T.C.
SAKARYA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

ENDÜSTRİ MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME ÇALIŞMASI

BANKACILIK SEKTÖRÜNDE VERİ MADENCİLİĞİ İLE MÜŞTERİLERİN KREDİ RİSKİNİN TAHMİN EDİLMESİ UYGULAMASI

HAZIRLAYAN

ÖZEN KINAY

MERT İSLAM UĞURLU

DANIŞMAN

DOÇ. DR. MERVE CENGİZ TOKLU

PROF. DR. İBRAHİM ÇİL

HAZİRAN 2023

İÇİNDEKİLER

TEŞEKKÜR

Bitirme tezi çalışmamız boyunca değerli bilgilerini esirgemeyen ve katkılarıyla bize yol gösteren hocalarımız, Sayın Doç. Dr. Merve Cengiz Toklu ve Sayın Prof. Dr. İbrahim Çil'e sonsuz teşekkürlerimizi sunarız.

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

ÖZET

Veri madenciliği büyük veri setlerinin, veri sahibi için yararlı ve anlaşılır olacak biçimde, umulmadık ilişkiler yakalamak ve özgün bir biçimde özetlemek için analiz edilmesidir. Veri madenciliği araçları ve teknikleri, farklı amaçlara hizmet eden farklı işlevlere sahiptir. Bu çalışmada Bankacılık sektöründe müşterilere ait özelliklere bakarak kredi için riskli olup olmadığını müşteriye kredi verilip verilmeyeceğini tahmin etmeye dayalı veri madenciliği çalışmasıdır. Bu çalışma NUMPY, PANDAS, MATPLOTLİB paketleri ve lojistik Regresyon, naive bayes, karar ağacı, kNN algoritmaları kullanılacaktır.

Anahtar Kelimeler: Veri madenciliği, python, kredi riski

ABSTRACT

Data mining is the analysis of large data sets to capture and uniquely summarize unexpected relationships in a way that is useful and understandable to the data owner. Data mining tools and techniques have different functions that serve different purposes. This is a data mining study based on estimating whether a loan will be given to the customer by looking at the characteristics of the customers in the banking sector. In this study, NUMPY, PANDAS, MATPLOTLIB packages and logistic regression, naive bayes, decision tree, kNN algorithm will be used.

Keywords: Data mining, python, credit risk

BÖLÜM 1. GİRİŞ

1.1.Veri Madenciliği Kavramı

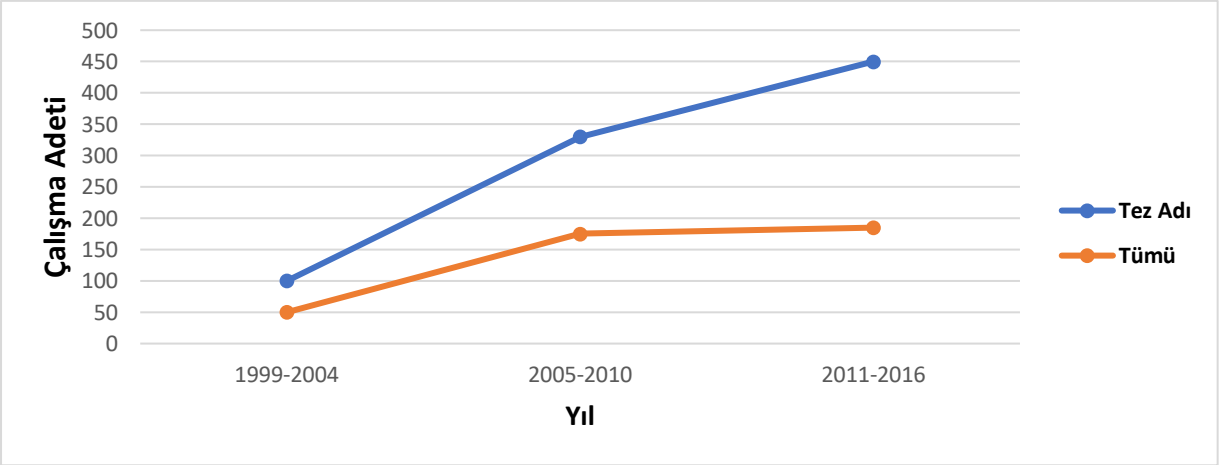
• Kullanıcı tarafından anlaşılabilir ve yararlı verileri yeni yollarla özetlemek ve veriler arasındaki beklenmeyen ilişkileri keşfetmek için büyük ölçekli gözlemsel verilerin analizidir,

• Büyük veri tabanlarında gizli bağlantıların ve genel örüntülerin araştırılmasıdır,

• Verilerden anlamı olan örüntülerin otomatik ya da yarı otomatik olarak keşfedilme sürecidir,

• Bir veri tabanındaki verilerden bilgileri otomatik olarak çıkarmak ve analiz etmek için bir veya daha fazla bilgisayar öğrenme tekniğini uygulama sürecidir (Aydın, 2015).

Veri madenciliği anahtar kelimeleri iki farklı filtre kullanılarak arandı. İlk aramada tez başlıkları içerisinde bilgi aranırken, ikinci aramada aynı kavram tüm alanlarda kritersiz olarak aranmıştır. Şekil 1, veri madenciliği ile ilgili arama sonuçlarını ve dolayısıyla yüksek lisans programlarının sayısını göstermektedir.

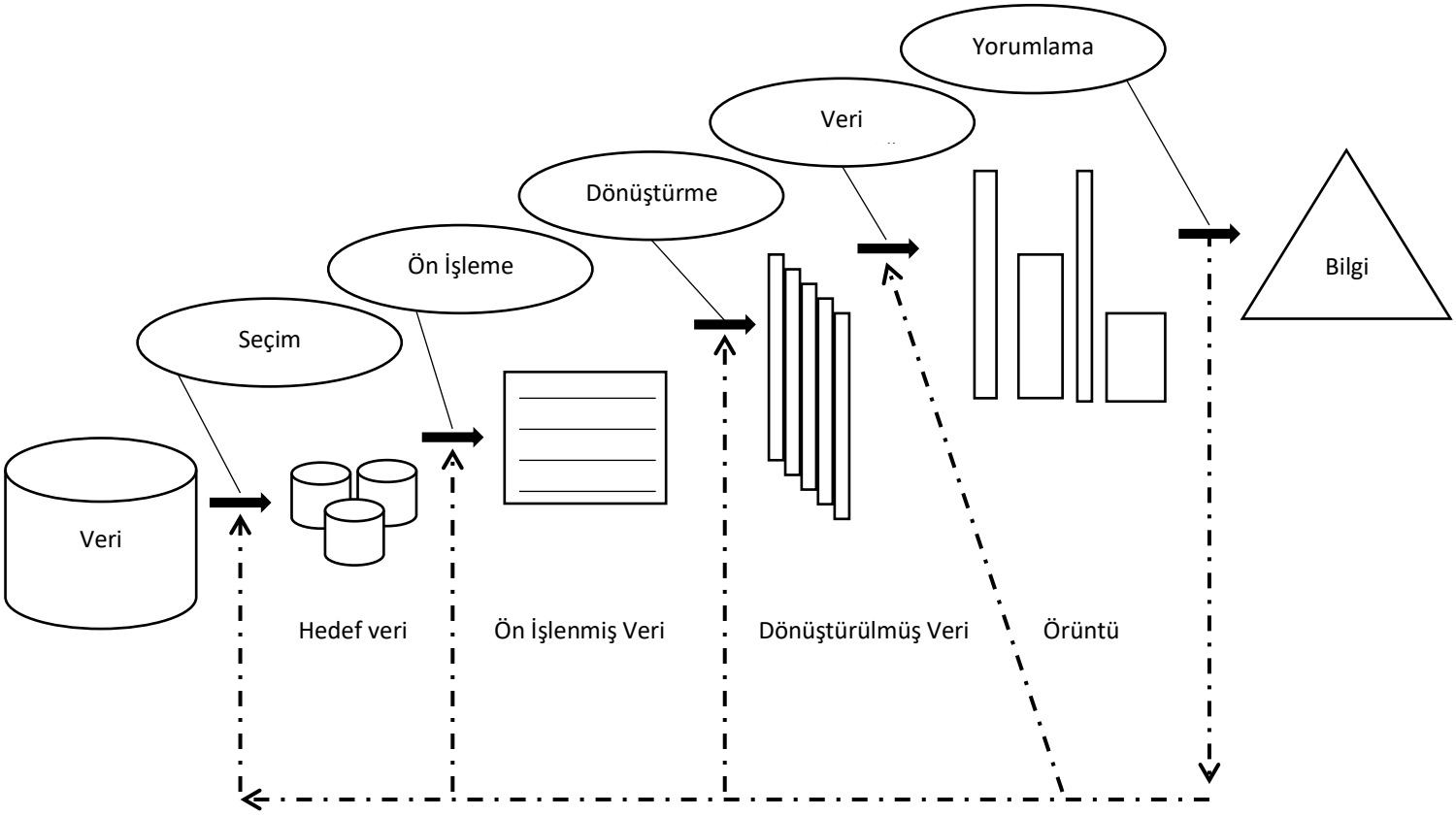


Şekil 1. Yıllara Göre Veri Madenciliği Lisansüstü Çalışmaları

1.2. Veri Madenciliği Süreci

1. Problemin tanımlanması,
2. Verilerin hazırlanması,
3. Modelin kurulması ve değerlendirilmesi,
4. Modelin kullanılması,
5. Modelin izlenmesi.

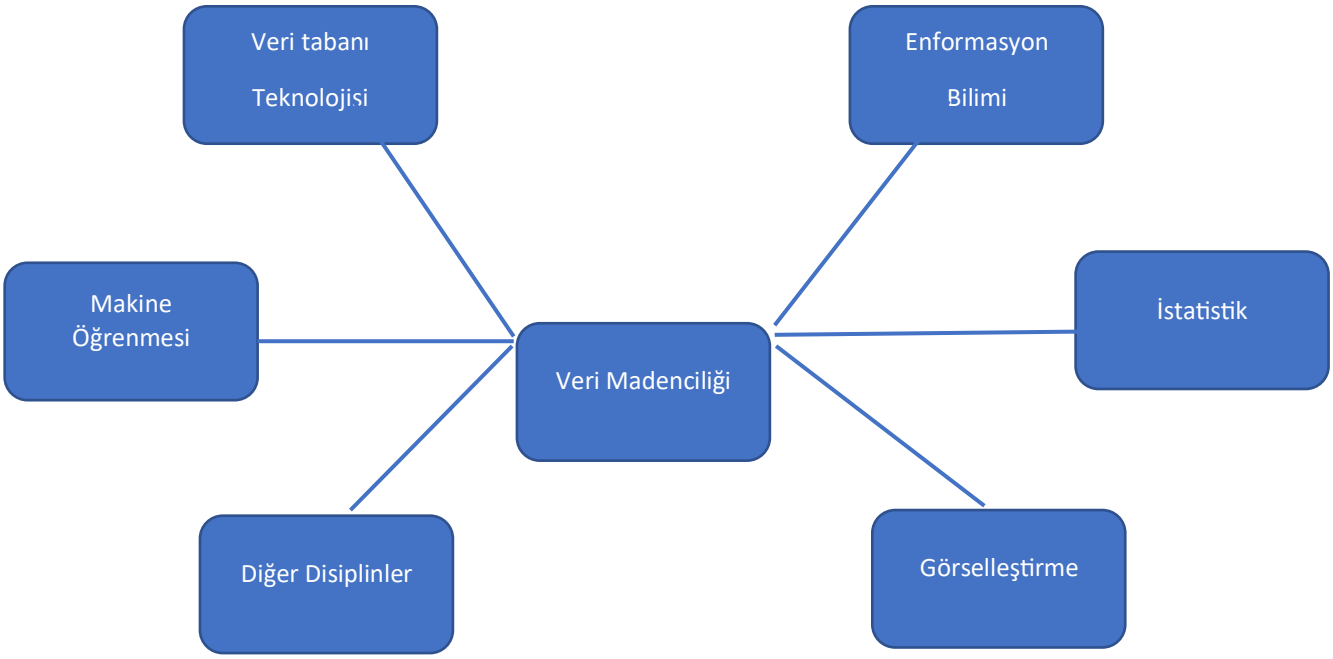
Veri madenciliğinin ilk aşamasında araştırılacak bilgiye karar verilir. Bu konuda sonuca ulaşmak için net ve planlı olmakta fayda var. Bir nesne tanımlandıktan sonra, veriler sınıflandırıcıya yerleştirilir ve veri işleme için en uygun veri tabanları seçilir. Araştırmanın yapıldığı veri tabanı, temizlenmeli ve hedef bilgiden uzaklaştıran, alakasız bilgilerden ayrılmalıdır. Sonrasında, hedefe en uygun olarak seçilen veri tabanında toplanan en doğru veriler, farklı yöntemlerle örüntülü bir şekilde işlenir. Elde edilen sağlıklı, işlenmiş ve ilgili veriler, amaçlanan kullanıma hazır hale gelir. Bu büyük, gizli ve ham verilerin faydalı bilgilere dönüştürülmesi ve ardından analiz edilmesi program ve teknikler tarafından otomatik olarak yapılmakta ve teknolojinin verdiği güçle birçok alanda avantajlar sunmaktadır (Taşçı & Şamlı, 2020).



