R ile Veri Bilimi'ne Giriş

Verisetinin yüklenmesi ve verisetiyle ilgili işlemler

```
data("mtcars") #mtcars isimli r içinde bulunan bir verisetini yükledik
mtcars # verisetini görüntüleme
```

```
##                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4  8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128           32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic        30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla     33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona      21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger   15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin        15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28         13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird   19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9          27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2      26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa       30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L     15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino       19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora      15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E         21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
print("Verisetinin Boyutu:")
```

```
## [1] "Verisetinin Boyutu:"
```

```r
dim(mtcars) # verisetinin boyutu
```

```
## [1] 32 11
```

Verisetimiz 32 gözlemden(satırdan) oluşan ve 11 sütunu(niteliği) bulunan arabaların belirli özelliklerini içeren bir veriseti. Özelliklerini tam olarak anlamak için internetten araştırma yapılabilir. Veya R'ın help kısmına mtcars yazıp gelen açıklamaya bakılabilir.

Curse of Dimensionality yani "Boyut Laneti" denilen 1950'lerde ortaya koyulmuş bir kavram vardır. Bu kavrama göre boyut arttıkça karmaşıklık artar ama ama modelin başarım sonucu hakkında bir şey söylenemez. Yani daha fazla özellik ile bir veri üzerinde çalışmak daha iyi sonuç alacağımız anlamına gelmez ama karmaşıklığı artıracağı kesindir.

Bu veriseti içindeki değişkenlerin türlerini görüp veriyi anlamaya çalışalım.

```r
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

Normalde burada bütün değişkenlerin numerik değerler olduğu görülüyor. Biz bu değerler üzerine bir model kurabiliriz, hatta gayet güzel bir accuracy değeri de alırız. Fakat böyle bir model kurmak hata olur. Çünkü anlaşılacağı gibi buradaki bütün değerler sayısal değil.

Peki bir değerin sayısal olmadığını nasıl anlarız? Eğer sayısal bir nitelik içerisinde 0 sayısı mutlak yokluğu ifade ediyorsa o değişken sayısal bir değer ifade eder. Yok eğer, 0 değeri mutlak yokluğu değil de bir değeri ifade ediyorsa o zaman o değer sayısal değil kategoriktir diyebiliriz.

Hemen Örnek verelim: mtcars veriseti açıklamasında vs kısmına bakalım: vs Engine (0 = V-shaped, 1 = straight)

Burada "vs" kısmının motorun v olup olmadığını belirttiği görülüyor ve bu özellik 0 ve 1 değerlerini alıyor sadece. Peki burada 0 kısmı mutlak yokluk mu ifade ediyor yoksa bir başka değeri mi gösteriyor. Tabi ki bu motorun v olup olmadığını, yani başka bir değeri gösteriyor. Yani bu değişken sayısal değildir.

O zaman bu sütunu sayısal değerden kategorik değişkene çevirelim ki modelimiz burada nasıl çalışacağını daha iyi anlayabilsin.

```r
mtcars$cyl <- as.factor(mtcars$cyl) #kategorik (faktör) değişkene çevirme)

#şimdi diğer kategorik değişkenleri de dönüştürelim
mtcars$vs <- as.factor(mtcars$vs)
mtcars$am <- as.factor(mtcars$am)
mtcars$gear <- as.factor(mtcars$gear)
mtcars$carb <- as.factor(mtcars$carb)

str(mtcars) #şimdi tekrar özelliklere bakalım
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 2 2 2 ...
##  $ am  : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
##  $ gear: Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
##  $ carb: Factor w/ 6 levels "1","2","3","4",..: 4 4 1 1 2 1 4 2 2 4 ...
```

```r
summary(mtcars) #temel istatistiklere yeniden bakalım
```

```
##       mpg        cyl         disp             hp             drat
##  Min.   :10.40   4:11   Min.   : 71.1   Min.   : 52.0   Min.   :2.760
##  1st Qu.:15.43   6: 7   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080
##  Median :19.20   8:14   Median :196.3   Median :123.0   Median :3.695
##  Mean   :20.09          Mean   :230.7   Mean   :146.7   Mean   :3.597
##  3rd Qu.:22.80          3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920
##  Max.   :33.90          Max.   :472.0   Max.   :335.0   Max.   :4.930
##        wt             qsec       vs     am     gear   carb
##  Min.   :1.513   Min.   :14.50   0:18   0:19   3:15   1: 7
##  1st Qu.:2.581   1st Qu.:16.89   1:14   1:13   4:12   2:10
##  Median :3.325   Median :17.71                 5: 5   3: 3
##  Mean   :3.217   Mean   :17.85                        4:10
##  3rd Qu.:3.610   3rd Qu.:18.90                        6: 1
##  Max.   :5.424   Max.   :22.90                        8: 1
```

Şimdi veri içindeki değerlere nasıl ulaşabileceğimizi görelim

```r
mtcars[1,2] # 1. satır 2. eleman
```

```
## [1] 6
## Levels: 4 6 8
```

```r
mtcars[2, ] # 2. satır tüm elemanlar
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```r
mtcars["hp"] # hp sütununu getirir.
```

```
##                     hp
## Mazda RX4          110
## Mazda RX4 Wag      110
## Datsun 710          93
## Hornet 4 Drive     110
## Hornet Sportabout  175
## Valiant            105
## Duster 360         245
## Merc 240D           62
## Merc 230            95
## Merc 280           123
## Merc 280C          123
## Merc 450SE         180
## Merc 450SL         180
## Merc 450SLC        180
## Cadillac Fleetwood 205
## Lincoln Continental 215
## Chrysler Imperial  230
## Fiat 128            66
## Honda Civic         52
## Toyota Corolla      65
## Toyota Corona       97
## Dodge Challenger   150
## AMC Javelin        150
## Camaro Z28         245
## Pontiac Firebird   175
## Fiat X1-9           66
## Porsche 914-2       91
## Lotus Europa       113
## Ford Pantera L     264
## Ferrari Dino       175
## Maserati Bora      335
## Volvo 142E         109
```

```r
#is.na(mtcars) # veriseti içinde eksik değer var mı?
```

İçinde eksik veriler de olan kendi verisetimizi oluşturalım

```r
veriseti <- c(3,5,NA,7,NA)
veriseti
```

```
## [1]  3  5 NA  7 NA
```

```r
print("Toplam Eksik Veri Sayısı:")
```

```
## [1] "Toplam Eksik Veri Sayısı:"
```

```r
sum(is.na(veriseti))
```

```
## [1] 2
```

```r
print("Eksik Verilerin Konumları (İndeksleri):")
```

```
## [1] "Eksik Verilerin Konumları (İndeksleri):"
```

```r
which(is.na(veriseti))
```

```
## [1] 3 5
```

Eğer eksik verilere rağmen ortalama hesaplamak istersek

```r
mean(veriseti, na.rm = TRUE)
```

```
## [1] 5
```

Eğer matris şeklinde bir veriseti oluşturmak istersek, matrix komutunu kullanırız. Eksik değerlerin toplamını sum ile görebilirken, kolon bazında, her kolonda kaç eksik değer olduğunu görmek için colsums ifadesini kullanırız.

```r
matris <- matrix(c(1:5, NA), nrow = 2)
veri <- as.data.frame(matris)
veri
```

```
##   V1 V2 V3
## 1  1  3  5
## 2  2  4 NA
```

```r
colSums(is.na(veri))
```

```
## V1 V2 V3
##  0  0  1
```

```r
print(sum(is.na(veri)))
```

```
## [1] 1
```

Görüldüğü gibi "as.veritipi" şeklinde bir veriyi istediğimiz tipe çevirebiliyoruz. Fakat bir verisetinde çalışırken yaptığımız dönüştürmelere dikkat etmemiz gerekir. Örneğin; as.integer ve as.numeric fonksiyonları tek başına bir ifadeyle düzgün çalışırken bir veriseti üzerinde çalıştırdığımızda veriyi ascii hale çevirebilir ve veri kaybı yaşayabiliriz. Ayrıca kategorik değişkenleri numerik olarak almamak için numeric ve integer fonksiyonlarını dikkatli kullanmalıyız.

