iris 数据集的分类建模

牛津徽 SA20008048

目录

1 Get Iris from R

```
library(caret)
library(rattle)
data(iris)
dim(iris)
```

```
## [1] 150   5
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
levels(iris$Species)
```

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

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 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
```

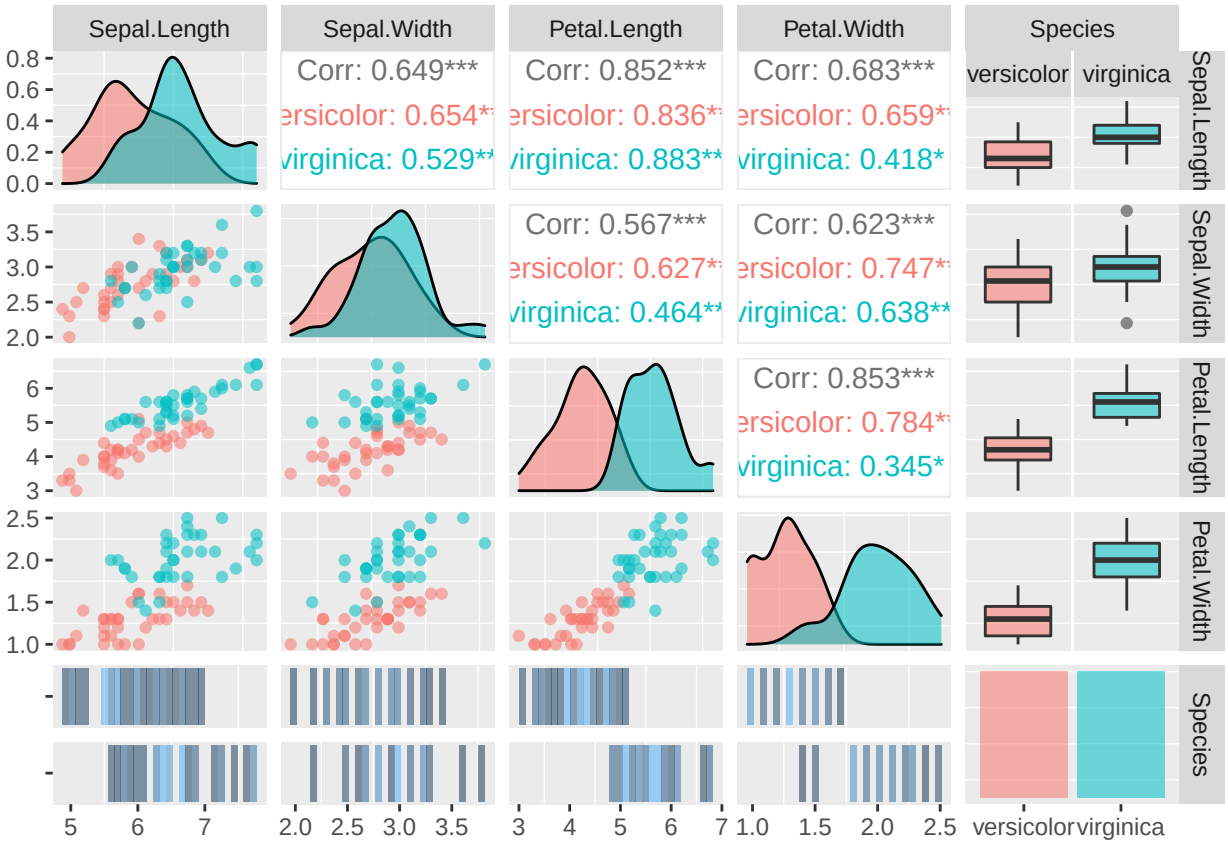
```
a <- subset(iris,Species%in%c("versicolor","virginica"))
#choose versicolor and virginica
a <- droplevels(a)
```

2 Create Training and Testing Sets