Şekil 2. Veri Tabanlarında Bilgi Keşfi

Bilgi keşfi kavramı, büyük veri tabanlarında aşamalı bir şekilde bilginin ortaya çıkarılması sürecini ifade eder. Veri tabanlarında bilgi keşfi, büyük miktardaki verilerin analiz edilmesi ve içerisindeki gizli veya anlamlı desenlerin, ilişkilerin veya trendlerin bulunması için kullanılan bir yaklaşımdır. Veri madenciliği bu arama sürecindeki belki de en önemli/kilit adımdır. Veri madenciliği birçok farklı alan tarafından desteklenmektedir; istatistik, makine öğrenmesi, yapay zekâ, örüntü tanıma alanlarındaki gelişmelerden de beslenmektedir.

1.3. Veri Madenciliğini Oluşturan Disiplinler

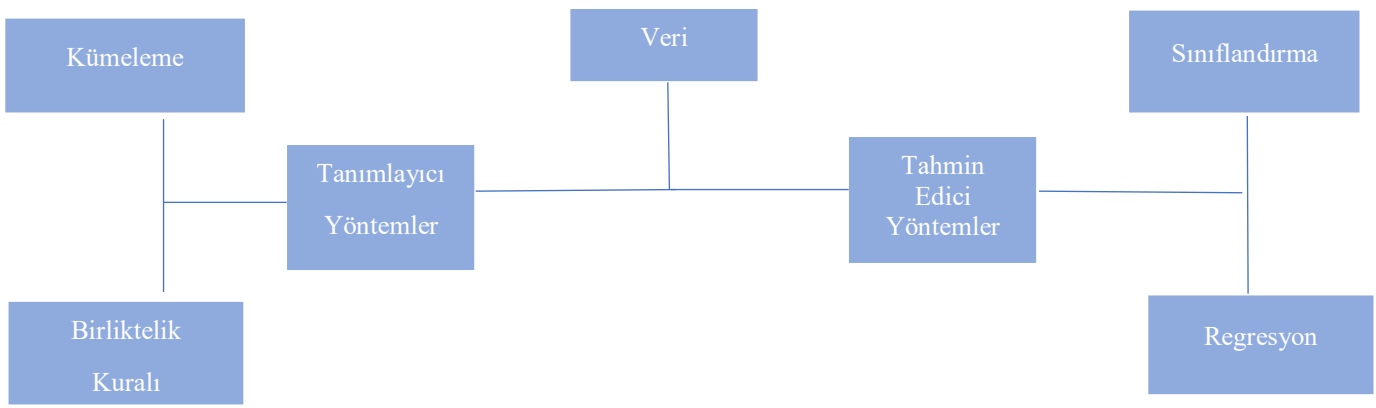


Şekil 3. Veri Madenciliğini Oluşturan Disiplinler

(Özbay, 2015).

1.4. Veri Madenciliği Yöntemleri

Veri madenciliği temel olarak, yazılım teknikleri kullanılarak verilerin analizidir. Bu analiz doğrulamaya dayalı yöntemle veya keşfetmeye dayalı yöntem kullanılarak yapılabilir. Bu yöntemler, verinin bilgiye dönüşümünü açıklamakla ilgilidir.



Şekil 4. Veri Madenciliği Yöntemleri

1.4.1. Tahmin edici yöntemler

Bilinen verileri kullanarak bilinmeyen bir değeri tahmin etmeye çalışırlar. Tahminin amacı, bazı değişkenlerin (örnek. kullanıcı müşteri) gelecekte nasıl davranacağını bilmek için verilerdeki kalıpları belirlemektir. Bu amaca, sınıflandırma regresyonu ve zaman serileri gibi yöntemler kullanılarak ulaşılır.

1.4.1.1. Sınıflandırma

Veri tabanlarındaki işlemler değişkenler tanımlanarak gerçekleştirilir. Değişkenler, bir niceliği veya ölçülebilir niteliği temsil eden ve değişkenlik gösteren sembolik değerlerdir. Belirli veri türlerini depolarlar. Bir sistemin işletilmesinde ya da bir deneyin gidişatı doğrultusunda değişebilecek değerlerin yerlerini tutan varsayımlardır.

1.4.1.2. Regresyon

Regresyon analizi, bir bağımlı değişken ile bir veya daha fazla bağımsız değişken arasındaki ilişkiyi bir fonksiyon şeklinde yazar ve bağımlı değişkenin o fonksiyonu kullanarak ulaşabileceği değerleri tahmin etmeye çalışır.

1.4.2. Tanımlayıcı Yöntemler

Tanımlayıcı modellerin amacı, karar vermeyi desteklemek için kullanılan verilerdeki desenleri tanımlamaktır. Bu modeller, büyük veri kümelerindeki desenleri ve ilişkileri tespit etmek ve verileri anlamlı bir şekilde ilişkilendirmek için kullanılır. Bu süreçte, veri analizi teknikleri ve istatistiksel yöntemler kullanılarak verilerde gizli olan yapılar ve ilişkiler ortaya çıkarılır. Tanımlayıcı modeller, veriye dayalı karar alma süreçlerinde anlamlı bilgi sağlamak ve verilerin anlamlandırılmasını kolaylaştırmak için kullanılır. Bu sayede, büyük veri kümelerindeki önemli desenler ve ilişkiler belirlenebilir ve bu bilgiler karar verme süreçlerine rehberlik edebilir.

1.4.2.1. Kümeleme

Veri tabanlarındaki veriler çok karmaşık yapılara ve çok büyük boyutlara sahip olabilir. Onlara uygulanan veri madenciliği teknikleri, başarılı olmak ve bu karmaşık verilerden anlamlı sonuçlar çıkarmak için mücadele ediyor. Bu tür problemlerde izlenen yöntem, verileri parçalara ayırmak, bölmek ve önce alt parçalardan başlayarak çözüm üretmeye başlamaktır.

1.4.2.2. Birliktelik Kuralı

En yaygın kullanılan veri madenciliği tekniklerinden biri olan birliktelik kuralı, belirli veri yapıları türleri arasındaki olası ilişki biçimini tanımlamak için kullanılan bir yöntemdir (Akçay, 2014).

1.5. Veri Madenciliğinin Uygulama Alanları

Veri madenciliğinin kullanımı belirli uygulama alanlarıyla sınırlandırılamaz. Veri madenciliği bilginin üretildiği ve saklandığı her yerde kullanılabilir. (Özbay, 2015).

Tablo 1. Veri madenciliği birçok farklı araştırma alanında kullanılmaktadır

(Emre & Erol, 2017).

Benzerlik	Ayrıntılar
	Uygulama
Sağlık/Tıp	Hastalık tahmini Hastalık analitiği Hastalık etkisi analizi İlaçların yan etkisi
Biyoenformatik	Biyolojik veri analizi
Astronomi	Astronomik gözlemlerle analiz etme
	İklim modellemeleri
Telekomünikasyon	Müşteri kayıp analizi Müşteriye özel kampanya analizi
Üretim	Hata tespiti
Yatırım	Yatırım portfolyosu yönetimi
Pazarlama	Müşteri davranış tahminleri Olurlu kampanya geliştirme Farklı müşteri gruplarının analizi Pazar sepeti analizi
Bankacılık	Dolandırıcılık tespiti Kara para aklama tespiti Müşteri sınıflandırması Pazar bölümlendirme
Sigortacılık	Sahtekarlık algılama Müşteri analitiği
Yapısal olmayan veri analizi	Sosyal medya verisi Web sitelerinin analizi
Eğitim	Başarı nedenlerinin belirlenmesi

1.6. Veri Madenciliğinde Kullanılan Algoritmalar

1.6.1. Lojistik Regresyon

Lojistik Regresyon, sınıflandırma problemlerini çözmek için yaygın olarak kullanılan bir makine öğrenimi algoritmasıdır. Bu algoritma, girdi verileriyle belirli bir çıktı sınıfı arasındaki ilişkiyi modeller ve yeni verileri bu modeli kullanarak sınıflandırır. Lojistik regresyon, bir sigmoid fonksiyonu kullanarak veri noktasını iki veya daha fazla sınıfa ayırır. Giriş özniteliklerinin ağırlıklarını ve bir sapmayı (bias) öğrenir (Mitchell, 1997).Eğitim veri kümesindeki özniteliklerle sınıf etiketleri arasındaki ilişkiyi modellemeye çalışır (James, 2013). Eğitim sonucunda, veri noktalarını sınıflandırmak için elde edilen ağırlıklar kullanılır. Bu algoritmalar, çeşitli makine öğrenimi problemlerinde kullanılır ve Python'da çeşitli kütüphanelerle uygulanabilirler (Pedregosa, 2011).

1.6.2. Naive Bayes

Naive Bayes, temel bir sınıflandırma algoritmasıdır ve Bayes teoremi prensibine dayanır. Naive Bayes, veri noktasının sınıfını belirlemek için özniteliklerin bağımsız olduğunu varsayar. Bu nedenle "naive" (saf) olarak adlandırılır (Raschka, 2017). Bayes teoremi kullanılarak sınıf olasılıkları hesaplanır ve en yüksek olasılığa sahip sınıf etiketi tahmin olarak seçilir. Sınıflandırma için çok kullanılan bir algoritmadır ve özellikle metin sınıflandırma gibi alanlarda etkilidir.

1.6.3. Karar Ağacı Algoritması

Karar Ağacı Algoritması hem sınıflandırma hem de regresyon problemlerini çözmek için kullanılan bir makine öğrenimi algoritmasıdır. Bu algoritma, veri tabanındaki özelliklerin değerlerine dayanarak bir dizi karar kuralı oluşturur. Veri kümesindeki öznitelikleri kullanarak bir karar ağacı oluşturur. Her iç düğüm, bir öznitelikle temsil edilir ve dallara ayrılır (Hastie, 2009).Yaprak düğümlerinde sınıf etiketleri veya tahmin değerleri bulunur. Karar ağacı, veri noktalarını sınıflandırmak veya tahmin yapmak için ağaç yapısını kullanır.