```r
a <- "12"
b1 <- as.integer(a)
b2 <- as.numeric(a)
print(a)
```

```
## [1] "12"
```

```r
print(b1)
```

```
## [1] 12
```

```r
print(b2)
```

```
## [1] 12
```

Vektör oluşturmak istediğimizde seq() fonksiyonunu kullanırız.

```r
d <- seq(from=1, to = 5, by = 0.5)
d
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Verisetinin içinden belirli özelliklere sahip verileri ayıklamak ve yeni bir veriseti oluşturmak istersek which fonksiyonundan yardım alabiliriz. Örneğin Sepal Length'i 7 den büyük ve Petal Width'i de 2.1'den büyük olan gözlemleri iris veriseti içinden alıp yeni_data isimli bir verisetine atayalım

```r
yeni_data <- iris[which(iris$Petal.Width>2.1 & iris$Sepal.Length>7),]
yeni_data
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 110          7.2         3.6          6.1         2.5 virginica
## 118          7.7         3.8          6.7         2.2 virginica
## 119          7.7         2.6          6.9         2.3 virginica
## 136          7.7         3.0          6.1         2.3 virginica
```

VERİSETİNDEKİ EKSİK GÖZLEMLERİ DOLDURMANIN YÖNTEMLERİ

1) DOĞRUSAL İNTERPOLASYON YÖNTEMİ: Bir doğru üzerindeki eksik bir noktayı bulmaya çalışır. Dolayısıyla formülü nokta formülüne benzerdir. Verisetinde doğrusallık varsa bir başka değişkendeki doğrusal artışın kendi değişkenimiz üzerindeki etkisinden faydalanarak eksik değerleri bulabiliriz.

2) MAKSİMUM BEKLENTİ YÖNTEMİ: İlk olarak belirlenen bir ratgele değer üzerinden belirlenen hassasiyete ulaşılana kadar, aritmetik ortalamanın rastgele değerden farkı yine aritmetik ortalamaya eklenerek devam edilir. Ortalamanın belirlenen değerden farkı hassasiyet değerinden küçük olana kadar devam edilir ve belirlenen değere ulaşıldığında bütün eksik gözlemler o değerle doldurulur.

3) JACKKNIFE YÖNTEMİ: Maksimum beklentiden farklı olarak bütün eksik değerlerin aynı sonuçla doldurulması yerine, her eksik değer için ayrı hesaplama yapılmasıdır. Bir kere hesaplanan değer artık bilinen değer olarak kabul edilip hesalamada bu değer bilinenler arsında kullanılır.

4) MODELLEME YÖNTEMİ: Belirli bir model belirlenerek bu model ile eksik değerler tahmin edilir.

5) EKSİK VERİYİ SİLME: Bu yöntemde eksik veriler silinir. Fakat veriseti küçükse bu yöntemi tercih etmek verisetini daha da küçülteceğinden modelin başarısını etkileyecektir.

Eksik Verilerle Çalışmak için Kullanacağımız Kütüphaneler

```r
library(VIM)
```

```
## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
library(missForest)
```

```
## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

## Loading required package: foreach

## Loading required package: itertools

## Loading required package: iterators

##
## Attaching package: 'missForest'

## The following object is masked from 'package:VIM':
##
##     nrmse
```

```r
library(mice)
```

```
##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
library(ISLR) # İÇİNDEN BASKETBOL VERİSETİNİ KULLANACAĞIZ
library(Hmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:Hmisc':
##
##     impute
```

Yukarıdaki Paketlerin yaptığı işleri şöyle sıralayabliriz.

1)Mice Paketi: (Multivariate Imputation via Chained Equations)

Regresyon tabanlı olarak eksik verileri tahmin eder. Sürekli değişkenler için lineer regresyon kategorik değişkenler için ise logistic resression kullanılır. Paketteki Yöntemler:

- PMM (Predictive Mean Matching): Sayısal değişkenler için kullanılır.
- logreg (Logistic Regression): İkili (binary) değişkenler için kullanılır.
- ploγreg (Bayesian Polynomial Regression): 2 ya da daha fazla faktör değişkenler için.
- Proportional Odds Model: İki ya da daha fazla sıralı değişkenler için.

2)missForest Paketi:

Adından da anlaşılacağı gibi eksik verileri random forest yöntemiyle doldurmayı sağlar. Bu paketi lineer olmayan bir yöntem kullanmak istediğimizde kullanabiliriz. Non-parametric tahminleme yönetemiyle çalışır, yani verinin normal dağılmış olması veya 30 gözlemden büyük olması gerekmez, dolayısıyla daha özgür çalışırız. Kullanacağımız fonksiyon paketadı ile aynı olan missForest fonksiyonudur.

3) VIM Paketi:

Özellikle eksik verilerin görselleştirilmesi için kullanılır. aggr ve barMiss fonksiyonları eksik veriyi görselleştirmek için kullanılır.

4) Hmisc Paketi:

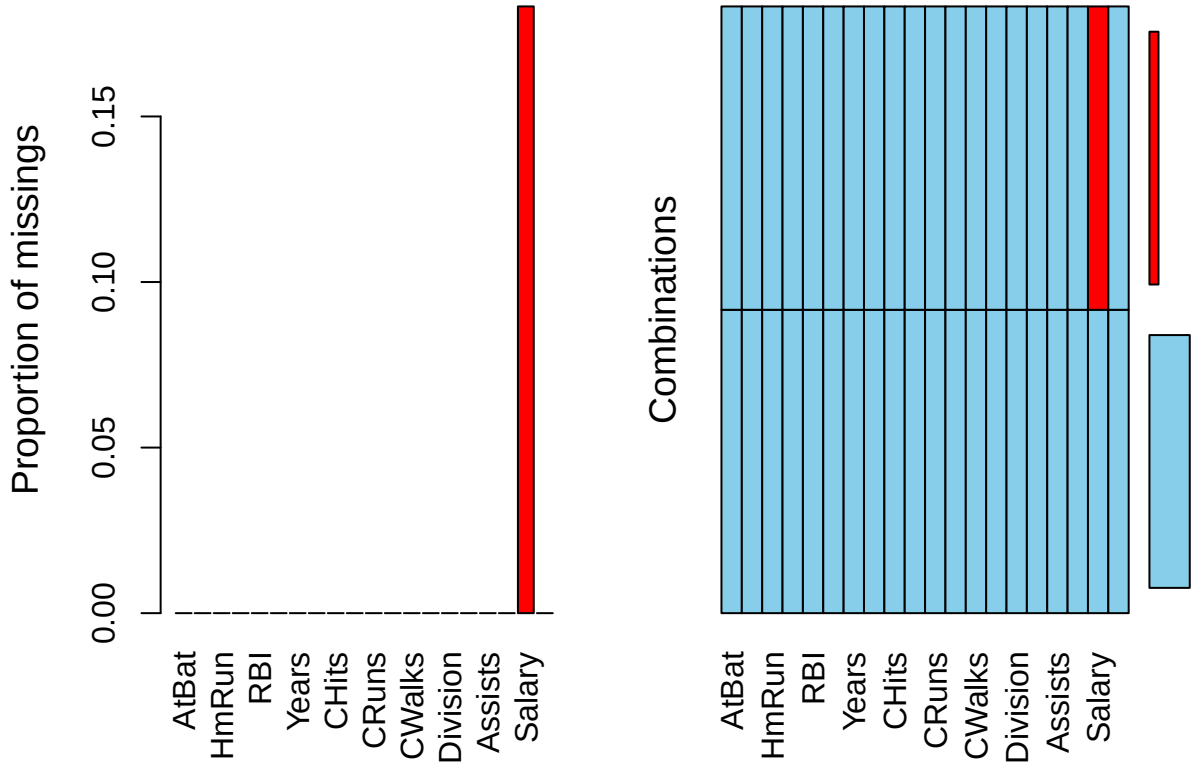
Bu paket daha tek bir alana yönelmek yerine daha çoklu bir kullanım amacıyla sunulmuştur. Veri manipülasyonu, görselleştirmesi, eksik veri doldurma ve modelleme için birçok fonksiyon içerir. help("hmisc") yazarak özellik ve fonksiyonlara ulaşılabilir.