```
set.seed(42)
inTrain<-createDataPartition(y=a$Species, p=0.70, list=FALSE)
training.Iris<-a[inTrain,]
testing.Iris<-a[-inTrain,]
```

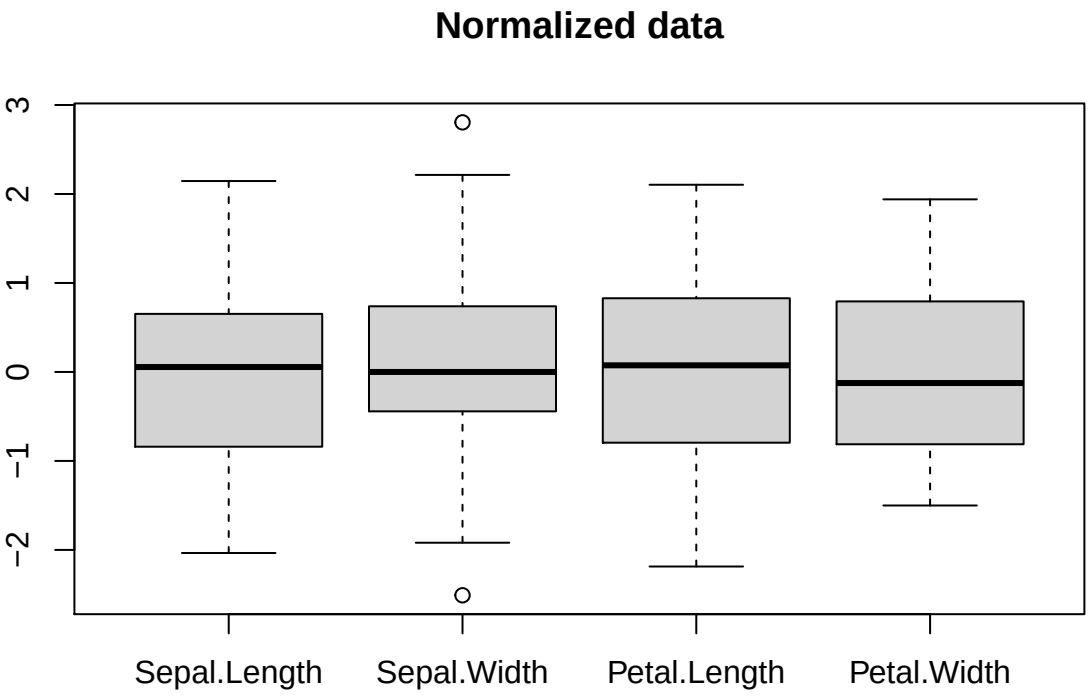
3 Display a pairs plot for the selected variables.

```
training.Iris %>%
  dplyr::mutate(Species=as.factor(Species)) %>%
  GGally::ggpairs(columns=c(1,2,3,4,5),
                  mapping=ggplot2::aes(colour=Species, alpha=0.5),
                  diag=list(continuous="density",
                            discrete="bar"),
                  upper=list(continuous="cor",
                             combo="box",
                             discrete="ratio"),
                  lower=list(continuous="points",
                             combo="denstrip",
                             discrete="facetbar")) +
  ggplot2::theme(panel.grid.major=ggplot2::element_blank())
```

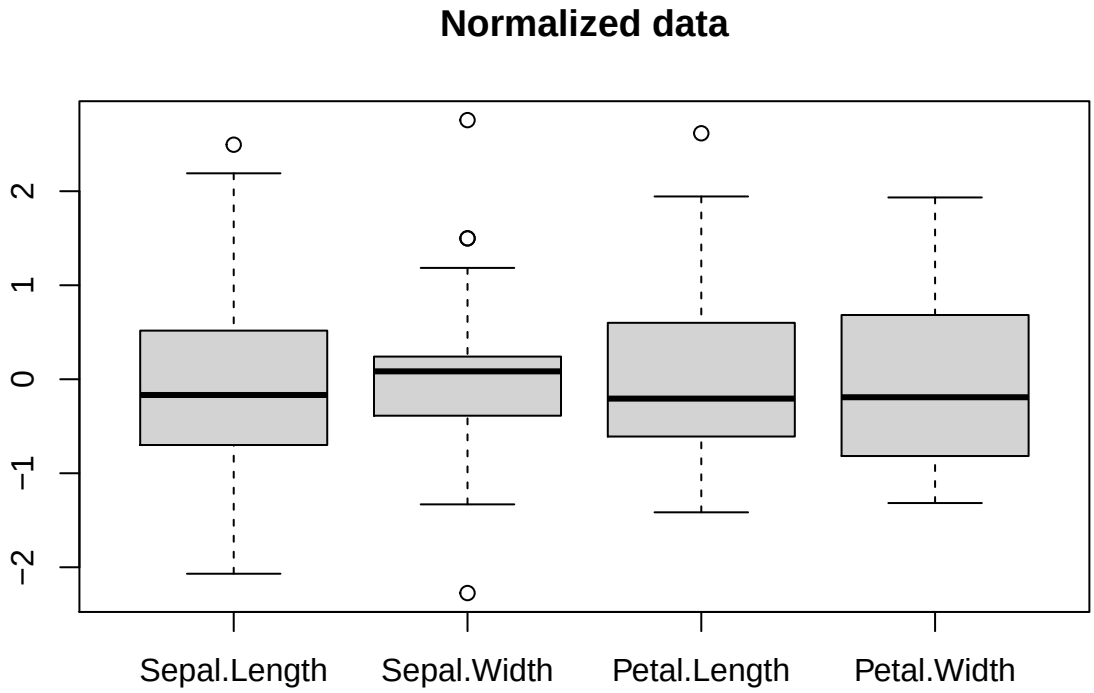

Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
Corr: 0.649***
Corr: 0.852***
Corr: 0.683***
versicolor
virginica
Corr: 0.567***
Corr: 0.623***
Corr: 0.853***
versicolor: 0.654**
versicolor: 0.836**
versicolor: 0.659**
virginica: 0.529**
virginica: 0.883**
virginica: 0.418*
versicolor: 0.627**
versicolor: 0.747**
virginica: 0.464**
virginica: 0.638**
versicolor: 0.784**
virginica: 0.345*
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Species
versicolor
virginica

4 preProcess, center and scale the data