1.6.4. KNN En Yakın Komşu Algoritması

k-En Yakın Komşu (kNN), temel bir sınıflandırma ve regresyon algoritmasıdır. Sınıflandırma için kullanıldığında, bir veri noktasını çevresindeki en yakın k komşu veri noktalarının çoğunluğuna dayanarak sınıflandırır. Regresyon için kullanıldığında, veri noktasını

çevresindeki en yakın k komşu veri noktasının ortalamasını kullanarak tahmin yapar. Öklidyen mesafe veya başka bir benzerlik metriği kullanılarak veri noktaları arasındaki uzaklık hesaplanır (Raschka, 2017).

1.7. Literatür Araştırması

Kıyas Kayaalp tarafından 2007 yılında yapılan bir yüksek lisans çalışmasında, ç fazlı bir asenkron motorun sargıları arasında oluşabilecek kısa devre veya izolasyon arızalarının yanı sıra motor şaftında meydana gelebilecek mekanik dengesizlik arızalarının tespiti için veri madenciliği teknikleri kullanılmıştır. Ali İnan'ın 2006 yılında yaptığı araştırma şu sonuçlarla sonuçlanmıştır: Bireylerin konum verilerinin toplanması, kullanılması ve dağıtılmasının gizliliğine ilişkin endişeler, veri madenciliği tekniklerinin mekansal-zamansal bilgi içeren verilere uygulanmasının önündeki tek engeldir. İnsanların seyahat yörüngeleri, genel ev ve iş adresleri kullanılarak kimlikleriyle eşleştirilebildiğinden, kimlik etiketlerini verilerden kaldırmak, bir kişinin mahremiyetini sağlamak için yeterli değildir. Ayrıca, mahremiyeti koruyan mevcut veri madenciliği teknikleri yeterli değildir, çünkü bu tekniklerin uzay-zamansal bilgi içeren verilere uygulanması, ardışık konum gözlemlerinin birbirinden bağımsız olmasını gerektirir.

Gökhan Yavaş tarafından yapılan başka bir çalışma (2003), veri madenciliği kullanarak mobil kullanıcıların hareket modellerini çıkarmak ve bu modelleri mobil kullanıcıların gelecekteki hareketlerini tahmin etmek için kullanmak için yeni bir algoritma geliştirdi. Bu üç adımlı algoritmanın ilk adımında, veri madenciliği kullanılarak kullanıcıların önceden kaydedilmiş hareket yollarından kullanıcı hareket örüntüleri çıkarılmaktadır. İkinci adımda hareket modellerinden hareket kuralları oluşturulur ve son adımda bu hareket kuralları kullanılarak kullanıcının hücreler arası bir sonraki hareketi tahmin edilir. Sunulan algoritmanın performansı simülasyonlar kullanılarak iki farklı değerlendirme yöntemiyle karşılaştırılmıştır.

N. Duru ve M. Canbay, 2007 yılında sismik verilerin veri madenciliği kullanılarak analizi üzerine bir çalışma yürüttüler. Bu çalışma, sismik veriler kullanılarak seçilen bir bölgenin sismik tehlikesine ilişkin bir veri madenciliği çalışmasını içermektedir. 2004 yılında Yaşar Doğan, Deniz Harp Okulu'nda su altı taktik sensör ağları için veri madenciliğine dayalı bir hedef sınıflandırma çalışması üretti. Bu çalışmada, denizaltıları, küçük su altı araçlarını, su

altı mayınlarını ve dalgıçları açık, sığ ve çok sığ suda sınıflandırmak için ucuz mikrosensörler kullanıldı (Savaş, Topaloğlu, & Yılmaz, 2012).

BÖLÜM 2. PROBLEMİN TANIMI

Bu problemde ele alınan konu Banka da bilgileri kayıtlı olan insanların kredi geçmişi (alınan krediler, zamanında ödenen krediler, gecikmeler, kritik hesaplar), kredi miktarı, bu bankadaki kredi sayısı, yaşı, harcanabilen gelir yüzdesi, mülk (gayrimenkul), kefilleri, TL cinsinden mevcut çek hesabının durumu, konut sayısı, bakımını sağlamakla yükümlü kişi sayısı gibi özelliklerine bakarak bir analiz sonucu kişinin krediye yeterliliği tespit edilmeye çalışılmasıdır. Bu yaptığımız analiz doğrultusunda bankalar bu bilgilere bakarak kişilerin telefon numaralarına veya ikametgâh adreslerine kredi kartı yollamak veya direkt kişi ile telefonla iletişime geçilerek bilgilendirme yapabilmektedir veya bankaya gelen bir müşteri kredi istediği zaman onun bilgilerini analiz ederek kredi riski taşıyıp taşımadığını tespit edebilmektedir.

BÖLÜM 3. PROBLEMİN ÇÖZÜMÜ TAKİP EDİLECEK AŞAMALAR

Çözüm Aşamaları	Yapılan İş	Çözüm Aracı, Yöntem, Teknik, Program vs.
1	Veri Setinin Alınması	Kaggle
2	Veri Setinin İşlenmesi	Python Numpy Pandas Matplotlib
3	Veri Setinin Analizi	Logistic Regression , Naive Bayes , kNN , Decision Tree

BÖLÜM 4. UYGULAMA

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

       import re
       from sklearn.model_selection import train_test_split
       from sklearn.preprocessing import LabelEncoder, MinMaxScaler
       from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

       from sklearn.naive_bayes import GaussianNB
       from sklearn.tree import DecisionTreeClassifier
       from sklearn.neighbors import KNeighborsClassifier
       from sklearn.linear_model import LogisticRegression


In[2]: df = pd.read_csv("credit_customers.csv") #csv dosyasını çağırma
In[3]: df.head() #verilerimizin ilk 5 satırına bakmak için kullanılır
```

In[4]: df.iloc[:,:8].head()

Out[4]:

Tablo 2. Veri Seti Çıktısı

	checking_status	duration	credit_history	purpose	credit_amount	savings_status	employment	installment_commitment
0	<0	6	critical/other existing credit	radon/tv	1169	no known savings	>=7	4
1	0<=X<200	48	existing paid	radio/tv	5951	<100	1<=X<4	2
2	no checking	12	critical/other existing credit	education	2096	<100	4<=X<7	2
3	<0	42	existing paid	furniture/equipment	7882	<100	4<=X<7	2
4	<0	24	delayed previously	new car	4870	<100	1<=X<4	3

In[5]: df.iloc[:, 8:17].head()

Out[5]:

Tablo 3. Veri Seti Çıktısı

	personal_status	other_parties	residence_since	property_magnitude	age	other_payment_plans	housing	existing_credits
0	male single	none	4	real estate	67	none	own	2
1	female div/dep/mar	none	2	real estate	22	none	own	1
2	male single	none	3	real estate	49	none	own	1
3	male single	guarantor	4	life insurance	45	none	for free	1
4	male single	none	4	no known property	53	none	for free	2

In[6]: df.iloc[:, 17:21].head()

Out[6]:

Tablo 4. Veri Seti Çıktısı

	num_dependents	own_telephone	foreign_worker	class
0	1	yes	yes	good
1	1	none	yes	bad
2	2	none	yes	good
3	2	none	yes	good
4	2	none	yes	bad

```
In[7] : df.rename(columns={"checking_status":"KD",
                    "duration":"S",
                   "credit_history":"KG",
                    "purpose":"KA",
                   "credit_amount":"KM",
                   "savings_status":"TD",
                   "employment":"CS",
                   "installment_commitment":"HGY",
                   "personal_status":"KDEB",
                   "other_parties":"DK",
                    "residence_since":"XYİ",
                    "property_magnitude":"M",
                    "age":"Y",
                    "other_payment_plans":"DTP",
                    "housing":"KKK",
                    "existing_credits":"MK",
                    "job":"is",
                    "num_dependents":"BKS",
                    "own_telephone":"T",
                    "foreign_worker":"Yİ",
                    "class":"siniflandirma"
```

```
        },inplace=True)
#mevcut sütun isimlerini değiştirildi
```

Sütun İsimleri

1. KD = Kontrol Durumu
2. S = Süre
3. KD = Kredi Geçmişi
4. KA = Kredinin Amacı
5. KM = Kredi Miktarı
6. TD = Tasarruf Durumu
7. CS = Çalışma Süresi
8. HGY = Harcanabilir Gelir Yüzdesi
9. DK = Diğer Kefiller
10. M = Mülk
11. Y = Yaş
12. KKK = Konut Kira/Kendi
13. MK = Mevcut Krediler
14. is = iş
15. BKS = Baktığı kişi sayısı
16. T = Telefon
17. Yİ = Yabancı İşçi
18. siniflandirma = Hedef değişken sütunumuz

In[8]: df.drop('XYİ',axis=1, inplace=True) # X yilindan beri ikametgah

df.drop('DTP',axis=1, inplace=True) # Diger taksit planlari

df.drop('KDEB',axis=1, inplace=True) #Kişisel durum cinsiyet / evli bekar

In[9]: df.head()
Out[9]:

Tablo 5. Sütun isimlerinin değiştirilmesi halindeki çıktı

	KD	S	KG	KA	KM	TD	CS	HGY	DK	M	Y	KKK	MK	is	BKS	T	Yİ	siniflandirma
0	<0	6	critical/other existing credit	radio/tv	1169	no known savings	>=7	4	none	real estate	67	own	2	skilled	1	yes	yes	good
1	0<=X<200	48	existing paid	radio/tv	5951	<100	1<=X<4	2	none	real estate	22	own	1	skilled	1	none	yes	bad
2	no checking	12	critical/other existing credit	education	2096	<100	4<=X<7	2	none	real estate	49	own	1	unskilled resident	2	none	yes	good
3	<0	42	existing paid	furniture/equipment	7882	<100	4<=X<7	2	guarantor	life insurance	45	for free	1	skilled	2	none	yes	good
4	<0	24	delayed previously	new car	4870	<100	1<=X<4	3	none	no known property	53	for free	2	skilled	2	none	yes	bad

In[10]: df["siniflandirma"] = df["siniflandirma"].replace("bad","kotu")

df["siniflandirma"] = df["siniflandirma"].replace("good","iyi")

#sınıflandırma hedef sütunumuzdaki bad kötü , good iyi olarak değiştirildi

In[11]: df.shape #satır ve sütun sayımız

Out[11]: (1000, 18)

In[12]: df.info()

#veri setimiz hakkındaki genel bilgileri ve veri tiplerini görmemiz için

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 18 columns):

#	Column	Non-Null Count	Dtype
0	KD	1000 non-null	object
1	S	1000 non-null	float64
2	KG	1000 non-null	object
3	KA	1000 non-null	object
4	KM	1000 non-null	float64
5	TD	1000 non-null	object
6	CS	1000 non-null	object
7	HGY	1000 non-null	float64
8	DK	1000 non-null	object
9	M	1000 non-null	object
10	Y	1000 non-null	float64
11	KKK	1000 non-null	object

