การติดตั้งและการใช้ ภาษา R ใน Machine Learning

อ้างอิง: Prediction: Machine Learning and Statistics (MIT OPEN COURSEWARESpring 2012)

<u>วิธีติดตั้ง R</u>

1. เข้าไปที่web CRAN เพื่อติดตั้งตัวแปลภาษา interpreter (ซึ่งทำงานด้วยภาษาซี)
2. ติดตั้ง RStudioเพื่อใช้เป็น editor

<u>วิธีใช้งานมี2 แบบคือ</u>

1. วิธีใช้งานเฉพาะตัวแปลภาษา (ไม่ใช้ RStudio editor) ใช้ขั้วคราว ง่ายๆ
- คลิก Ri386
- จะปรากฏหน้าต่าง ให้พิมพ์คำสั่งลงไป เช่น

2*3

- ต้องการพิมพ์โปรแกรม (ชุดคำสั่ง, สคริป)

กด File → New Script →พิมพ์ชุดคำสั่ง เช่น

```
v<- 1:5
m<- c(1,2,3,4)
n<- seq(from=10, to=50, by=2)
print(v)
print(m)
print(n)
```

กด Edit -> Run all เพื่อให้แสดงผลลัพธ์

2. วิธีใช้งานด้วยRStudio ใช้ทำงานที่ซับซ้อน

2.1 การกำหนด Working directory

- สร้าง folder ที่ต้องการเก็บโปรแกรม เช่น C:\R_code เอาไว้ก่อน
- ใช้ RStudio
- ถ้าต้องการเปลี่ยน home directory (เพื่อทำการเก็บไฟล์โปรแกรม)

2.2 เลือก Tools→ Global → General →Default Working → Browse

เลือก directory ที่ต้องการเก็บโปรแกรม เช่น C:\R_code

- ออกจาก RStudioโดย File → Quit

<u>ใช้งาน RStudioทำงาน</u><u>(นิยมใช้ที่สุด)</u>

1. เปิด RStudio
2. เลือก File → New file →RScript
3. พิมพ์โปรแกรม ในหน้าจอบนซ้าย เช่น

```
v<-1:5

v <-c(1,2,3,4,5) # c can be used to concatenate multiple vectors

 v <-seq(from=1,to=5,by=1)# same as above
```

4. เลือก Code → Run Region → Run All (Ctrl+Alt+R)
5. บันทึกไฟล์

File → Save As →ชื่อไฟล์.r

<u>การสั่งให้ R ทำงาน</u><u>มี 2 วิธีคือ (ใน RStudio)</u>

1. สั่งด้วยชุดโปรแกรม (บันทึกเป็น file) ดังกล่าวไปแล้ว
2. สั่งด้วยคำสั่งเดี่ยว ๆ ในแต่ละหน้าต่าง Console (ล่างซ้าย) เช่น

```
> 2*3
```

และสามารถใช้ ลูกศรขึ้นลง เพื่อเรียกคำสั่งก่อนหน้านี้

<u>คำสั่งของระบบที่น่าสนใจ</u>

```
> ? fn            #สอบถามการใช้งานคำสั่ง fn
>example(fn)       #แสดงตัวอย่างของ fn
```

```
>args(fn)                #แสดง argument
>?? "fn"                 #ค้นหาคำที่เกี่ยวข้องกับ fn
>data( )                 #แสดง dataset ที่มีให้ใช้
>data( cars)             #นำ dataset ชื่อ cars มาใช้งาน
>? cars          #ดูรายละเอียด dataset ชื่อ cars
>cars            #ดูข้อมูลของ cars
```

[เรียบเรียงจาก R for Machine Learning ของ Chang]

ฟังก์ชันพื้นฐาน

ประกอบด้วยการสร้างตารางข้อมูล, การวิเคราะห์และแสดงกราฟ ของข้อมูล

3.1 การสร้างข้อมูล กำหนดข้อมูลตัวแปลโดย ใช้ <- เช่น

```
>  v1<- 1:5                                    #ข้อมูลเวกเตอร์
>  v2<- c(3,6,1,2,9)
>  v3<- seq(from=1, to=5, by=1)
>  v4<- c(0,0,0,0,0)
>  v5<- seq(from=0, to=0,length.out=5)         #มีค่าเท่ากับ v4
```

