMOTE (2016)[105]												
34 ASMOBD (2012)[99]	×												77 SMOTE-PSO (2017)[19]												
35 SN-SMOTE (2012)[42]													78 CURE-SMOTE (2017)[74]												
36 ProWSyn (2013)[6]													79 SOMO (2017)[31]												
37 SL-graph-SMOTE (2013)[13]													80 NRAS (2017)[86]												
38 NRSBoundary-SMOTE (2013)[38]													81 Gaussian-SMOTE (2017)[61]												
39 LVQ-SMOTE (2013)[78]													82 CCR (2017)[57]												
40 SOI-CJ (2013)[90]													83 ANS (2017)[93]												
41 Assembled-SMOTE (2013)[111]													84 AMSCO (2018)[65]												
42 ISMOTE (2013)[64]													85 kmeans-SMOTE (2018)[33]												
43 ROSE (2014)[77]																									

[7] numaralı kaynakte, Veri Artırımı (Data Augmentation) için kullanılan tüm metodlar ve bu metodların veri artırımı yaparken hangi yöntemleri kullanarak veri artırımı işini gerçekleştirebileceği tüm detaylarıyla verilmiş durumda.

**Tablo 2.2** SMOTE metodu ile ML modellerinin doğruluk artışları üzerine yapılmış bir çalışmanın grafiği [7]

classifier	SVM		DT		kNN		MLP	
rank	sampler	AUC	sampler	AUC	sampler	AUC	sampler	AUC
1	DBSMOTE [12]	.9089	LVQ-SMOTE [78]	.8885	AHC [25]	.9153	CCR [57]	.9182
2	CURE-SMOTE [74]	.9068	ProWSyn [6]	.8861	ProWSyn [6]	.9139	SMOTE-IPF [89]	.9174
3	G-SMOTE [91]	.9062	Borderline-SMOTE2 [47]	.8860	Assembled-SMOTE [111]	.9095	Gaussian-SMOTE [61]	.9171
4	CE-SMOTE [23]	.9056	polynom-fit-SMOTE [43]	.8841	SMOTE-IPF [89]	.9092	DBSMOTE [12]	.9169
5	MDO [1]	.9042	SMOBD [17]	.8835	CE-SMOTE [23]	.9091	Assembled-SMOTE [111]	.9168
6	CCR [57]	.9037	Lee [62]	.8834	Gaussian-SMOTE [61]	.9090	polynom-fit-SMOTE [43]	.9166
7	polynom-fit-SMOTE [43]	.9037	Assembled-SMOTE [111]	.8834	Lee [62]	.9088	CE-SMOTE [23]	.9166
8	SMOTE-ENN [8]	.9029	SMOTE-IPF [89]	.8828	Borderline-SMOTE1 [47]	.9088	G-SMOTE [91]	.9166
baseline	SMOTE	.8999	SMOTE	.8809	SMOTE	.9082	SMOTE	.9156
baseline	no sampling	.8568	no sampling	.8124	no sampling	.8488	no sampling	.8248
rank	sampler	G	sampler	G	sampler	G	sampler	G
1	polynom-fit-SMOTE [43]	.8672	DE-oversampling [22]	.8658	polynom-fit-SMOTE [43]	.8820	Selected-SMOTE [56]	.8781
2	DBSMOTE [12]	.8661	SMOTE-ENN [8]	.8611	SMOTE-ENN [8]	.8785	SMOBD [17]	.8781
3	SMOTE-ENN [8]	.8649	Lee [62]	.8598	ProWSyn [6]	.8780	polynom-fit-SMOTE [43]	.8780
4	G-SMOTE [91]	.8629	Borderline-SMOTE2 [47]	.8598	SMOTE-IPF [89]	.8766	SMOTE-IPF [89]	.8777
5	CURE-SMOTE [74]	.8625	SMOTE-IPF [89]	.8590	SVM-balance [37]	.8756	Lee [62]	.8774
6	SMOTE-IPF [89]	.8616	SMOBD [17]	.8584	Lee [62]	.8748	ADOMS [96]	.8769
7	ProWSyn [6]	.8614	LVQ-SMOTE [78]	.8584	Assembled-SMOTE [111]	.8743	ProWSyn [6]	.8768
8	Lee [62]	.8612	SMOTE-ENN [8]	.8582	SMOTE-TomekLinks [8]	.8736	NDO-sampling [108]	.8767
baseline	SMOTE	.8595	SMOTE	.8563	SMOTE	.8733	SMOTE	.8748
baseline	no sampling	.5284	no sampling	.7237	no sampling	.7007	no sampling	.6005
rank	sampler	F1	sampler	F1	sampler	F1	sampler	F1
1	DBSMOTE [12]	.6774	ProWSyn [6]	.7006	polynom-fit-SMOTE [43]	.7276	Supervised-SMOTE [51]	.6784
2	polynom-fit-SMOTE [43]	.6763	Lee [62]	.6994	Supervised-SMOTE [51]	.7254	polynom-fit-SMOTE [43]	.6774
3	Supervised-SMOTE [51]	.6723	SMOBD [17]	.6994	CCR [57]	.7239	DBSMOTE [12]	.6749
4	CURE-SMOTE [74]	.6696	polynom-fit-SMOTE [43]	.6993	ProWSyn [6]	.7208	Assembled-SMOTE [111]	.6745
5	SMOTE-Cosine [56]	.6678	Assembled-SMOTE [111]	.6969	LLE-SMOTE [98]	.7208	SMOTE-IPF [89]	.6744
6	CCR [57]	.6677	Supervised-SMOTE [51]	.6962	Gaussian-SMOTE [61]	.7206	NEATER [4]	.6742
7	LLE-SMOTE [98]	.6677	SMOTE-IPF [89]	.6961	CURE-SMOTE [74]	.7201	ProWSyn [6]	.6739
8	G-SMOTE [91]	.6677	NRSBoundary-SMOTE [38]	.6957	CE-SMOTE [23]	.7200	SMOBD [17]	.6737
baseline	SMOTE	.6636	SMOTE	.6932	SMOTE	.7114	SMOTE	.6706
baseline	no sampling	.4692	no sampling	.6092	no sampling	.6351	no sampling	.4308
rank	sampler	P20	sampler	P20	sampler	P20	sampler	P20
1	polynom-fit-SMOTE [43]	.9918	LVQ-SMOTE [78]	.9908	DE-oversampling [22]	.9939	Gaussian-SMOTE [61]	.9956
2	DBSMOTE [12]	.9914	DE-oversampling [22]	.9904	polynom-fit-SMOTE [43]	.9927	DE-oversampling [22]	.9953
3	LVQ-SMOTE [78]	.9913	polynom-fit-SMOTE [43]	.9902	ProWSyn [6]	.9926	polynom-fit-SMOTE [43]	.9953
4	Gaussian-SMOTE [61]	.9912	E-SMOTE [28]	.9893	E-SMOTE [28]	.9925	DEAGO [9]	.9951
5	LLE-SMOTE [98]	.9911	Gaussian-SMOTE [61]	.9890	SVM-balance [37]	.9923	Selected-SMOTE [56]	.9951
6	SMOTE-ENN [8]	.9911	CCR [57]	.9888	Gaussian-SMOTE [61]	.9920	cluster-SMOTE [24]	.9951
7	MOT2LD [101]	.9910	SSO [88]	.9884	NDO-sampling [108]	.9919	SSO [88]	.9951
8	cluster-SMOTE [24]	.9910	LLE-SMOTE [98]	.9882	SMOTE-ENN [8]	.9919	LVQ-SMOTE [78]	.9950
baseline	SMOTE	.9896	SMOTE	.9862	SMOTE	.9913	SMOTE	.9947
baseline	no sampling	.3242	no sampling	.3033	no sampling	.3285	no sampling	.2864

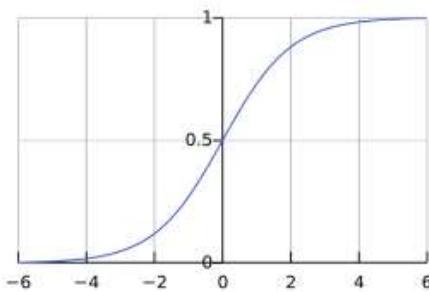
[7] numaralı kaynakta yine aynı şekilde bu metodların yapay zeka mimarileri ile kombinasyonları sonucu ortaya çıkan değerlerin tablo halleri gösterilmekte. Tasarım Projesi’nde Veri Artırımı (Data Augmentation) ve modellerle denemeler sonrasında hem ham hem zenginleştirilmiş verilerde modellerin doğruluğu tabloda bu şekilde benzer bir biçimde gösterilecektir.

### 3. Kullanılan Yöntemlerin Matematiksel Açıklaması

#### 3.1 Logistic Regression

Lojistik regresyon, iki veri faktörü arasındaki ilişkileri bulmak için matematikten yararlanan bir veri analizi tekniğidir. Lojistik regresyon, daha sonra diğerine dayalı bu faktörlerden birinin değerini tahmin etmek için bu ilişkiyi kullanır. Tahminin genellikle evet ya da hayır gibi sınırlı sayıda sonucu vardır. Bu sonuçları makine, 0 ve 1 gibi etiketler üzerinden çıktı verir ve öğrenir. Diğer çoğu ML yöntemlerine göre çok daha az karmaşıktır ve bu basitlik de hızlı bir tahmin yapabilme gücü verir. Çıktı olarak tabloda görüp algılayabilmek iki boyutlu grafikte çok daha kolay ve anlaşılırıdır [8].

$$f(x) = \frac{1}{1+e^{-x}} \quad (3.1)$$



Şekil 3.1 Sigmoid fonksiyonun grafiksel gösterimi

$f(x)$  şeklindeki bu formül, logistic regression eğrisinin sigmoid fonksiyonunu ifade eder. Çıktı olarak da soldaki biçimde bir grafik üretimi gerçekleştirir.

Lojistik regresyon analizi sınıflama ve atama işlemi yapmaya yardımcı olan bir regresyon yöntemidir. Normal dağılım varsayımları, süreklilik varsayımları ön koşulu yoktur. En basit açıklama olarak, bağımlı değişkenin tahmini değerlerini olasılık olarak hesaplayarak, olasılık kurallarına uygun sınıflama yapma imkanı veren bir yöntemdir. Binomial veya multinomial olabilir. Genel kullanım yöntemi binomial içindir. Projede kullanılan yöntem de aynı şekilde binomial olarak incelemesidir.

Lojistik regresyon analizi sonucunda elde edilen modelin uygun olup olmadığı “model ki-kare” testi ile, her bir bağımsız değişkenin modelde varlığının anlamlı olup olmadığı ise Wald istatistiği ile test edilir. Wald ki-kare istatistiği t değerlerinin karesine eşittir

Artıları ve eksiklerinden bahsetmek gereklidir,

- Artıları:
  - Veri kümesi doğrusal olarak ayrılabilir olduğunda iyi performans gösterir.
  - Overfit (aşırı uyum) durumuna daha az eğilimlidir ancak yüksek boyutlu veri kümelerinde aşırı uyum sağlayabilir. Bu senaryolarda aşırı uyumu önlemek için Regularization (Düzenlileştirme) (L1 ve L2) tekniklerini göz önünde bulundurmanız gereklidir.
  - Lojistik Regresyon sadece bir tahmin edicinin (katsayı büyütüğü) ne kadar ilgili olduğunun bir ölçüsünü vermekle kalmaz, aynı zamanda ilişkinin yönünü de (pozitif veya negatif) verir.
  - Uygulanması, yorumlanması çok kolaydır ve eğitilmesi oldukça verimlidir.
- Eksileri:
  - Lojistik Regresyonun temel sınırlaması, bağımlı değişken ile bağımsız değişkenler arasında doğrusallık varsayımlıdır. Gerçek dünyada veriler nadiren doğrusal olarak ayrılabilir. Çok zaman veriler karmakarışık olur.
  - Gözlem sayısı özellik sayısına nazaran belirli eşiklerden çok daha az ise kullanılmamalıdır, aksi takdirde aşırı uyuma yol açabilir.
  - Lojistik Regresyon yalnızca kesikli fonksiyonları tahmin etmek için kullanılabilir. Bu nedenle, Lojistik Regresyonun bağımlı değişkeni

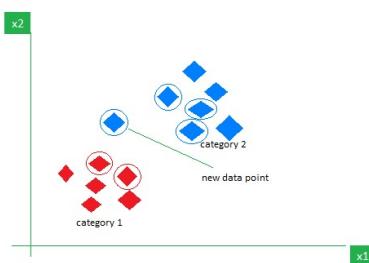
kesikli sayı kümesiyle sınırlıdır. Sürekli verilerin tahmin edilmesini engelleyen bir sorun ise bu kısıtlamanın kendisi sorunudur. Yani kısaca, dinamik verilerin tahmininde ciddi bir problem demektir.

### 3.2 K-Nearest Neigbor

K-Nearest Neigbor (KNN) algoritması, sınıflandırma ve regresyon görevleri için kullanılan popüler bir makine öğrenimi tekniğidir. Benzer veri noktalarının benzer etiketlere veya değerlere sahip olma eğiliminde olduğu fikrine dayanır. Eğitim aşamasında, KNN algoritması tüm eğitim veri kümesini referans olarak saklar. Tahminler yaparken, seçilen mesafe metriğini kullanarak giriş veri noktası ile tüm eğitim örnekleri arasındaki mesafeyi hesaplar. Genelde kullanılan metrik, Euclidian Distribution (Öklid Mesafesi) olmaktadır. Daha sonra algoritma, uzaklıklarına göre giriş veri noktasına en yakın K komşuyu belirler. Sınıflandırma durumunda, algoritma K komşular arasında en yaygın sınıf etiketini giriş veri noktasına için tahmin edilen etiket olarak atar. Burada k değerimizi, 2 sınıf olduğundan ötürü 2 olarak seçeceğiz. Mesafe metriklerini ise GridSearch algoritması üzerinden test ederek en iyisini seçeceğiz. Aşağıda problemimize uygulanabilecek KNN mesafe algoritmaları formülleri gösterilmektedir.

$$\text{euclidian distance}(x, X_i) = \sqrt{\sum_{j=1}^d (x_j - X_{ij})^2} \quad \text{manhattan distance}(x, y) = \sum_{i=1}^n |x_i - y_i|$$

Denklem 3.2.1, 3.2.2: öklidyan mesafe ve Manhattan mesafesi denklem gösterimi



**Şekil 3.2** KNN algoritması grafik örnek [9]

### KNN Algoritması gösterimi

- K-Nearest Neighbor, Denetimli Öğrenme teknigue dayanan en basit Makine Öğrenimi algoritmalarından biridir. Algoritma, mevcut tüm verileri depolar ve benzerliğe göre yeni bir veri noktasını sınıflandırır. Bu, yeni veriler ortaya çıktığında K-NN algoritması kullanılarak kolayca sınıflandırılabilcegi anlamına gelir. Hem Regresyon hem sınıflandırma problemleri için kullanılabilir, fakat
- K-NN veriler üzerinde herhangi bir varsayımda bulunmaz. Aynı zamanda eğitim setinden hemen öğrenmez, bunun yerine veri setini depolar ve sınıflandırma sırasında veri seti üzerinde bir eylem gerçekleştirir. Bu düzellikleri sebebiyle uzun zaman boyunca en beğenilen algoritmalarlardan biri olmuştur.

Artıları ve eksiklerinden bahsetmek gereklidir,

- Artıları:
  - Anlaması ve yorumlaması çok basit bir algoritmadır.
  - Doğrusal olmayan veriler için çok kullanışlıdır çünkü bu algoritmada veriler hakkında herhangi bir varsayımda yoktur.
  - Sınıflandırma ve regresyon için kullanabildiğimiz için çok yönlü bir algoritmadır.
  - Oldukça yüksek doğruluğa sahiptir.

- Eksileri:
  - Diğer denetimli öğrenme algoritmalarına kıyasla yüksek bellek depolaması gereklidir.
  - Çok fazla sınıf belirlemesi yapması gerektiğinde tahmin yavaşlar ve bellek kullanımı çok artar. İkili sınıflar için ise makul bellek kullanımı bulunduğu görülmüştür.
  - Verilerin ölçegine ve ilgisiz özelliklere çok duyarlıdır. Bu sebeple veriler scale edilmeden veya korelasyon matrisi üzerinden incelemeler yapılmış gereksiz özelliklerin çıkartılması işlemi yapılması elzemdir.

### 3.3 Support Vector Machine (SVM)

SVM algoritmasının amacı,  $N$  boyutlu bir uzayda ( $N$  - özellik sayısı) veri noktalarını belirgin bir şekilde sınıflandıran bir hiperdüzlem bulmaktadır. Hiperdüzlem, farklı sınıfların en yakın noktaları arasındaki marjin mümkün olduğu kadar maksimum olmasını sağlamaya çalışır. Hiperdüzlemin boyutu, özelliklerin sayısına bağlıdır. Giriş özelliklerinin sayısı iki ise hiperdüzlem yalnızca bir çizgidir.



**Sekil 3.3** SVM kernel tiplerine göre sınıflandırma grafikleri [10]

Lineer, non-lineer veya çembersel çizgiler ile çekirdek tipi seçimine göre sınıflandırma imkanı sunabilmesi SVM modellerinin en güçlü özelliklerinin başında gelmektedir. Özellikle ayırt edici durumların aslında ortalamaya yakın olması durumu mevcut ise çok başarılı çözümler üretebildiği görülmekte. Kernel tipleri SVM için şu şekildedir:

$$\begin{aligned}
 \text{Linear} &: K(w, b) = w^T x + b \\
 \text{Polynomial} &: K(w, x) = (\gamma w^T x + b)^N \\
 \text{Gaussian RBF} &: K(w, x) = \exp(-\gamma \|x_i - x_j\|^n) \\
 \text{Sigmoid} &: K(x_i, x_j) = \tanh(\alpha x_i^T x_j + b)
 \end{aligned} \tag{3.2}$$

SVM Algoritmasının Kernel tiplerine bağlı olarak türlerine incelemek gerekirse:

- Lineer SVM: Kullanımı çok tercih edilmez, çünkü verinin mükemmel şekilde doğrusal olarak ayrılabilir olduğu durumda kullanılabilir. Bunu sağlayabilmek demek de, verinin genelde bir boyut azaltma işlemeye tabi tutulmasını gerektirir [11].
- Polinomal (Non-lineer) SVM: Tahmin olarak az veride veya doğrusal olmayan veride çok daha başarılı sonuçlar verebilmektedir. Bunu sağlamasında en büyük etmenlerden biri de aşırı uyum(overfitting) ayarlanabilmesi adına polinomun en fazla kaçinci dereceden olabileceğinin de ayarlanabilme rahatlığından dolayıdır. Ezberleme ve bilgisayar gücünün dengesi ile başarı yakalanabilmesi sağlanabilir [12].
- Gaussian RBF Kernel Support Vector Machine: SVM sınıflandırmalarında en çok kullanılan kernel tipidir. Verimizdeki örnekleri sınıflara ayırmak için doğrusal bir karar sınırı kullanabileceğimiz daha yüksek boyutlu bir özellik alanına kaldırılmak için özelliklerimizin

doğrusal olmayan kombinasyonlarını oluşturmaktadır. RBF çekirdek işlevi şu şekilde tanımlanır:  $K(x, y) = \exp(-\gamma * \|xy\|^2)$

Gama değeri çekirdeğin genişliğini ve dolayısıyla modelin karmaşıklığını kontrol eder. Küçük bir gama değeri geniş bir çekirdeğe yol açacak ve bu da düşük varyanslı ve yüksek önyargılı daha basit bir modele yol açacak, büyük bir gama değeri ise dar bir çekirdeğe yol açacak ve yüksek varyanslı ve düşük önyargılı daha karmaşık bir modele yol açacaktır. Diğer önemli hiperparametre, marjı maksimuma çıkarmak ve yanlış sınıflandırma hatasını minimuma indirmek arasındaki dengeyi kontrol eden C'dir. Büyük bir C değeri daha küçük bir marja ve daha az yanlış sınıflandırmaya neden olurken, küçük bir C değeri daha büyük bir marja ve daha fazla yanlış sınıflandırmaya neden olacaktır [13].

