Yapay Sinir Ağları ile Regresyon Problemi Çözümü

ELİF EKMEKCİ

2023-02-07

Bu örnekte yapay sinir ağları çalışırken "StatCrucnch" sitesindeki "NBA" veri seti üzerine çalışmalar gerçekleştirilmiştir.

Amacımız verilen değişkenler ile oyuncuların verimlilik derecesinin(PER) tahminini yapmaktır.

VERİ SETİ AÇIKLAMASI

Bu veri setine dahil edilmiş oyuncular en az 40 maça çıkmış oyunculardır. Bu veri setinde bir NBA basketbolcusunun 2013-2014 sezonu boyunca ne kadar iyi olduğunu gösteren bazı değişkenlerle çalışılmıştır.(Bu sezonda MVP (EDO) Kevin Durant seçilmiştir).

Kaynak: https://www.statcrunch.com/app/index.php?dataid=1096769&groupid=958

DEĞİŞKENLER

- PER: Oyuncu Verimlilik Derecesi; dakika başına üretim ölçüsü, lig ortalaması 15 olacak şekilde standardize edilmiştir.
- Player: Basketbolcu isimleri
- Age: 1 Subat 2014 itibari ile oyuncu kaç yaşınndaydı?
- Games: Bir oyuncunun oynadığı maç sayısı
- Minutes: Bir oyuncunun oynadığı süre sayısı
- TS: Isabetli atış yüzdesi;2 sayılık,3 sayılık ve serbest atışları dikkate alan atış verimlilik ölçüsüdür.
- ORB: Ofansif ribaund yüzdesi.
- DRB: Defansif ribaund yüzdesi.
- TRB: Toplam ribaund yüzdesi.
- AST: Oyuncunun yaptığı asist yüzdesi.
- STL: Top çalma yüzdesi.
- BLK: Top bloklama yüzdesi.
- TOV: Top kaybı yüzdesi
- USG: Topa sahip olma yüzdesi
- ORtg: 100 ofansif atakta atılan sayı yüzdesi
- DRtg: 100 defansif atakta atılan sayı yüzdesi
- OWS: Bir oyuncun atakta kazandığı top yüzdesi
- DWS: Bir oyuncun defasnta kazandığıi top yüzdesi
- WS:Kazanma yüzdesi

1.ADIM: Kullanılan Paketlerin Yüklenmesi ve Aktifleştirilmesi

KULLANILAN PAKETLER

```
library(readr)
library(dplyr)
library(tidyr)
library(corrr)
library(corrplot)
library(ggcorrplot)
library(PerformanceAnalytics)
library(faraway)
library(neuralnet)
library(NeuralNetTools)
library(readxl)
library(corrplot)
```

2.ADIM: Veri Yükleme ve Düzenleme

- Öncelikle verimizi yükleyelim ve NBA_orj isimi ile tanımlayalım:

```
NBA_orj <- read_xlsx("/Users/elif/Desktop/NBA.xlsx")
```

- Verimizde yer alan "Age","Games","Minutes","PER","TS","STL" ve "WS" değişkenleri ile incelemeler yapacağız bu nedenle bu adımda değişkenlerimizin seçimini yapalım ve NBA isimi ile tanımlayalım:

```
NBA <- NBA_orj %>% select(c("Age","Games","Minutes",
  "PER","TS","STL","WS" ))
```

```
head(NBA)
```

```
## # A tibble: 6 x 7
##   Age   Games Minutes PER   TS    STL   WS
##   <chr> <chr> <chr>   <chr> <chr> <chr> <chr>
## 1 25    81    3122    29.8  0.635 1.7   19.2
## 2 29    77    2902    29.3  0.649 2.2   15.9
## 3 25    77    2797    26.9  0.591 1     14.3
## 4 20    67    2358    26.5  0.582 2     10.4
## 5 23    71    2298    26.1  0.555 2.4   7.9
## 6 28    62    2171    25.9  0.58  3.5   12.2
```

3.ADIM: Veri Keşfi

- Veri türlerini glimpse() ile kontrol edelim:

```
glimpse(NBA)
```

```
## Rows: 342
## Columns: 7
## $ Age     <chr> "25", "29", "25", "20", "23", "28", "25", "29", "25", "24", "3~
## $ Games   <chr> "81", "77", "77", "67", "71", "62", "46", "77", "78", "80", "8~
## $ Minutes <chr> "3122", "2902", "2797", "2358", "2298", "2171", "1412", "2982"~
## $ PER     <chr> "29.8", "29.3", "26.9", "26.5", "26.1", "25.9", "24.7", "24.4"~
```

```
## $ TS      <chr> "0.635", "0.649", "0.591", "0.582", "0.555", "0.58", "0.545", ~
## $ STL     <chr> "1.7", "2.2", "1", "2", "2.4", "3.5", "3.1", "1.7", "2.2", "1.~
## $ WS      <chr> "19.2", "15.9", "14.3", "10.4", "7.9", "12.2", "5.2", "10.7", ~
```

Veri türlerini incelediğimizde chr yani karakter yapıda olduklarını görüyoruz.Öncelikle değişkenlerimizi numerik değişkenlere çevirmeliyiz.

```
NBA$Age <- as.numeric(NBA$Age)
NBA$Games <- as.numeric(NBA$Games)
NBA$Minutes <- as.numeric(NBA$Minutes)
NBA$PER <- as.numeric(NBA$PER)
NBA$TS <- as.numeric(NBA$TS)
NBA$STL <- as.numeric(NBA$STL)
NBA$WS <- as.numeric(NBA$WS)
```

- Verimizin özet istatistiklerini inceleyelim:

```
summary(NBA)
```

```
##       Age            Games          Minutes          PER
##  Min.   :19.00   Min.   :40.00   Min.   : 265   Min.   : 3.80
##  1st Qu.:23.00   1st Qu.:58.00   1st Qu.:1006   1st Qu.:11.60
##  Median :26.00   Median :71.00   Median :1662   Median :13.90
##  Mean   :26.44   Mean   :67.38   Mean   :1645   Mean   :14.23
##  3rd Qu.:29.00   3rd Qu.:79.00   3rd Qu.:2236   3rd Qu.:16.50
##  Max.   :39.00   Max.   :82.00   Max.   :3122   Max.   :29.80
##        TS              STL              WS
##  Min.   :0.3900   Min.   :0.200   Min.   :-0.900
##  1st Qu.:0.5070   1st Qu.:1.200   1st Qu.: 1.400
##  Median :0.5350   Median :1.500   Median : 2.800
##  Mean   :0.5365   Mean   :1.599   Mean   : 3.587
##  3rd Qu.:0.5677   3rd Qu.:2.000   3rd Qu.: 5.100
##  Max.   :0.7300   Max.   :3.800   Max.   :19.200
```

PER değişkenini inceleyelim:

- Minimum değerin 3.80 ve maksimum değerin ise 29.80 olduğunu görüyoruz.
- İlk çeyreği (1st Qu.) 11.60'dır.Bu da tüm kayıtların %25'inin PER değerinin 11.60'ın altında olduğunu gösterir.
- Benzer şekilde üçüncü çeyrekte (3rd Qu.) 16.50 değeri tüm kayıtların %75'inin PER değerinin 16.50 'nin altında olduğunu gösterir.
- PER değerinin ortalaması ise bize aritmetik ortalamayı gösterir ve 14.23 olarak hesaplandığı görülür.

Ayrıca veri özetine baktığımzızda factor haline getirilecek değişken olmadığını da görmekteyiz.Dolayısıyla hatalı bir durum yoktur.

- Verimizde eksik gözlemler var mı yok mu apply komutuyla kontrol edelim:

```
apply(NBA,2,function(x) sum(is.na(x)))
```

```
##     Age   Games Minutes     PER      TS     STL      WS
##       0       0       0       0       0       0       0
```

Bu işlemi yaparken apply fonksiyonu kullanılmıştır.(Her değişkenin altında yazan değer,o değişkende kaç eksik gözlem olduğunu gösterir.) Sonuç olarak kayıp gözlem olmadığı tespit edilmiştir. Karşılaşılırsa, na.omit() işlevi, ilgili durumları bir veri çerçevesinden, matristen veya vektörden çıkarmak için kullanılabilir.