5)e1071 Paketi:

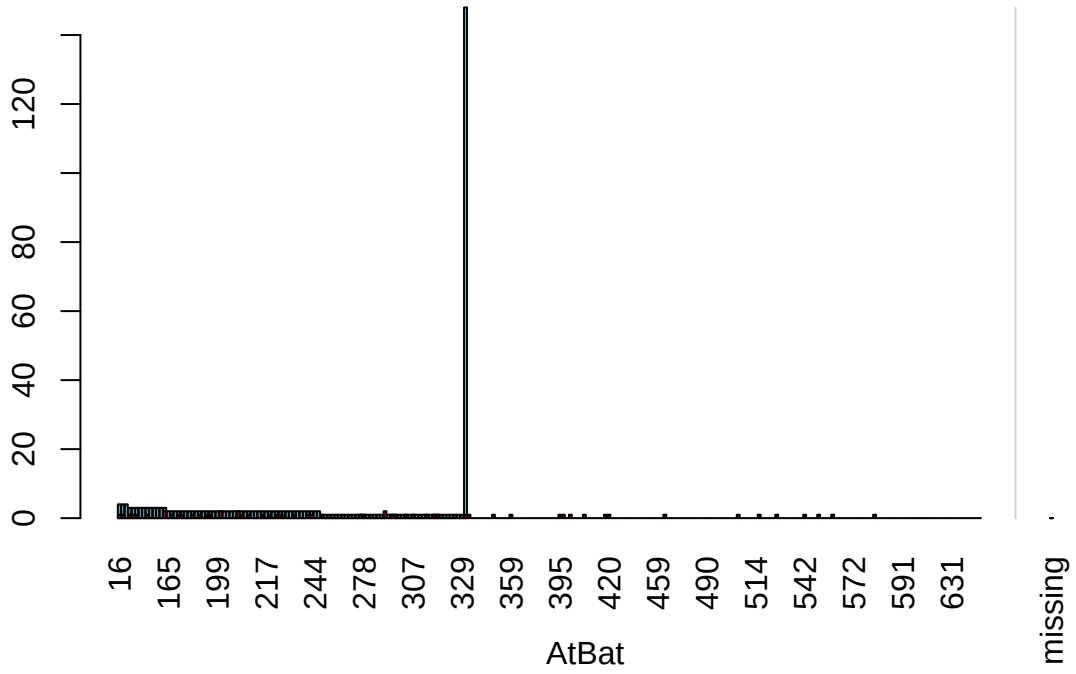
Bu paket SVM algoritması ile ilgili uygulamaları içerir. Eksik verileri SVM algoritması kullanarak doldurmak istiyorsak bu paketi kullanırız.

Eksik verileri görselleştirmek için VIM kütüphanesini kullanabiliriz. Bu kütüphane içinde veri doldurmak için de yöntemler vardır.

```
veri <- Hitters
aggr(veri)
```



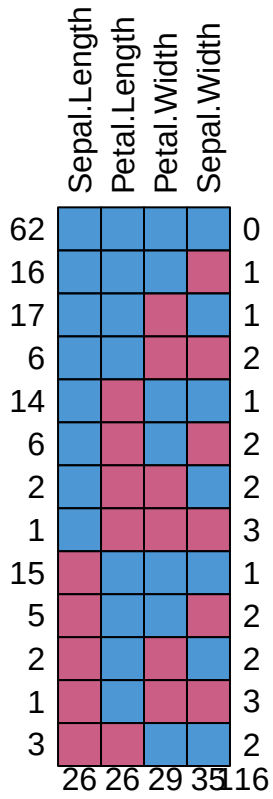

```
barMiss(veri)
```



Şimdi kendimiz eksik veriler oluşturup gözlemleyelim. İris verisetini kullanacağız.

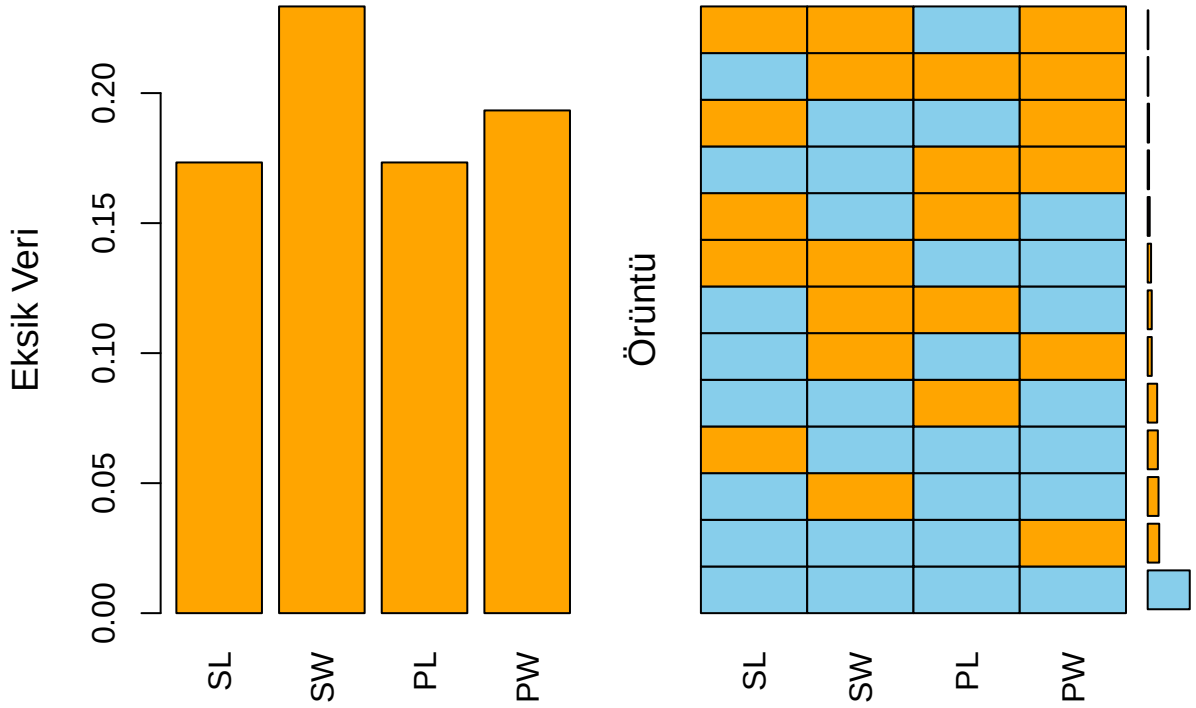
Bunun için missforest içinden eksik veri oluşturmaya yarayan prodNa isimli bir fonksiyon kullanacağız. Daha sonra mice kütüphanesinden md.pattern fonksiyonu ile eksik verileri görselleştireceğiz.

```
missedData <- prodNA(iris, noNA = 0.2) #0.2 oranında rasgele eksik gözlem oluşturacak

missedData <- subset(missedData, select = -c(Species)) #Tür değişkenini çıkardık.

md.pattern(missedData, rotate.names = TRUE) #Eksikleri Bulma ve Görselleştirme
```