- Sigmoid Kernel Support Vector Machine: Sigmoid çekirdek kavramı, çekirdek fonksiyonu denklemi ile bir yapay sinir ağının geliştirilmesidir:  $K(x, xi) = \tanh(\alpha x_i \cdot x_j + \beta)$ .

Sigmoid çekirdeğe sahip SVM sınıflandırması karmaşık bir yapıya sahiptir. Bu çekirdeklerle olan ilgi, sinir ağları ve lojistik regresyon, spesifik özellikler, doğrusallık ve kümülatif dağılım ile sınıflandırma konusundaki başarılarından kaynaklanmaktadır. Pozitif parametrelere sahip olmak zor olduğundan sigmoid çekirdeği genellikle sorunlu veya geçersizdir. Sigmoid fonksiyonu artık araştırmalarda yaygın olarak kullanılmamaktadır çünkü büyük bir dezavantajı vardır, yani sigmoid fonksiyonunun çıktı değeri aralığı sıfır üzerinde merkezlenmemektedir. Bu durum ideal olmayan geri yayılım sürecinin oluşmasına neden olur, böylece YSA'nın ağırlığı pozitif ve negatif değerler arasında eşit olarak dağılmaz ve 0 ve 1 uç değerlerine yaklaşma eğilimi gösterir [14].

Support Vector Machine artılarından ve eksilerinden bahsetmek gerekirse:

- Artıları:
  - Sınıflar arasında net bir ayırım marjı olduğunda nispeten iyi çalışır.
  - Yüksek boyutlu uzaylarda ciddi anlamda etkilidir. Ne kadar boyut o kadar etki. Hatta, boyut sayısının örnek sayısından fazla olması durumunda çok etkili olduğu görülmüştür.
  - Bellek açısından verimlidir.
- Eksileri:
  - SVM algoritması büyük veri kümeleri için uygun değildir.
  - Veri setinde çok fazla gürültü olduğunda, yani hedef sınıflar çakıştığında SVM çok iyi performans göstermez. (Bunu aşmak için boyut artırmak her zaman ek çözümüdür)
  - SVM, sınıflandırma hiperdüzleminin üstüne ve altına veri noktaları koyarak çalıştığından, sınıflandırma için olasılığa dayalı bir açıklama yoktur. Bu da diğer modellere göre en büyük eksiklerinden biridir [15].

### 3.4 Decision Tree Classifier

Karar Ağaçları, esasında hem regresyon hem kategorizasyon problemlerine rahatlıkla uygulanabilen bir algoritmadır. Bu kısımda kategorizasyon için Karar Ağaçlarından bahsedilecek.

Karar aacı, if-else koşuluna sahip akış şemasına benzer bir yapıdır. Üst düğüm kök düğümdür, soruya kök düğümden başlarsınız, ardından ait olduğunuz gruptara göre ağaç dalları arasında bir yaprak düşüme ulaşana kadar ilerlersiniz. Karar ağaçlarını anlayabilmek için Impurity(safsızlık) kavramını anlamamız gereklidir. Impurity ölçümü için ise üç metod kullanılmaktadır:

1. Entropy
2. Gini indeksi / Gini Safsızlığı
3. Standart sapma

- Entropy: Entropi, verileri doğru bir şekilde tanımlamak için gereken bilgi miktarıdır. Yani, eğer veriler homojense, yani tüm elementler benzerse o zaman entropi 0'dır (yani saf), aksi halde eğer elementler eşit olarak bölünmüşse entropi 1'e doğru hareket eder (yani saf değildir).

Yani 1. kare en düşük entropiye, 2. kare daha fazla entropiye ve 3. kare en yüksek entropiye sahiptir. Matematiksel olarak şu şekilde yazılır [16]:

$$\text{Entropy} = - \sum_{i=1}^n p_i * \log(p_i) \quad (3.3)$$

- Gini İndeksi: Düğümdeki safsızlığı ölçer. 0 ile 1 arasında bir değere sahiptir. Yani 0 değerindeki Gini endeksi, numunenin tamamen homojen olduğu ve tüm elemanların benzer olduğu anlamına gelirken, Gini endeksinin 1 değeri, elemanlar arasında maksimum eşitsizlik anlamına gelir. Her sınıfın olasılıklarının karelerinin toplamıdır. Şu şekilde gösterilmiştir [17]:

$$\text{Gini index} = 1 - \sum_{i=1}^n p_i^2 \quad (3.4)$$

- Standart Sapma: Karar ağaçlarında standart sapmanın azaltılmasını isteriz. Veri ne kadar homojen ise standart sapma o kadar 0'a yakınsar. Standart sapmanın hesaplanmasından bahsetmek gerekirse, hedef değişkenin standart sapmasının öngörülerin standart sapmasından çıkarılmasıdır; dolayısıyla daha yüksek standart sapma azaltımı, verilerde daha fazla homojenlik anlamına gelir ve bu, bölgeler için öngörücü değişkenin tanımlanmasına yardımcı olur. İki değişken için standart sapmanın matematiksel hesabı aşağıda belirtilmiştir. Bu formülasyon, her bir değişken için tek tek uygulanır [18, 19].

$$S(T, X) = \sum_{c \in X} P(c) S(c) \quad (3.5)$$

Decision Tree Classifier artırıları ve eksilerinden bahsetmek gerekirse:

- Artıları:

- Karar ağaçları özünde if-else yapısıyla çalışan, çok basit bir algoritmadır. Bu da algoritmayı anlayabilmeyi, yorumlayabilmeyi ve görselleştirebilmeyi çok kolaylaştırır.
- Doğrusal olmayan şekilde ayrılamayan verileri sınıflandırmak için kullanılabılırler.
- Karar ağaçları algoritmasının bir avantajı, eğer doğrusal olmayan verilerle çalışıyorsak, karar ağaçları birden fazla ağırlıklı kombinasyonu aynı anda dikkate almadığından, özelliklerin herhangi bir dönüşümünü gerektirmemeleridir. Daha da basitçe açıklamak gerekirse, ağaç tabanlı algoritmalar ekstradan bir normalleştirme işlemi yapmak gerekmekz. Yapılması bir problem yaratmaz fakat yapılmaması da herhangi bir sonuç odaklı sorun durumu yaratmaz.
- Diğer sınıflandırma algoritmalarına göre çok daha hızlı ve verimlidirler. Bu avantaj özellikle dinamik verilerin işlenmesinde büyük avantaj sağlar.
- Daha az veriyle de efektif sonuçlar yaratabilir. Bu tarz durumlarda en tercih edilen algoritma türlerinden biridir [20].
- Eksileri:
  - Eğer maksimum derinlik sınırlaması yapılmamışsa eğitim verilerine aşırı uyum sağlamaya eğilimlidir.
  - Verilerde küçük değişiklikler olması durumunda eğitimden önemli ölçüde farklı ağaçlar üretilebilir.
  - Veride gürültü olmasından karşı çok duyarlı davranışır.
  - Dengesiz verikümesiyle çalıştığımız durumlarda, verileri eğitim için dengelemek amacıyla ek bir ön işleme adımına ihtiyaç duyulacaktır. Alternatif olarak, üzerinde çalıştığınız uygulama buna izin veriyorsa, sınıf dengesizliklerini hesaba katmak için model içindeki ağırlıkları ayarlayabilirsiniz. Scikit -learn kütüphanesinde bunu class\_weight parametresi aracılığıyla destekler. Alternatif yol olarak, sınıf dengesizliklerini çözmek için stratified bir dağılım yapılmasını sağlayarak veri kümelerini ayırmayı de kolaylıkla yapabilirsiniz.
  - Veride görüldüğü maksimal değerlerin üstünü asla düşünmez. Tüm evreni kendisine verilen veri değerlerinden ibaret kabul eder [21]. (Zaten rassal ormanın çıkış fikri de bu durumu aşmak içindir.)

### 3.5 Random Forest Classifier (RFC)

Random Forest, tek bir sonuca ulaşmak için birden fazla Karar Ağacı (Decision Tree) çıktısını birleştirir. Algoritmanın gücü, karmaşık veri kümelerini ele alma ve aşırı uyumu azaltma yeteneğinde yatkınlıdır ve bu da onu makine öğreniminde çeşitli tahmin görevleri için değerli bir araç haline getirmektedir. Ensemble metodlarından Bagging(Bootstrap Aggregation) metodu üzerinde çalışır ve her ufak ağaç çıktıları birbirine paralel olarak ilerleterek gelişim gerçekleştirebilir. Bagging dediğimiz bu metodda, her ağaç yapısı veri setinin rastgele biçimde küçük bölgelerini görür ve buna göre bir karar ağaç yapısı ve entropi hesabı ile ağaçtaki kolların ayrılma eşiklerinin belirlenmesini sağlar. Bu ağaçların çıktıları ve entropi eşikleri sayesinde en iyi ağaç algoritması oluşturulur. Ayrıca Decision Tree'ler kullanılmasına rağmen her bir ağaç yapısının farklı ağırlık değerlerine sahip olması ve farklı veri setleri görmelerinden ötürü, outlier durumları tespit açısından çok başarılı bir yöntemdir. Çok hızlı bir algoritma olması sebebiyle özellikle yüksek hızda akan veride tespit işlemleri açısından çok başarılı bir algoritmadır.

Hiperparametre ayarlamaları, Random Forest Classifier için çok önemlidir. Modelin underfitting veya overfitting durumundan kurtarmak için parameter ayarlamaları çok etkifl olabilmektedir. Bazı parametrelerinden bahsetmek gerekirse:

1. n\_estimators: Algoritmanın tahminlerin ortalamasını almadan önce oluşturduğu ağaç sayısı  
max\_features: Rastgele ormanın bir düğümü bölmeyi düşündüğü maksimum özellik sayısı.

mini\_sample\_leaf: Dahili bir düğümü bölmek için gereken minimum yaprak sayısını belirler.  
kriter: Her ağaçtaki düğüm nasıl bölünür? (Entropi/Gini impurity/Log Loss)  
max\_leaf\_nodes: Her ağaçtaki maksimum yaprak düğümleri. Biz bir B-Tree şeklinde ağaç kullanmayacağımız için bu parametre ile oynamamak daha doğru. Default 2 olarak kalmalı.  
random\_state: örneğin rastgeleliğini kontrol eder. Model, belirli bir rastgele durum değerine sahipse ve aynı hiperparametreler ve eğitim verileri verilmişse, her zaman aynı sonuçları üretecektir.

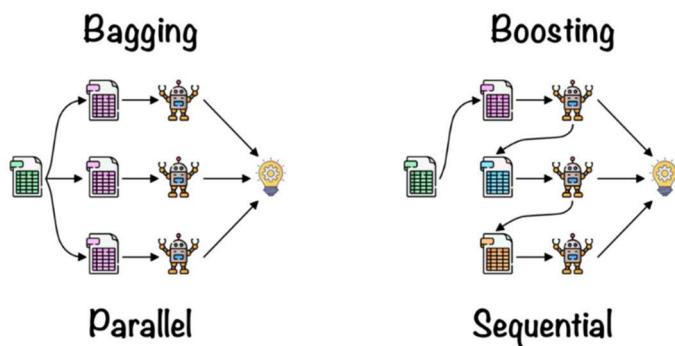
2. bootstrap: Default olarak True ayarlıdır. Eğer False olarak belirtildiğinde her bir ağaç düğümü tüm veriyi görecektir ve bu da Random Forest yapısının ezberli bir Decision Tree sistemine dönmesi demek olur [22].

Random Forest veya diğer ismiyle Rassal Ormanlar, esasında hem regresyon hem kategorizasyon problemlerine rahatlıkla uygulanabilen bir algoritmadır. Bu kısımda kategorizasyon için Rassal Ormanlardan bahsedilecek.

Rassal Ormanları anlayabilmek açısından öncelikle Ensemble metodu nedir bundan bahsedilmesi önemlidir. Ensemble metodları Bagging (Bootstrap Aggregation) ve Boosting olarak ikiye ayrılır.

**Ensemble:** Bu teknikler, birden fazla modeli birleştirerek tahmin yapmak için tek bir model yerine bir modeller koleksiyonu kullanmak ve bu sayede genel performansı artırmayı hedefleyen bir tekniktir.

- **Bagging:** Torbalama özellikle ağaçlar gibi yüksek varyanslı, düşük sapmalı süreçlerde işe yaramazdır. Regresyon için, aynı regresyon ağaçını eğitim verilerinin bootstrap örneklenmiş versiyonlarına birçok kez uydurur ve sonucun ortalamasını alırız. Sınıflandırma için, bir ağaç arama grubu tahmin edilen sınıf için oy kullanır. Yani, bagging ile örnek eğitim verilerinden farklı bir eğitim alt kümesi oluşturulur ve nihai çıktı çoğuluk oylaması ile verilir. Bu yöntemle çalışan en bilinen algoritma Rassal Ormanlardır [23].
- **Boosting:** her modelin bir önceki modelin hatalarını düzeltmeye çalıştığı sıralı bir süreçtir. Sonraki modeller önceki modele bağlıdır. Nihai modelin en yüksek doğruluğa sahip olmasını sağlayacak şekilde sıralı modeller oluşturarak zayıf öğrenenleri güçlü öğrenenlerle birleştirir. Örneğin ADABoost, XGBoost [24].



**Şekil 3.4** Ensemble metodlarının mantığının görsel karşılaştırmaları [25]

Rassal Ormanlar, esasında Karar ağaçlarından türeyen bir yöntem olduğundan ötürü matematiksel açıdan ekleyebileceğimiz ek bir şey yok. Bu noktada, Karar ağaçları ve normal ağaçlar arasındaki farklardan bahsedilecek:

#### Karar Ağaçları:

- Karar ağaçları, maksimum derinliğine kadar büyümeye izin verilirse normalde aşırı uyum sorunuyla karşı karşıya kalır.
- Tek bir karar ağaç hesaplamada daha hızlıdır.

- Özellikleri olan bir veri seti, bir karar ağacı tarafından girdi olarak alındığında, tahmin yapmak için bazı kurallar seti formüle edecektir.
- Verideki nümerik sınırların dışına çıkamaz.

### Rassal Ormanlar:

- Rassal ormanlar torbalama yöntemini kullanır. Orijinal veri kümelerinin bir alt kümelerini oluşturur ve nihai çıktı çoğuluk sıralamasına dayanır ve dolayısıyla aşırı uyum sorunu halledilir.
- Nispeten daha yavaştır.
- Rassal orman, gözlemleri rastgele seçer, bir karar ağacı oluşturur ve ortalama sonuç alınır. Herhangi bir formül seti kullanmaz.

Dolayısıyla Rassal Ormanların, ancak ağaçların çeşitli ve kabul edilebilir olması durumunda Karar Ağaçlarından çok daha başarılı olduğu sonucuna varabiliriz.

Rassal Ormanların Artıları ve eksilerinden bahsetmek gereklidir:

- Artıları:
  - Çıktı çoğuluk oyu veya ortalamaya dayalı olduğundan aşırı uyum sorununu çözer.
  - Veriler boş/eksik değerler içersel bile iyi performans gösterir.
  - Oluşturulan her karar ağacı diğerinden bağımsızdır; dolayısıyla paralelleşme özelliğini gösterir.
  - Çok sayıda ağaçın verdiği ortalama cevaplar alındığından oldukça kararlıdır.
  - Her durumda doğru olmasa da, her karar ağaçını oluştururken tüm nitelikler dikkate alınmadığı için çeşitliliği korur.
- Eksileri:
  - Rassal Orman, ağaçın yolunu takip ederek karar verilebilen Karar Ağaçlarına göre oldukça karmaşıktır.
  - Karmaşıklıktan dolayı eğitim süresi diğer modellere göre daha fazladır. Ne zaman bir tahmin yapmak zorunda kalsa, her karar ağaçının verilen girdi verileri için çıktı üretmesi gereklidir [26].

Modeli hızlı ve verimli bir şekilde oluşturmak isteyen biri varsa, Rassal Orman harika bir seçimdir; çünkü Rassal Ormanın en iyi yönlerinden biri, eksik değerleri işleyebilmesidir. Verimliliği nedeniyle çeşitli endüstrilerde yaygın olarak kullanılan, yüksek performanslı en iyi tekniklerden biridir. İkili, sürekli ve kategorik verileri işleyebilir. Genel olarak Rassal Orman, bazı sınırlamalara sahip hızlı, basit, esnek ve sağlam bir modeldir [27].

### 3.6 Gaussian Naive Bayes (Gaussian NB)

Tahminciler arasında bağımsızlık varsayımasına sahip, Bayes Teoremine dayanan bir sınıflandırma tekniğidir. Basit bir ifadeyle Naive Bayes sınıflandırıcısı, bir sınıfta belirli bir özelliğin varlığının başka herhangi bir özelliğin varlığıyla ilişkili olmadığını varsayar. İstatistikte, naive Bayes sınıflandırıcıları, Bayes teoremini uygulayan basit olasılıksal sınıflandırıcılar olarak kabul edilir. Bu teorem, veriler ve bazı ön bilgiler göz önüne alındığında bir hipotezin olasılığına dayanmaktadır. Naive Bayes sınıflandırıcısı, giriş verilerindeki tüm özelliklerin birbirinden bağımsız olduğunu varsayar; bu da gerçek dünya senaryolarında genellikle doğru değildir. Bununla birlikte, bu basitleştirici varsayıma rağmen, naive Bayes sınıflandırıcısı, birçok gerçek dünya uygulamasında verimliliği ve iyi performansı nedeniyle yaygın olarak kullanılmaktadır.

Ayrıca, naive Bayes sınıflandırıcılarının en basit Bayesian ağ modelleri arasında yer almamasına rağmen çekirdek yoğunluğu tahminiyle birleştirildiğinde yüksek doğruluk seviyelerine ulaşabildiklerini belirtmekte fayda var. Bu teknik, giriş verilerinin olasılık yoğunluk fonksiyonunu tahmin etmek için bir çekirdek fonksiyonunun kullanılmasını içerir ve sınıflandırıcının, veri dağılımının iyi tanımlanmadığı karmaşık senaryolarda performansını artırmamasına olanak tanır. Bir NB

modelinin oluşturulması kolaydır ve özellikle çok büyük veri kümeleri için kullanılabilir. Sadeliğin yanı sıra, Naive Bayes'in son derece karmaşık sınıflandırma yöntemlerinden bile daha iyi performans gösterdiği biliniyor. Naive denmesinin sebebi, 2 değişkenin bağımsız olmayabileceği varsayımdır.

