● Exploratory Data Analysis

1. Keşifsel Veri Analizi Nedir?

Keşifsel Veri Analizi (Exploratory Data Analysis - EDA), veriyi sistematik bir şekilde incelemek için **görselleştirme**, **dönüşüm** ve **modelleme** araçlarını kullanarak veri hakkında sorular oluşturma ve yanıtlar arama sürecidir.

EDA'nın Temel Adımları:

1. Verinizle ilgili **sorular** oluşturun.
2. Bu soruları yanıtlamak için **görselleştirme**, **dönüşüm** ve **modelleme** yöntemlerini kullanın.
3. Öğrendiklerinizi yeni sorular oluşturmak veya mevcut soruları detaylandırmak için kullanın.

EDA, belirli kuralları olmayan **tekrarlamalı** (iteratif) bir süreçtir. EDA'nın temelinde **veriyi anlamak** yatar.

---

2. EDA Sürecinde Sorular Oluşturmak

EDA sürecinde, verinizi anlamak için sorular sorarsınız. İki temel soru türü her zaman faydalıdır:

1. **Değişkenler içindeki varyasyon nedir?**
2. **Değişkenler arasındaki kovaryasyon nedir?**

Bu sorular, verinizi keşfederken **hangi grafik**, **hangi dönüşüm** ya da **hangi modeli** kullanmanız gerektiğini belirler.

---

3. Varyasyon

Varyasyon, bir değişkenin değerlerinin farklı ölçümler arasında değişme eğilimidir. Varyasyonu anlamanın en iyi yolu **görselleştirme** kullanarak değişkenin dağılımını incelemektir.

3.1 Dağılımların Görselleştirilmesi

- **Kategorik Değişkenler:** Küçük bir değer kümesi alabilen değişkenlerdir. Bunları **çubuk grafikleri (bar chart)** ile inceleriz.

Örnek Kod:

```
library(tidyverse)

ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```

- **Sürekli Değişkenler:** Sonsuz sayıda sıralı değer alabilen değişkenlerdir. Bunları **histogram** kullanarak inceleriz.

Örnek Kod:

```
ggplot(data = diamonds) +
  geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```

Histogram Örneği:
Binwidth parametresi, histogramdaki aralık genişliğini belirler. Farklı **binwidth** değerleri, farklı dağılım paternleri gösterebilir.

```
smaller <- diamonds %>% filter(carat < 3)

ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.1)
```

3.2 Tipik Değerler

- Grafikteki **yüksek çubuklar**, değişkenin yaygın değerlerini gösterir.
- Soru: **Hangi değerler yaygındır? Hangi değerler nadirdir?**

Örnek Kod:

```
ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.01)
```

3.3 Sıra Dışı Değerler (Outliers)

Outlier'lar, **veri setindeki beklenmeyen veya alışılmadık değerlerdir**.

Örnek Kod:

```
ggplot(diamonds) +
```

```
  geom_histogram(mapping = aes(x = y), binwidth = 0.5) +
  coord_cartesian(ylim = c(0, 50))
```

Outlier'ların Belirlenmesi ve Düzenlenmesi:

```
unusual <- diamonds %>%
  filter(y < 3 | y > 20) %>%
  select(price, x, y, z) %>%
  arrange(y)
```

---

4. Eksik Değerler

Veri setindeki eksik değerlerle iki şekilde başa çıkabilirsiniz:

1. **Eksik veriyi silmek:**

```
diamonds2 <- diamonds %>%
  filter(between(y, 3, 20))
```

2. **Eksik veriyi NA ile değiştirmek:**

```
diamonds2 <- diamonds %>%
  mutate(y = ifelse(y < 3 | y > 20, NA, y))
```

NA Değerlerinin Görselleştirilmesi:

```
ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +
  geom_point(na.rm = TRUE)
```

---

5. Kovaryasyon

Kovaryasyon, iki veya daha fazla değişkenin birlikte nasıl değiştiğini açıklar.

5.1 Kategorik ve Sürekli Değişkenler

Bir sürekli değişkenin kategorik değişkenlere göre dağılımını **boxplot** kullanarak görselleştiririz.

Örnek Kod:

```
ggplot(data = diamonds, mapping = aes(x = cut, y = price)) +
  geom_boxplot()
```

Kategorik Değişkenleri Yeniden Sıralamak:

```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = reorder(class, hwy, FUN =
median), y = hwy)) +
  coord_flip()
```

5.2 İki Kategorik Değişken

İki kategorik değişkenin ilişkisini **geom_tile** veya **geom_count** kullanarak görselleştiririz.

```
diamonds %>%
  count(color, cut) %>%
  ggplot(mapping = aes(x = color, y = cut)) +
    geom_tile(mapping = aes(fill = n))
```

5.3 İki Sürekli Değişken

İki sürekli değişkenin ilişkisini **scatterplot** ile görselleştiririz.

Örnek Kod:

```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price), alpha = 1 /
100)
```

Büyük veri setlerinde binleme (binning) kullanmak:

```
ggplot(data = smaller) +
  geom_hex(mapping = aes(x = carat, y = price))
```

---

6. Desenler ve Modeller

Verinizdeki desenler, değişkenler arasındaki ilişkileri ve kovaryasyonu gösterir. Bu desenler, veriniz hakkında tahmin yapmanızı sağlar.

Örneğin, **modelleme** ile bir değişkenin etkisini çıkararak başka ilişkileri inceleyebilirsiniz.

Örnek Kod:

```
mod <- lm(log(price) ~ log(carat), data = diamonds)

diamonds2 <- diamonds %>%
  add_residuals(mod) %>%
  mutate(resid = exp(resid))

ggplot(data = diamonds2) +
  geom_boxplot(mapping = aes(x = cut, y = resid))
```

---

7. ggplot2 Çağrıları

Gelecekte, ggplot2 kodlarını daha kısa yazmak için sadece gerekli argümanları kullanabiliriz.

```
diamonds %>%
  count(cut, clarity) %>%
  ggplot(aes(clarity, cut, fill = n)) +
    geom_tile()
```

• Data Visualisation

2.1 Giriş

“Basit bir grafik, veri analizcisinin zihnine diğer tüm araçlardan daha fazla bilgi getirir.” — John Tukey

Bu bölümde **ggplot2** kullanarak verilerinizi nasıl görselleştireceğinizi öğreneceksiniz. **ggplot2**, "Grafiklerin Grameri" (Grammar of Graphics) prensibine dayalı olarak çalışır. Bu sistem sayesinde, bir kez öğrendiğiniz temel yapı ile birçok farklı grafiği hızlıca oluşturabilirsiniz.

Gereksinimler

Bu bölümde, **tidyverse** paketindeki **ggplot2** kütüphanesini kullanacağız. Başlamadan önce tidyverse'ü yükleyin ve çağırın:

```
install.packages("tidyverse") # Paket yoksa yükleyin
library(tidyverse)            # Paketi yüklemek
```

2.2 İlk Adımlar

2.2.1 mpg Veri Seti

mpg veri seti, farklı araç modellerine ait **motor hacmi** ve **yakıt verimliliği** gibi bilgileri içerir.

Veri setini görmek için:

```
mpg
```

Önemli değişkenler:

- **displ**: Motor hacmi (litre).
- **hwy**: Otoban yakıt verimliliği (mil/galon).

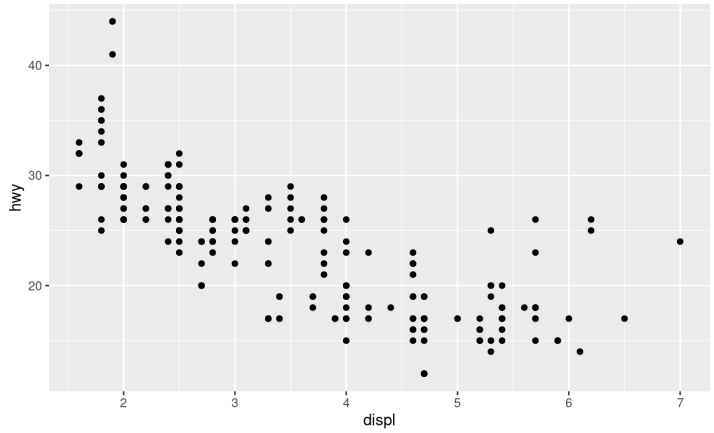
2.2.2 İlk Grafik Oluşturma

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

Açıklama:

- `ggplot()` ile bir grafik oluşturuyoruz.
- `geom_point()` ile **dağılım grafiği** ekliyoruz.
- `aes()` ile değişkenleri **x** ve **y** eksenlerine eşliyoruz.

Grafik: **Motor hacmi arttıkça yakıt verimliliği azalır**.



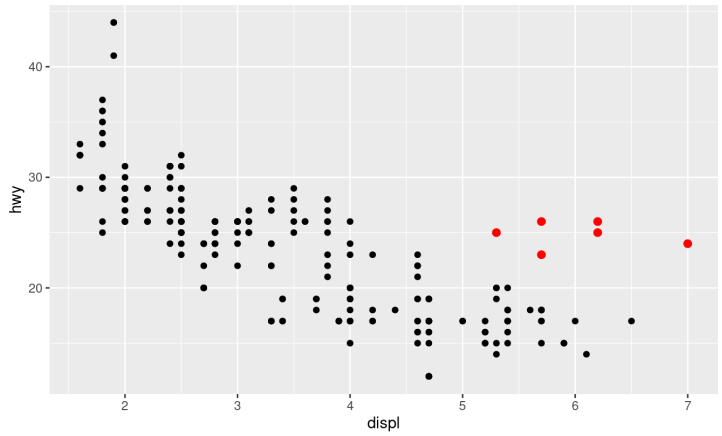
2.2.3 Grafik Şablonu

Grafik oluşturma şablonu:

```
ggplot(data = <VERİ_SETİ>) +
  <GEOM_FONKSİYONU>(mapping = aes(<EŞLEMELER>))
```

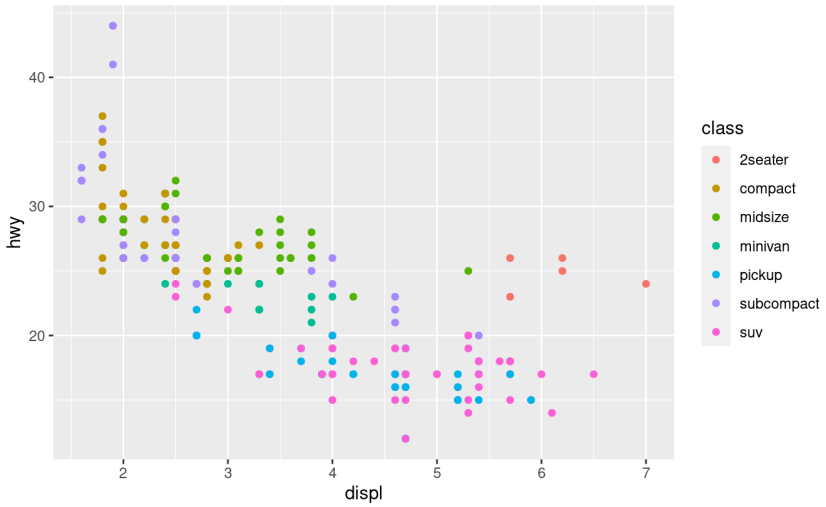
2.3 Estetik Eşlemeler (Aesthetic Mappings)

Estetikler: Grafiklerdeki görsel özelliklerdir (**renk, boyut, şekil** vb.).
Bir değişkeni bir estetiğe eşlemek için `aes()` kullanılır.



Örnek: Renk Eşlemesi

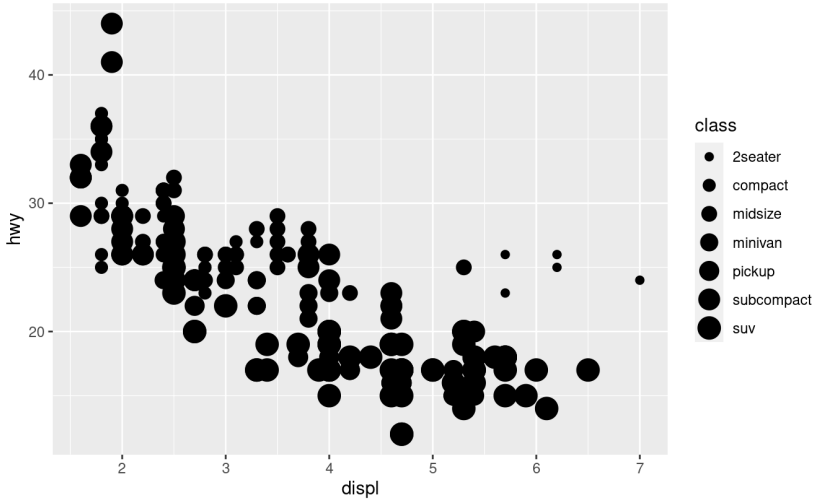
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



Örnek: Şekil ve Saydamlık

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```

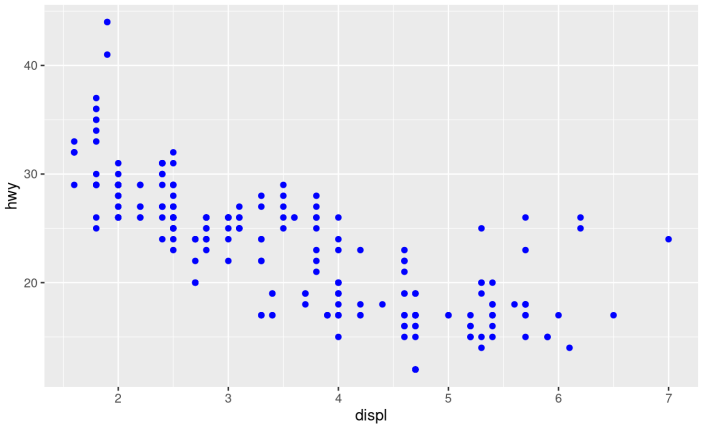
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
```



Manuel Estetik Ayarları

Tüm noktaları mavi yapmak için:

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color =
"blue")
```

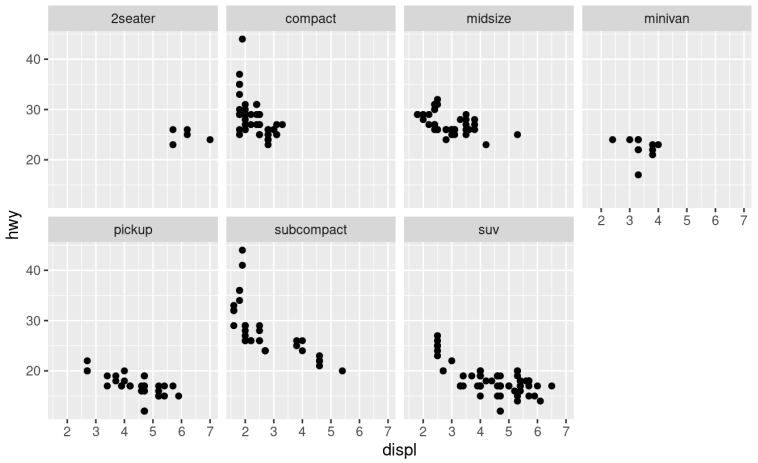


2.5 Facet'ler

Facet'ler, grafikleri **alt grafikler** (subplot) olarak bölmek için kullanılır.