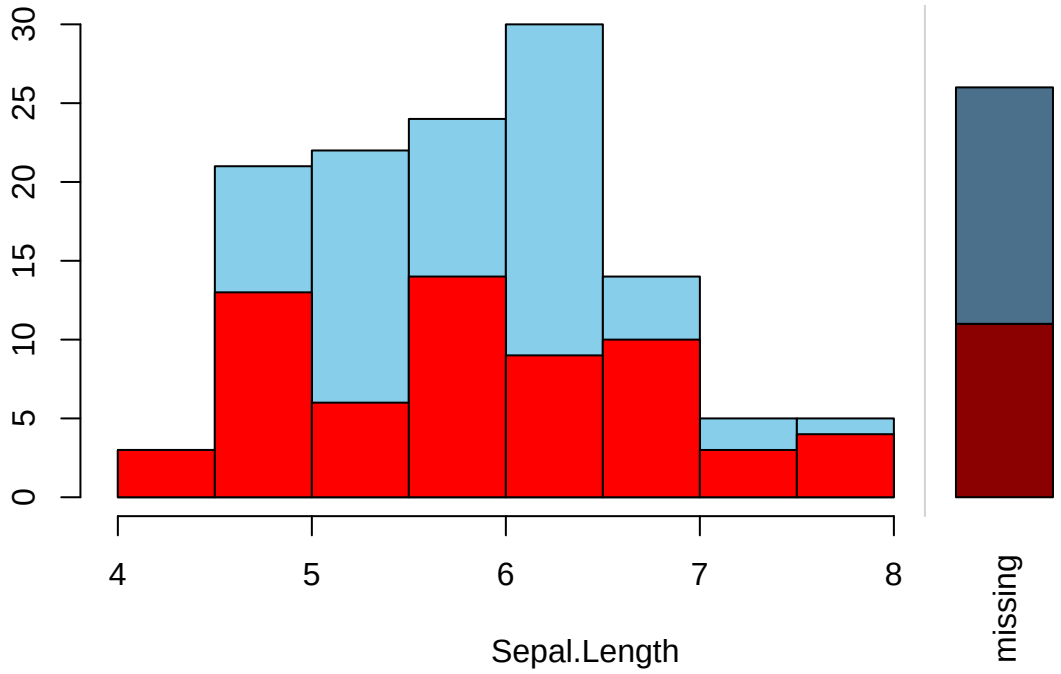



```
##    Sepal.Length Petal.Length Petal.Width Sepal.Width
## 62            1            1           1           1   0
## 16            1            1           1           0   1
## 17            1            1           0           1   1
## 6             1            1           0           0   2
## 14            1            0           1           1   1
## 6             1            0           1           0   2
## 2             1            0           0           1   2
## 1             1            0           0           0   3
## 15            0            1           1           1   1
## 5             0            1           1           0   2
## 2             0            1           0           1   2
## 1             0            1           0           0   3
## 3             0            0           1           1   2
##              26           26          29          35 116
```

```
#Bir de VIM ile görselleştirelim ek olarak.
mice_plot <- aggr(missedData, labels=c("SL", "SW", "PL", "PW"), gap=3,
                  ylab=c("Eksik Veri", "Örüntü"), col=c("skyblue", "orange"))
```




```
barMiss(missedData)
```




```
mice_plot
```

```
##
##  Missings in variables:
##       Variable Count
##  Sepal.Length    26
##   Sepal.Width    35
##  Petal.Length    26
##   Petal.Width    29
```

EKSİK VERİYİ PMM (PREDICTIVE MEAN METHOD) İLE DOLDURMA

```
predictedData <- mice(missedData, m = 5, method = "pmm", maxit = 5)
```

```
##
##  iter imp variable
##    1   1  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##    1   2  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##    1   3  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##    1   4  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##    1   5  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##    2   1  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##    2   2  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
```

```
##   2   3  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   2   4  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   2   5  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   3   1  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   3   2  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   3   3  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   3   4  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   3   5  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   4   1  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   4   2  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   4   3  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   4   4  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   4   5  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   5   1  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   5   2  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   5   3  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   5   4  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
##   5   5  Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
```

```
summary(predictedData) # özetleyelim
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##        "pmm"        "pmm"        "pmm"        "pmm"
## PredictorMatrix:
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length            0           1            1           1
## Sepal.Width             1           0            1           1
## Petal.Length            1           1            0           1
## Petal.Width             1           1            1           0
```

```
# 5 iterasyon yaptıktan sonra 5 farklı veri tamamlama alneratifi oluştu.
# Biz ise 3. iterasyona göre veriyi tamamlamayı seçtik.
irisPredicted <- complete(predictedData,3)
```

Orjinal ve tahmin edilen değerleri çizdirelim. Kırmızı noktalar tahmin değerleri iken çarpı işaretleri orjinal noktaları gösteriyor.

```
library(ggplot2)
ggplot(iris, aes(Sepal.Length, Sepal.Width)) + geom_point(shape=4,  size=4) + geom_point(aes(irisPredic
```

KORELASYON

Korelasyon iki değişken arasındaki doğrusal ilişkidir. Aynı yönlü veya ters yönlü olabilir ve dolayısıyla -1 ve +1 arasında değişir. (0 noktası ilişkinin olmadığı durumu gösterir). Fakat kesin bir şekilde belitmek gerekir ki; KORELASYON NEDENSELLİK DEĞİLDİR! Yani iki veya daha fazla değişken arasında bir ilişki olması bunun belirli bir nedeni olduğunu göstermez.

Peki korelasyonu nasıl kullanırız?

Eğer verisetimizde bağımsız değişkenlerden bazıları arasında güçlü bir korelasyon varsa (+1 veya -1'e yakın) bu değişkenlerden sadece birini modelimizde kullanmamız yetecek demektir.

Eğer bağımlı değişkenimiz ile bazı bağımsız değişkenler arasında güçlü bir korelasyon varsa burada çok daha dikkatli olmak gerekir. Çünkü bağımlı değişken ile güçlü lorelasyona sahip olan bağımsız değişkenimiz, model kurulduğunda diğer değişkenleri baskılayacak ve böylece tek tahmin edici kendisiymiş gibi hareket edecektir. Bu da modelde bias hatasına (yanlılık sorunu) sebep olacaktır. Bu tür bağımsız değişkenleri modelde kullanmamak gerekir.

Farklı hesaplanan korelasyon türleri vardır. Burada default olarak kullanaılanlar pearson korelasyonu üzerinden hesaplanmıştır.

CORRPLOT PAKETİ

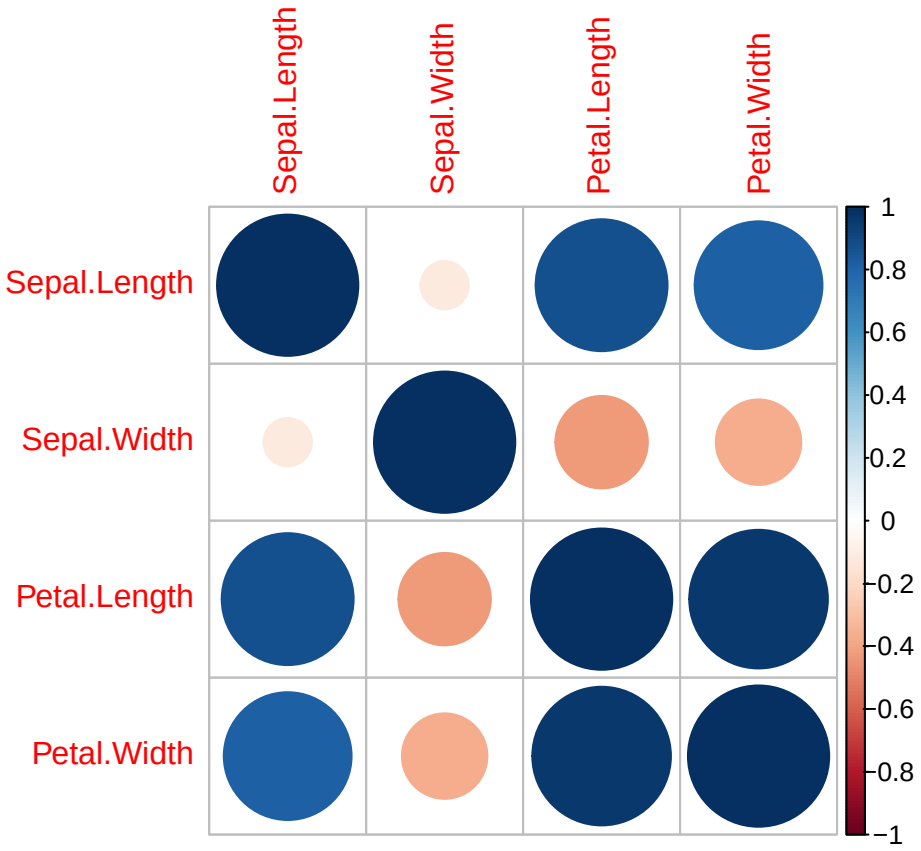
```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

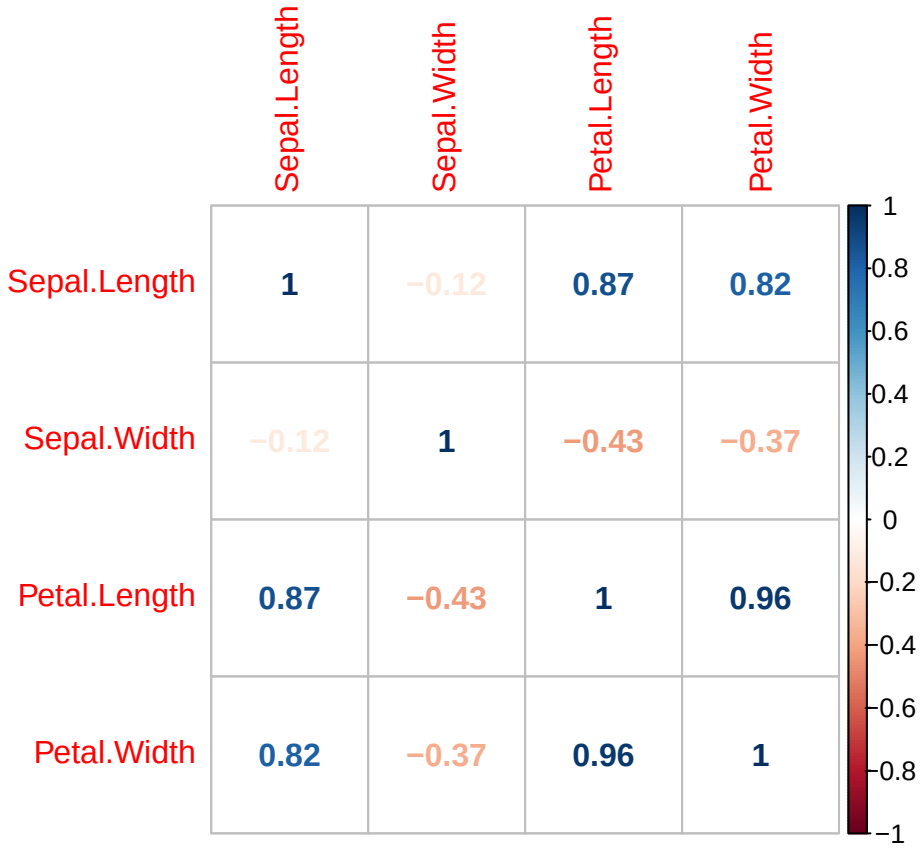
```r
data <- subset(x = iris, select = -c(Species))