- Korelasyon değerleri araştıralım:

Öncelikle tahmin edilecek olan hedef değişkenin açıklayıcı değişkenlerle olan ilişkisine bakalım:

```
NBA%>% correlate() %>% focus(PER)
```

```
## # A tibble: 6 x 2
##   term       PER
##   <chr>    <dbl>
## 1 Age     0.0534
## 2 Games   0.260
## 3 Minutes 0.537
## 4 TS      0.584
## 5 STL     0.130
## 6 WS      0.800
```

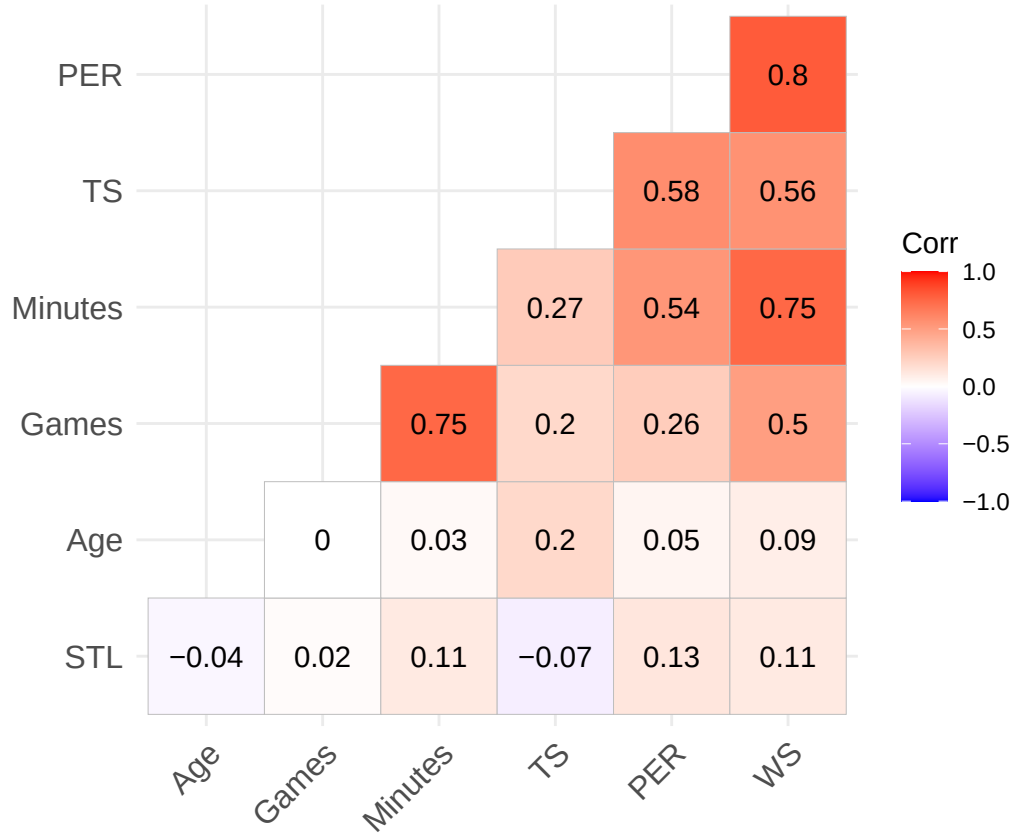
Korelasyon değerleri incelendiğinde:

- WS değişkeni ile hedef değişken PER arasında yüksek bir pozitif korelasyon vardır.
- Minutes ve TS değişkenleri ile hedef değişken PER arasında pozitif bir korelasyon vardır.
- Games,Age ve STL değişkenleri ile hedef değişken PER arasında düşük bir pozitif korelasyon vardır.
- Değişkenler ile hedef değişken PER arasında negatif bir korelasyon yoktur.

Değişkenler arasındaki korelasyonu görsel olarak incelemek istersek:

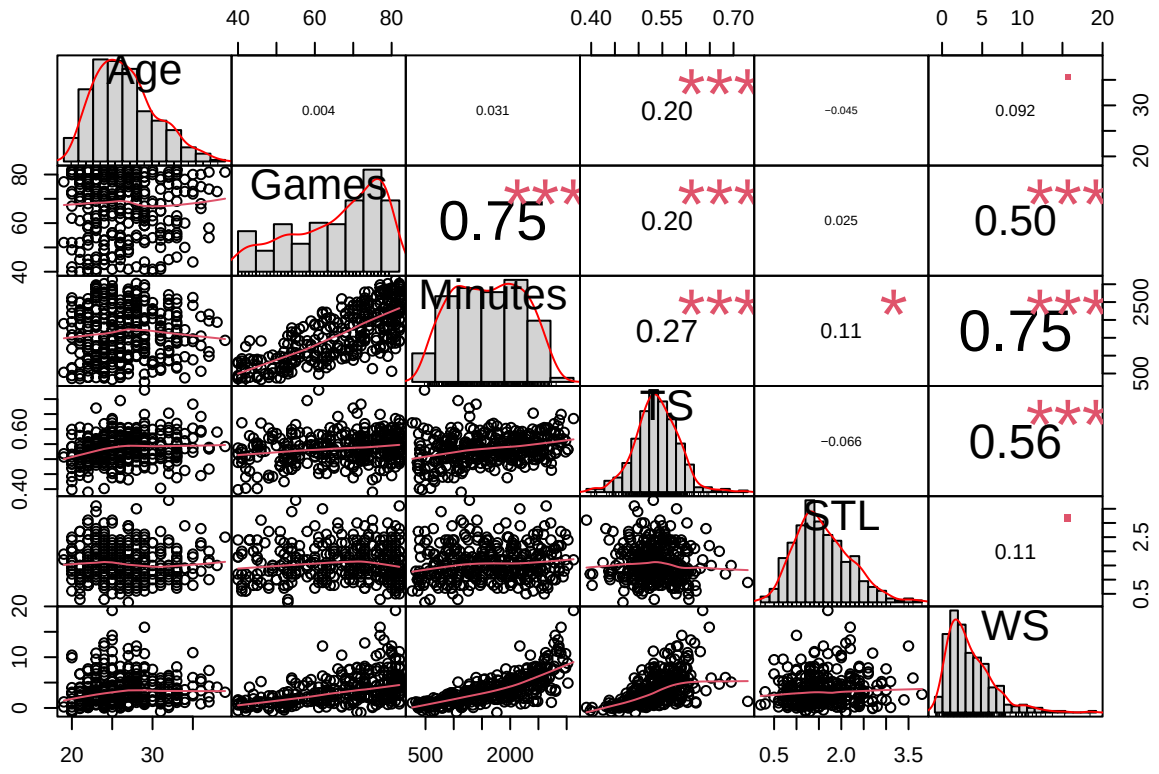
```
nba_cor <- cor(NBA, use="complete.obs")

ggcorrplot(nba_cor,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE)
```



- İkinci olarak, açıklayıcı değişkenler arasındaki doğrusal ilişkinin ölçütleri olan ikili korelasyon değerlerini gözden geçirelim:

```
chart.Correlation(NBA[,-4],histogram = TRUE,pch = 19)
```



Grafiği incelediğimizde,

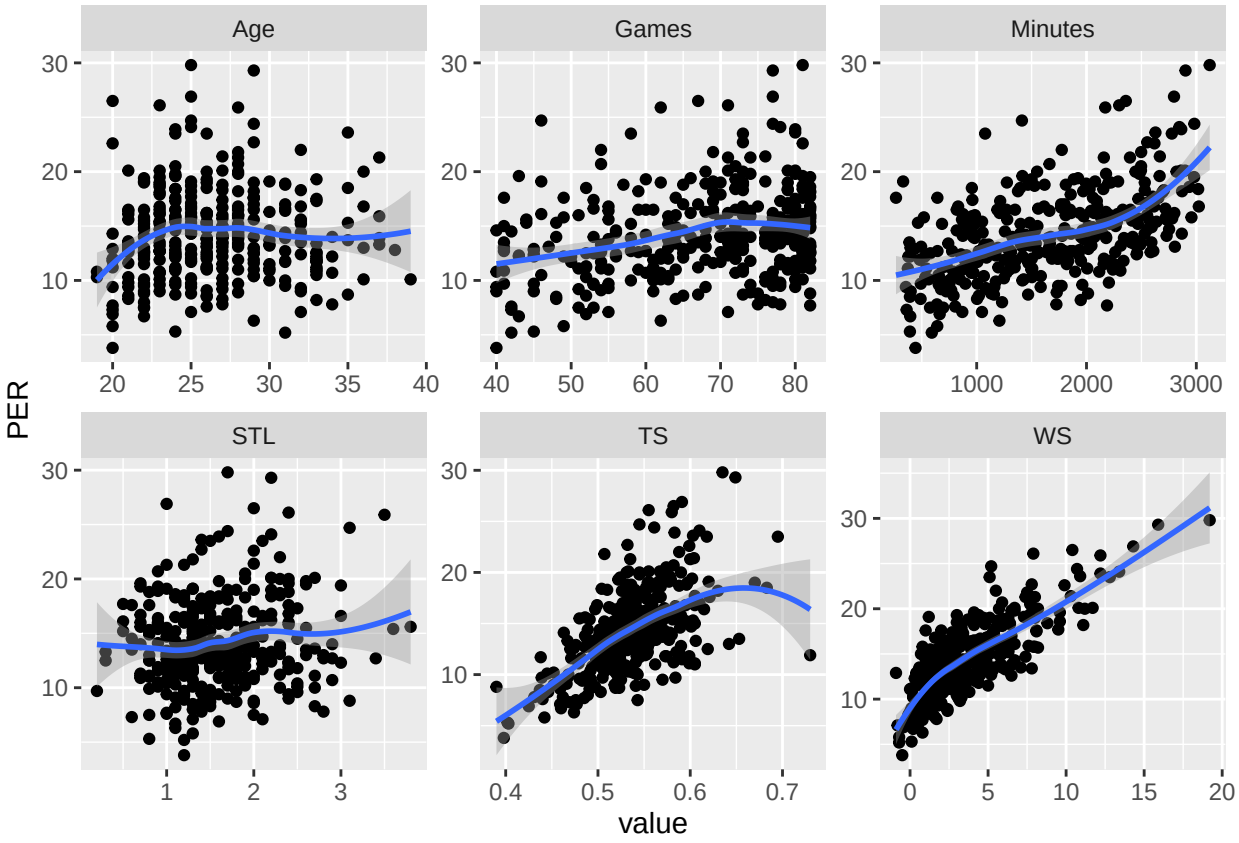
- En yüksek pozitif korelasyonlar:
  - 0.75 ile Games ve Minutes değişkenleri arasında
  - 0.75 ile Minutes ve WS değişkenleri arasında
  - 0.56 ile TS ve WS değişkenleri arasında
- Negatif korelasyonlar:
  - -0.045 ile Age ve STL değişkenleri arasında
  - -0.066 ile TS ve STL arasındadır.

