SVM example with Iris Data in R

Execute by Neha Raut

Problem Statement:

This program shows the classification of Iris data using Support Vector Machines classifier.

Load Data

```
#e1071 will be used for Support Vector Classification.
library("e1071")
library(GGally)
library(ggplot2)
```

```
data(iris)
```

Exploring the dataset

```
#first 4 are numeric and 5th is factor
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 ...
```

```
head(iris,5)
```

```
##   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
```

We will use sepal length, sepal width, petal length and petal width to predict the species of Flower.

Create SVM Model

```
svm_model <- svm(Species ~ ., data=iris,
                 kernel="radial") #linear/polynomial/sigmoid
```

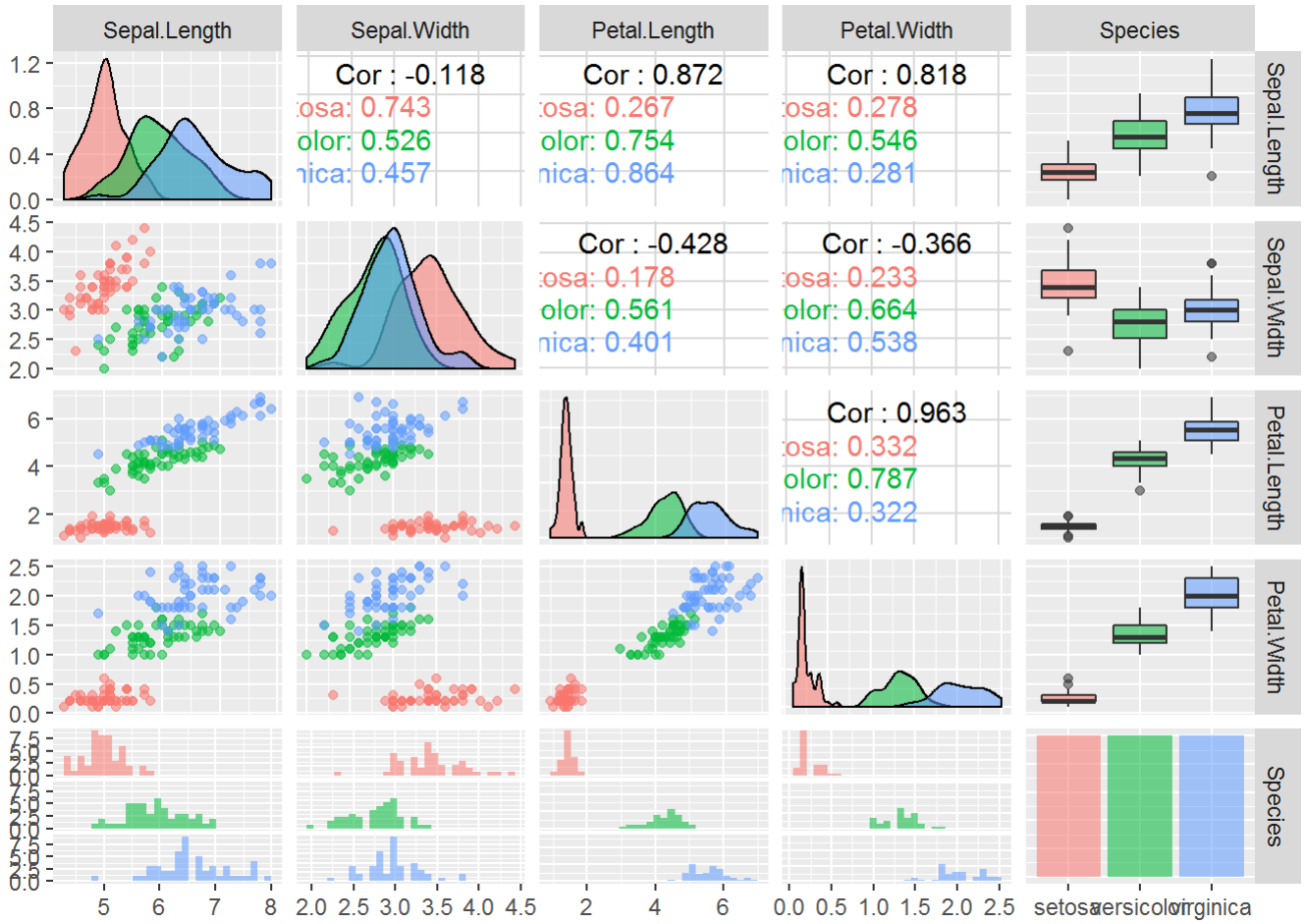
Since we are dealing with species which is char type, you will get svm type as a classsification defaulty, If data is quantitative that is continuous, you will get as regression