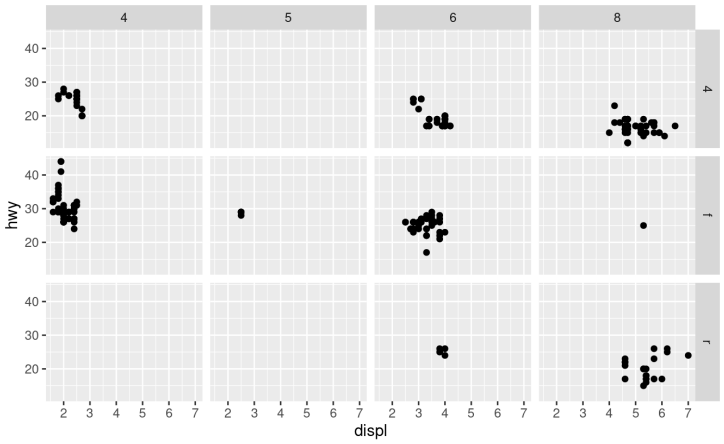
facet_wrap(): Tek Değişkenle Bölme

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```

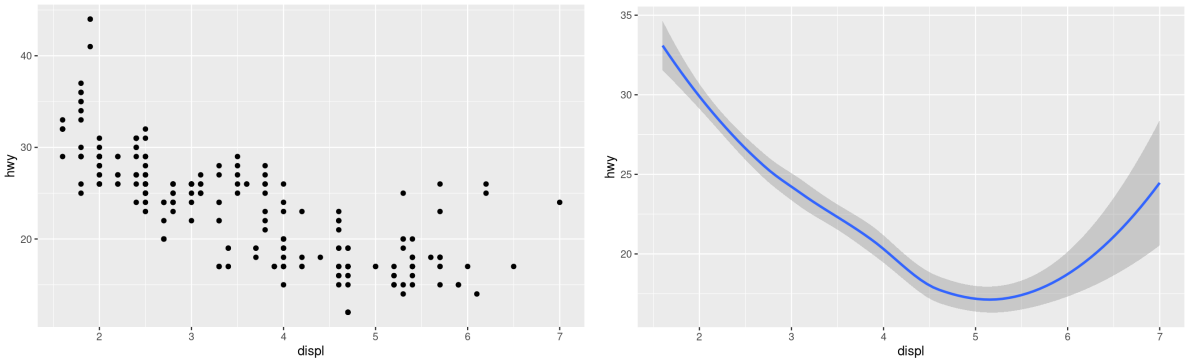


facet_grid(): İki Değişkenle Bölme

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



2.6 Geometrik Objeler (Geoms)



Dağılım Grafiği ve Çizgi Ekleme

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

Histogram

```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x = hwy), binwidth = 2)
```

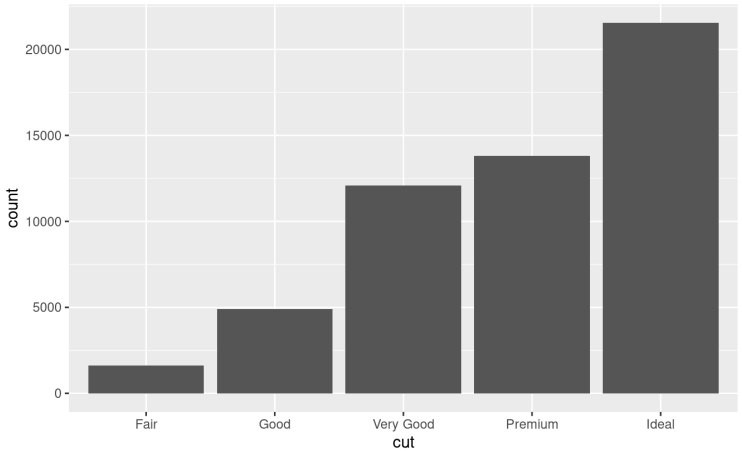
Kutu Grafiği (Boxplot)

```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = class, y = hwy))
```

2.7 İstatistiksel Dönüşümler (Stats)

Bar Grafiği

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
```

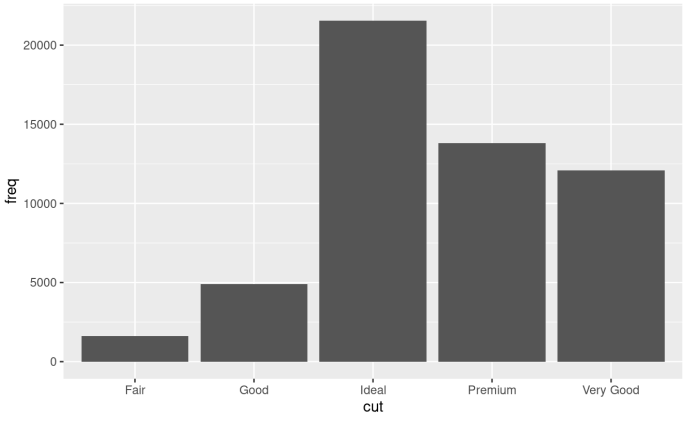


Özelleştirilmiş Bar Grafiği

Veriyi doğrudan kullanmak için:

```
demo <- tribble(
  ~cut,          ~freq,
  "Fair",        1610,
  "Good",        4906,
   "Very Good",  12082,
  "Premium",     13791,
   "Ideal",      21551)

ggplot(data = demo) +
  geom_bar(mapping = aes(x = cut, y = freq), stat =
"identity")
```



2.8 Konum Ayarları (Position Adjustments)

Stacked Bar Chart

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity))
```

Yan Yana Bar Grafiği

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = clarity), position =
"dodge")
```

Jitter ile Saçılma Grafiği

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), position =
"jitter")
```

2.9 Koordinat Sistemleri

Eksenleri Ters Çevirme (coord_flip)

```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```

Polar Koordinatlar (Pie Chart)

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = factor(1), fill = cut), width =
1) +
  coord_polar(theta = "y")
```

2.10 Grafiklerin Katmanlı Grameri

Grafik Şablonu

```
ggplot(data = <VERİ_SETİ>) +
  <GEOM_FONKSİYONU>(
     mapping = aes(<EŞLEMELER>),
     stat = <STAT>,
     position = <POSITION>
  ) +
  <KOORDİNAT_SİSTEMİ> +
  <FACET_FONKSİYONU>
```

Özet

Bu notta, **ggplot2** kullanarak veri görselleştirme temellerini öğrendiniz:

- **Veri Görselleştirme Şablonu**
- **Estetik Eşlemeler** (Renk, Şekil, Saydamlık)
- **Facet'ler** ile Alt Grafikler
- **Geometrik Objeler** (Scatter, Histogram, Boxplot)
- **Konum Ayarları** (Stacking, Jitter)
- **Koordinat Sistemleri** (Flip, Polar)

3.Missing Data Imputation

1. Giriş

İstatistiksel analizlerde eksik veriler **sıkça** karşılaşılan bir durumdur ve analiz sonucunu önemli ölçüde etkileyebilir. Eksik veriler, bir gözlemde bir değişken için veri saklanmadığında ortaya çıkar.

Eksik verileri ele almak için birkaç yöntem mevcuttur:

- Gözlemleri silme
- Eksik değerlerin yerine uygun tahminlerle doldurulması (**İmputasyon**)

Bu derste eksik verileri **impute etmek** (yerine değer koymak) için **MICE** paketi ve görselleştirme için **VIM** ve **Lattice** paketleri kullanılmıştır.

---

2. Veri Kaynağı

Veri seti: **NHANES** (Phase 1 of the Third National Health and Nutrition Examination Survey), **MICE** paketiyle birlikte gelir.

Veri Setini Yükleme

```
library(mice)  # MICE paketini yükle
data(nhanes)   # NHANES veri setini yükle
str(nhanes)    # Veri setinin yapısını incele
```

Çıktı:

- 25 gözlem ve 4 değişken içerir:
  - `age`: Yaş
  - `bmi`: Vücut Kitle İndeksi
  - `hyp`: Hipertansiyon durumu
  - `chl`: Kolesterol seviyesi

---

3. Eksik Değerleri Bulma

Yöntem 1: `summary()` Fonksiyonu

```
summary(nhanes)
```

Sonuç:

- 9 değer `bmi` için eksik
- 8 değer `hyp` için eksik
- 10 değer `chl` için eksik

Yöntem 2: `md.pattern()` Fonksiyonu

```
md.pattern(nhanes)
```

Çıktı:

- Tam veri içeren 13 gözlem mevcut.
- Bazı değişkenlerde farklı kombinasyonlarda eksiklikler var.

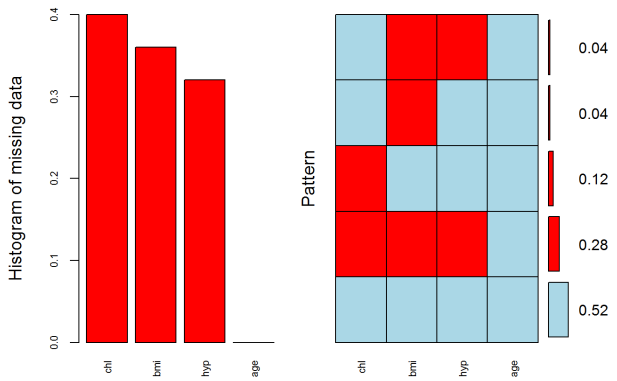
---

4. Eksik Veriyi Görselleştirme

`VIM` Paketi ile Görselleştirme

```
library(VIM)  # VIM paketini yükle
nhanes_plot <- aggr(nhanes,
                    col = c('lightblue', 'red'),
                    numbers = TRUE,
                    sortVars = TRUE,
                    labels = names(nhanes),
                    cex.axis = 0.7,
                    gap = 3,
                    ylab = c("Eksik Veri Histogramı", "Desen"))
```

- **Mavi**: Gözlemlenen veriler
- **Kırmızı**: Eksik veriler
- Eksik veri kombinasyonlarının frekanslarını gösterir.



---

5. Eksik Verilerin İşlenmesi

Tam Verileri Kullanarak Modelleme

Eksik verileri göz ardı edip analiz yapmak doğru değildir çünkü gözlem kaybına neden olur.

```
lreg.cc <- lm(chl ~ age + bmi, data = nhanes)
summary(lreg.cc)
```

- **13 gözlem** kullanıldı.
- Eksik veriye sahip **12 gözlem silindi.**

---

6. MICE Paketini Kullanarak İmputasyon

MICE ile Adımlar:

1. `mice()`: Eksik verileri doldurur.
2. `with()`: Tamamlanmış veri setlerini analiz eder.
3. `pool()`: Sonuçları birleştirir.

Eksik Verilerin Simülasyonu (İmputasyon)

```
impute <- mice(nhanes,
               m = 20,       # 20 tamamlanmış veri seti oluştur
               printFlag = FALSE,
               maxit = 40,
               seed = 2525)  # Rastgelelik için tohum değeri
```

Tamamlanmış Veri Setleri ile Regresyon Analizi

```
fit.mi <- with(data = impute, exp = lm(chl ~ age + bmi))
summary(pool(fit.mi))
```

Çıktı:

- `age` ve `bmi` değişkenleri anlamlıdır.
- Eksik değerler doldurularak modelde daha fazla gözlem kullanılmıştır.

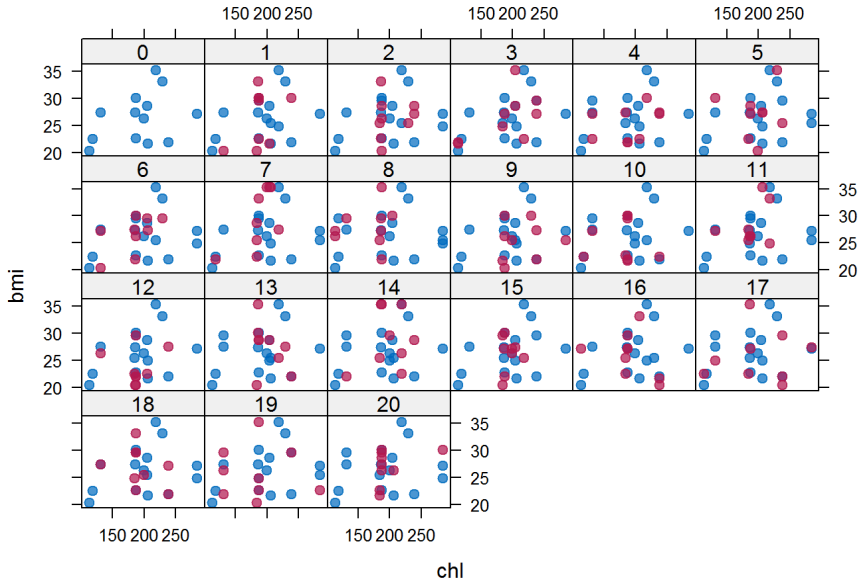
---

7. İmputasyon Sonuçlarının Görsel Kontrolü

Lattice Paketi ile Kontrol

```
library(lattice)  # Lattice paketini yükle
xyplot(impute, bmi ~ chl | .imp,
       pch = 20,
       cex = 1.4)
```

- **Mavi**: Gözlemlenen değerler
- **Kırmızı**: İmpute edilen değerler



---

8. Özet ve Sonuç

- Eksik verilerin silinmesi yerine **imputasyon** yapmak daha tutarlı sonuçlar verir.
- **MICE** paketi sayesinde veriler doldurularak analiz yapılır ve sonuçlar birleştirilir.
- **VIM** ve **Lattice** paketleri, verinin ve imputasyonun görsel analizini sağlar.