Matematiksel olarak Bayesian kuralı şöyle açıklanır [28]:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \quad (3.6)$$

Naive Bayes varsayımları şu şekildedir:

- Tüm değişkenler bağımsızdır.
- Tüm tahminlerin sonuç üzerinde eşit etkisi vardır.
- Öngörüler ayrık değerler alır.

Meseleyi Gaussian Naive Bayes üzerinden incelersek, Gaussian Naive Bayes, her bir özellikle ilişkili tüm sürekli değişkenlerin Gauss Dağılımı'na göre dağıtılaçığını varsayıduğumuzda kullanılır. Gauss Dağılımı Normal Dağılım olarak da adlandırılır [29].

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (3.7)$$

Yukarıdaki formül, her sınıf için giriş değerlerinin olasılıklarını bir frekans aracılığıyla hesapladı. Tüm dağılımin her bir sınıfı için  $x$ 'lerin ortalamasını ve standart sapmasını hesaplayabiliriz. Bu, her sınıfa ilişkin olasılıkların yanı sıra, sınıfa ilişkin her giriş değişkeninin ortalamasını ve standart sapmasını da saklamamız gerektiği anlamına gelir [30].

Naive Bayes Algoritmasının artılarından ve eksilerinden bahsetmek gerekirse:

- Artıları:
  - Az miktarda eğitim verisi gerektirir. Bu nedenle eğitim daha az zaman alır.
  - Sürekli ve ayrık verileri işler ve alakasız özelliklere duyarlı değildir. Bunu çekirdek tipleri sağlar.
  - Çok basit, hızlı ve uygulaması kolaydır.
  - Hem ikili hem de çok sınıflı sınıflandırma problemlerinde kullanılabilir.
  - Tahmin edici özelliklerin ve veri noktalarının sayısına göre doğrusal olarak ölçeklendiğinden yüksek düzeyde ölçeklenebilir.
- Eksileri:
  - Naive Bayes, tüm niteliklerin karşılıklı olarak bağımsız olduğunu ve bunun gerçek dünya verilerinde bulunmasını neredeyse imkansız olduğunu varsayar.
  - Kategorik bir değişkenin test veri setinde görünen ve eğitim veri setinde gözlemlenmeyen bir değeri varsa, model ona sıfır olasılık atayacak ve tahminde bulunamayacaktır. Bu, "Sıfır Frekans problemi" olarak adlandırdığımız şeydir ve yumoşatma teknikleri kullanılarak çözülebilir [31].

### 3.7 Gradient Boosting (GB)

Bu algoritmanın arkasındaki ana fikir, modelleri sıralı olarak oluşturmak ve bu sonraki modeller, önceki modelin hatalarını azaltmaya çalışmaktadır. Gradient Boosting'in üç ana bileşeni vardır:

1. Loss function(kayıp fonksiyonu): Kayıp fonksiyonunun rolü, modelin verilen verilerle tahmin yapmadada ne kadar iyi olduğunu tahmin etmektir. Bu, mevcut soruna bağlı olarak değişimdir.

2. Weak Learner (Zayıf öğrenici): Weak learner, verilerimizi sınıflandıran ancak bunu kötü yapan, belki de rastgelen tahmin daha iyi olmayan öğrenicilerdir. Bu da hata oranlarının yüksek olmasına sebep olabilir. Bunlar tipik olarak karar ağaçlarıdır (tipik karar ağaçlarından daha az karmaşık oldukları için karar kütükleri olarak da adlandırılırlar).
3. Additive Model (eklemeli model): Bu, ağaçların (zayıf öğreniciler) her seferinde bir adım eklenmesine yönelik yinelemeli ve sıralı bir yaklaşımdır. Her iterasyondan sonra, nihai modelimize daha yakın olmamız gereklidir. Başka bir deyişle, her iterasyon kayıp fonksiyonumuzun değerini azaltmalıdır [32].

Gradient Boosting Algoritmasının artıları ve eksilerinden bahsetmek gerekirse:

- Artıları:
  - Yüksek doğruluk oranları verirler. Bu konuda çok başarılı tasarımlı olan bir algoritmadır.
  - Verilerin ön işlenmesine gerek yoktur; kategorik ve sayısal değerler başarılı performans sergilemektedir.
  - Çoklu hiperparametre ayarlama seçenekleri ve çeşitli kayıp fonksiyonlarını optimize etme yeteneği, fonksiyonun son derece uyarlanabilir olmasını sağlar.
- Eksileri:
  - Gradyan destekli karar ağaç modelleri, hataları azaltmak için gelişmeye devam edecek. Bu, aykırı değerlerin aşırı vurgulanmasına ve aşırı uyumun ortaya çıkmasına neden olabilir.
  - Hesaplama açısından maliyetli; genellikle birkaç ağaç ( $>1000$ ) gerektirir; bu da zaman ve bellek açısından yüksek ihtiyaç gerektirir.
  - Grid Search ile parametre ayarlaması yapmak ve büyük bir model parametre araması yapılması efektif parametre bulma konusunda bir gereklilikdir. Bu da, çok fazla kaynak tüketimi anlamına gelir [33].

## 4. Kullanılan Kavramların Açıklanması:

### 4.1 StandardScaler ve RobustScaler

Standard Scale işlemi, ortalamayı kaldırarak ve birim varyansa ölçeklendirerek özellikleri standart hale getirir. Herhangi bir  $x$  örneğinin standart skoru şu şekilde hesaplanır:

$$z = (x - \mu) / s$$

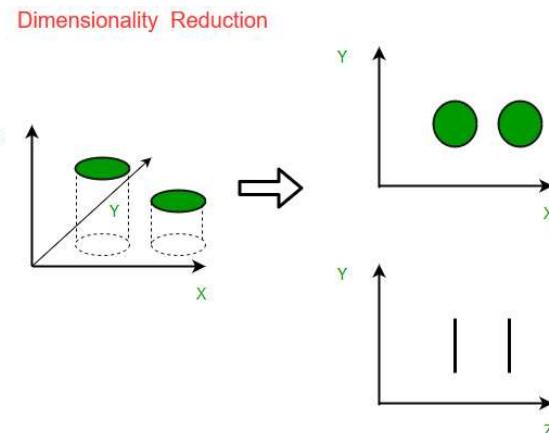
Standardizasyon, ortalama değerin 0, standart sapmanın ise 1 değerini aldığı, dağılımin normale yaklaşığı bir metoddür. Formülü şu şekildedir, elimizdeki değerden ortalama değerini çıkartıyoruz, sonrasında varyans değerine böölüyoruz.

Robust Scale işlemi, Aykırı değerlere sahip verilerde daha iyi sonuçlar verebilir. Yine veri dağılımı ile benzerlik gösterir ancak aykırı değerler dışında kalır. Medyan değeri sonradan kullanılmak üzere elenir ve değerler 1. ve 3. kartil aralığına oturtulur [34].

$$X - Q_1(X) / Q_3(X) - Q_1(X) \quad (4.1)$$

### 4.2 Boyut Azaltma Yöntemleri (t-SNE, PCA, Truncated SVD)

Yüksek boyutlu verilerle uğraşırken, verileri, verinin "özünü" yakalayan daha düşük boyutlu bir alt uzaya yansitarak boyutluluğu azaltmak genellikle yararlı olur. Buna boyutluluğun azaltılması denir. Yüksek boyutluluk yüzlerce, binlerce ve hatta milyonlarca girdi değişkeni anlamına gelebilir. Daha az girdi boyutu genellikle buna bağlı olarak daha az parametre veya makine öğrenimi modelinde serbestlik derecesi olarak adlandırılan daha basit bir yapı anlamına gelir. Çok fazla serbestlik derecesine sahip bir modelin eğitim veri kümese aşırı uyum sağlamaası muhtemeldir ve bu nedenle yeni veriler üzerinde iyi performans göstermeyecek [35].



Şekil 4.1 Boyut Azaltımının mantıksal gösterimi [36]

- t-SNE: t-Distributed Stochastic Neighbor Embedding(t-Dağıtılmış Stokastik Komşu Gömme), zellikle yüksek boyutlu veri kümelerinin görselleştirilmesi için çok uygun olan bir boyut azaltma tekniğidir. t-SNE, özellikle veri noktalarının birbirleriyle olan benzerliklerine dayalı olarak düşük boyutlu bir uzaya haritalamak için tasarlanmıştır. Bu yöntem, özellikle görsel anlamda anlamlı bir şekilde benzer veri noktalarını bir araya getirerek yüksek boyutlu verilerin yapısını korumaya çalışır. Temel olarak, t-SNE benzer veri noktalarını yakın bir uzayda, benzer olmayanları ise daha uzak bir uzayda temsil etmeye çalışır. Bu işlem, veri noktalarının birbirlerine olan uzaklıklarını ölçer, ardından düşük boyutlu bir haritada benzer veri noktalarını bir araya getirirken orijinal uzaklıkları korumaya çalışır [37].
- PCA (Principal Component Analysis- Temel Bileşen Analizi): büyük veri tablolarındaki bilgi içeriğini, daha kolay görselleştirilebilen ve analiz edilebilen daha küçük bir "özet endeksler" seti aracılığıyla özetlemenize olanak tanıyan istatistiksel bir prosedürdür. PCA, çok değişkenli veri analizinin temelini oluşturur. Çok esnek bir araçtır ve amacı, verilerden önemli bilgileri

cıkarmak ve bu bilgileri temel bileşenler adı verilen bir dizi özet indeks olarak ifade etmektir. İstatistiksel olarak PCA, K boyutlu uzayda verilere mümkün olduğu kadar en küçük kareler anlamında yaklaşan çizgiler, düzlemler ve hiperdüzlemler bulur. Bir dizi veri noktasının en küçük kareler yaklaşımı olan bir çizgi veya düzlem, çizgi veya düzlem üzerindeki koordinatların varyansını mümkün olduğu kadar büyük yapar [38].

- Truncated SVD (Truncated Singular Value Decomposition - Kesilmiş Tekil Değer Ayırıştırması): Orijinal bilgilerin çoğunu korurken yüksek boyutlu verilerin boyutlarını azaltmak için makine öğreniminde popüler bir tekniktir. Bu teknik özellikle verilerin çok sayıda özelliğe sahip olduğu, verimli hesaplamaların gerçekleştirilebilmesini veya verilerin görselleştirilmesini zorlaştıran senaryolarda kullanışlıdır. Truncated SVD, bir matrisi tekil değerlerine ve vektörlerine ayırtırıp ardından yalnızca en üstteki  $k$  tekil değerlerini ve vektörlerini seçerek çalışır; burada  $k$ , kullanıcı tanımlı bir parametredir. Bu, orijinal bilgilerin çoğunu yakalayan ve boyutların sayısını önemli ölçüde azaltan azaltılmış bir matrisle sonuçlanır. Truncated SVD'nin avantajlarından biri hem seyrek hem de yoğun matrislerle kullanılabilir, bu da onu çeşitli makine öğrenimi uygulamaları için çok yönlü bir teknik haline getirir. Ek olarak, Truncated SVD, verilerdeki gürültünün veya fazlalığın etkisinin azaltılmasına yardımcı olabilir ve bu da makine öğrenimi modellerinin doğruluğunu artırabilir. Ancak Truncated SVD'nin de bazı sınırlamaları olduğunu unutmamak önemlidir. Örneğin, özellikleri arasında karmaşık ilişkiler bulunan verilerle iyi çalışmaya bilir [39].

#### 4.3 Oversampling ve SMOTE

Oversampling, veri bilimi ve makine öğrenimi alanlarında kullanılan bir tekniktir ve özellikle sınıf dengesizliği problemiyle başa çıkmak için kullanılır. Sınıf dengesizliği, bir veri setinde farklı sınıflar arasında belirgin bir gözle görülür dengesizlik olduğunda ortaya çıkar. Örneğin, bir sınıf diğer sınıflara göre çok daha fazla örnek içerebilir, bu da modelin eğitimini ve performansını olumsuz etkileyebilir. Özellikle azınlık sınıfındaki veri noktalarının sayısını artırmak için kullanılan bir yöntemdir. Azınlık sınıfındaki veri noktaları, genellikle modelin yanlış sınıflandırmamasına veya azınlık sınıfına odaklanmamasına yol açabilecek kadar az olabilir. Bu durumda, Oversampling teknikleri azınlık sınıfındaki örneklerin sayısını artırarak bu dengesizliği gidermeye çalışır.

SMOTE ise, bu Oversampling metodlarından biridir. Sentetik Azınlık Aşırı Örnekleme Tekniği olarak isimlendirilen bu teknik ile, azınlık sınıfı için sentetik örneklerin üretildiği bir aşırı örnekleme gerçekleştiriliyor. Bu algoritma, rastgele aşırı örnekmenin ortaya çıkardığı aşırı uyum sorununun üstesinden gelmeye yardımcı olur. Birlikte bulunan pozitif örnekler arasında interpolasyon yardımıyla yeni örnekler oluşturmak için özellik alanına odaklanır [40].

#### 4.4 Cross Validation ve k-fold Cross Validation

Çarpraz Doğrulama olarak da bahsedilen bu yöntem, makine öğrenimi modellerinin başarısını tahmin etmek için kullanılan istatistiksel bir yöntemdir. Uygulamalı makine öğreniminde, belirli bir tahmine dayalı modelleme problemi için bir modeli karşılaştırmak ve seçmek için yaygın olarak kullanılır, çünkü anlaşılması kolaydır, uygulanması kolaydır ve genellikle diğer yöntemlere göre daha düşük bir önyargıya sahip olan beceri tahminleriyle sonuçlanır. Bu çarpraz doğrulamanın alt metodları kullanılarak modellerin başarısı hakkında doğru bir yorum yapmaya çalışırız.

$k$ -fold Cross Validation yönteminden bahsetmek gereklidir. Bir makine öğrenimi modeliniz ve bazı verileriniz varsa modelinizin uyup uymayacağını söylemek istersiniz. Verilerinizi eğitim ve test setine bölebilirsiniz. Modelinizi eğitim seti ile eğitin ve sonucu test seti ile değerlendirin. Ancak modeli yalnızca bir kez değerlendirdiniz ve iyi sonucunuzun şans eseri olup olmadığından emin değilsiniz. Model tasarımlı konusunda kendinize daha fazla güvenebilmek için modeli birden çok kez değerlendirmek istiyorsunuz. Prosedürün, belirli bir veri örneğinin bölüneceği grup sayısını ifade eden  $k$  adı verilen tek bir parametresi vardır. Bu nedenle prosedüre genellikle  $k$ -katlı çapraz doğrulama adı verilir.  $K$  için belirli bir değer seçildiğinde, model referansında  $k$  yerine kullanılabilir, örneğin  $k=10$ 'un 10 kat çapraz doğrulama haline gelmesi gibi. Genel prosedür aşağıdaki gibidir:

Veri kümesini rastgele karıştırın.

Veri kümesini k gruba ayırin

Her benzersiz grup için:

- Grubu bir bekletme veya test veri seti olarak alın
- Kalan grupları eğitim veri seti olarak alın
- Eğitim setine bir model yerleştirin ve test setinde değerlendirin
- Değerlendirme puanını koruyun ve modeli atın

Model değerlendirme puanları örneğini kullanarak modelin becerisini özetleyin

Bu yaklaşım, gözlem kümesinin rastgele olarak eşit büyüklikte k gruba veya kıvrımlara bölünmesini içerir. İlk katlama bir doğrulama kümesi olarak ele alınır ve yöntem geri kalan k-1 katlamaya uyarlanır [41].

#### 4.5 ROC – AUC Skorlamaları

“Area Under the Curve” (AUC) of the “Receiver Operating Characteristic” (ROC) olarak açılımları olan bu kavramlar, çizdikleri eğriler ve bu eğrilerin skorları ile makine öğrenimi sınıflandırıcımızın ne kadar iyi performans gösterdiğini görselleştirmemize yardımcı olur.

ROC eğrisi veya alıcı işletim karakteristik eğrisi, bir sınıflandırma modelinin ne kadar iyi performans gösterdiğini gösteren bir grafik gibidir. Modelin farklı kesinlik düzeylerinde nasıl karar verdiğiğini görmemize yardımcı olur. Eğrinin iki çizgisi vardır: biri modelin pozitif vakaları (gerçek pozitifler) ne sıklıkta doğru şekilde tanımladığını, diğer ise negatif vakaları yanlışlıkla pozitif (yanlış pozitifler) olarak ne sıklıkta tanımladığını gösterir. Bu grafiğe bakarak modelin ne kadar iyi olduğunu anlayabilir ve bize doğru ve yanlış tahminler arasında doğru dengeyi sağlayacak eşiği seçebiliriz. ROC eğrisi, ikili sınıflandırma problemleri için bir değerlendirme ölçüsüdür. Bu, çeşitli eşik değerlerinde TPR'yi FPR'ye karşı çizen ve esas olarak 'sinyalî' 'gürültüden' ayıran bir olasılık eğrisidir. Başka bir deyişle, bir sınıflandırma modelinin tüm sınıflandırma eşiklerindeki performansını gösterir. Eğri Altındaki Alan (AUC), ikili sınıflandırıcının sınıfları ayırt etme yeteneğinin ölçüsüdür ve ROC eğrisinin bir özeti olarak kullanılır. Bir ROC eğrisinde, daha yüksek bir X ekseni değeri, Gerçek negatiflerden daha fazla sayıda Yanlış pozitifi gösterir. Daha yüksek bir Y ekseni değeri, Yanlış negatiflerden daha yüksek sayıda doğru pozitifi gösterir. Dolayısıyla eşliğin seçimi, Yanlış pozitifler ile Yanlış negatifleri dengeleme yeteneğine bağlıdır [42].

TPR ve FPR kavramlarından da kısaca bahsetmek gerekirse, True Positive Rate ve False Positive Rate kavramları olarak, ikili sınıflandırma problemlerinde bir modelin etkinliğini değerlendirmek için kullanılan iki önemli performans ölçütüdür. TPR pozitif örnekleri ölçuyorsa FPR, model tarafından hatalı bir şekilde pozitif olarak sınıflandırılan negatif örneklerin oranını ölçer. FPR şu şekilde hesaplanır:

$$FPR = FP / (FP + TN) \quad (4.2)$$

FP (Yanlış Pozitif) – Yanlış sınıflandırılmış pozitif örnekler

TN (Gerçek Negatif) – Doğru şekilde sınıflandırılan negatif örnekler [43]

#### 4.6 L1, L2 ve ElasticNet Penalty Skorları

L1 penalty için diğer bir isimlendirme LASSO (Least Absolute Shrinkage and Selection Operator - En Az Mutlak Büzülme ve Seçim Operatörü), L2 penalty için diğer bir isimlendirme ise Ridge'dir. ElasticNet ise bu ikilinin kombinasyonudur.