Değişkenlerin dağılımlarını inceleyelim

Hedef değişken PER'in açıklayıcı değişkenlere karşı dağılım grafikleri oluşturalım ve yorumlayalım:

```
NBA %>%
gather(-PER, key = "var", value = "value") %>%
filter(var != "chas") %>%
ggplot(aes(x = value, y = PER)) +
```

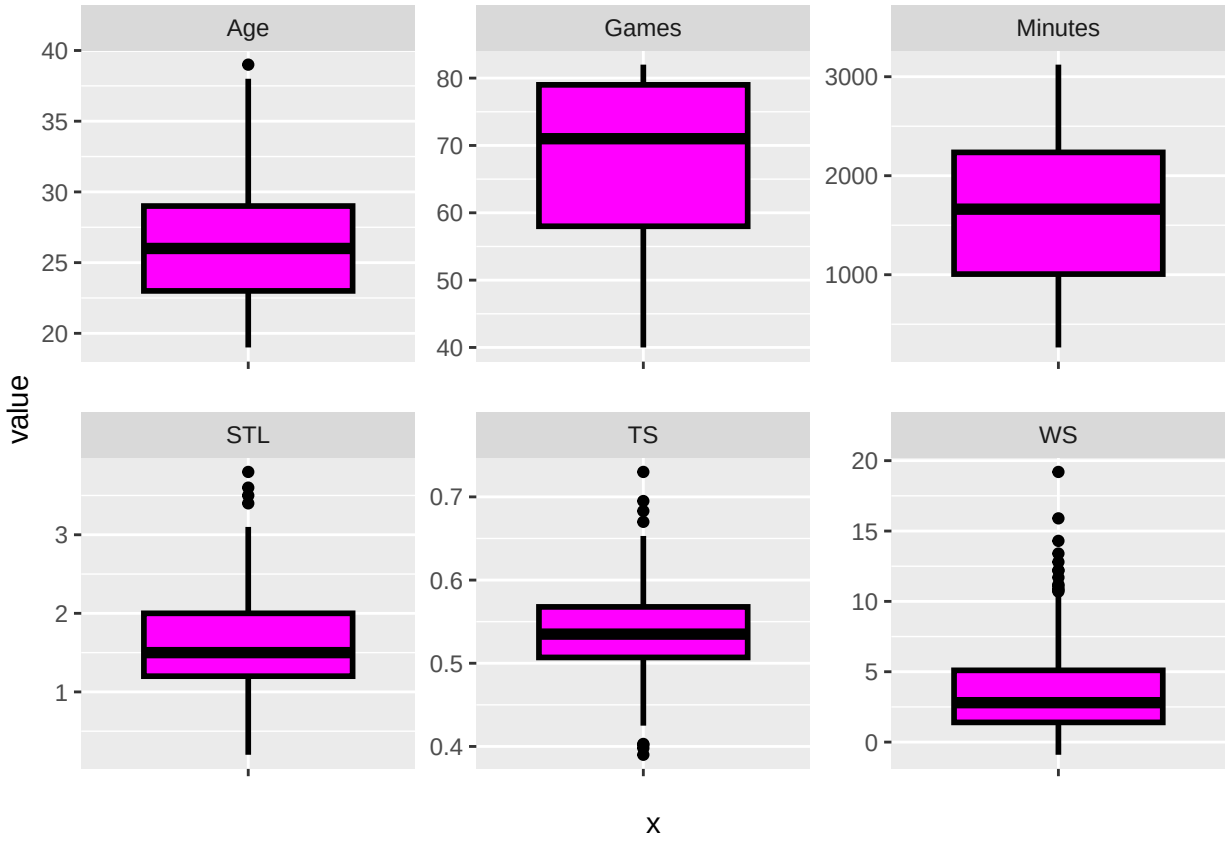
```
geom_point() +
stat_smooth() +
facet_wrap(~ var, scales = "free") +
theme_get()
```



Dağılım grafiklerinden hedef değişken PER değişkeni ile WS değişkeni arasında doğrusal bir ilişki olduğu görülmektedir.Diğer açıklayıcı değişkenler arasında doğrusal olmayan bir yapı olduğu görülmektedir.

- Açıklayıcı değişkenler için Kutu grafiği çizdirelim ve yorumlayalım:

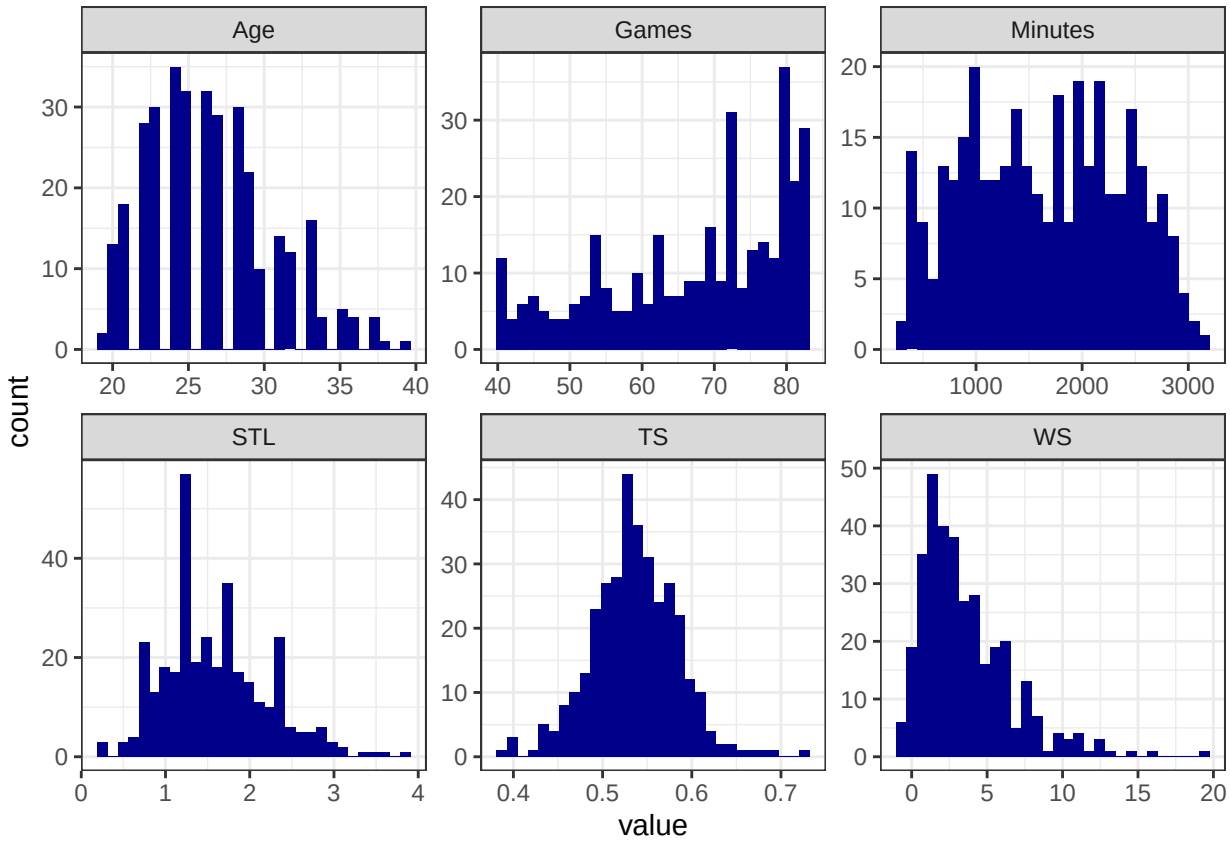
```
NBA %>%
gather(-PER, key = "var", value = "value") %>%
filter(var != "chas") %>%
ggplot(aes(x = '',y = value)) +
geom_boxplot(fill = '#FF00FF', color="black", size=1) +
facet_wrap(~ var, scales = "free") +
theme_get()
```



Kutu çizimleri incelendiğinde, verilerde bazı aykırı değerlerin olduğu görülür.

- Son olarak açıklayıcı değişkenlerin histogramını çizdirirsek:

```
NBA %>%
gather(-PER, key = "var", value = "value") %>%
filter(var != "chas") %>%
ggplot(aes(x = value)) +
geom_histogram(fill="darkblue") +
facet_wrap(~ var, scales = "free") +
theme_bw()
```



- Age ve STL değişkenlerinin aralarında hiçbir veri olmadan ayrılmış iki farklı tepe noktası vardır ve bu durum karışım dağılımının (mixture distribution) varlığına işaret eder.
- TS değişkeninin normal dağılım gösterdiği gözlemlenmektedir.
- Ayrıca burada çoğu değişkenin dağılımlarının çarpık olduğu gözlemlenmiştir.

4.ADIM: Veri Setini Normalleştirme

Verimizin yanıt değişkeni sürekli olduğu için standartlaştırma işlemi uygulamamız gerekmektedir.

Bu işlem için öncellikle sütunların maksimum ve minimum değerlerine bakalım;

```
maxs <- apply(NBA,2,max) # sutunların maksimum degerlerini verir
mins <- apply(NBA,2,min) # sutunların minimum degerlerini verir
```

Maksimum ve minimum değerleri bulduk,şimdi ise verimizi normalleştirelim. Center komutu ile verimizdeki tüm gözlemlerden minimum değerleri çıkartıp,Scale komutu ile değerleri maksimum- minimum değerine bölerek verimizi standart hale getirelim.