สามารถสร้าง matrix โดยใช้ cbindเพื่อต่อกันในแนวcolumn และ rbindเพื่อต่อกันในแนวrow เชื่อมชุดข้อมูลเวกเตอร์

```
>cbind(v0, v1, v2, v3, v4, v5)              #จะได้ผลลัพธ์อย่างไร
>rbind(v0, v1, v2, v3, v4, v5)              #จะได้ผลลัพธ์อย่างไร
```

วิธีสร้างเมตริกซ์อีกอย่างหนึ่งคือ กำหนดเวกเตอร์ก่อนแล้วจึงระบุชนิดเมตริกซ์

```
>  v<- seq(from=1, to=20, by=1)     # first:  create the vector
>  m1=matrix(v, nrow=4, ncol=5)      # second: set to the matrix
>  m2=matrix(v, nrow=4, ncol=5, byrow=TRUE) #หรือ จัดข้อมูลทีละแถว
>colnames(m2) <- C("Col1", "Col2", "Col3", "Col4", "Col5") #ใส่ชื่อ column
>rownames(m2) <- C("Row1", "Row 2", "Row 3", "Row 4") #ใส่ชื่อแถว
```

คำถาม m1 และ m2 เป็นชนิดข้อมูลชนิดใด ?

การเข้าถึงเวกเตอร์และเมตริกซ์โดยใช้ [ ] เช่น

```
>  v[3]                  #เอาข้อมูลตัวที่ 3 ของ เวกเตอร์ v
>  m2[, "Col2"] #เอาเฉพาะคอลัมน์ Col2  ของเมตริกซ์ m2
>  m2["Row4" ,]          #เอาเฉพาะแถว Row4
>  m2["Row4", "Col2"]            #เอาเฉพาะแถว Row4  และ คอลัมน์ Col2  ของเมตริกซ์ m2
>  m2[4,2]               #แถวที่ 4 คอลัมน์ที่ 2
>  length(v)             #ความยาว v
>nrow(m2)                #จำนวนแถวของ m2
>ncol(m2)                #จำนวนหลักของ m2
```

การดึงข้อมูลจาก Text file เข้ามาเป็นข้อมูล (save "haberman.data" ไว้ก่อน มีตัวคั่นข้อมูลคือ ,)

```
>  dataset<- read.table("C:\\Datasets\\haberman.data", header=FALSE, sep=",") #ตัวคั่นข้อมูลคือ , และไม่มี หัวตาราง
>  dataset2<- read.csv("C:\\Datasets\\haberman.csv", header=FALSE)  #ผลลัพธ์เหมือนข้างบน
```

การสุ่มจากความน่าจะเป็นที่ระบุไว้แล้ว

คือการสร้างชุดตัวเลขสุ่มที่มีการกระจายชนิดต่าง ๆ เช่น normal, exponential, poisson, uniform, binomial และเราต้องกำหนดพารามิเตอร์เพิ่มเติมเท่าที่จำเป็น เช่น

>norm_vec<-rnorm(n=10, mean=5, sd=2)

>exp_vec<-rexp(n=100, rate=3)

>pois_vec<-rpois(n=50, lambda=6)

>unif_vec<-runif(n=20, min=1, max=9)

>bin_vec<-rbinom(n=20, size=1000, prob=0.7)

ถ้าต้องการสุ่มแบบ random จำนวน 25 ตัวจากชุดข้อมูลทำดังนี้

> sample(vec, size25, replace=FALSE)

ถ้าต้องการให้สุ่มทุกครั้งแล้วได้ลำดับเท่าเดิมกำหนดด้วย set.seed(n) เช่น

>set.seed(10)

การวิเคราะห์ข้อมูล

mean()	ใช้หาค่าเฉลี่ย
var()	ใช้หาค่าความแปรปรวน
sd()	ใช้หาค่าความเบี่ยงเบนมาตรฐาน
min() , max()	ใช้หาค่าน้อยที่สุด, มากที่สุด
sum()	ใช้หาค่าผลรวม
rowSum() , colSum	ใช้หาค่าผลรวมแนวแถว, หลัก