Lasso'nun temel amacı model basılığı ve doğruluğu arasında bir denge bulmaktır. Bunu, bazı katsayıların tam olarak sıfır olmaya zorlandığı seyrek çözümleri teşvik eden geleneksel doğrusal regresyon modeline bir ceza terimi ekleyerek başarır. Bu özellik, LASSO'yu özellikle özellik seçimi için kullanışlı hale getirir; çünkü ilgisiz veya gereksiz değişkenleri otomatik olarak tanımlayıp atabilir. Daha doğru bir tahmin için regresyon yöntemleri yerine kullanılır. Bu model büzülmeyi kullanır.

Bütünlük, veri değerlerinin ortalama olarak merkezi bir noktaya doğru küçültülmesidir. Lasso prosedürü basit, seyrek modelleri (yani daha az parametreli modeller) teşvik eder. Bu özel regresyon türü, yüksek düzeyde çoklu bağlantı gösteren modeller için veya değişken seçimi/parametre eleme gibi model seçiminin belirli kısımlarını otomatikleştirmek istediğinizde çok uygundur. Bu metoda, cost function'a "katsayıların mutlak değerlerinin toplamı" şeklinde bir düzenleme terimi eklenir. Lasso, gereksiz değişkenlerin katsayılarını sıfır indirir ve böylece doğrudan özellik seçimini de gerçekleştirir. (aslında sıfır indirmesi olayı Ridge ile arasındaki en bariz fark, birazdan deagineceğiz.) [44, 45]

$$\arg \min_{\beta_0, \beta} \left\{ \frac{1}{N} \sum_{i=1}^N \left( y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j \right)^2 \right\} + \lambda \sum_{j=1}^p |\beta_j| \dots \dots \quad (4.3)$$

Ridge regresyonunda maliyet fonksiyonuna "katsayıların karelerinin toplamı" şeklinde ek bir terim eklenir. Ridge regresyonunun esas yaptığı, hata teriminin toplamını ve belirlemeye çalıştığımız katsayıların karelerinin toplamını en aza indirmeye çalışmaktadır. Katsayıların karelerinin toplamına 'düzenleme terimi' adı verilir ve aynı zamanda  $\lambda$  ile gösterilen düzenleme katsayısına da sahiptir [43, 44].

$$\arg \min_{\beta_0, \beta} \left\{ \frac{1}{N} \sum_{i=1}^N \left( y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j \right)^2 \right\} + \lambda \sum_{j=1}^p \beta_j^2 \dots \dots \quad (4.4)$$

ElasticNet Regülarizasyonu, hem L1 hem L2 regülarizasyonlarının birlikte kullandığı ve en optimize çıktıyı üretmek için ortaya çıkartılmış bir yöntemdir. Bir lambda değeri ayarlama ve seçmenin yanı sıra, elastik ağı aynı zamanda alfa parametresini ayarlamamızı da olanak tanır; burada  $\alpha = 0$  için Ridge ve  $\alpha = 1$  için Lasso'ya karşılık gelir. Basitçe söylemek gerekirse, alfa için 0 koyarsanız, ceza fonksiyonu L1 terimine düşer ve alfayı 1 olarak ayarlaysak L2 terimini elde ederiz. Bu nedenle elastik ağı optimize etmek için 0 ile 1 arasında bir alfa değeri seçebiliriz (burada her düzenlemenin ağırlığını ayarlayabiliriz, böylece elastik adını verebiliriz). Bu, etkili bir şekilde bazı katsayıları küçültecek ve seyrek seçim için bazlarını 0'a ayarlayacaktır [45, 46, 47].

$$J(\beta_1, \beta_2, \dots, \beta_m) = \sum_{i=1}^n (y_i - \sum_{j=1}^m x_{ij} \beta_j)^2 + \lambda (\alpha \sum_{j=1}^m |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^m \beta_j^2) \quad (4.5)$$

## 5. UYGULAMA

Uygulama, Banka işlemleri olarak toplanmış bu veride etiketli veri sayısı olarak minör konumda olan veri setinin artırılması ile doğruluk skorlarında yükselme olup olmadığını ölçme amacıyla taşımakta.

1. Kod, öncelikli olarak gerekli kütüphaneler çağrılması ve kurulumu işlemi ile başlamakta.
2. Daha sonrasında ise kullanacağımız veri setinin konumunun ayarlanması ve çağrılması ile devam etmekte.
3. Veri setinin ayarlanması ve herhangi bir problem veya boş değer olmaması amacıyla veri incelemesi ve “null” değer kontrolleri gerçekleştirildi.
4. Elimizdeki veri ile bir sınıflandırma problemine çözüm aradığımızdan ötürü, etiketli verinin dağılımı ve birbirlerine karşı oranları incelendi, fikir yürütüldü. Bu fikir yürütme sonucu şüpheli hareket etiketli verinin çok az olduğu görüldü.
5. Dağılım zamanları ve zamana bağlı dağılımların grafikleri oluşturuldu ve incelendi.
6. “Amount” ve “Time” kolonlarına scaling işlemleri uygulanarak bu noktada bariz belirleyici olmamaları adına bir işlem gerçekleştirildi. “Amount” kolonu için Standard Scaling işlemi, “Time” kolonu için ise Robust Scaling işlemi uygulandı.
7. x ve y setleri oluşturuldu. Y seti sadece etiket bilgisi, x ise kalan bütün bilgileri tutacak biçimde ayarlandı. X ve y setleri de bölünürken Stratified K-fold metodu ile Train ve test setleri oluşturularak dengesiz etiket dağılıminin, train ve test setlerinde herhangi bir problem oluşturmaması amaçlandı.
8. Korelasyon matrisleri oluşturuldu ve her bir özelliğin birbiri ile alakası incelendi. Bazı özellikler deneme amaçlı olarak sınıf etiketi ile korelasyon kutu grafiği de kurularak incelendi.
9. V1'den V28'e kadar olan özelliklerin şüpheli işlemler için aykırı değerlerinin belirlenmesi işlemi gerçekleştirildi ve bu aykırı değerler veri setinden çıkarıldı. Bu sayede daha iyi genellemeye yapabilmek amaçlanmaktadır.
10. t-SNE, PCA ve Truncated SVC boyut azaltma teknikleri kullanılarak veri setini iki boyuta indirmeye işlemleri yapılması amaçlandı. Daha sonrasında bu işlemler sonucu oluşturulan iki boyutlu grafikler bastırılıp incelendi.
11. Verinin %20'si test, kalani train veri seti olarak ayarlandı ve bazı modellerin default değerleri ile çalıştırıldı, sonuçları incelendi. Seçilen bu modeller Lojistik Regresyon, KNN, SVM, Karar Ağaçları, Rassal Ormanlar, Gaussian Naive Bayes, Gradyan Boosting Sınıflandırıcısıdır. Sonrasında bu algoritmalar 5 kümeli çapraz doğrulama metodu ile kontrol edilerek doğruluk skorları çıkarıldı.
12. Bahsedilen bu modeller, default parametrelerinden ayrı olarak efektif olabilecek parametre seçenekleri verilerek en iyi parametreleri seçebilmesi adına GridSearch metodu kullanılarak pek çok model doğruluğu denemesi yapıldı ve en iyi parametreli modeller seçilerek yola bu modellerle devam edildi, bu yeni modeller de çapraz doğrulama işlemine sokuldu ve artışlar gözlemlendi.
13. Elimizdeki bu modellerin Precision-Recall değerlerine göre oluşturulan ROC-AUC Skor tablosu ve her bir modelin skorlamaları oluşturuldu.
14. Lojistik Regresyon, KNN, SVM, Karar Ağaçları, Rassal Ormanlar, Gaussian Naive Bayes, Gradyan Boosting Sınıflandırıcı için SMOTE uygulanmamış ve seçilen en iyi parametrelerle eğitilmiş modellerin veri seti üzerinde Precision, Recall, F1 skor ve doğruluk hesaplaması yapıldı.
15. Aynı işlem bu sefer tekrarlanarak SMOTE uygulanmış veri setleri üzerinde denendi. Aynı parametreli modellerin kullanılmasına özen gösterildi. Modellerin Precision, Recall, F1 skor ve doğruluk hesaplaması yapıldı.
16. Hesaplaması yapıtlan tüm modellerin skorları tablo haline getirilip incelendi. Tahminleme doğruluğu açısından en iyi modelin **Rassal Ormanlar** olduğu görüldü. Değiştirilen parametrelerinde n\_estimators=100, criterion='gini' olarak seçildi. Diğer parametreler normal halindeki tanımlanmış parametreler idi.

## 5.1 Kodun Açıklanması ve Yöntemlerin Uygulanması

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA, TruncatedSVD
import matplotlib.patches as mpatches
import time
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import collections
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from imblearn.pipeline import make_pipeline as imbalanced_make_pipeline
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import NearMiss
from imblearn.metrics import classification_report_imbalanced
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, accuracy_score, classification_report
from collections import Counter
from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

