- In such scenarios, calculate the margin which is the distance between the nearest data point and hyper-plane. The plane has the maximum distance will be considered as the right hyperplane to classify the classes better.
  Here C is having the maximum margin and hence it will be considered as a right hyperplane.

Above are some scenarios to identify the right hyper-plane.
Note: For details on Classifying using SVM in Python, refer to Classifying data using Support Vector Machines(SVMs) in Python

Implementation of SVM in R

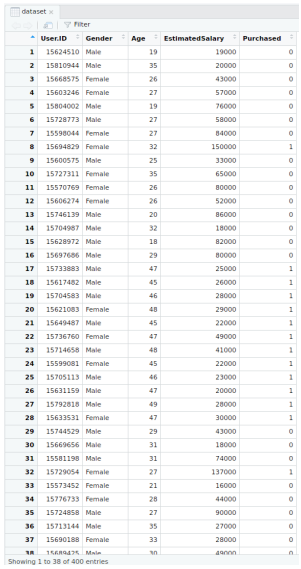
Here, an example is taken by importing a dataset of Social network aids from file Social.csv
The implementation is explained in the following steps:

- **Importing the dataset**

R

```
# Importing the dataset
dataset = read.csv('Social_Network_Ads.csv')
dataset = dataset[3:5]
```

- **Output:**



	User.ID	Gender	Age	EstimatedSalary	Purchased
1	15624510	Male	19	19000	0
2	15810944	Male	35	20000	0
3	15668575	Female	26	43000	0
4	15603246	Female	27	57000	0
5	15804002	Male	19	76000	0
6	15728773	Male	27	58000	0
7	15598044	Female	27	84000	0
8	15694829	Female	32	150000	1
9	15600575	Male	25	33000	0
10	15727311	Female	35	65000	0
11	15570769	Female	26	80000	0
12	15606274	Female	26	52000	0
13	15746139	Male	20	86000	0
14	15704987	Male	32	18000	0
15	15628972	Male	18	82000	0
16	15697686	Male	29	80000	0
17	15733883	Male	47	25000	1
18	15617482	Male	45	26000	1
19	15704583	Male	46	28000	1
20	15621083	Female	48	29000	1
21	15649487	Male	45	22000	1
22	15736760	Female	47	49000	1
23	15714658	Male	48	41000	1
24	15599081	Female	45	22000	1
25	15705113	Male	46	23000	1
26	15631159	Male	47	20000	1
27	15792818	Male	49	28000	1
28	15633531	Female	47	30000	1
29	15744529	Male	29	43000	0
30	15669656	Male	31	18000	0
31	15581198	Male	31	74000	0
32	15729054	Female	27	137000	1
33	15573452	Female	21	16000	0
34	15776733	Female	28	44000	0
35	15724858	Male	27	90000	0
36	15713144	Male	35	27000	0
37	15690188	Female	33	28000	0

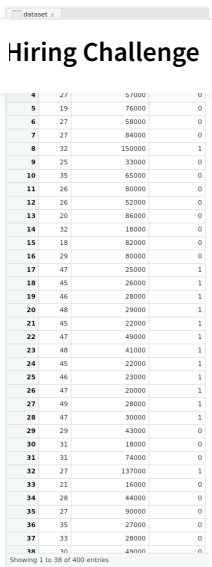
Showing 1 to 38 of 400 entries

-
- **Selecting columns 3-5**
  This is done for ease of computation and implementation (to keep the example simple).

R

```
# Taking columns 3-5
dataset = dataset[3:5]
```

- **Output:**



4	27	57000	0
5	19	76000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
9	25	33000	0
10	35	65000	0
11	26	80000	0
12	26	52000	0
13	20	86000	0
14	32	18000	0
15	18	82000	0
16	29	80000	0
17	47	25000	1
18	45	26000	1
19	46	28000	1
20	48	29000	1
21	45	22000	1
22	47	49000	1
23	48	41000	1
24	45	22000	1
25	46	23000	1
26	47	20000	1
27	49	28000	1
28	47	30000	1
29	29	43000	0
30	31	18000	0
31	31	74000	0
32	27	137000	1
33	21	16000	0
34	28	44000	0
35	27	90000	0
36	35	27000	0
37	33	28000	0
38	30	49000	0