การเขียนโปรแกรม

สามารถใช้คำสั่งควบคุม เช่น for, if, while, repeat รายละเอียดคือ

(จาก Introduction to R : Control Structures and Loopsและ Quick-R Control structuresและ Multivariate Statistical TechniquesMatrix Operations in R)

รูปแบบ คำสั่ง	ตัวอย่าง
Operator Description + addition - subtraction * multiplication / division ^ or ** exponentiation x %% y modulus (x mod y) 5%%2 is 1 x %/% y integer division 5%/%2 is 2	

Operator Description
< less than
<= less than or equal to
	# An example
	x <- c(1:10)
	x[(x>8) | (x<5)]
	# output = 1 2 3 4 9 10
	
	# How it works

>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

```
x <- c(1:10)
x
# Output: 1 2 3 4 5 6 7 8 9 10
x > 8
# Output: FFFFFFFFTT
x < 5
# Output: TTTTFFFFFF
x > 8 | x < 5
# Output: TTTTFFFFTT
x[c(T,T,T,T,F,F,F,T,T)]
1 2 3 4 9 10
3 > 2; # Output: TRUE
1:10 > 2;
# Output: FALSE, FALSE, TRUE, ..., TRUE
```

if (*cond*) *expr*
if (*cond*) *expr1* else *expr2*

```
x <- 10;
if(x > 9)
{
        # Execute this only if x > 9
        print("x is larger than 9");
}
```

ifelse(*test*,*yes*,*no*)

```
x <- c(6:-4)
ifelse(x >= 0, sqrt(x), NA)
## NA= the output of unavailable sqrt !
```

switch(*expr*, ...)

```
centre<-function(x, type){
switch(type,
        mean = mean(x),
        median = median(x),
        trimmed = mean(x, trim =.1))
}
```

for (*var* in *seq*) *expr*

```
for(i in 1:10)
{
        print(paste("i =", i));
}
```

while (*cond*) *expr*

```
p <- 0.5; # Probability of drawing a 1
b <- 0; # Result of draw
number<- 0; # Number of draws.
while(b != 1)
{b <- rbinom(1, 1, p); #Draw Bernoulli RV
number <- number + 1; # Increment
}
print(number);
```

myfunction<- function(*arg1, arg2, ...*){
statements
return(*object*)
}

```
norm <- function(x) sqrt(x%*%x)
norm(1:4)    #callfnfor matrix sqrt
mynorm<- function(x) sqrt(x*x)
mynorm(2) #cal for normal sqrt
```