```
 12  MK          1000 non-null   float64
 13  is          1000 non-null   object
 14  BKS         1000 non-null   float64
 15  T           1000 non-null   object
 16  Yİ          1000 non-null   object
 17  siniflandirma  1000 non-null   object
dtypes: float64(6), object(12)
memory usage: 140.8+ KB
```

In[13]: df.describe() #sayısal özniteliklerimizin istatiksel özelliklerine bakıldı

#kredi ödeme süresi ortalama 20.903000 min 4 max 72 ay std sapması ise 12.05

#kredi miktarı ortalama 3271.258 min 250 max 18424 std sapması ise 2822.73

Out[13]:

Tablo 6. Veri Setimizdeki Sayısal özniteliklerimizin İstatistiksel Özellikleri

	S	KM	HGY	Y	MK	BKS
count	1000	1000	1000	1000	1000	1000
mean	20,903	3271,258	2,973	35,546	1,407	1,155
std	12,05881	2822,737	1,118715	11,37547	0,577654	0,362086
min	4	250	1	19	1	1
25%	12	1365,5	2	27	1	1
50%	18	2319,5	3	33	1	1
75%	24	3972,25	4	42	2	1
max	72	18424	4	75	4	2

In[14]: sns.countplot(x = "siniflandirma",data=df)

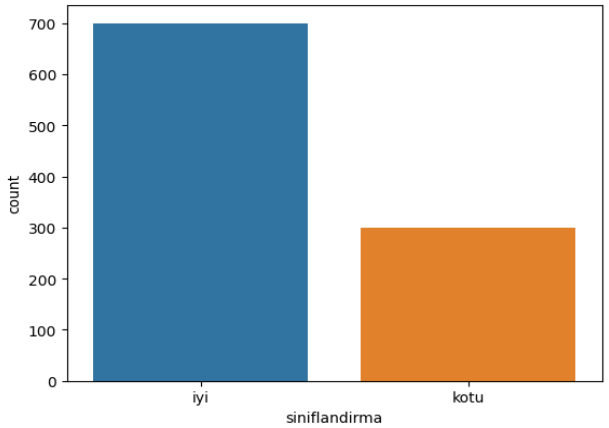
df.loc[:,"siniflandirma"].value_counts()

#sınıflandırma değerlerinin sayılarını görselleştirmek için kullanıldı

Out[14]: iyi 700

kotu 300

Name: siniflandirma, dtype: int64



Şekil 5. Sınıflandırma Değerleri

In[15]: df.isnull().sum()

#Boş veri kayıp veri var mı diye bakıldı

#.sum() her bir sütundaki eksik değer sayısını hesaplamak için kullanılır.

Out[15]:

KD	0
S	0
KG	0
KA	0
KM	0
TD	0
CS	0
HGY	0
DK	0
M	0
Y	0
KKK	0
MK	0
is	0
BKS	0
T	0
Yİ	0
siniflandirma	0

dtype: int64

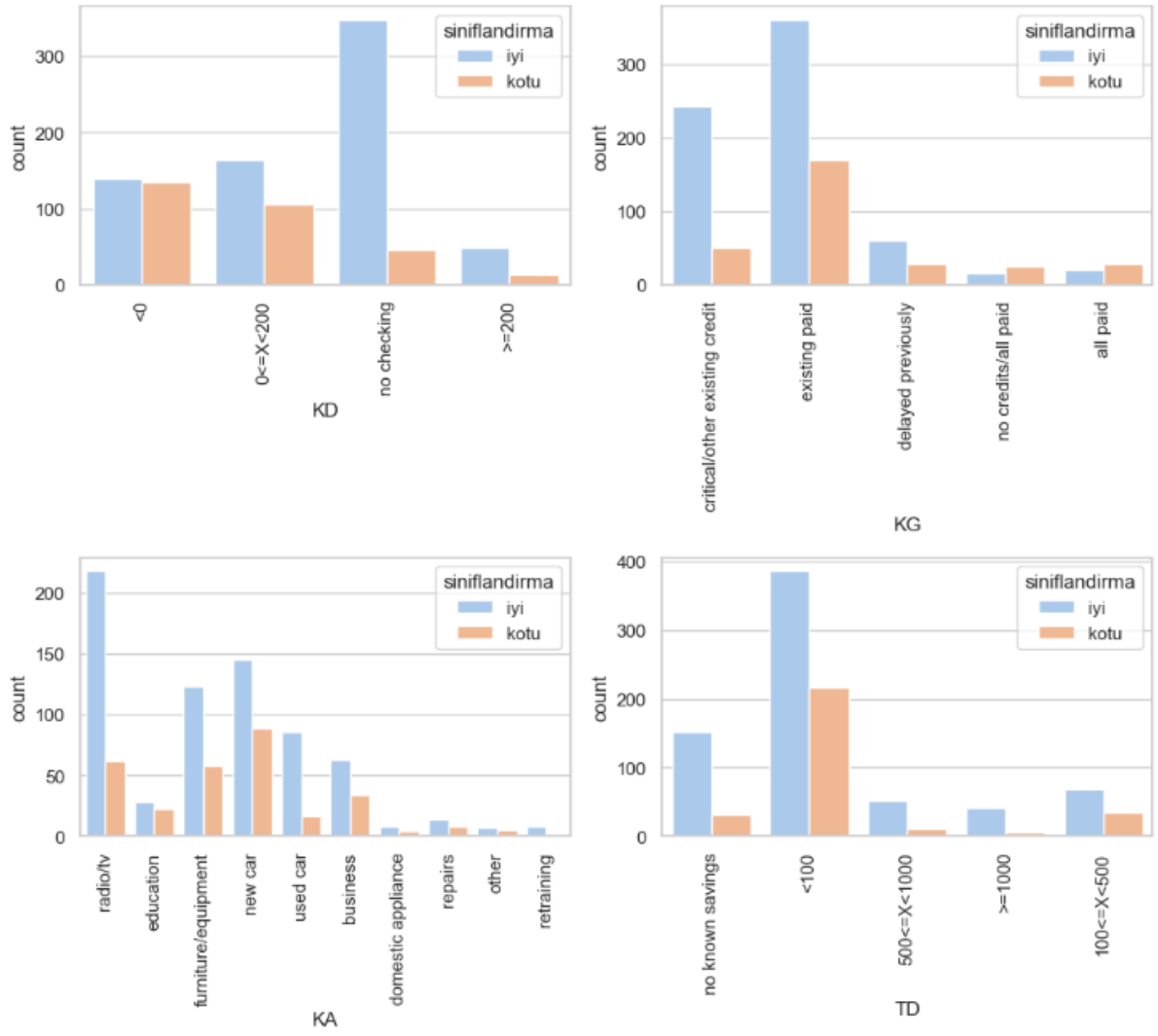
```
In[16]: for c in df.columns:
    print('Column {} unique values: {}'.format(c, len(df[c].unique())))
#Veri setindeki her bir sütun için benzersiz birbirinden farklı değer sayısını hesaplamak için kullanıldı

Column KD unique values: 4
Column S unique values: 33
Column KG unique values: 5
Column KA unique values: 10
Column KM unique values: 921
Column TD unique values: 5
Column CS unique values: 5
Column HGY unique values: 4
Column DK unique values: 3
Column M unique values: 4
Column Y unique values: 53
Column KKK unique values: 3
Column MK unique values: 4
Column is unique values: 4
Column BKS unique values: 2
Column T unique values: 2
Column Yİ unique values: 2
Column siniflandirma unique values: 2

In[17]: sns.set_theme(style="whitegrid", palette="pastel")
plt.figure(figsize = (10, 50))

kategorik_liste = df.select_dtypes(include='object').columns
x=0
for i in kategorik_liste:
    x+=1
    plt.subplot(11, 2, x)
```

plt.xticks(rotation=90) #x eksenindeki değerlerin 90 derece döndürülmesini sağlar, böylece sütun adları daha iyi okunabilir hale gelir.

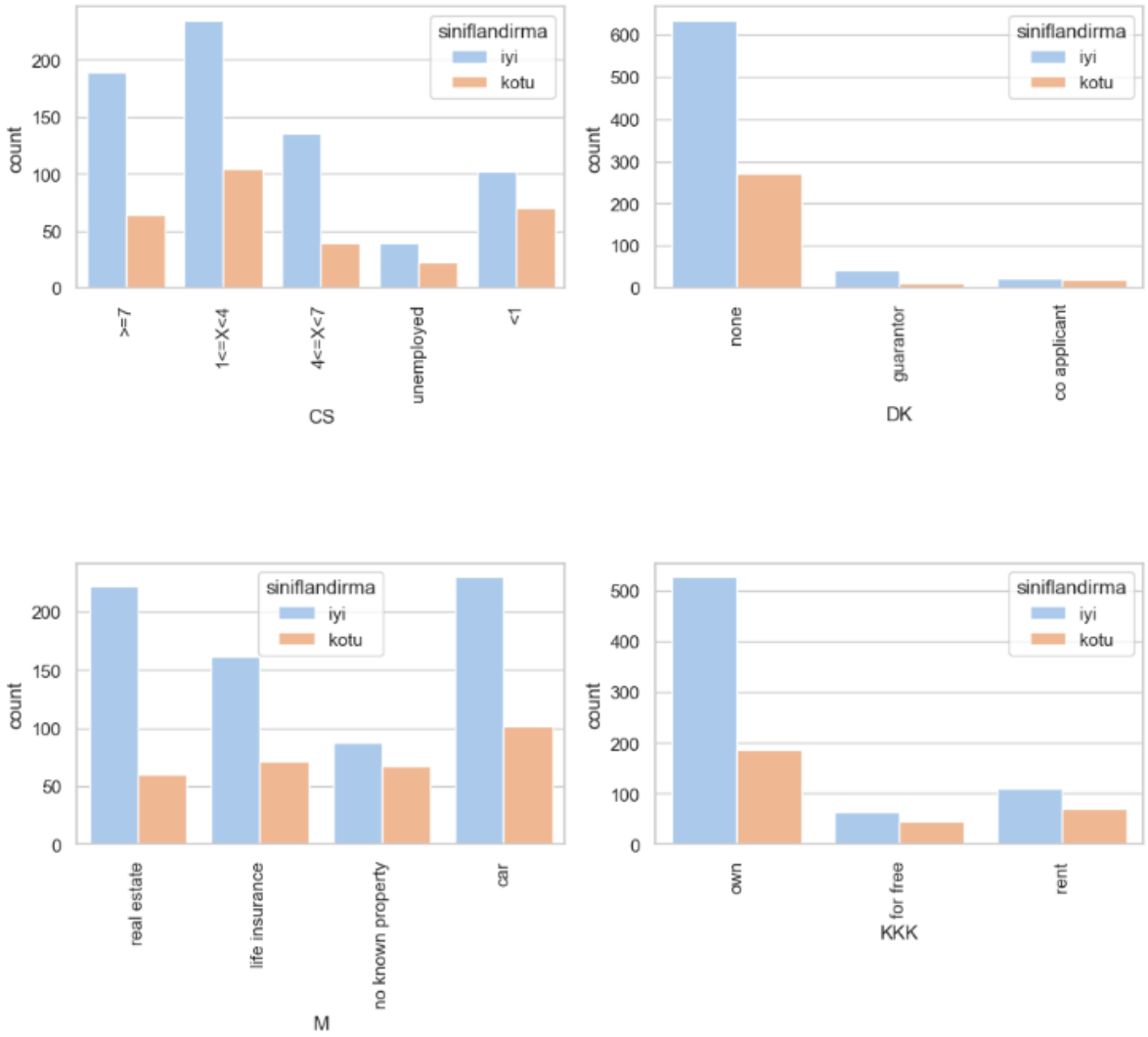
sns.countplot(x = i, hue = 'siniflandirma', data = df)

plt.tight_layout() #grafikler arasındaki boşlukları optimize etmek için kullanılır

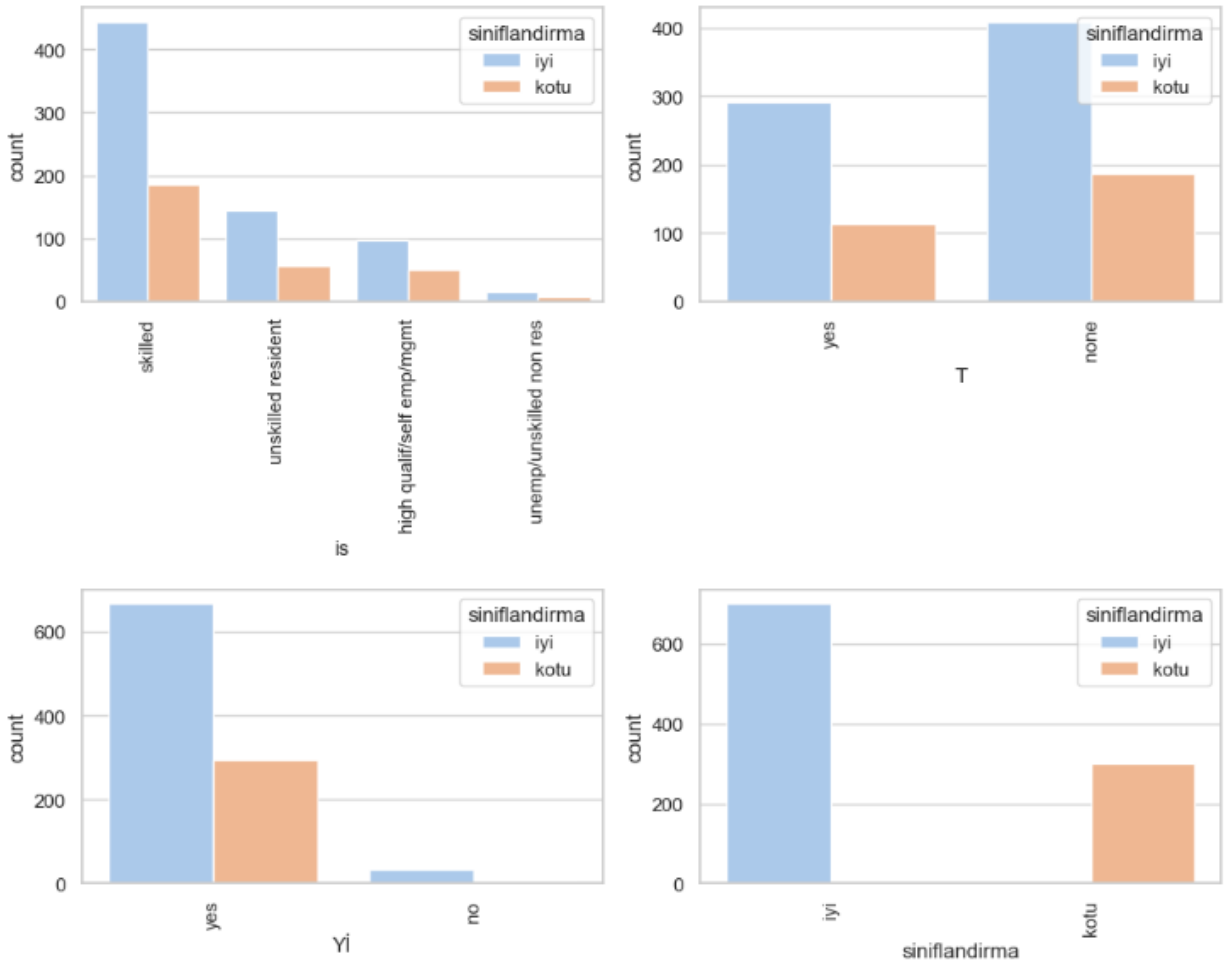
#Burda kategorik verilerimizin hedef (siniflandirma) ile arasındaki ilişkiyi gösteren grafik çizdirildi



Şekil 6. Kategorik Veriler ile hedef değişken sınıflandırma arasındaki ilişki grafiği



Şekil 7. Kategorik Veriler ile hedef değişken sınıflandırma arasındaki ilişki grafiği



Şekil 8. Kategorik Veriler ile hedef değişken sınıflandırma arasındaki ilişki grafiği