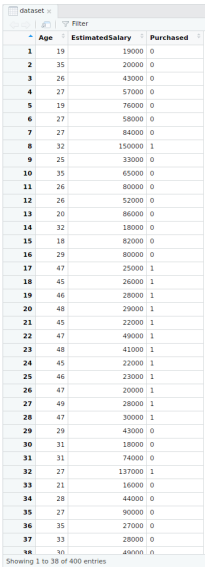
Showing 1 to 38 of 400 entries

-
- **Encoding the target feature**

R

```
# Encoding the target feature as factor
dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
```

- **Output:**



	Age	EstimatedSalary	Purchased
1	19	19000	0
2	35	20000	0
3	26	43000	0
4	27	57000	0
5	19	76000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
9	25	33000	0
10	35	65000	0
11	26	80000	0
12	26	52000	0
13	20	86000	0
14	32	18000	0
15	18	82000	0
16	29	80000	0
17	47	25000	1
18	45	26000	1
19	46	28000	1
20	48	29000	1
21	45	22000	1
22	47	49000	1
23	48	41000	1
24	45	22000	1
25	46	23000	1
26	47	20000	1
27	49	28000	1
28	47	30000	1
29	29	43000	0
30	31	18000	0
31	31	74000	0
32	27	137000	1
33	21	16000	0
34	28	44000	0
35	27	90000	0
36	35	27000	0
37	33	28000	0
38	30	49000	0

Showing 1 to 38 of 400 entries

-
- **Splitting the dataset**

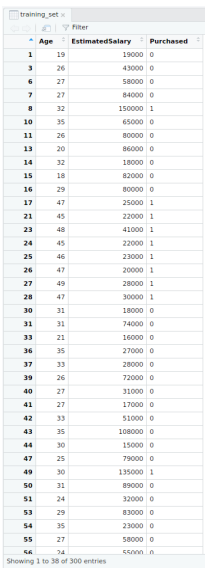
R

```
# Splitting the dataset into the Training set and Test set
install.packages('caTools')
library(caTools)

set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

- **Output:**
  - Splitter

```
> split
  [1]  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
 [12] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE
 [23]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
 [34] FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [45] FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
 [56]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
 [67]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
 [78]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
 [89] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[100]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE
[111]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[122]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
[133]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[144]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[155]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
[166]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
[177]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[188]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
[199] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[210]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[221]  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE
[232]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE
[243]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[254]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[265] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
[276]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
[287]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
[298]  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
[309]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[320]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[331]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
[342]  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
[353] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[364] FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
[375]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
[386]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
[397]  TRUE  TRUE  TRUE FALSE
```

-
- Training dataset



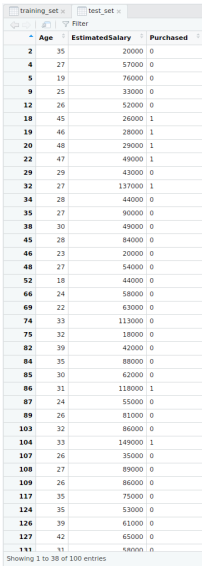
training_set

Filter

	Age	EstimatedSalary	Purchased
1	19	19000	0
3	26	43000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
10	35	65000	0
11	26	80000	0
13	20	86000	0
14	32	18000	0
15	18	82000	0
16	29	80000	0
17	47	25000	1
21	45	22000	1
23	48	41000	1
24	45	22000	1
25	46	23000	1
26	47	20000	1
27	49	28000	1
28	47	30000	1
30	31	18000	0
31	31	74000	0
33	21	16000	0
36	35	27000	0
37	33	28000	0
39	26	72000	0
40	27	31000	0
41	27	17000	0
42	33	51000	0
43	35	108000	0
44	30	15000	0
47	25	79000	0
49	30	135000	1
50	31	89000	0
51	24	32000	0
53	29	83000	0
54	35	23000	0
55	27	58000	0
56	24	55000	0

Showing 1 to 38 of 300 entries

-
- Test dataset



training_set test_set

Filter

	Age	EstimatedSalary	Purchased
2	35	20000	0
4	27	57000	0
5	19	76000	0
9	25	33000	0
12	26	52000	0
18	45	26000	1
19	46	28000	1
20	48	29000	1
22	47	49000	1
29	29	43000	0
32	27	137000	1
34	28	44000	0
35	27	90000	0
38	30	49000	0
45	28	84000	0
46	23	20000	0
48	27	54000	0
52	18	44000	0
66	24	58000	0
69	22	63000	0
74	33	113000	0
75	32	18000	0
82	39	42000	0
84	35	88000	0
85	30	62000	0
86	31	118000	1
87	24	55000	0
89	26	81000	0
103	32	86000	0
104	33	149000	1
107	26	35000	0
108	27	89000	0
109	26	86000	0
117	35	75000	0
124	35	53000	0
126	39	61000	0
127	42	65000	0
131	31	58000	0

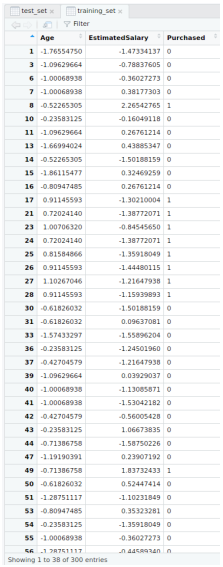
Showing 1 to 38 of 100 entries

-
- **Feature Scaling**

R