4.PCA

1. Giriş

PCA (**Principal Component Analysis**), yüksek boyutlu verilerde **boyut indirgeme** yöntemlerinden biridir. Temel amacı:

- **Veri setindeki varyansı** mümkün olduğunca az kayıpla açıklayacak yeni, birbirine dik (ortogonal) bileşenler oluşturmak.
- **Multicollinearity** sorununu ortadan kaldırmak.
- Regresyon modellerinde **aşırı uyumu** (overfitting) azaltmak.

PCA'nın temel mantığı: Verilerdeki en fazla varyansı açıklayan doğrusal bileşenleri bulmak ve onları sıralamaktır.

---

2. PCA'nın Temel İlkeleri

1. **Varyansın Önemi**:
    - Yüksek varyansa sahip değişkenler daha fazla bilgi içerir.
    - PCA, varyansı maksimize eden doğrusal kombinasyonları oluşturur.
2. **Ortogonal Bileşenler**:
    - Yeni bileşenler birbirine **dik** (ortogonal) olduğu için multicollinearity sorunu çözülür.
3. **Boyut İndirgeme**:
    - Varyansın çoğunu açıklayan bileşenler seçilerek boyut indirgeme sağlanır.
4. **Bileşenlerin Yorumu**:
    - Bileşenlerin ne anlama geldiğini yorumlamak zor olabilir. Ancak **biplot** grafikleri bu konuda yardımcı olabilir.

---

3. PCA Adımları

Adım 1: Veri Setini Oluşturma

```
# Örnek veri seti
Price <- c(6,7,6,5,7,6,5,6,3,1,2,5,2,3,1,2)
Software <- c(5,3,4,7,7,4,7,5,5,3,6,7,4,5,6,3)
Aesthetics <- c(3,2,4,1,5,2,2,4,6,7,6,7,5,6,5,7)
Brand <- c(4,2,5,3,5,3,1,4,7,5,7,6,6,5,5,7)
buy_computer <- tibble(Price, Software, Aesthetics, Brand)
```

Adım 2: PCA Modelini Oluşturma

PCA için `prcomp()` fonksiyonu kullanılır.

```
# PCA modelini oluştur
pca_buycomputer <- prcomp(buy_computer, scale = TRUE, center = TRUE)

# Model detaylarını incele
names(pca_buycomputer)
print(pca_buycomputer)
```

Adım 3: PCA Sonuçlarını İnceleme

- **Bileşenlerin Standart Sapması** ve **Varyans Oranı** incelenir.
- Önemli bileşenler, varyansın büyük kısmını açıklayan bileşenlerdir.

```
summary(pca_buycomputer, loadings = TRUE)
```

Çıktı:

Bileşen	PC1	PC2	PC3	PC4
Varyans Oranı	0.6076	0.2403	0.1162	0.03596
Kümülatif Oran	0.6076	0.8479	0.9640	1.0000

Bu sonuçlar:

- İlk iki bileşenin toplamda **%84.79** varyansı açıkladığını gösteriyor.

Neden İlk İki Bileşene Odaklanılıyor?

1. **Varyansın Büyük Kısmını Açıklama:**
    - **PC1 ve PC2**, toplam varyansın **%84.79'unu** açıklıyor.
    - Bu, verideki bilginin büyük kısmının yalnızca bu iki bileşende toplandığını gösteriyor.
2. **Azalan Katkı:**
    - **PC3** yalnızca **%11.62**
    - **PC4** ise yalnızca **%3.596** varyansı açıklıyor.
    - Bu bileşenlerin ek bilgi katkısı **çok küçük** olduğundan, onları modele dahil etmek genellikle gerekli değildir.
3. **Boyut İndirgeme:**
    - PCA'nın temel amacı, mümkün olan en az sayıda bileşenle verideki bilgi kaybını minimize etmektir.
    - İlk iki bileşeni kullanarak veri setinin boyutu azaltılırken, varyansın büyük kısmı korunmuş olur.

Diğer Bileşenlerin Rolü (PC3 ve PC4)

- **PC3** ve **PC4** düşük varyansa sahip olduğundan **gürültü** olarak düşünülebilir.
- Eğer detaylı analiz yapılması gerekiyorsa, örneğin çok küçük varyasyonların da incelenmesi gerekiyorsa, bu bileşenler de dahil edilebilir.
- Ancak çoğu uygulamada (örneğin makine öğrenmesi ya da regresyon), yalnızca **yüksek varyanslı** bileşenler kullanılır.

Adım 4: PCA Görselleştirme

Biplot Grafiği kullanarak bileşenlerin ve değişkenlerin katkısını görselleştirebiliriz.

```
library(ggbiplot)

# Biplot grafiği oluştur
g <- ggbiplot(pca_buycomputer, obs.scale = 1, var.scale = 1, groups
= as.character(OS),
              ellipse = TRUE, circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal', legend.position =
'top')
print(g)
```

Yorum:

- **PC1**: Brand ve Aesthetics değişkenleri en fazla katkıyı sağlıyor.
- **PC2**: Software değişkeni en fazla katkıyı sağlıyor.

---

4. PCA ile Regresyon Modeli Kurma

Adım 1: Bağımlı Değişken (OS) Ekleme

```
OS <- c(0,0,0,0,1,0,0,0,1,1,0,1,1,1,1,1)
```

Adım 2: Regresyon Modeli Oluşturma

PCA bileşenlerini kullanarak lojistik regresyon modeli kurulur:

```
model1 <- glm(OS ~ pca_buycomputer$x[,1] + pca_buycomputer$x[,2],
family = binomial)
summary(model1)
```

Çıktı:

- **PC1** anlamlıdır (`p-value < 0.05`).

Adım 3: Modelin Tahmin Performansı

Lojistik regresyon modeli, katılımcıların OS kullanımını tahmin edebilir:

```
fitted(model1)
```

Bu tahminler, OS'nin olasılıklarını verir. Örneğin:

- **Katılımcı 1 için 0.11 olasılık**: OS kullanıcısı olma olasılığı düşüktür.

---

5. PCA'nın Avantajları ve Dezavantajları

Avantajlar

1. **Boyut İndirgeme**: Daha az değişkenle analiz yapılır.
2. **Multicollinearity'yi Önler**: Bileşenler birbirine dik (ortogonal) olduğu için bağımsızdır.
3. **Aşırı Uyum (Overfitting) Sorunu Azaltılır.**
4. **Varyans Açıklama**: Bileşenler varyansın en büyük kısmını açıklar.

Dezavantajlar

1. **Yorumlama Güçlüğü**: PC1, PC2 gibi bileşenlerin neyi temsil ettiği doğrudan anlaşılamaz.
2. **Bağımlı Değişken Kullanılmaz**: PCA, yalnızca bağımsız değişkenleri dikkate alır.

---

6. Özet ve Sonuç

- PCA, özellikle **yüksek boyutlu veri setlerinde** güçlü bir **boyut indirgeme** ve analiz yöntemidir.
- **Varyansı maksimize eden** bileşenler bulunur ve sıralanır.
- PCA ile elde edilen bileşenler, regresyon analizlerinde kullanılabilir.

KAYNAKLAR:

https://www.datacamp.com/tutorial/pca-analysis-r

https://fderyckel.github.io/machinelearningwithr/princdipal-component-analysis.html

DYS'deki PCA örneği:(Breast Cancer Wisconsin Veri Seti)

1. Veri Seti ve Kurulum

Bu uygulamada **Breast Cancer Wisconsin** veri seti kullanılmaktadır. Veri setinde:

- **ID:** Hastaya ait benzersiz kimlik numarası.
- **Diagnosis:** Tümörün **iyi huylu (B)** veya **kötü huylu (M)** olduğunu belirtir.
- **30 Özellik:** Tümörün boyutu, şekli, yüzeyi gibi farklı özelliklerden oluşur.

Veri Yükleme ve Hazırlama

```
# Veri setini yükleme

wdbc <-
read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databas
es/breast-cancer-wisconsin/wdbc.data"), header=FALSE)



# Sütun isimlerini ayarlama

features <- c("radius", "texture", "perimeter", "area",
"smoothness", "compactness", "concavity", "concave_points",
"symmetry", "fractal_dimension")

names(wdbc) <- c("id", "diagnosis", paste0(features,"_mean"),
paste0(features,"_se"), paste0(features,"_worst"))
```

---

2. PCA ile Temel Bileşenleri Bulma

Amaç:

- PCA ile 30 değişkeni özetlemek.
- Varyansı en iyi açıklayan bileşenleri belirlemek.

PCA Modeli Oluşturma

`prcomp()` fonksiyonu kullanılarak PCA gerçekleştirilir:

```
wdbc.pr <- prcomp(wdbc[c(3:32)], center = TRUE, scale = TRUE)  # PCA
çalıştır

summary(wdbc.pr)  # PCA sonuçlarını incele
```

PCA Çıktısı:

Bileşen	PC1	PC2	PC3	...	PC30
Varyans Oranı	0.4427	0.1897	0.0939	...	0.00000
Kümülatif Oran	0.4427	0.6324	0.7263	...	1.0000

Yorum:

- **PC1** ve **PC2**, verinin toplam varyansının **%63.3'ünü** açıklıyor.
- PCA'nın temel amacı **boyut indirgeme** olduğu için ilk birkaç bileşen yeterlidir.

---

3. Önemli Bileşenlerin Seçimi

PCA'da hangi bileşenlerin önemli olduğunu anlamak için **screeplot** kullanılır:

```
screeplot(wdbc.pr, type = "l", npcs = 15, main = "Screeplot of the first 15 PCs")

abline(h = 1, col="red", lty=5)

legend("topright", legend=c("Eigenvalue = 1"),

       col=c("red"), lty=5, cex=0.6)
```

Screeplot Yorumlama:

- **Kırmızı çizgi:** Eigenvalue = 1
- Eigenvalue'si **1'in altında** olan bileşenler, tek bir değişkenden daha az varyans açıklar ve genellikle göz ardı edilir.
- İlk 6-8 bileşen genellikle yeterlidir.

---

4. PCA ile 2D Görselleştirme

PCA'nın güzelliği, yüksek boyutlu veriyi **2D bir alana** indirerek görselleştirebilmesidir.

Diagnosise Göre PCA Plotu

```
library(factoextra)

fviz_pca_ind(wdbc.pr, geom.ind = "point", pointshape = 21,
             pointsize = 2,
             fill.ind = wdbc$diagnosis,
             col.ind = "black",
             palette = "jco",
             addEllipses = TRUE,
             label = "var",
             col.var = "black",
             repel = TRUE,
             legend.title = "Diagnosis") +
  ggtitle("2D PCA-plot from 30 feature dataset") +
  theme(plot.title = element_text(hjust = 0.5))
```

Grafik Yorumlama:

- **Benign (B)** ve **Malignant (M)** grupları arasında **açık bir ayrım** gözlemleniyor.
- İlk iki bileşen (PC1 ve PC2), verideki varyansı oldukça iyi açıkladığı için bu ayrım netleşiyor.
- Bu durum, PCA'nın **sınıflandırma modellerine** katkı sağlayabileceğini gösteriyor.

---

5. Neden PCA Kullanılır?

1. **Boyut İndirgeme:**
    - 30 değişkeni iki bileşenle özetleyerek veriyi görselleştirme imkânı sağlar.
2. **Varyansı Açıklama:**
    - İlk iki bileşen, toplam varyansın **%63.3'ünü** açıklar.
3. **Sınıflandırma Performansı:**
    - PCA sonrası elde edilen bileşenler, sınıflandırma modellerinde kullanılabilir.
    - Bu örnekte, benign ve malignant tümörler PCA bileşenlerinde **ayrışmıştır**.

---

6. Sonuç

- PCA, yüksek boyutlu veri setlerinde önemli varyansları açıklayan bileşenler üretir.
- **İlk iki bileşen**, Breast Cancer Wisconsin veri setindeki **benign** ve **malignant** tümörleri ayırt etmekte oldukça etkilidir.
- Bu durum, PCA'nın bir **ön işleme** tekniği olarak sınıflandırma modellerinde kullanılabileceğini gösterir.

5.Logistic Regression

1. Giriş

Lojistik Regresyon, **classification problemlerinde** kullanılan bir **Makine Öğrenimi algoritmasıdır**.
Örneğin:

- **E-posta sınıflandırma:** Spam mi, değil mi?
- **Dolandırıcılık tespiti:** Online işlem dolandırıcılık mı, değil mi?
- **Tümör teşhisi:** Malign (kötü huylu) mi, Benign (iyi huylu) mi?

Lojistik regresyon, tahmin edilen çıktıyı **sigmoid (logistic) fonksiyonu** kullanarak **0 ve 1 arasında bir olasılık değeri** olarak dönüştürür.

---

2. Lojistik Regresyon Türleri

1. **Binary Sınıflandırma** (İkili): Tümör **Malign (1)** veya **Benign (0)**
2. **Multi-class Sınıflandırma** (Çoklu): **Kedi, köpek, koyun** gibi sınıflar.

---

3. Sigmoid Fonksiyonu

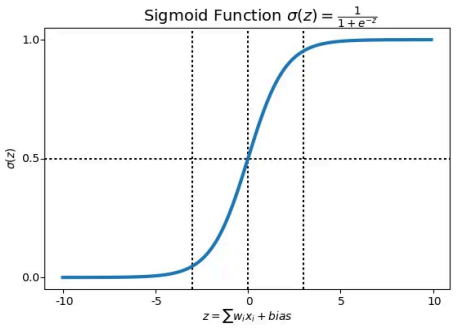
Sigmoid fonksiyonu, gerçek değerleri **0 ve 1 arasındaki olasılık değerlerine** dönüştürür.

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

Burada:

- $z = \beta_0 + \beta_1 X$
- $\sigma(z)$: Sigmoid fonksiyonu çıktısı

Sigmoid fonksiyonunun grafiği:



---

4. Lojistik Regresyon Hipotezi

Lojistik regresyonun hipotezi:

$h\Theta(x) = \beta_0 + \beta_1 X$
$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$
$Z = \beta_0 + \beta_1 X$
$h\Theta(x) = sigmoid(Z)$
$i.e.\ h\Theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

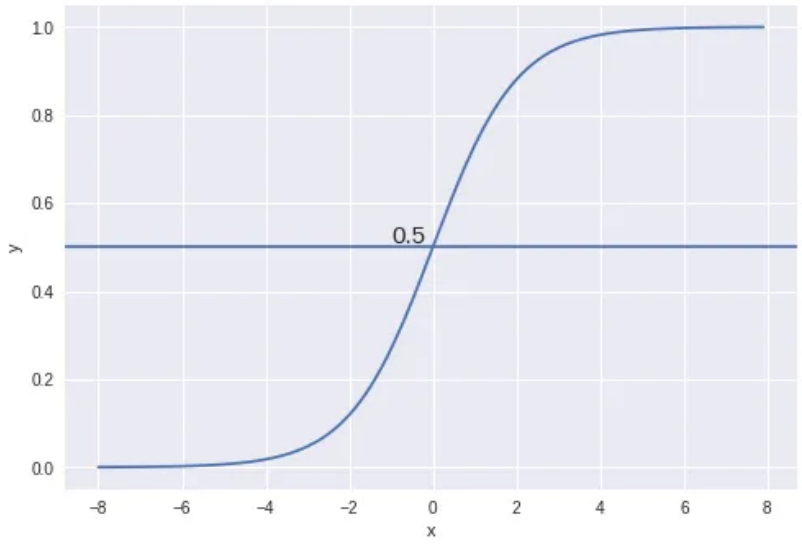
Bu formül, hθ(x)'nin 0 ve 1 arasında bir olasılık değeri vermesini sağlar.

