K. J. Somaiya College of Engineering,
Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)

Batch:- H2-1	**Roll No:-** 16010122151
Experiment No. 7	

Title: Classification using support vector machine

Aim: To implement classification using SVM using R libraries

Expected Outcome of Experiment:
CO3 : Understand the basic concept and techniques of Machine Learning regression and classification

Books/ Journals/ Websites referred:

1. Data Mining Concepts and Techniques Jiawei Han, Michelin Kamber, Jian Pie, 3rd edition

Procedure for Implementation in lab :

1. Select a dataset suitable for classification from UCI data repository or Kaggle.
    a. Data set used:
    b. Title:
    c. Source:
    d. Number of instances:
    e. Number of attributes:
    f. Attribute information **:**
2. Handle the missing values appropriately
3. Create the SVM model
4. Perform Visualization
5. Calculate prediction accuracy.

What are SVMs?

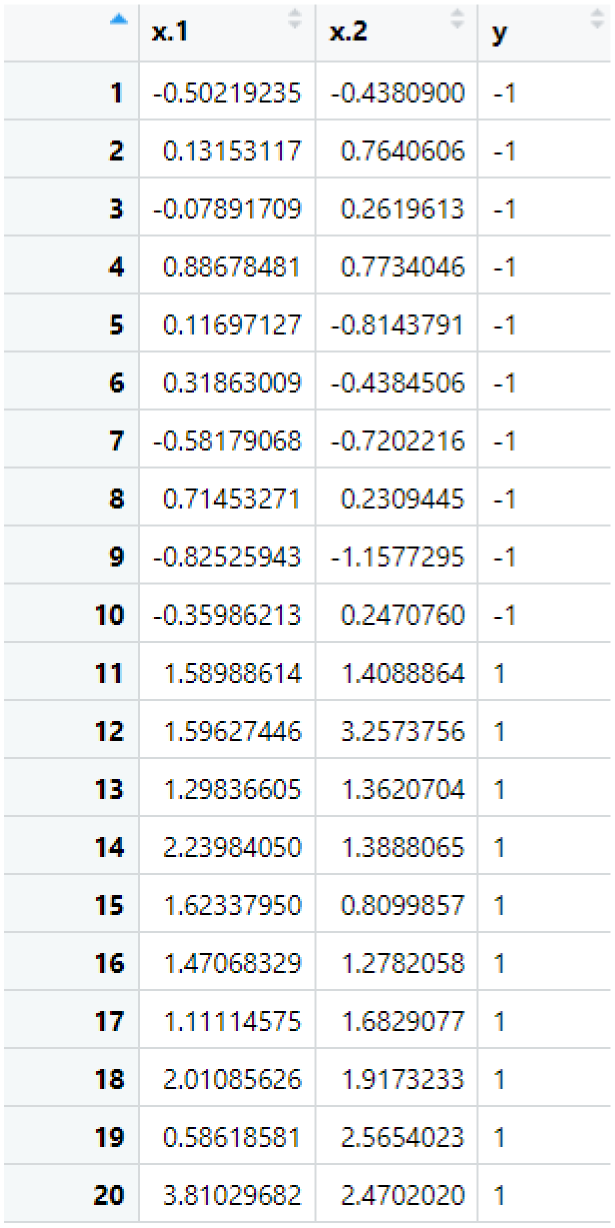
A support vector machine (SVM) is a supervised machine learning algorithm that classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space.

Maximal Margin Classifier

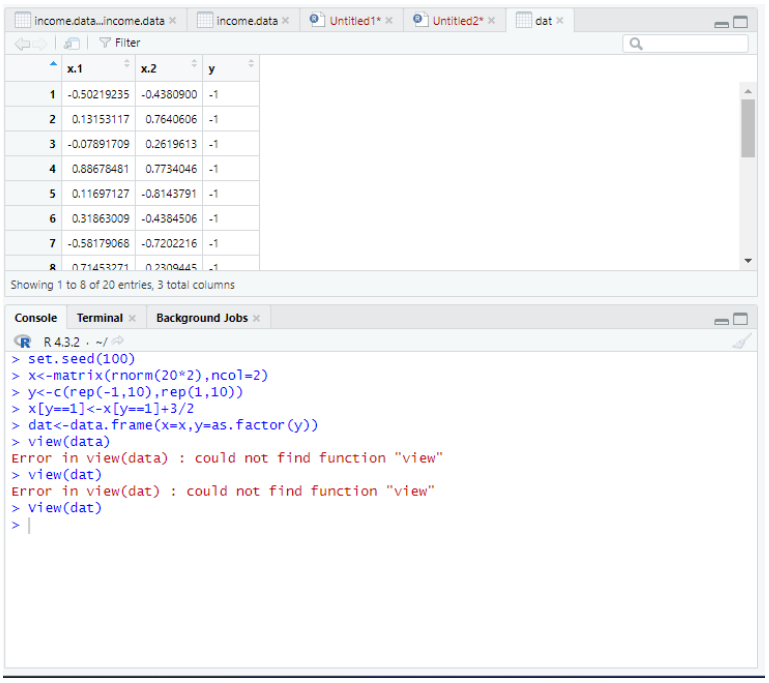
There are several types of SVM's. The simplest is the_maximal margin classifier (MMC). Though the MMC is elegant and simple, it cannot be applied to most data sets, since it requires the classes to be separable by a linear boundary.

To visualize an example of separated data, we will generate 40 random data points and assign them to two classes.

```
> set.seed(100)
> # Construct sample data set - completely separated
> x <- matrix(rnorm(20*2), ncol = 2)
> y <- c(rep(-1,10), rep(1,10))
> x[y==1,] <- x[y==1,] + 3/2
> dat <- data.frame(x=x, y=as.factor(y))
> View(dat)
```



	x.1	x.2	y
1	-0.50219235	-0.4380900	-1
2	0.13153117	0.7640606	-1
3	-0.07891709	0.2619613	-1
4	0.88678481	0.7734046	-1
5	0.11697127	-0.8143791	-1
6	0.31863009	-0.4384506	-1
7	-0.58179068	-0.7202216	-1
8	0.71453271	0.2309445	-1
9	-0.82525943	-1.1577295	-1
10	-0.35986213	0.2470760	-1
11	1.58988614	1.4088864	1
12	1.59627446	3.2573756	1
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1