```
In[18]: kategorik = ['KD','KG','KA','TD',
                     'CS','DK',
                     'M','KKK','is','T',
                     'Yİ','siniflandirma']
#Kategorik verilerimizi sayısal verilere değiştirmek için bir listeye içine attık
In[19]: le = LabelEncoder()
for column in kategorik:
    df[column] = le.fit_transform(df[column])
#LabelEncoder ile kategorik verilerimizi transform sayısal verilere dönüştürüldü
In[20]: df.head() #Bu işlemlerin doğruluğunu görebilmek için veri setinin ilk 5 satırını tekrar çağırma işlemi yapıldı.
Out[20] :
```

Tablo 7. Kategorik Verilerin Sayısal Hale İşenmesinden Sonraki Veri Seti Çıktısı

	KD	S	KG	KA	KM	TD	CS	HGY	DK	M	Y	KKK	MK	is	BKS	T	Yİ	siniflandirma
0	1	6	1	6	1169	4	3	4	2	3	67	1	2	1	1	1	1	0
1	0	48	3	6	5951	2	0	2	2	3	22	1	1	1	1	0	1	1
2	3	12	1	2	2096	2	1	2	2	3	49	1	1	3	2	0	1	0
3	1	42	3	3	7882	2	1	2	1	1	45	0	1	1	2	0	1	0
4	1	24	2	4	4870	2	0	3	2	2	53	0	2	1	2	0	1	1

In[21]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 18 columns):

#	Column	Non-Null Count	Dtype
0	KD	1000 non-null	int32
1	S	1000 non-null	float64
2	KG	1000 non-null	int32
3	KA	1000 non-null	int32
4	KM	1000 non-null	float64
5	TD	1000 non-null	int32
6	CS	1000 non-null	int32
7	HGY	1000 non-null	float64
8	DK	1000 non-null	int32
9	M	1000 non-null	int32
10	Y	1000 non-null	float64
11	KKK	1000 non-null	int32
12	MK	1000 non-null	float64
13	is	1000 non-null	int32

14 BKS 1000 non-null float64

15 T 1000 non-null int32

16 Yİ 1000 non-null int32

17 siniflandirma 1000 non-null int32

dtypes: float64(6), int32(12)

memory usage: 93.9 KB

In[22]: df.corr() #veri ön işleme ve hazırlama süreçlerinden biri olan korelasyon analizi yapıldı

Out[22]:

Tablo 8. Korelasyon Analizi

	KD	S	KG	KA	KM	TD	CS	HGY	DK	M	Y	KKK	MK	is	BKS	T	Yİ	siniflandirma
KD	1,00	-0,10	-0,11	0,09	-0,09	0,10	-0,03	0,03	0,08	-0,02	0,08	- 0,01	0,08	-0,03	0,03	0,04	-0,01	-0,30
S	-0,10	1,00	0,03	-0,00	0,62	0,02	0,00	0,07	-0,01	-0,16	-0,04	- 0,16	-0,01	-0,22	-0,02	0,16	0,14	0,21
KG	-0,11	0,03	1,00	-0,02	0,03	-0,03	-0,09	-0,06	0,01	-0,04	-0,16	0,08	-0,39	-0,01	-0,07	-0,04	0,00	0,10
KA	0,09	-0,00	-0,02	1,00	0,05	0,09	0,08	0,02	-0,02	-0,03	0,03	- 0,06	-0,02	-0,07	0,03	0,01	0,01	-0,14
KM	-0,09	0,62	0,03	0,05	1,00	0,07	0,04	-0,27	-0,04	-0,14	0,03	- 0,14	0,02	-0,26	0,02	0,28	0,05	0,15
TD	0,10	0,02	-0,03	0,09	0,07	1,00	0,06	0,03	0,03	0,04	0,09	- 0,03	0,02	-0,04	0,02	0,08	-0,01	-0,10
CS	-0,03	0,00	-0,09	0,08	0,04	0,06	1,00	0,07	0,01	-0,02	0,29	- 0,13	0,09	-0,14	0,03	0,11	0,06	0,01
HGY	0,03	0,07	-0,06	0,02	-0,27	0,03	0,07	1,00	0,01	-0,02	0,06	- 0,09	0,02	-0,08	-0,07	0,01	0,09	0,07

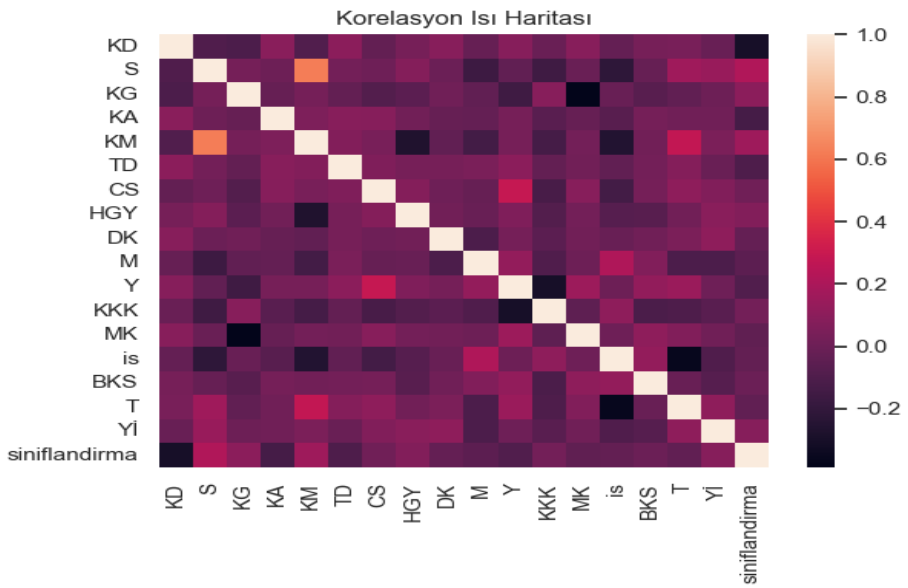
DK	0,08	-0,01	0,01	-0,02	-0,04	0,03	0,01	0,01	1,00	- 0,11	0,03	- 0,06	0,02	-0,01	0,01	0,05	0,11	-0,03
M	-0,02	-0,16	-0,04	-0,03	-0,14	0,04	-0,02	-0,02	-0,11	1,00	0,13	- 0,10	-0,00	0,21	0,07	-0,11	-0,11	-0,06
Y	0,08	-0,04	-0,16	0,03	0,03	0,09	0,29	0,06	0,03	0,13	1,00	- 0,30	0,15	-0,00	0,12	0,15	0,01	-0,09
KKK	-0,01	-0,16	0,08	-0,06	-0,14	-0,03	-0,13	-0,09	-0,06	-0,10	-0,30	1,00	-0,05	0,11	-0,11	-0,10	-0,06	0,02
MK	0,08	-0,01	-0,39	-0,02	0,02	0,02	0,09	0,02	0,02	-0,00	0,15	- 0,05	1,00	0,00	0,11	0,07	0,01	-0,05
is	-0,03	-0,22	-0,01	-0,07	-0,26	-0,04	-0,14	-0,08	-0,01	0,21	-0,00	0,11	0,00	1,00	0,13	-0,37	-0,10	-0,03
BKS	0,03	-0,02	-0,07	0,03	0,02	0,02	0,03	-0,07	0,01	0,07	0,12	- 0,11	0,11	0,13	1,00	-0,01	-0,08	-0,00
T	0,04	0,16	-0,04	0,01	0,28	0,08	0,11	0,01	0,05	- 0,11	0,15	- 0,10	0,07	-0,37	-0,01	1,00	0,11	-0,04
Yİ	-0,01	0,14	0,00	0,01	0,05	-0,01	0,06	0,09	0,11	- 0,11	0,01	- 0,06	0,01	-0,10	-0,08	0,11	1,00	0,08
siniflandirma	-0,30	0,21	0,10	-0,14	0,15	-0,10	0,01	0,07	-0,03	-0,06	-0,09	0,02	-0,05	-0,03	-0,00	-0,04	0,08	1,00

In[23]: sns.heatmap(df.corr())

plt.title('Korelasyon Isı Haritası')

#Verilerin arasındaki ilişkiyi görselleştirmek için ısı haritası kullanıldı

Out[23]: Text(0.5, 1.0, 'Korelasyon Isı Haritası')



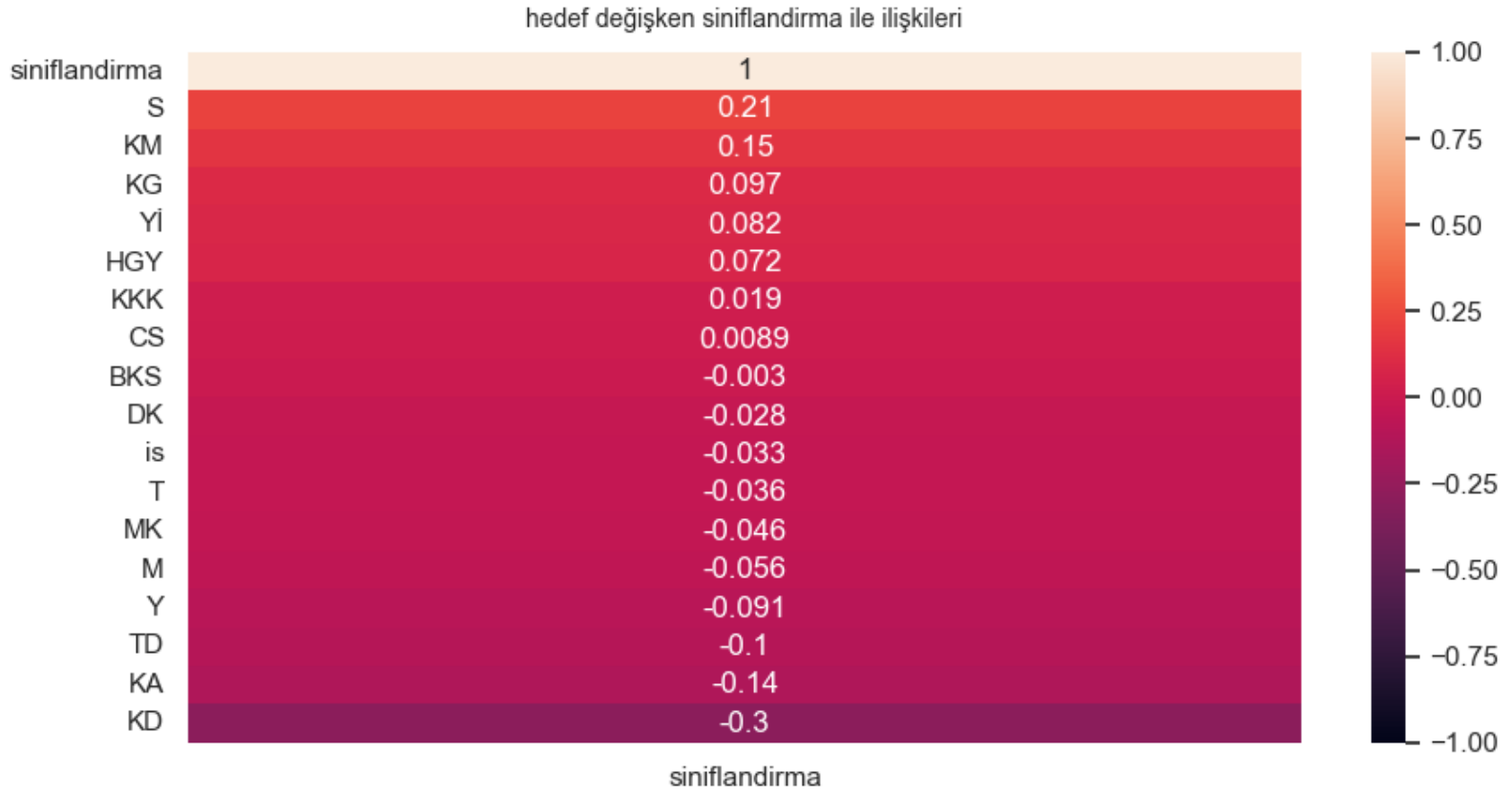
Şekil 9. Korelasyon Isı Haritası

In[24]: plt.figure (figsize = (10, 5))

heatmap = sns.heatmap (df.corr()[['siniflandirma']].sort_values (by = 'siniflandirma', ascending = False), vmin = -1, vmax = 1, annot = True)

heatmap.set_title ('hedef değişken siniflandirma ile ilişkileri', fontdict = {'fontsize':10}, pad = 10);

#hedef değişken (sınıflandırma) ile diğer değişkenler arasındaki korelasyonları görselleştirmek için kullanıldı



Şekil 10. Hedef değişken Sınıflandırma ile İlişkileri

In[25]: df.corr().nlargest(4,'siniflandirma').index
#siniflandirma değişkenine göre en yüksek korelasyona sahip olan dört değişkenin indekslerini döndürür.Yani sınıflandırma değişkeni ile en güçlü korelasyona sahip olan dört değişkenin indekslerini ver.

Out[25]: Index(['siniflandirma', 'S', 'KM', 'KG'], dtype='object')

4.1. Lojistik Regresyon

In[26]: from sklearn import linear_model #Sklearn kütüphanesinin linear_model modülü eklendi
from sklearn.model_selection import cross_val_score
log_reg = linear_model.LogisticRegression() #Lojistik model nesnesi oluşturuldu

In[27]: x=df[['S', 'KM', 'KG']]
y=df.iloc[:,17] #18.Sütunu yani hedef sütunumuz olan siniflandirmayı temsil eder

In[28]: x
#x değişkenine 'sure', 'kredi_miktari' ve 'kredi_gecmisi' sütunlarını attık

Tablo 9. X Değişkenine Bağlı Süre(S) Kredi Miktarı(KM) ve Kredi Geçmişi(KG)

	S	KM	KG
0	6	1169	1
1	48	5951	3
2	12	2096	1
3	42	7882	3
4	24	4870	2
...	...	...	...
995	12	1736	3
996	30	3857	3
997	12	804	3
998	45	1845	3
999	45	4576	1

In[29]: y
#y değişkenine veri çerçevesinin 20. sütununu atıyoruz. Bu sütun hedef değişkenimizi olan 'siniflandirma'yı temsil ediyor.

Out[29]:

Tablo 10. Veri Çerçevesinin Sınıflandırma Sütunu Olan Hedef Değişken

0	0
1	1
2	0
3	0
4	1
..	

995	0
996	0
997	0
998	1
999	0

Name: siniflandirma, Length: 1000, dtype: int32

In[30]: log_reg_score = cross_val_score(log_reg,x,y,cv =10,scoring='accuracy').mean()

#Burada modeli eğitip verinin kaç parçaya bölünüp bir öğrenme gerçekleştileceğini söylüyoruz.

Parametrelerin anlamları:

1. log_reg: Lojistik regresyon modeli.
2. x: Bağımsız değişkenlerin olduğu veri kümesi.
3. y: Hedef değişkenin olduğu veri kümesi.
4. cv: Katlama sayısı, yani veri kümesinin kaç parçaya bölüneceği.
5. scoring: Performans ölçütü, burada doğruluk (accuracy) kullanılmıştır.
6. Sonuç olarak, log_reg_score değişkenine lojistik regresyon modelinin çapraz doğrulama ile elde edilen ortalama doğruluk skoru atanır. Bu skor, modelin sınıflandırma performansını değerlendirmek için kullanılabilir.