---

5. Karar Sınırı (Decision Boundary)

Lojistik regresyonda bir **eşik değeri (threshold)** belirlenir. Örneğin, eşik değeri **0.5** olarak alınırsa:

- hθ(x)≥0.5h_{\theta}(x) \geq 0.5hθ(x)≥0.5: Sınıf 1
- hθ(x)<0.5h_{\theta}(x) < 0.5hθ(x)<0.5: Sınıf 0



---

6. Maliyet Fonksiyonu (Cost Function)

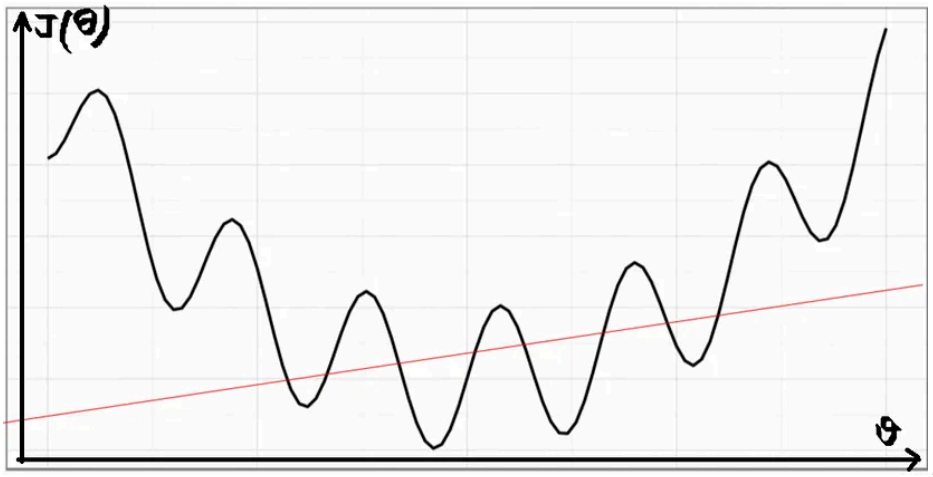
Lojistik regresyonda lineer regresyonun maliyet fonksiyonu kullanılamaz çünkü bu durumda maliyet fonksiyonu **non-convex** olur ve optimizasyon zordur.

Lojistik Regresyon Maliyet Fonksiyonu:

$log(\mathrm{h}\theta(\mathrm{x}))$ *if* $y = 1$

$-log(1-\mathrm{h}\theta(\mathrm{x}))$ *if* $y = 0$

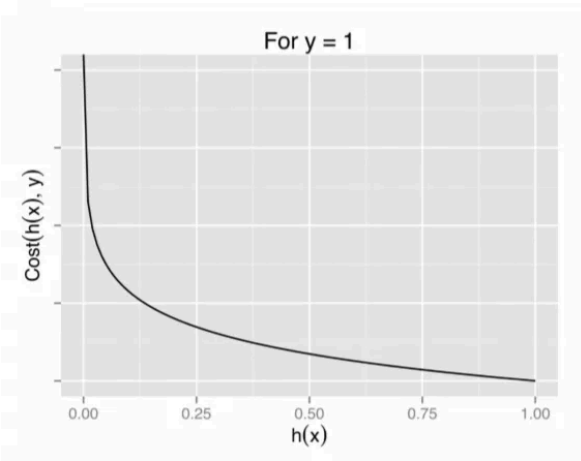
$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

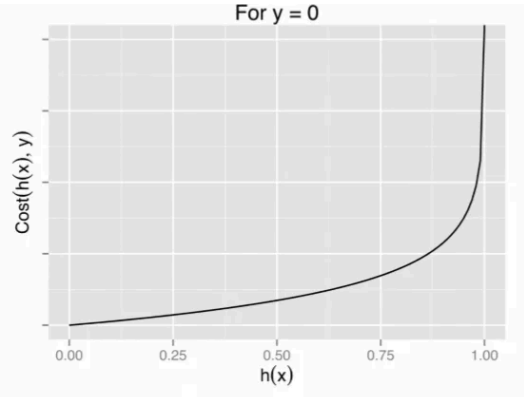


Non-convex function

The above two functions can be compressed into a single function

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1 - y^{(i)}\right)\log(1 - h\theta(x(i)))\right]$$





Above functions compressed into one cost function

---

7. Gradient Descent (Eğim İnişi)

Amaç: Maliyet fonksiyonunu minimize etmektir. ($\min J(\boldsymbol{\theta})$).
Gradient Descent, parametrelerin (β0,β1\beta_0, \beta_1β0,β1) güncellenmesini sağlar.

$$\theta j := \theta j - \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

Objective: To minimize the cost function we have to run the gradient descent function on each parameter

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

Burada:

- α\alphaα: Öğrenme oranı

---

8. Lojistik Regresyon Modelinin Performans Değerlendirilmesi,

Lojistik regresyon modelinin performansını değerlendirmek için aşağıdaki metrikler kullanılabilir:

1. AIC (Akaike Information Criteria)

- **AIC**, lojistik regresyonda **ayarlanmış R-kare**'ye benzer bir model uygunluk ölçütüdür.
- AIC değeri, modele dahil edilen bağımsız değişken sayısını cezalandırarak uygunluğu değerlendirir.
- **Düşük AIC değeri** daha iyi bir modeli ifade eder.

2. Null Deviance ve Residual Deviance

- **Null Deviance**:
    - Model yalnızca sabit bir **intercept** kullanarak tahmin yaparsa elde edilen deviance değeridir.
    - **Düşük değer**, daha iyi bir modeli gösterir.
- **Residual Deviance**:
    - Bağımsız değişkenler eklenerek yapılan tahminlerden elde edilen deviance değeridir.
    - **Düşük değer**, bağımsız değişkenlerin modele katkı sağladığını gösterir.

3. Confusion Matrix (Karmaşıklık Matrisi)

Confusion Matrix, **gerçek (Actual)** ve **tahmin edilen (Predicted)** değerleri gösteren bir tablodur.
Bu tablo yardımıyla aşağıdaki değerler hesaplanır:

- **True Positives (TP)**: Doğru olarak pozitif sınıflandırılanlar
- **True Negatives (TN)**: Doğru olarak negatif sınıflandırılanlar
- **False Positives (FP)**: Yanlış olarak pozitif sınıflandırılanlar
- **False Negatives (FN)**: Yanlış olarak negatif sınıflandırılanlar

Model Doğruluğu (Accuracy):

$$\text{Accuracy}=TP+TN \,/\, TP+TN+FP+FN$$

Özgüllük (Specificity):

$$\text{Specificity}=TN \,/\, TN+FP$$

Duyarlılık (Sensitivity):

$$\text{Sensitivity}=TP \,/\, TP+FN$$

1. Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. Precision:

$$Precision = \frac{TP}{TP + FP}$$

3. Recall (Sensitivity):

$$Recall = \frac{TP}{TP + FN}$$

4. F1-Score:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

5. Specificity:

$$Specificity = \frac{TN}{TN + FP}$$

4. ROC Eğrisi (Receiver Operating Characteristic)

ROC Eğrisi, modelin performansını değerlendirirken **True Positive Rate (Duyarlılık)** ile **False Positive Rate (1 - Specificity)** arasındaki dengeyi gösterir.

ROC Eğrisinin Temel Özellikleri:

1. **TPR (True Positive Rate):** Modelin doğru pozitif sınıfları tahmin etme oranı.
2. **FPR (False Positive Rate):** Modelin yanlış pozitif sınıfları tahmin etme oranı.

AUC (Area Under Curve):

- ROC eğrisinin altında kalan alanı ifade eder.
- **AUC değeri yüksek** olan modellerin tahmin gücü daha iyidir.
- **Mükemmel bir modelin ROC eğrisi:** TP = 1 ve FP = 0 olur. Eğri **sol üst köşeye** dokunur.

R ile Performans Ölçümü Uygulaması

```
# Lojistik Regresyon Modeli Kurulumu
logit_model <- glm(diagnosis ~ radius_mean + texture_mean +
perimeter_mean, family = binomial, data = data)

# Modelin Tahminleri
predictions <- predict(logit_model, type = "response")
predicted_classes <- ifelse(predictions > 0.5, 1, 0)

# Confusion Matrix
```

```
conf_matrix <- table(Predicted = predicted_classes, Actual =
data$diagnosis)
print(conf_matrix)

# Doğruluk, Sensitivity, Specificity Hesaplama
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
sensitivity <- conf_matrix[2,2] / (conf_matrix[2,2] +
conf_matrix[2,1])
specificity <- conf_matrix[1,1] / (conf_matrix[1,1] +
conf_matrix[1,2])

cat("Accuracy: ", accuracy, "\n")
cat("Sensitivity: ", sensitivity, "\n")
cat("Specificity: ", specificity, "\n")

# ROC Eğrisi ve AUC
library(pROC)
roc_curve <- roc(data$diagnosis, predictions)
plot(roc_curve, main = "ROC Curve")
auc_value <- auc(roc_curve)
cat("AUC: ", auc_value, "\n")
```

Yorumlama:

1. **AIC Değeri**: Modelin uygunluğunu değerlendirir, düşük olması istenir.
2. **Null Deviance ve Residual Deviance**: Modelin bağımsız değişkenler eklenerek ne kadar iyileştiğini gösterir.
3. **Confusion Matrix**: Modelin doğruluk, duyarlılık ve özgüllük değerlerini verir.
4. **ROC Eğrisi ve AUC**: Modelin sınıflandırma performansını özetler. **AUC ne kadar büyükse, modelin tahmin gücü o kadar iyidir.**

9. Lojistik Regresyon Örneği

Veri Setini Yükleme

Örnek: `Breast Cancer` veri seti (UCI'den):

```
# Gerekli paketleri yükleyin
library(dplyr)
```

```
library(ggplot2)

# Veriyi yükleme
wdbc <-
read.csv(url("http://archive.ics.uci.edu/ml/machine-learning-databas
es/breast-cancer-wisconsin/wdbc.data"), header = FALSE)

# Sütun isimlerini ayarlama
features <- c("radius", "texture", "perimeter", "area",
"smoothness", "compactness", "concavity", "concave_points",
"symmetry", "fractal_dimension")
names(wdbc) <- c("id", "diagnosis", paste0(features,"_mean"),
paste0(features,"_se"), paste0(features,"_worst"))
```

Veriyi Hazırlama

- `diagnosis` değişkenini binary formata çevirme:

```
wdbc$diagnosis <- ifelse(wdbc$diagnosis == "M", 1, 0)
```

- Model için gerekli sütunları seçme:

```
data <- wdbc %>% select(diagnosis, radius_mean, texture_mean,
perimeter_mean)
```

Model Kurma ve Analiz

Lojistik Regresyon Modeli:

```
# Modeli kur
logit_model <- glm(diagnosis ~ radius_mean + texture_mean +
perimeter_mean, family = binomial, data = data)

# Modelin özetini incele
summary(logit_model)
```

Tahmin ve Performans Ölçümü:

```
# Tahmin yap
```

```
predictions <- predict(logit_model, type = "response")

# Eşik değer uygulama (0.5)
predicted_classes <- ifelse(predictions > 0.5, 1, 0)

# Confusion Matrix oluştur
table(Predicted = predicted_classes, Actual = data$diagnosis)
```

Sonuçların Görselleştirilmesi

Modelin tahminlerini görselleştirme:

```
ggplot(data, aes(x = radius_mean, y = texture_mean, color =
factor(diagnosis))) +
  geom_point() +
  labs(title = "Lojistik Regresyon ile Sınıflandırma", color =
"Sınıf")
```

---

10. PCA ile Lojistik Regresyonda Multicollinearity'yi Giderme

Multicollinearity, bağımsız değişkenler arasında yüksek korelasyon olması durumudur. Bu durum lojistik regresyon gibi modellerde tahminleri dengesizleştirebilir. **PCA (Principal Component Analysis)**, bu sorunu çözmek için kullanılabilir.

Neden PCA Kullanılır?

- **Multicollinearity'yi Giderme:**
  - PCA, yüksek korelasyonlu değişkenlerden yeni **ortogonal (birbirine dik)** bileşenler türetir.
- **Boyut İndirgeme:**
  - Verinin varyansını büyük oranda açıklayan az sayıda bileşen kullanılır.
- **Modelin Kararlılığını Artırma:**
  - Lojistik regresyon modelleri, multicollinearity'den arındırılmış veriler üzerinde daha tutarlı çalışır.

Adım Adım PCA ve Lojistik Regresyon

1. PCA Uygulama:

```
# Gerekli paketler
library(dplyr)
library(caret)

# Veri setini hazırlama
data <- wdbc %>% select(-id)  # ID sütununu çıkar
data$diagnosis <- ifelse(data$diagnosis == "M", 1, 0)  # Binary
formatta diagnosis

# Bağımsız değişkenlere PCA uygulama
preProc <- preProcess(data[, -1], method = "pca", pcaComp = 5)  #
İlk 5 bileşen
pca_data <- predict(preProc, data[, -1])
pca_data <- data.frame(pca_data, diagnosis = data$diagnosis)
```

2. Lojistik Regresyon Modeli Kurma:

```
# Modeli oluşturma
logit_model_pca <- glm(diagnosis ~ ., data = pca_data, family =
binomial)
summary(logit_model_pca)
```

3. Model Performansını Değerlendirme:

```
# Tahmin ve Confusion Matrix
predictions <- predict(logit_model_pca, type = "response")
predicted_classes <- ifelse(predictions > 0.5, 1, 0)

# Performans Metrikleri
conf_matrix <- table(Predicted = predicted_classes, Actual =
pca_data$diagnosis)
print(conf_matrix)

accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
cat("Accuracy: ", accuracy, "\n")
```

11. Performans Metrikleri: Hangi Metrik Optimizasyon İçin Kullanılır?

Model performansını ölçerken farklı metrikler kullanılır. Her metrik, farklı durumlara uygundur:

Metrik	Tanım	Kullanıldığı Durum
Accuracy	Doğru tahminlerin tüm tahminlere oranı.	Veri dengeli ise kullanılabilir.
Precision	Doğru pozitiflerin, toplam pozitif tahminlere oranı.	Yanlış pozitifleri minimize etmek önemliyse kullanılır.
Recall (Sensitivity)	Doğru pozitiflerin, toplam gerçek pozitiflere oranı.	Kaçırılmaması gereken durumlarda kullanılır.
F1-Score	Precision ve Recall'ın harmonik ortalaması.	Veri dengesizse ve her ikisi de önemliyse kullanılır.
Specificity	Doğru negatiflerin, toplam negatiflere oranı.	Yanlış pozitifleri azaltmak önemliyse kullanılır.