```
# Feature Scaling
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])
```

- **Output:**
  - Feature scaled training dataset



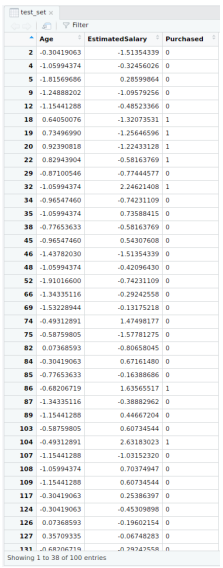
test_set × training_set ×

Filter

	Age	EstimatedSalary	Purchased
1	-1.76554750	-1.47334137	0
3	-1.09629664	-0.78837605	0
6	-1.00068938	-0.36027273	0
7	-1.00068938	0.38177303	0
8	-0.52265305	2.26542765	1
10	-0.23583125	-0.16049118	0
11	-1.09629664	0.26761214	0
13	-1.66994024	0.43885347	0
14	-0.52265305	-1.50188159	0
15	-1.86115477	0.32469259	0
16	-0.80947485	0.26761214	0
17	0.91145593	-1.30210004	1
21	0.72024140	-1.38772071	1
23	1.00706320	-0.84545650	1
24	0.72024140	-1.38772071	1
25	0.81584866	-1.35918049	1
26	0.91145593	-1.44480115	1
27	1.10267046	-1.21647938	1
28	0.91145593	-1.15939893	1
30	-0.61826032	-1.50188159	0
31	-0.61826032	0.09637081	0
33	-1.57433297	-1.55896204	0
36	-0.23583125	-1.24501960	0
37	-0.42704579	-1.21647938	0
39	-1.09629664	0.03929037	0
40	-1.00068938	-1.13085871	0
41	-1.00068938	-1.53042182	0
42	-0.42704579	-0.56005428	0
43	-0.23583125	1.06673835	0
44	-0.71386758	-1.58750226	0
47	-1.19190391	0.23907192	0
49	-0.71386758	1.83732433	1
50	-0.61826032	0.52447414	0
51	-1.28751117	-1.10231849	0
53	-0.80947485	0.35323281	0
54	-0.23583125	-1.35918049	0
55	-1.00068938	-0.36027273	0
56	-1.28751117	-0.44589340	0

Showing 1 to 38 of 300 entries

-
- Feature scaled test dataset



test_set ×

Filter

	Age	EstimatedSalary	Purchased
2	-0.30419063	-1.51354339	0
4	-1.05994374	-0.32456026	0
5	-1.81569686	0.28599864	0
9	-1.24888202	-1.09579256	0
12	-1.15441288	-0.48523366	0
18	0.64050076	-1.32073531	1
19	0.73496990	-1.25646596	1
20	0.92390818	-1.22433128	1
22	0.82943904	-0.58163769	1
29	-0.87100546	-0.77444577	0
32	-1.05994374	2.24621408	1
34	-0.96547460	-0.74231109	0
35	-1.05994374	0.73588415	0
38	-0.77653633	-0.58163769	0
45	-0.96547460	0.54307608	0
46	-1.43782030	-1.51354339	0
48	-1.05994374	-0.42096430	0
52	-1.91016600	-0.74231109	0
66	-1.34335116	-0.29242558	0
69	-1.53228944	-0.13175218	0
74	-0.49312891	1.47498177	0
75	-0.58759805	-1.57781275	0
82	0.07368593	-0.80658045	0
84	-0.30419063	0.67161480	0
85	-0.77653633	-0.16388686	0
86	-0.68206719	1.63565517	1
87	-1.34335116	-0.38882962	0
89	-1.15441288	0.44667204	0
103	-0.58759805	0.60734544	0
104	-0.49312891	2.63183023	1
107	-1.15441288	-1.03152320	0
108	-1.05994374	0.70374947	0
109	-1.15441288	0.60734544	0
117	-0.30419063	0.25386397	0
124	-0.30419063	-0.45309898	0
126	0.07368593	-0.19602154	0
127	0.35709335	-0.06748283	0
131	-0.68206719	-0.29242558	0

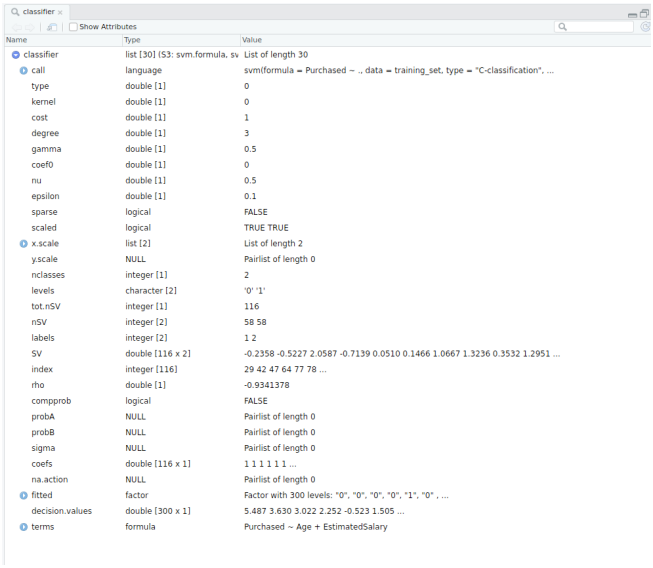
Showing 1 to 38 of 100 entries

-
- **Fitting SVM to the training set**

R

