ML

2022-11-18

R Markdown

Load data

We will use İris dataset which is already built in R, and is about flowers

```
data(iris)

dataset <- iris
```

Split data to train and test (validation)

```
# create a list of 80% of the row indees in the original dataset we can use for training
validation_index <- createDataPartition(dataset$Species, p=0.80, list=FALSE)
# select 20% of the data for validation
validation <- dataset[-validation_index,]
# use the remaining 80% of data to training and testing the models
dataset <- dataset[validation_index,]
```

a little summary of the data

dimensions

```
dim(dataset)
```

```
## [1] 120   5
```

Types of each variable

```
sapply(dataset, class)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##    "numeric"    "numeric"    "numeric"    "numeric"     "factor"
```

how the data looks like

```
head(dataset)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
## 7          4.6         3.4          1.4         0.3  setosa
```

as species is factor, lets take a look at its levels

```
levels(dataset$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

so we see a multi class data, not binary

lets take a look at the class distribution of each classes

```
#percantage of each class
percentage <- prop.table(table(dataset$Species)) * 100


cbind(freq=table(dataset$Species), percentage=percentage)
```

```
##            freq percentage
## setosa       40   33.33333
## versicolor   40   33.33333
## virginica    40   33.33333
```

```
#table(dataset$Species)
#this gives us frequencies of each unique value in dataset$species

#prop.table(table(dataset$Species))
#prop table gives us proportion of each values in a table, in that case,
#its output sums to one like:
#> prop.table(table(dataset$Species))

#    setosa versicolor  virginica
#0.3333333  0.3333333  0.3333333

#for that reason we multiplied the value by 100
```

so we see that each class is represented equally

Statistical summary of the dataset

```
summary(dataset)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length   Petal.Width          Species
##  Min.   :4.300   Min.   :2.000   Min.   :1.10   Min.   :0.100   setosa    :40
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.60   1st Qu.:0.300   versicolor:40
##  Median :5.800   Median :3.000   Median :4.40   Median :1.350   virginica :40
##  Mean   :5.871   Mean   :3.057   Mean   :3.79   Mean   :1.198
##  3rd Qu.:6.400   3rd Qu.:3.400   3rd Qu.:5.10   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.200   Max.   :6.90   Max.   :2.500
```

Lets make some visualization

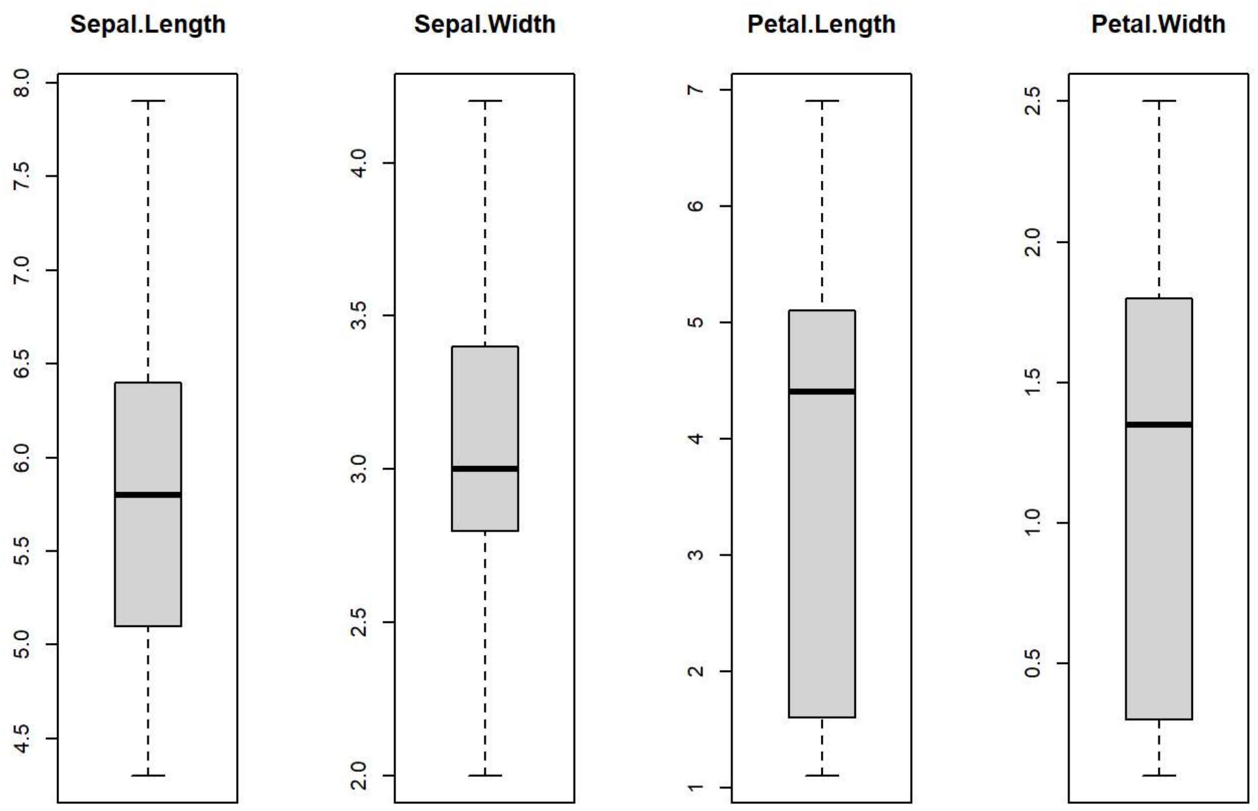
Univariate (one variable) plots