ROC Eğrisi ve AUC

ROC Eğrisi:

- **True Positive Rate (Sensitivity)** ve **False Positive Rate (1 - Specificity)** arasındaki ilişkiyi gösterir.

AUC (Area Under Curve):

- ROC eğrisinin altında kalan alan.
- **AUC değeri** ne kadar yüksekse, modelin performansı o kadar iyidir.

Örnek Uygulama: Performans Metrikleri ve ROC Eğrisi

```
# Model performansı için gerekli paketler
library(pROC)

# ROC eğrisi
roc_curve <- roc(pca_data$diagnosis, predictions)
plot(roc_curve, main = "ROC Curve")
auc_value <- auc(roc_curve)
cat("AUC: ", auc_value, "\n")
```

```r
# Precision, Recall ve F1-Score hesaplama
TP <- conf_matrix[2,2]
FP <- conf_matrix[1,2]
FN <- conf_matrix[2,1]
TN <- conf_matrix[1,1]

precision <- TP / (TP + FP)
recall <- TP / (TP + FN)
specificity <- TN / (TN + FP)
f1_score <- 2 * (precision * recall) / (precision + recall)

cat("Precision: ", precision, "\n")
cat("Recall (Sensitivity): ", recall, "\n")
cat("Specificity: ", specificity, "\n")
cat("F1-Score: ", f1_score, "\n")
```

Hangi Metrik Kullanılmalı?

1. **Dengeli Veri:**
    - **Accuracy** uygundur.
2. **Yanlış Pozitifler Önemli:**
    - **Precision** tercih edilir (örneğin: e-posta spam filtreleri).
3. **Yanlış Negatifler Kritik:**
    - **Recall** tercih edilir (örneğin: kanser teşhisi).
4. **Veri Dengesiz:**
    - **F1-Score** kullanılır.
5. **Genel Performans:**
    - **ROC Eğrisi** ve **AUC** en kapsamlı ölçüm sağlar.

Sonuç:

1. **PCA**, lojistik regresyon modellerinde **multicollinearity** sorununu ortadan kaldırır ve modeli stabilize eder.
2. Performans metrikleri, modele ve probleme bağlı olarak farklı durumlarda optimize edilmelidir.
3. **ROC Eğrisi** ve **AUC** modeli özetlemek için en güçlü performans ölçütleridir.

Bu yöntemlerle lojistik regresyon modeli optimize edilir ve doğru performans ölçümü yapılır.

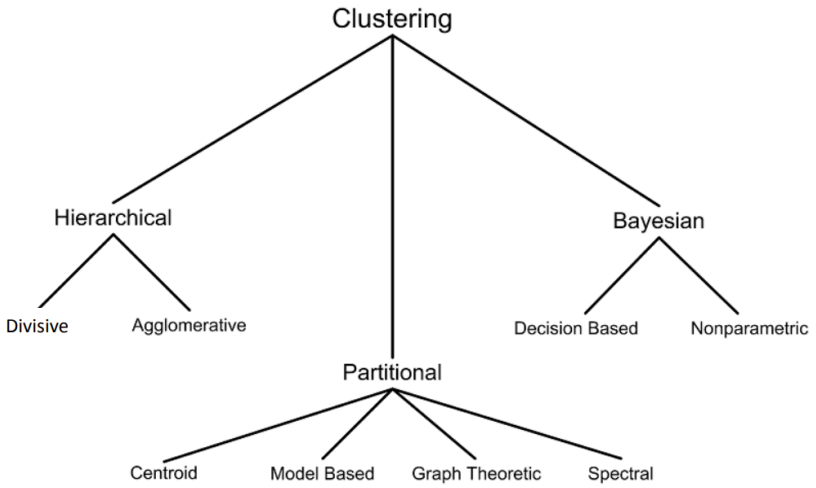
12. Sonuç

- **Lojistik Regresyon**, sınıflandırma problemlerinde **olasılık tahmini** sağlar.
- **Sigmoid Fonksiyonu**, tahminleri 0 ve 1 arasında normalize eder.
- **Gradient Descent**, model parametrelerini optimize ederek maliyet fonksiyonunu minimize eder.
- Model performansı **confusion matrix** ve **doğruluk oranı** gibi metriklerle değerlendirilir.

6.Clustering

Clustering, etiketsiz verileri benzerlik gruplarına ayıran bir **unsupervised öğrenme** yöntemidir. Gruplar, birbirlerine **benzer** ve diğer gruplara **farklı** olan veri noktalarından oluşur. Clustering yöntemleri; görsel nesne tanıma, biyolojik sınıflandırma ve görüntü segmentasyonu gibi çeşitli uygulamalarda kullanılır.

1. Clustering Techniques



1.1 Partitional (Bölme Tabanlı) Algoritmalar

- Veriyi önceden belirlenmiş bir küme sayısına (kkk) böler.
- **Örnekler**:
    - **K-Means**: Klasik ve hızlı bir algoritma.
    - **K-Medoids**: Outlier'lara daha az duyarlıdır.
    - **Model-Based Clustering**: Kümeler, bir modele dayalı olarak tanımlanır.(Gaussian Mixture Models (GMM))

1.2 Hierarchical (Hiyerarşik) Algoritmalar

- Verileri bir ağaç yapısı şeklinde organize eder.
- **Türleri**:
    - **Agglomerative (Birleştirici)**: Küçük kümeler birleştirilir.
    - **Divisive (Bölücü)**: Büyük kümeler parçalanır.
- **Örnekler**: Dendrogram kullanarak görselleştirilen yöntemler.

1.3 Density-Based Clustering

- Kümeler, veri noktalarının yoğunluğuna göre oluşturulur.
- **Örnekler**:
    - **DBSCAN**: Gürültü noktalarını tanımlar.
    - **OPTICS**: Yoğunluk tabanlı kümeleri ölçeklendirebilir.

1.5 Grid-Based Clustering

- Veri alanını bir ızgaraya böler ve kümeleri bu gridler üzerinden belirler.
- **Örnekler**: STING, CLIQUE.

Teknik	Avantajları	Dezavantajları
Hierarchical	Küme sayısı gerekmez, dendrogram sağlar.	Büyük veri setlerinde yavaştır.
Partitional	Hızlı ve verimlidir (örneğin K-Means).	Küme sayısının önceden belirlenmesi gerekir.
Density-Based	Gürültüleri tanır, karmaşık kümeleri bulur.	Yoğunluk parametrelerine duyarlıdır.
Bayesian	Olasılıksal modelleme sağlar.	Hesaplama karmaşıklığı yüksektir.
Grid-Based	Büyük veri setlerinde hızlıdır.	Küçük grid hücreleri detay kaybına yol açabilir.

2. K-Means Clustering

K-Means algoritması, veriyi önceden belirlenmiş bir küme sayısına böler.

Algoritmanın Adımları

1. Rastgele kkk merkezi seç.
2. Her veri noktasını en yakın merkeze ata.
3. Her kümenin yeni merkezini hesapla.
4. Kriter karşılanana kadar 2. ve 3. adımları tekrarla.

Güçlü Yönleri

- **Basit** ve anlaşılır.
- **Verimli**: O(tkn)O(tkn)O(tkn) zaman karmaşıklığı.
- Büyük veri setleri için uygundur.

Zayıf Yönleri

- Küme sayısının (kkk) önceden belirlenmesi gerekir.
- **Outlier'lara duyarlıdır**.
- Küre şeklinde olmayan kümelerde başarısız olabilir.
- Başlangıç noktalarına hassastır.

---

3. Hiyerarşik Kümeleme

Veriler arasında mesafeye dayalı olarak kümeler oluşturulur. Sonuçlar bir **dendrogram** ile görselleştirilir.

Dendrogram Nedir?

- Kümeleme işlemini görselleştiren bir ağaç yapısıdır.
- Y ekseni: Kümeler arası uzaklık.
- X ekseni: Veri noktaları veya kümeler.

Hiyerarşik Yöntemler

1. **Agglomerative (Birleştirici)**:
    - Veri noktaları önce kendi kümeleri olarak başlar.
    - En yakın kümeler birleştirilir.
2. **Divisive (Bölücü)**:
    - Tüm veri tek bir kümede başlar.
    - Kademeli olarak daha küçük kümelere bölünür.

Avantaj ve Dezavantajlar

- **Avantajlar**:
    - Küme sayısı önceden belirlenmek zorunda değildir.
    - Küçük veri setleri için uygundur.
- **Dezavantajlar**:
    - Büyük veri setlerinde yavaş.
    - Sonuçlar, kullanılan mesafe ölçütüne duyarlıdır.

---

4. Optimum Küme Sayısının Belirlenmesi

Küme sayısını seçmek için üç temel yöntem vardır:

4.1 Dirsek Yöntemi (Elbow Method)

WSS'nin (Within-Cluster Sum of Squares) eğrisindeki kırılma noktası optimum küme sayısını verir.

```
fviz_nbclust(employed_pop, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2) +
  labs(subtitle = "Dirsek Yöntemi")
```

4.2 Silhouette Yöntemi

Silhouette skorlarına dayalı olarak optimum küme sayısını belirler.

```
fviz_nbclust(employed_pop, kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette Yöntemi")
```

4.3 Gap İstatistiği

Farklı kkk değerleri için küme içi varyansı değerlendirir.

```
fviz_nbclust(employed_pop, kmeans, nstart = 25, method = "gap_stat",
nboot = 500)
```

---

5. R ile Uygulama

K-Means Örneği

```
# Veri setini yükleme
employed_pop <-
read.csv("https://statsandr.com/blog/data/Eurojobs.csv", sep = ",",
dec = ".", header = TRUE, row.names = 1)

# K-Means model
model <- kmeans(employed_pop, centers = 3)

# Görselleştirme
library(factoextra)
fviz_cluster(model, employed_pop, ellipse.type = "norm")
```

Hiyerarşik Kümeleme Örneği

```
# Hiyerarşik model
hclust_model <- hclust(dist(employed_pop), method = "average")

# Dendrogram çizimi
plot(hclust_model)
rect.hclust(hclust_model, k = 2, border = "blue")
```

6. Kümeleme Kalite Değerlendirme

- **Cohesion (Tutarlılık)**: Veri noktalarının centroid'e yakınlığı.
- **Separation (Ayrışma)**: Farklı kümeler arasındaki mesafe.

R'de kalite değerlendirme:

```
BSS <- model$betweenss
TSS <- model$totss
cat("BSS / TSS: ", BSS / TSS * 100, "%")
```

---

7. Kümeleme Uygulamaları

- **Pazarlama**: Müşteri segmentasyonu.
- **Biyoloji**: Gen ifadeleri analizi.
- **Görüntü İşleme**: Görüntü segmentasyonu.
- **Sosyal Bilimler**: Demografik analiz.

7. Performance Analysis

Makine öğrenmesi modellerinin başarısını değerlendirmek için **performans metrikleri** kullanılır. Bu metrikler, modelin ne kadar iyi çalıştığını ölçmekte kritik bir rol oynar. Farklı problem türleri (sınıflandırma, regresyon vb.) için farklı metrikler kullanılır.

---

1. Sınıflandırma Metrikleri

Sınıflandırma, veriyi belirli kategorilere ayırmayı amaçlayan bir makine öğrenmesi problemidir.

1.1 Confusion Matrix (Karışıklık Matrisi)

Confusion matrix, sınıflandırma modelinin tahmin sonuçlarını özetleyen bir tablodur. **True Positive (TP)**, **True Negative (TN)**, **False Positive (FP)** ve **False Negative (FN)** gibi değerleri içerir.

	Gerçek Pozitif (1)	**Gerçek Negatif (0)**
Tahmin Pozitif	True Positive (TP)	False Positive (FP)
Tahmin Negatif	False Negative (FN)	True Negative (TN)

1.2 **Accuracy (Doğruluk)**

Modelin tüm doğru tahminlerinin toplam tahminlere oranıdır.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Avantajı**: Kolay ve anlaşılırdır.
- **Dezavantajı**: Veri dengesizliği olan durumlarda yanıltıcı olabilir.

1.3 **Precision (Kesinlik)**

Pozitif olarak tahmin edilen örneklerin ne kadarının gerçekten pozitif olduğunu gösterir.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Yüksek **Precision**, modelin "yanlış pozitifleri" minimuma indirdiğini gösterir.

1.4 **Recall (Duyarlılık veya TPR - True Positive Rate)**

Gerçek pozitif örneklerin ne kadarının doğru tahmin edildiğini gösterir.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- Yüksek **Recall**, modelin pozitif sınıfı iyi bir şekilde tahmin ettiğini gösterir.

1.5 **F1 Score**

Precision ve Recall değerlerinin harmonik ortalamasıdır. Dengesiz veri setleri için idealdir.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- F1 skoru, Precision ve Recall arasındaki dengeyi sağlar.

1.6 **ROC-AUC (Receiver Operating Characteristic – Area Under Curve)**

- **ROC Eğrisi**: Farklı eşik değerlerinde **TPR** ve **FPR** (**False Positive Rate**) değerlerinin grafiğidir.
- **AUC** (**Area Under Curve**): ROC eğrisinin altında kalan alan.
  - AUC = 1: Mükemmel model.
  - AUC = 0.5: Rastgele tahmin.

---

2. **Regresyon Metrikleri**

Regresyon, sürekli bir değeri tahmin etmeyi amaçlayan bir makine öğrenmesi problemidir.

2.1 **Mean Absolute Error (MAE)**

Tahmin edilen ve gerçek değerler arasındaki ortalama mutlak farktır.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Yüksek değerler, modelin büyük hatalar yaptığını gösterir.