$ScaledData = \frac{x_{observed} - x_{min}}{x_{max} - x_{min}}$

Bu işlem ile max-min normalization işlemi yapıyoruz,bu işlem verinin dağılımını ve yapısını bozmamaktadır.

```
scaled <- as.data.frame(scale(NBA,center = mins,scale = maxs-mins))
head(scaled)
```

```
##    Age     Games   Minutes       PER        TS       STL        WS
## 1 0.30 0.9761905 1.0000000 1.0000000 0.7205882 0.4166667 1.0000000
## 2 0.50 0.8809524 0.9229961 0.9807692 0.7617647 0.5555556 0.8358209
## 3 0.30 0.8809524 0.8862443 0.8884615 0.5911765 0.2222222 0.7562189
## 4 0.05 0.6428571 0.7325866 0.8730769 0.5647059 0.5000000 0.5621891
## 5 0.20 0.7380952 0.7115856 0.8576923 0.4852941 0.6111111 0.4378109
## 6 0.45 0.5238095 0.6671334 0.8500000 0.5588235 0.9166667 0.6517413
```

5.ADIM: Veri Kümesini Bölme

Bu adımda verimizi test ve train olarak ayıralım.Verimizden %75'lik örneklem çekerek bunu train ve kalan %25'lik kısmıyla test veri setimizi oluşturalım:

```
set.seed(121519016)
index <- sample(1:nrow(NBA),round(0.75*nrow(NBA)))
train <- scaled[index,] # %75'lik kismi train olarak ayirdik
test <- scaled[-index,] # geriye kalan %25'lik kismi test olarak ayirdik
```

6.ADIM: Yapay Sinir Ağı Modeli Oluşturma ve Eğitme

Verimizi test ve train olarak ayırdıktan sonra bu adımda yapay sinir ağı modelimizi kurmaya başlayalım.

```
set.seed(121519016)
n <- names(train) # train veri setindeki degiskenlerin isimlerini aliyor
f <- as.formula(paste("PER~",paste(n [!n %in% "PER"],collapse = " + ")))
#bu komut kolay yoldan model kurmamizi sagliyor
f
```

```
## PER ~ Age + Games + Minutes + TS + STL + WS
```

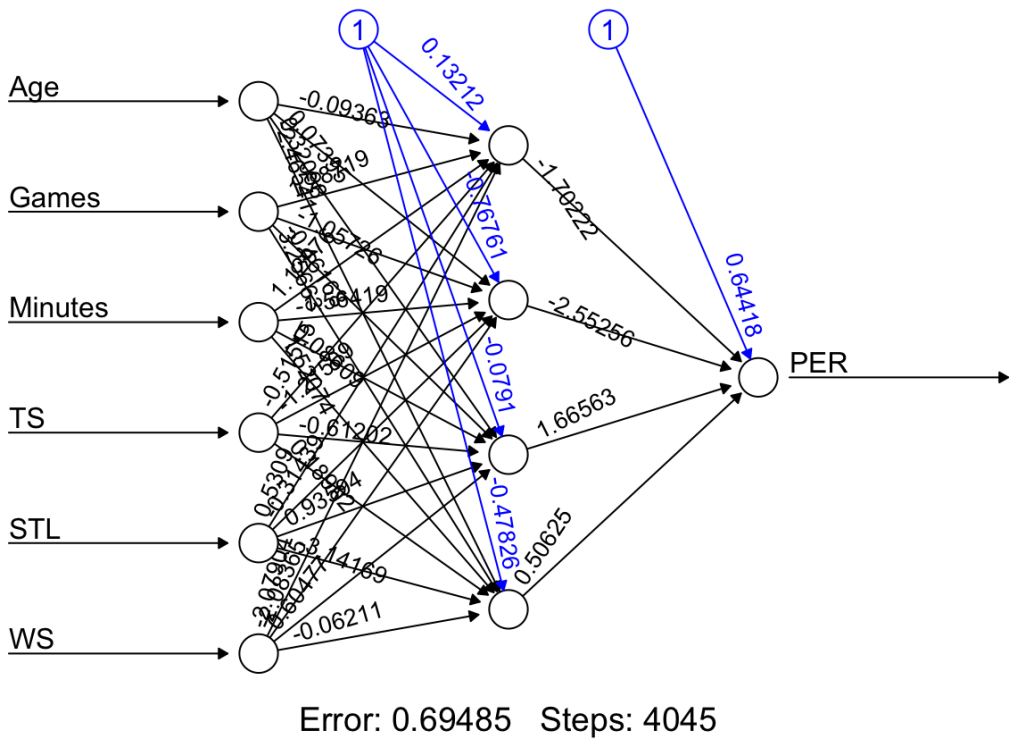
Tek katmanlı sinir ağı çizdirelim:

```
nn <- neuralnet(f,data = train,hidden = 4, linear.output = TRUE ) # yapay sinir agi modeli
```

Burada hidden komutu değişken sayımızın 2/3'ü olarak alınır.hidden = 4 almamızın sebebi yanıtı etkileyen değişkenlerimizin sayısının 6 olmasından kaynaklıdır. Yanıt değişkenimiz sürekli olduğundan linear.output değerini TRUE olarak kullanmalıyız.**(Sınıflandırma problemlerinde linear.output FALSE olarak kullanılır.)**

Şimdi nn ile oluşturduğumuz Yapay Sinir Ağı modelini plot ile çizdirelim;

```
plot(nn)
```



Yapay sinir ağı için çizdirdiğimiz plota baktığımızda gizli katmanında 4 tane nöronun olduğu görülmektedir. Error: 0,69485 ve Steps: 4045 çıkmıştır.

Bu değerler min-max normalization yapılmış haline göre hesaplanmıştır. Şimdi yaptığımız min-max normalization işlemini geri döndürme işlemini yapmalıyız.

```
set.seed(121519016)
pr.nn.train <- compute(nn,train[,-4])
pr.nn.train_real <- pr.nn.train$net.result*(max(NBA$PER)-min(NBA$PER))+min(NBA$PER)
# geri dondurme islemi yapiyoruz
training.real <- (train$PER) * (max(NBA$PER) - min(NBA$PER)) + min(NBA$PER)
# train seti icin gercek gozlemleri hesapliyoruz
```

Şimdi kurduğumuz Yapay Sinir Aği Modeli için train veri seti üzerinden RMSE,AMPE ve MdAPE değerlerini hesaplayalım;

- Train set üzerinden RMSE değerini hesaplayalım:

```
RMSE.nn.train <- (sum((training.real - pr.nn.train_real)^2) / nrow(train))^0.5
RMSE.nn.train
```

```
## [1] 1.91564
```

Yapay Sinir Ağı modelimiz için Train veri seti üzerinden hesaplanan RMSE değerimiz 1.91564 çıkmıştır.

- Train set üzerinden MAPE değerini hesaplayalım:

```
MAPE.nn.train <- mean(abs(((training.real - pr.nn.train_real)/training.real)))
MAPE.nn.train
```

```
## [1] 0.1135937
```

Yapay Sinir Ağı modelimiz için Train veri seti üzerinden hesaplanan MAPE değerimiz 0.1135937 çıkmıştır.

- Train set üzerinden MdAPE değerini hesaplayalım:

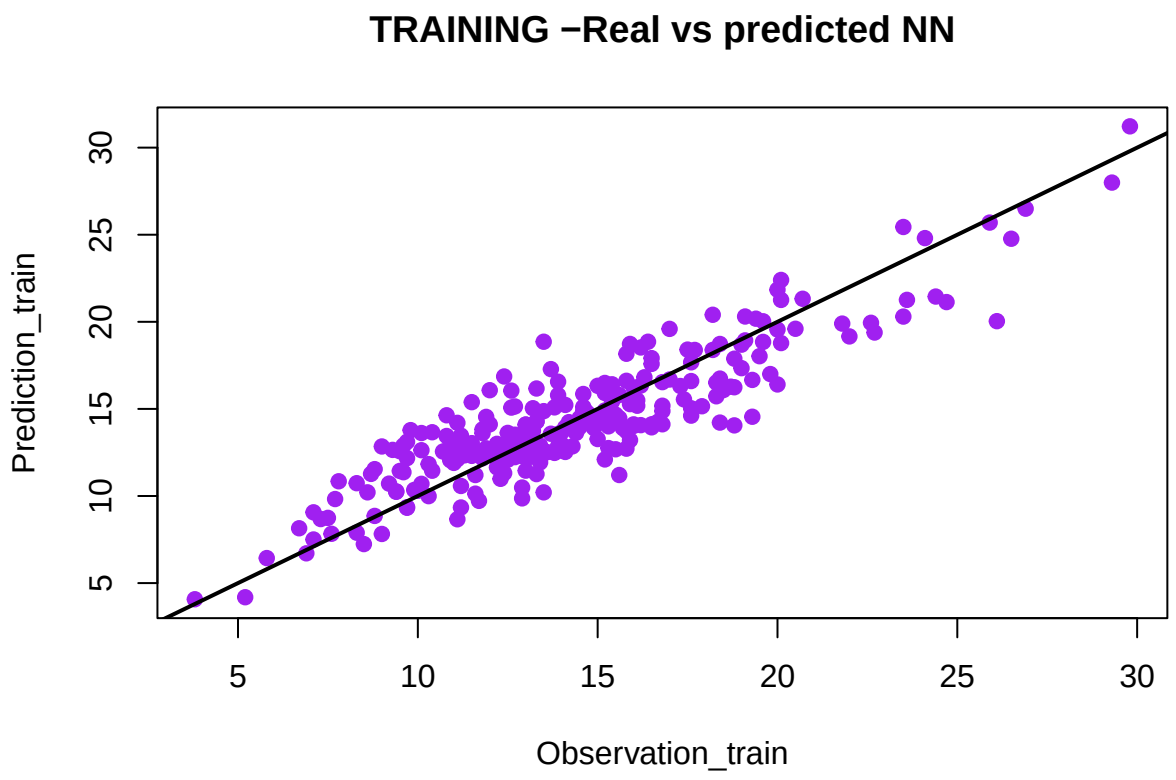
```
MdAPE.nn.train <- median(abs(((training.real - pr.nn.train_real)/training.real )))
MdAPE.nn.train
```

```
## [1] 0.08890611
```

Yapay Sinir Ağı modelimiz için Train veri seti üzerinden hesaplanan MdAPE değerimiz 0.08890611 çıkmıştır.