Exploratory Visualization

Lets have a closer look at the parameters and judge before hand if a good model can be created or not.

```
ggpairs(iris, ggplot2::aes(colour = Species, alpha = 0.4))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



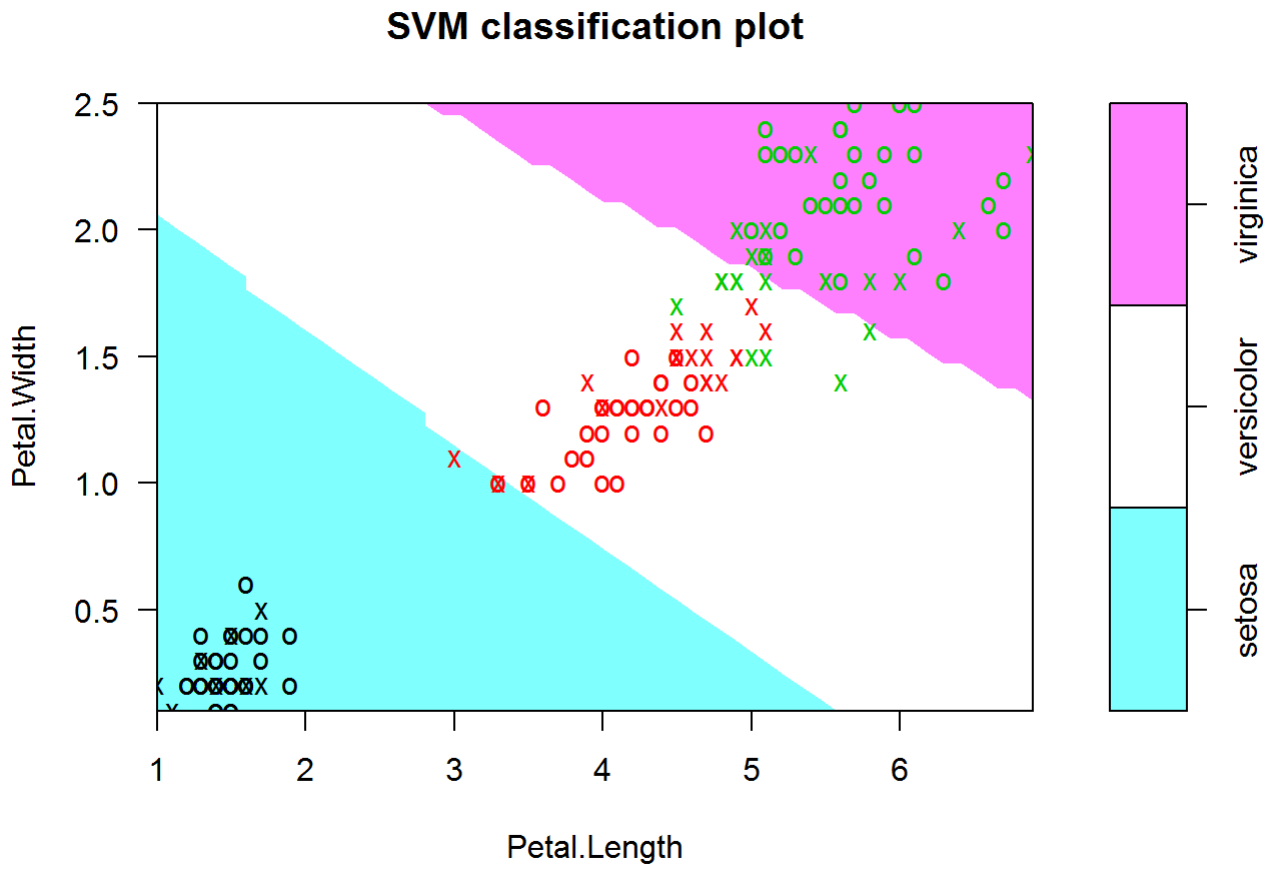
I personally like this graph because you can deduce so much information from one single chart. Lets have a look!

We can clearly see from the Histograms of Petal.length and Petal.width that we can clearly seperate out Setosa species with very high confidence.

However, Versicolor and Virginica Species are overlapped. If we look at the scatterplot of Sepal.Length vs Petal.Length and Petal.Width vs Petal.Length, we can distintly see a seperator that can be draw between the groups of Species.

Looks like we can just use Petal.Width and Petal.Length as parameters and come with a good model. SVM seems to be a very good model for this type of data.