2.2 **Mean Squared Error (MSE)**

Tahmin hatalarının karelerinin ortalamasıdır. Büyük hatalara daha fazla ağırlık verir.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2.3 **Root Mean Squared Error (RMSE)**

MSE'nin karekökü dür. Hataları gerçek değerlerle aynı birime çevirir.

$$\text{RMSE} = \sqrt{\text{MSE}}$$

- Küçük RMSE değerleri, modelin daha iyi performans gösterdiğini gösterir.

2.4 **R-Squared (R^2)**

Modelin veriyi ne kadar iyi açıkladığını ölçer.

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

- R^2 değeri 1'e ne kadar yakınsa model o kadar iyidir.

---

3. **Sınıflandırma ve Regresyon için Ortak Metrikler**

3.1 **Log Loss (Logaritmik Kayıp)**

Modelin tahminlerinin olasılık değerleri üzerinden doğruluğunu ölçer.

$$\text{Log Loss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

- Daha düşük Log Loss değeri daha iyi performansı gösterir.

3.2 **Cross-Entropy**

- Sınıflandırma problemlerinde sınıf olasılıklarını tahmin eden modeller için kullanılır.
- Gerçek ve tahmin edilen dağılımlar arasındaki farkı ölçer.

4. Özel Durum Metrikleri

4.1 Gini Coefficient

- Modelin eşitsiz tahminler üzerindeki doğruluğunu ölçer. Özellikle finansal problemlerde kullanılır.

4.2 Kappa Statistics

- Rastgele tahmin olasılığını düzelterek sınıflandırma modelinin doğruluğunu ölçer.

---

5. Özet Tablosu

Metrik	Kullanım Alanı	Tanım
Accuracy	Sınıflandırma	Doğru tahminlerin oranı.
Precision	Sınıflandırma	Pozitif tahminlerin doğruluğu.
Recall	Sınıflandırma	Gerçek pozitiflerin tespiti.
F1 Score	Sınıflandırma	Precision ve Recall'un harmonik ortalaması.
ROC-AUC	Sınıflandırma	Model performansının görsel ölçütü.
MAE	Regresyon	Ortalama mutlak hata.
MSE	Regresyon	Hataların karelerinin ortalaması.
RMSE	Regresyon	MSE'nin karekökü.
R^2	Regresyon	Modelin açıklayıcı gücü.
Log Loss	Ortak (Sınıflandırma)	Olasılıklı tahminlerin kaybı.

8. Examples

Iris Veri Seti Üzerinde R ile Detaylı Analiz

Aşağıda **Iris** veri seti kullanılarak adım adım analiz gerçekleştirilmiştir. **Exploratory Data Analysis (EDA)**, **Veri Görselleştirme**, **PCA**, **Eksik Veri Doldurma**, **Lojistik Regresyon**, **Clustering** ve **Performans Analizi** başlıkları detaylı kodlar ve açıklamalarla sunulmuştur.

---

1. Exploratory Data Analysis (EDA)

Verinin genel yapısını anlamak için temel incelemeler yapılır.

```
# Gerekli kütüphaneler
library(tidyverse)  # Veri işleme
library(caret)      # Modelleme ve veri bölme

# Veri setini yükleme
data("iris")
df <- iris

# İlk 6 satırı görüntüleme
head(df)

# Veri setinin yapısı
str(df)

# Özet istatistikler
summary(df)

# Türlere göre dağılım
table(df$Species)
```

Çıktı:

- **Sepal.Length, Sepal.Width, Petal.Length, Petal.Width** adında 4 sürekli değişken.
- **Species**: Hedef değişken (3 sınıf: `setosa`, `versicolor`, `virginica`).

---

2. Data Visualization

Veri setindeki özelliklerin dağılımını ve ilişkilerini inceleyelim.

2.1. Boxplot (Türlere Göre Dağılım)

```
# Türlere göre boxplot
df %>%
  gather(key = "Feature", value = "Value", -Species) %>%
  ggplot(aes(x = Species, y = Value, fill = Species)) +
  geom_boxplot() +
  facet_wrap(~Feature, scales = "free") +
  theme_minimal() +
  ggtitle("Iris Özelliklerinin Türlere Göre Dağılımı")
```

2.2. Scatter Plot (Özellikler Arası İlişki)

```
# Özellikler arası ilişki (scatter plot)
pairs(df[1:4], col = df$Species, main = "Özellikler Arası İlişki")
```

---

3. Principal Component Analysis (PCA)

PCA kullanarak boyut indirgeme gerçekleştirelim.

```
# PCA uygulama
pca_model <- prcomp(df[, 1:4], center = TRUE, scale. = TRUE)

# PCA sonuçları
summary(pca_model)

# PCA bileşenlerinin görselleştirilmesi
library(factoextra)
fviz_pca_ind(pca_model,
             geom.ind = "point",
             col.ind = df$Species,
             palette = c("red", "blue", "green"),
             addEllipses = TRUE,
             legend.title = "Species",
             title = "PCA: İlk İki Bileşen")
```

Yorum:
İlk iki ana bileşen (PC1 ve PC2), verideki toplam varyansın büyük kısmını açıklamaktadır.

---

4. Missing Data Imputation (Eksik Veri Doldurma)

Bazı değerleri rastgele eksik hale getirip dolduralım.

```
# Eksik veri ekleme
set.seed(42)
df_missing <- df
df_missing[sample(1:nrow(df), 10), "Sepal.Length"] <- NA

# Eksik verileri görüntüleme
summary(df_missing)

# Mice kullanarak eksik veri doldurma
library(mice)
imputed_data <- mice(df_missing, method = "pmm", m = 5)
df_filled <- complete(imputed_data)

# Eksik verilerin doldurulduğunu kontrol etme
summary(df_filled)
```

---

5. Logistic Regression (Lojistik Regresyon)

5.1. Veri Bölme ve Model Kurulumu

```
# Eğitim ve test veri setlerinin oluşturulması
set.seed(123)
train_index <- createDataPartition(df$Species, p = 0.8, list =
FALSE)
train_data <- df[train_index, ]
test_data <- df[-train_index, ]

# Lojistik regresyon modeli
model <- multinom(Species ~ ., data = train_data)

# Test setinde tahmin
pred <- predict(model, test_data)
```

```
# Performans ölçümü
conf_matrix <- confusionMatrix(pred, test_data$Species)
print(conf_matrix)
```

Çıktı:

- Doğruluk oranı (**Accuracy**), Precision, Recall ve F1 Score gibi metrikler görüntülenir.

---

6. Clustering (Kümeleme)

6.1. K-Means Kümeleme

```
# K-Means model kurulumu
set.seed(123)
kmeans_model <- kmeans(df[, 1:4], centers = 3)

# Küme sonuçlarını görselleştirme
fviz_cluster(kmeans_model, data = df[, 1:4], geom = "point",
ellipse.type = "norm", ggtheme = theme_minimal())

# Kümelerin detayları
print(kmeans_model$centers)
```

6.2. Hiyerarşik Kümeleme

```
# Hiyerarşik kümeleme
hc_model <- hclust(dist(df[, 1:4]), method = "ward.D2")

# Dendrogram çizimi
plot(hc_model, main = "Hiyerarşik Kümeleme Dendrogramı", sub = "",
xlab = "")
rect.hclust(hc_model, k = 3, border = "red")
```

---

7. Performance Analysis (Performans Analizi)

K-Means Performansı

```
# K-Means performansı: BSS / TSS oranı
BSS <- kmeans_model$betweenss
TSS <- kmeans_model$totss
cat("K-Means BSS/TSS Oranı: ", BSS / TSS * 100, "%\n")
```

Lojistik Regresyon Performansı

```
# Confusion Matrix sonuçları
print(conf_matrix)

# Doğruluk oranı
cat("Lojistik Regresyon Doğruluk Oranı: ",
conf_matrix$overall["Accuracy"])
```

Sonuç

Bu çalışma kapsamında Iris veri seti üzerinde:

1. **EDA** ile verinin genel yapısı incelendi.
2. **Veri Görselleştirme** ile dağılım ve ilişkiler görselleştirildi.
3. **PCA** ile boyut indirgeme yapıldı.
4. **Eksik Veriler Dolduruldu** (Missing Data Imputation).
5. **Lojistik Regresyon** modeli eğitildi ve performansı değerlendirildi.
6. **Kümeleme** yöntemleri (K-Means ve Hiyerarşik) uygulandı.
7. **Performans Analizi** ile modeller değerlendirildi.

Breast Cancer Veri Seti Üzerinde R ile Detaylı Analiz

1. Veri Setinin Hazırlanması ve EDA

```
# Gerekli kütüphaneleri yükleme
library(tidyverse)
library(caret)
library(ggplot2)
library(factoextra)
library(mice)    # Eksik veri doldurma için

# Breast Cancer veri setini yükleme
data("BreastCancer", package = "mlbench")
df <- BreastCancer

# Veri setinin ilk satırları
head(df)

# Veri yapısını kontrol etme
str(df)

# Tür dönüşümü: ID sütunu kaldır, faktörleri numeric yap
df <- df %>% select(-Id)
df[] <- lapply(df, function(x) ifelse(is.factor(x),
as.numeric(as.character(x)), x))

# Eksik verileri kontrol etme
summary(df)
```

Yorum:

- Veri setinde **30 bağımsız değişken** ve **1 hedef değişken** (`Class`: benign/malignant) bulunur.
- Tür dönüşümleri yapılarak veri analizine hazır hale getirildi.

---

2. Veri Görselleştirme

2.1. Hedef Değişkenin Dağılımı

```
# Sınıf dağılımı
ggplot(df, aes(x = Class)) +
  geom_bar(fill = "steelblue") +
  ggtitle("Hedef Değişken Dağılımı")
```

2.2. Özelliklerin Histogramı

```
# Histogram ile özellik dağılımı
df %>%
  gather(key = "Feature", value = "Value", -Class) %>%
  ggplot(aes(x = Value, fill = Class)) +
  geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
  facet_wrap(~Feature, scales = "free") +
  theme_minimal() +
  ggtitle("Özelliklerin Dağılımı")
```

---

3. PCA (Principal Component Analysis)

PCA ile Boyut İndirgeme

```
# PCA uygulama
pca_model <- prcomp(df[, -10], center = TRUE, scale. = TRUE)

# PCA özet sonuçları
summary(pca_model)

# PCA görselleştirme
fviz_pca_ind(pca_model,
             geom.ind = "point",
             col.ind = df$Class,
             palette = c("blue", "red"),
             addEllipses = TRUE,
             legend.title = "Class",
             title = "PCA: İlk İki Bileşen")
```

---

4. Missing Data Imputation (Eksik Veri Doldurma)

Eksik Veri Doldurma

```
# Eksik veri ekleme (örnek)
set.seed(42)
df_missing <- df
df_missing[sample(1:nrow(df), 20), "Cl.thickness"] <- NA

# Eksik verileri doldurma
imputed_data <- mice(df_missing, method = "pmm", m = 5)
df_filled <- complete(imputed_data)

# Eksik verilerin doldurulduğunu kontrol etme
summary(df_filled)
```

---

5. Logistic Regression (Lojistik Regresyon)

5.1. Veri Bölme ve Model Kurulumu

```
# Eğitim ve test setlerine ayırma
set.seed(123)
train_index <- createDataPartition(df$Class, p = 0.8, list = FALSE)
train_data <- df[train_index, ]
test_data <- df[-train_index, ]

# Logistic Regression Modeli
model <- glm(Class ~ ., data = train_data, family = binomial)

# Model özet
summary(model)

# Test setinde tahmin
pred_prob <- predict(model, test_data, type = "response")
pred <- ifelse(pred_prob > 0.5, 1, 0)

# Confusion Matrix ve performans ölçümü
conf_matrix <- confusionMatrix(as.factor(pred),
as.factor(test_data$Class))
print(conf_matrix)
```

6. Clustering (Kümeleme)

6.1. K-Means Kümeleme

```
# K-Means uygulama
set.seed(123)
kmeans_model <- kmeans(df[, -10], centers = 2)

# Küme sonuçlarını görselleştirme
fviz_cluster(kmeans_model, data = df[, -10],
             geom = "point",
             ellipse.type = "norm",
             ggtheme = theme_minimal(),
             main = "K-Means Kümeleme")
```

6.2. Hiyerarşik Kümeleme

```
# Hiyerarşik kümeleme
hc_model <- hclust(dist(df[, -10]), method = "ward.D2")

# Dendrogram çizimi
plot(hc_model, main = "Hiyerarşik Kümeleme Dendrogramı")
rect.hclust(hc_model, k = 2, border = "red")
```

---

7. Performance Analysis (Performans Analizi)

K-Means Performansı

```
# Kümeleme performansı: BSS / TSS oranı
BSS <- kmeans_model$betweenss
TSS <- kmeans_model$totss
cat("K-Means BSS/TSS Oranı: ", round(BSS / TSS * 100, 2), "%\n")
```

Logistic Regression Performansı

```
# Confusion Matrix ve Accuracy
print(conf_matrix)

# Doğruluk oranı
cat("Lojistik Regresyon Doğruluk Oranı: ",
round(conf_matrix$overall["Accuracy"], 4), "\n")
```

Pima Indians Diabetes Veri Seti Üzerinde R ile Detaylı Analiz

1. Veri Setinin Hazırlanması ve EDA

```
# Gerekli kütüphaneleri yükleyelim
library(tidyverse)
library(caret)
library(ggplot2)
library(mice)      # Eksik veri doldurma için
library(factoextra) # PCA ve clustering için

# Veri setini yükleyelim
data(PimaIndiansDiabetes, package = "mlbench")
df <- PimaIndiansDiabetes

# İlk satırları görüntüleyelim
head(df)

# Veri yapısını inceleyelim
str(df)

# Eksik veri kontrolü
summary(df)

# Sınıf dağılımını inceleyelim
table(df$diabetes)
```

Yorum:

- **8 bağımsız değişken** ve **1 hedef değişken (diabetes)** bulunur.
- `diabetes` değişkeni: `pos` (pozitif) ve `neg` (negatif).