```
#independent variables to x
x <- dataset[,1:4]

#dependent variable to y
y <- dataset[,5]
```

lets see the distribituon of independent variables

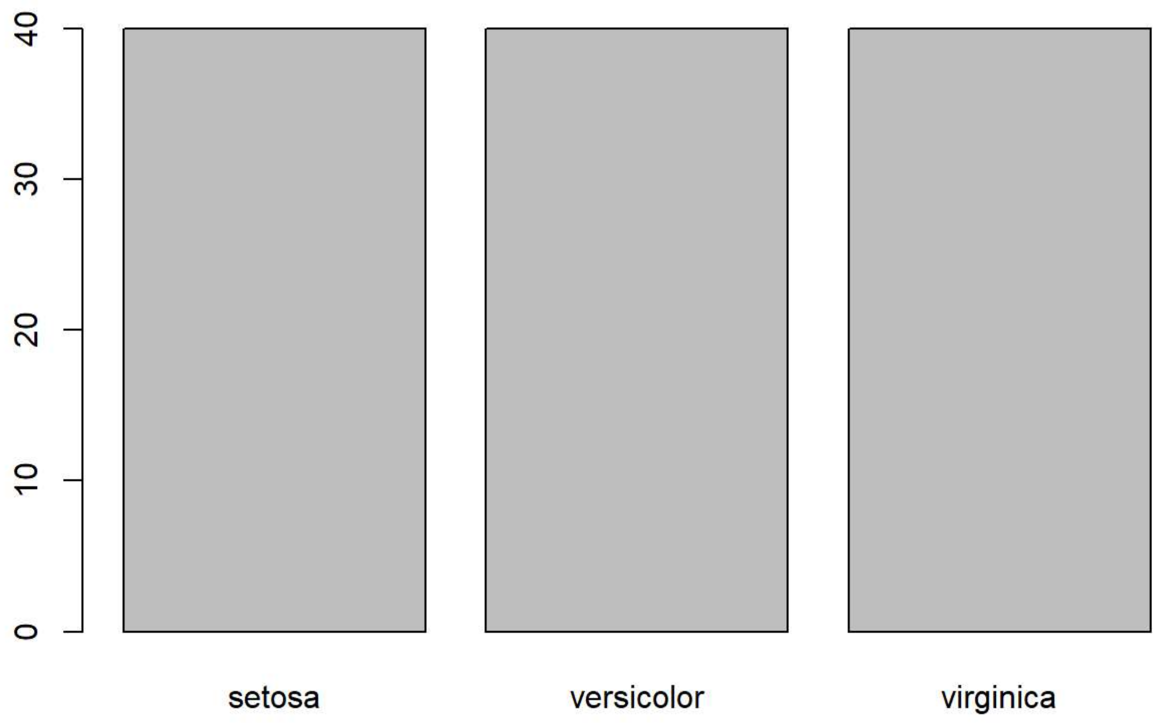
```
#this is for setting the number of boxplots
par(mfrow=c(1,4))
  for(i in 1:4) {
  boxplot(x[,i], main=names(iris)[i])
  }
```



```
#gvisualization of each 4 variable with for loop
#took the names of each boxplot from iris object which is already defined
```

we have the dependent variable as y so lets see its distribution of its different classes