Verimizdeki yanit degiskenimiz PER'in train veri seti uzerinden Gercek Ve Tahmin NN lerinin plotunu cizdirelim;

```
Observation_train <- training.real
Prediction_train <- pr.nn.train_real
plot(Observation_train,Prediction_train,col = "purple",
     main = "TRAINING -Real vs predicted NN",pch=19,cex= 1)
abline(0,1,lwd = 2)
```



Train verisinin gerçek değerleri(x-ekseni) ve ANN üzerinden tahmin edilen degerleri(y-ekseni) üzerinde görülmektedir. Verimizdeki yanıt değişkenimiz PER'in train veri seti üzerinden Gercek Ve Tahmin NN'lerinin plotuna baktığımızda çoğu değişkenimiz doğru tahmin edilmiştir. Yapay Sinir Agi tahminleri ile orijinal veri seti değerlerimiz uyumlu çıkmıştır.

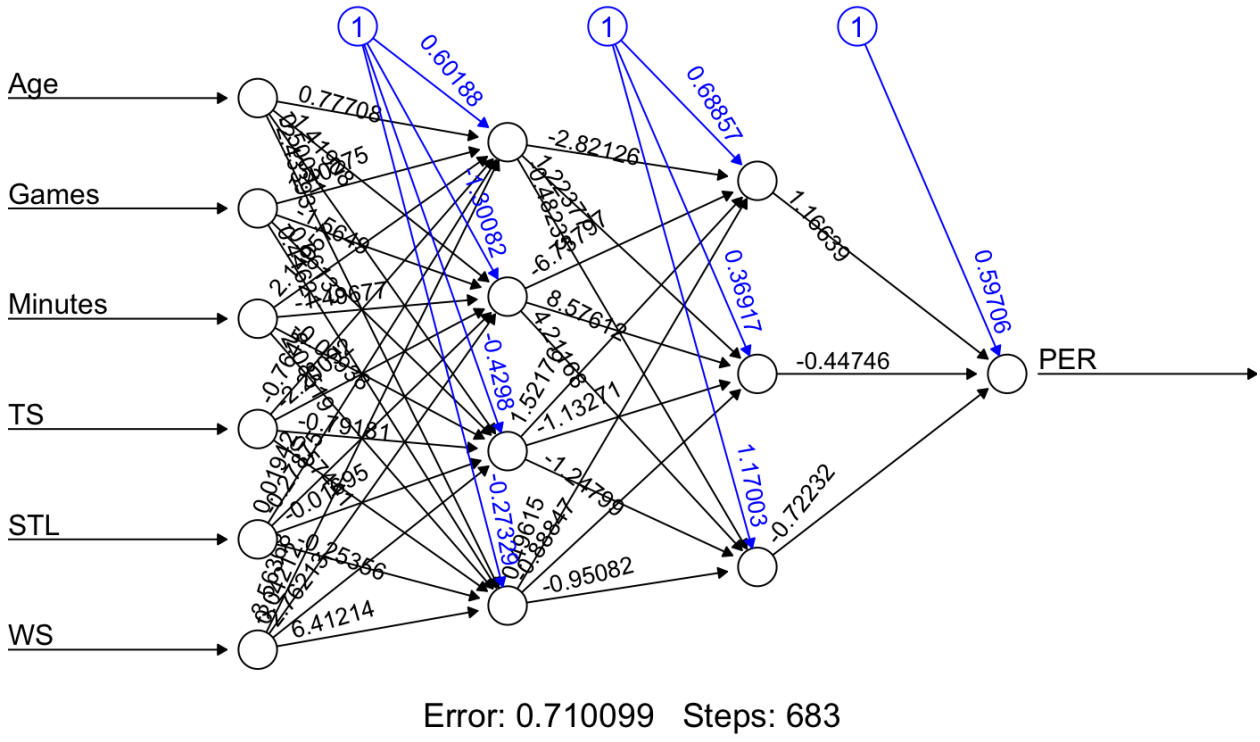
İki katmanlı sinir ağı çizdirelim:

Şimdi Yapay Sinir Ağı modelimizi hidden = c(4,3) komutu ile iki katmanlı olarak kuralım. İlk katmanda 4 nöron ve ikinci katmanda 3 nöron kullanarak inceleyelim;

```
set.seed(121519016)
nn1 <- neuralnet(f,data = train,hidden = c(4,3),linear.output = TRUE)
```

nn1 ile oluşturduğumuz Yapay Sinir Ağı modelimizi plot ile çizdirelim;

```
plot(nn1)
```



Yapay Sinir Ağı için çizdirdiğimiz plota baktığımızda ilk gizli katmanında 4 tane nöronun olduğu ve ikinci gizli katmanında 3 nöron olduğu gözükmektedir. Error: 0.710099 ve Steps: 683 çıkmıştır.

Bu değerler min-max normalization yapılmış haline göre hesaplanmıştır. Şimdi yaptığımız min-max normalization işlemini geri döndürme işlemini yapmalıyız.

```
set.seed(121519016)
pr.nn1.train <- compute(nn1,train[,-4])
# PER (yanit degiskeni) disindaki degiskenleri modele sokuyor
pr.nn1.train_real <- pr.nn1.train$net.result*(max(NBA$PER)-min(NBA$PER))+min(NBA$PER)
# geri dondurme islemi yapiyoruz
```

```
# min-max ile çarpıp min ekledik
training.real1 <- (train$PER)*(max(NBA$PER) - min(NBA$PER)) + min(NBA$PER)
# train seti icin gercek gozlemleri hesapliyoruz
```

Şimdi kurduğumuz Yapay Sinir Aği Modeli için train veri seti üzerinden RMSE,AMPE ve MdAPE değerlerini hesaplayalım;

- Train set üzerinden RMSE değerini hesaplayalım:

```
RMSE.nn1.train <- (sum((training.real1 - pr.nn1.train_real)^2) / nrow(train))^0.5
RMSE.nn1.train
```

```
## [1] 1.936547
```

Yapay Sinir Ağı modelimiz için Train veri seti üzerinden hesaplanan RMSE değerimiz 2.326441 çıkmıştır.

- Train set üzerinden MAPE değerini hesaplayalım:

```
MAPE.nn1.train <- mean(abs(((training.real1 - pr.nn1.train_real)/training.real1)))
MAPE.nn1.train
```

```
## [1] 0.1146769
```

Yapay Sinir Ağı modelimiz için Train veri seti üzerinden hesaplanan MAPE değerimiz 0.0.1352771 çıkmıştır.

- Train set üzerinden MdAPE değerini hesaplayalım:

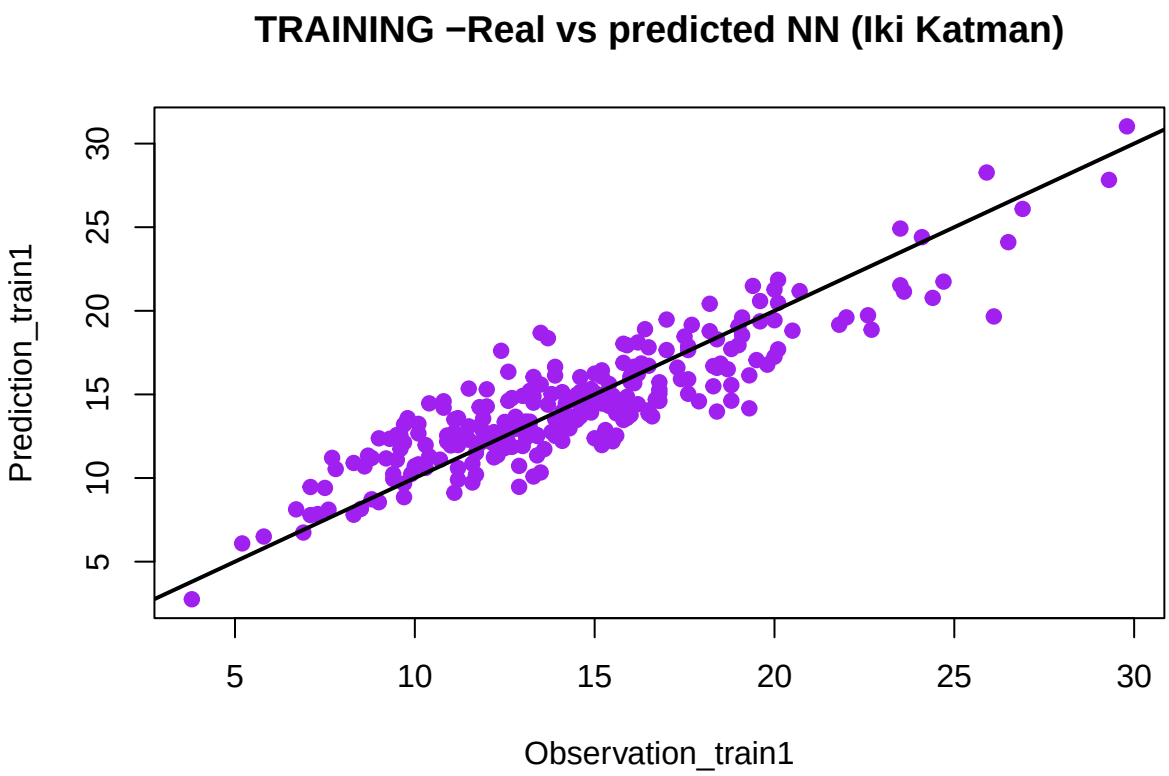
```
MdAPE.nn1.train <- median(abs(((training.real1 - pr.nn1.train_real)/training.real1 )))
MdAPE.nn1.train
```

```
## [1] 0.08869471
```

Yapay Sinir Ağı modelimiz için Train veri seti üzerinden hesaplanan MdAPE değerimiz 0.103829 çıkmıştır.