---

2. Veri Görselleştirme

2.1. Hedef Değişkenin Dağılımı

```
# Sınıf dağılımı (pozitif/negatif)
ggplot(df, aes(x = diabetes, fill = diabetes)) +
  geom_bar() +
  ggtitle("Sınıf Dağılımı") +
  theme_minimal()
```

2.2. Özelliklerin Dağılımı

```
# Özelliklerin histogramları
df %>%
  gather(key = "Feature", value = "Value", -diabetes) %>%
  ggplot(aes(x = Value, fill = diabetes)) +
  geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
  facet_wrap(~Feature, scales = "free") +
  theme_minimal()
```

---

3. PCA (Principal Component Analysis)

PCA ile boyut indirgeme gerçekleştirip veriyi görselleştirelim.

```
# PCA uygulama
pca_model <- prcomp(df[, -9], center = TRUE, scale. = TRUE)

# PCA sonuçlarını özetleme
summary(pca_model)

# PCA görselleştirme
fviz_pca_ind(pca_model,
             geom.ind = "point",
             col.ind = df$diabetes,
             palette = c("blue", "red"),
             addEllipses = TRUE,
             legend.title = "Diabetes",
             title = "PCA: İlk İki Bileşen")
```

---

4. Eksik Veri Doldurma (Missing Data Imputation)

Eksik verileri doldurmak için **MICE** paketini kullanalım.

```
# Eksik veri kontrolü
summary(df)

# Eksik verilerin doldurulması
imputed_data <- mice(df, method = "pmm", m = 5)
df_filled <- complete(imputed_data)
```

```
# Doldurulan veriyi kontrol edelim
summary(df_filled)
```

---

5. Logistic Regression (Lojistik Regresyon)

Lojistik regresyon modeli ile diyabet tahmini yapalım.

5.1. Veri Bölme ve Model Kurulumu

```
# Veri setini eğitim ve test olarak ayıralım
set.seed(123)
train_index <- createDataPartition(df_filled$diabetes, p = 0.8, list
= FALSE)
train_data <- df_filled[train_index, ]
test_data <- df_filled[-train_index, ]

# Lojistik regresyon modeli
model <- glm(diabetes ~ ., data = train_data, family = binomial)

# Model özet bilgisi
summary(model)
```

5.2. Test Seti Üzerinde Tahmin

```
# Tahmin yapalım
pred_prob <- predict(model, test_data, type = "response")
pred <- ifelse(pred_prob > 0.5, "pos", "neg")

# Performans ölçütü: Confusion Matrix
conf_matrix <- confusionMatrix(as.factor(pred),
as.factor(test_data$diabetes))
print(conf_matrix)
```

Çıktı:

- **Accuracy**, Precision, Recall, F1 Score gibi ölçütler.

---

6. Clustering (Kümeleme)

6.1. K-Means Kümeleme

```
# K-Means modeli
set.seed(123)
kmeans_model <- kmeans(df_filled[, -9], centers = 2)

# Kümeleme görselleştirme
fviz_cluster(kmeans_model, data = df_filled[, -9],
             geom = "point", ellipse.type = "norm",
             ggtheme = theme_minimal(),
             main = "K-Means Kümeleme")
```

6.2. Hiyerarşik Kümeleme

```
# Hiyerarşik kümeleme
hc_model <- hclust(dist(df_filled[, -9]), method = "ward.D2")

# Dendrogram çizimi
plot(hc_model, main = "Hiyerarşik Kümeleme Dendrogramı")
rect.hclust(hc_model, k = 2, border = "red")
```

---

7. Performans Analizi

Logistic Regression Performansı

```
# Confusion Matrix sonuçları
print(conf_matrix)

# Doğruluk oranı
cat("Doğruluk Oranı: ", round(conf_matrix$overall["Accuracy"], 4),
"\n")
```

K-Means Performansı

```
# Kümeleme performansı: BSS / TSS oranı
BSS <- kmeans_model$betweenss
TSS <- kmeans_model$totss
cat("K-Means BSS/TSS Oranı: ", round(BSS / TSS * 100, 2), "%\n")
```

E-Commerce Customer Segmentation

1. Veri Setinin Hazırlanması ve EDA

Veri Seti Özeti:

- Özellikler: `CustomerID`, `Annual Income`, `Spending Score`, `Age`, `Gender`
- Amaç: Müşterileri harcamalarına ve gelirlerine göre gruplamak.

R Kodları:

```
# Gerekli kütüphaneleri yükleme
library(tidyverse)
library(ggplot2)
library(factoextra) # Clustering görselleştirme için
library(mice)       # Eksik veri doldurma için
library(dbscan)     # DBSCAN kümeleme için
library(cluster)    # Hiyerarşik kümeleme için

# Veri setini yükleme (örnek veri seti kullanımı)
data <-
read.csv("https://raw.githubusercontent.com/datasciencedojo/datasets
/master/Mall_Customers.csv")

# Veri setinin ilk 6 satırını gösterme
head(data)

# Veri setinin yapısı
str(data)

# Eksik veri kontrolü
summary(data)

# Temel istatistikler
summary(data[, c("Age", "Annual.Income..k..",
"Spending.Score..1.100.")])
```

---

2. Veri Görselleştirme

2.1. Özelliklerin Dağılımı

```
# Histogram: Gelir ve Harcama Puanı Dağılımı
ggplot(data, aes(x = Annual.Income..k..)) +
  geom_histogram(fill = "steelblue", bins = 20) +
  ggtitle("Yıllık Gelir Dağılımı") +
  theme_minimal()

ggplot(data, aes(x = Spending.Score..1.100.)) +
  geom_histogram(fill = "purple", bins = 20) +
  ggtitle("Harcama Puanı Dağılımı") +
  theme_minimal()
```

2.2. Gelir ve Harcama Puanı Arasındaki İlişki

```
# Scatter Plot
ggplot(data, aes(x = Annual.Income..k.., y = Spending.Score..1.100.,
color = Gender)) +
  geom_point(size = 3) +
  ggtitle("Gelir ve Harcama Puanı Arasındaki İlişki") +
  theme_minimal()
```

---

3. PCA (Principal Component Analysis)

PCA ile Boyut İndirgeme

```
# PCA uygulama
pca_data <- prcomp(data[, c("Annual.Income..k..",
"Spending.Score..1.100.")], center = TRUE, scale. = TRUE)

# PCA sonuçlarını görselleştirme
fviz_pca_ind(pca_data,
             geom.ind = "point",
             palette = "jco",
             title = "PCA: İlk İki Bileşen",
             addEllipses = TRUE)
```

---

4. Eksik Veri Doldurma (Missing Data Imputation)

```
# Eksik verileri doldurma
imputed_data <- mice(data, method = "pmm", m = 5)
data_filled <- complete(imputed_data)

# Eksik verilerin doldurulduğunu kontrol etme
summary(data_filled)
```

---

5. Clustering (Kümeleme)

5.1. K-Means Kümeleme

```
# K-Means uygulama
set.seed(123)
kmeans_model <- kmeans(data[, c("Annual.Income..k..",
"Spending.Score..1.100.")], centers = 5)

# Kümeleri görselleştirme
fviz_cluster(kmeans_model, data = data[, c("Annual.Income..k..",
"Spending.Score..1.100.")],
             geom = "point", ellipse.type = "norm",
             main = "K-Means Kümeleme")
```

5.2. DBSCAN Kümeleme

```
# DBSCAN uygulama
dbscan_model <- dbscan(data[, c("Annual.Income..k..",
"Spending.Score..1.100.")], eps = 8, minPts = 5)

# DBSCAN sonuçlarını görselleştirme
plot(data$Annual.Income..k.., data$Spending.Score..1.100., col =
dbscan_model$cluster + 1,
     pch = 19, main = "DBSCAN Kümeleme")
```

5.3. Hiyerarşik Kümeleme

```
# Hiyerarşik kümeleme
hc_model <- hclust(dist(data[, c("Annual.Income..k..",
"Spending.Score..1.100.")]), method = "ward.D2")
```

```
# Dendrogram çizimi
plot(hc_model, main = "Hiyerarşik Kümeleme Dendrogramı")
rect.hclust(hc_model, k = 5, border = "red")
```

---

6. Performans Analizi

Kümeleme Performansı (K-Means ve Hiyerarşik)

```
# K-Means BSS/TSS oranı
BSS <- kmeans_model$betweenss
TSS <- kmeans_model$totss
cat("K-Means BSS/TSS Oranı: ", round(BSS / TSS * 100, 2), "%\n")

# DBSCAN Küme Sayısı
cat("DBSCAN Küme Sayısı: ", length(unique(dbscan_model$cluster)) -
1, "\n")
```

Sonuç

Bu çalışmada:

1. **EDA** ile verinin temel özellikleri incelendi.
2. **Görselleştirme** ile gelir ve harcama puanları analiz edildi.
3. **PCA** ile boyut indirgeme yapıldı.
4. **K-Means, DBSCAN ve Hiyerarşik Kümeleme** teknikleri uygulandı.
5. **Performans Analizi** ile kümelerin anlamlılığı değerlendirildi.

Bu tarz bir örnek **büyük veri setleri** ve **karmaşık kümeleme algoritmalarının karşılaştırmasını** içerdiği için zorlayıcıdır ve gerçek dünya projeleri için uygundur.

KOD ÖZET

```
# Exploratory Data Analysis (EDA) R Kodları ve Açıklamaları

## 1. Veri Setini Tanıma

### Veri setinin genel yapısını anlama
# Veri setini yükleme
data <- read.csv("diamonds.csv")

# İlk birkaç satırı gösterir
head(data)

# Son birkaç satırı gösterir
tail(data)

# Veri setinin yapısını kontrol eder: değişken isimleri, türleri
str(data)

# Özet istatistiklerini verir: min, max, median, mean, NA sayıları vb.
summary(data)

# Veri setinin boyutunu kontrol eder: satır ve sütun sayısı
dim(data)

# Sütun isimlerini gösterir
colnames(data)

# Veri türlerini gösterir
sapply(data, class)

# Eksik veri sayısını sütun bazında kontrol eder
colSums(is.na(data))

# Benzersiz gözlem sayısını gösterir
nrow(unique(data))


## 2. Tek Değişkenli Analiz

### Sayısal Değişkenler
# Histogram: Sayısal bir değişkenin dağılımını gösterir
hist(data$price,
     main = "Histogram of Price",
     xlab = "Price",
     col = "lightblue",
     border = "black")
```

```
# Boxplot: Sayısal bir değişkenin medyanını, min-max değerlerini gösterir
boxplot(data$price,
        main = "Boxplot of Price",
        ylab = "Price",
        col = "orange")

# Ortalama, medyan ve standart sapmayı hesaplama
mean(data$price, na.rm = TRUE)   # Ortalama
median(data$price, na.rm = TRUE) # Medyan
sd(data$price, na.rm = TRUE)     # Standart sapma

# Değişkenin minimum ve maksimum değerlerini bulma
min(data$price, na.rm = TRUE)
max(data$price, na.rm = TRUE)


### Kategorik Değişkenler
# Frekans tablosu
table(data$cut)

# Kategorik değişkenin barplot'u
barplot(table(data$cut),
        main = "Barplot of Cut",
        col = "skyblue",
        xlab = "Cut",
        ylab = "Frequency")


## 3. İki Değişkenli Analiz

### Sayısal - Sayısal İlişki
# Scatter Plot: İki sayısal değişken arasındaki ilişki
plot(data$carat, data$price,
     main = "Scatter Plot of Carat vs Price",
     xlab = "Carat",
     ylab = "Price",
     col = "blue",
     pch = 19)

# Korelasyon matrisi
num_data <- data[, sapply(data, is.numeric)] # Sadece sayısal değişkenler
cor_matrix <- cor(num_data, use = "complete.obs") # Eksik veriyi göz ardı eder
print(cor_matrix)

# Korelasyon ısı haritası
library(corrplot)
corrplot(cor_matrix, method = "color", addCoef.col = "black")
```

```
### Sayısal - Kategorik İlişki
# Kategorik değişkenlere göre boxplot
boxplot(price ~ cut,
        data = data,
        main = "Boxplot of Price by Cut",
        xlab = "Cut",
        ylab = "Price",
        col = "lightgreen")


## 4. Eksik Veriyi Kontrol Etme
# Sütun bazında eksik veri sayısı
colSums(is.na(data))

# Eksik veriye sahip satırları bulma
data[!complete.cases(data), ]

# Eksik veriyi görselleştirme
library(VIM)
aggr(data, col = c("navyblue", "red"), numbers = TRUE, sortVars = TRUE, gap = 3)


## 5. Veri Dağılımını İnceleme (Pair Plot ve Density Plot)

# Pair plot: Sayısal değişkenler için
pairs(num_data, main = "Pair Plot of Numeric Variables")

# Density plot
plot(density(data$price, na.rm = TRUE),
     main = "Density Plot of Price",
     xlab = "Price",
     col = "red")


## 6. Veri Setindeki Outlier'ları Kontrol Etme

# Aykırı değerleri bulma
boxplot(data$carat,
        main = "Boxplot of Carat",
        col = "lightblue")

# Aykırı değerlerin yerlerini görme
outliers <- boxplot.stats(data$carat)$out
print(outliers)

# Aykırı değerli satırları görüntüleme
```

```
data[data$carat %in% outliers, ]
```

VISUALIZATION

```
# Data Visualization R Kodları ve Açıklamaları

## 1. Histogram: Sayısal Değişken Dağılımı
# Price değişkeni için histogram
ggplot(diamonds, aes(x = price)) +
  geom_histogram(fill = "lightblue", color = "black", bins = 30) +
  ggtitle("Histogram of Price") +
  xlab("Price") +
  ylab("Frequency")


## 2. Bar Plot: Kategorik Değişken Dağılımı
# Cut değişkeni için bar plot
ggplot(diamonds, aes(x = cut, fill = cut)) +
  geom_bar() +
  ggtitle("Barplot of Cut") +
  xlab("Cut") +
  ylab("Count")


## 3. Scatter Plot: İki Sayısal Değişken Arasındaki İlişki
# Carat ve Price arasındaki ilişki
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(alpha = 0.5, color = "blue") +
  ggtitle("Scatter Plot of Carat vs Price") +
  xlab("Carat") +
  ylab("Price")


## 4. Boxplot: Kategorik Değişkene Göre Sayısal Değişken
# Price değişkeninin Cut kategorisine göre boxplot'u
ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
  geom_boxplot() +
  ggtitle("Boxplot of Price by Cut") +
  xlab("Cut") +
  ylab("Price")
```

```r
## 5. Density Plot: Sayısal Değişken Yoğunluk Grafiği
# Price değişkeninin yoğunluk grafiği
ggplot(diamonds, aes(x = price)) +
  geom_density(fill = "red", alpha = 0.5) +
  ggtitle("Density Plot of Price") +
  xlab("Price") +
  ylab("Density")


## 6. Korelasyon Isı Haritası
# Sayısal değişkenler arasındaki korelasyonu görselleştirme
library(corrplot)
library(reshape2)

# Korelasyon matrisi
num_data <- diamonds[, sapply(diamonds, is.numeric)]
cor_matrix <- cor(num_data, use = "complete.obs")

# Korelasyon ısı haritası
melted_cor <- melt(cor_matrix)
ggplot(melted_cor, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +
  ggtitle("Correlation Heatmap") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))


## 7. Violin Plot: Kategorik ve Sayısal Değişken İlişkisi
# Price değişkeninin Cut kategorisine göre violin plot'u
ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
  geom_violin() +
  ggtitle("Violin Plot of Price by Cut") +
  xlab("Cut") +
  ylab("Price")
```