```
In[31]: log_reg_score

Out[31]: 0.707

In[32]: log_reg.fit(x,y)

Out[32]:LogisticRegression
        LogisticRegression()


In[33]: sure = 45
kredi_miktari = 1169
kredi_gecmisi = 5
prediction = log_reg.predict([[sure,kredi_miktari,kredi_gecmisi]])
C:\Users\Mert\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature
names, but LogisticRegression was fitted with feature names
  warnings.warn(

In[34]: Prediction

Out[34]: array([1])
```

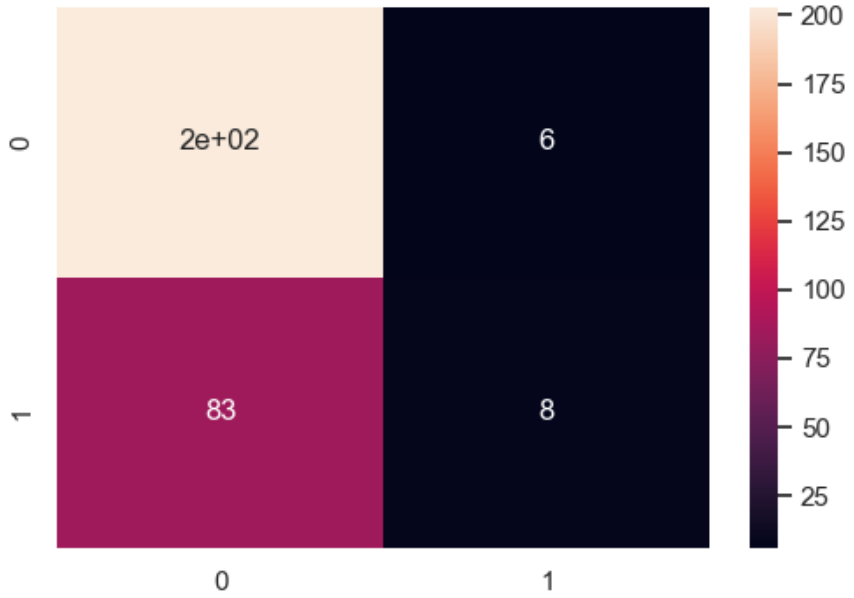
In[35]: X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42) #Bu satır veri kümemizin eğitim ve test kümelerini ayırır x_train ve x_test sure,kredi_miktari_kredi_gecmisi eğitim ve test verilerini içeren veri kümesidir.y_train ve y_test bağımlı değişken siniflandirmadir.

#Tekrar modeli çağırmamıza gerek yok çünkü yukarıda çağırmıştık log_reg = linear_model.LogisticRegression()
log_reg.fit(X_train, y_train) # Bu satırda, oluşturulan lojistik regresyon modeli X_train ve y_train verileri kullanılarak eğitilir.
y_pred = log_reg.predict(X_test) # Bu satırda, eğitilen model X_test verilerini kullanarak tahmin yapar ve tahmin sonuçlarını y_pred değişkenine kaydeder.
report = classification_report(y_test, y_pred) #bu ve
print(report)# Bu satırlarda, classification_report fonksiyonu kullanılarak sınıflandırma raporu oluşturulur ve ekrana yazdırılır.
cf_matrix = confusion_matrix(y_test, y_pred) # Bu satırlarda, karışıklık matrisi (confusion_matrix) hesaplanır ve seaborn kütüphanesi kullanılarak bir ısı haritası olarak görselleştirilir. #y_test gerçek hedef değerlerini içeren test veri setidir. Bu veri seti, modelin tahminlerini karşılaştırmak için kullanılır.

#y_pred ise modelin test veri seti üzerinde yaptığı tahminlerdir. Model, test veri setindeki örneklerin sınıf etiketlerini tahmin eder ve bu tahminler y_pred değişkeninde tutulur.
plt.figure(figsize=(6, 4))
sns.heatmap(cf_matrix, annot=True)
plt.show()

Tablo 11. Lojistik Regresyon Skor Tablosu

	precision	recall	f1-score	support
0	0.71	0.97	0.82	209
1	0.57	0.09	0.15	91
accuracy			0.70	300
macro avg	0.64	0.53	0.49	300
weighted avg	0.67	0.70	0.62	300



Şekil 11. Lojistik Regresyonun karışıklık Matrisi

4.2. Naive Bayes

Algoritma bir eleman için her durumun olasılığını hesaplar ve olasılık değeri en yüksek olana göre sınıflandırır 3 gruba ayrılır. GaussianNB MultinominalNB BernoulliNB Biz sadece Gaussianı kullandık

In[36]: naive_bayes = GaussianNB() #Nesnemizi oluşturduk

naive_bayes.fit(X_train, y_train) #Modeli eğitiyoruz

naive_bayes_score = cross_val_score(naive_bayes, X_train, y_train, cv=10, scoring='accuracy').mean()

In[37]: naive_bayes_score

Out[37]: 0.7214285714285713

In[38]: y_pred = naive_bayes.predict(X_test) # test veri seti x_test üzerindeki verileri kullanarak modelin tahminlerini yapıyoruz.

print(classification_report(y_test, y_pred)) # Sınıflandırma raporunu (classification report) yazdırıyoruz. Bu rapor, modelin sınıflandırma performansını farklı metriklerle değerlendirir: kesinlik (precision), duyarlılık (recall), F1 skoru vb.

nb_accuracy=accuracy_score(y_test, y_pred) # Tahminlerin doğruluk oranını hesaplıyoruz.

nb_f1=f1_score(y_test, y_pred) # Tahminlerin F1 skorunu hesaplıyoruz.

nb_recall=recall_score(y_test, y_pred) # Tahminlerin duyarlılık (recall) skorunu hesaplıyoruz.

nb_precision=precision_score(y_test, y_pred) # Tahminlerin kesinlik (precision) skorunu hesaplıyoruz.

cf_matrix=confusion_matrix(y_test, y_pred) # Karmaşıklık matrisini hesaplıyoruz. Bu matris, modelin doğru ve yanlış sınıflandırmalarını gösterir.

plt.figure(figsize = (6,4))

sns.heatmap(cf_matrix, annot=True)#Karmaşıklık matrisini ısı haritası olarak görselleştiriyoruz.