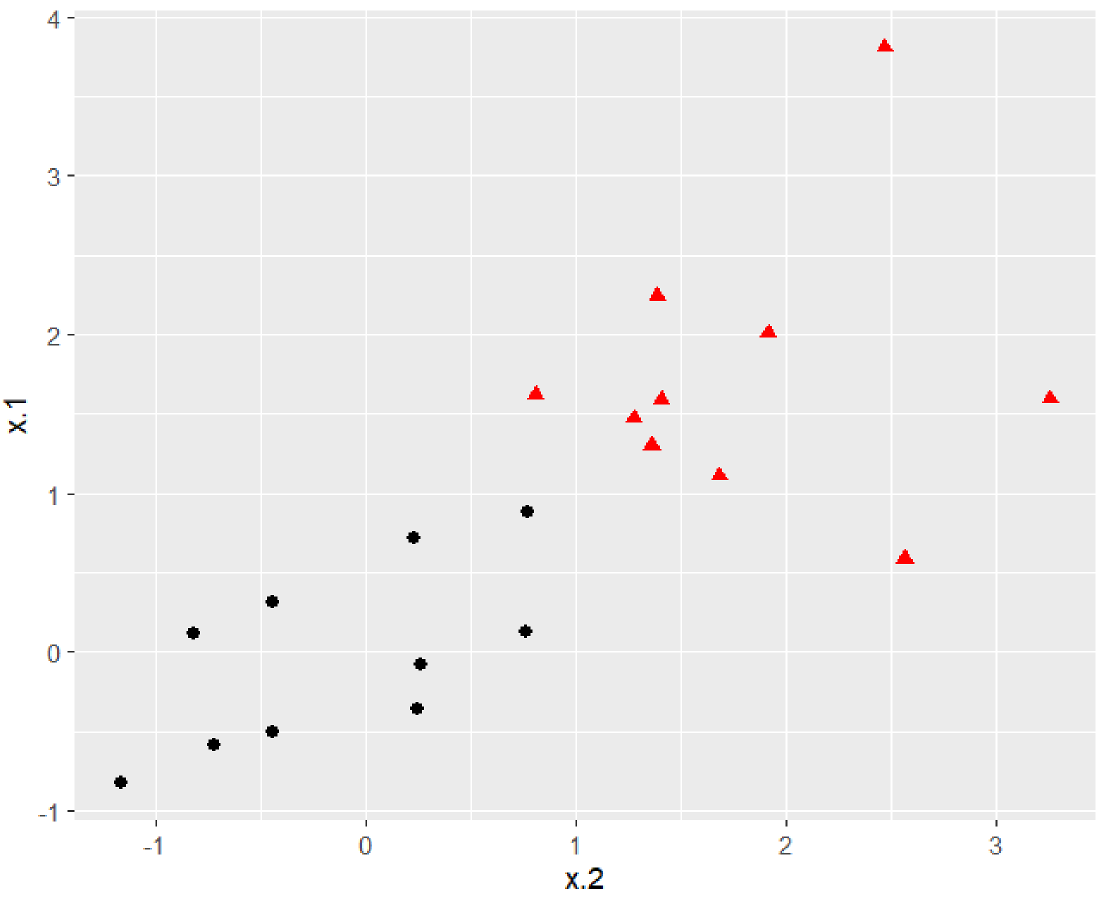
	x.1	x.2	y
1	-0.50219235	-0.4380900	-1
2	0.13153117	0.7640606	-1
3	-0.07891709	0.2619613	-1
4	0.88678481	0.7734046	-1
5	0.11697127	-0.8143791	-1
6	0.31863009	-0.4384506	-1
7	-0.58179068	-0.7202216	-1
8	0.71453271	0.2309445	-1

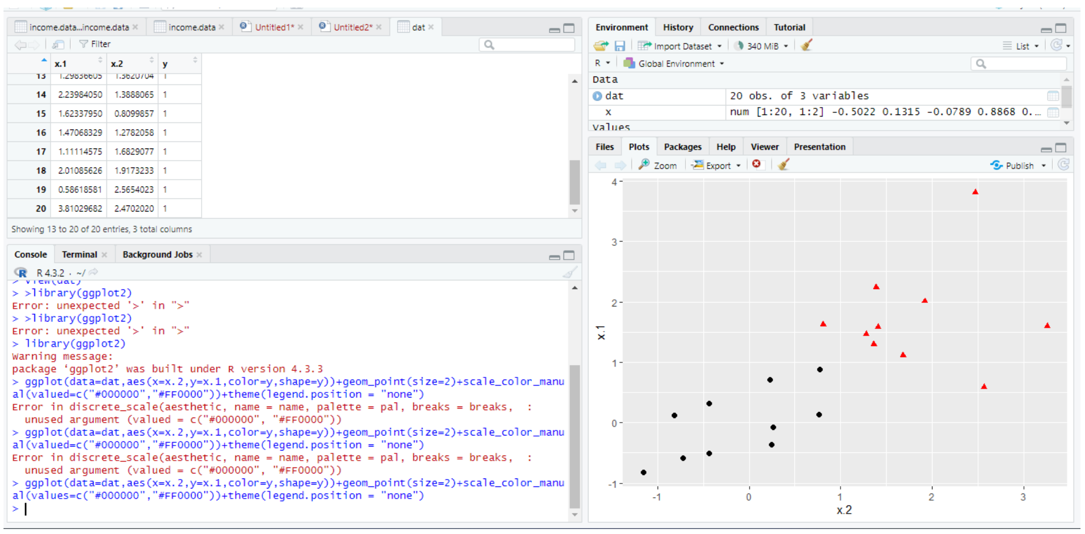
Showing 1 to 8 of 20 entries, 3 total columns

```
> set.seed(100)
> x<-matrix(rnorm(20*2),ncol=2)
> y<-c(rep(-1,10),rep(1,10))
> x[y==1]<-x[y==1]+3/2
> dat<-data.frame(x=x,y=as.factor(y))
> view(data)
Error in view(data) : could not find function "view"
> view(dat)
Error in view(dat) : could not find function "view"
> view(dat)
> |
```

Now we will plot this data

```
>library(ggplot2
> ggplot(data = dat, aes(x = x.2, y = x.1, color = y, shape = y)) +
+ geom_point(size = 2) +
+ scale_color_manual(values=c("#000000", "#FF0000")) +
+ theme(legend.position = "none")
```





	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

Showing 13 to 20 of 20 entries, 3 total columns

```
Data
dat     20 obs. of 3 variables
  x     num [1:20, 1:2] -0.5022 0.1315 -0.0789 0.8868 0....
Values
```