```
#training set
preObj<-preProcess(training.Iris[,-5], method = c("center", "scale"))
preObjData<-predict(preObj,training.Iris[,-5])
boxplot(preObjData, main="Normalized data" )
```




```
training.Iris_N <- transform(preObjData,Species=training.Iris$Species)
#testing set
preObj<-preProcess(testing.Iris[,-5], method = c("center", "scale"))
preObjData<-predict(preObj,testing.Iris[,-5])
boxplot(preObjData, main="Normalized data" )
```




```
testing.Iris_N <- transform(preObjData,Species=testing.Iris$Species)
```

5 Decision Tree

```
## The 'rpart' package provides the 'rpart' function.

library(rpart, quietly=TRUE)

## Reset the random number seed to obtain the same results each time.

set.seed(42)
```

5.1 Build the Decision Tree model.

```
rpart <- rpart(Species ~ .,
                   training.Iris_N,
                   method="class",
                   parms=list(split="information"),
                   control=rpart.control(usesurrogate=0,
                                         maxsurrogate=0),
                   model=TRUE)

## Generate a textual view of the Decision Tree model.

print(rpart)
```

```
## n= 70
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 70 35 versicolor (0.50000000 0.50000000)
##   2) Petal.Length< -0.04140941 32  0 versicolor (1.00000000 0.00000000) *
##   3) Petal.Length>=-0.04140941 38  3 virginica (0.07894737 0.92105263) *
```

5.2 Evaluate model performance on the testing dataset.

5.2.1 Generate an Error Matrix for the Decision Tree model.

```
#### Obtain the response from the Decision Tree model.

pr_rpart <- predict(rpart, newdata=testing.Iris_N,
                    type="class")

#### Generate the confusion matrix showing counts.

rattle::errorMatrix(testing.Iris_N$Species, pr_rpart, count=TRUE)
```

```
##             Predicted
## Actual       versicolor virginica Error
##   versicolor         15         0   0.0
##   virginica           4        11  26.7
```

```
#### Generate the confusion matrix showing proportions.