```
plot(svm_model, data=iris,
     Petal.Width~Petal.Length,
     slice = list(Sepal.Width=3, Sepal.Length=4)
     )
```



from graph you can see data, support vector(represented by cross sign) and decision boundry, belong to 3 types of species

White color represented predicted class for second species(versicolor)

Pink color represented predicted class for third species(virginica)

Also we have 52 Support vector, 8 of them belongs to first species(You can see 8 cross in first class), 22 of them belongs to second species, 21 of them belongs to third species.

Predict each Species

Confusion matrix and missclasscation Error

```
pred = predict(svm_model,iris)
tab = table(Predicted=pred, Actual = iris$Species)
tab
```

```
##            Actual
## Predicted    setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          2        48
```

Out of the total data points, 50 data points belongs to first species and model also predicted 50 belongs to first species, same for second and third category.

Here, 2 data points belong to third category(virginica) but model predicted them to be from second category(versicolor) Similarly 2 data points belong to second category(versicolor) but model predicted them to be from third category(virginica).So here is missclassification

Get missclassification rate

```
1-sum(diag(tab)/sum(tab))
```

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

I have find missclassification rate for remaining three kernal,

For linear,I get missclassification rate: 0.03333333 i.e% 3.3% (higher than radial)

For polynomial, missclassification rate: 0.04666667 i.e 4.7%

for sigmoid,missclassification rate: 0.1133333 i.e 11.3% <- worst case

Hence Radial One is best

Parameter Tunning

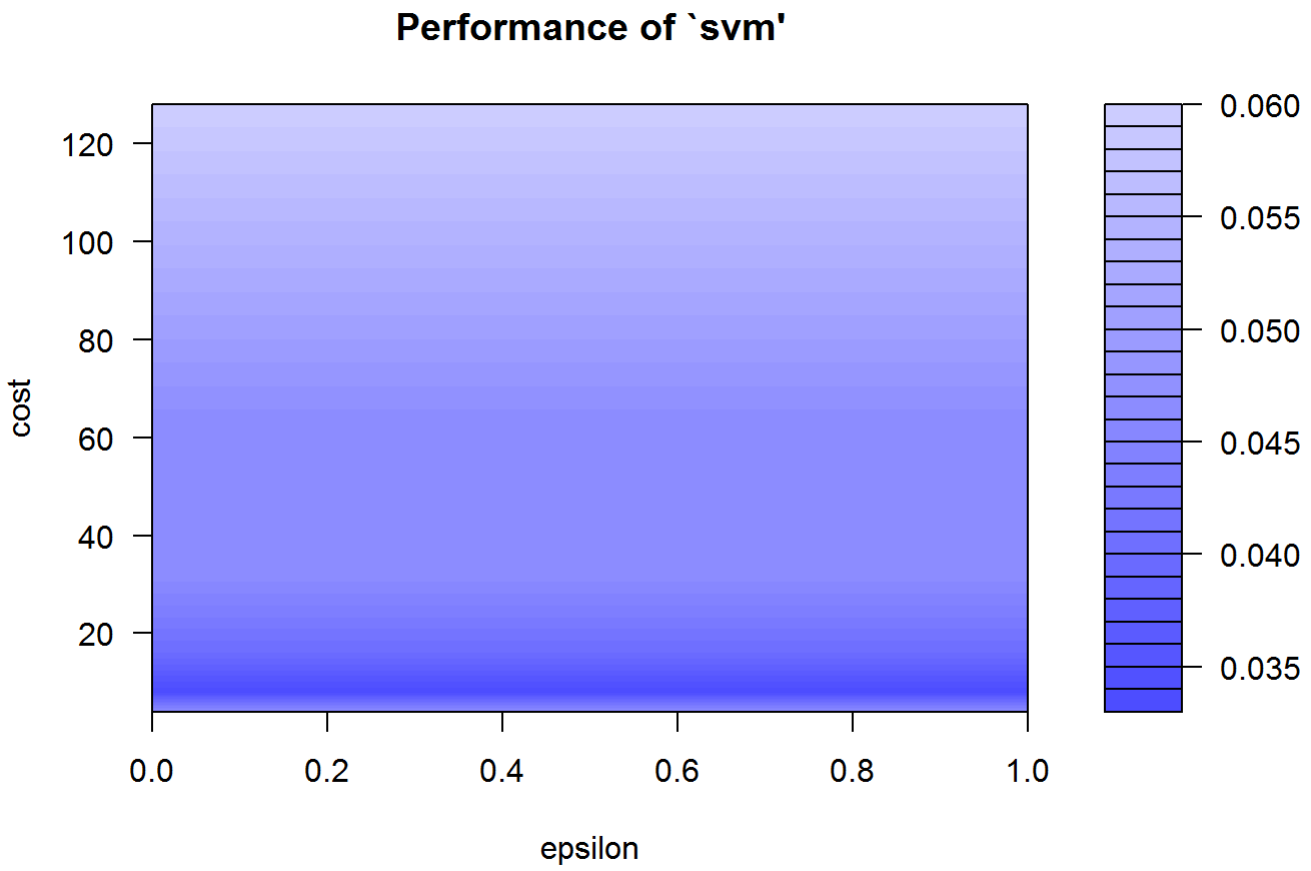
It help you to select best model

eps= start with 0 and end with 1 so 11 different values(0.1,0.2,…..1.0)

cost capture cost of constraint voilation and default values of cost is 1 If cost is too high this means high penalty for non seperable points [Model may store too many support vectors], which will lead to overfitting Similarly too small cost value will lead to underfitting

Here i am trying 11*7=77 different combinations

```
set.seed(123)
tmodel=tune(svm,Species~., data=iris,
     ranges=list(epsilon= seq(0,1,0.1), cost = 2^(2:7)))
plot(tmodel)
```



Plot gives us performance evauation of SVM for 2 parameter we have used.Dark region means Lower missclassification range