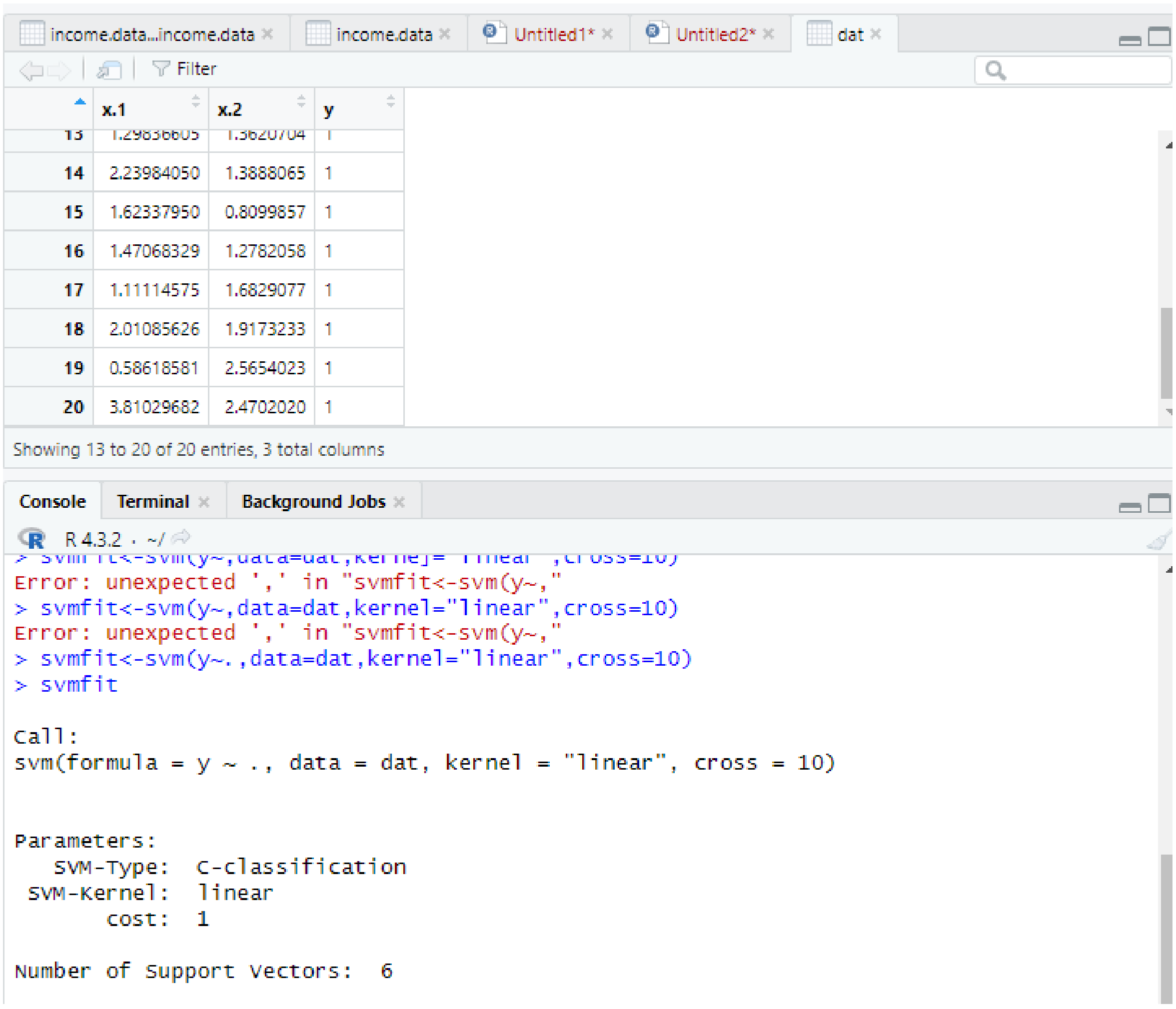
Console   Terminal   Background Jobs

R 4.3.2 · ~/

```
> View(dat)
> >library(ggplot2)
Error: unexpected '>' in ">"
> >library(ggplot2)
Error: unexpected '>' in ">"
> library(ggplot2)
Warning message:
package ‘ggplot2’ was built under R version 4.3.3
> ggplot(data=dat,aes(x=x.2,y=x.1,color=y,shape=y))+geom_point(size=2)+scale_color_manu
al(valued=c("#000000","#FF0000"))+theme(legend.position = "none")
Error in discrete_scale(aesthetic, name = name, palette = pal, breaks = breaks,  :
  unused argument (valued = c("#000000", "#FF0000"))
> ggplot(data=dat,aes(x=x.2,y=x.1,color=y,shape=y))+geom_point(size=2)+scale_color_manu
al(valued=c("#000000","#FF0000"))+theme(legend.position = "none")
Error in discrete_scale(aesthetic, name = name, palette = pal, breaks = breaks,  :
  unused argument (valued = c("#000000", "#FF0000"))
> ggplot(data=dat,aes(x=x.2,y=x.1,color=y,shape=y))+geom_point(size=2)+scale_color_manu
al(values=c("#000000","#FF0000"))+theme(legend.position = "none")
> |
```

Upon visual inspection, we can see that infinitely many lines exist that split the two classes. To find the line that maximizes the margin between the classes, we use the svm() method from the e1071 library. We will use 10-fold cross validation.

```
> library(e1071)
> svmfit <- svm(y~., data = dat, kernel = "linear", cross=10)
> svmfit

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  6
```



	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

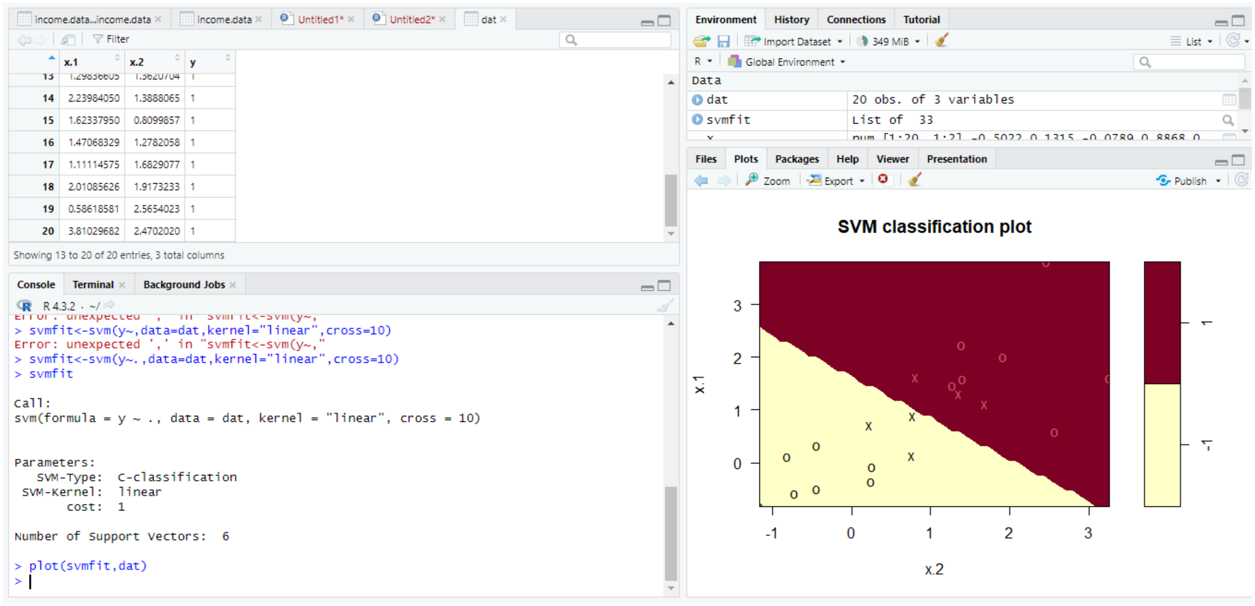
Showing 13 to 20 of 20 entries, 3 total columns

```
> svmfit<-svm(y~,data=dat,kernel="linear",cross=10)
Error: unexpected ',' in "svmfit<-svm(y~,"
> svmfit<-svm(y~,data=dat,kernel="linear",cross=10)
Error: unexpected ',' in "svmfit<-svm(y~,"
> svmfit<-svm(y~.,data=dat,kernel="linear",cross=10)
> svmfit

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  6
```