```
plot(y)
```

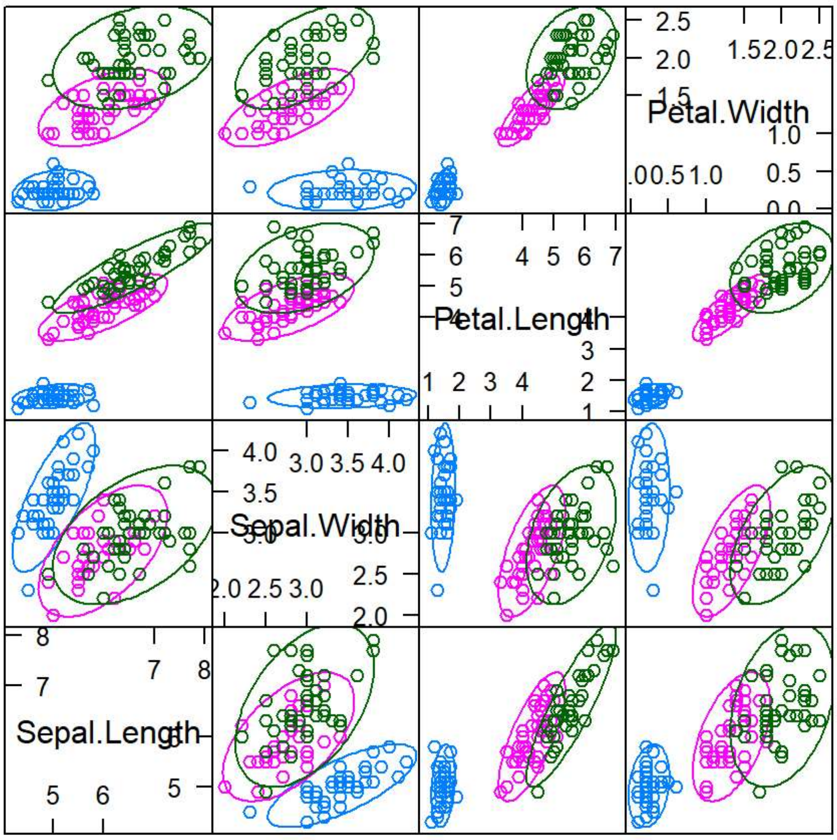


they are equal as we have seen earlier

Multivariate Plots (relationship of variables btw each other)

ScatterplotMatrix

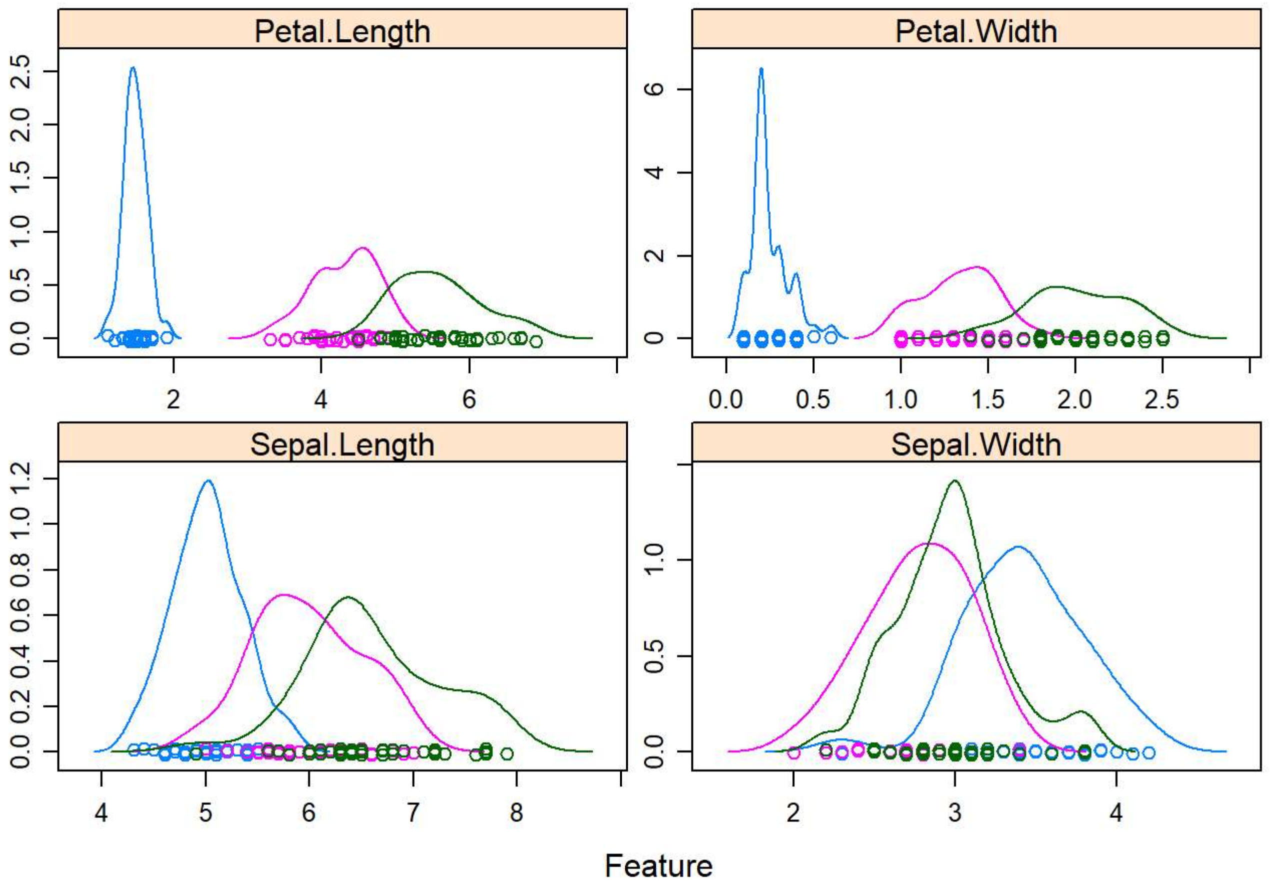
```
featurePlot(x=x, y=y, plot="ellipse")
```



Scatter Plot Matrix

Distribution of each variable with density plot