(KOR <- cor(data)) # parantez içinde yazılan ifade ekrana da yazdırılır.
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

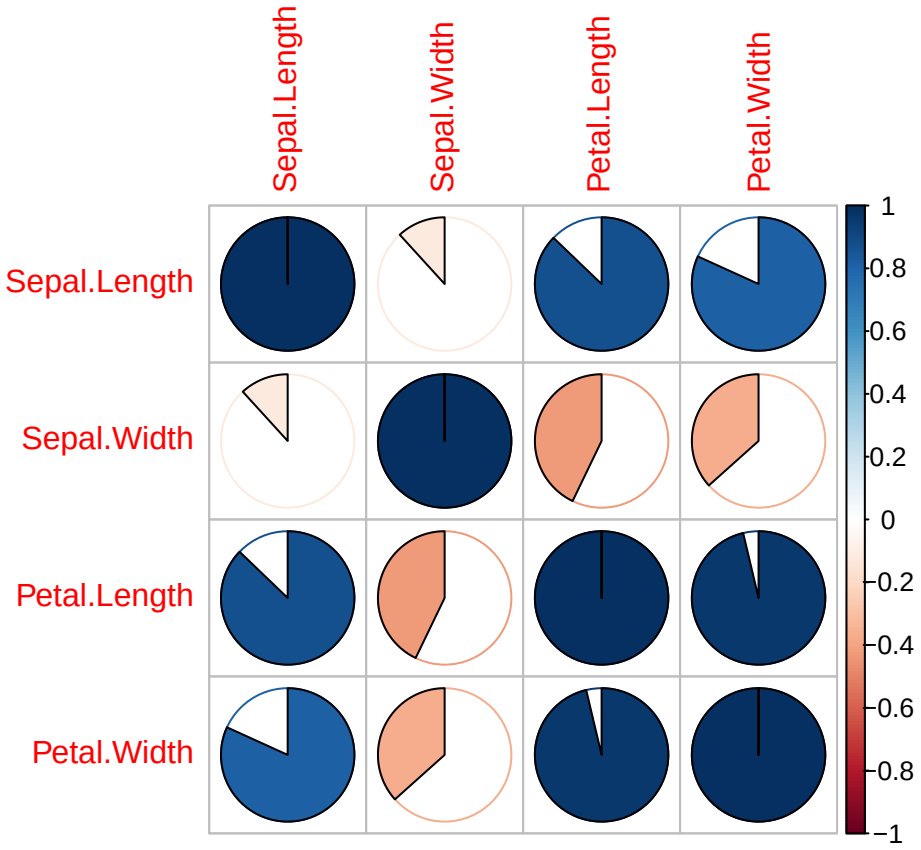
```r
#grafiğer dökerek gösterirsek daha anlaşılır olur.

#3 farklı görünümde grafik çizelim.
corrplot(KOR, method = "circle")
```




```r
corrplot(KOR, method = "number")
```




```
corrplot(KOR, method = "pie")
```



Görüldüğü gibi Petal Width ve Petal Length arasında pozitif yönlü güçlü bir ilişki var. Aynı zamanda Petal Length ile Sepal Length arasında da güçlü bir ilişki var. O zaman bu üçünden sadece birini (Petal Length'i) kullanmak yeterlidir. Hatta kısa ve basit bir modelle deneyelim.

```
model_tam <- lm(as.integer(Species)~., data = iris, method = "qr")
yeni_data <- subset(iris, select = c(-Petal.Width, -Sepal.Length))
model_az_veri <- lm(as.integer(Species)~., data = yeni_data, method = "qr")

tam_tahmin <- predict(model_tam, iris[,1:4])
az_veriyle_tahmin <- predict(model_az_veri, yeni_data[,1:2])
```

Yukarıdaki verilerle yapılan tahmin tamsayı sonuç değil noktalı değerler döndürecek. Biz bu verileri tamsayı yaparak sonuçlara bakalım

Aşağıda for döngüsüyle belirli aralıklardaki değerleri yakın olduğu tamsayıya yuvarladık. R içinde bunu çok daha kısa yollarla yapabilirsiniz ve çok gerekmedikçe for döngüsü kullanmak mantıklı değildir. Ama sadece örnek amaçlı olduğundan yapıyoruz şimdilik.

```
a <-1
for (x in tam_tahmin) {
    if (x>0 && x<=1.5) {
        tam_tahmin[a] = 1
    }
    if (x>1.5 && x<=2.5) {
        tam_tahmin[a] = 2
    }
```

```
    if (x>2.5){
        tam_tahmin[a] = 3
    }
    a = a + 1
}

a<-1
for (x in az_veriyle_tahmin) {
    if (x>0 && x<=1.5) {
        az_veriyle_tahmin[a] = 1
    }
    if (x>1.5 && x<=2.5) {
        az_veriyle_tahmin[a] = 2
    }
    if (x>2.5){
        az_veriyle_tahmin[a] = 3
    }
    a = a + 1
}
yarım_verili_model_hatası <- 0
tam_verili_model_hatası <- 0
sıra_no <- 1
for (i in as.integer(iris$Species)) {
    if (i != az_veriyle_tahmin[sıra_no]) {
        yarım_verili_model_hatası = yarım_verili_model_hatası + 1
    }
    if (i != tam_tahmin[sıra_no]) {
        tam_verili_model_hatası = tam_verili_model_hatası + 1
    }
    sıra_no = sıra_no + 1
}

print("Tam verili model MSE:")
```

```
## [1] "Tam verili model MSE:"
```

```
ModelMetrics::mse(as.integer(iris$Species), tam_tahmin)
```

```
## [1] 0.02666667
```

```
print("Tam verili model hata sayısı:")
```

```
## [1] "Tam verili model hata sayısı:"
```

```
print(tam_verili_model_hatası)
```

```
## [1] 4
```

```
print("Yarım verili model MSE:")
```

```
## [1] "Yarım verili model MSE:"
```

```
ModelMetrics::mse(as.integer(iris$Species), az_veriyle_tahmin)
```

```
## [1] 0.04
```

```
print("Yarım verili model hata sayısı:")
```

```
## [1] "Yarım verili model hata sayısı:"
```

```
print(yarım_verili_model_hatası)
```

```
## [1] 6
```

Gördüğümüz gibi iki sütünu kaldırmamıza rağmen sadece 2 tane daha fazla hata yaptık. Yani aslında verinin yarısıyla aynı sonucu verebilecek bir model kurabildik. Şimdi burada bahsetmemiz gereken konu boyutsallık lanetidir.