การทำงานกับเมตริซ์ (Matrix)

```
# the matrix function

# R wants the data to be entered by columns starting with column one

# 1st arg: c(2,3,-2,1,2,2) the values of the elements filling the columns

# 2nd arg: 3 the number of rows

# 3rd arg: 2 the number of columns

> A <- matrix(c(2,3,-2,1,2,2),3,2)

> A

     [,1] [,2]

[1,]    2    1

[2,]    3    2

[3,]   -2    2

# Test: Is it a Matrix

>is.matrix(A)

[1] TRUE

>is.vector(A)

[1] FALSE

#  Multiplication by a Scalar

> c <- 3

>c*A

     [,1] [,2]

[1,]    6    3

[2,]    9    6

[3,]   -6    6

#  Matrix Addition & Subtraction

> B <- matrix(c(1,4,-2,1,2,1),3,2)

> B

     [,1] [,2]

[1,]    1    1

[2,]    4    2

[3,]   -2    1

> C <- A + B

> C

     [,1] [,2]

[1,]    3    2

[2,]    7    4

[3,]   -4    3

> D <- A - B

> D

     [,1] [,2]

[1,]    1    0

[2,]   -1    0

[3,]    0    1

# Matrix Multiplication

> D <- matrix(c(2,-2,1,2,3,1),2,3)

> D

     [,1] [,2] [,3]

[1,]    2    1    3

[2,]   -2    2    1

> C <- D %*% A

> C

     [,1] [,2]

[1,]    1   10

[2,]    0    4

> C <- A %*% D

> C

     [,1] [,2] [,3]

[1,]    2    4    7

[2,]    2    7   11

[3,]   -8    2   -4

> D <- matrix(c(2,1,3),1,3)

> D

     [,1] [,2] [,3]

[1,]    2    1    3

> C <- D %*% A

> C

     [,1] [,2]

[1,]    1   10

> C <- A %*% D  #why is this an ERROR ?

Error in A %*% D : non-conformable arguments

# Transpose of a Matrix

> AT <- t(A)

> AT

     [,1] [,2] [,3]

[1,]    2    3   -2

[2,]    1    2    2

> ATT <- t(AT)

>ATT
```

```
     [,1] [,2]
[1,]    2    1
[2,]    3    2
[3,]   -2    2
## Common Vectors
# Unit Vector
> U <- matrix(1,3,1)
> U
     [,1]
[1,]    1
[2,]    1
[3,]    1
# Zero Vector
> Z <- matrix(0,3,1)
> Z
     [,1]
[1,]    0
[2,]    0
[3,]    0
# Common Matrices
# Unit Matrix , all element=1
> U <- matrix(1,3,2)
> U
     [,1] [,2]
[1,]    1    1
[2,]    1    1
[3,]    1    1
# Zero Matrix , all element=0
> Z <- matrix(0,3,2)
> Z
     [,1] [,2]
[1,]    0    0
[2,]    0    0
[3,]    0    0
#  Diagonal Matrix
> S <- matrix(c(2,3,-2,1,2,2,4,2,3),3,3)
> S
     [,1] [,2] [,3]
[1,]    2    1    4
[2,]    3    2    2
[3,]   -2    2    3
> D <- diag(S)  #  Diagonal Matrix
> D
[1] 2 2 3
> D <- diag(diag(S))
> D
     [,1] [,2] [,3]
[1,]    2    0    0
[2,]    0    2    0
[3,]    0    0    3
#  Identity Matrix
> I <- diag(c(1,1,1)) # Incorrect, please correct here
> I
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
#  Symmetric Matrix
> C <- matrix(c(2,1,5,1,3,4,5,4,-2),3,3)
> C
     [,1] [,2] [,3]
[1,]    2    1    5
[2,]    1    3    4
[3,]    5    4   -2
> CT <- t(C)  #What is this, please explain
> CT
     [,1] [,2] [,3]
[1,]    2    1    5
[2,]    1    3    4
[3,]    5    4   -2
#  Inverse of a Matrix
> A <- matrix(c(4,4,-2,2,6,2,2,8,4),3,3)
> A
     [,1] [,2] [,3]
[1,]    4    2    2
[2,]    4    6    8
[3,]   -2    2    4
> AI <- solve(A)  #What is this, please explain
> AI
     [,1] [,2] [,3]
```

```
[1,]  1.0 -0.5  0.5
[2,] -4.0  2.5 -3.0
[3,]  2.5 -1.5  2.0
> A %*% AI  #What is this, please explain
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> AI %*% A
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
# Inverse & Determinant of a Matrix
> C <- matrix(c(2,1,6,1,3,4,6,4,-2),3,3)
> C
     [,1] [,2] [,3]
[1,]    2    1    6
[2,]    1    3    4
[3,]    6    4   -2
> CI <- solve(C)
CI
          [,1]       [,2]       [,3]
[1,]  0.2156863 -0.25490196  0.13725490
[2,] -0.2549020  0.39215686  0.01960784
[3,]  0.1372549  0.01960784 -0.04901961
> d <- det(C)
>d
[1] -102
# Rank of a Matrix
> A <- matrix(c(2,3,-2,1,2,2,4,7,0),3,3)
> A
     [,1] [,2] [,3]
[1,]    2    1    4
[2,]    3    2    7
[3,]   -2    2    0
>matA<- qr(A) #What is this, please explain
>matA$rank #What is this, please explain
[1] 3
> A <- matrix(c(2,3,-2,1,2,2,4,6,-4),3,3)
> A
     [,1] [,2] [,3]
[1,]    2    1    4
[2,]    3    2    6
[3,]   -2    2   -4
>matA<- qr(A)
>matA$rank
[1] 2
# note column 3 is 2 times column 1
#  Number of Rows & Columns
> X <- matrix(c(3,2,4,3,2,-2,6,1),4,2)
> X
     [,1] [,2]
[1,]    3    2
[2,]    2   -2
[3,]    4    6
[4,]    3    1
>dim(X)
[1] 4 2
> r <- nrow(X)
>r
[1] 4
> c <- ncol(X)
>c
[1] 2
#  Computing Column & Row Sums
# note the uppercase S
> A <- matrix(c(2,3,-2,1,2,2),3,2)
> A
     [,1] [,2]
[1,]    2    1
[2,]    3    2
[3,]   -2    2
> c <- colSums(A)
>c


[1] 3 5


> r <- rowSums(A)
>r
```

```
[1] 3 5 0
> a <- sum(A)
>a
[1] 8
#  Computing Column & Row Means
# note the uppercase M
> cm <- colMeans(A)
> cm
[1] 1.000000 1.666667
>rm<- rowMeans(A)
>rm
[1] 1.5 2.5 0.0
> m <- mean(A)
>m
[1] 1.333333
#  Horizontal Concatenation
> A
     [,1] [,2]
[1,]    2    1
[2,]    3    2
[3,]   -2    2
```

```
> B <- matrix(c(1,3,2,1,4,2),3,2)
> B
     [,1] [,2]
[1,]    1    1
[2,]    3    4
[3,]    2    2
> C <- cbind(A,B)
> C
     [,1] [,2] [,3] [,4]
[1,]    2    1    1    1
[2,]    3    2    3    4
[3,]   -2    2    2    2
#  Vertical Concatenation (Appending)
> C <- rbind(A,B)
> C
     [,1] [,2]
[1,]    2    1
[2,]    3    2
[3,]   -2    2
[4,]    1    1
[5,]    3    4
[6,]    2    2
```