(per_rpart <- rattle::errorMatrix(testing.Iris_N$Species, pr_rpart))
```

```
##             Predicted
## Actual       versicolor virginica Error
##   versicolor       50.0       0.0   0.0
##   virginica        13.3      36.7  26.7
```

```
#### Calculate the overall error percentage.

cat("Calculate the overall error percentage:",
    100-sum(diag(per_rpart), na.rm=TRUE))
```

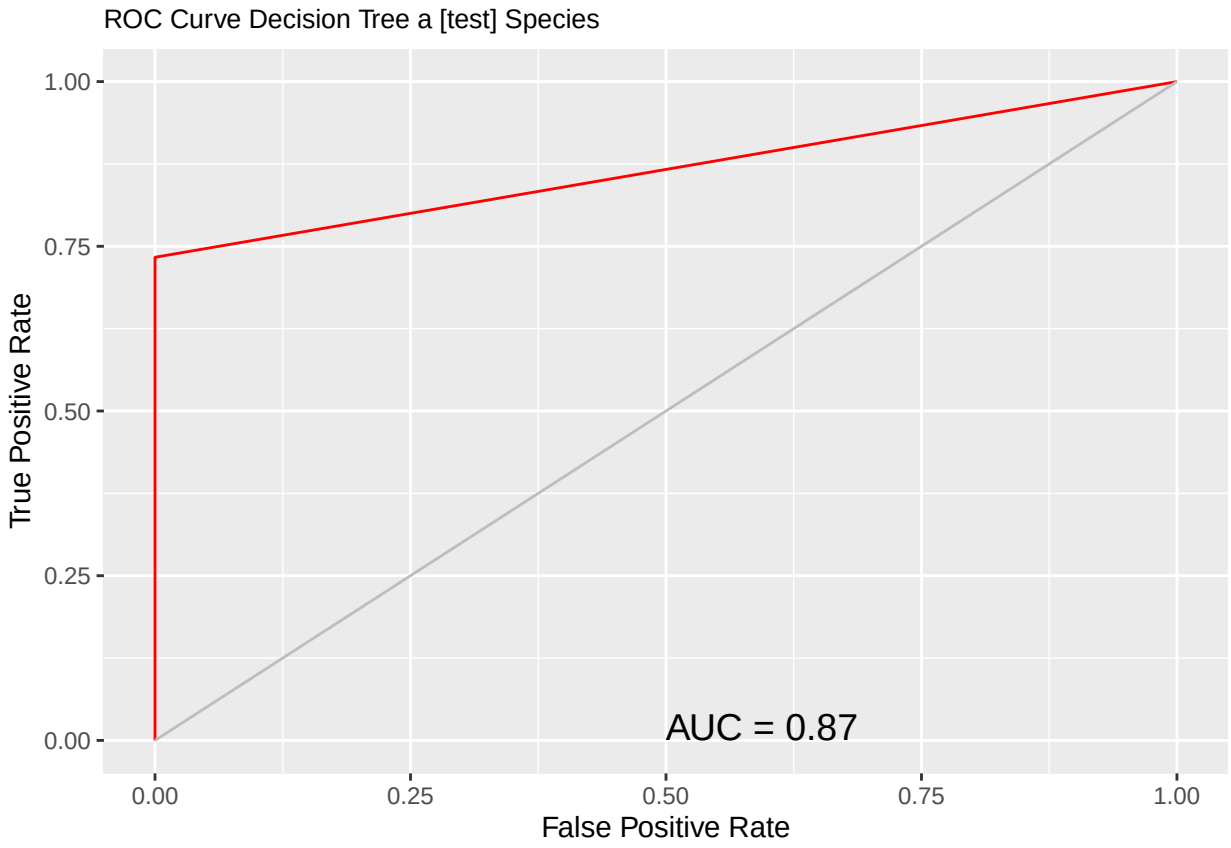
```
## Calculate the overall error percentage: 13.3
```

```
#### Calculate the averaged class error percentage.

cat("Calculate the averaged class error percentage:",
    mean(per_rpart[,"Error"], na.rm=TRUE))
```

```
## Calculate the averaged class error percentage: 13.35
```

5.2.2 ROC Curve: requires the ROCR package.