import warnings

warnings.filterwarnings('ignore')
```

Şekil 5.1 Kütüphanelerin yüklenmesi

Bu kod bloğu, Python'da yaygın olarak kullanılan veri analizi ve makine öğrenimi kütüphanelerini içerir ve bir dizi işlevi gerçekleştirir. İşlevler şunlardır:

numpy (import numpy as np): Sayısal hesaplamalar için kullanılan bir kütüphane.

pandas (import pandas as pd): Veri manipülasyonu ve analizi için kullanılan veri çerçeveleri (dataframe) sağlayan bir kütüphane.

matplotlib.pyplot (import matplotlib.pyplot as plt): Grafik çizimleri için kullanılan bir görselleştirme kütüphanesi.

seaborn (import seaborn as sns): Veri görselleştirmesi için kullanılan bir başka görselleştirme kütüphanesi.

sklearn (scikit-learn): Makine öğrenimi için kullanılan bir kütüphane. Sınıflandırma, regresyon, kümeleme, boyut indirgeme ve model değerlendirme gibi birçok algoritmayı içerir.

LogisticRegression: Lojistik regresyon modeli.

SVC (Support Vector Classifier): Destek vektör makineleri (SVM) kullanarak sınıflandırma yapar.

KNeighborsClassifier: K-en yakın komşu algoritmasıyla sınıflandırma yapar.

DecisionTreeClassifier: Karar ağacı algoritmasıyla sınıflandırma yapar.

RandomForestClassifier: Random forest algoritmasıyla sınıflandırma yapar.

GaussianNB: Naive Bayes algoritmasının Gauss dağılımı ile uygulanması.

SMOTE ve NearMiss: Sınıf dengesizliği (class imbalance) problemini çözmek için kullanılan örnek artırma (oversampling) ve örnek azaltma (undersampling) teknikleri.

classification\_report\_imbalanced, precision\_score, recall\_score, f1\_score, roc\_auc\_score, accuracy\_score: Dengesiz veri setleri üzerinde performans değerlendirme metrikleri.

collections, Counter: Veri yapıları üzerinde işlem yapmak için kullanılan Python standart kütüphaneleri.

train\_test\_split: Veri setini eğitim ve test setlerine bölmek için kullanılan fonksiyon.

KFold, StratifiedKFold: Çapraz doğrulama (cross-validation) için kullanılan fonksiyonlar.

GradientBoostingClassifier: Gradyan arttırma (Gradient Boosting) algoritmasıyla sınıflandırma yapar.

confusion\_matrix: Sınıflandırma modellerinin performansını değerlendirmek için kullanılan bir metrik.

Bu kütüphaneler ile gerekli işlemleri yaparken çalıştıracağımız fonksiyonların bulunduğu kütüphane ve bu kütüphanelerin alt fonksiyonları işleme alındı.

In [3]:	df = pd.read_csv('creditcard.csv') df.head()																																																																																																						
Out[3]:	<table border="1"> <thead> <tr> <th></th><th>Time</th><th>V1</th><th>V2</th><th>V3</th><th>V4</th><th>V5</th><th>V6</th><th>V7</th><th>V8</th><th>V9</th><th>...</th><th>V21</th><th>V22</th><th>V23</th><th>V24</th><th>V2</th></tr> </thead> <tbody> <tr><td>0</td><td>0.0</td><td>-1.359807</td><td>-0.072781</td><td>2.536347</td><td>1.378155</td><td>-0.338321</td><td>0.462388</td><td>0.239599</td><td>0.098698</td><td>0.363787</td><td>...</td><td>-0.018307</td><td>0.277838</td><td>-0.110474</td><td>0.066928</td><td>0.12853</td></tr> <tr><td>1</td><td>0.0</td><td>1.191857</td><td>0.266151</td><td>0.166480</td><td>0.448154</td><td>0.060018</td><td>-0.082361</td><td>-0.078803</td><td>0.085102</td><td>-0.255425</td><td>...</td><td>-0.225775</td><td>-0.638672</td><td>0.101288</td><td>-0.339846</td><td>0.16717</td></tr> <tr><td>2</td><td>1.0</td><td>-1.358354</td><td>-1.340163</td><td>1.773209</td><td>0.379780</td><td>-0.503198</td><td>1.800499</td><td>0.791461</td><td>0.247676</td><td>-1.514654</td><td>...</td><td>0.247998</td><td>0.771679</td><td>0.909412</td><td>-0.689281</td><td>-0.32764</td></tr> <tr><td>3</td><td>1.0</td><td>-0.966272</td><td>-0.185226</td><td>1.792993</td><td>-0.863291</td><td>-0.010309</td><td>1.247203</td><td>0.237609</td><td>0.377436</td><td>-1.387024</td><td>...</td><td>-0.108300</td><td>0.005274</td><td>-0.190321</td><td>-1.175575</td><td>0.64737</td></tr> <tr><td>4</td><td>2.0</td><td>-1.158233</td><td>0.877737</td><td>1.548718</td><td>0.403034</td><td>-0.407193</td><td>0.095921</td><td>0.592941</td><td>-0.270533</td><td>0.817739</td><td>...</td><td>-0.009431</td><td>0.798278</td><td>-0.137458</td><td>0.141267</td><td>-0.20601</td></tr> </tbody> </table> <p>5 rows × 31 columns</p>		Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2	0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.12853	1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.16717	2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.32764	3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.64737	4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.20601
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2																																																																																							
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.12853																																																																																							
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.16717																																																																																							
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.32764																																																																																							
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.64737																																																																																							
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.20601																																																																																							

Şekil 5.2 Veri setinin okunması

Bu kod blogu ile Python kodumuza kullanacağımız veri setinin tanıtılması ve ufak bir incelenmesi işlemi gerçekleştirilmişdir. Elimizde BankChurners.csv isimli Banka üzerindeki müşteri işlemlerinin şüpheli veya normal hareket olup olmadığına etiketlendirilmiş bir verisi mevcut. Bu veri üzerinde belirtilen modellerin eğitimi gerçekleştirilecek ve daha sonrasında aynı veri üzerinde SMOTE teknigi ile şüpheli hareketlerin verisinin zenginleştirilmesi gerçekleştirilecek, veri zenginleştirmenin model açısından iyi bir yöntem olup olmadığına karşılaştırılması gerçekleştirilecektir.

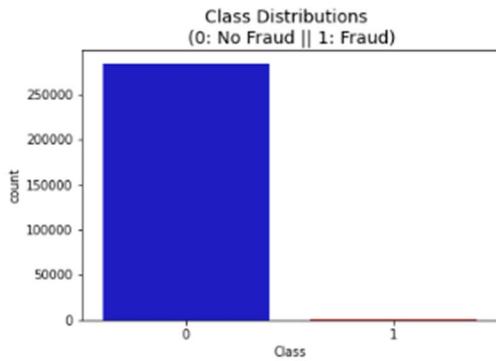
In [4]:	df.describe().T																																																																																																																																																																																																																																																																																																
Out[4]:	<table border="1"> <thead> <tr> <th></th><th>count</th><th>mean</th><th>std</th><th>min</th><th>25%</th><th>50%</th><th>75%</th><th>max</th></tr> </thead> <tbody> <tr><td>Time</td><td>284807.0</td><td>9.481388e+04</td><td>47488.145955</td><td>0.000000</td><td>54201.500000</td><td>84892.000000</td><td>139320.500000</td><td>172792.000000</td></tr> <tr><td>V1</td><td>284807.0</td><td>1.168375e-15</td><td>1.958696</td><td>-56.407510</td><td>-0.920373</td><td>0.018109</td><td>1.315642</td><td>2.454930</td></tr> <tr><td>V2</td><td>284807.0</td><td>3.416908e-16</td><td>1.651309</td><td>-72.715728</td><td>-0.598550</td><td>0.065488</td><td>0.803724</td><td>22.057729</td></tr> <tr><td>V3</td><td>284807.0</td><td>-1.379537e-15</td><td>1.516255</td><td>-48.325589</td><td>-0.890385</td><td>0.179846</td><td>1.027198</td><td>9.382558</td></tr> <tr><td>V4</td><td>284807.0</td><td>2.074095e-15</td><td>1.415889</td><td>-5.683171</td><td>-0.848640</td><td>-0.019847</td><td>0.743341</td><td>16.875344</td></tr> <tr><td>V5</td><td>284807.0</td><td>9.604088e-16</td><td>1.380247</td><td>-113.743307</td><td>-0.691597</td><td>-0.054338</td><td>0.611928</td><td>34.801666</td></tr> <tr><td>V6</td><td>284807.0</td><td>1.487313e-15</td><td>1.332271</td><td>-26.160508</td><td>-0.768298</td><td>-0.274187</td><td>0.398565</td><td>73.301626</td></tr> <tr><td>V7</td><td>284807.0</td><td>-5.556487e-16</td><td>1.237094</td><td>-43.557242</td><td>-0.554078</td><td>0.040103</td><td>0.570438</td><td>120.589494</td></tr> <tr><td>V8</td><td>284807.0</td><td>1.213481e-16</td><td>1.194353</td><td>-73.216718</td><td>-0.208630</td><td>0.022358</td><td>0.327348</td><td>20.007208</td></tr> <tr><td>V9</td><td>284807.0</td><td>-2.408331e-15</td><td>1.098832</td><td>-13.434086</td><td>-0.643098</td><td>-0.051429</td><td>0.597139</td><td>15.594995</td></tr> <tr><td>V10</td><td>284807.0</td><td>2.239053e-15</td><td>1.088850</td><td>-24.568282</td><td>-0.535426</td><td>-0.092917</td><td>0.453923</td><td>23.745136</td></tr> <tr><td>V11</td><td>284807.0</td><td>1.673327e-15</td><td>1.020713</td><td>-4.797473</td><td>-0.762494</td><td>-0.032757</td><td>0.739593</td><td>12.018913</td></tr> <tr><td>V12</td><td>284807.0</td><td>-1.247012e-15</td><td>0.999201</td><td>-18.683715</td><td>-0.405571</td><td>0.140033</td><td>0.618238</td><td>7.848392</td></tr> <tr><td>V13</td><td>284807.0</td><td>8.190001e-16</td><td>0.995274</td><td>-5.791881</td><td>-0.648539</td><td>-0.013588</td><td>0.682505</td><td>7.126883</td></tr> <tr><td>V14</td><td>284807.0</td><td>1.207264e-15</td><td>0.958596</td><td>-19.214325</td><td>-0.425574</td><td>0.050601</td><td>0.493150</td><td>10.528766</td></tr> <tr><td>V15</td><td>284807.0</td><td>4.887456e-15</td><td>0.915316</td><td>-4.498945</td><td>-0.582884</td><td>0.048072</td><td>0.648821</td><td>8.877742</td></tr> <tr><td>V16</td><td>284807.0</td><td>1.437716e-15</td><td>0.876253</td><td>-14.129855</td><td>-0.468037</td><td>0.068413</td><td>0.523298</td><td>17.315112</td></tr> <tr><td>V17</td><td>284807.0</td><td>-3.772171e-16</td><td>0.849337</td><td>-25.162799</td><td>-0.483748</td><td>-0.065678</td><td>0.399875</td><td>9.253526</td></tr> <tr><td>V18</td><td>284807.0</td><td>9.564149e-16</td><td>0.838176</td><td>-9.498746</td><td>-0.498850</td><td>-0.003638</td><td>0.500807</td><td>5.041069</td></tr> <tr><td>V19</td><td>284807.0</td><td>1.039917e-15</td><td>0.814041</td><td>-7.213527</td><td>-0.456299</td><td>0.003735</td><td>0.458949</td><td>5.591971</td></tr> <tr><td>V20</td><td>284807.0</td><td>6.406204e-16</td><td>0.770925</td><td>-54.497720</td><td>-0.211721</td><td>-0.062481</td><td>0.133041</td><td>39.420904</td></tr> <tr><td>V21</td><td>284807.0</td><td>1.654087e-16</td><td>0.734524</td><td>-34.830382</td><td>-0.228395</td><td>-0.029450</td><td>0.186377</td><td>27.202839</td></tr> <tr><td>V22</td><td>284807.0</td><td>-3.568593e-16</td><td>0.725702</td><td>-10.933144</td><td>-0.542350</td><td>0.008782</td><td>0.528564</td><td>10.503090</td></tr> <tr><td>V23</td><td>284807.0</td><td>2.578648e-16</td><td>0.624480</td><td>-44.807735</td><td>-0.161848</td><td>-0.011193</td><td>0.147842</td><td>22.528412</td></tr> <tr><td>V24</td><td>284807.0</td><td>4.473286e-15</td><td>0.605847</td><td>-2.838627</td><td>-0.354588</td><td>0.040978</td><td>0.439527</td><td>4.584549</td></tr> <tr><td>V25</td><td>284807.0</td><td>5.340915e-16</td><td>0.521278</td><td>-10.295397</td><td>-0.317145</td><td>0.016594</td><td>0.350716</td><td>7.519589</td></tr> <tr><td>V26</td><td>284807.0</td><td>1.683437e-15</td><td>0.482227</td><td>-2.804551</td><td>-0.328984</td><td>-0.052139</td><td>0.240952</td><td>3.517346</td></tr> <tr><td>V27</td><td>284807.0</td><td>-3.680091e-16</td><td>0.403832</td><td>-22.565879</td><td>-0.070840</td><td>0.001342</td><td>0.091045</td><td>31.612198</td></tr> <tr><td>V28</td><td>284807.0</td><td>-1.227390e-16</td><td>0.330083</td><td>-15.430084</td><td>-0.052980</td><td>0.011244</td><td>0.078280</td><td>33.847808</td></tr> <tr><td>Amount</td><td>284807.0</td><td>8.834962e+01</td><td>250.120109</td><td>0.000000</td><td>5.600000</td><td>22.000000</td><td>77.166000</td><td>25691.160000</td></tr> <tr><td>Class</td><td>284807.0</td><td>1.727488e-03</td><td>0.041527</td><td>0.000000</td><td>0.000000</td><td>0.000000</td><td>1.000000</td><td></td></tr> </tbody> </table>		count	mean	std	min	25%	50%	75%	max	Time	284807.0	9.481388e+04	47488.145955	0.000000	54201.500000	84892.000000	139320.500000	172792.000000	V1	284807.0	1.168375e-15	1.958696	-56.407510	-0.920373	0.018109	1.315642	2.454930	V2	284807.0	3.416908e-16	1.651309	-72.715728	-0.598550	0.065488	0.803724	22.057729	V3	284807.0	-1.379537e-15	1.516255	-48.325589	-0.890385	0.179846	1.027198	9.382558	V4	284807.0	2.074095e-15	1.415889	-5.683171	-0.848640	-0.019847	0.743341	16.875344	V5	284807.0	9.604088e-16	1.380247	-113.743307	-0.691597	-0.054338	0.611928	34.801666	V6	284807.0	1.487313e-15	1.332271	-26.160508	-0.768298	-0.274187	0.398565	73.301626	V7	284807.0	-5.556487e-16	1.237094	-43.557242	-0.554078	0.040103	0.570438	120.589494	V8	284807.0	1.213481e-16	1.194353	-73.216718	-0.208630	0.022358	0.327348	20.007208	V9	284807.0	-2.408331e-15	1.098832	-13.434086	-0.643098	-0.051429	0.597139	15.594995	V10	284807.0	2.239053e-15	1.088850	-24.568282	-0.535426	-0.092917	0.453923	23.745136	V11	284807.0	1.673327e-15	1.020713	-4.797473	-0.762494	-0.032757	0.739593	12.018913	V12	284807.0	-1.247012e-15	0.999201	-18.683715	-0.405571	0.140033	0.618238	7.848392	V13	284807.0	8.190001e-16	0.995274	-5.791881	-0.648539	-0.013588	0.682505	7.126883	V14	284807.0	1.207264e-15	0.958596	-19.214325	-0.425574	0.050601	0.493150	10.528766	V15	284807.0	4.887456e-15	0.915316	-4.498945	-0.582884	0.048072	0.648821	8.877742	V16	284807.0	1.437716e-15	0.876253	-14.129855	-0.468037	0.068413	0.523298	17.315112	V17	284807.0	-3.772171e-16	0.849337	-25.162799	-0.483748	-0.065678	0.399875	9.253526	V18	284807.0	9.564149e-16	0.838176	-9.498746	-0.498850	-0.003638	0.500807	5.041069	V19	284807.0	1.039917e-15	0.814041	-7.213527	-0.456299	0.003735	0.458949	5.591971	V20	284807.0	6.406204e-16	0.770925	-54.497720	-0.211721	-0.062481	0.133041	39.420904	V21	284807.0	1.654087e-16	0.734524	-34.830382	-0.228395	-0.029450	0.186377	27.202839	V22	284807.0	-3.568593e-16	0.725702	-10.933144	-0.542350	0.008782	0.528564	10.503090	V23	284807.0	2.578648e-16	0.624480	-44.807735	-0.161848	-0.011193	0.147842	22.528412	V24	284807.0	4.473286e-15	0.605847	-2.838627	-0.354588	0.040978	0.439527	4.584549	V25	284807.0	5.340915e-16	0.521278	-10.295397	-0.317145	0.016594	0.350716	7.519589	V26	284807.0	1.683437e-15	0.482227	-2.804551	-0.328984	-0.052139	0.240952	3.517346	V27	284807.0	-3.680091e-16	0.403832	-22.565879	-0.070840	0.001342	0.091045	31.612198	V28	284807.0	-1.227390e-16	0.330083	-15.430084	-0.052980	0.011244	0.078280	33.847808	Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000	77.166000	25691.160000	Class	284807.0	1.727488e-03	0.041527	0.000000	0.000000	0.000000	1.000000	
	count	mean	std	min	25%	50%	75%	max																																																																																																																																																																																																																																																																																									
Time	284807.0	9.481388e+04	47488.145955	0.000000	54201.500000	84892.000000	139320.500000	172792.000000																																																																																																																																																																																																																																																																																									
V1	284807.0	1.168375e-15	1.958696	-56.407510	-0.920373	0.018109	1.315642	2.454930																																																																																																																																																																																																																																																																																									
V2	284807.0	3.416908e-16	1.651309	-72.715728	-0.598550	0.065488	0.803724	22.057729																																																																																																																																																																																																																																																																																									
V3	284807.0	-1.379537e-15	1.516255	-48.325589	-0.890385	0.179846	1.027198	9.382558																																																																																																																																																																																																																																																																																									
V4	284807.0	2.074095e-15	1.415889	-5.683171	-0.848640	-0.019847	0.743341	16.875344																																																																																																																																																																																																																																																																																									
V5	284807.0	9.604088e-16	1.380247	-113.743307	-0.691597	-0.054338	0.611928	34.801666																																																																																																																																																																																																																																																																																									
V6	284807.0	1.487313e-15	1.332271	-26.160508	-0.768298	-0.274187	0.398565	73.301626																																																																																																																																																																																																																																																																																									
V7	284807.0	-5.556487e-16	1.237094	-43.557242	-0.554078	0.040103	0.570438	120.589494																																																																																																																																																																																																																																																																																									
V8	284807.0	1.213481e-16	1.194353	-73.216718	-0.208630	0.022358	0.327348	20.007208																																																																																																																																																																																																																																																																																									
V9	284807.0	-2.408331e-15	1.098832	-13.434086	-0.643098	-0.051429	0.597139	15.594995																																																																																																																																																																																																																																																																																									
V10	284807.0	2.239053e-15	1.088850	-24.568282	-0.535426	-0.092917	0.453923	23.745136																																																																																																																																																																																																																																																																																									
V11	284807.0	1.673327e-15	1.020713	-4.797473	-0.762494	-0.032757	0.739593	12.018913																																																																																																																																																																																																																																																																																									
V12	284807.0	-1.247012e-15	0.999201	-18.683715	-0.405571	0.140033	0.618238	7.848392																																																																																																																																																																																																																																																																																									
V13	284807.0	8.190001e-16	0.995274	-5.791881	-0.648539	-0.013588	0.682505	7.126883																																																																																																																																																																																																																																																																																									
V14	284807.0	1.207264e-15	0.958596	-19.214325	-0.425574	0.050601	0.493150	10.528766																																																																																																																																																																																																																																																																																									
V15	284807.0	4.887456e-15	0.915316	-4.498945	-0.582884	0.048072	0.648821	8.877742																																																																																																																																																																																																																																																																																									
V16	284807.0	1.437716e-15	0.876253	-14.129855	-0.468037	0.068413	0.523298	17.315112																																																																																																																																																																																																																																																																																									
V17	284807.0	-3.772171e-16	0.849337	-25.162799	-0.483748	-0.065678	0.399875	9.253526																																																																																																																																																																																																																																																																																									
V18	284807.0	9.564149e-16	0.838176	-9.498746	-0.498850	-0.003638	0.500807	5.041069																																																																																																																																																																																																																																																																																									
V19	284807.0	1.039917e-15	0.814041	-7.213527	-0.456299	0.003735	0.458949	5.591971																																																																																																																																																																																																																																																																																									
V20	284807.0	6.406204e-16	0.770925	-54.497720	-0.211721	-0.062481	0.133041	39.420904																																																																																																																																																																																																																																																																																									
V21	284807.0	1.654087e-16	0.734524	-34.830382	-0.228395	-0.029450	0.186377	27.202839																																																																																																																																																																																																																																																																																									
V22	284807.0	-3.568593e-16	0.725702	-10.933144	-0.542350	0.008782	0.528564	10.503090																																																																																																																																																																																																																																																																																									
V23	284807.0	2.578648e-16	0.624480	-44.807735	-0.161848	-0.011193	0.147842	22.528412																																																																																																																																																																																																																																																																																									
V24	284807.0	4.473286e-15	0.605847	-2.838627	-0.354588	0.040978	0.439527	4.584549																																																																																																																																																																																																																																																																																									
V25	284807.0	5.340915e-16	0.521278	-10.295397	-0.317145	0.016594	0.350716	7.519589																																																																																																																																																																																																																																																																																									
V26	284807.0	1.683437e-15	0.482227	-2.804551	-0.328984	-0.052139	0.240952	3.517346																																																																																																																																																																																																																																																																																									
V27	284807.0	-3.680091e-16	0.403832	-22.565879	-0.070840	0.001342	0.091045	31.612198																																																																																																																																																																																																																																																																																									
V28	284807.0	-1.227390e-16	0.330083	-15.430084	-0.052980	0.011244	0.078280	33.847808																																																																																																																																																																																																																																																																																									
Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000	77.166000	25691.160000																																																																																																																																																																																																																																																																																									
Class	284807.0	1.727488e-03	0.041527	0.000000	0.000000	0.000000	1.000000																																																																																																																																																																																																																																																																																										

Şekil 5.3 Veri setinin incelenmesi

Verideki özelliklerin genel durumları ve herhangi bir null, NaN veya nümerik olmayan bir değer var mı incelemesi gerçekleştirildi. Herhangi bir ek durum olmadığı gözlemlendi ve buna göre bir yol haritası çizildi.

```
In [9]: colors = ["#0101DF", "#DF0101"]

sns.countplot(x='Class', data=df, palette=colors)
plt.title('Class Distributions \n (0: No Fraud || 1: Fraud)', fontsize=14)
plt.show()
```



**Şekil 5.4** Veri setindeki sütunların etiket oranlarının grafikleştirilmesi

Gördüğümüz üzere veride fraud verisi çok çok az. Gelecek işlemlerde bunun için veri zenginleştirmesi yapmak doğru bir yol olacaktır.

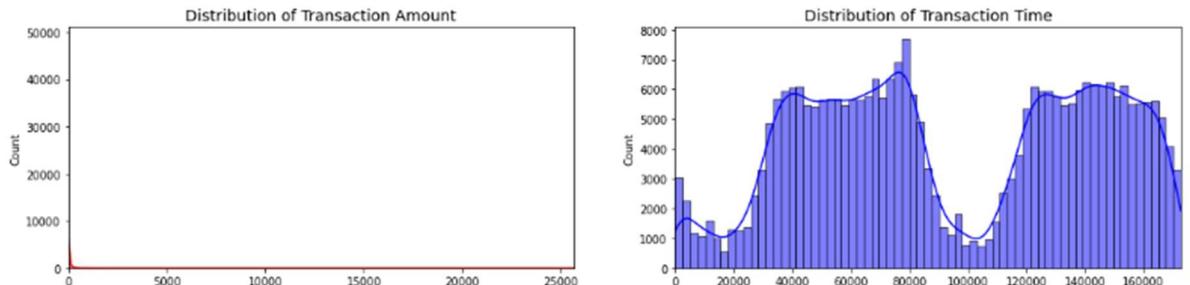
```
In [10]: fig, ax = plt.subplots(1, 2, figsize=(18,4))

amount_val = df['Amount'].values
time_val = df['Time'].values

sns.histplot(amount_val, ax=ax[0], color='r', kde=True)
ax[0].set_title('Distribution of Transaction Amount', fontsize=14)
ax[0].set_xlim([min(amount_val), max(amount_val)])

sns.histplot(time_val, ax=ax[1], color='b', kde=True)
ax[1].set_title('Distribution of Transaction Time', fontsize=14)
ax[1].set_xlim([min(time_val), max(time_val)])

plt.show()
```



**Şekil 5.5** Amount ve time kolonlarının dağılımının dağılımının grafikleştirilmesi

Dağılımları görerek bu özelliklerin ne kadar çarpık olduğu hakkında bir fikir sahibi olabiliriz, ayrıca diğer özelliklerin dağılımlarını da görebiliriz. Dağılımların daha az çarpık olmasına yardımcı olabilecek teknikler vardır ve bunlar gelecekte bu kodda uygulanacaktır.

```
In [11]: # Since most of our data has already been scaled we should scale the columns that are left to scale (Amount and Time)

# RobustScaler is less prone to outliers.
#minmax_scaler = MinMaxScaler()
std_scaler = StandardScaler()
rob_scaler = RobustScaler()

df['scaled_amount'] = rob_scaler.fit_transform(df['Amount'].values.reshape(-1,1))
df['scaled_time'] = rob_scaler.fit_transform(df['Time'].values.reshape(-1,1))

df.drop(['Time','Amount'], axis=1, inplace=True)
```

```
In [12]: scaled_amount = df['scaled_amount']
scaled_time = df['scaled_time']

df.drop(['scaled_amount', 'scaled_time'], axis=1, inplace=True)
df.insert(0, 'scaled_amount', scaled_amount)
df.insert(1, 'scaled_time', scaled_time)

# Amount and Time are Scaled!

df.head()
```

```
Out[12]:
   scaled_amount  scaled_time      V1      V2      V3      V4      V5      V6      V7      V8 ...      V20      V21      V22      V
0       1.783274 -0.994983 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239509  0.098898 ...  0.251412 -0.018307  0.277838 -0.1104
1      -0.269825 -0.994983  1.191857  0.266151  0.186480  0.448154  0.060018 -0.082361 -0.078803  0.085102 ... -0.069083 -0.225775 -0.638672  0.1012
2       4.983721 -0.994972 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791481  0.247876 ...  0.524980  0.247998  0.771679  0.9094
3      1.418291 -0.994972 -0.986272 -0.185226  1.792993 -0.083291 -0.010309  1.247203  0.237609  0.377438 ... -0.208038 -0.108200  0.005274 -0.1903
4       0.670579 -0.994980 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941 -0.270533 ...  0.408542 -0.009431  0.798278 -0.1374
```

5 rows × 31 columns

**Şekil 5.6** Amount ve time kolonlarının scale edilmesi

Bu kod blogu, veri setindeki 'Amount' (Miktar) ve 'Time' (Zaman) sütunlarının ölçeklendirilmesini sağlar. 'Amount' ve 'Time' sütunları, daha önceden ölçeklendirilmemişse, bu sütunları ölçeklendirmek için RobustScaler kullanılarak veri setine eklenir.

İşlemlerin adımları şu şekildedir:

Ölçekleyiciler (scaler'lar) tanımlanır, 'Amount' ve 'Time' sütunları ölçeklenir:

RobustScaler kullanılarak 'Amount' ve 'Time' sütunları ölçeklenir. Önce 'Amount' sütunu için ölçeklendirme yapılır ve sonuç 'scaled\_amount' adında yeni bir sütuna eklenir. Ardından 'Time' sütunu için ölçeklendirme yapılır ve sonuç 'scaled\_time' adında yeni bir sütuna eklenir. 'Amount' ve 'Time' sütunları veri setinden çıkarılır. Ölçeklenmiş 'Amount' ve 'Time' sütunları, veri setine eklenir: Önce 'scaled\_amount' sütunu veri setinden çıkarılır ve 'Amount' sütununun yerine eklenir. Benzer şekilde 'scaled\_time' sütunu 'Time' sütununun yerine eklenir.

Son olarak, artık ölçeklenmiş 'Amount' ve 'Time' sütunlarına sahip olan veri seti gösterilir.