K. J. Somaiya College of Engineering,
Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)

Visualization:

```
> plot(svmfit,dat)
```

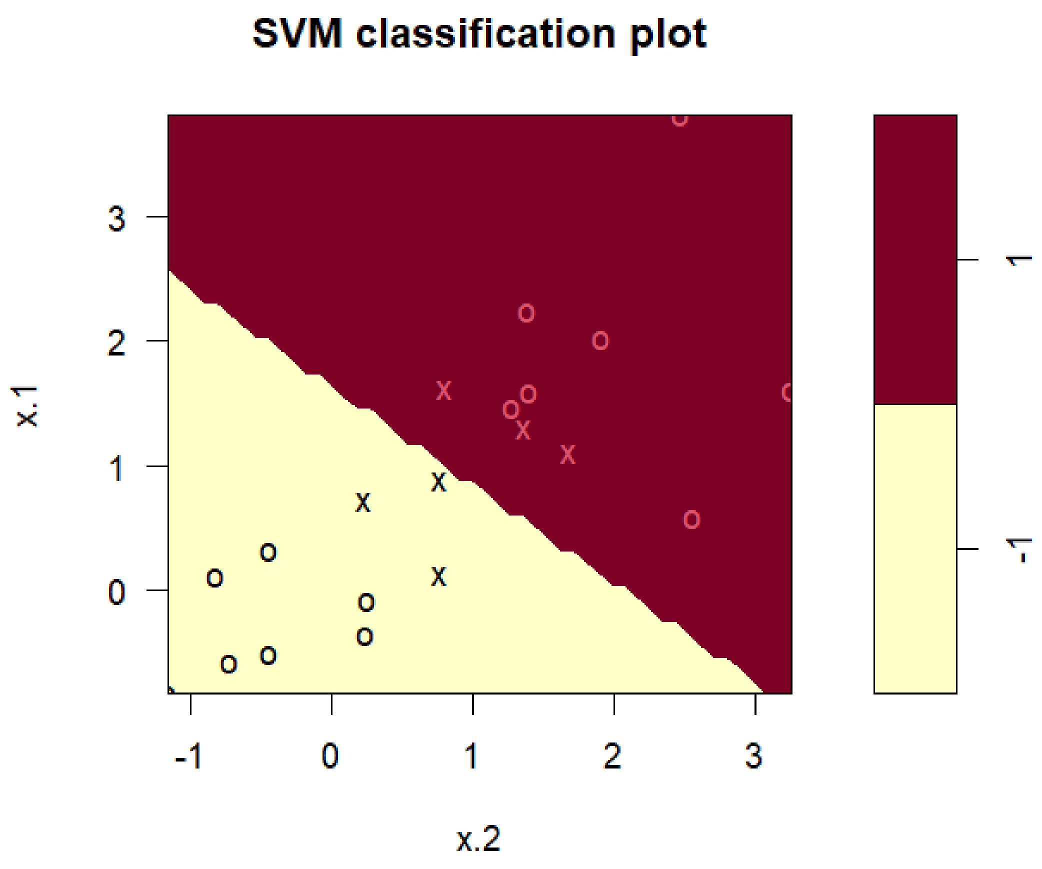


	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

Showing 13 to 20 of 20 entries, 3 total columns

```
Error: unexpected ',' in "svmfit<-svm(y~,"
> svmfit<-svm(y~,data=dat,kernel="linear",cross=10)
Error: unexpected ',' in "svmfit<-svm(y~,"
> svmfit<-svm(y~.,data=dat,kernel="linear",cross=10)
> svmfit

Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  6

> plot(svmfit,dat)
> |
```

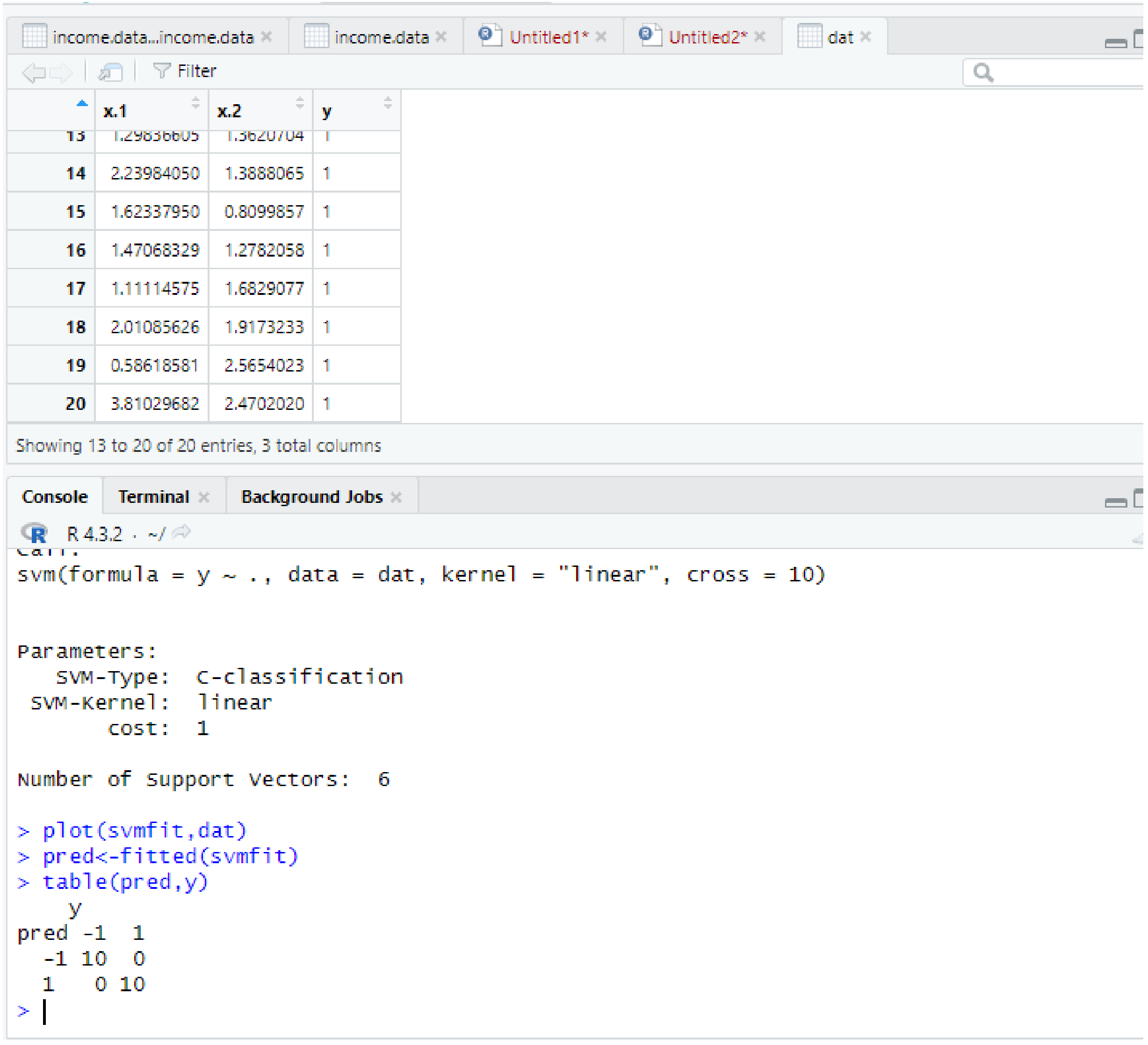
```
Data
dat        20 obs. of 3 variables
svmfit     List of  33
  x        num [1:20, 1:2] -0.5022 0.1315 -0.0789 0.8868 0
```



Finding the accuracy of the model:

```
> pred <- fitted(svmfit)
> table(pred,y)
      y
pred -1  1
  -1 10  0
  1   0 10
```

Accuracy = 100% (dataset is too small)



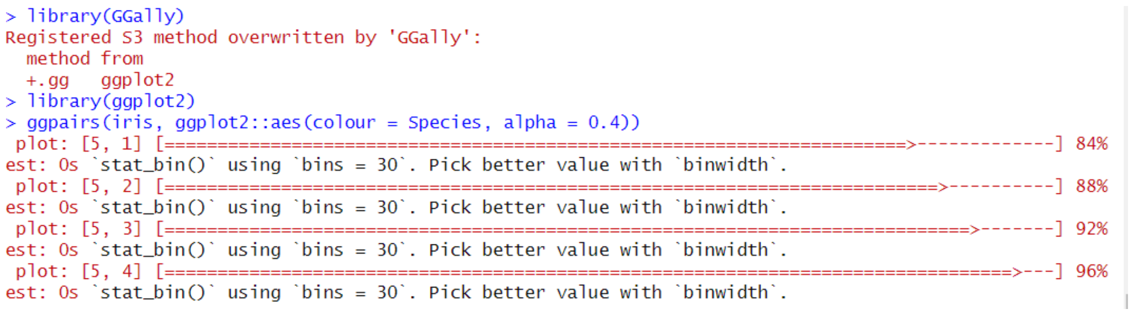
	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

Showing 13 to 20 of 20 entries, 3 total columns

```
Call:
svm(formula = y ~ ., data = dat, kernel = "linear", cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  6

> plot(svmfit,dat)
> pred<-fitted(svmfit)
> table(pred,y)
    y
pred -1  1
  -1 10  0
  1   0 10
> 
```