CURSE OF DIMENSINALITY (BOYUTSALLIK LANETİ)

Daha önce de dediğimiz gibi (en başta) boyut arttıkça karmaşıklık artar ama model başarısı için aynı şey söylenemez. İşte yukarıda verinin yarısını attığımız halde neredeyse aynı başarıyı almamızı sağlayan şey de boyutsallık laneti dediğimiz kavramla çok benzerdir.

Yukarıda yaptığımız gibi verinin daha az kısmıyla da aynı başarı yakalanabilir. Hatta bazen başarının artması bile söz konusu olabilir. Bu yüzden bütün veriyi alarak çalışıp modeli karmaşıklaştırmak yerine boyut azaltma yöntemlleri kullanarak daha basit, daha açıklanabilir modeller kurmak çok daha mantıklıdır.

Boyut azaltmak için çeşitli yöntemler bulunmaktadır. Bunlardan bazıları:

-PCA (Principal Component Analysis) - Temel Bileşenler Analizi -LDA (Linear Discriminant Analysis) - SWD -MARS

Bizim burada açıklayacağımız yöntem PCA yöntemidir.

PCA (Principal Component Analysis) - Temel Bileşenler Analizi

Principal Component Analysis verisetinin kendi değişkenleri ile ifade edilmesinin yerine, bu değişkenleri kullanarak kendi değişkenlerini (PC) oluşturur. Yani kendisi yeni bir boyut tanımlayarak veriyi bu boyutta ifade eder. Bunu yaparken de veriseti içinde veriye en çok katkı yapanları bir noktada toplayarak çalışır. PCA özvektör ve özdeğerler üzerinden hesaplama yaparak sonuç hesaplar.

Büyük bir verisetindeki en önemli değişkenler hangileridir?

- Bileşenlerin oluşturulması (PC)
- Yüksek boyutlardaki verisetinin mümkün olan en fazla enformasyonu içerecek şekilde daha düşük bir boyutta incelenmesi
- Daha az bileşenle daha fazla anlama sahip görselleştirmenin yapılması
- 3 veya daha fazla boyutlu verisetlerinde uygulanması daha iyi sonuçlar elde edilmesini sağlayacaktır.

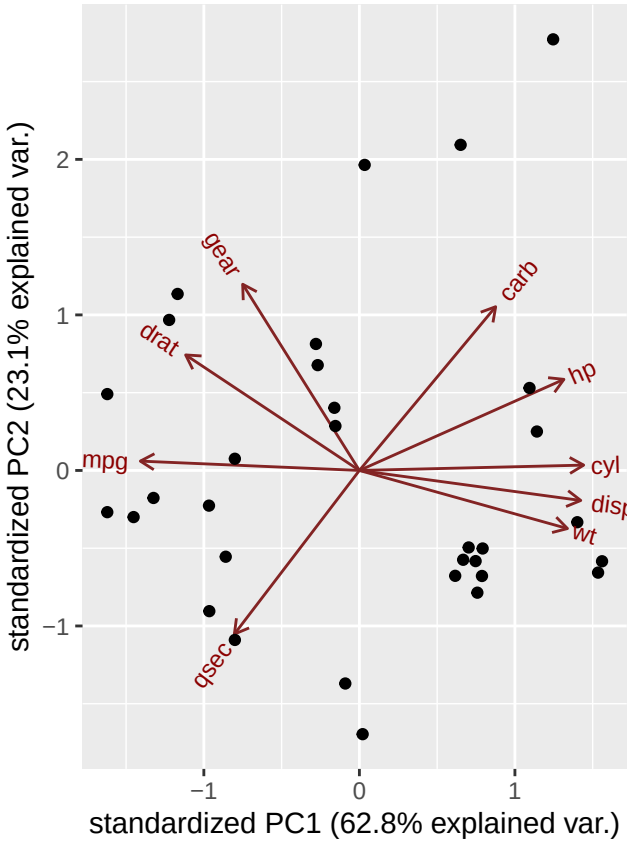
```
#library("devtools")
#install_github("vqv/ggbiplot")
library("ggbiplot")
```

```
## Loading required package: plyr

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize

## Loading required package: scales
```

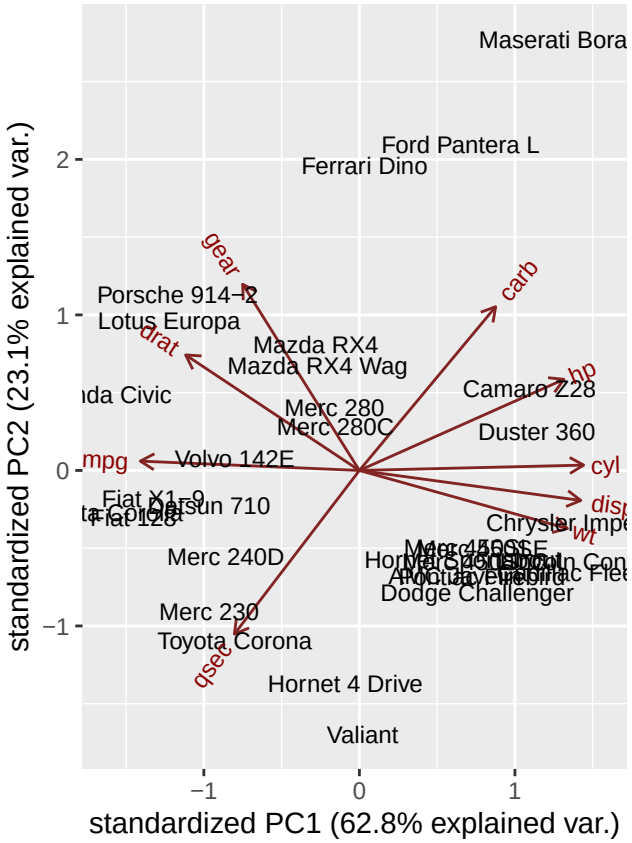
```
rm(mtcars)
data("mtcars")
pca_info <- prcomp(mtcars[, c(1:7, 10,11)], center = TRUE, scale. = TRUE)
summary(pca_info)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.3782 1.4429 0.71008 0.51481 0.42797 0.35184 0.32413
## Proportion of Variance 0.6284 0.2313 0.05602 0.02945 0.02035 0.01375 0.01167
## Cumulative Proportion  0.6284 0.8598 0.91581 0.94525 0.96560 0.97936 0.99103
##                           PC8     PC9
## Standard deviation     0.2419 0.14896
## Proportion of Variance 0.0065 0.00247
## Cumulative Proportion  0.9975 1.00000
```

```
ggbiplot(pca_info)
```




```
ggbiplot(pca_info, labels = rownames(mtcars))
```



PCA Analizi yapıldığında 2 tane bileşen oluşturuldu. Bunlardan PC1 verisetindeki varyansın %62.8'ini açıklarken, PC2 %23.1'ini açıklıyor. Toplamda bütün varyansın %80'inden fazlası bu iki bileşenle açıklanabiliyor. Verisetindeki özelliklerin bu yeni bileşen boyutları üzerindeki yön ve değerleri de grafikteki gibi oluşmuş.

PCA yönteminin matematiksel açıklaması internette bulunabilir. Ama grafikte gördüğümüz veriseti özelliklerinin PC1 ve PC2 uzayı üzerindeki dağılımının özvektörlerle ve özdeğerlerle oluştuğunu anlayabiliyoruz.