```
# Fitting SVM to the Training set
install.packages('e1071')
library(e1071)

classifier = svm(formula = Purchased ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')
```

- **Output:**
  - Classifier detailed



Name	Type	Value
classifier	list [30] (S3: svm.formula, sv	List of length 30
call	language	svm(formula = Purchased ~ ., data = training_set, type = "C-classification", ...
type	double [1]	0
kernel	double [1]	0
cost	double [1]	1
degree	double [1]	3
gamma	double [1]	0.5
coef0	double [1]	0
nu	double [1]	0.5
epsilon	double [1]	0.1
sparse	logical	FALSE
scaled	logical	TRUE TRUE
x.scale	list [2]	List of length 2
y.scale	NULL	Pairlist of length 0
nclasses	integer [1]	2
levels	character [2]	'0' '1'
tot.nSV	integer [1]	116
nSV	integer [2]	58 58
labels	integer [2]	1 2
SV	double [116 x 2]	-0.2358 -0.5227 2.0587 -0.7139 0.0510 0.1466 1.0667 1.3236 0.3532 1.2951 ...
index	integer [116]	29 42 47 64 77 78 ...
rho	double [1]	-0.9341378
compprob	logical	FALSE
probA	NULL	Pairlist of length 0
probB	NULL	Pairlist of length 0
sigma	NULL	Pairlist of length 0
coefs	double [116 x 1]	1 1 1 1 1 1 ...
na.action	NULL	Pairlist of length 0
fitted	factor	Factor with 300 levels: "0", "0", "0", "0", "1", "0" , ...
decision.values	double [300 x 1]	5.487 3.630 3.022 2.252 -0.523 1.505 ...
terms	formula	Purchased ~ Age + EstimatedSalary

- Classifier in nutshell

```
> classifier

Call:
svm(formula = Purchased ~ ., data = training_set, type = "C-classification",
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.5

Number of Support Vectors:  116
```

- **Predicting the test set result**

R

```
# Predicting the Test set results
```

```
y_pred = predict(classifier, newdata = test_set[-3])
```

- **Output:**

```
> y_pred
  2   4   5   9  12  18  19  20  22  29  32  34  35  38  45  46  48
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 52  66  69  74  75  82  84  85  86  87  89 103 104 107 108 109 117
  0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
124 126 127 131 134 139 148 154 156 159 162 163 170 175 176 193 199
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
200 208 213 224 226 228 229 230 234 236 237 239 241 255 264 265 266
  0   1   1   1   0   1   0   1   1   1   0   1   1   1   0   1   1
273 274 281 286 292 299 302 305 307 310 316 324 326 332 339 341 343
  1   1   1   0   1   1   1   0   1   0   0   0   0   1   0   1   0
347 353 363 364 367 368 369 372 373 380 383 389 392 395 400
  1   1   0   1   1   1   0   1   0   1   1   0   0   0   0
Levels: 0 1
> 
```

-

- **Making Confusion Matrix**

R