Verimizdeki yanit degiskenimiz PER'in train veri seti uzerinden Gercek Ve Tahmin NN lerinin plotunu cizdirelim;

```
Observation_train1 <- training.real1
Prediction_train1 <- pr.nn1.train_real
plot(Observation_train1,Prediction_train1,col = "purple",
     main = "TRAINING -Real vs predicted NN (Iki Katman)",pch=19,cex= 1)
abline(0,1,lwd = 2)
```



Train verisinin gerçek değerleri(x-ekseni) ve ANN üzerinden tahmin edilen degerleri(y-ekseni) üzerinde görülmektedir. Verimizdeki yanıt değişkenimiz PER'in train veri seti üzerinden Gercek Ve Tahmin NN'lerinin plotuna baktığımızda çoğu değişkenimiz doğru tahmin edilmiştir. Yapay Sinir Agi tahminleri ile orijinal veri seti değerlerimiz uyumlu çıkmıştır.

7.ADIM: Sinir Ağını Test Edin

Bu adımda test setinde, eğitimli sinir ağının performansının bir değerlendirmesi yapalım.

Tek Katmanlı Model için hesaplarsak:

```
set.seed(121519016)
pr.nn.test <- compute(nn, test[-4])
# PER (yanit degiskeni) disindaki degiskenleri modele sokuyoruz

# geri dondurme islemi yapiyoruz
pr.nn.test_real <- pr.nn.test$net.result *(max(NBA$PER) - min(NBA$PER)) + min(NBA$PER)

# train seti icin gercek gozlemleri hesapliyoruz
testing.real <- (test$PER) * (max(test$PER) - min(test$PER)) + min(test$PER)
```

- Test seti üzerinden RMSE değerini hesaplayalım:

```
RMSE.nn.test <- (sum((testing.real - pr.nn.test_real)^2) / nrow(test))^0.5
RMSE.nn.test
```

```
## [1] 14.16808
```

Yapay Sinir Ağı modelimiz için Test veri seti üzerinden hesaplanan RMSE değerimiz 14.16808 çıkmıştır.

- Test seti üzerinden MAPE değerini hesaplayalım:

```
MAPE.nn.test <- mean(abs(((testing.real - pr.nn.test_real)/testing.real)))
MAPE.nn.test
```

```
## [1] 44.31403
```

Yapay Sinir Ağı modelimiz için Test veri seti üzerinden hesaplanan MAPE değerimiz 44.31403 çıkmıştır.

- Test seti üzerinden MdAPE değerini hesaplayalım:

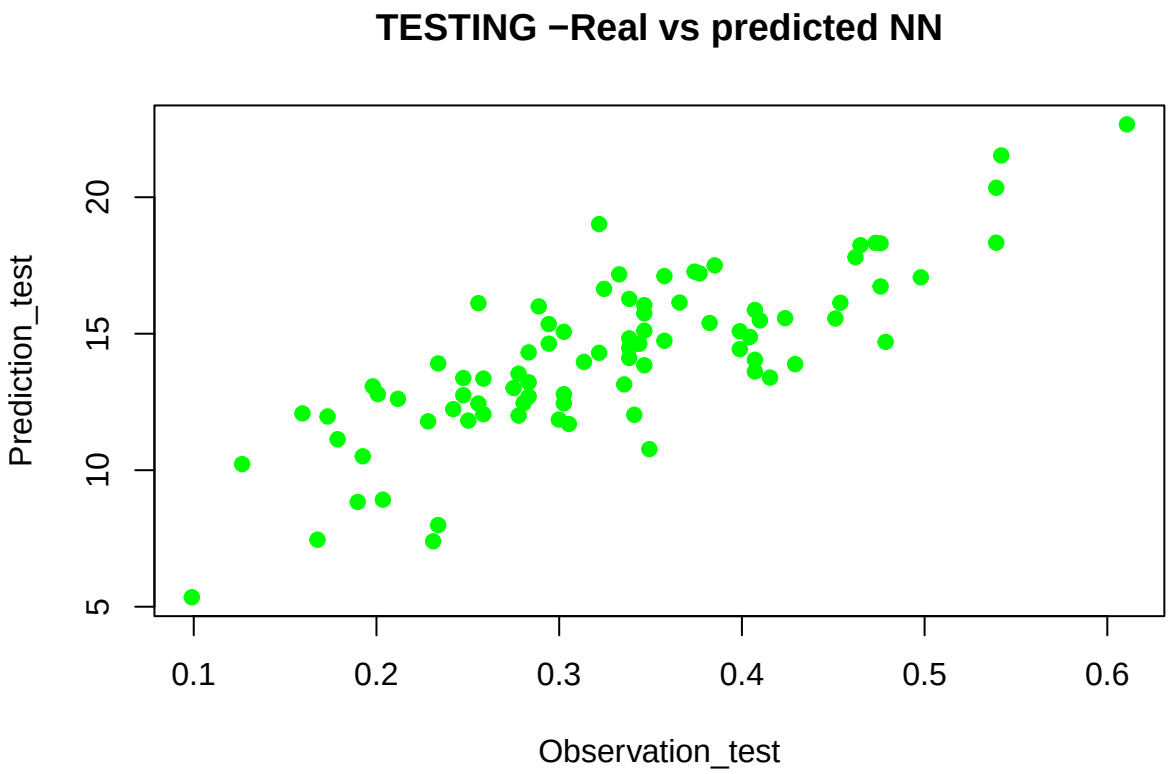
```
MdAPE.nn.test <- median(abs(((testing.real - pr.nn.test_real)/testing.real)))
MdAPE.nn.test
```

```
## [1] 43.27008
```

Yapay Sinir Ağı modelimiz için Train veri seti üzerinden hesaplanan MdAPE değerimiz 43.27008 çıkmıştır.

Verimizdeki yanit degiskenimiz PER'in test veri seti uzerinden Gercek Ve Tahmin NN lerinin plotunu cizdirelim;

```
Observation_test <- testing.real
Prediction_test <- pr.nn.test_real
plot(Observation_test,Prediction_test,col = "green",
     main = "TESTING -Real vs predicted NN",pch=19,cex= 1)
abline(0,1,lwd = 2)
```



Test verisinin gerçek değerleri(x-ekseni) ve ANN üzerinden tahmin edilen degerleri(y-ekseni) üzerinde görülmektedir. Verimizdeki yanıt değişkenimiz PER'in test veri seti üzerinden Gercek Ve Tahmin NN'lerinin plotuna baktığımızda çoğu değişkenimiz doğru tahmin edilmiştir. Yapay Sinir Agi tahminleri ile orijinal veri seti değerlerimiz uyumlu çıkmıştır.

İki Katmanlı Model için hesaplarsak:

```
set.seed(121519016)
pr.nn1.test <- compute(nn1, test[-4]) # PER (yanit degiskeni) disindaki degiskenleri modele sokuyoruz

# geri dondurme islemi yapiyoruz
pr.nn1.test_real <- pr.nn1.test$net.result *(max(NBA$PER) - min(NBA$PER)) + min(NBA$PER)

# train seti icin gercek gozlemleri hesapliyoruz
testing.real1 <- (test$PER) * (max(test$PER) - min(test$PER)) + min(test$PER)
```

- Test seti üzerinden RMSE değerini hesaplayalım:

```
RMSE.nn1.test <- (sum((testing.real1 - pr.nn1.test_real)^2) / nrow(test))^0.5
RMSE.nn1.test
```

```
## [1] 13.9994
```

Yapay Sinir Ağı modelimiz için Test veri seti üzerinden hesaplanan RMSE değerimiz 13.9994 çıkmıştır.

- Test seti üzerinden MAPE değerini hesaplayalım:

```
MAPE.nn1.test <- mean(abs(((testing.real1 - pr.nn1.test_real)/testing.real1)))
MAPE.nn1.test
```

```
## [1] 43.889
```

Yapay Sinir Ağı modelimiz için Test veri seti üzerinden hesaplanan MAPE değerimiz 43.889 çıkmıştır.