```
# density plots for each attribute by class value
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, plot="density", scales=scales)
```



this gave us distribution of attributes for each 3 different classes

Making Predictions with 5 different models and evaluating them

We will use 10 fold cross validation and use accuracy as metric

```
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"
```

We will build and make prediction with five different models which are:

Linear Discriminant Analysis (LDA) Classification and Regression Trees (CART). k-Nearest Neighbors (kNN). Support Vector Machines (SVM) with a linear kernel. Random Forest (RF)

```
# a) linear algorithms
set.seed(7)
fit.lda <- train(Species~., data=dataset, method="lda", metric=metric, trControl=control)
# b) nonlinear algorithms
# CART
set.seed(7)
fit.cart <- train(Species~., data=dataset, method="rpart", metric=metric, trControl=control)
# kNN
set.seed(7)
fit.knn <- train(Species~., data=dataset, method="knn", metric=metric, trControl=control)
# c) advanced algorithms
# SVM
set.seed(7)
fit.svm <- train(Species~., data=dataset, method="svmRadial", metric=metric, trControl=contro
l)
# Random Forest
set.seed(7)
fit.rf <- train(Species~., data=dataset, method="rf", metric=metric, trControl=control)
```

Getting, summarizing and comparing results

```
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

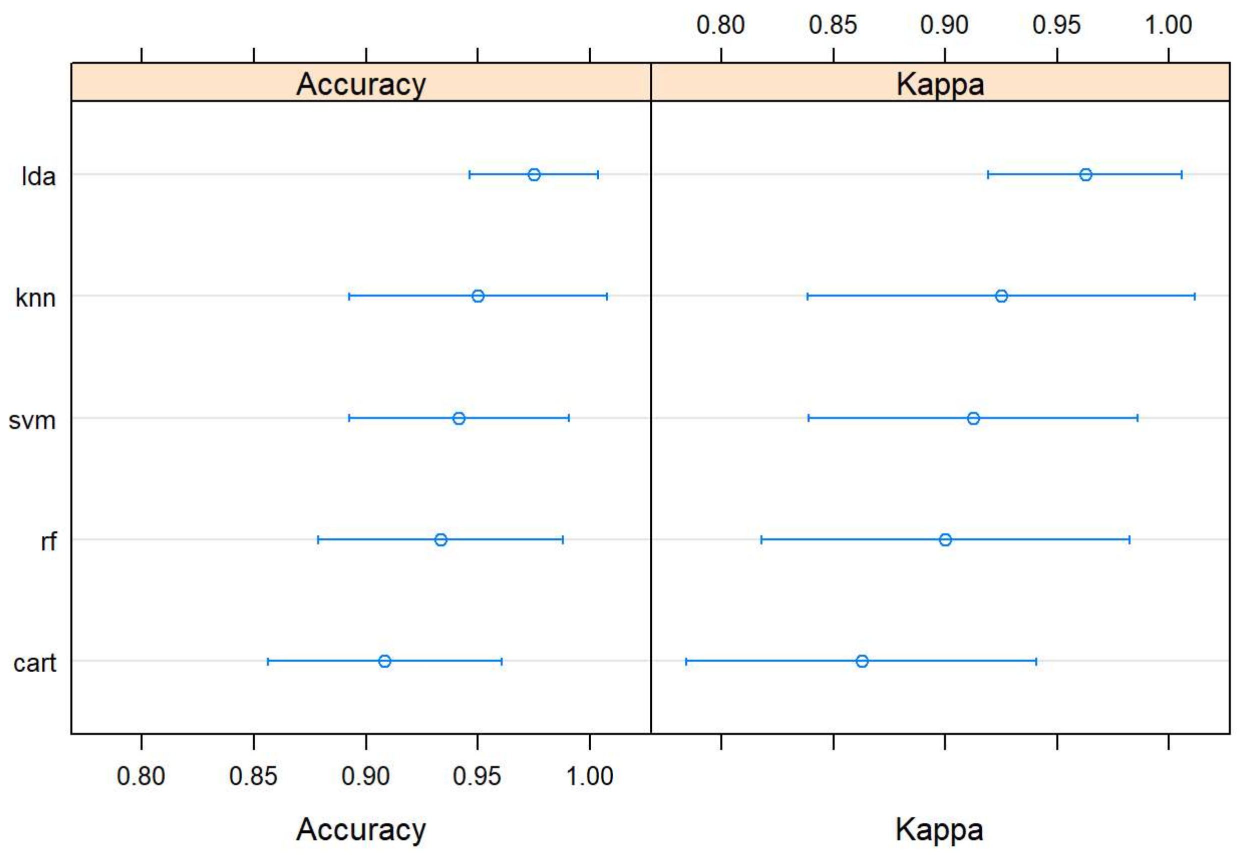
```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## lda  0.9166667 0.9375000 1.0000000 0.9750000 1.0000000    1    0
## cart 0.7500000 0.9166667 0.9166667 0.9083333 0.9166667    1    0
## knn  0.7500000 0.9166667 1.0000000 0.9500000 1.0000000    1    0
## svm  0.8333333 0.9166667 0.9583333 0.9416667 1.0000000    1    0
## rf   0.7500000 0.9166667 0.9166667 0.9333333 1.0000000    1    0
##
## Kappa
##       Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
## lda  0.875 0.90625 1.0000 0.9625   1.000    1    0
## cart 0.625 0.87500 0.8750 0.8625   0.875    1    0
## knn  0.625 0.87500 1.0000 0.9250   1.000    1    0
## svm  0.750 0.87500 0.9375 0.9125   1.000    1    0
## rf   0.625 0.87500 0.8750 0.9000   1.000    1    0
```