```
library(ROCR)

#### ROC Curve: requires the ggplot2 package.

library(ggplot2, quietly=TRUE)

#### Generate an ROC Curve for the rpart model on a [test].

pr_rpart_roc <- predict(rpart, newdata=testing.Iris_N)[,2]

#### Remove observations with missing target.

no.miss   <- na.omit(testing.Iris_N$Species)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(pr_rpart_roc[-miss.list], no.miss)
} else
{
  pred <- prediction(pr_rpart_roc, no.miss)
}

pe <- performance(pred, "tpr", "fpr")
au <- performance(pred, "auc")@y.values[[1]]
pd <- data.frame(fpr=unlist(pe@x.values), tpr=unlist(pe@y.values))
p <- ggplot(pd, aes(x=fpr, y=tpr))
p <- p + geom_line(colour="red")
p <- p + xlab("False Positive Rate") + ylab("True Positive Rate")
p <- p + ggtitle("ROC Curve Decision Tree a [test] Species")
p <- p + theme(plot.title=element_text(size=10))
p <- p + geom_line(data=data.frame(), aes(x=c(0,1), y=c(0,1)), colour="grey")
p <- p + annotate("text", x=0.50, y=0.00, hjust=0, vjust=0, size=5,
                  label=paste("AUC =", round(au, 2)))
print(p)
```




```
#### Calculate the area under the curve for the plot.

no.miss   <- na.omit(testing.Iris_N$Species)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(pr_rpart_roc[-miss.list], no.miss)
} else
{
  pred <- prediction(pr_rpart_roc, no.miss)
}
performance(pred, "auc")
```

```
## A performance instance
##   'Area under the ROC curve'
```

6 Support vector machine.

```
## The 'kernlab' package provides the 'ksvm' function.

library(kernlab, quietly=TRUE)
```

6.1 Build a Support Vector Machine model.

```
set.seed(42)
ksvm <- ksvm(as.factor(Species) ~ .,
                 data=training.Iris_N,
                 kernel="rbfdot",
                 prob.model=TRUE)

## Generate a textual view of the SVM model.

ksvm
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.43890636546426
##
## Number of Support Vectors : 31
##
## Objective Function Value : -11.5446
## Training error : 0.014286
## Probability model included.
```

6.2 Evaluate model performance on the testing dataset.

6.2.1 Generate an Error Matrix for the SVM model.

```
#### Obtain the response from the SVM model.

pr_ksvm <- kernlab::predict(ksvm, newdata=na.omit(testing.Iris_N))

#### Generate the confusion matrix showing counts.

rattle::errorMatrix(na.omit(testing.Iris_N)$Species, pr_ksvm, count=TRUE)
```

```
##             Predicted
## Actual       versicolor virginica Error
##   versicolor         15         0   0.0
##   virginica           1        14   6.7
```

```
#### Generate the confusion matrix showing proportions.

(per_ksvm <- rattle::errorMatrix(na.omit(testing.Iris_N)$Species, pr_ksvm))
```

```
##             Predicted
## Actual       versicolor virginica Error
##   versicolor       50.0       0.0   0.0
##   virginica         3.3      46.7   6.7
```

```
#### Calculate the overall error percentage.

cat("Calculate the overall error percentage:",
    100-sum(diag(per_ksvm), na.rm=TRUE))
```