#Precision (Kesinlik): Sınıf 0 için doğru tahmin oranı %71, sınıf 1 için ise %46'dır. Yani, model sınıf 0'ı tahmin ederken %71 oranında doğru, sınıf 1'i tahmin ederken ise %46 oranında doğru tahmin yapmıştır.

#Recall (Duyarlılık): Sınıf 0 için doğru sınıf tespit oranı %93, sınıf 1 için ise %14'tür. Model, sınıf 0'ı daha iyi tespit ederken sınıf 1'i daha düşük bir başarıyla tespit etmiştir.

#F1-score: Sınıf 0 için F1-score değeri 0.81, sınıf 1 için ise 0.22'dir. F1-score, kesinlik ve duyarlılık değerlerini dengeli bir şekilde birleştirir. Sınıf 0 için iyi bir F1-score elde edilirken, sınıf 1 için düşük bir F1-score değeri vardır.

#Support: Sınıf 0 için 209 örnek, sınıf 1 için 91 örnek bulunmaktadır. Bu, her sınıfın gerçek veri setinde kaç örneğe sahip olduğunu gösterir.

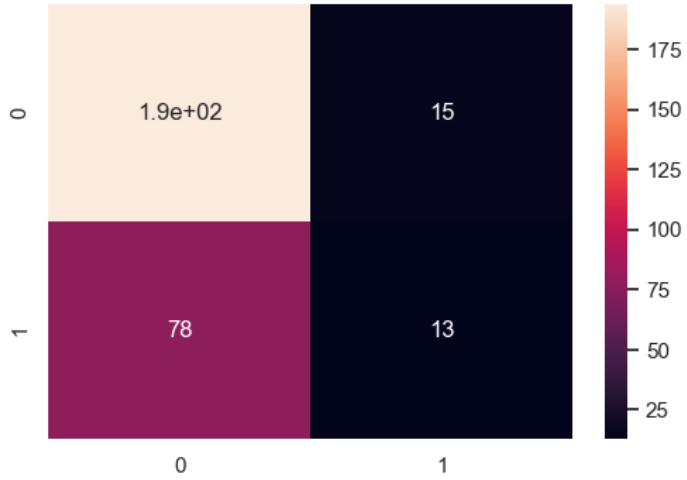
#Accuracy (Doğruluk): Modelin doğru sınıflandırma oranı %69'dur. Yani, veri noktalarının %69'u doğru bir şekilde sınıflandırılmıştır.

#Macro Avg ve Weighted Avg: Bu iki metrik, sınıfların ortalama performansını ölçer. Macro avg, tüm sınıfların performansını eşit olarak değerlendirirken, weighted avg sınıfların örnek sayılarına göre değerlendirilmektedir.

Tablo 12. Naive Bayes Skor Tablosu

	precision	recall	f1-score	support
0	0.71	0.93	0.81	209
1	0.46	0.14	0.22	91
accuracy			0.69	300
macro avg	0.59	0.54	0.51	300
weighted avg	0.64	0.69	0.63	300

Out[38]:
<Axes: >



Şekil 12. Naive Bayes Karışıklık Matrisi

In[39]: naive_bayes.fit(x, y)

Out[39]: GaussianNB

GaussianNB()

In[40]: sure = 45

kredi_miktari = 1169

kredi_gecmisi = 5

prediction1 = naive_bayes.predict([[sure, kredi_miktari, kredi_gecmisi]]

C:\Users\Mert\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but GaussianNB was fitted with feature names warnings.warn(

In[41]: prediction1

Out[41]: array([1])

4.3. Karar Ağacı Algoritması

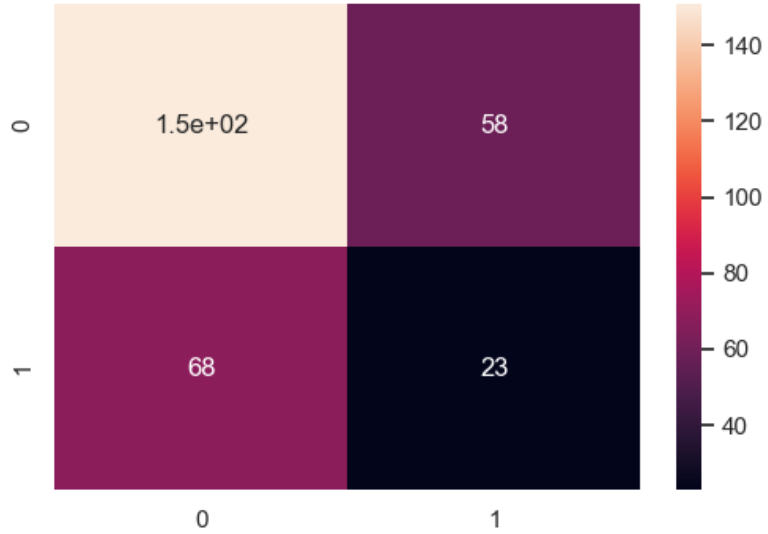
Sınıflama , özellik ve hedefe göre karar düğümleri ve yaprak düğümlerinden oluşan ağaç yapısı formunda model oluşturan bir sınıflandırma yöntemidir.

In[42]: X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42) *# Veri kümesini eğitim ve test kümelerine ayırma*

decision_tree = DecisionTreeClassifier() # Karar ağacı modeli oluşturma
decision_tree.fit(X_train, y_train) # Modeli eğitme
y_pred = decision_tree.predict(X_test) # Test verileri üzerinde tahmin yapma
report = classification_report(y_test, y_pred) # Sınıflandırma raporunu oluşturma ve yazdırma
print(report)
cf_matrix = confusion_matrix(y_test, y_pred) # Karışıklık matrisini oluşturma
plt.figure(figsize=(6, 4)) # Karışıklık matrisini ısı haritası olarak görselleştirme
sns.heatmap(cf_matrix, annot=True)
plt.show()

Tablo 13. Karar Ağacı Algoritması Skor Tablosu

	precision	recall	f1-score	support
0	0.69	0.72	0.71	209
1	0.28	0.25	0.27	91
accuracy			0.58	300
macro avg	0.49	0.49	0.49	300
weighted avg	0.57	0.58	0.57	300



Şekil 13. Karar Ağacı Algoritması Karışıklık Matrisi

4.4. KNN En Yakın Komşu Algoritması

In[43]: X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

knn = KNeighborsClassifier()

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

report = classification_report(y_test, y_pred)

print(report)

cf_matrix = confusion_matrix(y_test, y_pred)

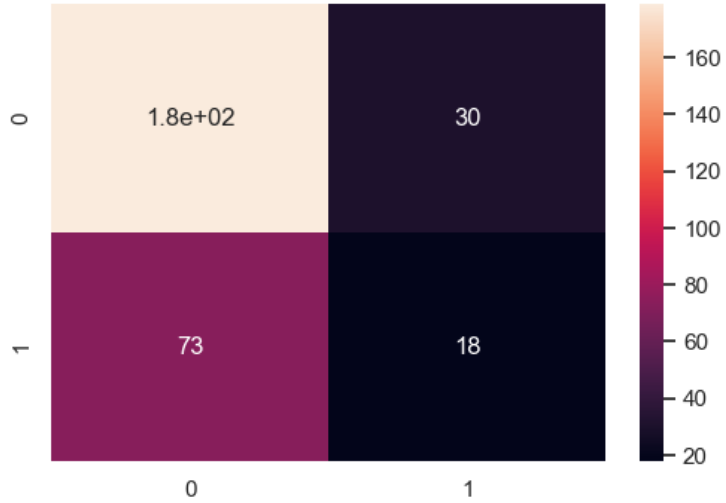
plt.figure(figsize=(6, 4))

sns.heatmap(cf_matrix, annot=True)

plt.show()

Tablo 14. KNN En Yakın Komşu Algoritması Skor Tablosu

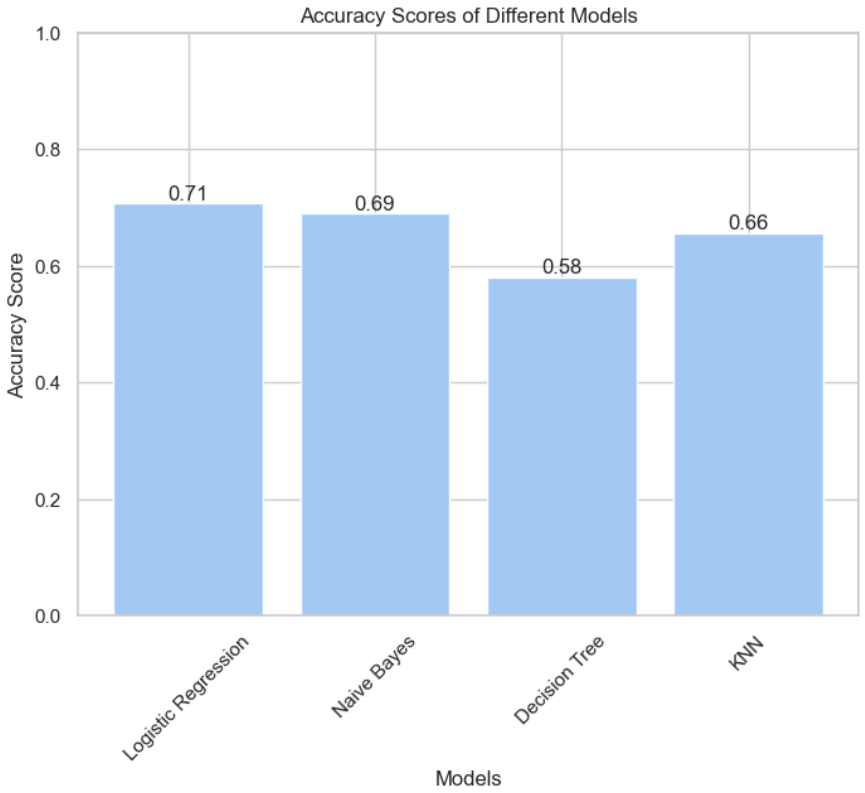
	precision	recall	f1-score	support
0	0.71	0.86	0.78	209
1	0.38	0.20	0.26	91
accuracy			0.66	300
macro avg	0.54	0.53	0.52	300
weighted avg	0.61	0.66	0.62	300



Şekil 14. KNN En Yakın Komşu Algoritması Karışıklık Matrisi