```
summary(tmodel)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  epsilon cost
##        0    8
##
## - best performance: 0.03333333
##
## - Detailed performance results:
##    epsilon cost      error dispersion
## 1      0.0    4 0.04666667 0.06324555
## 2      0.1    4 0.04666667 0.06324555
## 3      0.2    4 0.04666667 0.06324555
## 4      0.3    4 0.04666667 0.06324555
## 5      0.4    4 0.04666667 0.06324555
## 6      0.5    4 0.04666667 0.06324555
## 7      0.6    4 0.04666667 0.06324555
## 8      0.7    4 0.04666667 0.06324555
## 9      0.8    4 0.04666667 0.06324555
## 10     0.9    4 0.04666667 0.06324555
## 11     1.0    4 0.04666667 0.06324555
## 12     0.0    8 0.03333333 0.06478835
## 13     0.1    8 0.03333333 0.06478835
## 14     0.2    8 0.03333333 0.06478835
## 15     0.3    8 0.03333333 0.06478835
## 16     0.4    8 0.03333333 0.06478835
## 17     0.5    8 0.03333333 0.06478835
## 18     0.6    8 0.03333333 0.06478835
## 19     0.7    8 0.03333333 0.06478835
## 20     0.8    8 0.03333333 0.06478835
## 21     0.9    8 0.03333333 0.06478835
## 22     1.0    8 0.03333333 0.06478835
## 23     0.0   16 0.04000000 0.06440612
## 24     0.1   16 0.04000000 0.06440612
## 25     0.2   16 0.04000000 0.06440612
## 26     0.3   16 0.04000000 0.06440612
## 27     0.4   16 0.04000000 0.06440612
## 28     0.5   16 0.04000000 0.06440612
## 29     0.6   16 0.04000000 0.06440612
## 30     0.7   16 0.04000000 0.06440612
## 31     0.8   16 0.04000000 0.06440612
## 32     0.9   16 0.04000000 0.06440612
## 33     1.0   16 0.04000000 0.06440612
## 34     0.0   32 0.04666667 0.07062333
## 35     0.1   32 0.04666667 0.07062333
## 36     0.2   32 0.04666667 0.07062333
## 37     0.3   32 0.04666667 0.07062333
## 38     0.4   32 0.04666667 0.07062333
## 39     0.5   32 0.04666667 0.07062333
## 40     0.6   32 0.04666667 0.07062333
```

```
## 41     0.7   32 0.04666667 0.07062333
## 42     0.8   32 0.04666667 0.07062333
## 43     0.9   32 0.04666667 0.07062333
## 44     1.0   32 0.04666667 0.07062333
## 45     0.0   64 0.04666667 0.07062333
## 46     0.1   64 0.04666667 0.07062333
## 47     0.2   64 0.04666667 0.07062333
## 48     0.3   64 0.04666667 0.07062333
## 49     0.4   64 0.04666667 0.07062333
## 50     0.5   64 0.04666667 0.07062333
## 51     0.6   64 0.04666667 0.07062333
## 52     0.7   64 0.04666667 0.07062333
## 53     0.8   64 0.04666667 0.07062333
## 54     0.9   64 0.04666667 0.07062333
## 55     1.0   64 0.04666667 0.07062333
## 56     0.0  128 0.06000000 0.07981460
## 57     0.1  128 0.06000000 0.07981460
## 58     0.2  128 0.06000000 0.07981460
## 59     0.3  128 0.06000000 0.07981460
## 60     0.4  128 0.06000000 0.07981460
## 61     0.5  128 0.06000000 0.07981460
## 62     0.6  128 0.06000000 0.07981460
## 63     0.7  128 0.06000000 0.07981460
## 64     0.8  128 0.06000000 0.07981460
## 65     0.9  128 0.06000000 0.07981460
## 66     1.0  128 0.06000000 0.07981460
```