<u>การเลือกข้อมูลจากเวกเตอร์</u>เช่น

```
>  v<- c(1, 3, 0, -1, 8)    #สร้าง ข้อมูล เวกเตอร์
>  v[v>0] #จะได้ 1 3 8
>  v>0  #จะได้ TRUE  TRUE  FALSE  FALSE  TRUE  TRUE
```

<u>การวาดกราฟแสดงลักษณะข้อมูล</u>

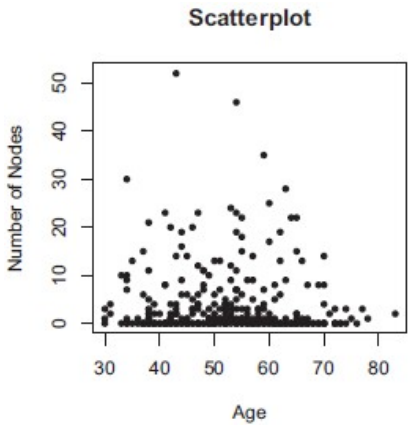
จะใช้ข้อมูล Haberman's Survival data (จาก UCI) และมีการอ่านเข้ามาแล้วเก็บไว้ที่ dataset

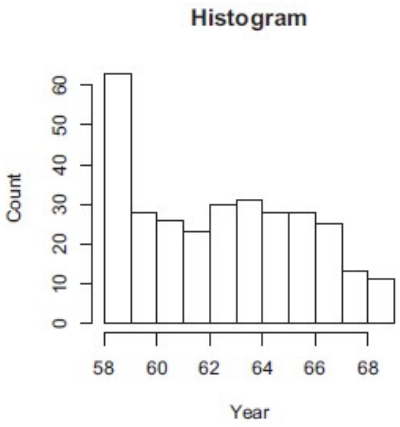
กราฟที่นิยมใช้คือ Scatter plot, Histogram, Boxplot

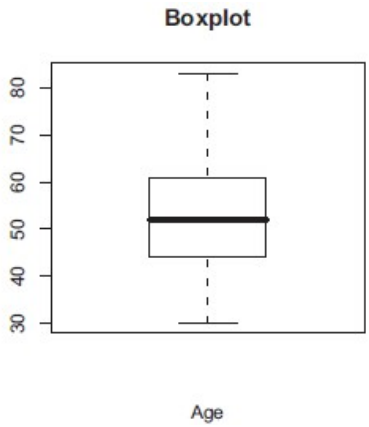
1.ต้องการแสดงรูปย่อย 3 รูป เรียงแถวเดียว

```
>  par(mfrow = c(1,3))
>plot(dataset[,1], dataset[,3], main="Scatterplot", xlab="Age", ylab="Number of Nodes", pch=20)
>hist(dataset[,2], main="Histogram", xlab="Year", ylab="Count")
>boxplot(dataset[,1], main="Boxplot", xlab="Age")
```