```

In [13]: # veriyi train ve test kisimlari olarak ayiralim.
print('No Frauds', round(df['Class'].value_counts()[0]/len(df) * 100,2), '% of the dataset')
print('Frauds', round(df['Class'].value_counts()[1]/len(df) * 100,2), '% of the dataset')

X = df.drop('Class', axis=1)
y = df['Class']

sss = StratifiedKFold(n_splits=5, random_state=None, shuffle=False)

for train_index, test_index in sss.split(X, y):
    print("Train:", train_index, "Test:", test_index)
    original_Xtrain, original_Xtest = X.iloc[train_index], X.iloc[test_index]
    original_ytrain, original_ytest = y.iloc[train_index], y.iloc[test_index]

# We already have X_train and y_train data thats why I am using original to distinguish and to not overwrite these variables.
# original_Xtrain, original_Xtest, original_ytrain, original_ytest = train_test_split(X, y, test_size=0.2, random_state=42)

# Check the Distribution of the Labels

# Turn into an array
original_Xtrain = original_Xtrain.values
original_Xtest = original_Xtest.values
original_ytrain = original_ytrain.values
original_ytest = original_ytest.values

# See if both the train and test Label distribution are similarly distributed
train_unique_label, train_counts_label = np.unique(original_ytrain, return_counts=True)
test_unique_label, test_counts_label = np.unique(original_ytest, return_counts=True)
print('-' * 100)

print('Label Distributions: \n')
print(train_counts_label/ len(original_ytrain))
print(test_counts_label/ len(original_ytest))

No Frauds 99.83 % of the dataset
Frauds 0.17 % of the dataset
Train: [ 30473  30496  31002 ... 284804 284805 284806] Test: [    0     1     2 ... 57017 57018 57019]
Train: [    0     1     2 ... 284804 284805 284806] Test: [ 30473  30496  31002 ... 113964 113965 113966]
Train: [    0     1     2 ... 284804 284805 284806] Test: [ 81609  82400  83053 ... 170946 170947 170948]
Train: [    0     1     2 ... 284804 284805 284806] Test: [150654 150660 150661 ... 227866 227867 227868]
Train: [    0     1     2 ... 227866 227867 227868] Test: [212516 212644 213092 ... 284804 284805 284806]
-----
Label Distributions:

[0.99827076 0.00172924]
[0.99827952 0.00172048]

```

**Şekil 5.7** Veri setinin eğitim ve test setlerine bölünmesi

Bu kod blogu, veri setini eğitim ve test setlerine ayırmak için Stratified K-Fold çapraz doğrulama tekniğini kullanır. Ayrıca, veri setinin sınıf dağılımını kontrol eder ve eğitim ve test setlerinin sınıf dağılımlarını karşılaştırır.

İşlemlerin adımları şu şekildedir:

Veri setinin sınıf dağılımı hesaplanır ve ekrana yazdırılır:

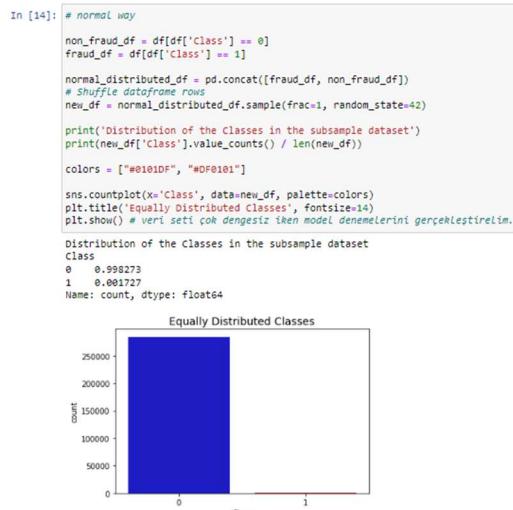
'Class' sütunu üzerindeki hileli ('Frauds') ve hileli olmayan ('No Frauds') işlemlerin yüzdesi hesaplanır ve yazdırılır. Veri seti, bağımlı değişken (etiket) ve bağımsız değişkenler (özellikler) olarak ayrılır:

X, bağımsız değişkenleri (etiket sütunu hariç) içerir. y, bağımlı değişkeni (sadece etiket sütunu) içerir. Stratified K-Fold çapraz doğrulama yapılır:

StratifiedKFold nesnesi, veri setini parçalara bölmek için kullanılır. n\_splits=5 ile 5 parçağa bölmeye yapılır. sss.split(X, y) ile veri seti parçalara ayrılır ve eğitim (original\_Xtrain, original\_ytrain) ve test(original\_Xtest, original\_ytest) setleri oluşturulur. Eğitim ve test setlerinin etiket dağılımları kontrol edilir:

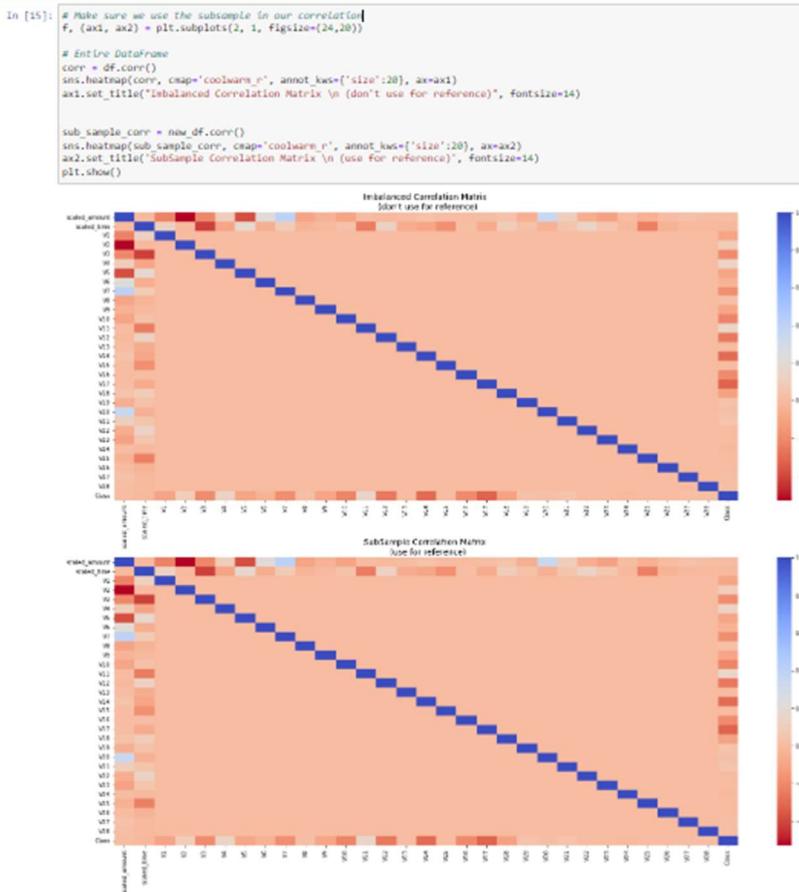
np.unique() ve return\_counts=True kullanılarak her iki setteki etiketlerin benzersiz değerleri ve sayıları alınır. Eğitim ve test setlerinin sınıf dağılımı yüzdesi hesaplanır ve yazdırılır. Bu, her iki setin de sınıflarının ne kadar dengeli olduğunu kontrol etmek için yapılır.

Bu işlem, veri setini eğitim ve test setlerine bölmek için Stratified K-Fold yöntemini kullanırken, bu setlerin sınıf dengesini gözlemlemek ve eğitim/test setlerinin benzer dağılımlara sahip olup olmadığını kontrol etmek amacıyla yapılmıştır. Bu adımlar, modelin genelleme yeteneğini doğrulamak ve dengesiz veri kümeleriyle başa çıkmak için önemlidir.



**Şekil 5.8** Veri setindeki sütunların etiket oranlarının grafikleştirilmesi

Bu kod bloğu, dengesiz sınıf dağılımına sahip bir veri setini dengelemek amacıyla hileli ve hileli olmayan işlemlerin (Class 1 ve Class 0) eşit dağılımlı bir alt örneklemini oluşturur ve bu yeni dengelemiş veri setinin sınıf dağılımını göstermektedir.



**Şekil 5.9** Veri setindeki özelliklerin birbiriyle korelasyonunun incelenmesi

Bu kod bloğu, korelasyon matrislerini görselleştirmek için ısı haritası kullanır. Bu görselleştirmeler, dengesiz ve dengelemiş (alt örnekleme yapılmış) veri setleri için korelasyonları karşılaştırmak için kullanılır.

```
In [16]: f, axes = plt.subplots(ncols=4, figsize=(20,4))

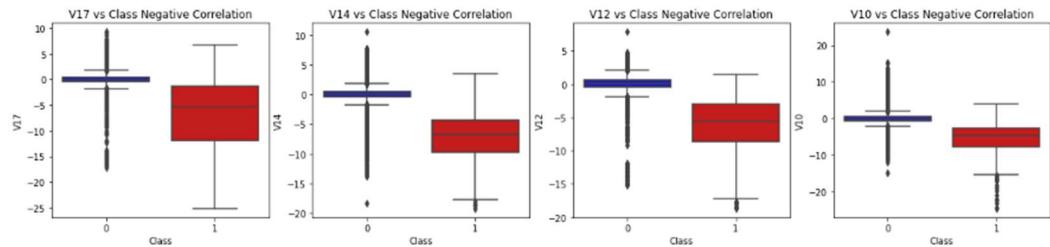
# Negative Correlations with our Class (The Lower our feature value the more likely it will be a fraud transaction)
sns.boxplot(x="Class", y="V17", data=new_df, palette=colors, ax=axes[0])
axes[0].set_title('V17 vs Class Negative Correlation')

sns.boxplot(x="Class", y="V14", data=new_df, palette=colors, ax=axes[1])
axes[1].set_title('V14 vs Class Negative Correlation')

sns.boxplot(x="Class", y="V12", data=new_df, palette=colors, ax=axes[2])
axes[2].set_title('V12 vs Class Negative Correlation')

sns.boxplot(x="Class", y="V10", data=new_df, palette=colors, ax=axes[3])
axes[3].set_title('V10 vs Class Negative Correlation')

plt.show()
```

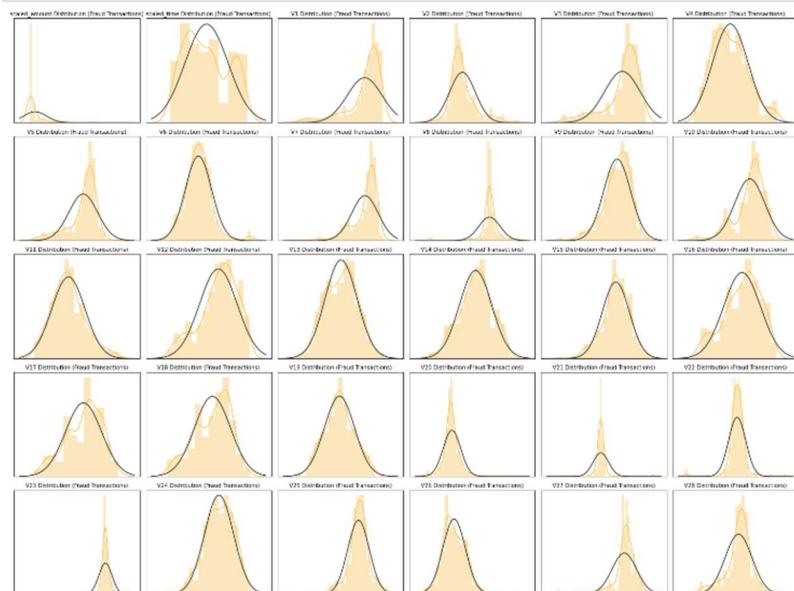


**Şekil 5.10** Veri setindeki bazı özelliklerin sınıf etiketi ile ilişkisinin görselleştirilmesi

Bu kod blogu, sınıf etiketi (Class) ile belirli özelliklerin ilişkisini kutu grafiği (boxplot) ile görselleştirir. Bu grafikler, sınıf etiketi (hileli veya hileli olmayan) ile özellikler arasındaki ilişkiyi analiz etmek için kullanılır.

```
In [17]: f, axes = plt.subplots(5, 6, figsize=(20, 15))
axes = axes.ravel()

for i, column in enumerate(new_df.columns[0:30]): # 'V1' ile 'V28' arasındaki özellikler
    v_fraud_dist = new_df[column].loc[new_df['Class'] == 1].values
    sns.distplot(v_fraud_dist, ax=axes[i], fit=norm, color="#FFC35B")
    axes[i].set_title(f'({column}) Distribution (Fraud Transactions)', fontsize=10)
    axes[i].set_xticks([]) # X eksenini etiketlerini kapatır
    axes[i].set_xlabel('') # X eksenini etiketini kapatır
    axes[i].set_yticks([]) # Y eksenini etiketlerini kapatır
    plt.tight_layout()
plt.show()
```



**Şekil 5.11** Tüm özelliklerin class=1 etiketli veriye göre dağılımının normal dağılımı

Bu kod blogu, alt örnekleme yapılmış (dengelenmiş) veri setindeki hileli işlemlerin (Class=1) belirli özelliklerin (V1'den V28'e kadar) dağılımlarını gösteren grafikler oluşturur. Bu grafikler, hileli işlemlerin belirli özelliklerde nasıl dağıldığını görselleştirmek için kullanılır.

```
In [18]: # Kullanılacak özelliklerin Listesi
features = new_df.columns[0:30] # 'V1' ile 'V28' arası özellikler

for feature in features:
    fraud_values = new_df[feature].loc[new_df['Class'] == 1].values
    q25, q75 = np.percentile(fraud_values, 25), np.percentile(fraud_values, 75)
    iqr = q75 - q25

    cut_off = iqr * 1.5
    lower_bound, upper_bound = q25 - cut_off, q75 + cut_off

    print(f'{feature} Lower Bound: {lower_bound}')
    print(f'{feature} Upper Bound: {upper_bound}')

    outliers = [x for x in fraud_values if x < lower_bound or x > upper_bound]
    print(f'{feature} Outliers: {outliers}')
    print(f'Feature {feature} Outliers for Fraud Cases: {len(outliers)}')

new_df = new_df.drop(new_df[(new_df[feature] > upper_bound) | (new_df[feature] < lower_bound)].index)
print(f'Number of Instances after outliers removal for {feature}: {len(new_df)}')
print('----' * 22)

scaled_amount Lower Bound: -2.4919304129113393
scaled_amount Upper Bound: 3.370711940194229
scaled_amount Outliers: [5.650806958708866, 24.97980856563963, 9.713547124991267, 3.4653811220568715, 8.747292670998393, 6.107315028295955, 9.798225389506044, 6.4098372109271295, 16.724516174107457, 3.7473625375532733, 29.38802976315238, 11.282191015161043, 8.567036959407533, 10.547474324041083, 6.850415706001537, 4.228184168238664, 4.643750436665968, 3.507440788094739, 3.5515964507790123, 4.4450499545867395, 18.34695731153497, 7.97736323621882, 4.108991825613079, 7.084468664850136, 4.152868022877831, 7.678054915112137, 19.10934115838748, 6.535457276601691, 4.137637113113953, 20.72144204569273, 4.042199399147628, 12.622231537762872, 13.613777684622372, 4.051002585062531, 5.142178439181164, 18.615943547823658, 4.57039055404178, 4.758611052888982, 3.82868720743388016, 15.021169566128695, 4.548487668553064, 8.555858310626782, 4.16418640396842, 5.998323202682876, 19.285404876685533, 9.863899951093412, 8.086774261161183, 4.216726053238315, 6.850415706001537, 10.90644868301544, 9.90512118472718, 11.218193250890799, 4.16404667085866, 3.8133165653601617, 4.030182351706238, 9.798225389506044, 7.364773283029414, 9.75868019443863, 4.6943338223992175, 4.722979109900091, 9.764549710053798, 6.957311534968211, 4.108991825613079, 10.401173758121988, 4.781527282889681, 6.04792845647803, 5.899112694753822, 9.028470900579893, 4.513379445259554]
Feature scaled_amount Outliers for Fraud Cases: 69
Number of Instances after outliers removal for scaled_amount: 263373
-----
scaled_time Lower Bound: -1.9777840435155487
scaled_time Upper Bound: 1.9323652768477073
scaled_time Outliers: []
```

**Şekil 5.12** Tüm özelliklerin aykırı değerlerinin bulunması

Bu kod, V1'den V28'e kadar olan her bir özelliğin dolandırıcılık işlemleri için outlier'larını belirler ve bu aykırı değerleri veri setinden kaldırır. Aykırı değerlerin tespitinde (Çeyrekler Arası Aralık) metodu kullanılır. Bu işlem, her bir özellik için hileli işlemlerin aykırı değerlerini belirler ve bu aykırı değerleri veri setinden çıkararak veri setini temizler. Aykırı değerlerin çıkarılması, modelin daha iyi genelleme yapmasına yardımcı olabilir ve veri setindeki anormallikleri azaltabilir.

```
In [21]: # Dimensionality Reduction and Clustering:
X = new_df.drop('Class', axis=1)
y = new_df['Class']

# T-SNE Implementation
t0 = time.time()
X_reduced_tsne = TSNE(n_components=2, random_state=42).fit_transform(X.values)
t1 = time.time()
print("T-SNE took {:.2} s".format(t1 - t0))

# PCA Implementation
t0 = time.time()
X_reduced_pca = PCA(n_components=2, random_state=42).fit_transform(X.values)
t1 = time.time()
print("PCA took {:.2} s".format(t1 - t0))

# TruncatedSVD
t0 = time.time()
X_reduced_svd = TruncatedSVD(n_components=2, algorithm="randomized", random_state=42).fit_transform(X.values)
t1 = time.time()
print("Truncated SVD took {:.2} s".format(t1 - t0))

File "C:\Users\Asus\anaconda3\envs\notebook\lib\site-packages\joblib\externals\loky\backend\context.py", line 282, in _count_physical_cores
    raise ValueError(f"found {cpu_count_physical} physical cores < 1")
T-SNE took 3.9e+02 s
PCA took 0.39 s
Truncated SVD took 0.31 s
```

**Şekil 5.13** Veri setinde boyut azaltma işlemi uygulanması

Bu kod parçacığı, boyut azaltma tekniklerini (T-SNE, PCA, Truncated SVD) kullanarak veri setini iki boyuta indirger. Bu işlem, veri görselleştirmesi ve kümelenme analizi gibi amaçlar için boyut azaltma işlemlerini gerçekleştirir.