- Test seti üzerinden MdAPE değerini hesaplayalım:

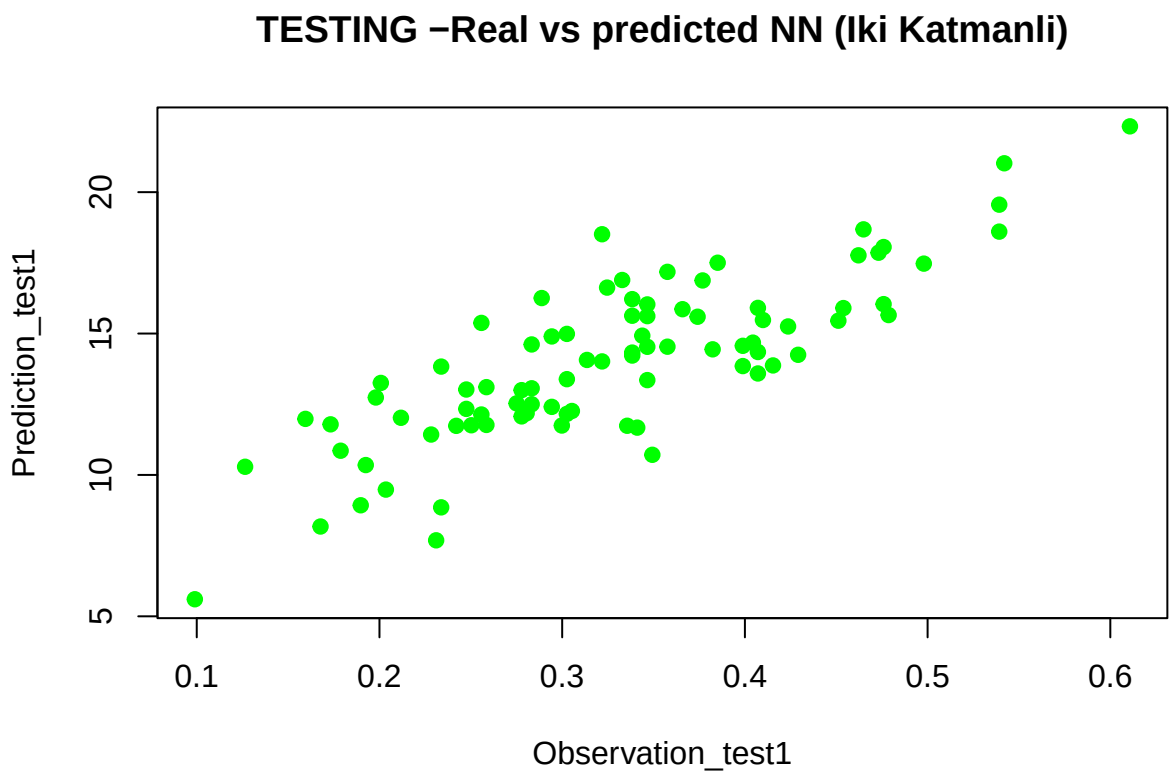
```
MdAPE.nn1.test <- median(abs(((testing.real1 - pr.nn1.test_real)/testing.real1)))
MdAPE.nn1.test
```

```
## [1] 42.49579
```

Yapay Sinir Ağı modelimiz için test veri seti üzerinden hesaplanan MdAPE değerimiz 42.49579 çıkmıştır.

Verimizdeki yanit degiskenimiz PER'in test veri seti uzerinden Gercek Ve Tahmin NN lerinin plotunu cizdirelim;

```
Observation_test1 <- testing.real1
Prediction_test1 <- pr.nn1.test_real
plot(Observation_test1,Prediction_test1,col = "green",
     main = "TESTING -Real vs predicted NN (Iki Katmanli)",pch=19,cex= 1)
abline(0,1,lwd = 2)
```



Baktigimiz tek katmanli Yapay Sinir Agi modelimizin yani ikinci modelin RMSE'si daha düşük gelmistir.(13.95191) Acaba bu durum gerçekten tek katmanlı Yapay Sinir Ağı modelimizin yani ilk modelin daha iyi olduğunu mu gösterir? Acaba test verisini farklı seçseydik de aynı durum söz konusu olur muydu ? Bu amaçla Cross Validation yapmak daha sağlıklıdır.

Şimdi ilk olarak ilk modelin (tek katmanlı) daha sonra ikinci modelin (iki katmanlı) Cross Validation Errorlarını hesaplayalım;

```
set.seed(121519016)

cv.error1 <- NULL
cv.error2 <- NULL

k <- 10
for(i in 1:k){index <- sample(1:nrow(scaled),round(0.9*nrow(scaled)))
train.cv <- scaled[index,]
test.cv <- scaled[-index,]
nn <- neuralnet(f,data=train.cv,hidden= 4,linear.output=T)
nn1 <- neuralnet(f,data=train.cv,hidden=c(4,3),linear.output=T)
pr.nn <- compute(nn,test.cv[-4])
pr.nn <- pr.nn$net.result*(max(NBA$PER)-min(NBA$PER))+min(NBA$PER)
pr.nn1 <- compute(nn1,test.cv[-4])
pr.nn1 <- pr.nn1$net.result*(max(NBA$PER)-min(NBA$PER))+min(NBA$PER)
test.cv.r <- NBA[-index,]$PER
cv.error1[i] <- (sum((test.cv.r - pr.nn)^2)/nrow(test.cv))^0.5
cv.error2[i] <- (sum((test.cv.r - pr.nn1)^2)/nrow(test.cv))^0.5
}
```

Burada yapmak istediğimiz şey; veriyi 10 kez %90 train, %10 test olarak ayırıp her defada yeni bir RMSE değeri elde etmektir. Bu işlem sonucunda 10 farklı train ve test verisi üzerinde 10 farklı RMSE değeri bulmuş oluyoruz.

Tek katmanlı Yapay Sinir Ağı modelimizin yani ilk modelin 10 foldluk Cross Validation Error değerlerini hesaplayalım;

```
cv.error1
```

```
##  [1] 2.103738 2.105995 2.217205 1.983043 2.022738 2.355369 2.188131 2.057795
##  [9] 2.202447 2.289112
```

İki katmanlı Yapay Sinir Ağı modelimizin yani ikinci modelin 10 foldluk Cross Validation Error değerlerini hesaplayalım;

```
cv.error2
```

```
##  [1] 1.931754 2.015906 2.237837 2.124898 2.014942 2.459503 2.077802 2.168298
##  [9] 2.110665 2.213370
```

```
mean(cv.error1)
```

```
## [1] 2.152557
```

Tek katmanlı Yapay Sinir Ağı modelimizin yani ilk modelin 10 foldluk Cross Validation Error değerlerinin ortalaması 2.152557 çıkmıştır.

```
mean(cv.error2)
```

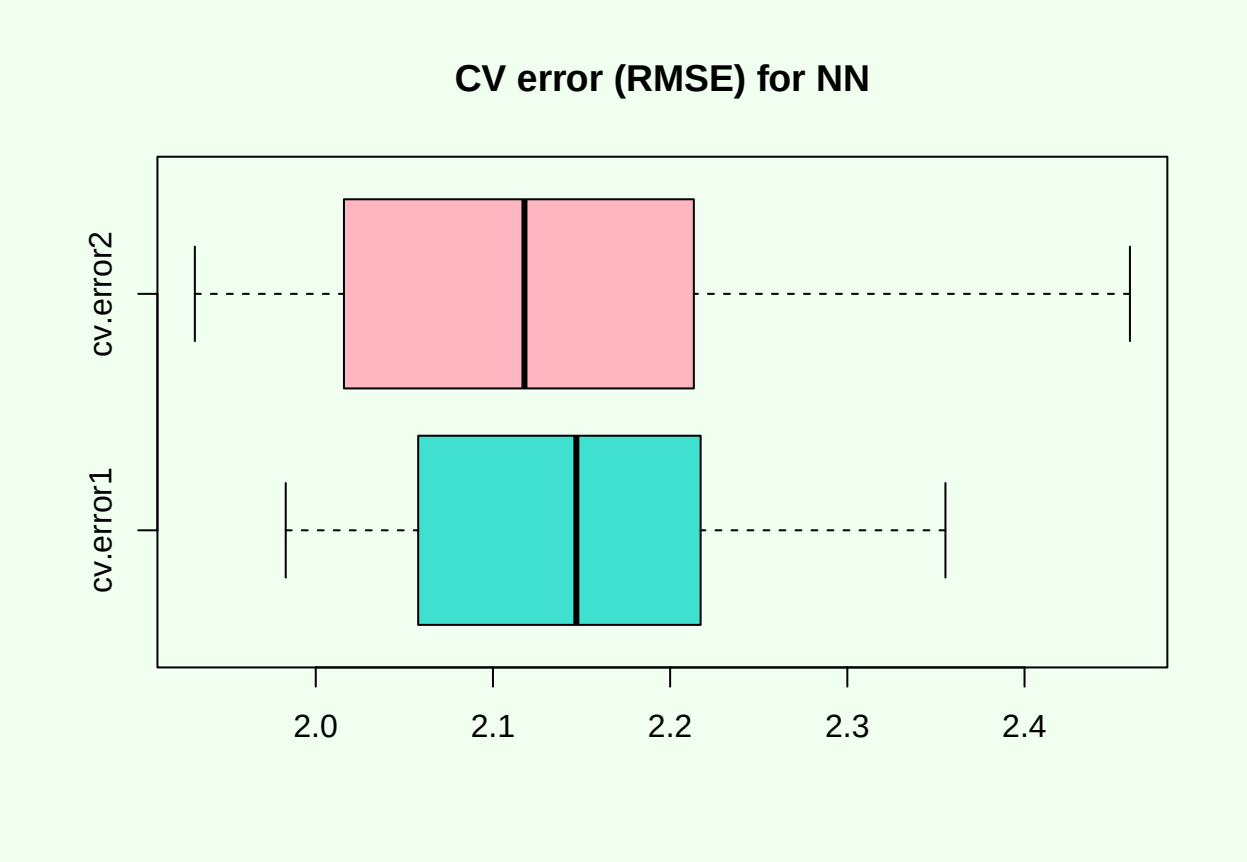
```
## [1] 2.135498
```

Iki katmanlı Yapay Sinir Ağı modelimizin yani ikinci modelin 10 foldluk Cross Validation Error degerlerinin ortalamasi 2.135498 çıkmıştır.