UNSUPERVISED LEARNING (DENETİMSİZ VEYA GÖZETİMSİZ ÖĞRENME)

Denetimli öğrenme algoritmalarında veriden bir model kurulduğunda o verideki özellikler ile tahmin edilen gözlemin gerçek sonucu da verimizde etkiket halinde bulunur. Biz verimiz üzerinde model kurarken aslında modelin hangi sonuçlara ulaşması gerektiğini, hangi tahminleri yapması gerektiğini biliriz. Çünkü gerçekte gözlenen değeri de biliyoruzdur.

Fakat bir verisetinden ne öğreneceğimizi bilmiyorsak ve bazı yöntemler ile yine de veriyi modellemek stiyorsak ne yaparız? İşte o zaman da denetimsiz öğrenme yöntemleri kullanılır. Denetimsiz öğrenmede bağımlı-bağımsız değişken ayrımı yoktur, verinin tamamından bir sonuca ulaşılmaya çalışır.

Örneğin, bir marketin müşterileri belirli segmentlere ayrılıp buna göre pazarlama yöntemleri geliştirilmek isteniyor olsun. Burada model kuracak olan kişi önceden belirlenmiş bir segmentasyon verisine sahip değildir. Bu yüzden müşterileri otomatik olarak kümeleyip ayırabilen bir model kurmak zorundadır.

En sık kullanılan unsupervised learning yöntemi kümeleme algoritmalarıdır. Kümeleme birbirine benzeyen veya farklılaşan verileri bir araya toplayıp diğerlerinden ayırarak, belirli parçalar (kümeler) oluşturma işlemidir.

Veriden bahsettiğimizde aslında her zaman bir veri uzayından bir çok boyutlu uzaydan bahsederiz. Dolayısıyla bu uzaydaki veriler de birbirine göre uzaklıklarına göre kümelenebilir. Kümeleme algoritmaları

da bu uzaklıklardan yararlanır. Dolayısıyla burada en önemli parametre hangi uzaklık ölçüm yöntemini kullanacağını belirlemektir. Öklid, Manhattan, Minkowski vb yöntemler kullanılabilir.

Kümeleme algoritmalarında her kümedeki elemanların o kümenin merkezine olan uzaklıkları hesaplanarak bir küme belirlenmeye çalışılır. Kaç tane küme oluşturulacağına ise WCSS (Within cluster sum of squares) gibi bazı değerlere bakılarak karar verilmeye çalışılır.

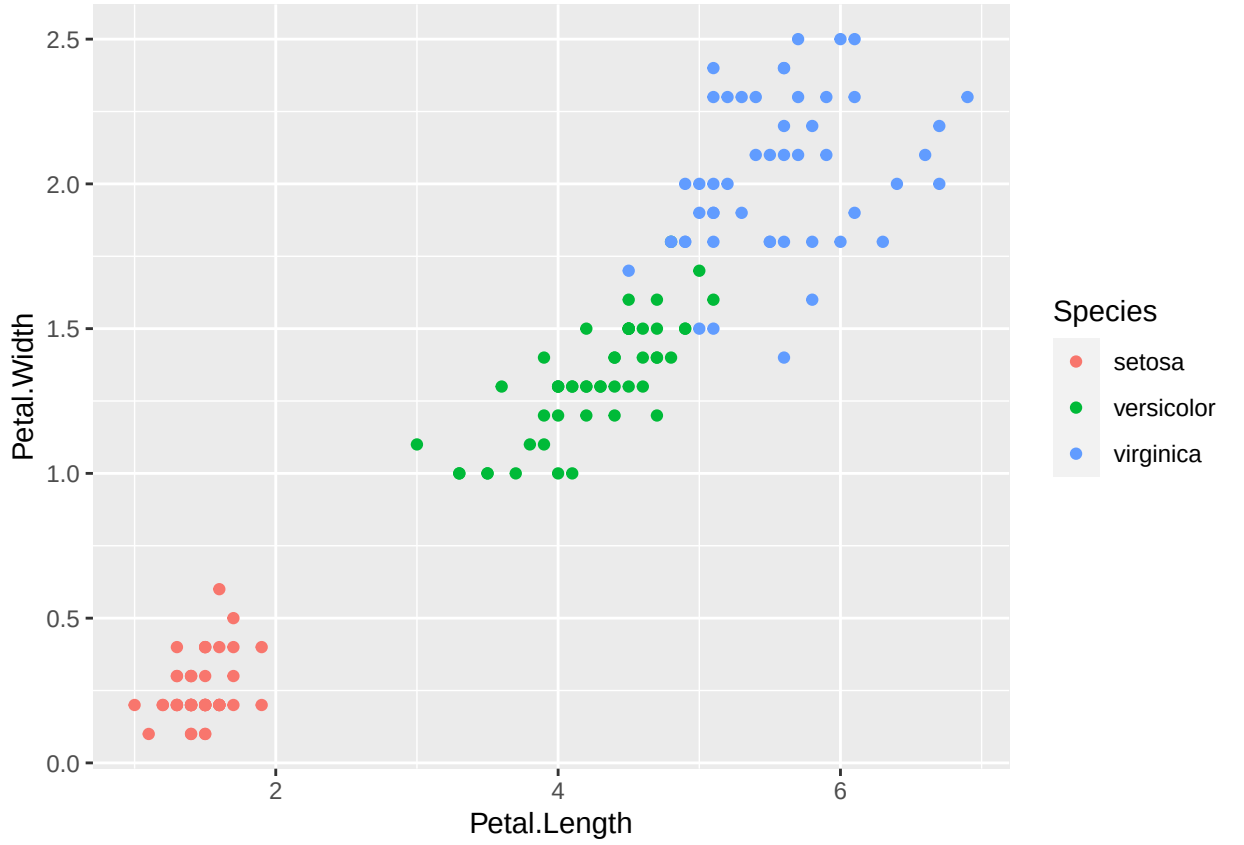
3 Tür Kümeleme Modeli Oluşturulabilir:

- Bağlantı Tabanlı Modeller: Farklı grup üyeleri arasındaki farklılığa bağlı olarak yapılan hiyerarşik kümeleme yöntemidir. (SLINK Modeli)
- Yoğunluk Tabanlı Modeller: Bir noktanın kendisi ile olan farklılıkları belli bir değerden az olan belli bir sayıdaki diğer noktalar ile çevrelenmesine bağlı olarak yapılan kümeleme/gruplama türüdür. (DBSCAN Modeli)
- Centroid Tabanlı Modeller: Her küme o kümenin merkezi durumundaki tek bir nokta ile ifade edilir. (K-MEANS Modeli)

Genel olarak istediğimiz şey küme elemanları arası benzerlik en yüksek (mesafe en düşük), kümeler arası benzerlik en düşük veya farklılık en yüksek (mesafe en yüksek) olmalıdır.