```
mymodel=tmodel$best.model
summary(mymodel)
```

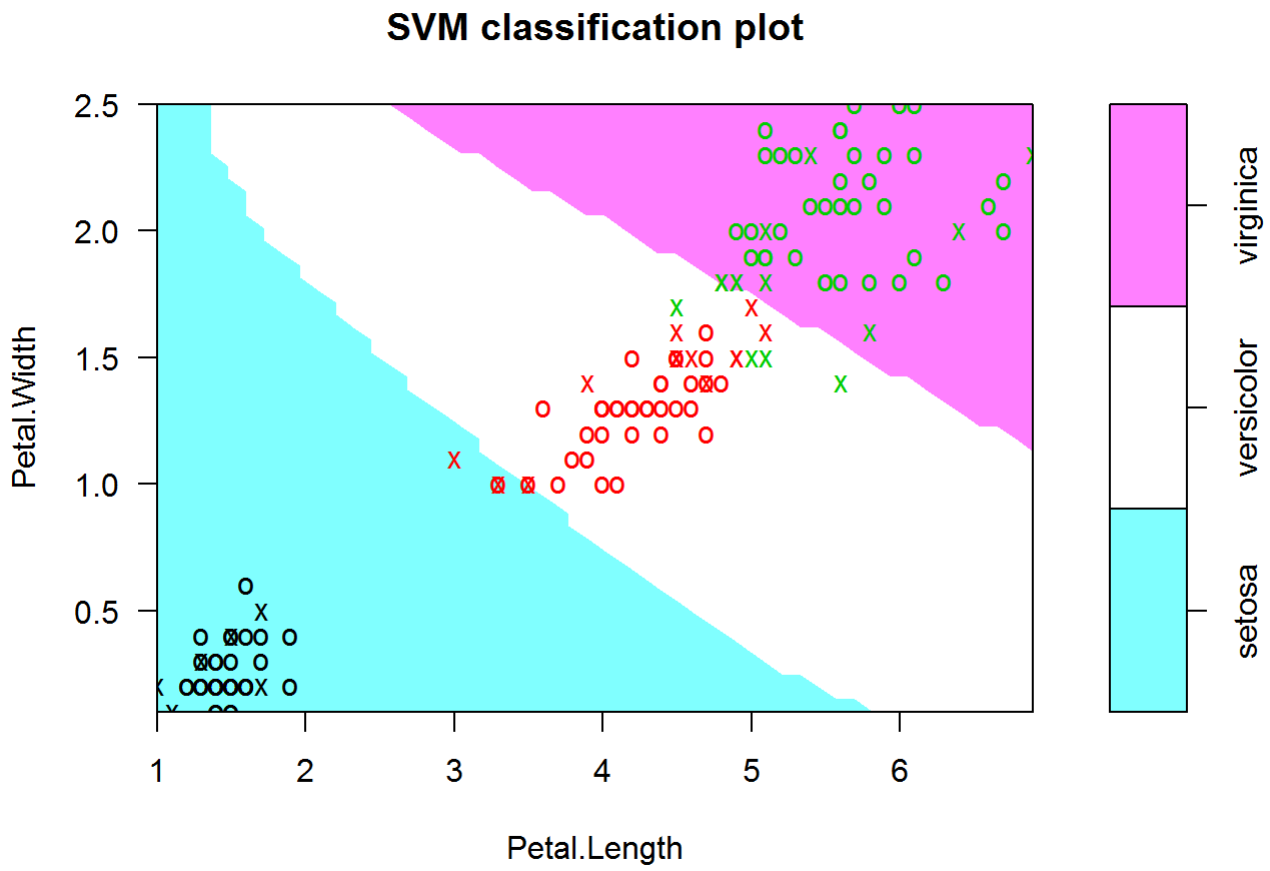
```
##
## Call:
## best.tune(method = svm, train.x = Species ~ ., data = iris, ranges = list(epsilon = seq(0,
##     1, 0.1), cost = 2^(2:7)))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  8
##       gamma:  0.25
##
## Number of Support Vectors:  35
##
##  ( 6 15 14 )
##
##
## Number of Classes:  3
##
## Levels:
##  setosa versicolor virginica
```

Best Model

```
mymodel=tmodel$best.model
summary(mymodel)
```

```
##
## Call:
## best.tune(method = svm, train.x = Species ~ ., data = iris, ranges = list(epsilon = seq(0,
##     1, 0.1), cost = 2^(2:7)))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  8
##       gamma:  0.25
##
## Number of Support Vectors:  35
##
##  ( 6 15 14 )
##
##
## Number of Classes:  3
##
## Levels:
##  setosa versicolor virginica
```

```
plot(mymodel, data=iris,
     Petal.Width~Petal.Length,
     slice = list(Sepal.Width=3, Sepal.Length=4)
)
```



Look at confusion matrix and missclassification rate using best parameter

```
pred1 = predict(mymodel,iris)
tab1 = table(Predicted=pred1, Actual = iris$Species)
tab1
```

```
##             Actual
## Predicted    setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         0
##   virginica       0          2        50
```

You can see out of 150 data points, only two are missclassified

```
1-sum(diag(tab1)/sum(tab1))
```

```
## [1] 0.01333333
```

Rate is 1.3% which is significantly less than what we have earliear