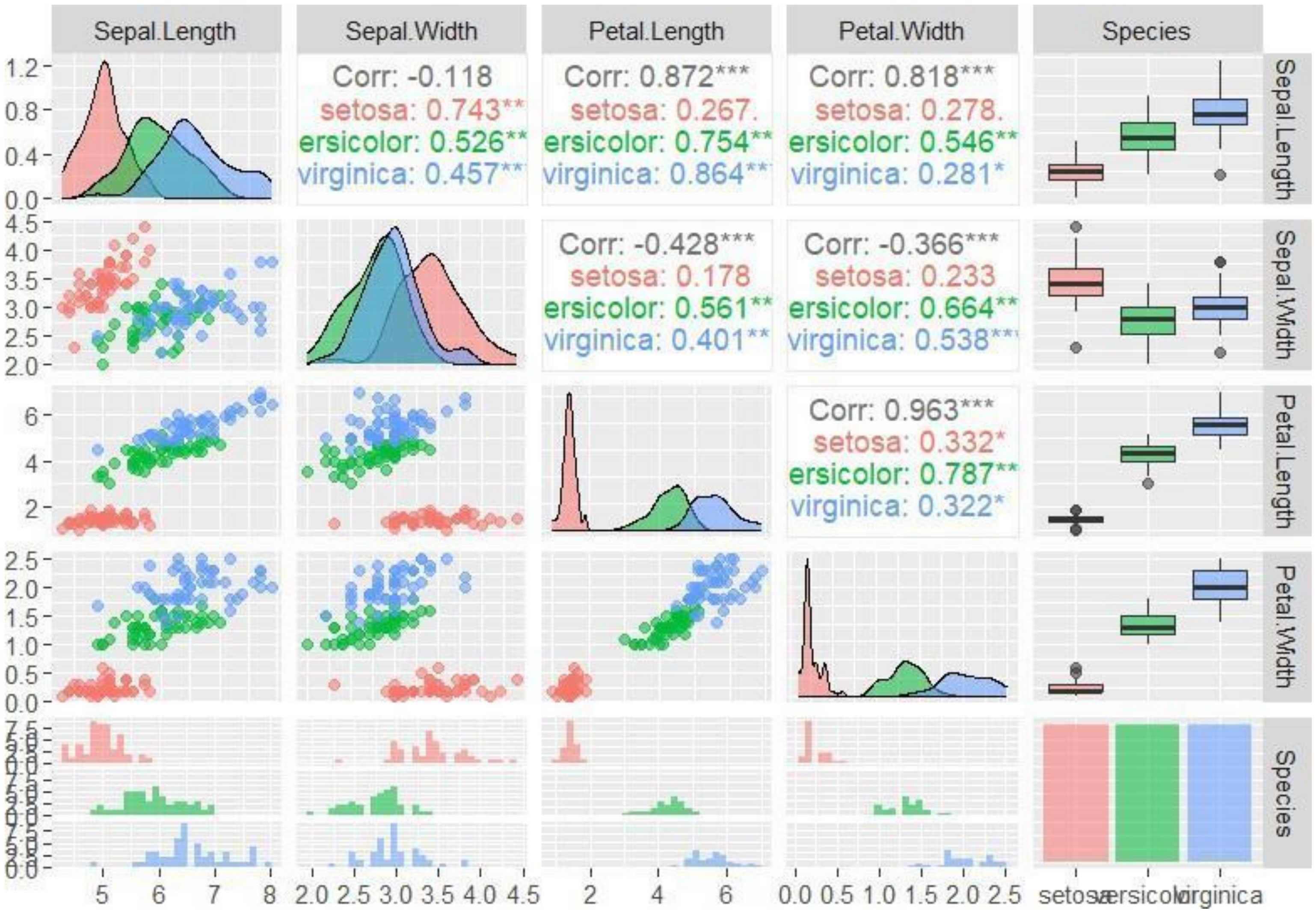
Applying SVM to Iris Dataset (4 predictors, 3 classes)

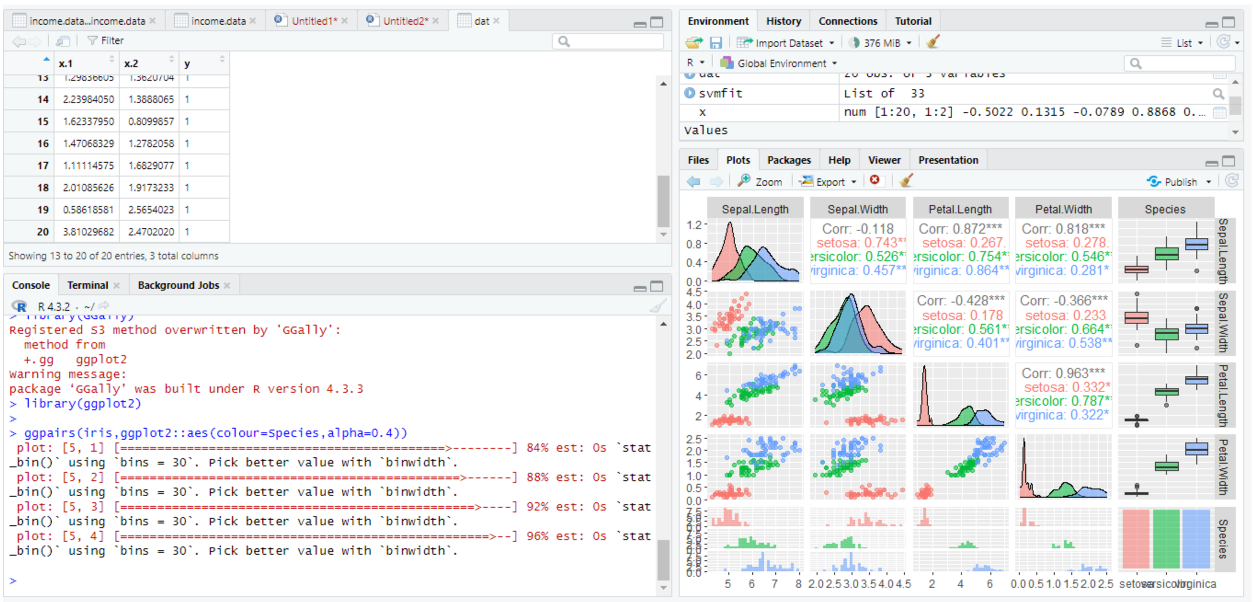
Since there are 4 predictors, it's not possible to have a simple 2D visualization. Hence we will select only 2 features, just for the sake of visualization. We will use all the four features for training the SVM model.



```
> library(GGally)
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
> library(ggplot2)
> ggpairs(iris, ggplot2::aes(colour = Species, alpha = 0.4))
 plot: [5, 1] [=====================================================================>-------------] 84%
est: 0s `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5, 2] [========================================================================>----------] 88%
est: 0s `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5, 3] [===========================================================================>-------] 92%
est: 0s `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5, 4] [===============================================================================>---] 96%
est: 0s `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

K. J. Somaiya College of Engineering,
Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)





	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

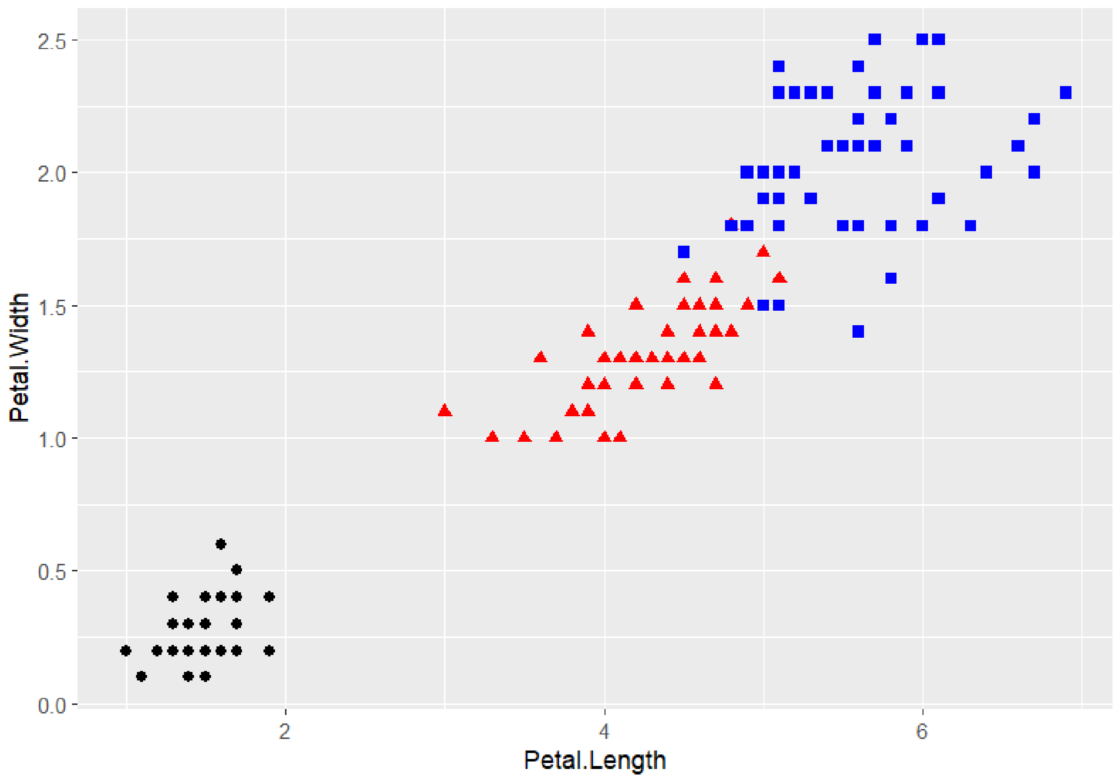
Showing 13 to 20 of 20 entries, 3 total columns