```

In [22]: f, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(24,6))
# Labels = ['No Fraud', 'Fraud']
f.suptitle('Clusters using Dimensionality Reduction', fontsize=14)

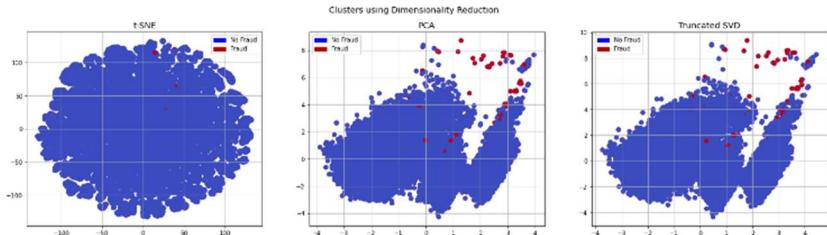
blue_patch = mpatches.Patch(color='#00A0FF', label='No Fraud')
red_patch = mpatches.Patch(color='#AF0000', label='Fraud')

# t-SNE scatter plot
ax1.scatter(X_reduced_tsne[:,0], X_reduced_tsne[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)
ax1.scatter(X_reduced_tsne[:,0], X_reduced_tsne[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
ax1.set_title('t-SNE', fontsize=14)
ax1.grid(True)
ax1.legend(handles=[blue_patch, red_patch])

# PCA scatter plot
ax2.scatter(X_reduced_pca[:,0], X_reduced_pca[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)
ax2.scatter(X_reduced_pca[:,0], X_reduced_pca[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
ax2.set_title('PCA', fontsize=14)
ax2.grid(True)
ax2.legend(handles=[blue_patch, red_patch])

# TruncatedSVD scatter plot
ax3.scatter(X_reduced_svd[:,0], X_reduced_svd[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)
ax3.scatter(X_reduced_svd[:,0], X_reduced_svd[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
ax3.set_title('Truncated SVD', fontsize=14)
ax3.grid(True)
ax3.legend(handles=[blue_patch, red_patch])
plt.show()

```



**Şekil 5.14** Boyut azaltma uygulanmış verinin görselleştirilmesi

Bu grafikler, boyut azaltma yöntemlerinin sonuçlarını görsel olarak karşılaştırmak ve veri noktalarının sınıflandırma sonuçlarını anlamak için kullanılır. Veri noktalarının farklı boyut azaltma yöntemleri tarafından nasıl gruplandırıldığını gözlemlemek ve veri setinin yapısal farklılıklarını anlamak için bu tür grafikler önemli bir araçtır.

```

In [23]: X = new_df.drop('Class', axis=1)
y = new_df['Class']

In [24]: # Our data is already scaled we should split our training and test sets
# This is explicitly used for normal way data.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [25]: # Turn the values into an array for feeding the classification algorithms.
X_train = X_train.values
X_test = X_test.values
y_train = y_train.values
y_test = y_test.values

In [26]: # Let's implement simple classifiers
classifiers = {
    "LogisticRegression": LogisticRegression(),
    "KNearest": KNeighborsClassifier(),
    "Support Vector Classifier": SVC(),
    "DecisionTreeClassifier": DecisionTreeClassifier(),
    "RandomForestClassifier": RandomForestClassifier(),
    "GaussianNB": GaussianNB(),
    "GBClassifier": GradientBoostingClassifier()
}

In [27]: # Now our scores are getting even high scores even when applying cross validation.
for key, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    training_score = cross_val_score(classifier, X_train, y_train, cv=5)
    print("Classifiers: ", classifier.__class__.__name__, "Has a training score of", round(training_score.mean(), 6) * 100, "% accuracy score")

```

**Şekil 5.15** YZ modellerinin default değerlerinin modele uygulanması ve doğruluk skorları

Bu kod parçası, çeşitli sınıflandırma algoritmalarını (LogisticRegression, KNeighborsClassifier, SVC, DecisionTreeClassifier, RandomForestClassifier, GaussianNB, GradientBoostingClassifier)

eğiterek ve bu algoritmaların çapraz doğrulama skorlarını değerlendirerek modellerin eğitim başarısını ölçer.

```
In [28]: t0 = time.time()
# Use GridSearchCV to find the best parameters.

# Logistic Regression
log_reg_params = {'penalty': ['l1', 'l2','elasticnet'],'tol':[3e-5,6e-5,1e-4,3e-4,6e-4], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
grid_log_reg = GridSearchCV(LogisticRegression(), log_reg_params)
grid_log_reg.fit(X_train, y_train)
# we automatically get the logistic regression with the best parameters.
log_reg = grid_log_reg.best_estimator_
t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
<
Fitting oversample data took :41.151628255844116 sec

In [29]: t0 = time.time()

knears_params = {"n_neighbors": list(range(2,5,1)), 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']}
grid_knears = GridSearchCV(KNeighborsClassifier(), knears_params)
grid_knears.fit(X_train, y_train)
# KNearest best estimator
knears_neighbors = grid_knears.best_estimator_

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
Fitting oversample data took :936.2669985294342 sec

In [30]: t0 = time.time()
# Support Vector Classifier best estimator
svc_params = {'C': [0.5, 0.7, 0.9, 1], 'kernel': ['rbf', 'poly', 'sigmoid', 'linear']}
grid_svc = GridSearchCV(SVC(), svc_params)
grid_svc.fit(X_train, y_train)

# SVC best estimator
svc = grid_svc.best_estimator_

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
Fitting oversample data took :622.7747869491577 sec

In [31]: t0 = time.time()
# DecisionTree Classifier best estimator
tree_params = {"criterion": ["gini", "entropy"], "min_samples_leaf": list(range(2,7,1))}
grid_tree = GridSearchCV(DecisionTreeClassifier(), tree_params)
grid_tree.fit(X_train, y_train)

# tree best estimator
tree_clf = grid_tree.best_estimator_

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
Fitting oversample data took :190.36638116836548 sec

In [32]: t0 = time.time()
# Random Forest best estimator
rf_params = {"n_estimators":[50,100,150], "criterion": ["gini", "entropy","log_loss"]}

grid_rf = GridSearchCV(RandomForestClassifier(), rf_params)
grid_rf.fit(X_train, y_train)

# random forest best estimator
rf_clf = grid_rf.best_estimator_

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
Fitting oversample data took :2139.439531326294 sec

In [33]: t0 = time.time()
# Naive Bayes best estimator
nb_params = {"var_smoothing": [1e-10, 1e-9, 1e-8]}
grid_nb = GridSearchCV(GaussianNB(), nb_params)
grid_nb.fit(X_train, y_train)

# Gaussian NB best estimator
gaussian_nb = grid_nb.best_estimator_

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
Fitting oversample data took :0.8987891674041748 sec

In [34]: t0 = time.time()
#GB best estimator
gb_params = {"loss": ["log_loss","exponential"]}

grid_gb = GridSearchCV(GradientBoostingClassifier(), gb_params)
grid_gb.fit(X_train, y_train)

# Gradient Boosting Classifier best estimator
gb_clf = grid_gb.best_estimator_

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
Fitting oversample data took :2033.2294204235077 sec
```

**Şekil 5.16** YZ modelleri için en iyi parametrelerinin bulunması

Bu kodlar ile, GridSearchCV kullanılarak sınıflandırıcı modellerin en iyi parametreleri, en iyi doğruluk oranlarını veren seçecek biçimde ayarlaması yapılarak seçilmesi sağlanır. En iyi parametre bilgileri best\_estimator\_ fonksiyonu ile kaydedilir. Ayrıca her bir işlemin kaç saniye sürdüğü hesaplanır.

```

In [35]: t0 = time.time()

# Overfitting case

log_reg_score = cross_val_score(log_reg, X_train, y_train, cv=5)
print('Logistic Regression Cross Validation Score: ', round(log_reg_score.mean() * 100, 4).astype(str) + '%')

knears_score = cross_val_score(kneighbors_neighbors, X_train, y_train, cv=5)
print('Knears Neighbors Cross Validation Score', round(knears_score.mean() * 100, 4).astype(str) + '%')

svc_score = cross_val_score(svc, X_train, y_train, cv=5)
print('Support Vector Classifier Cross Validation Score', round(svc_score.mean() * 100, 4).astype(str) + '%')

tree_score = cross_val_score(tree_clf, X_train, y_train, cv=5)
print('DecisionTree Classifier Cross Validation Score', round(tree_score.mean() * 100, 4).astype(str) + '%')

rf_score = cross_val_score(rf_clf, X_train, y_train, cv=5)
print('Random Forest Classifier Cross Validation Score', round(rf_score.mean() * 100, 4).astype(str) + '%')

nb_score = cross_val_score(gaussian_nb, X_train, y_train, cv=5)
print('Naive Bayes Cross Validation Score', round(nb_score.mean() * 100, 4).astype(str) + '%')

gb_score = cross_val_score(gb_clf, X_train, y_train, cv=5)
print('Gradient Boosting Cross Validation Score', round(nb_score.mean() * 100, 4).astype(str) + '%')

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))

Logistic Regression Cross Validation Score: 99.9538%
Knears Neighbors Cross Validation Score 99.9636%
Support Vector Classifier Cross Validation Score 99.9567%
DecisionTree Classifier Cross Validation Score 99.9489%
Random Forest Classifier Cross Validation Score 99.9597%
Naive Bayes Cross Validation Score 98.9113%
Gradient Boosting Cross Validation Score 98.9113%
Fitting oversample data took :1059.5587484836578 sec

```

**Şekil 5.17** En iyi parametrelere sahip modellerin çapraz doğrulama skorları

Bu kod parçası, eğitilmiş modeller için çapraz doğrulama skorlarını hesaplar ve ekrana yazdırır. Çapraz doğrulama skorları, her bir sınıflandırıcı modelinin genel performansını değerlendirmek için kullanılır. Adımlar şu şekildedir:

1. Her bir sınıflandırıcı için `cross_val_score()` fonksiyonu kullanılarak çapraz doğrulama yapılır.
2. Çapraz doğrulama sonuçları ekrana yazdırılır.

Bu adımlar, eğitilmiş modellerin performansını ölçmek ve genelleme yeteneklerini değerlendirmek için kullanılır. Çapraz doğrulama skorları, bir modelin genel performansını ölçerken aşırı uydurma (overfitting) durumlarını belirlemeye de yardımcı olabilir. Yüksek çapraz doğrulama skorları genellikle iyi bir genelleme yeteneğini işaret ederken, farklı modellerin skorları karşılaştırılarak en uygun modelin seçilmesine yardımcı olabilir.

```

In [36]: t0 = time.time()

X = df.drop('Class', axis=1)
y = df['Class']

sss = StratifiedKFold(n_splits=5, random_state=None, shuffle=False)

for train_index, test_index in sss.split(X, y):
    Xtrain, Xtest = X.iloc[train_index], X.iloc[test_index]
    ytrain, ytest = y.iloc[train_index], y.iloc[test_index]

accuracy = []
precision = []
recall = []
f1 = []
auc = []

# Implementing NearMiss Technique
# Distribution of NearMiss (Just to see how it distributes the labels, we won't use these variables)
nm = NearMiss(sampling_strategy='majority')
X_nearmiss, y_nearmiss = nm.fit_resample(X.values, y.values)
print('NearMiss Label Distribution: {}'.format(counter(y_nearmiss)))

# Cross validating the right way
for train, test in sss.split(Xtrain, ytrain):
    pipeline = imbalanced_make_pipeline(NearMiss(sampling_strategy='majority'), log_reg)
    model = pipeline.fit(Xtrain.iloc[train], ytrain.iloc[train])
    prediction = model.predict(Xtrain.iloc[test])

    accuracy.append(pipeline.score(Xtrain.iloc[test], ytrain.iloc[test]))
    precision.append(precision_score(ytrain.iloc[test], prediction))
    recall.append(recall_score(ytrain.iloc[test], prediction))
    f1.append(f1_score(ytrain.iloc[test], prediction))
    auc.append(roc_auc_score(ytrain.iloc[test], prediction))

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))

NearMiss Label Distribution: Counter({0: 492, 1: 492})
Fitting oversample data took :2.9197844372558594 sec

```

**Şekil 5.18** NearMiss çapraz doğrulama skorlarının yazdırılması

Bu kod parçası, NearMiss örneklem teknğini kullanarak dengesiz veri seti üzerinde sınıflandırma modeli eğitir ve çapraz doğrulama skorlarını değerlendirir. Bu kod parçası, dengesiz veri seti üzerinde sınıflandırma modelini eğitmek ve NearMiss örneklem teknığının nasıl kullanılabileceğini göstermek için tasarlanmıştır. NearMiss, örneklem sayısını azaltarak, sınıf dengesizliğini gidermeye yönelik bir

örnekleme tekniğidir. Bu işlem, dengesizlikle başa çıkmak ve daha iyi bir sınıflandırma modeli oluşturmak için kullanılır.

```
In [39]: t0 = time.time()

# Create a DataFrame with all the scores and the classifiers names.

log_reg_pred = cross_val_predict(log_reg, X_train, y_train, cv=5,
                                 method="decision_function")

knears_pred = cross_val_predict(knears_neighbors, X_train, y_train, cv=5)

svc_pred = cross_val_predict(svc, X_train, y_train, cv=5,
                             method="decision_function")

tree_pred = cross_val_predict(tree_clf, X_train, y_train, cv=5)

rf_pred = cross_val_predict(rf_clf, X_train, y_train, cv=5)

nb_pred = cross_val_predict(gaussian_nb, X_train, y_train, cv=5)

gb_pred = cross_val_predict(gb_clf, X_train, y_train, cv=5)

print('Logistic Regression: ', roc_auc_score(y_train, log_reg_pred))
print('KNear Neighbors: ', roc_auc_score(y_train, knears_pred))
print('Support Vector Classifier: ', roc_auc_score(y_train, svc_pred))
print('Decision Tree Classifier: ', roc_auc_score(y_train, tree_pred))
print('Random Forest Classifier: ', roc_auc_score(y_train, rf_pred))
print('Naive Bayes Classifier: ', roc_auc_score(y_train, nb_pred))
print('Gradient Boosting Classifier: ', roc_auc_score(y_train, gb_pred))

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))

Logistic Regression: 0.9633629651506201
KNear Neighbors: 0.8349803110848592
Support Vector Classifier: 0.9142049616066155
Decision Tree Classifier: 0.7799655443985037
Random Forest Classifier: 0.8199803110848594
Naive Bayes Classifier: 0.8846593817688646
Gradient Boosting Classifier: 0.8099655443985037
Fitting oversample data took :1068.0876171588898 sec
```

**Şekil 5.19** En iyi parametrelere sahip YZ modellerinin auc skorlarının bulunması

Bu kod parçası, eğitilmiş sınıflandırıcıların, çapraz doğrulama kullanarak tahminleri ve bu tahminlerin ROC-AUC skorlarını hesaplar. Bu adımlar, eğitilmiş sınıflandırıcıların sınıflandırma performansını ROC-AUC skorları üzerinden ölçmeye yönelikdir. ROC-AUC skoru, sınıflandırıcının sınıflandırma yeteneğini değerlendirmek için kullanılan bir metiktir. ROC eğrisi altındaki alan (AUC), sınıflandırıcının sınıflandırma performansının ölçüsüdür. Bu skorlar, farklı sınıflandırıcıların performanslarını karşılaştırmak ve en iyi modeli seçmek için kullanılabilir.

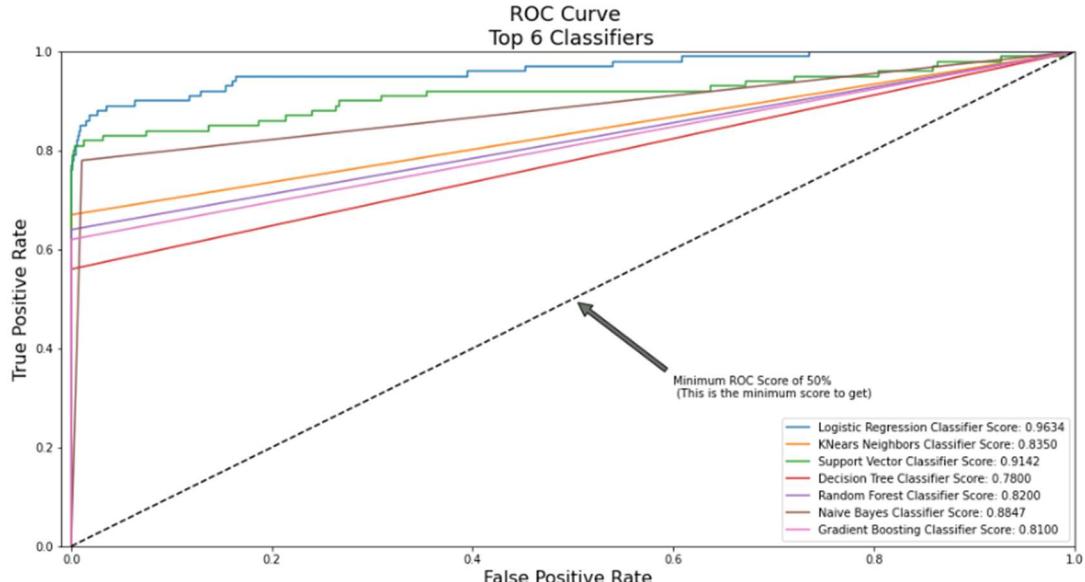
```
In [73]: t0 = time.time()

log_fpr, log_tpr, log_threshold = roc_curve(y_train, log_reg_pred)
knear_fpr, knear_tpr, knear_threshold = roc_curve(y_train, knears_pred)
svc_fpr, svc_tpr, svc_threshold = roc_curve(y_train, svc_pred)
tree_fpr, tree_tpr, tree_threshold = roc_curve(y_train, tree_pred)
rf_fpr, rf_tpr, rf_threshold = roc_curve(y_train, rf_pred)
nb_fpr, nb_tpr, nb_threshold = roc_curve(y_train, nb_pred)
gb_fpr, gb_tpr, gb_threshold = roc_curve(y_train,gb_pred)

def graph_roc_curve_multiple(log_fpr, log_tpr, knear_fpr, knear_tpr, svc_fpr, svc_tpr, tree_fpr, tree_tpr, rf_fpr, rf_tpr, nb_fpr,
plt.figure(figsize=(16,8))
plt.title('ROC Curve \n Top 6 Classifiers', fontsize=18)
plt.plot(log_fpr, log_tpr, label='Logistic Regression Classifier Score: {:.4f}'.format(roc_auc_score(y_train, log_reg_pred)))
plt.plot(knear_fpr, knear_tpr, label='KNearest Neighbors Classifier Score: {:.4f}'.format(roc_auc_score(y_train, knears_pred)))
plt.plot(svc_fpr, svc_tpr, label='Support Vector Classifier Score: {:.4f}'.format(roc_auc_score(y_train, svc_pred)))
plt.plot(tree_fpr, tree_tpr, label='Decision Tree Classifier Score: {:.4f}'.format(roc_auc_score(y_train, tree_pred)))
plt.plot(rf_fpr, rf_tpr, label = 'Random Forest Classifier Score: {:.4f}'.format(roc_auc_score(y_train, rf_pred)))
plt.plot(nb_fpr, nb_tpr, label = 'Naive Bayes Classifier Score: {:.4f}'.format(roc_auc_score(y_train, nb_pred)))
plt.plot(gb_fpr, gb_tpr, label = 'Gradient Boosting Classifier Score: {:.4f}'.format(roc_auc_score(y_train, gb_pred)))
plt.plot([0, 1], [0, 1], 'k--')
plt.axis([-0.01, 1, 0, 1])
plt.xlabel('False Positive Rate', fontsize=16)
plt.ylabel('True Positive Rate', fontsize=16)
plt.annotate('Minimum ROC Score of 50% \n (This is the minimum score to get)', xy=(0.5, 0.5), xytext=(0.6, 0.3),
            arrowprops=dict(facecolor='#6E726D', shrink=0.05),
            )
plt.legend()

graph_roc_curve_multiple(log_fpr, log_tpr, knear_fpr, knear_tpr, svc_fpr, svc_tpr, tree_fpr, tree_tpr, rf_fpr, rf_tpr, nb_fpr,
plt.show()

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))
```



Fitting oversample data took :0.2917757034301758 sec

**Şekil 5.20** En iyi parametrelere sahip YZ modellerinin ROC grafiklerinin çizdirilmesi

Bu kod parçası, eğitilmiş sınıflandırıcıların ROC eğrilerini bir araya getirerek grafik üzerinde görselleştirir.

Adımlar şu şekildedir:

1. Her bir sınıflandırıcının False Positive Rate (FPR) ve True Positive Rate (TPR) değerlerini hesaplamak için `roc_curve()` fonksiyonu kullanılır.
2. Bu değerler, grafikte çizdirilecek ROC eğrilerini oluşturmak için kullanılır.
3. Grafik üzerinde ROC eğrileri çizilir ve her bir sınıflandırıcının ROC AUC skoru etiket olarak eklenir.
4. Grafik, False Positive Rate'e karşı True Positive Rate'i gösterir. Ayrıca, 0.5'in altındaki ROC eğrisine sahip sınıflandırıcıların performansını değerlendirmek için bir anotasyon eklenmiştir.

Bu grafik, farklı sınıflandırıcıların performansını karşılaştırmak için ROC eğrilerini bir arada gösterir. Her bir eğri, ilgili sınıflandırıcının False Positive Rate ve True Positive Rate değerlerini temsil eder. Bu şekilde, sınıflandırıcıların performansları hakkında görsel bir karşılaştırma yapılabilir.