Cross Validation sonucumuza gore iki katmanlı Yapay Sinir Ağı modelimizin yani ikinci modelin 10 foldluk Cross Validation Error değerlerinin ortalaması daha düşük çıkmıştır.şimdi ikinci modelin daha iyi olduğunu söyleyebiliriz.

Yapay Sinir Ağı Modellerimiz üzerinden Cross Validation ile hesaplanan cv.error1 ve cv.error2 değerlerimizin Boxplotunu çizdirelim:

```
op = par(bg = "honeydew")
boxplot(cv.error1,cv.error2, names=c("cv.error1","cv.error2"),main="CV error (RMSE) for NN",
        horizontal=TRUE,col=c("turquoise","lightpink"))
```



Yapay Sinir Ağı modellerimiz üzerinden Cross Validation ile hesaplanan Error degerlerimizin Boxplotuna baktığımızda cv.error1 grafiğinin sola çarpık dağılım gösterdiğini görmekteyiz.

8.ADIM: Cross Validation ile MSE Hesaplama ve Görselleştirme

```
set.seed(121519016)
cv.error <- NULL
k <- 10
for(i in 1:k){index <- sample(1:nrow(scaled),round(0.9*nrow(scaled)))
train.cv <- scaled[index,]
test.cv <- scaled[-index,]
nn <- neuralnet(f,data=train.cv,hidden=4,linear.output=T)
nn1 <- neuralnet(f,data=train.cv,hidden=c(4,3),linear.output=T)
pr.nn <- compute(nn,test.cv[-4])
pr.nn <- pr.nn$net.result*(max(NBA$PER)-min(NBA$PER))+min(NBA$PER)
pr.nn1 <- compute(nn1,test.cv[-4])
pr.nn1 <- pr.nn1$net.result*(max(NBA$PER)-min(NBA$PER))+min(NBA$PER)
test.cv.r <- NBA[-index,]$PER
cv.error[i] <- (sum((test.cv.r - pr.nn)^2)/nrow(test.cv))
}
```

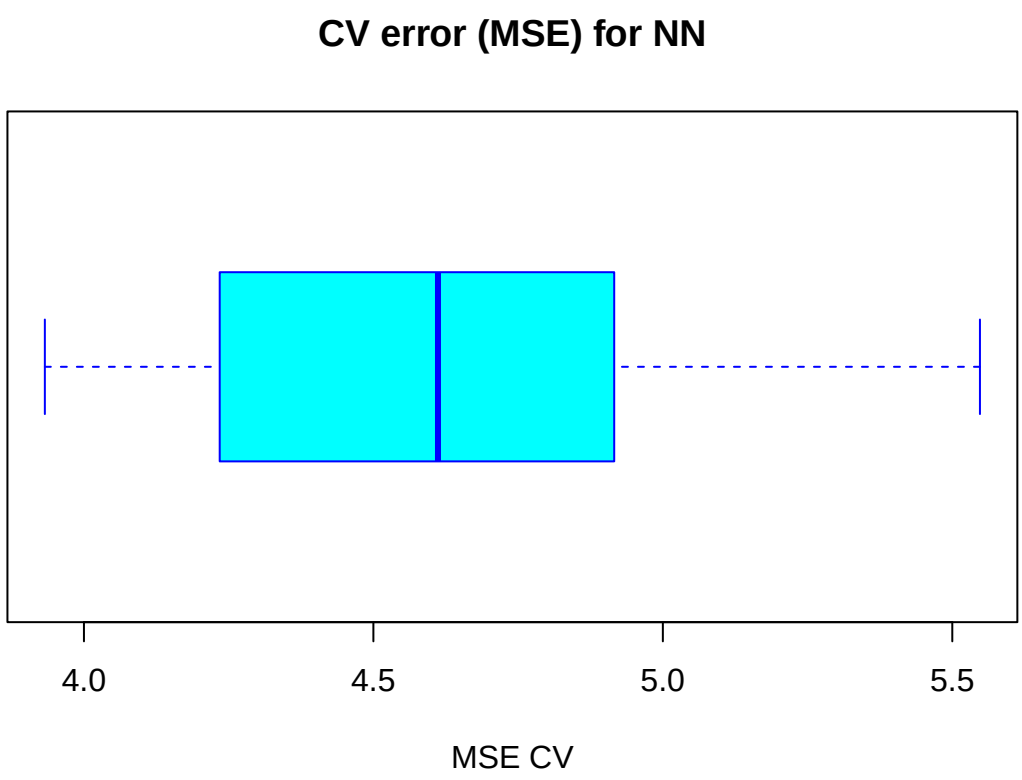
Burada yapmak istediğimiz şey; veriyi 10 kez %90 train, %10 test olarak ayırıp her defada yeni bir MSE değeri elde etmektir. Bu işlem sonucunda 10 farklı train ve test verisi üzerinde 10 farklı MSE değeri bulmuş olduk.

Şimdi bu 10 değerin ortalamasını alalım ve görselleştirelim;

```
mean(cv.error)
```

```
## [1] 4.646186
```

```
boxplot(cv.error,xlab='MSE CV',col='cyan',
border='blue',names='CV error (MSE)',
main='CV error (MSE) for NN',horizontal=TRUE)
```

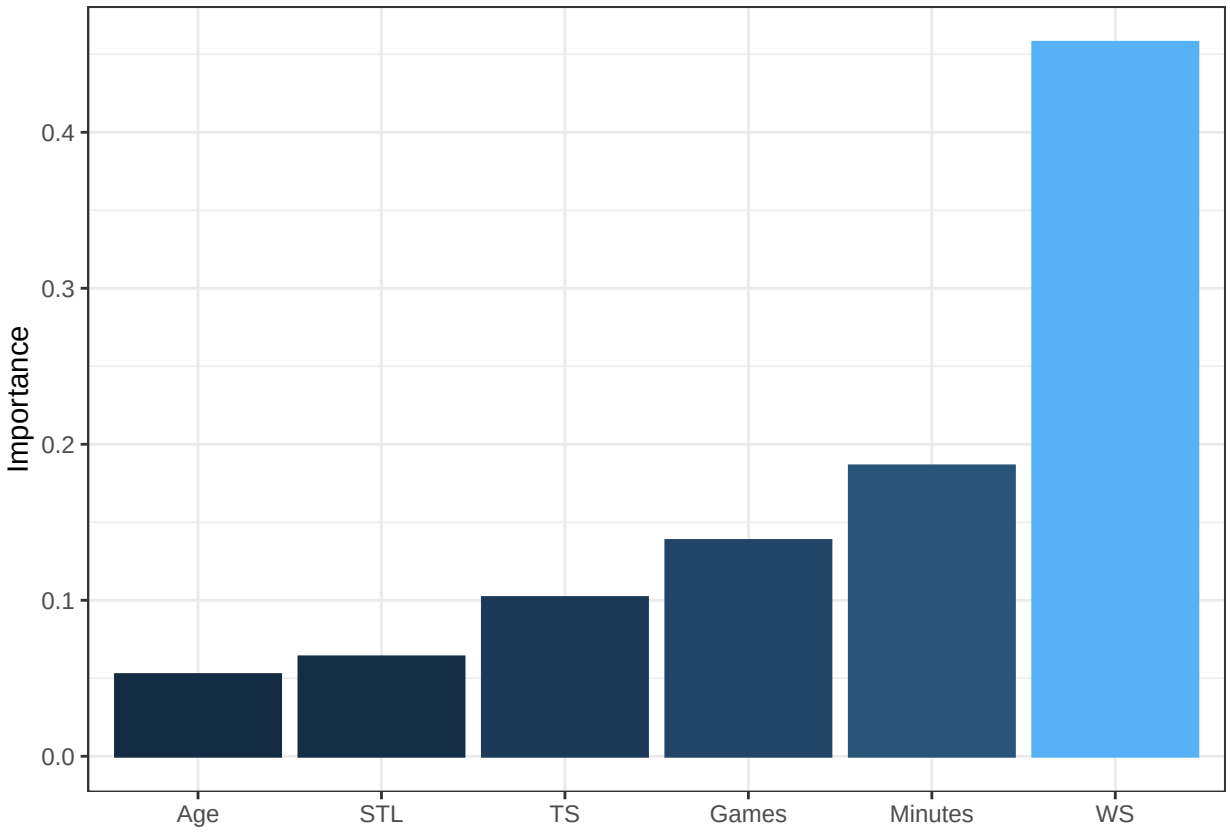


Yapay Sinir Ağı modellerimiz uzerinden Cross Validation ile hesaplanan Error değerlerimizin Boxplotuna baktığımızda cv.error grafiğinin sola çarpık dağılıma sahip olduğu görülmektedir.

9.ADIM: Garson Algoritması ile Parametre Önemi Belirleme

Garson Algoritması, model ağırlıkların yapısını bozarak denetimli bir sinir ağındaki tek bir cevap değişkenleri için açıklayıcı değişkenlerin göreceli önemini tanımlar. Garson algoritması ile yapay sinir ağlarında parametrelerin önemi belirlenir. Algoritma sadece bir gizli katmanı ve bir bağımlı değişkenli kurulmuş sinir ağları modelleri için çalışır. Verimizdeki tahmin doğruluğuna en çok katkı sunan değişkeni görmek için grafik çizdirelim:

```
set.seed(121519016)
nn2 <- neuralnet(f,data = train,hidden = 4,linear.output = FALSE)
garson(nn2)
```



Bu grafikte bağımlı değişkeni etkileyecek en önemli parametleri önem sırası ve önem gücüne göre sıralı görebilmekteyiz. İncelediğimiz veri setindeki tahmin doğruluğuna en çok katkı sunan değişkenin "WS"(Kazanma yüzdesi) değişkeni olduğnu görüyoruz.