```
## Calculate the overall error percentage: 3.3
```

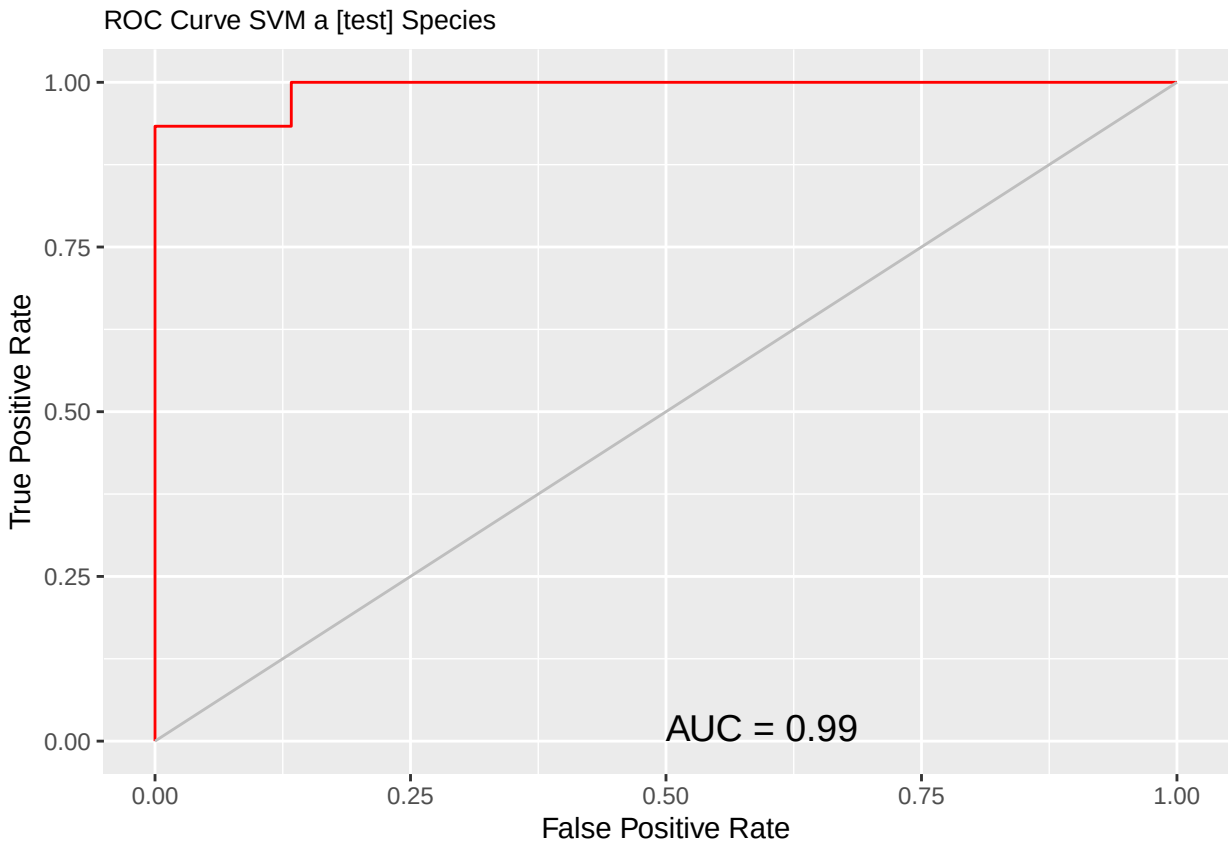
```
#### Calculate the averaged class error percentage.

cat("Calculate the averaged class error percentage:",
    mean(per_ksvm[,"Error"], na.rm=TRUE))
```

```
## Calculate the averaged class error percentage: 3.35
```

6.2.2 ROC Curve: requires the ROCR package.

```
library(ROCR)

#### ROC Curve: requires the ggplot2 package.

library(ggplot2, quietly=TRUE)

#### Generate an ROC Curve for the ksvm model on a [test].

pr_ksvm_roc <- kernlab::predict(ksvm, newdata=na.omit(testing.Iris_N),
                            type    = "probabilities")[,2]

#### Remove observations with missing target.

no.miss   <- na.omit(na.omit(testing.Iris_N)$Species)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred <- prediction(pr_ksvm_roc[-miss.list], no.miss)
} else
{
  pred <- prediction(pr_ksvm_roc, no.miss)
}

pe <- performance(pred, "tpr", "fpr")
au <- performance(pred, "auc")@y.values[[1]]
pd <- data.frame(fpr=unlist(pe@x.values), tpr=unlist(pe@y.values))
p <- ggplot(pd, aes(x=fpr, y=tpr))
p <- p + geom_line(colour="red")
p <- p + xlab("False Positive Rate") + ylab("True Positive Rate")
p <- p + ggtitle("ROC Curve SVM a [test] Species")
p <- p + theme(plot.title=element_text(size=10))
p <- p + geom_line(data=data.frame(), aes(x=c(0,1), y=c(0,1)), colour="grey")
p <- p + annotate("text", x=0.50, y=0.00, hjust=0, vjust=0, size=5,
                  label=paste("AUC =", round(au, 2)))
```

```
print(p)
```




```
#### Calculate the area under the curve for the plot.

no.miss   <- na.omit(testing.Iris_N$Species)
miss.list <- attr(no.miss, "na.action")
attributes(no.miss) <- NULL

if (length(miss.list))
{
  pred_ksvm <- prediction(pr_ksvm_roc[-miss.list], no.miss)
} else
{
  pred_ksvm <- prediction(pr_ksvm_roc, no.miss)
}
performance(pred_ksvm, "auc")

## A performance instance
##   'Area under the ROC curve'
```