```
In[44]: model_names = ['Logistic Regression', 'Naive Bayes', 'Decision Tree', 'KNN']

scores = [log_reg_score, nb_accuracy, decision_tree.score(X_test, y_test), knn.score(X_test, y_test)]


plt.figure(figsize=(8, 6))
plt.bar(model_names, scores)
plt.xlabel('Models')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores of Different Models')
plt.ylim(0, 1)
```

```
plt.xticks(rotation=45)
for i in range(len(model_names)):
    plt.text(i, scores[i], f'{scores[i]:.2f}', ha='center', va='bottom')
plt.show()
```



Şekil 15. Modellerin Doğruluk Skorları

Sonuç olarak Logistic Regression seçilmelidir.

BÖLÜM 5. İSTATİKSEL ANALİZ

df.describe() #sayısal özniteliklerimizin istatiksel özelliklerine bakıldı

#kredi ödeme süresi ortalama 20.903000 min 4 max 72 ay std sapması ise 12.05

#kredi miktarı ortalama 3271.258 min 250 max 18424 std sapması ise 2822.73

Tablo 15. Veri Setimizdeki Sayısal özniteliklerimizin İstatistiksel Özellikleri

	S	KM	HGY	Y	MK	BKS
count	1000	1000	1000	1000	1000	1000
mean	20,903	3271,258	2,973	35,546	1,407	1,155
std	12,05881	2822,737	1,118715	11,37547	0,577654	0,362086
min	4	250	1	19	1	1
25%	12	1365,5	2	27	1	1
50%	18	2319,5	3	33	1	1
75%	24	3972,25	4	42	2	1
max	72	18424	4	75	4	2

S: Bu sütunun istatistiksel özellikleri şu şekildedir:

count: Bu sütunda 1000 adet veri bulunmaktadır.

mean: Bu sütundaki verilerin ortalaması 20.903'tür.

std: Bu sütundaki verilerin standart sapması 12.058814'tür.

min: Bu sütundaki en küçük değer 4'tür.

25%: Bu sütundaki verilerin ilk çeyreği (alt sınır) 12'dir.

50%: Bu sütundaki verilerin medyan değeri (ortanca) 18'dir.

75%: Bu sütundaki verilerin üçüncü çeyreği (üst sınır) 24'tür.

max: Bu sütundaki en büyük değer 72'dir.

KM: Bu sütunun istatistiksel özellikleri:

count: 1000 veri bulunmaktadır.

mean: Ortalama 3271.258'dir.

std: Standart sapma 2822.736876'dır.

min: En küçük değer 250'dir.

25%: İlk çeyrek 1365.5'tir.

50%: Medyan değer 2319.5'tir.

75%: Üçüncü çeyrek 3972.25'tir.

max: En büyük değer 18424'tür.

HGY: Bu sütunun istatistiksel özellikleri:

count: 1000 veri bulunmaktadır.

mean: Ortalama 2.973'tür.

std: Standart sapma 1.118715'tir.

min: En küçük değer 1'dir.

25%: İlk çeyrek 2'dir.

50%: Medyan değer 3'tür.

75%: Üçüncü çeyrek 4'tür.

max: En büyük değer 4'tür.

Y: Bu sütunun istatistiksel özellikleri:

count: 1000 veri bulunmaktadır.

mean: Ortalama 35.546'dır.

std: Standart sapma 11.375469'dır.

min: En küçük değer 19'dur.

25%: İlk çeyrek 27'dir.

50%: Medyan değer 33'tür.

75%: Üçüncü çeyrek 42'dir.

max: En büyük değer 75'tir.

MK: Bu sütunun istatistiksel özellikleri:

count: 1000 veri bulunmaktadır.

mean: Ortalama 1.407'dir.

std: Standart sapma 0.577654'dir.

min: En küçük değer 1'dir.

25%: İlk çeyrek 1'dir.

50%: Medyan değer 1'dir.

75%: Üçüncü çeyrek 2'dir.

max: En büyük değer 4'tür.

BKS: Bu sütunun istatistiksel özellikleri:

count: 1000 veri bulunmaktadır.

mean: Ortalama 1.155'tir.

std: Standart sapma 0.362086'dır.

min: En küçük değer 1'dir.

25%: İlk çeyrek 1'dir.

50%: Medyan değer 1'dir.

75%: Üçüncü çeyrek 1'dir.

max: En büyük değer 2'dir.

BÖLÜM 6. GERÇEKCİ KISITLAR VE KOŞULLAR ALTINDA DEĞERLENDİRME

6.1. Ekonomik Açıdan Değerlendirme

Kredi riski analizi için ekonomik göstergelerin kullanımı oldukça önemlidir. İşsizlik oranı, faiz oranları, enflasyon gibi faktörler kredi riski tahmininde etkili olabilir.

6.2. Çevre Sorunları Açısından Değerlendirme

Çevresel faktörlerin kredi riski üzerindeki etkisi giderek daha fazla önem kazanmaktadır. Sürdürülebilirlikle ilgili faktörler, çevresel riskler ve şirketlerin çevresel performansı değerlendirilebilir.

6.3. Sürdürülebilirlik Açıdan Değerlendirme

Sürdürülebilirlik kriterleri, finansal kurumların risk değerlendirmelerinde dikkate alınabilir. ESG (Çevresel, Sosyal ve Kurumsal Yönetişim) faktörlerinin kredi riski tahminine dahil edilmesi önemlidir. Üretilebilirlik: Üretim verimliliği ve sektörel faktörler, kredi riski analizinde dikkate alınabilir. Şirketlerin gelirleri, karlılık oranları ve sektörel performansı değerlendirilebilir.

6.4. Etik Açısından Değerlendirme

Kredi riski analizi yaparken adaletli ve ayrımcılık yapmayan bir yaklaşım benimsenmelidir. Etnik köken, cinsiyet, yaş gibi kişisel faktörlerin ayrımcılığa neden olmadığından emin olunmalıdır.

6.5. Sağlık Açısından Değerlendirme

Kişilerin sağlık durumu ve sağlık harcamaları, kredi riski analizinde değerlendirilebilir. Kronik hastalıklar, sağlık sigortası kapsamı gibi faktörler kredi riskini etkileyebilir.

6.6. Güvenlik Açısından Değerlendirme

Kimlik hırsızlığı, sahtekarlık ve dolandırıcılık gibi güvenlik sorunları kredi riskiyle ilişkilendirilebilir. Kredi başvurularında güvenlik önlemleri ve kimlik doğrulama süreçleri önemlidir.

BÖLÜM 7. SONUÇ

Kredi riski taşıyan kişileri çeşitli analizler kullanarak önceden tahmin etmek ve bu sayede insanların kredi riski taşıyıp taşımadığını borç verilip verilmeyeceğini anlamak mümkündür. Bu çalışmada Logistic Regression , Naive Bayes , Decision Tree ve KNN algoritmaları kullanılmıştır. Bu algoritmalar sonucunda KNN %66 , Naive Bayes %69, Decision Tree %58, Logistic Regression %71 doğruluk skoru bulunmuştur . Analiz ettiğimiz veri tabanı için seçilen en uygun algoritma Logistic Regressiondur. Lojistik regresyon analizi sonucunda %71 başarılı tahmin skoru elde edildi. Kredi riskine sebep olan temel faktör borç ödeme süresidir. Borç ödeme süresini Kredi Miktarı ve Kredi Geçmişi takip etmektedir. Logistic Regression algoritması genellikle sınıflandırma problemlerinde kullanılan etkili bir yöntemdir ve bu durumda da uygun bir seçim olmuş gibi görünmektedir. Bu tahmine dayalı sonuçlar neticesinde zaman kaybının önüne geçilebilir 1000 verilik dataya 3000+ kadar daha fazla insanın verileri eklenerek modellerin doğruluk skorlarını daha fazla arttırılabilmektedir. Bu sayede hedef değişkenimiz olan sınıflandırmayla ilişkili olan değişkenleri de daha iyi yorumlayabilmekteyiz.

KAYNAKÇA

Akçay, A. (2014). Bilgi ve Belge Yönetiminde veri madenciliği.

Aydın, S. (2015). Veri Madenciliği Ve Anadolu Üniversitesi Uzaktan Eğitim Sisteminde Bir Uygulama.

Emre, İ., & Erol, Ç. (2017). Veri Analizinde İstatistik mi Veri Madenciliği mi ? Bilişim Teknolojileri Dergisi.

Hastie, T. T. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

James, G. W. (2013). Introduction to Statistical Learning. Springer.

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

Özbay, Ö. (2015). Veri Madenciliği Kavramı ve Eğitimde Veri Madenciliği Uygulamaları. Uluslararası Eğitim Bilimleri Dergisi.

Pedregosa, F. V. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research.

Raschka, S. &. (2017). Python Machine Learning. Packt Publishing.

Savaş, S., Topaloğlu, N., & Yılmaz, M. (2012). Veri Madenciliği ile Türkiye'deki Uygulama Örnekleri.

Taşçı, M., & Şamlı, R. (2020). Veri Madenciliği İle Kalp Hastalığı Teşhisi. Avrupa Bilim ve Teknoloji Dergisi.

ÖZGEÇMİŞ

Ben Özen,2001 yılında Samsun'da doğdum. İlkokul, ortaokul ve lise eğitimimi Bafra'da tamamladım. Sakarya üniversitesi Endüstri Mühendisliği 4.sınıf öğrencisi olarak eğitimime devam etmekteyim. Anadolu Üniversitesinde ikinci üniversite olarak Yönetim Bilişimi Sistemleri okumaktayım.

Ben Mert,1998 yılında Çanakkale'de doğdum. İlkokul, ortaokul ve lise eğitimimi Gelibolu'da tamamladım. Ege Üniversitesinde Bilgisayar Programcılığı bölümünden mezun oldum. DGS ile kazandığım Sakarya Endüstri Mühendisliğini 4.sınıf öğrencisi olarak devam etmekteyim.