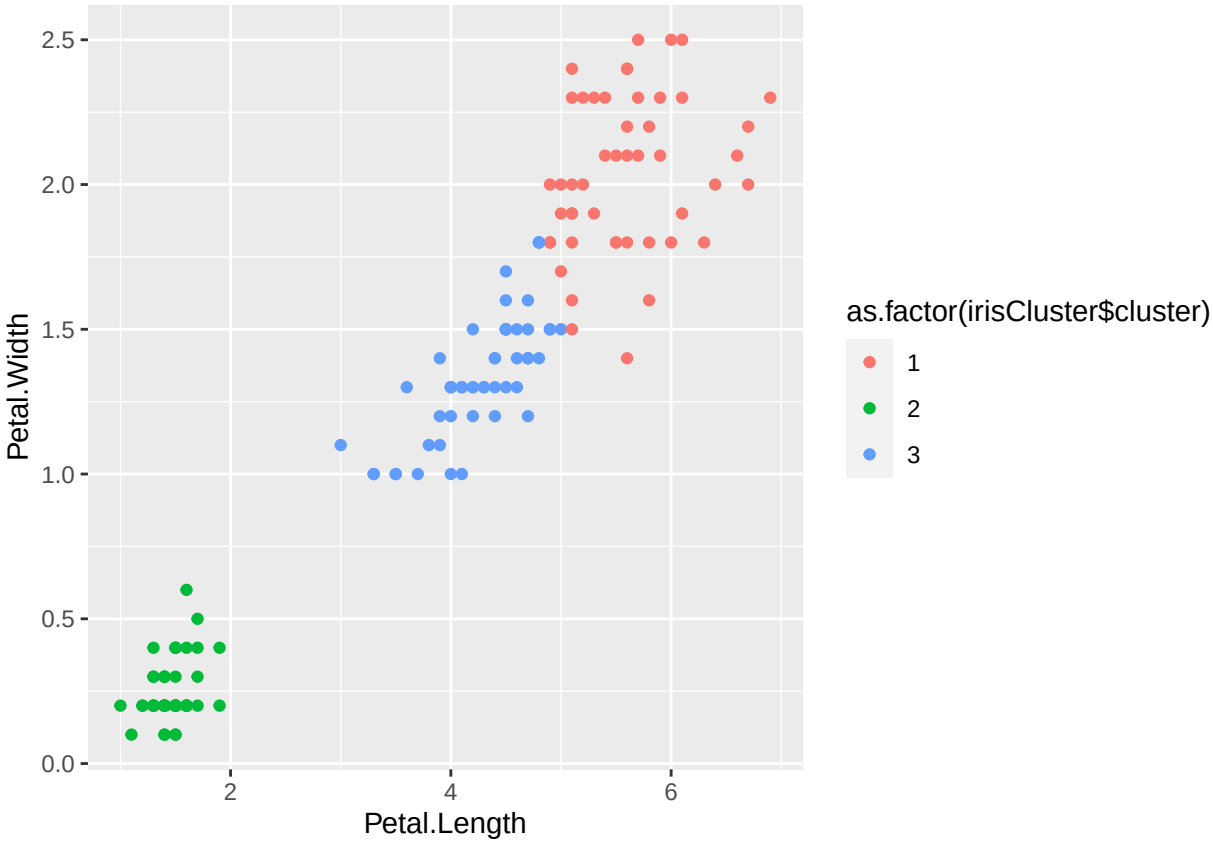
İris verisini gerçekte nasıl kümelendiğini görelim

```
ggplot(iris, aes(Petal.Length, Petal.Width, color=Species)) + geom_point()
```



Şimdi ise kendimiz bir model ile algoritma oluşturup nasıl kümelendiğine bakalım

```
irisCluster <- kmeans(iris[,3:4], centers = 3, nstart = 20)
ggplot(iris, aes(Petal.Length, Petal.Width, color=as.factor(irisCluster$cluster))) + geom_point()
```



Aşağı yukarı benzer bir kümeleme yapıldığını görebiliyoruz. Bir tablo oluşturup hatalara bakalım

```
table(irisCluster$cluster, iris$Species)
```

```
##
##     setosa versicolor virginica
##   1      0          2        46
##   2     50          0         0
##   3      0         48         4
```

Hesaplamayı farklı yapmasından dolayı asal köşegen üzerinde görememiş olsak bile aslında anlaşılıyor ki toplamda 6 hatalı gözlem tahminimiz olmuş. Grafikteki benzerlik de görülüyordu zaten.

SUPRVISED LEARNING (GÖZETİMLİ/DENETİMLİ ÖĞRENME) ALGORİTMALARI

Yukarıda anlattığımız k-means yönteminde bu sefer verisetindeki sınıfların önceden etiketlenmiş olduğunu, yani, sınıfları artık bildiğimizi düşünelim. Yani iris verisetindeki (orjinaldeki) hangi gözlemin hangi türe ait olduğunu biliyorsam artık gözetimli bir algoritma uygulayabilirim demektir.

Bunun için knn algoritmasını kullanacağız. Fakat modelin nasıl çalıştığını test etmek için bu sefer test ve train olarak ayırmam gerekecek.

Verisetini 3 şekilde ayırabiliriz:

- Hold Out Yöntemi (Ayırıp Tutma)
- K-FOLD Yöntemi (K Katlı Çarpraz Doğrulama)
- LOOCV - Leave One Out Yöntemi

En sık karşılaşılan yöntem hold out olmasına rağmen tavsiye edilmez. Fakat kolay olması açısından biz de burada bunu kullanacağız.

KNN (K NEAREST NEIGHBORHOOD) / K (SAYIDA) EN YAKIN KOMŞU ALGORİTMASI

KNN algoritması k–means gibi bir kümeleme algoritmasıdır.(aslında sınıflandırma algoritmasıdır ama şimdilik pek fark yok denilebilir.) Fakat gözetimli bir algoritma olarak çalışır. Algoritma gözetimli olduğu için verisetindeki kümeler bilinir. Dolayısıyla kümeleri kendisi bulmak yerine şunu yapar; Yeni bir gözlem geldiğinde önceden gördüğü ve öğrendiği verisetinde (uzayda) yeni gelen noktaya en yakın k sayıda noktaya bakar. Baktığı k sayıda nokta çoğunlukla hangi kümeye aitse veya yeni nokta hangilerine daha yakınsa, yeni gelen gözlemi de o kümedendir diye tahmin eder. Tıpkı k-means gibi burada da mesafe ölçmek için kullanılacak yöntem çok önemlidir. Fakat bundan daha önemli olan bakılacak komşu sayısının yani k değerinin kaç olduğunu belirlemektir. Bazı kaynaklarda k değerinin gözlem değerinin karekökü olarak belirlenebileceği belirtilirken, bazı kaynaklarda ise gözlem sayısının karekökünü orta nokta kabul edip ona göre k değerleri hesaplamak önerilir.

```
library(class)
library(caret)
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:survival':
##
##     cluster
```

```
data("mtcars")
veri <- mtcars

veri$am <- as.factor(veri$am)
ayir <- createDataPartition(y=veri$am, p = 0.75, list = FALSE)

egitim <- veri[ayir,]
test <- veri[-ayir,]

egitim_v <- egitim[,-9]
test_v <- test[,-9]

egitim_h <- egitim[[9]]
test_h <- test[[9]]

tahmin <- knn(egitim_v, test_v, egitim_h, k = 6)

(cm <- confusionMatrix(test_h, tahmin))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 3 1
##          1 0 3
##
##                Accuracy : 0.8571
##                  95% CI : (0.4213, 0.9964)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 0.1243
##
##                   Kappa : 0.72
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 1.0000
##             Specificity : 0.7500
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4286
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.5714
##       Balanced Accuracy : 0.8750
##
##        'Positive' Class : 0
##
```

Yukarıda modelin sonuçları görülüyor. Fakat Accuracy, Sensitivity, Specificity, Kappa gibi birçok metrik ölçülmüş. Bunların hepsi iyi olsa bile bu modelin başarılı olduğu anlamına gelmez. Şöyle bir örnek verelim; 100 tane kedi köpek resmi arasından hangisinin kedi hangisinin köpek olduğunu bulmak istediğimizde, eğer test setimizde sadece 10 kedi resmi varsa ve model hepsini köpek olarak tahmin etmişse, Accuracy ve Sensitivity %90 çıkmasına rağmen Specificity 0% çıkar. Köpek olma durumunu pozitif olarak belirlediğimizde;

Sensitivity = TP/(TP + FN) = 90% (Modelin köpekleri tanıma durumu) Specificity = TN/(TN + FP) = 0% (Modelin kedileri tanıma durumu) Accuracy = (TN + TP)/(TN+TP+FN+FP) = 90% (Ortak başarı)

Bu demektir ki bizim modelimiz kedi olma durumunu hiç öğrenememiş. Yani karşımıza bir kedi geldiğinde onu tanımamız mümküğn değil. Hatta ve hatta model hiç bir şey öğrenememiş ve her gelen veriye köpek diyip geçmiş olabilir. Yani böyle bir durumda model aslında köpekleri de tanımıyor olabilir.

Dolayısıyla tek bir ölçüm yöntemine, hatta 2-3 tanesine göre bile karar vermek doğru değildir. ROC eğrisi çizdirmek, F-Testi değeri, vs. gibi başka yöntemler de her zaman tercih edilmelidir.

SON