```
# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
```

- **Output:**

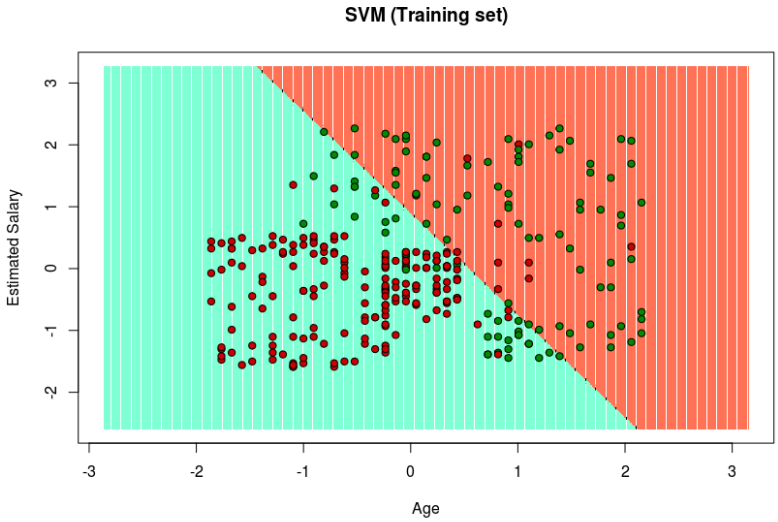
```
> cm
   y_pred
     0  1
  0 57  7
  1 13 23
```

-
- **Visualizing the Training set results**

R

```
# installing library ElemStatLearn
library(ElemStatLearn)

# Plotting the training data set results
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set)

plot(set[, -3],
     main = 'SVM (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)

points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'coral1', 'aquamarine'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```

- **Output:**

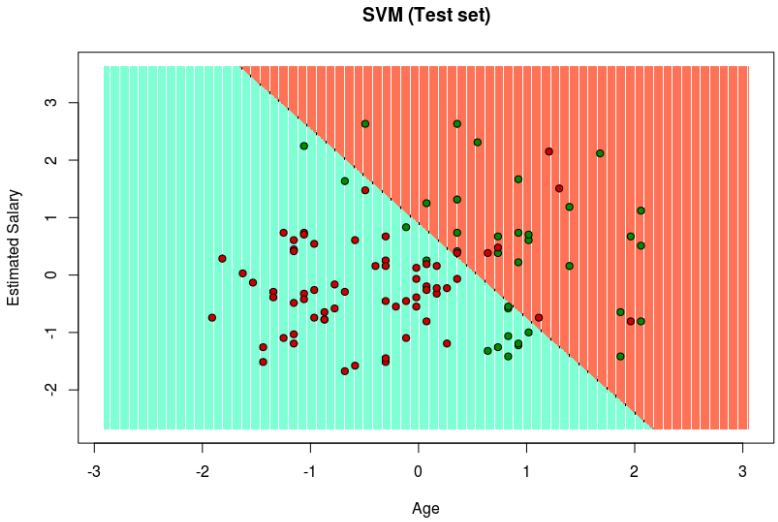


-
- **Visualizing the Test set results**

R

```
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)

grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(classifier, newdata = grid_set)

plot(set[, -3], main = 'SVM (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)

points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'coral1', 'aquamarine'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```

- **Output:**



Since in the result, a hyper-plane has been found in the Training set result and verified to be the best one in the Test set result. Hence, SVM has been successfully implemented in R.

3

Related Articles

1. Classifying data using Support Vector Machines(SVMs) in Python
2. Predicting Stock Price Direction using Support Vector Machines
3. Introduction to Support Vector Machines (SVM)
4. ML | Classifying Data using an Auto-encoder
5. Classifying Clothing Images in Python
6. Image classification using Support Vector Machine (SVM) in Python
7. Train a Support Vector Machine to recognize facial features in C++
8. Major Kernel Functions in Support Vector Machine (SVM)
9. Differentiate between Support Vector Machine and Logistic Regression