```
svmfit    List of  33
  x       num [1:20, 1:2] -0.5022 0.1315 -0.0789 0.8868 0...
Values
```

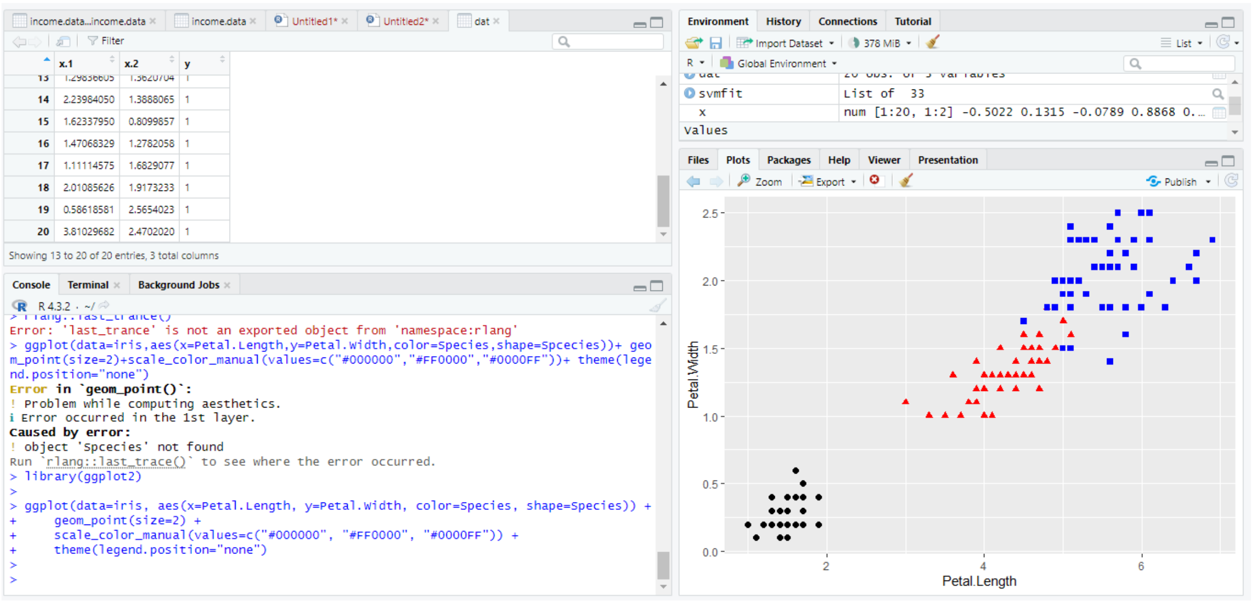
```
R 4.3.2 · ~/
> library(GGally)
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
Warning message:
package 'GGally' was built under R version 4.3.3
> library(ggplot2)
>
> ggpairs(iris,ggplot2::aes(colour=Species,alpha=0.4))
 plot: [5, 1] [==========================================>--------] 84% est: 0s `stat
_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5, 2] [============================================>------] 88% est: 0s `stat
_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5, 3] [==============================================>----] 92% est: 0s `stat
_bin()` using `bins = 30`. Pick better value with `binwidth`.
 plot: [5, 4] [================================================>--] 96% est: 0s `stat
_bin()` using `bins = 30`. Pick better value with `binwidth`.

>
```

Clearly from the Histograms of Petal.length and Petal.width that we can clearly separate out Setosa species with very high confidence. However, Versicolor and Virginica Species are overlapped. If we look at the scatterplot of Petal.Width vs Petal.Length, we can distinctly see a separator that can be drawn between the groups of Species. Looks like we can just use Petal.Width and Petal.Length as parameters for visualization.

```
> ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width, color = Species, shape = Species)) + geom_point(size = 2)
+ scale_color_manual(values=c("#000000", "#FF0000","#0000FF"))+ theme(legend.position = "none")
```



```
> lin.svm.iris <- svm(Species~., data = iris, kernel = "linear", cross=10)
> lin.svm.iris

Call:
svm(formula = Species ~ ., data = iris, kernel = "linear", cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  29
```



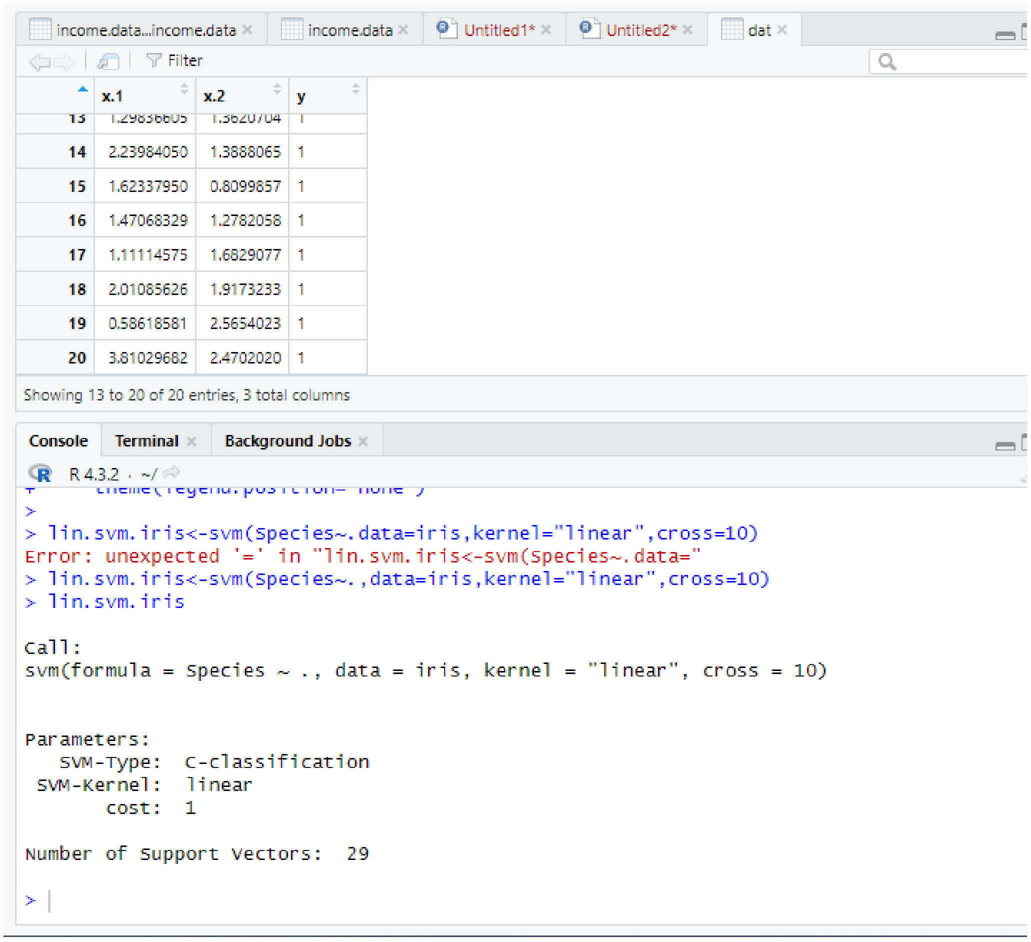
	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

Showing 13 to 20 of 20 entries, 3 total columns