Missing Data Imputation (Eksik Veri Doldurma) R Kodları ve Açıklamaları

1. Eksik Veriyi Tespit Etme

```
# Eksik değerlerin kontrolü
colSums(is.na(diamonds)) # Sütun bazında eksik değer sayısı

# Eksik veri olan satırları görüntüleme
diamonds[!complete.cases(diamonds), ]
```

2. Basit Yöntemlerle Eksik Veri Doldurma

Ortalama ile Doldurma (Mean Imputation)

```
# Sayısal bir sütunu ortalama ile doldurma
diamonds$price[is.na(diamonds$price)] <- mean(diamonds$price, na.rm = TRUE)
```

Medyan ile Doldurma

```
# Medyan ile doldurma
diamonds$price[is.na(diamonds$price)] <- median(diamonds$price, na.rm = TRUE)
```

Mod (En Sık Tekrar Eden Değer) ile Doldurma

```
# Modu hesaplama fonksiyonu
get_mode <- function(x) {
  uniqx <- unique(x[!is.na(x)])
  uniqx[which.max(tabulate(match(x, uniqx)))]
}

# Eksik değerleri mod ile doldurma
diamonds$cut[is.na(diamonds$cut)] <- get_mode(diamonds$cut)
```

3. Gelişmiş Yöntem: MICE ile Eksik Veri Doldurma

```
# Gerekli kütüphane
library(mice)

# Eksik veri eklenmiş örnek veri seti
set.seed(123)
diamonds_missing <- diamonds
diamonds_missing[sample(1:nrow(diamonds), 50), "price"] <- NA
diamonds_missing[sample(1:nrow(diamonds), 50), "carat"] <- NA

# Eksik veri durumunu kontrol etme
colSums(is.na(diamonds_missing))
```

```
# MICE ile doldurma
imputed <- mice(diamonds_missing, m = 5, method = "pmm", seed = 123)

# Doldurulmuş veri setini tamamlama
diamonds_complete <- complete(imputed)

# Eksik değerlerin doldurulmuş halini kontrol etme
head(diamonds_complete)
colSums(is.na(diamonds_complete))
```

4. Görselleştirme: Eksik Veri Analizi

```
# Gerekli kütüphane
library(VIM)

# Eksik veri görselleştirme
aggr(diamonds_missing, col = c("navyblue", "red"),
     numbers = TRUE, sortVars = TRUE, gap = 3,
     main = "Missing Data Visualization")
```

PCA (Principal Component Analysis) R Kodları ve Açıklamaları

1. Veriyi Hazırlama ve Standardize Etme
# Sayısal değişkenleri seçme
numeric_data <- diamonds[, sapply(diamonds, is.numeric)]

# Veriyi standardize etme
scaled_data <- scale(numeric_data)

2. PCA Uygulama
# PCA uygulama
pca_result <- prcomp(scaled_data, center = TRUE, scale. = TRUE)

# PCA özet bilgisi
summary(pca_result)

# Bileşenlerin standard sapmalarını gösterme
print(pca_result$sdev)

# Varyansın grafiğini (Scree Plot) çizme
screeplot(pca_result, type = "bar", main = "Scree Plot for PCA Components")

3. Bileşenlerin Açıklama Gücü
# Her bileşenin açıkladığı varyans oranını hesaplama
explained_variance <- pca_result$sdev^2 / sum(pca_result$sdev^2)
cumulative_variance <- cumsum(explained_variance)

Grafik çizme: Kümülatif varyans oranı
plot(cumulative_variance, type = "b",
    main = "Cumulative Variance Explained",
    xlab = "Number of Principal Components",
    ylab = "Cumulative Proportion of Variance")

4. Bileşenlerin Görselleştirilmesi

Biplot: İlk iki bileşeni ve değişken yönlerini gösterme
biplot(pca_result, main = "PCA Biplot")

PCA Scatter Plot: İlk iki bileşende veri noktalarının dağılımı
# Yeni veri seti: İlk iki bileşen
pca_data <- data.frame(PC1 = pca_result$x[,1], PC2 = pca_result$x[,2])

# Scatter plot
library(ggplot2)

```
ggplot(pca_data, aes(x = PC1, y = PC2)) +
  geom_point(alpha = 0.5, color = "blue") +
  ggtitle("PCA Scatter Plot") +
  xlab("PC1") +
  ylab("PC2")
```

Logistic Regression (Lojistik Regresyon) R Kodları ve Açıklamaları

1. Veriyi Hazırlama

```
# Veri setini yükleme
data(diamonds)
library(ggplot2)

# Binary sınıf oluşturma: 'Ideal' cut sınıfını 1, diğerlerini 0 yapalım
diamonds$cut_binary <- ifelse(diamonds$cut == "Ideal", 1, 0)

# Veri setinin yapısını kontrol etme
head(diamonds)
table(diamonds$cut_binary) # Sınıf dağılımını gör
```

2. Lojistik Regresyon Modeli Oluşturma

```
# Logistic Regression modeli oluşturma
model <- glm(cut_binary ~ carat + price,
             data = diamonds,
             family = "binomial")

# Model özetini gösterme
summary(model)
```

3. Tahmin Yapma

```
# Olasılıkları tahmin etme
predicted_probs <- predict(model, type = "response")
```

```
# Olasılıkları sınıflara dönüştürme (Eşik: 0.5)
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)

# Tahminleri kontrol etme
head(predicted_classes)


## 4. Performans Değerlendirme
library(caret)

# Gerçek ve tahmin edilen değerlerle Confusion Matrix
conf_matrix <- confusionMatrix(factor(predicted_classes), factor(diamonds$cut_binary))
print(conf_matrix)

# Accuracy, Precision, Recall gibi metrikleri görüntüleme
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Pos Pred Value"]
recall <- conf_matrix$byClass["Sensitivity"]

print(paste("Accuracy:", round(accuracy, 2)))
print(paste("Precision:", round(precision, 2)))
print(paste("Recall:", round(recall, 2)))


## 5. ROC-AUC Skoru ve ROC Eğrisi
library(pROC)

# ROC-AUC hesaplama
roc_curve <- roc(diamonds$cut_binary, predicted_probs)

# ROC eğrisi
plot(roc_curve, main = "ROC Curve for Logistic Regression")
auc_score <- auc(roc_curve)
print(paste("AUC Score:", round(auc_score, 2)))
```

K-means Clustering R Kodları ve Açıklamaları

1. Veriyi Hazırlama
# Veri setini yükleme
data(diamonds)

# Sayısal değişkenleri seçme
numeric_data <- diamonds[, sapply(diamonds, is.numeric)]

# Veriyi standardize etme
scaled_data <- scale(numeric_data)

# İlk birkaç satırı kontrol etme
head(scaled_data)

2. Küme Sayısını Belirleme (Elbow Method)
# Küme sayısını belirlemek için Elbow Method
wss <- c()
for (k in 1:10) {
  kmeans_result <- kmeans(scaled_data, centers = k, nstart = 10)
  wss[k] <- kmeans_result$tot.withinss
}

Elbow grafiği
plot(1:10, wss, type = "b",
     main = "Elbow Method for Optimal K",
     xlab = "Number of Clusters (K)",
     ylab = "Within Sum of Squares")

3. K-means Kümeleme
# K-means kümeleme (örneğin k = 3)
set.seed(123)
kmeans_result <- kmeans(scaled_data, centers = 3, nstart = 10)

# Küme sonuçlarını inceleme
print(kmeans_result)

# Her gözlemin kümesini gösterme
head(kmeans_result$cluster)

4. Kümeleri Görselleştirme
library(ggplot2)

```
# PCA ile ilk iki bileşeni kullanma
pca_result <- prcomp(scaled_data)
pca_data <- data.frame(PC1 = pca_result$x[,1],
                       PC2 = pca_result$x[,2],
                       Cluster = factor(kmeans_result$cluster))

# Scatter plot: Kümelerin PCA üzerinde gösterimi
ggplot(pca_data, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point(alpha = 0.6, size = 2) +
  ggtitle("K-means Clustering Results") +
  xlab("Principal Component 1") +
  ylab("Principal Component 2") +
  theme_minimal()


## 5. Küme Merkezlerini İnceleme
# Küme merkezlerini gösterme
print(kmeans_result$centers)
```

Performance Metrics (Performans Değerlendirme) R Kodları ve Açıklamaları

1. Gerekli Kütüphaneler
```
library(caret)
library(pROC)
```

2. Veri Seti Hazırlama
```
# Örnek Veri Seti: Binary sınıf oluşturma
data(diamonds)
diamonds$cut_binary <- ifelse(diamonds$cut == "Ideal", 1, 0)

# Veriyi eğitim ve test setine ayırma
set.seed(123)
train_index <- createDataPartition(diamonds$cut_binary, p = 0.7, list = FALSE)
train_data <- diamonds[train_index, ]
test_data <- diamonds[-train_index, ]
```

3. Logistic Regression Modeli Oluşturma
```
model <- glm(cut_binary ~ carat + price, data = train_data, family = "binomial")
```

4. Test Setinde Tahmin Yapma
```
# Olasılıkları ve sınıfları tahmin etme
predicted_probs <- predict(model, test_data, type = "response")
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)
```

5. Confusion Matrix ve Temel Metrikler
```
# Confusion Matrix
conf_matrix <- confusionMatrix(factor(predicted_classes), factor(test_data$cut_binary))
print(conf_matrix)

# Performans Metrikleri
accuracy <- conf_matrix$overall["Accuracy"]
precision <- conf_matrix$byClass["Pos Pred Value"]
recall <- conf_matrix$byClass["Sensitivity"]
f1_score <- 2 * (precision * recall) / (precision + recall)

# Metrikleri yazdırma
print(paste("Accuracy:", round(accuracy, 2)))
print(paste("Precision:", round(precision, 2)))
print(paste("Recall:", round(recall, 2)))
print(paste("F1-Score:", round(f1_score, 2)))
```

6. ROC-AUC Skoru ve ROC Eğrisi
```
# ROC-AUC hesaplama
roc_curve <- roc(test_data$cut_binary, predicted_probs)
plot(roc_curve, main = "ROC Curve for Logistic Regression", col = "blue", lwd = 2)
auc_score <- auc(roc_curve)
print(paste("AUC Score:", round(auc_score, 2)))
```

R Studioda denediğim kod:

```
install.packages("tidyverse")
install.packages("ggplot2")
install.packages("factoextra")
install.packages("mice")
install.packages("dbscan")


library(tidyverse)
library(ggplot2)
library(factoextra) # Clustering görselleştirme için
library(mice)       # Eksik veri doldurma için
library(dbscan)     # DBSCAN kümeleme için
library(cluster)    # Hiyerarşik kümeleme için

# Veri setini yükleme (örnek veri seti kullanımı)
data <- read.csv("D:/dataset/Mall_Customers.csv")

# Veri setinin ilk 6 satırını gösterme
head(data)

# Veri setinin yapısı
str(data)

# Eksik veri kontrolü
summary(data)

# Temel istatistikler
summary(data[, c("Age", "Annual.Income..k..", "Spending.Score..1.100.")])

# ID sütununu kaldırma
data <- data %>% select(-CustomerID)

# Temizlenmiş veri setini görüntüleme
head(data)



# Histogram: Gelir ve Harcama Puanı Dağılımı
ggplot(data, aes(x = Annual.Income..k..)) +
  geom_histogram(fill = "steelblue", bins = 20) +
  ggtitle("Yıllık Gelir Dağılımı") +
  theme_minimal()

ggplot(data, aes(x = Spending.Score..1.100.)) +
  geom_histogram(fill = "purple", bins = 20) +
  ggtitle("Harcama Puanı Dağılımı") +
```

```
  theme_minimal()



# PCA uygulama
pca_model <- prcomp(df[, -10], center = TRUE, scale. = TRUE)

# PCA özet sonuçları
summary(pca_model)

# PCA görselleştirme
fviz_pca_ind(pca_model,
             geom.ind = "point",
             col.ind = df$Class,
             palette = c("blue", "red"),
             addEllipses = TRUE,
             legend.title = "Class",
             title = "PCA: İlk İki Bileşen")



# Elbow yöntemi ile optimum küme sayısını belirleme
fviz_nbclust(data[, c("Annual.Income..k..", "Spending.Score..1.100.")], kmeans, method =
"wss") +
  geom_vline(xintercept = 5, linetype = 2, color = "red") +
  ggtitle("Elbow Method for Optimal Clusters")


# K-Means kümeleme modeli (5 küme olarak belirleyelim)
set.seed(123)
kmeans_model <- kmeans(data[, c("Annual.Income..k..", "Spending.Score..1.100.")], centers
= 5)

# Kümeleme sonuçlarını görselleştirme
fviz_cluster(kmeans_model, data = data[, c("Annual.Income..k..", "Spending.Score..1.100.")],
             geom = "point", ellipse.type = "norm", palette = "jco", main = "K-Means Kümeleme")
# Mesafe matrisi hesaplama
distance_matrix <- dist(data[, c("Annual.Income..k..", "Spending.Score..1.100.")])

# Hiyerarşik kümeleme modeli
hc_model <- hclust(distance_matrix, method = "ward.D2")

# Dendrogramı görselleştirme
plot(hc_model, main = "Hiyerarşik Kümeleme Dendrogramı", xlab = "", sub = "")

# Küme sayısını belirtip dendrogram üzerinde gösterme
rect.hclust(hc_model, k = 5, border = "red")
```

```
# DBSCAN modeli uygulama
set.seed(123)
dbscan_model <- dbscan(data[, c("Annual.Income..k..", "Spending.Score..1.100.")], eps = 8,
minPts = 5)

# Küme sonuçlarını görselleştirme
plot(data$Annual.Income..k.., data$Spending.Score..1.100., col = dbscan_model$cluster +
1, pch = 19,
     main = "DBSCAN Kümeleme Sonuçları", xlab = "Annual Income", ylab = "Spending
Score")
```