result shows us both accuracy metric results and another matric whic is kappa

both metrics shows us that the most accurate model is LDA

we can also visualize the result

```
dotplot(results)
```



Confidence Level: 0.95

we can also see the result for only LDA

```
print(fit.lda)
```

```
## Linear Discriminant Analysis
##
## 120 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.975     0.9625
```

Now that we build the model, we can make predictions and test the results, thus we can see the accuracy of the model with our validation data set which our model did not see and learn yet

making prediction with lda and seeing its accuracy with confusion matrix

```
predictions <- predict(fit.lda, validation)
confusionMatrix(predictions, validation$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          0        10
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.8843, 1)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 4.857e-15
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           1.0000
## Specificity                 1.0000            1.0000           1.0000
## Pos Pred Value              1.0000            1.0000           1.0000
## Neg Pred Value              1.0000            1.0000           1.0000
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.3333
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            1.0000           1.0000
```

lda model predictic every 30 cases in our validation data correctly

İMPORTANT NOTE: caret library does not support model tuning and configuration, and we did not tuned and configured our model here, we only built the model and made prediction

Lastly, we can create a random observation and test it to see how we can predict the unseen cases after we built the model

Create new observation in the form of DataFrame

```
Sepal.Length <- 4.1
Sepal.Width <- 7.4
Petal.Length <- 2.2
Petal.Width <- 3.5

test <- data.frame(Sepal.Length, Sepal.Width, Petal.Length, Petal.Width)
```

Now that we created a test case, we can receive the prediction

```
predictiontest <- predict(fit.lda, test)

print(predictiontest)
```

```
## [1] setosa
## Levels: setosa versicolor virginica
```

As can be seen, the prediction our model gives us is setosa