```
dat       20 obs. of 3 variables
svmfit    List of  33
x         num [1:20, 1:2] -0.5022 0.1315 -0.0789 0.8868 0....
Values
```

Console Terminal Background Jobs
R 4.3.2 · ~/

```
> rlang::last_trance()
Error: 'last_trance' is not an exported object from 'namespace:rlang'
> ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width,color=Species,shape=Spcecies))+ geo
m_point(size=2)+scale_color_manual(values=c("#000000","#FF0000","#0000FF"))+ theme(lege
nd.position="none")
Error in `geom_point()`:
! Problem while computing aesthetics.
i Error occurred in the 1st layer.
Caused by error:
! object 'Spcecies' not found
Run `rlang::last_trace()` to see where the error occurred.
> library(ggplot2)
>
> ggplot(data=iris, aes(x=Petal.Length, y=Petal.Width, color=Species, shape=Species)) +
+     geom_point(size=2) +
+     scale_color_manual(values=c("#000000", "#FF0000", "#0000FF")) +
+     theme(legend.position="none")
>
>
```



	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

Showing 13 to 20 of 20 entries, 3 total columns

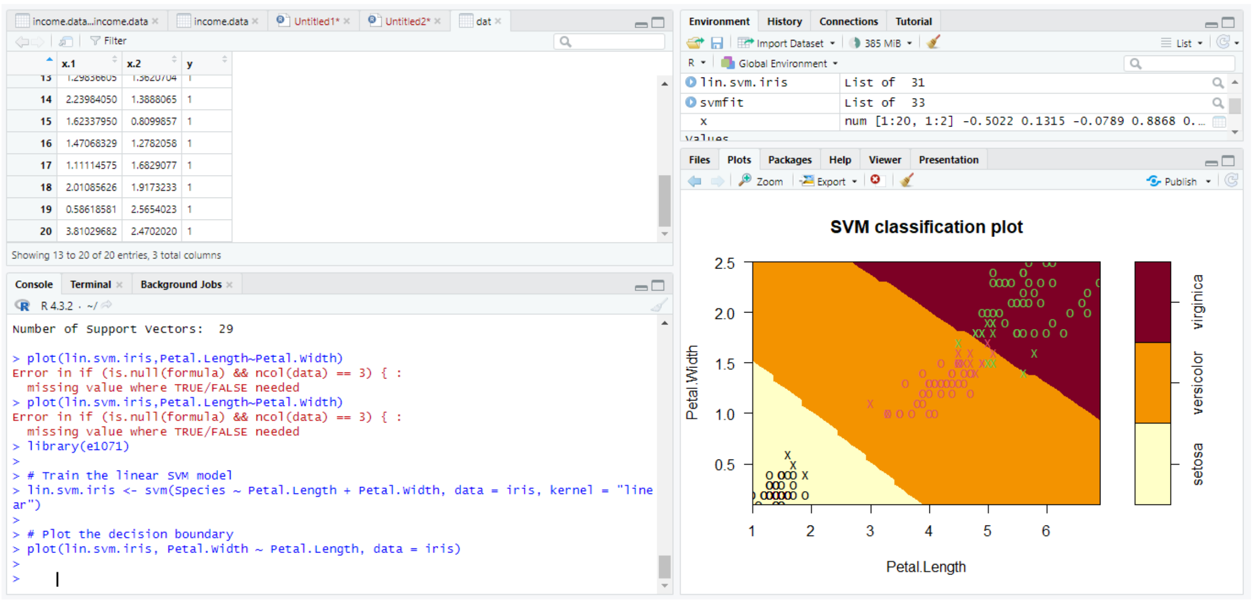
Console Terminal Background Jobs
R 4.3.2 · ~/

```
+     theme(legend.position="none")
>
> lin.svm.iris<-svm(Species~.data=iris,kernel="linear",cross=10)
Error: unexpected '=' in "lin.svm.iris<-svm(Species~.data="
> lin.svm.iris<-svm(Species~.,data=iris,kernel="linear",cross=10)
> lin.svm.iris

Call:
svm(formula = Species ~ ., data = iris, kernel = "linear", cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  29

> |
```

K. J. Somaiya College of Engineering,
Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)

Visualization:

```
> plot(lin.svm.iris,iris,Petal.Length~Petal.Width)
```

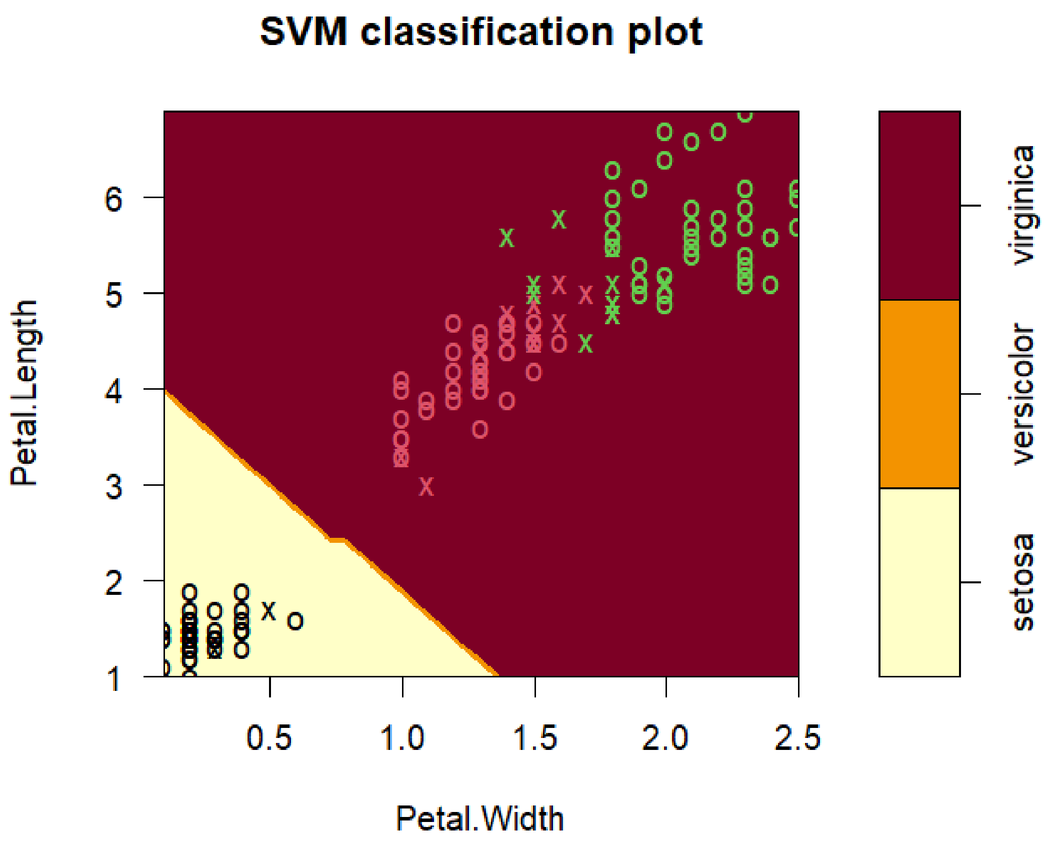


	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

Showing 13 to 20 of 20 entries, 3 total columns

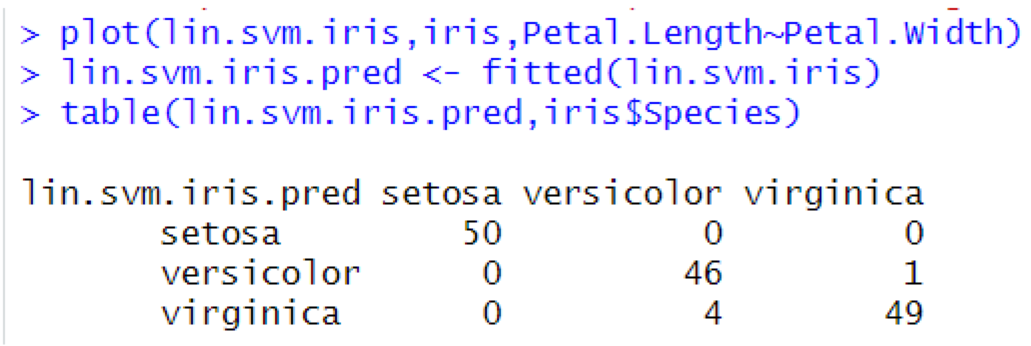
lin.svm.iris	List of 31
svmfit	List of 33
x	num [1:20, 1:2] -0.5022 0.1315 -0.0789 0.8868 0.…