```
In [55]: t0 = time.time()

precision_logreg, recall_logreg, threshold_logreg = precision_recall_curve(y_train, y_pred_logreg)

# Overfitting Case
print('---' * 45)
print('Overfitting: \n')
print('Recall Score: {:.2f}'.format(recall_score(y_train, y_pred_logreg)))
print('Precision Score: {:.2f}'.format(precision_score(y_train, y_pred_logreg)))
print('F1 Score: {:.2f}'.format(f1_score(y_train, y_pred_logreg)))
print('Accuracy Score: {:.2f}'.format(accuracy_score(y_train, y_pred_logreg)))
print('---' * 45)

# How it should look Like
print('---' * 45)
print('How it should be:\n')
print('Accuracy Score: {:.2f}'.format(np.mean(accuracy)))
print('Precision Score: {:.2f}'.format(np.mean(precision)))
print('Recall Score: {:.2f}'.format(np.mean(recall)))
print('F1 Score: {:.2f}'.format(np.mean(f1)))
print('---' * 45)

t1 = time.time()
print("Fitting oversample data took :{} sec".format(t1 - t0))

-----
Overfitting:
Recall Score: 0.84
Precision Score: 0.00
F1 Score: 0.01
```

**Şekil 5.21** Lojistik regresyon modelinde en iyi parametrelere sahip modeli ile SMOTE uygulanması öncesi doğruluk tahminlemesi yaptırılması

SMOTE uygulanması öncesinde, normal veride her bir modelin accuracy, precision, recall ve f1 skorları değerleri için sistem oluşturuldu ve değerler toplanıp kaydedildi.

```
In [66]: from sklearn.metrics import average_precision_score

# Modelinizin tahmin puanlarını oluşturun (örneğin predict_proba'dan gelen puanlar)
y_score = model.predict_proba(X_test)[:, 1]

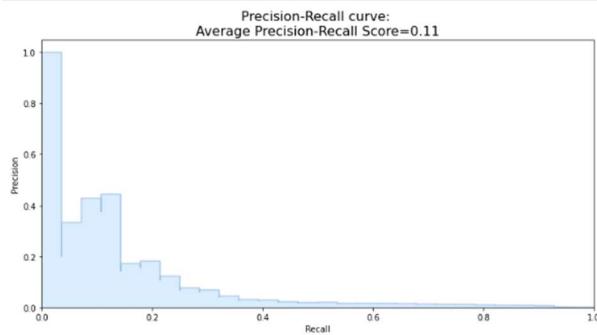
# Gerçek test etiketleri ile ortalama hassasiyet-recall skorunu hesaplayın
average_precision = average_precision_score(y_test, y_score)

fig = plt.figure(figsize=(12,6))

precision, recall, _ = precision_recall_curve(y_test, y_score)

plt.step(recall, precision, color="#004a93", alpha=0.2,
         where='post')
plt.fill_between(recall, precision, step='post', alpha=0.2,
                 color="#48a6ff")

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall curve: \nAverage Precision-Recall Score={:.2f}'.format(
    average_precision), fontsize=16)
plt.show()
```



**Şekil 5.22** SMOTE öncesi modellerin precision-recall skor grafiğinin çizdirilmesi

Elimizdeki tüm modellerin genelinin precision-recall averaj skorlarına bakıldığındá, çok da yüksek olmayan bir sonuç grafiği ile karşılaştık.

```
In [78]: from sklearn.model_selection import train_test_split, StratifiedKFold, RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from imblearn.pipeline import make_pipeline
from imblearn.over_sampling import SMOTE
import numpy as np

# Veri bölmelerini ayıralım
original_xtrain, original_xtest, original_ytrain, original_ytest = train_test_split(features, labels, test_size=0.2, random_state=42)

# StratifiedKFold
sss = StratifiedKFold(n_splits=5, shuffle=True)

accuracy_lst = []
precision_lst = []
recall_lst = []
f1_lst = []
auc_lst = []

log_reg_params = {"penalty": ['l2', 'none'], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
rand_log_reg = RandomizedSearchCV(LogisticRegression(solver='lbfgs'), log_reg_params, n_iter=4)

for train_index, test_index in sss.split(original_xtrain, original_ytrain):
    smote = SMOTE(sampling_strategy='minority')
    X_resampled, y_resampled = smote.fit_resample(original_xtrain[train_index], original_ytrain[train_index])
    model = rand_log_reg.fit(X_resampled, y_resampled)
    best_est = rand_log_reg.best_estimator_
    prediction = best_est.predict(original_xtrain[test_index])

    accuracy_lst.append(np.mean(prediction == original_ytrain[test_index]))
    precision_lst.append(precision_score(original_ytrain[test_index], prediction))
    recall_lst.append(recall_score(original_ytrain[test_index], prediction))
    f1_lst.append(f1_score(original_ytrain[test_index], prediction))
    auc_lst.append(roc_auc_score(original_ytrain[test_index], prediction))

print('*'*45)
print('Accuracy: {:.4f}'.format(np.mean(accuracy_lst)))
print('Precision: {:.4f}'.format(np.mean(precision_lst)))
print('Recall: {:.4f}'.format(np.mean(recall_lst)))
print('F1: {:.4f}'.format(np.mean(f1_lst)))
print('*'*45)
```

**Şekil 5.23** SMOTE uygulaması sonrası Lojistik Regresyon doğruluk skorlarının bulunması

SMOTE uygulamaları işlemeye geçildi. Logistic Regression'dan başlanarak kullandığımız tüm modellere, GridSearchCV işlemi sonucu seçilen en iyi tahminleyici(`best_estimator_`) modelinin kullanılmasıyla Accuracy, Precision, Recall ve F1 skorlarının bulunması işlemi yapıldı.

Sonuç olarak, karşılaştırma tablosu verileri oluşturulması sağlandı.

## 6. Sonuçlar

	Normal Logistic Regression	Normal K-Near Neighbors	Normal SVM	Normal Decision Tree	Normal Random Forest	Normal Gaussian Naive Bayes	Normal Gradient Boosting
Recall	0.84	0.73	0.7	0.65	1	0.77	0.73
Precision	0.0044	1	0.9859	0.9559	1	0.0623	0.9733
F1	0.0087	0.8439	0.8187	0.7738	1	0.1153	0.8343
Accuracy	0.8127	0.9997	0.9997	0.9996	1	0.9884	0.9997
	SMOTE Logistic Regression	SMOTE K-Near Neighbors	SMOTE SVM	SMOTE Decision Tree	SMOTE Random Forest	SMOTE Gaussian Naive Bayes	SMOTE Gradient Boosting
Recall	0.9198	0.8706	0.7998	0.7763	0.9652	0.8932	0.8004
Precision	0.0722	0.4676	0.9959	0.6985	0.9423	0.6357	0.8143
F1	0.1339	0.6076	0.8776	0.8999	0.8996	0.8667	0.9005
Accuracy	0.9794	0.998	0.9248	0.9959	0.9963	0.7774	0.9925

Tablo 6.1: Eğitilmiş bütün modellerin hata ve doğruluk skorlarının tablolaştırılması

Aynı veri değerlerinde en iyi parametreleri ile seçilen bu modeller topluluğunda, SMOTE uygulanmasının bu tip minör etiketli veri oranı çok düşük olan veri setlerinde iyi bir geliştirme yöntemi olduğu, özellikle precision-recall skorlarının yükseltilmesi açısından çok değerli olduğu gözlemlendi.

En başarılı tahmin modeli kısmına gelirsek, Rassal Ormanlar'ın öğrenirken ezberlememe ve doğru tahminleme açısından çok başarılı olduğu görüldü.

Sonuç olarak, Yapay Zeka uygulamaları kullanılarak banka şüpheli hareket tespiti sistemi oluşturma noktasında en efektif modelin Rassal Ormanlar olacağına karar verildi.

## Kaynaklar

- [1] <https://www.btkakademi.gov.tr/portal/course/python-ile-makine-ogrenmesi-11800>  
<https://towardsdatascience.com/> (Erişim Tarihi: Ekim 2023)
- [2] [https://www.researchgate.net/profile/Holger-Wallmeier/publication/326345140/figure/fig1/AS:659206017400834@1534178370785/Schematic-of-machine-learning-Machine-learning-can-be-realized-by-two-different\\_W640.jpg](https://www.researchgate.net/profile/Holger-Wallmeier/publication/326345140/figure/fig1/AS:659206017400834@1534178370785/Schematic-of-machine-learning-Machine-learning-can-be-realized-by-two-different_W640.jpg) (Erişim Tarihi: Ekim 2023)
- [3] <https://anaconda.org/>  
[https://www.youtube.com/watch?v=CS4cs9xVecg&list=PLkDaE6sCZn6Ec-XTbcX1uRg2\\_u4xOEky0](https://www.youtube.com/watch?v=CS4cs9xVecg&list=PLkDaE6sCZn6Ec-XTbcX1uRg2_u4xOEky0) (Erişim Tarihi: Ekim 2023)
- [4] [https://www.kdnuggets.com/wp-content/uploads/wijaya\\_7\\_smote\\_variations\\_oversampling\\_2.png](https://www.kdnuggets.com/wp-content/uploads/wijaya_7_smote_variations_oversampling_2.png)  
[https://www.nrigroupindia.com/e-book/Introduction%20to%20Machine%20Learning%20with%20Python%20\(%20PDFDrive.com%20\)-min.pdf](https://www.nrigroupindia.com/e-book/Introduction%20to%20Machine%20Learning%20with%20Python%20(%20PDFDrive.com%20)-min.pdf) (Erişim Tarihi: Ekim 2023)
- [5] [https://www.mphasis.com/content/dam/mphasis-com/global/en/home/innovation/next-lab/Mphasis\\_Data-Augmentation-for-Tabular-Data\\_Whitepaper.pdf](https://www.mphasis.com/content/dam/mphasis-com/global/en/home/innovation/next-lab/Mphasis_Data-Augmentation-for-Tabular-Data_Whitepaper.pdf) (Erişim Tarihi: Ekim 2023)
- [6] <https://www.sciencedirect.com/science/article/pii/S0306261921010527> (Erişim Tarihi: Ekim 2023)
- [7] <https://www.sciencedirect.com/science/article/pii/S0306261921010527> (Erişim Tarihi: Ekim 2023)
- [8] <https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/Logistic-curve.svg/480px-Logistic-curve.svg.png> (Erişim Tarihi: Ekim 2023)
- [9] <https://static.javatpoint.com/tutorial/machine-learning/images/k-nearest-neighbor-algorithm-for-machine-learning3.png> (Erişim Tarihi: Ekim 2023)
- [10] [https://miro.medium.com/v2/resize:fit:720/format:webp/1\\*eU9PzjVcLNbNEzBC2g\\_iWg.jpeg](https://miro.medium.com/v2/resize:fit:720/format:webp/1*eU9PzjVcLNbNEzBC2g_iWg.jpeg)
- [11] <https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/> (Erişim Tarihi: Ekim 2023)
- [12] [https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine#:~:text=A%20support%20vector%20machine%20\(SVM\)%20is%20a%20machine%20learning%20algorithm,classes%2C%20labels%2C%20or%20outputs](https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine#:~:text=A%20support%20vector%20machine%20(SVM)%20is%20a%20machine%20learning%20algorithm,classes%2C%20labels%2C%20or%20outputs) (Erişim Tarihi: Ekim 2023)
- [13] <https://www.geeksforgeeks.org/how-to-make-better-models-in-python-using-svm-classifier-and-rbf-kernel/> (Erişim Tarihi: Ekim 2023)
- [14] <https://medium.com/analytics-vidhya/introduction-to-svm-and-kernel-trick-part-1-theory-d990e2872ace> (Erişim Tarihi: Ekim 2023)
- [15] <https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107> (Erişim Tarihi: Ekim 2023)
- [16] <https://www.datascienceprophet.com/wp-content/uploads/2020/08/quation.webp> (Erişim Tarihi: Ekim 2023)
- [17] <https://www.datascienceprophet.com/wp-content/uploads/2020/08/equation-1.webp> (Erişim Tarihi: Ekim 2023)
- [18] <https://www.datascienceprophet.com/understanding-the-mathematics-behind-decision-tree-algorithm-part-ii/> (Erişim Tarihi: Ekim 2023)
- [16] <https://www.datascienceprophet.com/wp-content/uploads/2020/08/stat.webp> (Erişim Tarihi: Ekim 2023)
- [19] <https://www.educba.com/decision-tree-advantages-and-disadvantages/> (Erişim Tarihi: Ekim 2023)
- [20] [https://insidelearningmachines.com/advantages\\_and\\_disadvantages\\_of\\_decision\\_trees/](https://insidelearningmachines.com/advantages_and_disadvantages_of_decision_trees/)
- [21] <https://www.math.mcgill.ca/yyang/resources/doc/randomforest.pdf> (Erişim Tarihi: Ekim 2023)
- [22] <https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/> (Erişim Tarihi: Ekim 2023)
- [23] <https://editor.analyticsvidhya.com/uploads/6690812.png> (Erişim Tarihi: Ekim 2023)

- [24] <https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/> (Erişim Tarihi: Ekim 2023)
- [25] <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=A.%20Random%20Forest%20is%20a,model%20is%20a%20decision%20tree.> (Erişim Tarihi: Ekim 2023)
- [26] <https://editor.analyticsvidhya.com/uploads/396233.png> (Erişim Tarihi: Ekim 2023)
- [27] <https://editor.analyticsvidhya.com/uploads/4919118.png> (Erişim Tarihi: Ekim 2023)
- [28] <https://www.upgrad.com/blog/gaussian-naive-bayes/> (Erişim Tarihi: Aralık 2023)
- [29] <https://medium.com/analytics-vidhya/na%C3%AFve-bayes-algorithm-5bf31e9032a2>
- [30] <https://blog.paperspace.com/gradient-boosting-for-classification/> (Erişim Tarihi: Aralık 2023)
- [31] <https://deepchecks.com/question/what-is-the-advantage-of-gradient-boosting/> (Erişim Tarihi: Aralık 2023)
- [32] <https://www.veribilimiokulu.com/veri-hazirliginin-vazgecilmezi-ozellik-olceklendirme/> (Erişim Tarihi: Aralık 2023)
- [33] <https://machinelearningmastery.com/dimensionality-reduction-for-machine-learning/> (Erişim Tarihi: Aralık 2023)
- [34] [https://media.geeksforgeeks.org/wp-content/uploads/Dimensionality\\_Reduction\\_1.jpg](https://media.geeksforgeeks.org/wp-content/uploads/Dimensionality_Reduction_1.jpg) (Erişim Tarihi: Aralık 2023)
- [35] <https://lvdmaaten.github.io/tsne/> (Erişim Tarihi: Aralık 2023)
- [36] <https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-component-analysis-pca-and-how-it-is-used-507186> (Erişim Tarihi: Aralık 2023)
- [37] <https://dataaspirant.com/truncated-svd/> (Erişim Tarihi: Aralık 2023)
- [38] <https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/> (Erişim Tarihi: Aralık 2023)
- [39] <https://machinelearningmastery.com/k-fold-cross-validation/> (Erişim Tarihi: Aralık 2023)
- [40] <https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/#:~:text=AUC%20ROC%20stands%20for%20%E2%80%9CArea,summary%20of%20the%20ROC%20curve.> (Erişim Tarihi: Aralık 2023)
- [41] [https://www.iguazio.com/glossary>true-positive-rate/#:~:text=The%20true%20positive%20rate%20\(TPR\)%20and%20false%20positive%20rate%20\(the%20effectiveness%20of%20a%20model.](https://www.iguazio.com/glossary>true-positive-rate/#:~:text=The%20true%20positive%20rate%20(TPR)%20and%20false%20positive%20rate%20(the%20effectiveness%20of%20a%20model.) (Erişim Tarihi: Aralık 2023)
- [42] [https://miro.medium.com/v2/resize:fit:720/format:webp/1\\*Zcbx-devZbkbD5mgnIWsUg.png](https://miro.medium.com/v2/resize:fit:720/format:webp/1*Zcbx-devZbkbD5mgnIWsUg.png) (Erişim Tarihi: Aralık 2023)
- [43] <https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/> (Erişim Tarihi: Aralık 2023)
- [44] <https://medium.com/codex/mathematical-background-of-lasso-and-ridge-regression-23b74737c817> (Erişim Tarihi: Aralık 2023)
- [45] [https://miro.medium.com/v2/resize:fit:720/format:webp/1\\*kZL56tkhW6WcGXb75JyeuA.png](https://miro.medium.com/v2/resize:fit:720/format:webp/1*kZL56tkhW6WcGXb75JyeuA.png) (Erişim Tarihi: Aralık 2023)
- [46] <https://iq.opengenus.org/elastic-net-regularization/> (Erişim Tarihi: Aralık 2023)
- [47] [https://iq.opengenus.org/content/images/2020/10/elastic\\_cost\\_func.png](https://iq.opengenus.org/content/images/2020/10/elastic_cost_func.png) (Erişim Tarihi: Aralık 2023)

## **Özgeçmiş**

### **Kişisel Bilgiler**

Adı ve Soyadı: Canberk Ustaoglu

### **Eğitim Durumu**

Lisans Öğrenimi: Yıldız Teknik Üniversitesi

### **İş Deneyimi**

Stajlar: Softtech Staj Programı (Ocak 2022 – Nisan 2022),

Interprobe YZ Araştırmaları Stajı ( Temmuz 2022 – Eylül 2022)

Projeler: Google Developers Machine Learning Bootcamp (Ocak 2022 – Nisan 2022),

AIBOSS YZ Projesi (Temmuz 2023 - ....)

Çalıştığı Kurumlar: Interprobe Bilgi Teknolojileri A.Ş. - Yapay Zeka Mühendisi (Eylül 2022 – Aralık 2023)

**İletişim:** [canberk.ustaoglu@std.yildiz.edu.tr](mailto:canberk.ustaoglu@std.yildiz.edu.tr)

+905070609730