จะได้รูปดังนี้







การกำหนด formula ในการสร้างโมเดล

Formula คือ สมการสูตรที่ใช้เป็นพารามิเตอร์ในฟังก์ชันการสร้างโมเดล เช่น ต้อการระบุว่า y ขึ้นอยู่กับ x1, x2, x3 ในลักษณะเชิงเส้นทำได้ดังนี้

y ~x1+x2+x3

เมื่อ y, x1, x2, x3 คือ ชื่อคอลัมน์ในเมตริกซ์ที่จะใช้งาน

linear regression มีรูปแบบคือ

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon$$

เมื่อ e คือ การกระจายแบบปกติที่มีค่าเฉลี่ย=0 และ ความแปรปรวน = S^2เราต้องการหาสัมประสิทธิ์ (คือ B ทุกตัว) ต่อไปนี้คือการสร้าง linear model

```
>lm_model<-lm(y ~ x1 + x2, data=as.data.frame(cbind(y,x1,x2)))
>summary(lm_model)
```

Machine Learning

คือการทำให้คอมพิวเตอร์สามารถแยกแยะสิ่งที่สนใจได้ถูกต้อง ซึ่งมีขั้นตอน (เทคนิค) หลายแบบ คือ

รูปแบบข้อมูลคือ (อาจจะเรียก column ว่า feature)

X1	X2	……	…….	Y (class)

Prediction (การทำนาย)

ก่อนการทำนายผล จะต้องมีการสร้างโมเดลจากข้อมูลฝึก (train data) เมื่อได้โมเดลแล้ว (เช่น linear regression) ก็จะเอาโมเดลนี้ไปทดสอบกับข้อมูลทดสอบ (test data) ฟังก์ชันทดสอบ คือ predict(param1, param2)

พารามิเตอร์ ใน predict() คือ

1.param1 คือ โมเดลที่สร้างไว้แล้วจากข้อมูลฝึก

2. param2 คือ ข้อมูลทดสอบ

เช่น

```
>predicted_values<-predict(lm_model, newdata=as.data.frame(cbind(x1_test, x2_test)))
```

Apriori Algorithm

```
>dataset<-read.csv("C:\\Datasets\\mushroom.csv", header = TRUE)
>mushroom_rules<-apriori(as.matrix(dataset), parameter = list(supp = 0.8, conf = 0.9))
>summary(mushroom_rules) > inspect(mushroom_rules)
```

Logistic Regression

>glm_mod<-glm(y ~ x1+x2, family=binomial(link="logit"), data=as.data.frame(cbind(y,x1,x2)))

K-Means Clustering

>kmeans_model<-kmeans(x=X, centers=m)

k-Nearest Neighbor Classification

>knn_model<-knn(train=X_train, test=X_test, cl=as.factor(labels), k=K)

จะได้ knn model คือ factor vector ของ class attributes ใน the test set.

Naive Bayes

(ต้อง Install : the e1071 package ก่อนใช้งาน) มี formula เหมือน linear regression

>nB_model<-naiveBayes(y ~ x1 + x2, data=as.data.frame(cbind(y,x1,x2)))

Decision Trees (CART)

(ต้อง Install : rpartpackage)

>cart_model<-rpart(y ~ x1 + x2, data=as.data.frame(cbind(y,x1,x2)), method="class")

สามารถใช้ plot.rpartและ text.rpart เพื่อแสดง decision tree.

AdaBoost

(ต้อง Install : rpart package) โดยที่ labels(class)จะเป็น vector of 0-1

>boost_model<-ada(x=X, y=labels)

Support Vector Machines (SVM)

(ต้อง Install : e1071 package).

เมื่อ X เป็นเมตริกซ์ของหลาย colและ class เป็น 0-1 , C คือregularization parameter

>svm_model<-svm(x=X, y=as.factor(labels), kernel ="radial", cost=C) > summary(svm_model)