```
Number of Support Vectors:  29

> plot(lin.svm.iris,Petal.Length~Petal.Width)
Error in if (is.null(formula) && ncol(data) == 3) { : 
  missing value where TRUE/FALSE needed
> plot(lin.svm.iris,Petal.Length~Petal.Width)
Error in if (is.null(formula) && ncol(data) == 3) { : 
  missing value where TRUE/FALSE needed
> library(e1071)
> 
> # Train the linear SVM model
> lin.svm.iris <- svm(Species ~ Petal.Length + Petal.Width, data = iris, kernel = "line
ar")
> 
> # Plot the decision boundary
> plot(lin.svm.iris, Petal.Width ~ Petal.Length, data = iris)
> 
>     |
```



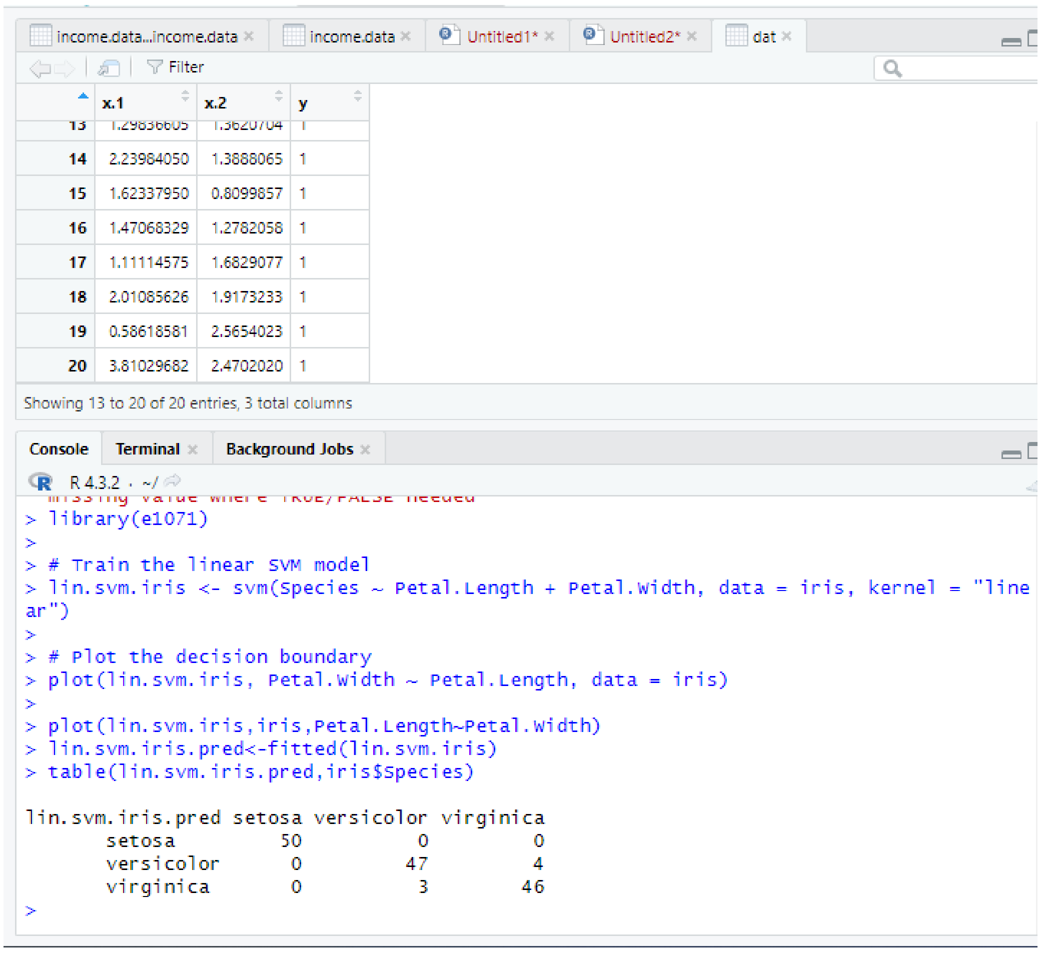
Note: The hyperplane here will be 3-D (n predictors = (n-1) D hyperplane)

Finding the accuracy of the model:



```
> plot(lin.svm.iris,iris,Petal.Length~Petal.Width)
> lin.svm.iris.pred <- fitted(lin.svm.iris)
> table(lin.svm.iris.pred,iris$Species)

lin.svm.iris.pred setosa versicolor virginica
       setosa         50          0         0
       versicolor      0         46         1
       virginica       0          4        49
```

Accuracy = (50+46+49)/150 = 0.967



	x.1	x.2	y
13	1.29836605	1.3620704	1
14	2.23984050	1.3888065	1
15	1.62337950	0.8099857	1
16	1.47068329	1.2782058	1
17	1.11114575	1.6829077	1
18	2.01085626	1.9173233	1
19	0.58618581	2.5654023	1
20	3.81029682	2.4702020	1

Showing 13 to 20 of 20 entries, 3 total columns

```
missing value where TRUE/FALSE needed
> library(e1071)
>
> # Train the linear SVM model
> lin.svm.iris <- svm(Species ~ Petal.Length + Petal.Width, data = iris, kernel = "line
ar")
>
> # Plot the decision boundary
> plot(lin.svm.iris, Petal.Width ~ Petal.Length, data = iris)
>
> plot(lin.svm.iris,iris,Petal.Length~Petal.Width)
> lin.svm.iris.pred<-fitted(lin.svm.iris)
> table(lin.svm.iris.pred,iris$Species)

lin.svm.iris.pred setosa versicolor virginica
       setosa         50          0         0
       versicolor      0         47         4
       virginica       0          3        46
>
```

Conclusion:
We learnt basic classification using support vector machine (SVM) using R.

<u>Post lab Questions:</u>

Q1. Evaluating the Impact of Different Kernels:
In this experiment, you used a linear kernel for the SVM. Explore non-linear kernels, such as a polynomial kernel or a radial basis function (RBF) kernel. Train the SVM model again using a non-linear kernel of your choice. Evaluate the performance metrics (accuracy, precision, recall, etc.) for both the linear and non-linear models. Explain the observed differences in performance based on the nature of your data and the chosen kernels.

Q2. Explain underfitting and overfitting in machine learning. Did you observe any of these when you performed the experiment? How do you handle these issues?
Underfitting occurs when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both training and test datasets. Overfitting happens when a model is excessively complex, capturing noise instead of true patterns, leading to excellent performance on the training set but poor generalization to unseen data. Regularization techniques like L1/L2 regularization, cross-validation, and using simpler models can help mitigate these issues.